AFIT/DS/ENG/03-01

EXPLICIT BUILDING-BLOCK MULTIOBJECTIVE GENETIC ALGORITHMS: THEORY, ANALYSIS, AND DEVELOPMENT

DISSERTATION

Jesse B. Zydallis, Captain, USAF

AFIT/DS/ENG/03-01

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

Approved for public release; distribution unlimited.

The views expressed in this dissertation are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

AFIT/DS/ENG/03-01

EXPLICIT BUILDING-BLOCK MULTIOBJECTIVE GENETIC ALGORITHMS: THEORY, ANALYSIS, AND DEVELOPMENT

DISSERTATION

Presented to the Faculty of the

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

Jesse B. Zydallis, B.S., M.S.

Captain, USAF

March, 2003

Approved for public release; distribution unlimited.

AFIT/DS/ENG/03-01

EXPLICIT BUILDING-BLOCK MULTIOBJECTIVE
GENETIC ALGORITHMS:
THEORY, ANALYSIS, AND DEVELOPMENT

Jesse B. Zydallis, B.S., M.S.
Captain, USAF

Approved:

Dr. Gary B. Lamont Dissertation Advisor	Date
Dr. James T. Moore Committee Member	Date
Capt Antony J. Pohl Committee Member	Date
Dr. Laurence D. Merkle Committee Member	Date
Dr. (Maj) David A. Van Veldhuizen Committee Member	Date
Dr. Richard F. Deckro Dean's Representative	Date

Accepted:

Robert A. Calico, Jr.
Dean, Graduate School of Engineering and Management

Acknowledgements

Completing the requirements for a Doctor of Philosophy degree is one of the hardest tasks that I have ever attempted. Throughout this journey, my accomplishments, failures, hard work, TDYs to conferences and the birth of my daughter have all added to make me a better person. It is in these pages that I would like to thank all of the people who have aided and supported me in my quest for knowledge. Without the support of the people mentioned here and many others I would not have made it to where I am today. Many people have helped me over the last few years including fellow classmates, instructors, past supervisors, co-workers, friends, and family. While there are too many of you to mention by name, I do want to mention a few people in no specific order who have especially had an effect on me.

First I would like to thank my fellow MATH 600 classmates, in particular Kevin Anchor and Tony Kadrovach. What a trying time it getting through that class but we all survived. Kevin, you constantly shot holes into my attempts to solve problems and through this I have developed a much better approach to problem solving. Our late night discussions about computers and the best internet deals wrapped around discussions and arguments over the various MATH and STAT problems assigned from the classes we took together definitely challenged me and kept me sane over the past three years. I also enjoyed the many conversations we had in the LIZZARD lab about the classes we took and world events. Tony, without your help in MATH 600, the Digital Communications classes we took together, and in learning C I would still be sitting at a SUN workstation trying to figure out why my printf statements don't work without a semicolon at the end of the line. I thoroughly enjoyed our 0630 morning discussions about cusps and swarms or improvements to our houses and how hot or cold it was in our office.

I would also like to thank a number of colleagues and professors from various universities and industry including David Van Veldhuizen, Joshua Knowles, Thomas Reid, Greg Gunsch, Carlos Coello Coello, Eckart Zitzler, Marco Laumanns, David Corne, Kalyanmoy Deb, and David Goldberg for the intellectual conversations that we had. You have all aided me in one way or another and I appreciate your help.

I would like to thank my fellow MS and PhD students who sweated with me through the excruciating heat of the past summers and the chilling cold of the winters in the Bldg 640 Penthouse. You have all helped me with either Matlab, C and MPI programming or just the occasional meaningless conversation that we all need to maintain our sanity, specifically I would like to thank Todd Hale, Scott Sallberg, Richard Day, Eddie Ochoa, Ken Pascoe, Chuck Ormsby, and Steve Michaud. I would also like to thank my friends, especially Gerard DeBlois, Joe Pado, and Robert Chankalian who continually asked me when I would be finished which helped to motivate me to finish the PhD as soon as possible.

I would like to thank all of my dissertation committee members and the Dean's representative for their useful comments, suggestions, and help in producing a high quality research document. To my research advisor, Dr. Lamont, without your guidance, work ethic, motivation, and support I would not be at this juncture in life. Your support in submitting papers and attending many conferences has helped me gain the insight and knowledge necessary to complete this research effort. I value your opinions and thank you for expanding my horizons. I'm sure our paths will cross again. I would also like to thank Colonel Don Kitchen. Sir, your insight and career guidance has been invaluable to me. You have motivated and guided me throughout my time at AFIT in order to help me become a better Air Force officer. Thank you. I would like to extend my thanks to the support personnel and secretaries at AFIT who have all helped me at some point in time, especially MJ McCormick, and Lavonne Allen.

I would also like to thank God and my family, especially my parents and grandparents. Without their great support and example I could not have made it to where I am today. Finally and most importantly I would like to thank my wife. Without her support, understanding, and motivation to never give up I could not have completed the PhD. Over the last three years you raised our daughter, kept our family together, and motivated me to complete this challenge while I remained locked in a room somewhere studying or typing on the computer. To my wife and daughter who helped me realize what is truly important in life, thank you.

Jesse B. Zydallis

Table of Contents

Page

Acknowledgements iii

List of Figures x

List of Tables xiv

List of Abbreviations xvi

Abstract xviii

I. Introduction and Overview 1-1

1.1 MOEA Overview 1-2

1.2 Research Goals and Specific Objectives 1-4

1.3 Research Approach 1-9

1.4 Document Organization 1-10

II. Multiobjective Evolutionary Algorithm Introduction 2-1

2.1 Evolutionary Algorithm Overview 2-1

2.2 Building Block Hypothesis 2-10

2.3 Multiobjective Evolutionary Algorithm Overview 2-13

2.4 Other Approaches to Solving MOPs 2-15

2.5 MOP Domain Formalization 2-17

2.5.1 Pareto Terminology 2-18

2.5.2 MOP Formulation 2-19

2.6 MOEA Algorithm Domain Formulation 2-25

2.7 MOEA Archiving 2-27

2.8 Constraint Handling in MOEAs 2-28

2.9 MOEA Ranking 2-36

Page

2.10 MOEA Niching, Fitness Sharing, and Mating Restrictions . . 2-37

2.11 MOEA Theory 2-40

2.12 Summary 2-43

III. Contemporary MOEA Development 3-1

3.1 Multiobjective Evolutionary Algorithms 3-2

3.1.1 Implicit Building Block Manipulating MOEAs . . . 3-5

3.1.2 Explicit Building Block Manipulating MOEAs . . . 3-20

3.1.3 MOEA Summary 3-27

3.2 MOEA Metrics 3-28

3.2.1 Chi-Square-Deviation Metric 3-31

3.2.2 Conservative Distance Convergence Metric 3-32

3.2.3 Error Ratio Metric 3-33

3.2.4 Relative Coverage Metric 3-33

3.2.5 Maximum Pareto Front Error Metric 3-34

3.2.6 Average Pareto Front Error Metric 3-35

3.2.7 Hyperarea Metric 3-36

3.2.8 Generational Distance Metric 3-37

3.2.9 7-Point Average Distance Metric 3-38

3.2.10 Spacing (Range) Metric 3-39

3.2.11 Overall Nondominated Vector Generation and Ratio Metric 3-40

3.2.12 Progress Measure Metric 3-42

3.2.13 Generational Nondominated Vector Generation Metric 3-42

3.2.14 Nondominated Vector Addition Metric 3-43

3.2.15 $D1_R$ 3-43

3.2.16 R1 and $R1_R$ 3-44

3.2.17 R2 and $R2_R$ 3-44

Page

3.2.18 R3 and $R3_R$ 3-45

3.2.19 Visualization 3-45

3.3 MOMGA-II Performance Metrics 3-46

3.4 Summary 3-48

IV. MOEA MOP Test Suites 4-1

4.1 Unconstrained Test Suite MOPs 4-2

4.2 Constrained Test Suite MOPs 4-6

4.3 Deceptive MOPs 4-11

4.4 Discrete MOPs 4-14

4.4.1 Modified Multiobjective Knapsack Problem 4-15

4.5 Real World MOPs 4-19

4.5.1 Advanced Logistics Problem (ALP) 4-20

4.6 Summary 4-26

V. MOEA Symbolic Formulations 5-1

5.1 Multiobjective Evolutionary Algorithm Operators 5-1

5.2 Subpopulation Based Approaches 5-4

5.3 Ranking 5-6

5.4 Niching Based Approaches 5-6

5.5 Mating Restrictions and Comparison Sets 5-8

5.6 Multiobjective Evolutionary Algorithm Formulation 5-9

5.7 Summary 5-10

VI. MOMGA-II Development 6-1

6.1 Introduction 6-1

6.2 Background of the MOMGA 6-2

6.3 MOMGA-II - Development 6-6

6.4 Design of Experiments 6-12

Page

6.5 MOMGA-II Results and Statistical Analysis 6-15

6.5.1 Standard MOP Test Suite Experimental Results . . 6-16

6.5.2 Constrained MOP Test Suite Experimental Results . 6-42

6.5.3 Real-World MOP Test Suite Experimental Results . 6-46

6.6 MOEA BB Testing Based on Empirical Data 6-75

6.7 Summary 6-84

VII. Parallel MOEA Development 7-1

7.1 Introduction 7-1

7.2 Fundamental Background 7-3

7.2.1 pMOEA Notation 7-3

7.2.2 pMOEA Motivation and Paradigms 7-6

7.3 pMOEA Analyses and Issues 7-25

7.3.1 pMOEA Observations 7-25

7.3.2 pMOEA Suitability Issues 7-31

7.3.3 pMOEA Hardware and Software Architecture Issues 7-32

7.3.4 pMOEA Test Function Issues 7-33

7.3.5 pMOEA Metric/Parameter Issues 7-35

7.4 pMOEA Development 7-38

7.4.1 pMOEA Implementation Issues 7-41

7.4.2 Parallel MOEA Theory 7-56

7.5 A “Generic” pMOEA 7-56

7.5.1 Engineering the pMOMGA-II 7-58

7.5.2 “Genericizing” a pMOEA 7-60

7.6 pMOMGA-II Testing 7-61

7.6.1 Experimental Design 7-61

7.6.2 Experimental Results and Analysis 7-63

7.7 Summary 7-69

Page

VIII. MOEA Population Sizing 8-1

8.1 Single Objective Population Sizing 8-1

8.2 Multiobjective Population Sizing 8-7

8.3 MOEA Population Sizing Reality 8-18

8.4 Summary 8-21

IX. Conclusions and Documentation 9-1

9.1 Introduction 9-1

9.2 Contributions 9-2

9.2.1 Research Goal 9-2

9.2.2 Symbolic Formulations 9-3

9.2.3 Explicit BB-Based MOEA Development 9-3

9.2.4 BB Insight 9-5

9.2.5 Parallel MOEAs 9-6

9.2.6 MOEA Population Sizing 9-7

9.3 Future MOEAs and their Operators 9-8

9.4 Publications 9-12

Appendix A. EA Mathematical Formulation Background A-1

A.1 Evolutionary Operators A-1

A.2 Evolutionary Algorithm Notation and Formulations A-5

A.3 Generic MOEA A-8

Appendix B. Additional MOP Details B-1

B.1 Unconstrained MOPs B-1

B.2 Constrained MOPs B-3

B.3 Test Function Generator B-4

Bibliography BIB-1

Vita VITA-1

List of Figures

Figure		Page
2.1.	Evolutionary Algorithm Pseudocode [74]	2-6
2.2.	Cut and Splice	2-8
3.1.	Generic MOEA Pseudocode [31]	3-4
3.2.	Example Minimization MOP PF_{true} and PF_{known}	3-38
4.1.	MOP-CT (Tanaka), $a = .1, b = 16$, Original PF_{true} (P_{true}) regions	4-9
4.2.	MOP-CT (Tanaka), $a = .1, b = 32$, PF_{true} (P_{true}) regions	4-9
4.3.	MOP-CT (Tanaka), $a = .1, b = 16$, PF_{true} (P_{true}) regions	4-9
4.4.	MOP-CT (Tanaka), $a = .15, b = 32$, PF_{true} (P_{true}) regions	4-9
4.5.	MOP-CT (Tanaka), $a = .1(x^2 + y^2 + 5xy), b = 32$, PF_{true} (P_{true}) periodic regions	4-10
4.6.	MOP-CT (Tanaka), $a = .1(x^2+y^2+5xy), b = 8(x^2+y^2)$, PF_{true} (P_{true})	4-10
4.7.	Pedagogical Deceptive MOP	4-13
5.1.	Multiobjective Evolutionary Algorithm Outline	5-10
6.1.	MOMGA Pseudocode	6-4
6.2.	MOMGA-II Pseudocode	6-7
6.3.	MOP1 PF_{known} Comparison	6-20
6.4.	MOP1 NSGA-II and PAES Results [46]	6-21
6.5.	MOP2 PF_{known} Comparison	6-22
6.6.	MOP3 PF_{known} Comparison	6-23
6.7.	MOP4 PF_{known} Comparison	6-24
6.8.	MOP4 NSGA-II and SPEA Results [46]	6-25
6.9.	MOP6 PF_{known} Comparison	6-26
6.10.	Overall Generational Distance Performance	6-29

Figure Page

6.11. Overall ONVG Performance 6-30

6.12. Overall Spacing Performance 6-31

6.13. MOMGA MOP1 Metrics 6-37

6.14. MOMGA MOP2 Metrics 6-38

6.15. MOMGA MOP3 Metrics 6-39

6.16. MOMGA MOP4 Metrics 6-40

6.17. MOMGA MOP6 Metrics 6-41

6.18. MOP-CT, $a = .1, b = 16$, PF_{true} vs PF_{known} 6-44

6.19. MOP-CT, $a = .1, b = 32$, PF_{true} vs PF_{known} 6-44

6.20. MOP-CT, $a = .1(x^2 + y^2 + 5xy), b = 32$,$PF_{true}$ vs PF_{known} 6-44

6.21. MOP-CT, $a = .1(x^2 + y^2 + 5xy), b = 8(x^2 + y^2)$, PF_{true} vs PF_{known} 6-44

6.22. MOP-ALP Discrete 3D Integer Search Space 6-56

6.23. True Pareto Front Versus MOMGA-II Pareto Front 6-57

6.24. MOMGA-II Pareto Front - Expanded Problem 6-58

6.25. Initial Population for 100 Item 2 Knapsack MMOKP 6-64

6.26. Elitist 100 Item 2 Knapsack MMOKP 6-66

6.27. 100 Item 2 Knapsack MMOKP 6-67

6.28. 250 Item 2 Knapsack MMOKP 6-68

6.29. 500 Item 2 Knapsack MMOKP 6-69

6.30. 750 Item 2 Knapsack MMOKP 6-70

6.31. Ranked Pareto Fronts 6-76

6.32. BB Sizes and Their Contribution to the Pareto Fronts 6-77

6.33. Single Objective Values for Deceptive MOP 6-80

6.34. Pareto Optimal Solution Paths for Deceptive MOP 6-81

6.35. Pareto Front Paths for Deceptive MOP 6-82

6.36. MOP4 Pareto Front Generation 6-83

6.37. 100 Item MMOKP BB Analysis 6-85

Figure Page

7.1. Generic MOEA Pseudocode [31] 7-8

7.2. Master-Slave pMOEA Paradigm [31] 7-11

7.3. Island pMOEA Migration Paradigm [31] 7-15

7.4. Diffusion pMOEA Migration Paradigm [31] 7-18

7.5. Example Hierarchical Paradigm 7-20

7.6. (a) Dominated Region (Standard Definition) and (b) Dominated Region (New Definition) [55] 7-29

7.7. Generic Objective Function Processor Allocation ([55]) 7-29

7.8. Problem-Algorithm Domain Interaction [31, 194] 7-39

7.9. 7-42

7.10. Generic Master-Slave pMOEA Pseudocode 7-57

7.11. Generic Island pMOEA Pseudocode 7-57

7.12. Generic Diffusion pMOEA Pseudocode 7-58

7.13. pMOEA Evolution 7-60

7.14. Parallel MOP4 Processor PF_{known} Comparison 7-63

7.15. Parallel MOP4 PF_{known} Comparison 7-65

7.16. MMOKP PF_{known} Comparison 7-66

7.17. MMOKP PF_{known} Comparison 7-67

7.18. pMOMGA-II MOP4 Metrics 7-72

8.1. Distributions of the mean fitness values of two BBs, H_1 and H_2. . 8-3

8.2. Distributions of the mean fitness values of BBs, H_1 and H_2 with an increased number of samples. 8-6

8.3. Distributions of the mean fitness values of BBs, BB H_1 and H_2 for a two dimensional MOP 8-10

A.1. Population Transformation. A-2

A.2. Random Population Transformation. A-2

A.3. Evolutionary Operator. A-3

Figure Page

A.4. Evolutionary Algorithm Outline A-7

B.1. MOP-C5 connected PF_{true} regions [152] B-4

List of Tables

Table		Page
2.1	Types of Selection	2-7
2.2	Types of Crossover	2-7
2.3	MOEA Theory	2-42
3.1	MOEA Characteristics	3-23
4.1	MOEA Test Suite Functions	4-5
4.2	Constrained MOEA Test Suite Functions	4-6
4.3.	Possible Multiobjective NP-Complete Functions [31]	4-15
4.4.	Tasks	4-24
4.5.	Resource Levels	4-24
4.6.	Desired Task Capability Ratios	4-24
4.7.	Desired Capability Matrix	4-25
4.8.	Task Suitability / Lift Consumption Matrix	4-25
6.1.	Nonparametric Kruskal-Wallis Results for Standard MOP Test Suite	6-31
6.2.	Nonparametric Mann-Whitney Results for Standard MOP Test Suite	6-32
6.3.	Timing Comparison	6-34
6.4.	MOMGA-II Execution Time Improvement Over MOMGA	6-35
6.5.	Nonparametric Mann-Whitney Results for Standard MOP Test Suite Timing	6-35
6.6.	Constrained MOP	6-43
6.7.	MOMGA-II Parameter Values for MMOKP	6-59
6.8.	2 Knapsack MMOKP Results	6-68
6.9	Knapsack Problem Results	6-74
7.1.	Parallel Processing System Characteristics	7-9

Table Page

7.2. Key Parallel MOEA Characteristics 7-38

7.3. pMOEA Migration Schemes 7-44

7.4. pMOEA Replacement Schemes 7-48

7.5 Parallel MOP 4 Results 7-73

7.6 Parallel 100 Item, 2 Knapsack MMOKP Results 7-73

8.1. Estimate of MOEA Population Size for MOPs 1-6 ($l = 24$) 8-19

8.2. Estimate of MOEA Population Size for MMOKP ($l = 100$) 8-20

B.1 MOP-C5 P_{true} solution values, [152] B-3

B.2 Deb's MOEA Test Suite Functions B-5

List of Abbreviations

Abbreviation		Page
SO	Single Objective	1-1
MOPs	Multiobjective Optimization Problems	1-1
MOEAs	Multiobjective Evolutionary Algorithms	1-2
EA	Evolutionary Algorithm	1-2
BB	Building Block	1-3
EVOPs	Evolutionary Operators	1-5
pEA	Parallel Evolutionary Algorithms	1-7
pMOEA	Parallel Multiobjective Evolutionary Algorithms	1-7
OP	Optimization Problem	2-1
EA	Evolutionary Algorithm	2-4
ES	Evolutionary Strategies	2-4
EP	Evolutionary Programming	2-4
GP	Genetic Programming	2-4
GAs	Genetic Algorithms	2-4
SGA	Simple Genetic Algorithm	2-9
BBH	Building Block Hypothesis	2-10
NFL	No Free Lunch	2-14
OR	Operations Research	2-15
MCDM	Multi-Criteria Decision Making	2-16
DM	Decision Maker	2-17
VEGA	Vector Evaluated Genetic Algorithm	3-2
MOGA	Multiple Objective Genetic Algorithm	3-6
NPGA	Niched Pareto Genetic Algorithm	3-7
NSGA	Nondominated Sorting Genetic Algorithm	3-8
MMOSGA	Mendelian Multiobjective Simple Genetic Algorithm	3-9

Abbreviation		Page
NPGA2	Niched Pareto Genetic Algorithm 2	3-11
NSGA-II	Nondominated Sorting Genetic Algorithm II	3-11
$(1+1)$-PAES	Pareto Archived Evolution Strategy	3-12
MOO	Multiobjective Optimization	3-12
M-PAES	Memetic Pareto Archived Evolution Strategy	3-14
SPEA	Strength Pareto Evolutionary Algorithm	3-15
PESA	Pareto Envelope-Based Selection Algorithm	3-17
MOGLS	Multiple-Objective Genetic Local Search Algorithm	3-17
MOMGA	Multiobjective Messy Genetic Algorithm	3-21
mGA	Messy Genetic Algorithm	3-21
MOMGA-II	Multiobjective Messy Genetic Algorithm II	3-22
PFSCO	Pareto Front Set Combination Operator	5-3
pMOEA	Parallel Multiobjective Evolutionary Algorithm	7-1
pEA	Parallel Evolutionary Algorithm	7-2
CLT	Central Limit Theorem	8-4

AFIT/DS/ENG/03-01

Abstract

This dissertation research emphasizes explicit Building Block (BB) based MOEA performance and detailed symbolic representations. An explicit BB-based MOEA for solving constrained and real-world MOPs is developed, the Multiobjective Messy Genetic Algorithm II (MOMGA-II) to validate symbolic BB concepts. The MOMGA-II provides insight into solving difficult MOPs that is generally not realized through the use of implicit BB-based MOEA approaches. This insight is necessary to increase the effectiveness of all MOEA approaches.

Parallel MOEA (pMOEA) concepts are presented to potentially increase MOEA computational efficiency and effectiveness. Communications in a pMOEA implementation is extremely important, hence innovative migration and replacement schemes are detailed and tested. These parallel concepts support the development of the first explicit BB-based pMOEA, the pMOMGA-II. MOEA theory is also advanced through the derivation of the first MOEA population sizing theory. The sizing theory presented derives a conservative estimate of the MOEA population size necessary to achieve good results with a specified level of confidence. Validated results illustrate insight into building block phenomena, good efficiency, excellent effectiveness, and motivation for future research in the area of explicit BB-based MOEAs.

EXPLICIT BUILDING-BLOCK MULTIOBJECTIVE GENETIC ALGORITHMS: THEORY, ANALYSIS, AND DEVELOPMENT

I. Introduction and Overview

Optimization problems are encountered daily in each of our lives. While most of us may fail to recognize the structure of these problems, they exist at many levels of complexity. Such optimization problems can vary from relatively simple, single input variable, single objective (SO) problems (*e.g. how do I get to the grocery store in the least amount of time*) to multivariate, multiobjective optimization problems (MOPs) of great complexity (*e.g. advanced logistics planning problems concerning resource allocation*). While obtaining the optimal solution to an MOP and hence solving it is the ultimate goal of any attempt to optimize an MOP, the desire of most researchers is to find an acceptable solution to MOPs. Since many real world problems are MOPs, this investigation concentrates on finding acceptable solutions to MOPs using a relatively new, innovative, evolutionary search approach.

Generating acceptable solutions to MOPs is an area of increased interest by researchers in many disciplines, including Computer Engineering, Operations Research, and Computer Science to name a few. As the computational power of personal computers and supercomputers continues to increase, with processor speeds just about doubling every couple of years according to Moore's Law, problems that were previously too computationally complex to tackle become more manageable. [1] The increased computational capability of today's computers encourages researchers to attempt to solve problems that previously were untractable and use increased levels of precision. These facts contribute to an increasing interest in the solving of MOPs, as they typically provide a more realistic formulation

[1] Moore's Law is the common name for Gordon E. Moore's observation in 1965 that there shall continue an exponential growth in the number of transistors able to be placed on an integrated circuit at least every two years [147].

of the problem. Increasingly, evolutionary algorithms based upon biological models are used to accomplish this task, as seen in numerous conference proceedings and published papers [26, 62, 63, 109, 120, 175, 202] [2].

Multiobjective Evolutionary Algorithms (MOEAs) are contemporary algorithms to solve MOPs and are part of the 'soft computing' umbrella of search algorithms. This includes Genetic Algorithms, Evolution Strategies, Evolutionary Programming, and Genetic Programming and their extension to MOEA implementations. As more researchers use MOEAs, many analyze the variety of such algorithms in an attempt to understand how and why they work. Even though SO Evolutionary Algorithms (EA) have existed for a few decades, the theoretical analysis of EAs is not complete, and hence, neither is the theoretical analysis of MOEAs. These statements illustrate the need for additional development, and theoretical analyses of MOEAs.

1.1 MOEA Overview

MOEAs are relatively new algorithms, but over the past few years there has been an explosion with respect to the interest in MOEAs and the number of publications generated each year [26, 62, 63, 109, 120, 175, 202]. To further illustrate this fact, in the year 2001, the first International Conference on Multi-Criterion Optimization using MOEAs was held in Zurich, Switzerland. This was an entire conference dedicated to MOEAs; theory, design, analysis, and applications. Also there have been many Multi-Criterion conferences in other pedagogical communities, including Operations Research, that employ a variety of other MOP solution techniques [96]. These points illustrate the intense interest in MOEAs for solving academic and real-world MOPs.

Since the MOEA field is no longer in its infancy, the use of MOEAs is accepted among researchers as being able to produce efficient and effective results for various real-world problems. Someone might question *Are there any contributions that remain to be made?* The answer to this question is YES! There are always improvements and theoretical

[2] Dr. Carlos Coello Coello maintains a database of MOEA publications at the web address http://www.lania.mx/ ccoello/EMOO/EMOObib.html. Many researchers in the EA and MOEA fields recognize this database as one of the most complete listings of publications in the MOEA field. Currently this database is updated multiple times a year and contains well over 1000 MOEA citations.

developments that can be made. Contributions can come in the form of new MOEA operators, theoretical developments and more. Additionally, many researchers are delving into different aspects of MOEAs in pursuit of increasing MOEA efficiency and effectiveness. Efficiency is defined as a measure of resource use, such as memory requirements, CPU utilization, network bandwidth, and file storage as well as the total wall clock time required for an MOEA to execute. Effectiveness is defined as a measure of the quality of the solutions that are generated by the MOEA.

MOEA researchers are categorized into two groups, the researchers interested in understanding the inner workings, operators, development, and theory of MOEAs and the researchers interested in solely applying MOEAs to generate MOP solutions. The major emphasis of this research effort places it into the former category. An understanding of the inner workings of MOEAs is presented and this understanding is capitalized on in order to advance the state-of-the-art with respect to MOEAs.

Many researchers in the MOEA field have recognized the need for additional theoretical contributions [31, 44, 85, 167, 184]. Researchers have accepted MOEAs as valid search algorithms but additional theoretical contributions can still be made. Theoretical contributions in conjunction with analyses of different MOEAs leads to a deeper understanding of the reasons why certain MOEAs perform the same or better than others. This research makes substantial theoretical and practical contributions to the field of MOEAs. This theoretical contribution is made through addressing the development of a population sizing equation for MOEAs.

Specifically, the analysis of explicit Building Block (BB) based MOEAs is conducted to illustrate the advantages and disadvantages of this rarely used approach. Additionally, there is a valid need for guidance as to setting MOEA parameter values and understanding MOEA operators. Symbolic formulations of MOEA operators and an analysis of the effect of explicit BBs on overall solution quality is presented. Additionally, a clear procedure for the use of parallel concepts in MOEAs is presented to aid MOEA researchers in using this relatively new resource to the MOEA field. As a whole, this research effort presents practical methods that may increase the efficiency and effectiveness of MOEAs.

Various MOPs are presented to attempt and solve via the use of the MOEA approach presented. These problems include high dimensional, real-world applications that EAs, and by extension, MOEAs are suited to solve. Many MOEA publications address attempts to solve small pedagogical examples but larger problems that are considered more realistic of real-world applications are typically missing [29, 46, 102, 107, 130, 191]. While pedagogical examples are useful for evaluating the performance of MOEAs on MOPs with known solutions, they are not truly indicative of the anticipated performance of MOEAs as applied to real-world or high dimensionality application MOPs. Real-world problems may be of various classes including those of high dimensionality in both the genotype or input variables and the phenotype or number of fitness functions. Many times real-world application MOPs are constrained and hence may be more difficult for an MOEA to solve. The Pareto fronts (phenotype solution sets) have varying characteristics and those of highest interest include - disconnected, connected, continuous, discrete, concave, and convex characteristics. Specifically, a real-world Air Force application, the Advanced Logistics problem, is analyzed and used to test the performance of the MOEA of interest. Research into solving real-world MOPs through the use of MOEAs is of direct benefit to the Air Force and the MOEA community.

1.2 Research Goals and Specific Objectives

The primary goal of this research is to advance the state-of-the-art with respect to explicit BB-based MOEAs. The research goal is accomplished through MOEA development and analysis. BBs are considered by many to be an important aspect of any evolutionary approach and are a focus of this effort. Limited research into BB-based evolutionary approaches [184, 187, 191, 192] has shown promising results, and hence, the focus of this effort is on extending MOEA BB concepts as reflected in the following research goal:

Research Goal: Advance the state-of-the-art with respect to explicit Building Block Based MOEAs.

The research presented in this dissertation advances the theory, design, and implementation of MOEAs as well as the analysis of MOPs. Contributions are made in the

MOEA research area of explicit manipulation of BBs, which has not been addressed in detail in the current MOEA literature. This effort has five objectives and the contributions of this effort arise out of meeting each of the five objectives. The objectives are presented in the order in which they are addressed in this dissertation and not in terms of importance.

The first objective is to present a clear symbolic formulation of MOEA operators and the algorithm. Such a formulation can aid in understanding MOEA operators and provide researchers a clear definition to use when implementing these operators. A good understanding of MOEA operators aids researchers in correctly implementing MOEA operators and in developing new MOEAs and operators with improved performance over existing implementations.

Objective 1: Present a symbolic formulation of MOEA operators and algorithm details.

A detailed summary of many contemporary MOEAs is presented to illustrate the difference in MOEAs and summarize the contributions that have been made in the field. This is useful for the design of new MOEAs or a validation of current MOEAs. This effort categorizes MOEAs as either explicit BB-based or implicit BB-based MOEAs. Explicit BB-based MOEAs analyze the population of individuals to explicitly identify the BBs or partial strings that lead to the generation of good solutions. Explicit BB-based MOEAs use evolutionary operators to capitalize on the identification of good BBs and subsequently manipulate those good BBs in order to generate good solutions. The term implicit BB-based MOEA defines an MOEA that does not explicitly identify the "good" BBs, (the ones and zeros) or partial strings, in the population. Implicit BB-based MOEAs assume that the good BBs are present in the population and through evolutionary operators these BBs are implicitly manipulated in a necessary fashion to obtain good results.

Previous to the research completed in this effort, all of the existing MOEAs in the community were implicit BB-based MOEAs with the exception of the MOMGA [184, 190]. The MOMGA explicitly finds the "best" BBs in the population and then exploits these BBs through the use of evolutionary operators (EVOPs) to generate "better" solutions. This research effort makes contributions to the MOEA community through presenting a

thorough analysis of the effect of the explicit manipulation of BBs on solution quality in MOEAs. Since most other researchers only implicitly address BBs, the potential exists for these researchers to overlook certain aspects of MOEA development and MOP solutions that are found through the use of explicit BB-based MOEAs.

In testing MOEAs, parallel MOEAs, and other associated BB concepts, different MOPs and metrics should be used in order to compare their performance [42, 43, 52, 53, 54, 113, 114, 184, 188, 189, 192, 194, 195, 208, 209, 212, 214]. A number of constrained MOPs are used to analyze the performance of the MOMGA-II. The constrained MOPs selected to evaluate the performance of the MOMGA-II contain characteristics that differ from MOPs contained in other test suites. MOP Test suites are only useful if the test suite MOPs reflect the characteristics and class of MOP that an MOEA is designed to attempt and solve. The MOMGA-II is used to attempt and solve constrained MOPs formulated with integer based decision variables. Considering that existing MOP test suites typically do not contain MOPs of these characteristics, a real-world and NP-Complete MOP are selected and proposed for test suite inclusion. Many real-world MOPs are formulated with constraints and integer based decision variables and hence optimizing an MOEA for this class of problems.

The second objective is to develop an explicit BB-based MOEA for solving real-world applications including real-world Air Force MOPs and constrained MOPs with integer based decision variables. The literature shows that an explicit BB-based MOEA can achieve the performance that matches or exceeds that of implicit BB-based MOEAs in terms of effectiveness, but the efficiency of the explicit BB-based MOEAs degrades rapidly with increased problem sizes [184]. The degradation of the efficiency and execution time of explicit BB-based MOEAs as problem sizes increase makes these MOEAs impractical to apply to MOPs of high dimensionality and time consuming fitness evaluations, the characteristics of many real-world MOPs. Hence an explicit BB-based MOEA, the MOMGA-II, is another advancement in illustrating the usefulness of BBs in MOEAs. The MOMGA-II is developed to use explicit BB concepts to attempt and solve real-world application MOPs in much less time than previous explicit BB-based approaches. The performance of this

new MOEA is analyzed through the use of various MOPs and metrics to present a limited statistical comparison to other MOEA approaches.

Objective 2: Develop an explicit BB-based MOEA for solving constrained and real-world MOPs.

The third objective is to demonstrate that explicit BB-based MOEAs may provide insight into MOPs that otherwise is not found with implicit BB approaches. The effect of attempting to solve MOPs with different BB sizes is analyzed. Related to this objective is analyzing BBs applied to various MOPs to determine if a trend exists, i.e., *Do "hard" MOPs typically require larger BB sizes to find the optimal solution set?* In the context of this effort, a hard MOP is one in which an MOEA has difficulty in attempting to find good potential solutions to the MOP. The use of an explicit BB-based MOEA allows one to conduct an analysis of the relationship between BB sizes and the difficulty of MOPs.

Objective 3: Demonstrate that explicit BB-based MOEAs may provide insight into solving difficult MOPs.

Parallelization of MOEAs is a relatively new concept. Parallel evolutionary algorithms (pEA) have existed over three decades. Due to the relatively recent development of MOEAs, the parallel MOEA (pMOEA) field is a new field within the MOEA area. There exists room for meaningful contributions to the MOEA community through the analysis of parallel concepts as applied to MOEAs. The fourth objective is to describe parallel concepts for MOEAs, present innovative migration and replacement schemes for use in parallel MOEA paradigms, and develop the first explicit BB-based parallel MOEA. A description of pMOEA concepts provides insight into pMOEAs and aids researchers in the development of pMOEAs. Migration and replacement schemes are used during the communication processes that occurs in a pMOEA and directly effect the overall pMOEA performance. Since the goals of parallelization are typically to improve the efficiency and/or effectiveness of a serial MOEA, it is important to develop effective migration and replacement schemes for use in pMOEAs.

The three major parallel MOEA paradigms, Master-Slave, Island, and Diffusion are concentrated on in the discussion of parallel MOEA concepts. [3] Implementation details in generating pMOEAs is important to identify and understand as the pMOEA field is new and largely undefined. Details of the implementation of a pMOEA are extremely important to MOEA researchers for use in determining how to design a new parallel MOEA, identifying issues that must be addressed in the design process, recognizing considerations that must be made when deciding whether to design a pMOEA from scratch or to parallelize an existing MOEA, and realizing the differences between pEAs and pMOEAs. pMOEA implementation details must be understood in order to develop efficient and effective pMOEAs. Additionally, the communication process that occurs within the parallel implementation is critical to achieving the desired pMOEA performance. As a discussion of migration and replacement schemes is missing from pMOEA publications, this research effort clearly identifies these schemes and discusses the advantages and disadvantages of each scheme. Considering that explicit BB-based MOEAs have been shown to achieve good performance, an effort is made to improve the performance of an explicit BB-based MOEA through the integration of parallel concepts.

Objective 4: Describe parallel concepts for MOEAs, present innovative migration and replacement schemes for use in parallel MOEA paradigms, and develop the first explicit BB-based parallel MOEA.

The fifth objective of this effort is to advance the theory of MOEAs, in particular, the theory of MOEA population sizing. A generic population sizing equation for MOEAs is developed. It is difficult to determine an initial population size to use in MOEAs that provides a level of confidence that an MOEA will generate good solutions. The population sizing theory presented in support of this objective determines with statistical confidence the initial population size to use in MOEAs. This is an important step in advancing the theory of MOEAs and their application to MOPs.

[3] It is noted that other parallel paradigms exist in support of other disciplines [119]. In the context of this research effort, the three major parallel EA and MOEA paradigms, Master-Slave, Island, and Diffusion, are discussed.

Objective 5: Enhance MOEA theory by deriving MOEA population sizing theory.

To summarize, the research focuses on finding "good" solutions to scientific and engineering MOPs. This effort uses MOEAs, pMOEA concepts, and population sizing theoretical developments in order to find "good" solutions to MOPs. The concepts presented are validated through the application of the MOMGA-II to various test suite MOPs and a statistical analysis of the results.

1.3 Research Approach

Accomplishing the goal and objectives of this research effort involves the use of good engineering principles. A haphazard approach is sure to fail considering the difficulty and complexity associated with each objective. In order to advance the state-of-the-art of MOEAs, one must become an expert in the areas of EAs and MOEAs.

An extensive analysis of the existing literature in the areas of EAs, MOEAs, and parallel processing techniques was completed to achieve this expertise. The knowledge and insight gained from this literature review was used to design and develop a new innovative MOEA, the MOMGA-II, for use by researchers in solving real-world MOP applications. A number of MOPs are identified and proposed for test suite inclusion and these real-world MOPs are used in testing and analyzing the performance of the MOMGA-II.

In order to discuss with any certainty the quality of the results obtained, various metrics are analyzed. A subset of the metrics presented are selected for analyzing MOEA performance. Hypothesis testing is used to analyze the results of the MOMGA-II and state with statistical certainty how the results compare to existing methods.

As large improvements over existing methods are always welcomed, it is recognized that even small improvements can result in large cost savings to the warfighter effort. The algorithmic results of this effort can be integrated with existing methods to attempt and find the solution to complex MOPs. The Air Force has a direct interest in this research as it is sponsored by the Information Technology Division of the Air Force Research Laboratory

at Wright-Patterson AFB in Ohio. This research presents concepts and new ideas that directly support the Air Force and the solving of real-world Air Force applications.

1.4 Document Organization

This document is organized as follows. The following chapter, Chapter II, presents necessary background and historical information in the form of key EA and MOEA definitions and terminology. Different optimization approaches are discussed in order to illustrate the rationale for the selection of an evolutionary approach over other approaches. A discussion of BBs follows as BBs are integral to this effort. Multiobjective optimization terminology and the algorithm domain formulation is presented to aid the reader's understanding of the basic concepts used. Finally a discussion of MOEA operators and theory completes the chapter.

Contemporary MOEA development as well as the metrics necessary to evaluate the performance of MOEAs follows in Chapter III. The results of an extensive literature review of MOEAs is presented and provides specific details of the design and functionality of some popular and highly referenced MOEAs. Most importantly, the MOEA metrics selected for use are discussed and compared to the other metrics presented.

Chapter IV discusses MOP test suites and the MOPs selected for testing the performance of the MOMGA-II. MOPs of various classes are presented in this chapter and some of these MOPs are proposed for use by other researchers. The proposed MOPs contain various characteristics that differ from MOPs contained in other MOP test suites, thereby illustrating their usefulness for MOEA testing. The following chapter, Chapter V, discusses symbolic formulations for MOEA operators and an MOEA. Chapter VI presents the overall design approach, testing, results, and analysis of the MOMGA-II, the MOEA chosen for use. The MOMGA-II is applied to unconstrained, constrained, NP-Complete, and real-world MOPs and the associated results and analyses are presented. Chapter VII discusses parallel concepts and their application to MOEAs, as well as the design and testing considerations necessary when applying parallel concepts to MOEAs. Chapter VIII presents the theoretical analysis conducted for developing an MOEA population sizing

equation to obtain “good” results given some specified level of confidence. The conclusions and recommendations for future work follow.

II. Multiobjective Evolutionary Algorithm Introduction

This chapter presents background information to aid the reader in understanding the necessary knowledge supporting this dissertation. A brief description of Evolutionary Algorithms (EAs) is presented followed by a discussion of Multiobjective Evolutionary Algorithms (MOEAs) and other approaches to solving Multiobjective Optimization Problems (MOPs). The Building Block Hypothesis (BBH), MOP and MOEA terminology and definitions are presented. Various classes of MOPs (unconstrained and constrained) are discussed in this chapter. Complex MOPs are a focus of this document and the terminology associated with these problems and different solution methods are described in some detail. An MOEA algorithm domain formulation is presented detailing a consistent terminology that is used throughout this effort. Since a number of MOPs that are the focus of this research effort are constrained MOPs, a discussion of constraint handling in MOEAs is presented. This chapter concludes with a table of MOEA theoretical areas that other researchers have addressed through various publications and identifies an area of theoretical development that is currently lacking in the literature. The theoretical area of MOEA population sizing is largely unaddressed in the current literature and is a focus of this research effort. The following sections present discussions of topics to provide an extensive foundation for the research.

2.1 Evolutionary Algorithm Overview

Various techniques exist to solve optimization problems (OP). These techniques can be grouped into three categories: enumeration, deterministic, and stochastic techniques [140, 159]. Each of these techniques has advantages and disadvantages. While it is not within the scope of this effort to detail all the possible techniques for solving OPs, an explanation of the categories and motivation for using these approaches is presented. Enumeration techniques involve conducting a total enumeration of the search space in order to look at every possible solution to the OP and be guaranteed to generate the optimal solution(s). While this technique results in the research objective, generation of the optimal solution, enumeration techniques are generally infeasible when applied to large problems, and are best applied to problems containing a few discrete decision variables [159, 205].

A total enumeration of the search space is not the best approach if the enumeration of the space is too complex to finish within a reasonable amount of time or if the problem is constrained (enumeration techniques typically do not detect infeasible combinations of decision variables prior to computing their fitness value). These are just two reasons that enumeration techniques are not always a good technique to use. One would like to generate solutions to an OP within a reasonable amount of time and enumeration approaches cannot meet this requirement for complex, large scale OPs.

Deterministic techniques for solving OPs include: Divide and Conquer, Dynamic Programming, Hill Climbing, Branch and Bound, Depth-First (Greedy, Backtracking), Breadth-First, and Best-First (A^*, Z^*) search and tabu search to name a few. These approaches typically incorporate problem domain knowledge to reduce the size of the search space that is actually analyzed [140]. Throughout the deterministic search process, certain paths may be deemed inferior to others, and only paths of potentially better solution quality are fully explored. Many deterministic search techniques proceed to search the space through through a tree or graph like process. While these techniques are typically more efficient than conducting an enumeration of the space, they are still time consuming when applied to large scale OPs.

Divide and conquer takes the approach of decomposing a difficult problem into multiple, manageable smaller problems. The manageable subproblems are solved recursively to yield a solution to the original OP [140]. Dynamic programming uses a recursive procedure to solve a problem. The main idea is to decompose a large problem into small, one stage, incremental subproblems. The subproblems are solved and the solutions to these subproblems are used to solve larger and larger subproblems. Eventually the original problem is solved [205]. Hill Climbing uses a greedy approach that compares multiple solutions and chooses the best of those compared. This approach assumes that the path to the optimal solution is one that involves selecting the best solution at each step of the search process [140].

Breadth-first, depth-first, and best-first search solve a problem through a tree like structured search process. Breadth-first search generates all nodes at a specific level of the tree structure prior to the generation of nodes at the next lower level of the tree [140].

Depth-first search is similar to breadth-first search but generates all of the nodes, down to the lowest depth or level, of a specific branch of the tree prior to solving the next branch [159]. Backtracking is used to go back if a previously promising node turns out to yield a poor solution. Backtracking allows one to get back to a previous promising node and continue the search process down an alternate path [148]. Best-first search always selects the next most promising node to evaluate and is a greedy based approach [57]. A^* and Z^* are special cases of best-first search algorithms. A^* and Z^* use the cost and estimates of the cost to continue searching until the solution is found in order to select a path to evaluate [140].

Branch and bound is a heuristic that works on the concept that continually partitioning the search space and finding the lower bound on any solution can be used to find the optimal solution. The bound of each of the nodes is used to determine which node is the most promising and the algorithm branches to that node [140]. A heuristic search method uses all of the available information in order to follow the most promising path to a final solution [148]. Tabu search takes a different approach to solving optimization problems. In tabu search a finite (tabu) list is maintained of previous movements within the search space. The purpose of this list is to prevent or penalize the search process when moves are selected that yield a solution already visited in the search space. The tabu list aids the search process in visiting unexplored areas of the search space and the tabu list itself is updated as the search progresses [73, 159]. Tabu search uses information gained from its movement throughout the search space to explore new regions of the search space and move back to previously explored regions that are of high quality.

The final category of search technique discussed is that of stochastic techniques, the focus of this effort. Stochastic searching techniques make use of some type of randomness or random process in the search process [140]. Stochastic techniques include: Random Search, Simulated Annealing, Monte Carlo, and Evolutionary Algorithms to name a few. These approaches use a random process and maintain some type of record of good solutions found. This information is used in the search process to generate better solutions as the search progresses.

A purely random search randomly generates solutions until some stopping criteria is met [140]. At the conclusion of the search, the best solution found is presented to the user. As this process is entirely random, knowledge of good solutions or promising areas of the search space are not recorded and hence one cannot always expect this method to yield solutions of high quality. A Monte Carlo search randomly generates solutions to a problem multiple times. When a large enough sample size is used, the distribution of values found approximates the true output distribution [159].

Simulated annealing is an approach that mimics the process of annealing metals to improve their overall strength. The simulated annealing search process moves throughout the search space to areas of improved solution quality. Areas of the the search space that do not improve the overall solution quality are also explored based on a randomly generated probability that is typically small. The exploration of areas of the solution space of lesser quality may prevent premature convergence to a local optima. As the algorithm progresses, a temperature like coefficient is used to "cool" or change the probability with which solutions of lower quality are accepted [148, 159].

Since these techniques typically do not search the entire space for very large OPs, they are not guaranteed to find the optimal solution unless they are executed for an infinite amount of time. However, these approaches typically generate good solutions for problems in which the search space is not totally chaotic. This research effort is concerned with attempting to solve difficult OPs, that may have multimodal landscapes, are time consuming to solve, are of high dimensionality, and require an acceptable solution within a reasonable amount of time. This means that the landscape is not totally chaotic, but that there may be multiple local optima and a single or multiple global optima. OPs having these characteristics are difficult to solve but evolutionary approaches are suited to solve this type of OP [74]. Hence an evolutionary approach is the focus of this effort.

Evolutionary Algorithms (EA) have been a focus of development for a number of years [74]. EAs are a population based class of algorithm that have a direct link to biology and the concept of optimal processes. These algorithms can be decomposed into four main sub-classes, Evolutionary Strategies (ES), Evolutionary Programming (EP), Genetic Programming (GP), and Genetic Algorithms (GAs). The difference in these approaches

lies in the operators and representation used. Genetic algorithms typically use a binary representation for the population members and crossover but little to no mutation whereas evolutionary programming uses a real-valued or symbolic representation for the population members and primarily uses a mutation operator to generate solutions. Evolutionary strategies use a real-valued representation for the population along with a combination of crossover and mutation. Genetic programming solves an optimization problem through a tree like structure consisting of programs that are randomly altered. A number of problems exist with the implementation of GP including the details of randomly altering a computer program while maintaining a correct implementation [13]. The reader is directed to the work of Thomas Bäck [13] for a more historical and theoretical discussion concerning the background of ES, EP, and GAs. The discussion presented primarily concentrates on GAs.

The motivation for use of EAs originated from attempts to find more efficient and effective methods of solving optimization problems. A further motivation is to solve problems that cannot currently be solved to optimality due to a lack of computing power necessary to find the optimal solution in a reasonable amount of time. EAs can be useful when applied to problems in which the search space is large, is not known to be smooth or unimodal (the optimal solution lies in a single peak or valley), a solution of good quality must be found very quickly, if the fitness functions exhibit noise, or if the problem is not well understood and hence a better optimization method is not known to work well on this problem [146]. EAs are not guaranteed to find the optimal solution unless given an infinite amount of time, but they offer the ability to find a "good" solution, and sometimes the optimal solution to optimization problems in an acceptable amount of time. It is important to understand that there may be more than one optimal solution as often occurs in real-world MOPs. These optimization problems involve the maximization or minimization of an *objective* or *fitness function*, where the fitness function allows one to compare the quality of a solution to other solutions. The overall goal of this optimization is to find the global optimum over the entire search space.

Genetic Algorithms date back to John Holland's original work in 1962 on the theory of adaptive systems [90], and his work on adaptive and reproductive plans [91]. This is considered by many to be the starting point of most implementations of Genetic Al-

gorithms [13, 14, 74, 140, 201]. Theoretical analysis and symbolic formulations of single objective GAs has been discussed in numerous publications, with some of the predominant sources being Thomas Bäck [13] and David Goldberg [74].

The basic concept behind EAs is that of evolution through time. To expand on this, evolution is defined as "a process of continuous change from a lower, simpler, or worse to a higher, more complex, or better state [136]." EAs are stochastic, population based algorithms designed to "evolve" a population over time (generations). "Evolution" occurs through the use of selection and recombination operators [201]. Generic pseudocode for an EA is presented in Figure 2.1.

```
Randomly Initialize the Population
Evaluate the Fitness Function Values
For i = 1 to Maximum Number of Generations
    Perform Recombination
        Perform Crossover Based on p_c
        Perform Mutation Based on p_m
    Evaluate the Fitness Function Values
    Perform Selection to Generate the Next Population
End Loop
```

Figure 2.1 Evolutionary Algorithm Pseudocode [74]

A typical EA begins with a random generation of bit strings of a specified length. Each bit string a is referred to as an individual and the collection of all μ individuals represents the parent population P. Each bit position within a population member is referred to as the gene position or locus. The value in any gene position is referred to as the gene value or allelic value, a 1 or 0 for a binary EA.

The random generation of a population of individuals is meant to uniformly distribute the initial population throughout the landscape. Most EAs are binary and this effort concentrates on the binary class of EAs. However, real-valued and other variants exist. Once generated, the parent individuals $\{a_1, \ldots, a_\mu\}$ are evaluated to determine their objective function values $f(a)$. It is noted that an EA has both an objective function and a fitness function. The objective function is defined by the problem of interest whereas the fitness function is a measure of the quality of the candidate solution [13, 132, 184]. Following the

evaluation of the population members, a selection operator is applied. Various selection operators exist. Some of the most popular ones are presented in Table 2.1 [13, 74, 201].

Table 2.1: Types of Selection

Selection Type	**Description**
Tournament	n individuals chosen at random, the one best of the n is selected.
Roulette Wheel (Fitness Proportionate)	Each population member is assigned a proportion of the roulette wheel equal to the ratio of its fitness to the sum of the entire population's fitness
Elitist	Choose the best population members ("Survival of the Fittest").

Once the selection process is complete, the population P enters the recombination or "mating" phase of the algorithm. In the recombination process, individuals are typically randomly selected from the population with a probability of crossover, p_c. In the case of a binary EA with one-point crossover, a crossover point is randomly chosen and two population members are "crossed over" at that location. All of the bits following the crossover point are exchanged between the two population members. Crossover continues until an intermediate population P' of size μ is generated. Many types of crossover exist; a partial listing is presented in Table 2.2 and the interested reader is referred elsewhere for a more detailed description [13, 78, 201].

Table 2.2: Types of Crossover

Crossover Type	**Description**
Single Point	Random crossover point chosen, all bits following the crossover point are exchanged
2-point (n-point)	2 (n) crossover points are randomly chosen. The segment of the string between the 2 (n) points is exchanged.
Uniform	Choose each bit randomly from each of the parents, with equal probability. Each bit is independent of other bits and the crossover point.

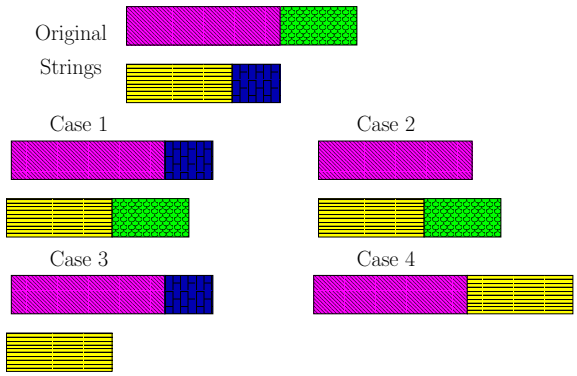

Figure 2.2 Cut and Splice

Table 2.2: (continued)

Crossover Type	**Description**
Cut and Splice	Crossover operator for variable length string encodings. Different crossover points are chosen in each parent with bits following the crossover points exchanged.

Figure 2.2 illustrates an example of the Cut and Splice operation, or crossover for variable string length encodings. Some EAs allow the individual string lengths to vary throughout the evolutionary process but do maintain a maximum possible string length. In conducting cut and splice, one uses two probabilities during the operation, a probability that each string is cut at a random point and a probability that each string is spliced. Four potential results from a cut and splice operation performed on two selected strings are illustrated. Case 1 illustrates the result of performing two cut and two splice operations. Case 2 illustrates the result of a single cut and splice operation, Case 3 illustrates this same situation in the reverse order, and Case 4 illustrates the results of zero cut and two splice operations.

Following the crossover process, mutation occurs on each population member with a probability of mutation p_m. The mutation process involves a random choice of a bit position or loci to flip from a 1 to a 0 or vice-versa (for a binary representation). The mutation process yields an intermediate population P'' of size μ.

Following crossover and mutation, a selection mechanism is applied to generate the next generation's population. Some of the the most popular selection mechanisms are presented in Table 2.1. The purpose of the selection mechanism is to keep the "best" population members, which in essence contain the "best" Building Blocks (BBs), in the population as the generations progress. An elitist tournament selection mechanism would implement a "survival of the fittest" concept where x population members would be randomly chosen from the population and their fitness values compared to each other, only the "fittest" member of the x being compared would survive and be placed into the next generation's population $P(t+1)$. A random selection mechanism randomly selects members from the population to be placed into the next population. A roulette wheel selection operator assigns each population member a probability of being selected based on the percentage of the total population fitness that each member represents. The selection mechanism is executed μ times so that the population remains constant from generation to generation.

Numerous single objective EAs have been designed, implemented, and tested using various metrics and test suites optimization problems. Two examples are presented to aid the reader's understanding of a simple EA. One of the most referenced single objective EAs is the Simple Genetic Algorithm (SGA) originally created by Goldberg [74] and sometimes referred to as a canonical GA [201]. The main concept of this algorithm is fairly basic and easy to implement. A starting population of individuals is randomly created using a binary string of a specified length. Following the generation of the initial population, the members are evaluated. Selection is then applied to the population in order to generate an intermediate population of individuals. Tournament selection is one of the most popular and is often used. Recombination operators are applied to this intermediate population with their respective probabilities of crossover and mutation. Recombination operators are used to "move" the population towards better solutions to the fitness function being solved. The resultant population following the recombination operators becomes the next generation's population. The algorithm then executes until it reaches the stopping criteria, typically a user specified number of generations.

Another simplistic GA that can be used is the steady state GA [201]. The steady state GA was designed to converge slowly to a final solution and hence the GA operators are only applied to a single individual at a time. The population is randomly created as in the simple GA and the fitness values are calculated according to rank. Recombination occurs between two individuals; the resultant child is put back into the parent population, and mutation follows. The child replaces the worst individual in the population, thereby maintaining a constant population size. This process allows for a much slower rate of convergence than the simple GA and others like it since only one population member is modified at a time. This slow convergence allows for the potential to explore different areas of the landscape.

2.2 Building Block Hypothesis

This section discusses the background of BBs. The cornerstone of many of the GAs to date is the Building Block Hypothesis (BBH), also referred to as the Schema Theorem [13, 74, 184, 201]. A schema or BB is a partial string in which certain allelic values are present in specific loci positions. These schema are defined to be strings of length l drawn from the alphabet {0,1,*}. An example of this is the schema 0*1, which represents the set of all 3-bit binary strings containing a 0 in the first position and a 1 in the last position; 0*1 = {001,011}. Further, a BB is a partial string that an epistatic relationship exists between the BB alleles in their loci positions (genes). Epistatic relationships reflect the fact that one gene value may mask the effect of another gene [154]. In other words, if two specific genes are present within a string, then the two genes interact with each other, effecting the overall fitness value.

A good BB is a BB that when combined with other genes, to generate a fully specified individual, typically yields a fitness value of high quality. A fully specified individual is one in which all of the gene values are present. It is important to understand that a BB can only be evaluated when combined with other gene values to generate a fully specified individual. As multiple combinations of genes can be combined with a BB, a BB that typically yields a high fitness value when evaluated with various combinations of genes is designated a good BB. The evaluation of a good BB typically yields fitness values that are of higher

quality than the evaluation of the fitness value obtained for strings containing different allelic values in the same gene positions. Depending on the actual optimization problem, only one good BB may exist or many may exist that lead to the global optima. The overall concept is that the identification of the good BBs by an EA or MOEA leads to a higher probability for the EA or MOEA to generate the optimal solution to the problem versus an algorithm that does not find the good BBs. Hence explicit BB-based EAs and MOEAs may have an advantage over implicit BB-based EAs and MOEAs in finding solutions to problems.

Two measures that are of importance in determining the probability that a particular schema (H) survives within a GA population when subjected to crossover and mutation are the defining length and the order of the schema. The defining length $\delta(H)$ of a schema H is the distance between the first and last positions in the string that are specified. In the example presented, the defining length is 3-1=2. The last term of importance is the order of a schema. The order $o(H)$ is defined to be the overall number of specified positions within the schema. In the example, the order is 2 since positions 1 and 3 are specified.

An order o BB is defined by Merkle as:

Definition 1 ((Order-o) potential building block [132]): *Let $\mathcal{A}$ be a non-empty set (the* genic alphabet*), $\ell \in \mathbb{Z}^+$ (the* nominal string length*), $\mathcal{L} \triangleq \{1, \ldots, \ell\}$ (the* loci*), and $\mathcal{S} \triangleq \{(a_1, l_1), \ldots, (a_o, l_o)\} \in 2^{\mathcal{A} \times \mathcal{L}}$ a set of genes. If the loci of $\mathcal{S}$ are distinct, i.e., $\mathcal{S}$ satisfies $i = j \Longleftrightarrow l_i = l_j$, then it is called an* order-o potential building block *or simply a* potential building block*.* □

A specific schema increases its presence in the population based on the ratio of the average fitness $f(H)$ of the schema to the average fitness of the population $\bar{f}$, i.e., a schema with an average fitness above the average fitness of the population ($f(H) > \bar{f}$) receives an increasing number of copies within the population through reproduction. Essentially, reproduction increases the number of above average schema and decreases the number of below average schema, driving the population to converge to a good solution. Therefore

the schema growth equation is:

$$m(H, t+1) = m(H, t) * \frac{f(H)}{\bar{f}} \tag{2.1}$$

Where $m(H, t+1)$ represents the schema H in the population at time $t+1$, $f(H)$ is the average fitness of the strings representing H at time t, and $\bar{f}$ is the average fitness of the population. This yields the number of schema H expected to be present in the subsequent generation.

Crossover and mutation may have the effect of disrupting schema in the population or the positive effect of combining and manipulating good BBs to yield good solutions. Crossover has a disruptive effect if the crossover point falls within the defining length of the schema and mutation has this effect if a defined bit is mutated. If single-point crossover is used, choosing a random crossover point p_c, then the probability of a schema surviving is p_s. Single point crossover is used for illustration purposes, other crossover schemes may be used and require the equations to be adjusted accordingly.

$$p_s \geq 1 - p_c * \frac{\delta(H)}{l-1} \tag{2.2}$$

To add the effect of crossover to the effect of reproduction, assuming independence of reproduction and crossover operators, the following estimate for the number of schema H expected in the next generation is obtained [74]:

$$m(H, t+1) \geq m(H, t) * \frac{f(H)}{\bar{f}} * \left[1 - p_c * \frac{\delta(H)}{l-1}\right] \tag{2.3}$$

The estimate of the combined effect of crossover and reproduction is obtained by multiplying the expected number of schema following reproduction by the survival probability of crossover with reproduction. Mutation effects a single bit position with probability p_m. In order for mutation to disrupt a particular schema, it must effect one of the specified bit positions. In order to determine this, the probability that a particular allele survives is $(1 - p_m)$ and since each mutation event is independent, a schema survives if each of the specified bits $o(H)$ remains unchanged, and hence the probability of survival is $(1-p_m)^{o(H)}$,

which can be approximated by $1 - o(H) * p_m$ if $p_m << 1$ [74, 201]. A particular schema receives an expected number of copies in the next generation when subject to reproduction, crossover, and mutation as stated:

$$m(H, t+1) \geq m(H, t) * \frac{f(H)}{\bar{f}} * \left[1 - p_c * \frac{\delta(H)}{l-1} - o(H) * p_m\right] \tag{2.4}$$

2.3 Multiobjective Evolutionary Algorithm Overview

Since single objective problems were the initial focus of EA researchers, there exist a large number of publications concerning the theory, implementation and results of single objective EA research [13, 14, 74, 132]. With the increased processing power of modern computers, researchers gained the ability to find better solutions to many of the single objective problems that were previously too time consuming to optimize. This led the EA community and many researchers in industry to begin applying EAs to real-world optimization problems [25, 27, 28, 49, 51, 93, 117, 193], of which there are hundreds of application based publications in the literature [26]. In using EAs to attack these problems and combining them with parallel concepts, researchers found that they were now able to find solutions for their problems quite rapidly when compared to previous use of EAs.

One of the issues encountered when attempting to solve real-world optimization problems is in choosing a method to optimize the problem which takes into account all of the objectives. In the past, EA researchers combined the objective functions into one single objective function and used a single objective EA to find an acceptable solution. As more and more researchers from industry became interested in the power of EAs, many recognized the need to explicitly employ individual objective functions. The use of individual objective functions means that problems formulated in this manner do not encounter the need to determine weights for each of the objectives as is done in the combined objective function aggregation formulation. The decreased execution time of the EA, a desire to develop good, feasible solutions to real-world optimization problems, and Multiobjective

Optimization Theory led some EA researchers to begin experimenting with more than just the basic single objective optimization problem [78, 170]. [1]

Results achieved via use of Single-Objective EAs (SOEAs) as applied to MOPs lacked efficiency and were not effective for many continuous MOPs. For example, SOEA approaches using aggregated MOP objective functions (e.g., weighted sum) generate only a single solution per run and require multiple executions varying weight aggregations in order to generate multiple solutions. Although it is possible to use SOEAs in solving an MOP, these methods are typically not as effective or efficient as MOEAs, which are designed to return a small number of MOP solutions per run. Additionally, some traditional SOEA approaches are known to be unable to identify the complete Pareto front if certain problem constraints exist [31]. However, many MOEAs have been successfully applied to real-world design and constrained optimization problems, leading to their increased visibility and use, but MOEA efficiency objectives still exist.

The development and analysis of single objective EAs has motivated the development of MOEAs. MOEAs are of particular interest to engineers, scientists, and problem solvers due to the fact that many of the real-world problems that exist today are multiobjective and in many cases constrained. An example of this is the Advanced Logistics Problem [216] which involves logistic research into resource allocation. The Advanced Logistics Problem is a constrained real-world multiobjective optimization problem. A more detailed list of real-world MOPs can be found in [26, 31, 44].

If a generic MOEA can be designed, implemented, and its performance validated, this would be of great interest to researchers and those in industry attempting to solve MOPs. The development of a generic evolutionary algorithm for MOPs has been a goal of MOEA researchers that has produced a number of publications. The most widely known and referenced algorithms are discussed in some detail in Section 3.1. However, the No Free Lunch (NFL) Theorem [204] states that it is not possible to find one algorithm that outperforms every other algorithm in the attempted solving of all classes of optimization problems. Does this mean that the goal of finding a generic MOEA to solve MOPs is

[1] It is noted that the concept of Multiobjective Optimization Theory has existed since the early 1940s in investment problems and was extended outside this problem domain area in the 1960s [31].

pointless? No, this research advances the theory of MOEAs and the development of better operators. These advances can lead to the development of better MOEAs suited to particular classes of MOPs. While the design of an MOEA that outperforms every other MOEA on all classes of MOPs may not be possible, it is possible to design a generic MOEA that performs better than other MOEAs on a particular class of MOP or on average performs well across a variety of MOPs.

Many of the existing MOEAs incorporate a variety of genetic operators and are based on a range of evolutionary methods and operators. Prior to discussing the validation of any MOEA, the formulation of that algorithm and its comparison to a generic MOEA's formulation must be addressed. This helps to clarify the underlying concepts of the respective algorithm and aid in understanding the theory associated with it.

2.4 Other Approaches to Solving MOPs

Multiobjective optimization problems can be solved through the use of many different methods. Each of these methods has advantages and disadvantages that typically are dependent on the specific problem being solved, the characteristics of the decision variable or genotype space and the subsequent characteristics of the objective function or phenotype space. Some of the approaches found in the Operations Research (OR) community include aggregation methods, knowledge based methods, mathematical programming and constrained optimization methods to list a few [159].

Aggregation methods sum or aggregate the fitness functions together into a single objective function, typically with different weighting factors applied to each of the objective functions to show a preference towards a particular objective [148]. The difficulty in utilizing this method is in determining the correct weighting factor(s) to use [67].

Knowledge based methods, sometimes referred to as Surrogate methods, incorporate knowledge about the objective function values found up to a point in time to predict whether or not a particular solution can improve on the best solutions found so far [148]. Another method of handling multiple objectives in a single objective algorithm is to treat one objective as a single objective optimization problem and treat the other objectives as

constraints [31]. This method can work very well but may involve additional overhead in the handling of infeasible points in the space. Iterative methods are also popular, and in this approach, the decision maker watches the search process progress and interactively guides the search into different areas of the landscape that he or she may be most interested.

Lexicographic methods consider the objectives one at a time [159]. The order in which the objectives are considered is the order of importance specified by the researcher. The most important objective is optimized first, followed by the second most important and so on until all of the objectives are optimized. Mathematical programming methods refer to linear and nonlinear programming approaches, as well as others, to optimization problems. Linear programming procedures solve problems consisting of objective functions and constraints that are linear in the decision variables. Nonlinear programming is similar except the decision variables are nonlinear in nature [57].

In the late 1800s, the mathematical foundations were laid for the infinite dimensional ordered spaces necessary to understand MOPs [177]. Around 1944, John von Neumann and Oskar Morgenstern mentioned an optimization problem that contained conflicting objectives [197]. This was the start of the idea of MOPs. However, neither they, nor anyone else, elaborated on this problem until the 1950s when Tjalling C. Koopmans discussed the idea of "efficient" in his book [118]. When the concept of vector maximization problems was presented by Kuhn and Tucker in 1951, multiobjective optimization became a discipline of its own [31]. By the 1960s, MOPs became more common in the field of economics and "trade-off" became a common term to use [32]. The interest in this area grew and the application of multiobjective optimization to other areas began to increase.

Multi-Criteria Decision Making (MCDM) is a term used by operational researchers to describe the process of choosing a solution from the set of Pareto optimal points. The two main ideas of MCDM are based on outranking concepts and utility functions [92]. In the outranking concept, comparisons are made based on the goal of determining if a preference, indifference, or incomparability exists between the pair of objects. This approach may be computationally expensive. The utility function approach, based on the Multi Attribute Utility Analysis (MAUA), assumes that a utility function is available to identify the set of solutions [92, 93]. MCDM is a very important aspect of solving MOPs. The solution

of an MOP is a set of solution vectors that the Decision Maker (DM) must analyze to see which one of the vectors should be chosen. Once a solution set is generated, the issue becomes how to choose the final solution and when does the DM specify their preference. This preference specification may be done prior to, during, or after execution of the search. Additionally, the DM aspect of the search may take as much execution time as the search procedure [92].

Multiobjective Combinatorial Optimization (MOCO) is another term used by the OR community to reference multiobjective works. Some of the approaches used in OR include Multiobjective Simulated Annealing, Evolutionary Algorithms, Tabu Search, Linear Programming, etc [59]. As some authors have stated, a more involved effort between the OR and EA communities may benefit both communities as more MOEA researchers are attempting to integrate decision analysis concepts into their MOEAs [34].

The last method to be discussed is that of using a Pareto based approach. A French-Italian economist named Vilfredo Pareto (1848-1923) first developed the principle of Multiobjective optimization for use in economics. His theories became collectively known as Pareto's optimality concept [31]. These principles allow one to analyze the results of a simultaneous optimization of multiple objectives and determine the set of points that are optimal with respect to all of the other points generated. The work presented in this research uses the concepts of Pareto since this method does not require the researcher to determine an optimal weight for the objective functions. Nor does it require a selection of one objective function that is of greatest importance. This method allows for a simultaneous optimization of multiple objective functions that lends itself to utilization in MOEAs.

2.5 MOP Domain Formalization

Some of the methods that exist to solve OPs and MOPs are discussed in previous sections of this chapter. This research effort uses an MOEA to attempt and find solutions to MOPs as well as integrating parallel processing concepts and addressing MOEA theory. Prior to discussing what an MOP consists of and presenting formal definitions, some terminology must be discussed. MOPs require a method of evaluating the quality of a

solution. In SO optimization, one can compare two potential solutions to each other and through the single fitness value determine which of the solutions is of higher quality. The comparison of two potential MO solutions presents the challenge of determining the same objective, which solution is better, but with the addition of multiple fitness values. Pareto Optimality is used to determine the quality of multiple MO solutions in this effort.

2.5.1 Pareto Terminology. The concept of Pareto Optimality is integral to the theory and analysis of MOPs. Pareto Optimality Theory and the associated Pareto concepts are used as a way to determine if one solution is "better" than another in a multiobjective comparison. Although single-objective optimization problems may have a unique optimal solution, continuous MOPs usually have a possibly uncountable set of solutions, which when evaluated produce vectors whose components represent trade-offs in decision space. Some key Pareto concepts, for minimization of continuous MOPs, are defined mathematically as [17]:

Definition 2 (Pareto Dominance): *A vector $\vec{u} = (u_1, \ldots, u_k)$ is said to dominate another vector $\vec{v} = (v_1, \ldots, v_k)$ if and only if u is partially less than v, i.e., $\forall i \in \{1, \ldots, k\},\ u_i \leq v_i \wedge \exists i \in \{1, \ldots, k\} : u_i < v_i$.* □

Definition 3 (Pareto Optimality): *A solution $x \in \Omega$ is said to be Pareto optimal with respect to Ω if and only if there is no $x' \in \Omega$ for which $\vec{v} = F(x') = (f_1(x'), \ldots, f_k(x'))$ dominates $\vec{u} = F(x) = (f_1(x), \ldots, f_k(x))$.* □

The phrase "Pareto optimal" is taken to mean with respect to the entire decision variable space unless otherwise specified. The variable x represents a vector of decision variables and F represents a vector of objective function values.

Definition 4 (Pareto Optimal Set): *For a given MOP $F(x)$, the Pareto optimal set ($\mathcal{P}^*$) is defined as:*

$$\mathcal{P}^* := \{x \in \Omega \mid \neg\exists\, x' \in \Omega \;\; F(x') \preceq F(x)\}. \tag{2.5}$$

□

Definition 5 (Pareto Front): *For a given MOP $F(x)$ and Pareto optimal set $\mathcal{P}^*$, the Pareto front ($\mathcal{PF}^*$) is defined as:*

$$\mathcal{PF}^* := \{\vec{u} = F(x) = (f_1(x), \ldots, f_k(x)) \mid x \in \mathcal{P}^*\}. \tag{2.6}$$

□

Pareto optimal solutions are those solutions within the search space whose corresponding objective vector components cannot be simultaneously improved. These solutions are also termed *non-inferior*, *admissible*, or *efficient* solutions, with the entire solution set represented by $\mathcal{P}^*$ [190, 219]. Their corresponding vectors are termed *nondominated*; selecting a vector(s) from this nondominated vector set (the Pareto front set $\mathcal{PF}^*$) implicitly indicates acceptable Pareto optimal solutions (genotypes). These solutions may have no clearly apparent relationship besides their membership in the Pareto optimal set. It is simply the set of all solutions whose associated vectors are nondominated; these solutions are placed into the set of Pareto optimal solutions based on their *phenotypical* expression. Their expression (the nondominated vectors), in phenotype space, is known as the *Pareto front* [190, 219].

2.5.2 MOP Formulation. The process of finding the global minimum or maximum of any single objective function is referred to as Global Optimization. In general, this is presented in Definition 6 as stated in Bäck and others [13, 114, 184]:

Definition 6 (Global Minimum): *Given a function $f : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}$, $\Omega \neq \emptyset$, for $\vec{x} \in \Omega$ the value $f^* \triangleq f(\vec{x}^*) > -\infty$ is called a global minimum if and only if*

$$\forall \vec{x} \in \Omega : \quad f(\vec{x}^*) \leq f(\vec{x}) \ . \tag{2.7}$$

Then, $\vec{x}^*$ is the global minimum solution(s), f is the objective function, and the set Ω is the feasible region. The problem of determining the global minimum solution(s) is called the global optimization problem. □

In Definition 6, the n-dimensional decision variable space Ω and the objective function space, denoted as $\mathbb{R}$, are both shown to be subsets of real values in the definition. These two spaces may not be restricted to the same subset of real values. For example, the decision variable space may be restricted to integer values whereas the fitness function space may be restricted to real values. This is dependent on the mapping of the objective function. The point is that the decision variable and objective function spaces may or may not be of the same type or have the same characteristics (continuous, disconnected, discrete, etc.).

This single objective formulation must be modified to reflect the nature of multi-objective problems where there may be a set of solutions found through the analysis of associated Pareto Optimality Theory. Many times multiobjective problems require the decision maker to make a choice, which is essentially a tradeoff, of one solution over another in objective space [31, 44, 74, 184, 207]. Prior to presentation of the associated multiobjective definition, an MOP must be defined.

Multiobjective problems are those where the intent is to optimize k objective functions simultaneously. This may involve the maximization of all k functions, the minimization of all k functions or a combination of maximization and minimization of the k functions. An MOP and an MOP global minimum (or maximum) is formally defined as [31, 44, 114, 184, 207]:

Definition 7 (General MOP): *In general, an MOP minimizes (or maximizes)* $F(\vec{x}) = (f_1(\vec{x}), \ldots, f_k(\vec{x}))$ *subject to* $g_i(\vec{x}) \leq 0,\ i = 1, \ldots, m,\ \vec{x} \in \Omega$. *An MOP solution minimizes the components of a vector* $F(\vec{x})$ *where* $\vec{x}$ *is a* n*-dimensional decision variable vector* ($\vec{x} = x_1, \ldots, x_n$) *from some universe* Ω. *The vector function* $F(\vec{x})$ *maps the set* Ω *into the set* Λ *which represents all of the possible values of the objective functions.* □

Definition 8 (MOP Global Minimum): *Given a function* $F : \Omega \subseteq \mathbb{R}^n \rightarrow \mathbb{R}^k$, $\Omega \neq \emptyset$, $k \geq 2$, *for* $\vec{x} \in \Omega$ *the set* $\mathcal{PF}^* \triangleq F(\vec{x}_i^*) > (-\infty, \ldots, -\infty)$ *is called the global minimum if and only if*

$$\forall \vec{x} \in \Omega : \quad F(\vec{x}_i^*) \preceq F(\vec{x}) \ . \tag{2.8}$$

Then, $\vec{x}_i^*$, $i = 1, \ldots, n$ is the global minimum solution set (i.e., $\mathcal{P}^*$), F is the multiple objective function, and the set Ω is the feasible region. The problem of determining the global minimum solution set is called the MOP global optimization problem. □

An MOP consists of k objectives reflected in the k objective functions, m constraints on the objective functions and n decision variables. The k objective functions may be linear or nonlinear in nature [31, 44]. The evaluation (fitness) function, $F : \Omega \longrightarrow \Lambda$, is a mapping from the decision variables ($\vec{x} = x_1, \ldots, x_n$) into output vectors ($\vec{y} = a_1, \ldots, a_k$), where $F(\vec{x}) = \vec{y}$ [114, 184, 207]. The same statement holds as it does for the single objective optimization problem; the two spaces, Ω and Λ, are subsets of the real values, and the characteristics of the spaces may be connected, disconnected, continuous, discrete, etc.

MOPs typically consist of competing objective functions [65, 114, 207]. The competing objective functions may be independent or dependent on each other. An example MOP is a company's quest to purchase a backbone for their computer network that provides the greatest throughput at the least monetary cost. The objectives of maximizing throughput and minimizing cost are highly dependent on each other as increased cost typically results in increased throughput and vice-versa.

Van Veldhuizen validated the concept that BBs exist and are useful in the multiobjective domain [184]. Even though the Schema Theorem was originally developed in terms of single objective functions, the applicability of the Schema Theorem is easily extended to multiobjective functions. Van Veldhuizen states that BBs are not handled differently by MOEAs as compared to EAs. Even though an MOEA simultaneously optimizes multiple fitness functions, the genotypical (decision variable) representation of potential solutions (population members) in an MOEA is no different than the genotypical representation used in a single objective EA. Since the same genotype structure is used in both single objective EAs and MOEAs, MOEA BBs are defined the same as single objective EA BBs. A symbolic definition of a single objective BB is presented in Definition 1 and defines a BB as a combination of gene values (alleles) that are located in distinct but not necessarily adjacent locations (loci). The definition of a good single objective BB is presented in Definition 9

Definition 9 (Good Single Objective Building Block): *A good single objective BB, meets the requirements of Definition 1 and the mean fitness value of the BB evaluates to a good fitness value over a number of different allelic combinations placed in the unspecified loci.* □

Even though single and multiobjective BBs are identically defined, good BBs are not identical in the single objective and multiobjective domains. The difference lies in the comparison conducted in order to determine which of two selected population members have the best fitness. In a single objective problem, a direct comparison of the two fitness values is conducted and one clearly can determine which of the two fitness values is better or the fitness values may be equal. In the context of multiobjective optimization and MOEAs, Pareto dominance criteria is used to determine the nondominated population member(s). Hence a good multiobjective BB is presented in Definition 10.

Definition 10 (Good Multiobjective Building Block): *A good multiobjective BB, meets the requirements of Definition 1 and the mean fitness value of the BB dominates (evaluate to a good fitness value as compared to) the fitness values of other BBs or population members in comparison testing based on Pareto dominance criteria. The evaluation of a multiobjective BB is conducted over a number of different allelic combinations placed in the unspecified loci.* □

Therefore the structure of a multiobjective BB is identical to a single objective BB but the determination of the quality of a multiobjective BB is based upon Pareto dominance criteria. In the remainder of this effort, the statement 'a good BB' refers to either Definition 9 or Definition 10 dependent on the domain of interest, i.e., a discussion of good BBs in MOEAs implicitly refers to Definition 10 as defining a good BB.

It is necessary to define additional terminology to remain consistent with the terminology used in the EA field. The term *objective* is defined as the goal of the MOP to be achieved and is represented by the functions f_k. The *objective space*, Λ, refers to the coordinate space within which vectors, $f_k(\vec{x})$, resulting from the MOP evaluation of the k objective functions, are plotted [184].

The genotype and phenotype spaces are important to understand and characterize when discussing any MOP. The characteristics of the genotype and phenotype spaces of an MOP may directly effect the search process and the ability of an MOEA to generate high quality results. This is especially important when analyzing why a particular MOEA performs poorly in solving various MOPs. In classifying MOPs, two broad categories exist: unconstrained and constrained MOPs. Besides these two broad categories, MOPs can be analyzed in terms of the characteristics of their spaces and the solution sets generated.

In general, one initially selects an MOP mathematical model with ($\mathcal{P}^*$, $\mathcal{PF}^*$) and then discretizes this model. This discretization leads to a computational model (discrete algorithmic formulation with finite word length data structures) that an MOEA can process in order to attempt to solve the problem. The set PF_{true} is defined by the functions composing an MOP; it is of fixed size, fixed resolution, does not change, and represents the solution set that the MOEA is attempting to find. The set PF_{true} can be considered a subset of the theoretical true solution, $\mathcal{PF}^*$, of potential infinite resolution and cardinality. The theoretical $\mathcal{P}^*$ and $\mathcal{PF}^*$ sets in many MOPs cannot be characterized since their spaces may encompass the reals ($\mathbb{R}$) or a subset of the reals as specified in the previous definitions. Since one cannot represent solutions of infinite precision on a computer, $\mathcal{P}^*$ and $\mathcal{PF}^*$ cannot be truly realized. An example requiring infinite precision decision variables is seen in the attempt to represent π on a computer; π cannot be represented at its true, infinite resolution.

The discrete genotype solution set P_{true} of an MOP may be connected, disconnected, symmetric or scalable. The discrete phenotype solution set, PF_{true} , of an MOP may have the characteristics of being connected, disconnected, concave, or convex. A discrete, connected Pareto front consists of points that form a single curve or surface in which there are no discontinuities between points. Since the Pareto front is discretized, gaps may exist between individual points points cannot exist between these gaps due to the resolution of the decision variables used. MOP attributes can be determined through a visualization of the solution sets. An increase or decrease in the resolution of the MOP formulation, as explicitly specified in the MOEA, may result in a change in the characteristics of the solution sets [190].

Additionally, a continuous MOP must be discretized when formulated on any current computer system. This discretization is based on the hardware limitations of the computer system used. Since the resolution of the decision variables can have an effect on the solutions generated, this illustrates the importance of reporting both the genotype and phenotype resolutions used when attempting to solve an MOP on any real-world computer system. It must also be recognized that an MOEA may perform well at a given resolution but that does not guarantee "good" performance at other resolutions when applied to a given MOP.

Unconstrained MOPs are the class of MOPs most frequently referenced in the literature in terms of test suites and testing of MOEAs [31, 44, 114, 184, 207]. Unconstrained MOPs are those MOPs in which the k objective functions are solved simultaneously with no additional constraints. Unconstrained MOPs vary in levels of complexity but generally may be easier for an MOEA to search and solve than constrained MOPs since all of the search space consists of feasible solutions [31, 44]. Constrained MOPs are typically more difficult problems for MOEAs to attempt to find solutions to than unconstrained MOPs [31, 44, 46, 138, 141, 160, 213, 215]. Constrained MOPs may have multiple constraints on each of the k objective functions and these constraints may be strict equality and/or inequality constraints. The constraints, which can be linear or non-linear, restrict the feasible area of the search space, in some cases so that the percentage of feasible solutions is $<< 0.001\%$ of the total search space. It is important to understand the definitions of feasible and optimal solutions:

Definition 11 (Feasible Solution): *A feasible solution is a solution containing the choice of decision variable values that satisfies all of the constraints [159].* □

Definition 12 (Optimal Solution): *An optimal solution is a feasible solution that obtains objective function values as good as those of any other feasible solutions [159].* □

MOPs in which the feasible search space is greatly restricted due to constraints imposed on the problem may pose a challenge for an MOEA to generate any feasible solutions. This brings up an interesting dilemma. If a larger percentage of the search space is infeasible than feasible, *how does the algorithm handle this infeasibility? Additionally,*

how can the MOEA ever find a feasible solution besides by pure chance and the stochastic nature of EAs? The answer to these two questions lie in the ingenuity of computer engineers and computer scientists in constraint handling techniques. A limited number of MOEA researchers have addressed constraint handling issues and have proposed a variety of methods for implementation [31, 44, 46, 138, 141, 160, 213, 215]. These constraint handling techniques are described in detail in Section 2.8.

2.6 MOEA Algorithm Domain Formulation

An MOEA's complex structure can lead to confusion in discussing the algorithmic process that takes place. To prevent further inconsistencies in discussions of MOEAs, Van Veldhuizen [184, 190] refined Pareto terminology to clarify MOEA discussions. He stated at any given generation of an MOEA, a "current" set of Pareto optimal solutions (with respect to the *current* MOEA generational population) exists and is termed $P_{current}(t)$, where t represents the generation number. The set $PF_{current}$ represents the Pareto optimal solutions found within the current generation when analyzing the current population in isolation of the other generations and the entire search space. These solutions may not be optimal with respect to the entire search space, and hence may not be globally optimal but are the best solutions currently existing in the population. Because of the manner in which Pareto optimality is defined, $P_{current}(t)$ is always a non-empty solution set [184].

The sets $P_{current}(t)$, P_{known}, and P_{true} are solution sets of MOEA genotypes where each set's respective phenotypes form a Pareto front. We term the associated Pareto front, the solution set in phenotype domain, for each of these solution sets as $PF_{current}(t)$, PF_{known}, and PF_{true}. In the remainder of this document, the Pareto front is presented as the solution set since Pareto dominance criteria is used and the process of finding the solution sets takes place in the phenotype domain. Knowledge of the respective Pareto optimal set is implied as the search process manipulates genotypical information in order to generate "better" solutions. When using an MOEA to solve MOPs, the implicit assumption is that one of the following holds: $P_{known} = P_{true}$, $P_{known} \subset P_{true}$, or $PF_{known} \in [PF_{true}, PF_{true} + \epsilon]$ (PF_{known} is within some ϵ distance of PF_{true}) over some norm (Euclidean, RMS, etc.) [31, 184].

Solutions on the Pareto front represent optimal solutions in the sense that improving the value in one dimension of the objective function vector leads to a degradation in at least one other dimension of the objective function vector. The decision maker makes a tradeoff decision when presented with a number of optimal solutions for the MOP at hand, i.e., the Pareto front. There exists a difference in terminology between an acceptable compromise solution and a Pareto Optimal Solution [70]. The decision maker typically chooses only one of the associated Pareto optimal solutions, $\vec{u} \in \mathcal{P}^*$, as being the acceptable compromise solution, even though all of the solutions in $\mathcal{P}^*$ are nondominated. In reality the decision maker chooses a solution $\vec{u} \in PF_{known}$ as the acceptable compromise solution. The set $\mathcal{P}^*$ may only be realizable when utilizing decision variables at a comparable resolution to the hardware chosen that result in fitness values that remain within the hardware precision limitations. This is a result of our inability to represent real-valued solutions of infinite resolution on current computing platforms.

In the execution of an MOEA, the decision maker chooses a solution from the solution set provided by the MOEA, i.e., a solution from the set P_{known}, since finding the set $\mathcal{P}^*$ may not be achievable. The decision maker takes into account the human's preference in selecting a solution from the set of solutions P_{known}. The human preference factor encourages engineers and scientists to attempt to find a good distribution of points, as well as all of the points, in the Pareto optimal set (P_{true}, which is realizable on the computing platforms used) since the decision maker's preferences may not be known a-priori or may change over time. Even though the decision maker's preferences may change over time or all points may not be considered equally preferred in the decision maker's mind, the set P_{true} (and P_{known}) cannot change unless the objective functions and corresponding constraints are modified (or the MOEA is modified and executed again). Therefore one can conclude that if the MOEA finds a large number of, and a good distribution of, points along P_{known} (and ideally $P_{known} = P_{true}$), the decision maker can change their preferences and still select a good tradeoff solution (which may be an optimal solution).

2.7 MOEA Archiving

Most MOEAs use a secondary population, also referred to as an archive or an external archive, to store a cumulative set of nondominated solutions found throughout the generations of an MOEA [184, 186]. [2] Many MOEAs store points into an archive for post-processing while others actively use this set of points throughout the search process. Maintaining the good solutions generated throughout MOEA execution illustrates the need for an efficient method of storing points in the archive and a detailed description of the process of adding and removing points from the archive.

In many MOEA implementations, an MOEA stores the current Pareto front points, $PF_{current}(t)$, found with respect to the population of generation t into the archive at the conclusion of generation t. Different secondary population storage strategies exist; the simplest is to maintain a running archive of members from $PF_{current}$, to which the set $PF_{current}(t)$ is appended following each generation. At the conclusion of the MOEA execution, the archive contains the combination of all of the $PF_{current}$ sets found at the conclusion of each generation. The archive must be analyzed in order to determine which of the $PF_{current}$ members in the archive are members of the final set of solutions found by the MOEA, PF_{known}.

It is important to understand that each $PF_{current}(t)$ set generated and stored into the archive is optimal only with respect to the population members it was compared to in generation t. As additional $PF_{current}$ sets are included in the archive, equivalent, better, or even worse solutions may be present in these sets. Since the points in each $PF_{current}$ set most likely have not been compared to all of the other points in the archive prior to inclusion, a comparison between all of the points in the archive must be conducted. This comparison must be made in order to determine which of the points present in the archive are nondominated and belong in the final solution set generated by the MOEA, PF_{known}.

Another method of handling the inclusion of new points into the archive is to conduct the Pareto analysis prior to the addition of points to the archive. If the comparison is completed each time a point is considered for placement into the archive, the archive is

[2] A secondary population is referred to as an archive throughout the remainder of this document.

denoted as $P_{known}(t)$ and contains the known Pareto front points generated from generation 1 through generation t. This is not to be confused with PF_{known}, the known Pareto front found through termination of the MOEA. As nondominated points are found and included in the archive, the points present in the archive and the cardinality of the archive changes. Following each generation, t, the population members in $PF_{current}(t)$ and the archive are analyzed. This Pareto analysis is conducted to determine which of the combined members are elements of $PF_{known}(t)$ and should remain in the archive. The archive is modified to reflect only the set of nondominated members ($PF_{known}(t)$). At the conclusion of any generation in the MOEA search process, a set, $PF_{known}(t)$, exists explicitly defining the "best" or nondominated solutions generated by the MOEA. This strategy of continually analyzing the points considered for inclusion into the archive may be necessary if an MOEA is required to select members from $PF_{known}(t)$ throughout the search process.

At the conclusion of an MOEA search process, both archiving strategies result in the exact same solution set, PF_{known}. The difference in the archiving strategies lies in the point in time in which the processing to find the nondominated solution set takes place. Most MOEAs keep an archive so that "good" solutions found throughout the search process are not lost. An external archive allows the number of nondominated solutions stored to be greater than the population size. If an archive is not used, then the MOEA can only store a maximum number of nondominated solutions equal to the population size specified in the MOEA. An archive can use dynamically allocated memory so as to prevent a restriction on the cardinality of PF_{known}. At termination of the search process PF_{known} is presented as the final solution set. Additionally, $PF_{known}(0)$ is defined as the empty set ($\emptyset$) and PF_{known} alone as the *final* set of Pareto optimal solutions returned by the MOEA at termination [184, 219].

2.8 Constraint Handling in MOEAs

Many of the problems discussed in the literature are unconstrained MOPs. Unconstrained problems may be easier for MOEAs to solve as the feasible search space is not restricted and these MOPs may require less processing than constrained MOPs. This is due to the additional overhead of calculating the constraints, which can be substantial if

a highly complex, nonlinear constraint set is used. Soft constraints (inequality) or hard constraints (strict equality) may be imposed on the MOPs. Additionally these problems may be continuous or discrete, contain disjoint Pareto fronts or continuous Pareto fronts, be integer based or real-valued, etc. All of these factors that be considered in deciding how the MOEA should handle the constraints. With the increased use of MOEAs for solving MOPs, constrained MOPs are an area of great interest. This is especially true since there exists no one best MOEA method of handling constraints in all MOPs.

A number of constraint handling methods have been implemented in single objective optimization problems solved by EAs but much less development has been done with MOEAs. Additionally, discrete optimization problems with hard constraints are difficult problems to solve with MOEAs [213, 215, 216, 218]. This is especially true if the constraints drastically reduce the size of the feasible space. Three approaches to dealing with constraints and infeasible solutions are presented and include the discarding of the infeasible population members, penalizing the fitness values of the infeasible members and the repair of infeasible members.

One method, that is an extension from the single objective EA area, is discarding all of the infeasible members in the population, sometimes referred to as the death penalty method [140]. This appears to be an acceptable method but in fact does not always yield good results, dependent on the size of the feasible space in relation to the total size of the search space. If the size of the feasible space is extremely small with respect to the size of the infeasible space, an MOEA may have to discard whole populations of individuals prior to finding one feasible solution. In this situation, it is possible for the MOEA to execute for an extremely long time before generating even a single feasible solution. This method may be extremely ineffective and may generate solutions that exhibit a poor fitness. The goal of utilizing an MOEA is to generate solutions in an efficient and effective manner. In highly constrained MOPs, the discarding of infeasible solutions can be detrimental to the efficiency and effectiveness of the algorithm [216].

Another easily implemented method is to execute the MOEA without regard to the constraints and then post-process the solution sets to remove the infeasible solutions. This method assumes that through the search process that an MOEA finds a large portion of

feasible members even though the MOEA is attempting to solve a different optimization problem, the unconstrained MOP. An assumption is also made that the feasible points are nondominated with respect to infeasible points. If infeasible solutions dominate the feasible solutions, PF_{known} would not contain any feasible solutions. In order to obtain the feasible points found, the MOEA must store the entire population after each generation to an archive or check for feasible solutions to store after each generation. This method may be ineffective as the MOEA is attempting to solve the unconstrained MOP even though the researcher would like to obtain the solution(s) to the constrained MOP. However, this method does not require a large amount of overhead but assumes that the constrained and unconstrained fronts are relatively close to one another or identical. If the constrained and unconstrained fronts are not close to each other, one might expect the performance of the MOEA using this method to be unpredictable and the search process may be wasted.

Penalty functions have been used in the single objective area [140] to penalize potential solutions that are infeasible. Penalty functions degrade the fitness values of the infeasible population members, but still allow their BBs to remain in the population and potentially influence the final solution. Penalty functions are typically based on some type of distance measure in which a member that is further from the feasible region is penalized more than a member that is closer to the feasible region. Static and dynamic penalty functions exist. Static methods penalize the constraint violations an identical amount each time the constraint is violated. Dynamic penalties vary the amount of the penalty based on the number of generations that have been completed [140].

Michalewicz used the annealing concept of simulated annealing to penalize infeasible population members [138]. In this annealing penalty function, a temperature parameter is used to decrease the amount of penalty inflicted on infeasible members over time. This is similar to the annealing process used in simulated annealing in which the temperature is lowered over time. The lowering of the temperature increases the selective pressure and hence results in a lower probability of choosing infeasible solutions as the algorithm executes. This is the traditional exploration versus exploitation tradeoff. One would like the algorithm to explore as much as possible at the start and migrate to exploiting

as much as possible towards the termination of the algorithm and hence generate good solutions [140].

Adaptive or dynamic penalties are yet another alternative to the fixed decreasing penalty term used in the simulated annealing penalty function. Adaptive penalties can be implemented in different ways but one method is to adapt the penalty based on the feasibility of the population [140]. In this adaptive scheme, if the best members of the population remain feasible, then the penalty term is slowly decreased, whereas if the best members of the population are infeasible, then the penalty term is increased. The difficulty in using penalty functions for MOPs is in deciding which criteria to use in implementing a penalty function. Optimal parameter settings for penalty functions are sometimes difficult, if not impossible, to determine.

Another penalty technique involves the use of a modified binary tournament selection operation to compare two solutions. This operation results in the feasible solution always being selected [101]. In this technique, if both solutions are infeasible then the one "closest," in terms of a distance metric, to the constraint boundary is selected. If both solutions are feasible, then Horn's Niched Pareto MOEA is employed (discussed in Chapter III, Section 3.1.1.3).

Ray, Kang, Chye's penalty approach is yet another method to handle constraints in a Pareto fashion [160]. Ray, Kang, Chye's method is described as performing a nondominated check of the constraint violations. Three different checks are made of the entire population. In the first, a Pareto ranking is performed using the k objective function values and then storing these in a vector R_{obj}. A second Pareto ranking is performed using the J constraints without the objective function values and stored in the vector R_{con}. A feasible solution has zero constraint violations and thus obtains a R_{con} value of 1. The third Pareto ranking, R_{com} combines information from the other two rankings. The objective function and constraint violations are used together but there is no need for a penalty parameter as the criteria are compared individually in a domination check. Following the computation of all of the ranks, the feasible solutions with the best rank in R_{com} are placed into the new population. If there remains available space, crossover and selection are applied to the remaining solutions based on the rank R_{obj} for selection and R_{con} for crossover.

Deb presents an alternate approach to the constraint handling issue in MOEAs, presented in Definition 13 [46, 50]:

Definition 13 (***Pareto Based Constraint Handling Selection):***
In comparing two solutions, i and j:

1. *If both solutions, i and j, are feasible, choose the solution with the "better" value.*
2. *If solution i is feasible and solution j is not, choose solution i.*
3. *If neither solution, i or j, is feasible, but solution i has a smaller overall constraint violation, choose solution i.*

□

Deb's approach combines the concepts of Pareto dominance with that of constraint handling. This approach is conceptually straight forward as the selection process is similar to that of a Pareto based selection mechanism. The problem with this method appears when analyzing the third case, if neither solution, i or j, is feasible, but solution i has a smaller overall constraint violation, choose solution i. The issue that arises is in determining which of the members has the overall higher constraint violation. For example, consider an MOEA selected to solve a two objective constrained MOP with constraints. Assume that two population members are compared and they are both infeasible. If there are three constraints and both members violate two of the constraints, it is difficult to determine which member should be selected.

A problem also arises in a situation where the three constraints are of different orders of magnitude and resolution. If population member 1 violates the constraints by 3.40 and 1299.45 for constraints 1 and 2, and population member 2 violates the constraints by 1313.65 and 127.32 for constraints 2 and 3, one might assume that population member 1 is better since the sum of the violation is less. The problem is that population member 1 may be very close to violating constraint 3 but population member 2 is very far from violating constraint 1. It appears that population member 1 is preferred but overall population member 2 is further from a constraint violation due to its value with respect to constraint 1. Additionally, in terms of actual distances from the constraint, a value of 127.32 may

only be two "steps" away from the constraint boundary whereas a value of 3.40 may be 100 "steps" away from the constraint boundary due to the resolution of the decision variables and constraints. A step is defined as the number of feasible solutions that exist in a straight line from the constraint boundary to the population member in question at the given resolution. The situation described above can lead to an incorrect choice of the "better" population member. However, note that this method may be one of the easiest to implement with non-linear real-valued constraints where utilizing a mechanism to repair (mutate a population member until it is feasible) a population member is not possible. It is also possible to normalize the constraint values to obtain a better indication of differences in constraint violations. In [46], Deb compares his method, in Definition 13, to Ray, Kang, Chye's penalty approach and shows that his Pareto Based Constraint Handling Selection performs better on a limited test suite.

Laumanns, Thiele, Deb, and Zitzler propose an approach to solving MOPs through the use of ϵ approximations and the formation of ϵ-approximate and ϵ-Pareto front sets [122]. The authors provide the pseudocode for implementation of these concepts into an algorithm but never actually implement them. The idea is that if one can find an approximation of an area in the phenotype space that may move towards the Pareto front set, this approximated set would be more useful to provide the decision maker than just providing an enormous set of points (PF_{known}) to sift through. The approximation can then be used by the decision maker to isolate regions of the Pareto front to further explore and reduce the size of the Pareto front set. Additionally with an approximation, it is conceivable that through subsequent generations of the MOEA, PF_{true} can be found. ϵ-Dominance, the ϵ-approximate Pareto front Set and the ϵ-Pareto front Set are defined as [122]:

Definition 14 (ϵ-Dominance): *Let $f, g \in \mathbb{R}^{+^m}$. Then f is said to ϵ-dominate g for some $\epsilon > 0$, denoted as $f \succ_\epsilon g$, iff $\forall i \in \{1, \ldots, m\}$:*

$$(1 + \epsilon) \cdot f_i \geq g_i \tag{2.9}$$

□

Definition 15 (ϵ-approximate Pareto front Set): *Let $F \in \mathbb{R}^{+^m}$ be a set of vectors and $\epsilon > 0$. Then a set $F_\epsilon \subseteq F$ is called an ϵ-approximate Pareto front Set, if any vector $g \in F$ is ϵ-dominated by at least one vector $f \in F_\epsilon$, i.e.,*

$$\forall g \in F : \exists f \in F_\epsilon \text{ such that } f \succ_\epsilon g. \tag{2.10}$$

The set of all ϵ-approximate Pareto front Sets of F is denoted as $P_\epsilon(F)$. □

Definition 16 (ϵ-Pareto front Set): *Let $F \in \mathbb{R}^{+^m}$ be a set of vectors and $\epsilon > 0$. Then a set $F_\epsilon \subseteq F$ is called an ϵ-Pareto front Set of F if:*

1. *F^*_ϵ is an ϵ-approximate Pareto front Set of F, i.e., $F^*_\epsilon \in P_\epsilon(F)$, and*
2. *F^*_ϵ contains Pareto front Points of F only, i.e., $F^*_\epsilon \subseteq F^*$.*

The set of all ϵ-Pareto front sets of F is denoted as $P^*_\epsilon(F)$. □

Laumanns, et al., state that the concept of ϵ-Pareto front sets is not new [122]. The problem of reducing the number of points that the decision maker must analyze is one that is of great importance and looked at more extensively in the OR and decision making fields. ϵ-Pareto front sets may be useful in situations where the researcher is seeking to reduce the number of points presented to the decision maker. In difficult MOPs, many MOEAs do not find points that are close to PF_{true}. In these cases the ϵ Pareto front set concept may yield worse solutions than a standard Pareto Set implementation. The goal of the ϵ-Pareto front set approach is to reduce the number of points found, but this may not be a worthwhile goal for those MOPs where very few solutions are found or only solutions of "low" quality are found. Hence the ϵ-Pareto front set approach is useful once the search process is somewhat fine-tuned and a large number of solutions are generated. In other cases, this approach may be ineffective. In the context of this dissertation, the ϵ-approximate approach is not considered since a good distribution of, and a large number of, points on the front are the objective.

Repairing infeasible population members may be a useful alternative to the other constraint handling approaches if a large portion of the initial population is infeasible [140]. Repair is the process of mutating a population member so as to meet the constraints

imposed on the objective functions. As previously mentioned, the death penalty method could cause the removal of the entire population in a problem with a large infeasible space. In a problems with small feasible regions, a repair of the infeasible population members may be less time consuming then re-initializing the population in the hope of generating a feasible member.

The GENOCOP III system [22, 23, 138, 140, 141] repairs infeasible solutions through the use of a reference set. In this implementation, a set of feasible population members is stored. Once an infeasible solution is encountered, a feasible solution is selected from the feasible reference set. The system randomly generates points on a line between the infeasible and feasible solution until a new feasible solution is generated. The new feasible solution replaces the infeasible individual in the population. The disadvantage of this method is that it requires a feasible reference set to begin with. In problems where a feasible point is not known a-priori and the feasible space is much smaller than the infeasible space, this method could be inefficient. Other possibilities for repairing population members may be more problem and encoding specific. One example of this is the repair of population members when using integer based decision variables and strict equality constraints [216]. One can randomly choose decision variables to repair and adjust their value until the constraints are satisfied. It is also noted that the incorporation of additional problem domain information is useful in aiding the search process [74] and hence should be useful in aiding the repair process versus the use of a generic repair mechanism.

Each of the aforementioned methods for handling constraints have advantages and disadvantages. The use of any one method may be based on the characteristics of the MOP or the computational requirements to implement a particular method. To obtain effective results when utilizing constraint handling techniques, one must have a good understanding of the characteristics of the MOP and then make an intelligent choice as to the constraint handling method to use. In any MOP, a constraint handling method that takes into account problem domain information typically may yield "better" solutions than a method that is not tailored to the specific MOP or class of MOP. Additionally, the overhead of the constraint handling method is an important factor to consider. Many real-world highly constrained MOPs take a considerable amount of CPU time to calculate their objective

function values and the additional overhead of some of the mentioned constraint handling approaches may not be desired. The constraint handling approach that is used in this research effort involves analyzing the MOP and the potential constraint handling methods to make an intelligent decision as to the method that appears to offer the best potential to obtain good results. In situations where problem specific methods have been designed and tested, these problem specific methods are selected over the generic methods previously mentioned.

2.9 MOEA Ranking

Many MOEAs operate using a standard Pareto dominance analysis of the population but there are others that use a ranking scheme. Rank based fitness assignments were first proposed by Goldberg [74] and later used in a number of MOEAs [61, 64, 66, 68, 92, 176, 183, 213]. The specific rank assigned to each population member is dependent on the scheme used. The rank assignment may be based on the ranked front that each population member is assigned to [74], or the number of population members that dominate a specific point [64].

The Pareto ranking scheme proposed by Goldberg involves calculating the Pareto front, assigning members of the front rank 1, calculating the Pareto front of the remaining dominated members, assigning the new front the next incremented rank and repeating the process. At the conclusion of the ranking process, each population member has an assigned rank from 1 to the number of fronts found. Fonseca's method involves assigning population members a rank equal to the number of points that dominate it [64]. This scheme assigns nondominated points a rank of 0 and the dominated members a rank equivalent to the number of points that dominate them. Zitzler and Thiele use a more complex ranking scheme [213, 215]. Their method uses a secondary population and calculates the rank of a population member based off of the proportion of evaluated members that it dominates. However, this scheme does not assign all of the Pareto front points the same rank as is done in other ranking schemes. Once ranking is complete, all of the MOEA operators execute as they would in an MOEA that does not implement ranking. Analyzing the efficiency

shows that Goldberg's [74] and Zitzler's [183] method are the least efficient but an in-depth effectiveness study has not been completed.

2.10 MOEA Niching, Fitness Sharing, and Mating Restrictions

Niching and fitness sharing are MOEA operators for which a fair amount of research has been conducted. The main idea is that EAs tend to converge to a single solution over time. In the case of MOEAs, convergence to a single solution is undesirable as one would like to find as many points on the Pareto front, PF_{known} , as possible. Further, one would like these points to be evenly distributed across the front if one cannot generate the entire Pareto front. The concepts of niching and fitness sharing attempt to prevent an MOEA from converging to a single solution or a small, localized area of the Pareto front. Niching operators can operate in the genotype or phenotype spaces and attempt to restrict the generation of points within a certain distance of existing points [64]. Niching operators are also referred to as crowding operators and many MOEAs use these operators [47, 64, 69, 93, 94, 176, 184].

In an MOP with a large number of points contained in PF_{true} , where the Pareto front forms a continuous curve or surface, one would like to obtain a final solution set, P_{known} , consisting of points evenly spaced along the curve or surface. This is true of MOPs of varying characteristics. Even if an MOEA does not find all of the points in PF_{true} , or any for that matter, one would like the MOEA to generate a good distribution of points across the Pareto surface(s) found versus a tight cluster of points existing in only one area of the surface.

Niching can be accomplished through the use of a parameter, σ_{share}, which specifies how close solutions in $\mathbb{R}^n$ (genotype) or $\mathbb{R}^k$ (phenotype) can be [93, 94]. If an MOEA generates two points that are within this σ_{share} distance, one of these points is replaced in the population with another point that does not violate this distance constraint. A difficulty with implementing niching is in determining the value to use for σ_{share}. If knowledge exists as to the number of niches (areas of local optimum), n_i, that exist in the solution set for the MOP, this parameter can be set fairly easily, otherwise this setting is not so obvious.

In most real-world problems, the number of local optima is not known, and setting niching parameters is difficult.

Niching can also be used in conjunction with a selection operator to aid in obtaining a good distribution of points along the Pareto surface. In this method, the next generation's population is restricted to only solutions that meet the niching requirements. A distance metric $\mathcal{D}$ is used in order to determine how far apart population members reside in relation to other population members. The distance metric can be the Hamming Distance (first norm), the Euclidean distance (second norm) or the Max-based distance (∞-norm) [64] to name a few possibilities. A distance metric can be used in isolation or can be used with a niche count, n_i to allow up to a certain number of solutions to reside in a specific area of the space [94].

Fitness sharing is similar to niching in that its goal is to keep a good distribution of points along the front but completes this through a modification of the fitness values. The population members' are assigned fitness values based on the distance between members and the average fitness values of the entire population [93]. Integration of niching into the fitness function maintains the genetic material of the population members that would have been removed by other niching schemes but instead induces a type of penalty in the fitness value assignment.

Goldberg [74], and later Fonseca and Fleming [64], presented the use of mating restrictions to avoid competition between population members that are a large distance from each other. The mating restriction assumes that population members close to each other, defined by some Euclidean distance metric, are similar in the genotype domain. A parameter σ_{mating} is presented for mating restrictions. σ_{share} and σ_{mating} are useful in reducing the size of the Pareto front set while maintaining a uniform distribution of population members across the front.

Typical mating restriction operators disallow mating between individuals that are further apart than a predefined σ_{mate} value in terms of their phenotype distance from each other. While this is one idea of what a mating restriction should be, there are others. Some researchers feel that instead of generating distinct species by isolating the mating

between those of similar attributes, they should be preventing incest by allowing mating only between individuals that are different enough [31].

A few MOEA researchers have developed niching and fitness sharing variants for use in MOEAs with the goal of finding equidistantly spaced points on PF_{known}. Fonseca and Flemming [69] proposed the use of fitness sharing in conjunction with ranking. In this method, the fitness sharing occurs in the phenotype space and only between solutions with the same rank. They compare the distance between two points to an a-priori specified σ_{share} parameter. If the distance is less than the σ_{share} value, then the niche count is adjusted. Srinivas and Deb [176] implement a similar scheme but calculate the distance in genotype space instead.

Horn [93, 94] implements niching in an entirely different manner, referred to as equivalence class sharing. In his implementation, the selection operator is a modified Pareto based binary tournament selection. This operator selects a solution if it dominates other population members randomly selected and it dominates a randomly selected set of population members of size t_{dom}. He further states that fitness sharing only occurs if neither of the two solutions dominate each other. In this situation, the niche count is computed by counting the number of population members that are within a σ_{share} distance of the member in question. The solution containing the smaller niche count (has fewer members in its immediate area) is selected.

A few other variants can be found in the literature. One uses the ranking scheme in Goldberg [74] combined with a phenotype sharing, another uses both phenotype and genotype distances to calculate the niche count, and in yet another variant, ranking is implemented but the sharing is not restricted to solutions containing the same rank. Deb [47] presents a niching scheme, referred to as crowding that does not use a σ_{share} parameter. Other authors have proposed various methods that use different hypervolumes in determining if two points are close to each other

The selection of the σ_{share} parameter directly effects the performance of the niching method, as does the population size. There is no clear evidence to support the use of one niching scheme over another, although niching as a whole typically improves the results of

MOEAs [47, 64, 69, 93, 94, 176, 184]. Like many of the other operators discussed, there is no one best niching method identified in the literature.

2.11 MOEA Theory

In order to attempt and solve MOPs efficiently and effectively, an in-depth understanding of EAs, MOEAs, and their associated theory is necessary. With an understanding of the underlying theory, one has the ability to evaluate and compare the various approaches, or one's own approach for efficiency and effectiveness when applied to a variety of problems. Additionally, one may need an effective approach for only one class of problem or a generalized approach to solve a variety of problems with varying characteristics. This is a problem for researchers as the amount of developed theory for EAs is rather large but that for MOEAs is not quite as large.

One important concept, not given vary much attention in the MOEA community, is that of BBs and their applicability to the MOEA domain [26]. Van Veldhuizen's work is the only published research discussing the usefulness and advantages of explicitly finding and manipulating BBs in this domain [184, 190]. Using his work as an initial starting point into explicit BB-based MOEAs, this research effort concentrates on the concept of explicit manipulation of BBs. A large number of MOEA publications use implicit BB-based MOEAs as compared to explicit BB-based MOEAs as Van Veldhuizen and this research addresses [26]. The explicit use of BBs in an MOEA involves the MOEA identifying good BBs and combining those good BBs to yield good solutions to the MOP. The identification of BBs and their effect on the end results of the MOEA, P_{known} and PF_{known} , is an important analysis to make when attempting to advance the theory of MOEAs and MOP problem solving.

The theoretical foundation of MOEAs is based on single objective EA theory with the integration of Pareto concepts. While both of these areas have existed for some time now, the combination of the two fields leave a gap in some of the developed theory. Until recently, there has been little published theory related to MOEAs, convergence of MOEAs, convergence rates of MOEAs, population sizing of MOEAs, and parallel MOEAs. The major published theoretical analysis conducted for MOEAs is limited to publications by

Van Veldhuizen [184, 189], Deb [44], Hanne [85], Coello Coello, Van Veldhuizen and Lamont [31], Laumanns [122], and Rudolph [162, 163, 164, 165, 166, 167]. While several researchers have made contributions to MOEA theory [31, 44, 85, 163, 164, 167, 184], the field is not fully defined and additional theoretical contributions are needed and welcomed. The works mentioned contain MOEA theory in the form of definitions, corollaries, theorems, and proofs. Some researchers continue to develop their own MOEAs without a discussion of the underlying theory and without answering the basic questions, *Why is my MOEA better than another? And is my MOEA theoretically different than others?*

These questions are not easy to answer, and in some cases, the MOEA theory may not yet be developed enough to support such conclusions. This reinforces the need for theoretical research and advancement in the area of MOEAs. Additionally, the number of application based publications is much greater than that of theoretical publications as solving MOPs is the goal of many MOEA researchers [26, 184]. This is an area of concern as more and more researchers are utilizing MOEAs to solve real-world MOPs. The concern is warranted as many of the application based publications do not describe how the parameter values used in the MOEAs are selected or why certain operators are chosen over other operators. This is disconcerting as MOEA researchers are attempting to increase the efficiency and effectiveness of their MOEAs, yet some do not explain, theoretically, why certain modifications to their algorithms increase or decrease performance.

Much of the theoretical work related to MOEAs has to deal with that of MOEA convergence to the solution sets, P_{true} and PF_{true}. Rudolph [163, 166, 167] presents a convergence theorem for MOEAs using a stochastic transition matrix to define a chain of population movements and based on the generalization of finding optimal elements within a partially ordered set. He later presented a hypothesis and corollary for MOEA convergence when the population size is held constant and shows for a particular ES that the MOEA converges with probability 1 to P_{true}. Hanne presents another convergence theorem, that concentrates on continuous objective functions, through the use of cones, efficient sets, and other mathematics [84, 86].

The major areas of theoretical contributions have dealt with convergence to the Pareto front [85, 163], relations and partially ordered sets [162, 164, 166], and Markov

chain relations to MOEA convergence [167, 184]. There is limited work dealing with the area of fitness functions, and test problem generators which is discussed in Chapter IV. The major theoretical contributions published in the MOEA community, according to [31], are summarized in Table 2.3.

Table 2.3: MOEA Theory [31]

Researcher(s)	**Paper Focus**
Fonseca and Fleming [68]	MOEA mathematical formulations
Rudolph [163], Rudolph & Agapie [167]	MOEA convergence
Van Veldhuizen and Lamont [185]	MOEA convergence and Pareto terminology
Van Veldhuizen and Lamont [189]	MOEA benchmark test problems
Hanne (1999) [85]	MOEA convergence and Pareto terminology
Deb & Meyarivan [48], Deb et al. [52]	Constrained test problems
Rudolph [162, 164]	MOEA search under partially ordered sets
Ehrgott [58]	Analysis of the computational complexity of multiobjective combinatorial optimization problems
Rudolph [166]	Limit theory for EAs under partially ordered fitness sets
Laumanns et al. [121]	Mutation control
Hanne [86]	Convergence to the Pareto optimal set

Contributions to to the MOEA field in terms of Pareto ranking schemes, niching and crowding (σ_{share}), and restrictions on mating are more common [184]. Many of the currently available publications discuss various implementations of the concepts aforementioned and present mostly validation studies with few proofs. One reason for the limited number of proofs is the complexity associated with proving these conjectures and the overall difficulty in generalizing the concepts to all MOEAs. It is also noted that MOEA population sizing theory is undeveloped and is a focus of this research effort.

2.12 Summary

This chapter presents the EA background, MOEA background, and definitions necessary to understand the results of this research effort. An overview of Evolutionary Algorithms is presented to help the reader develop a good understanding of the evolutionary process. A discussion of other optimization approaches is presented to illustrate the utility of using an evolutionary approach to attempt and solve the types of problems that are concentrated on, real-world MOPs of varying levels of complexity. The BBH is presented to provide the reader with an understanding of what BBs are and how useful they can be in the attempted solving of MOPs.

A discussion of MOPs is followed by a detailed overview of Multiobjective Evolutionary Algorithms and their associated operators. The problem and algorithm domains are discussed to clarify the notation that is used in the discussion presented. Approaches for handling constrained MOPs are discussed to help the reader understand the rationale for using specific methods in the testing and results presented from this research effort. MOEA operators are discussed in detail in this chapter as these operators are an integral part of generating good solutions. This leads into the next chapter and a more detailed discussion of MOEAs and the metrics to evaluate their performance.

Theoretical analyses into the workings of MOEAs has been conducted and is summarized in this chapter. This aids a researcher in identifying areas in the MOEA field that have not been fully explored in terms of theoretical development. Deb states that "the construction of all these test problems has been done without much knowledge of how multiobjective GAs work" [43:25]. This statement illustrates the need for further theoretical research to be done in the area of understanding how and why MOEAs work. This is an area of active interest to the community for the reason that Deb states as well as the goal of fully understanding MOEAs. A fuller understanding of MOEAs can lead to new and potentially better MOEAs. Understanding the theory and the reasons why MOEAs work and have certain properties leads to even more advances in the field. It is noted that population sizing theory has not been developed for MOEAs as of yet. This research effort makes, what is seen as, the first contribution towards MOEA population sizing theory and is discussed in detail in Chapter VIII.

III. Contemporary MOEA Development

A focus of this research effort is on the advancement of explicit BB-based MOEA approaches. The advancement of such approaches and their associated operators aids in the development of an explicit BB-based MOEA that achieves increased efficiency or effectiveness over other approaches to solving certain classes of MOPs. The current chapter focuses on understanding MOEA approaches and metrics for evaluating MOEA results. An understanding of MOEAs and MOEA metrics leads to improvements to explicit BB-based MOEA approaches, operators, and theory which in turn leads to improvements to the efficiency and effectiveness of such MOEAs. Prior to discussing the MOEA used to conduct experimental data runs and validate associated concepts, a thorough discussion of contemporary and recently implemented MOEAs is necessary. While some may view contemporary MOEAs as just that, contemporary, and no longer used, that is not true in the MOEA community. Contemporary MOEA approaches and associated operators are still in use and appear in many recently implemented MOEAs.

This chapter presents a summary of some of the popular and highly referenced contemporary and recent MOEAs from the literature and a discussion of MOEA metrics [26]. Knowledge of contemporary and recent MOEA developments is important for MOEA researchers to recognize and understand. Such knowledge aids a researcher in obtaining a good understanding of the differences and similarities between various MOEA implementations and their associated operators. From this understanding one may develop new ideas and insight into increasing the performance of his/her own MOEA. This background discussion is provided as a resource to the reader to aid his/her understanding of the differences between various MOEA approaches and ideas for incorporation into new MOEAs. This understanding then leads to new developments or new breakthroughs in the advancement of the state-of-the-art for the MOEA community.

The discussion presented in this chapter decomposes the MOEA field into implicit and explicit BB-based MOEA implementations. Since the majority of MOEAs implicitly manipulate BBs, explicit BB concepts are not typically discussed in the literature. This chapter presents a clear and concise discussion of explicit BB-based concepts, and insight into the advantages of explicit BB manipulation. Further, the discussion present in this

chapter illustrates the limited amount of research conducted, previous to this effort, into explicit BB-based MOEAs.

The second half of this chapter presents a summary and discussion of MOEA metrics. The choice of a single or multiple metrics to use in the evaluation or comparison of the performance of various MOEA approaches is an important issue to the MOEA community. As a multitude of MOEA metrics exist to choose from, a researcher must be able to justify the selection of specific metrics for use. Additionally, a researcher must also be aware of the advantages and disadvantages of using certain metrics. A thorough summary of MOEA metrics is presented followed by an evaluation of each metric and a selected set of metrics recommended for use in the evaluation and comparison of MOEA approaches.

3.1 Multiobjective Evolutionary Algorithms

The first recognized MOEA is the Vector Evaluated Genetic Algorithm (VEGA) implemented by Schaffer and published in 1984 [169]. Following Schaffer's work, little research into MOEAs was published until the mid 1990s when a number of new, "improved" MOEAs began to appear in the literature [64, 93, 176]. The large gap in the time between Schaffer's work and the next published MOEA research may be attributed to the low computational power, by today's standards, of the computers available in the mid 1980s and the fact that the MOEA field was in its infancy. The lack of computational power and memory available in computers of the mid 1980s may have discouraged researchers from conducting additional research into MOEAs as the MOEA field was largely unproven and few publications existed. As computers became more powerful, more researchers began to experiment with the concept of MOEAs but processing power and memory constraints remained. Even in the early 1990s, EAs and MOEAs were typically implemented in a fashion so as to conserve as much memory as possible but many times remained time consuming algorithms to use. Many EA implementations performed bit manipulations and stored population members in unsigned integer representations to conserve valuable memory. The mGA and fmGA are two EAs that were engineered in a fashion to conserve as much memory as possible [77, 78]. Considering that EA researchers were conserving as much memory as possible to make it feasible to use such algorithms, it is understandable

that little MOEA research was conducted. MOEAs typically require additional computational power and memory as multiple functions are evaluated and their values stored. As computers with increased processing power and available memory became available in the 1990s, MOEA research became more feasible and additional MOEA implementations began to surface.

Initially, the first MOEAs were simple extensions to single objective EAs. Such implementations typically were extensions of the SGA [93, 170, 176]. Extending the SGA is the first logical step into implementing an MOEAs and there are a number of advantages of this approach, including the reuse of existing code as well as using code that has been thoroughly debugged through years of use. The interest in extending the SGA drew from what appears to be a goal of implementing an MOEA that is conceptually "easy" to understand, does not require substantial code modifications to an existing algorithm, and is efficient in memory utilization. More recently, new and innovative ideas in engineering MOEAs as well as improvements to existing MOEAs have been accomplished and resultant research has drifted from the SGA [29, 30, 115, 213].

Some authors have looked to classifying MOEAs and MOPs. Van Veldhuizen presents one of the most complete analyses of MOEA literature up to and including work completed in 1999 [184]. Recently Deb has published a book on MOEAs and real-world applications [44] and more recently, Coello Coello et al. [31] have published a book containing one of the most complete discussions on MOEAs. These publications address the theoretical aspects of MOEAs as well as discuss a number of highly referenced MOEAs, issues surrounding the use of MOEA metrics and test suites, and discuss the use of MOEAs in real-world applications.

The classification of MOEAs and MOPs is an important benefit to the community. A classification provides insight for researchers to determine which combination of MOEA operators and parameter settings perform efficiently and effectively when applied to MOPs of certain characteristics. While it is impossible to classify every MOEA and MOP, general information about the algorithms, their operators and the characteristics of the MOPs is useful information. A current listing of popular and highly referenced contemporary and recent MOEAs is presented in this chapter, classified by the method used in

the MOEA to manipulate BBs. Figure 3.1 presents a generic MOEA pseudocode reflective of the more common MOEAs found in the literature (e.g., Multi-Objective Genetic Algorithm (MOGA), Niched Pareto GA (NPGA), Nondominated Sorting GA I & II (NSGA-I & II), Strength Pareto EA (SPEA), Pareto Archive Evolutionary Strategy (PAES) and Multi-Objective messy GA I & II (MOMGA-I & II)). This generic MOEA structure aids researchers in understanding the structure and basic operators that comprise an MOEA. Additionally, a generic MOEA provides insight into the inner workings of an MOEA and identifies areas of improvement or areas where new operators or combinations of operators may increase MOEA performance.

```
Perform Population Initialization (Size P)
Compute Each Population Member's Fitness
                    (w.r.t. k functions)
Loop
    Perform Clustering/Niching/Crowding
    Execute EVOPs)
    Compute Each Population Member's Fitness
                    (w.r.t. k functions)
    Conduct Selection
    Generate PF_current (t); Update PF_known (t)
End Loop
Conduct Local Search (If Specified)
Generate PF_known ; Present As Final Solutions
```

Figure 3.1 Generic MOEA Pseudocode [31]

"Good" BBs are defined in this document as a combination of gene values (not necessarily in adjacent locations (loci) and typically a substring of the population member) that upon evaluation, tend to produce "good" solutions, independent of the other gene values in the string. An in-depth discussion of BBs is presented in Chapter II, Section 2.2. BBs and the Schema Theorem have been identified by a number of authors as being the key to identifying "good" solutions to optimization problems for a number of years [78, 79, 80, 201]. Hence this research effort recognizes and classifies MOEAs into two categories, those that implicitly manipulate BBs and those that explicitly manipulate BBs. A discussion of the MOEAs contained in each category follows.

3.1.1 Implicit Building Block Manipulating MOEAs. MOEAs that implicitly manipulate BBs are the most common form and most commonly referenced MOEA [26, 62, 120]. Implicit BB-based MOEAs do not attempt to explicitly identify the "good" BBs present in the population. The identification and manipulation of "good" BBs is based entirely on the idea that recombination and selection operators perform well enough such that MOEAs indirectly find these BBs and in turn generate "good" solutions. In many cases, implicit BB-based MOEAs find the good BBs present in the population but this is generally unknown to both the user and the MOEA. The following listing of implicit BB-based MOEAs encompasses a broad spectrum of MOEAs and includes MOEAs that are highly referenced in the literature. The information presented in this chapter about each MOEA can lead to new, innovative ideas and the design of new MOEAs with increased performance over existing MOEAs.

3.1.1.1 Vector Evaluated Genetic Algorithm. Schaffer created the Vector Evaluated Genetic Algorithm (VEGA) as a modification of the single objective SGA [64, 170]. The VEGA consists of the original SGA with a modification of the selection mechanism to reflect the multiple objective functions. A proportional selection mechanism is used during MOEA execution to create a subpopulation of individuals for each objective function. This means that the application of the VEGA to a k objective function MOP leads to the generation of k subpopulations of size N/k created by the algorithm, where N is the overall population size.

The approach used in the VEGA has the inherent disadvantage that the subpopulations typically produce individuals that are extremely fit in one of the k objective functions but typically not as fit in the other objective functions [170]. This result means that the population is split and may have the same effect as executing a single objective EA using a linear combination of the k objective functions, i.e., results in finding the extreme (end) points of the Pareto front [93]. While the approach used in the VEGA may not yield a large number or good distribution of points along the front, an advantage of the VEGA is that it yields the endpoints or extreme points of the curve or surface representing the Pareto front. The endpoints are identified by many MOEA researchers as being difficult points to

generate [100, 123, 215]. A single objective EA may generate the best solution in any one objective of an MOP and hence the VEGA may generate the extreme points of each objective. Some heuristics are used in the VEGA to attempt and prevent the Pareto optimal members of the population from converging only to the extreme points on the front [170]. These heuristics are integrated into the selection operator. The heuristic methods used in the VEGA are random preference selection, a method based on weighting nondominated points above dominated points for selection, and a method of promoting mating between subpopulations. Through testing each of the heuristic approaches, the authors note that the best results were obtained from the random preference selection [170].

3.1.1.2 Multiple Objective Genetic Algorithm. Carlos Fonseca and Peter Flemming created the Multiple Objective Genetic Algorithm (MOGA) to effectively handle MOPs using a generalized GA approach [64]. The MOGA uses a rank-based fitness assignment mechanism. A rank is associated with each of the population members according to the number of individuals that dominate it. All individuals on the Pareto front are assigned a rank of 1 whereas the remaining members are assigned a rank based upon the number of individuals that dominate them. The fitness assignment method used in the MOGA takes into account the rank of the population member and the average fitness value of the population. The process for assigning the fitness values is as follows: the population is sorted by rank; fitness values are assigned to individuals based on an interpolation of the best rank to the worst rank according to some specified function. Finally, individuals assigned the same rank receive an averaged fitness value. This ensures that all the population members of the same rank are sampled with an identical frequency. This information is used to maintain a constant global population fitness with an appropriate amount of selective pressure [64].

Additionally, the MOGA implements the concept of fitness sharing, also referred to as crowding, or niching, and uses a σ_{share} parameter. The σ_{share} parameter is referred to as the niche count and must be set carefully. Fonseca and Flemming use niching in the context of the phenotype domain to attempt to obtain a uniform distribution of points along the Pareto front.

A restriction on mating is also encoded into the MOGA. Goldberg originally stated that the purpose of restricted mating is to avoid excessive competition of population members that are not within a specified neighborhood [74]. Additionally, goals of the decision maker are integrated into the fitness function. Since the decision maker ultimately chooses a subset of solutions from the Pareto front, it is useful to a priori determine a region of the Pareto front that is of the most interest. This allows external preferences, in the form of problem domain information, to be inserted into the MOGA almost as a local search operator. The authors proceed to insert this modified fitness function, referred to as a "Goal Attainment Method" into the algorithm and further modify the ranking scheme to include "goal preferences." Goal preference is essentially a weighting approach. A further generalization is made to allow the decision maker preferences, or goals to be implemented through a utility function. The utility function serves the purpose of attempting to illustrate the method that the decision maker would use to select one Pareto front point over another.

The MOGA makes use of Gray Coded population members, two point reduced surrogate crossover and binary mutation. All of the nondominated points found are stored into an archive. This archive is used to determine the σ_{share} value to use in subsequent generations and the MOGA execution continues. Nondominated points are continually written to the archive until the algorithm reaches its stopping criteria.

3.1.1.3 Niched Pareto Genetic Algorithm. Around the same time as the development of the MOGA, Horn, Nafpliotis and Goldberg designed and implemented the Niched Pareto Genetic Algorithm (NPGA) [93]. The main differences between the the SGA and the NPGA lie in the selection mechanism and fitness assignment. The NPGA implements tournament selection with Pareto domination. During non-dominant tournaments, fitness sharing is implemented. To introduce a controlled, increased domination pressure over the standard tournament selection operator, the concept of a comparison set is used. The comparison set is a randomly chosen set in which each member is compared against two randomly selected members from the population. The dominance of the randomly selected population members with respect to all of the members in the comparison

set determines if that member is chosen for reproduction. A parameter, t_{dom}, is introduced to specify the size of the comparison set. In the cases where neither or both members are dominated by the comparison set, a crowding mechanism is implemented.

The crowding mechanism used is a fitness sharing method mentioned earlier for members on the Pareto front. This mechanism takes into account continuously updating fitness sharing along with a niche count parameter to try and obtain an even distribution of population members along the front. The niche count looks only at members in the partially filled next generation versus the current generation of members. In the case where neither or both population members are dominated by the comparison set, the niching mechanism is used to select the better candidate. This is done by selecting the member with the smallest niche count and is referred to as equivalence class sharing. The authors note success with this algorithm and the importance of the t_{dom} parameter on convergence.

3.1.1.4 Nondominated Sorting Genetic Algorithm. In 1994 Srinivas and Deb created the Nondominated Sorting Genetic Algorithm (NSGA) [176]. This algorithm is identical to the SGA with the exception of the selection operator. A similar concept is applied as was used in the NPGA and MOGA, use of a ranking scheme for selection. Prior to selection, the population is ranked based on each individual's classification of nondominated or dominated point. Subsequent to ranking, the nondominated points found are given a large dummy fitness value and are referred to as the first nondominated front. Since each of these members are Pareto optimal, they are assigned the same fitness value so as to give each member an equal chance at reproduction.

Sharing is used to maintain diversity in the population. Each nondominated member is shared with the dummy fitness value assigned to it. The sharing mechanism performs selection on what the authors refer to as a degraded fitness value. This degraded value is obtained by dividing the individual's original fitness by a number proportional to the number of individuals surrounding the member. After the sharing mechanism is executed, the remaining population members are processed in the same manner to identify the next front of nondominated points. The next set of points are assigned a lower dummy fitness

value than the minimum shared dummy fitness of the previous set. This effectively means that each front contains a "better" shared fitness value than the preceding front.

Once the whole population is partitioned into the various fronts, reproduction occurs using the dummy fitness values. Specifically, stochastic remainder proportionate selection is used. Individuals within the first front have the largest fitness value and have more copies placed into the population. This allows for a quicker convergence of the population while sharing helps to distribute the population over the front. Sharing is implemented by calculating a sharing function value between individuals within the same ranked front and the niche count is calculated by summing the sharing values of all the population members on a specific front. The shared fitness value is calculated by dividing the individual's dummy fitness value by its respective niche count.

3.1.1.5 Mendelian Multiobjective Simple Genetic Algorithm. The Mendelian Multiobjective Simple Genetic Algorithm (MMOSGA) created by Kadrovach, Michaud, Zydallis and Lamont takes a different approach than most MOEAs to attempt to solve complex MOPs [106, 107]. The MMOSGA uses the concept of dominant and recessive genes as part of the recombination procedure. The MMOSGA is based on the SGA with the data structure of the SGA augmented to reflect a diploid chromosome structure. This diploid structure is used in conjunction with a dominance table to determine which allele values pass on to the child population member. The recombination procedure selects two parents at random and uses a dominance table when there exist competing allele values to determine which of the parent's genes propagate to the child. As this process executes, the values within the dominance table are updated based on the allele values that generate the best/worst individuals, i.e., a reward/penalize method is implemented in the dominance table for each locus position and allele value. The dominance table aids in the convergence of the population towards the Pareto front.

A $(\mu + \lambda)$ approach is used in the selection process, where μ is the parent population size and λ is the child population size. An individual is chosen for mating by randomly choosing two individuals from the population and selecting the individual with the best fitness based on Pareto dominance. The individuals are then evaluated, the dominance

table updated, and the process repeats itself until the user specified number of child population members are created. There is no mutation operator present in the MMOSGA and the algorithm keeps an external archive of Pareto Optimal solutions.

The MMOSGA differs from almost all other MOEAs in its diploid chromosome structure and the dominance table that is used to add selective pressure to the population. The authors propose a number of modifications to the current algorithm to increase the efficiency and effectiveness of the algorithm. A thorough theoretical analysis of why the meiotic process with a diploid chromosome works was not conducted.

3.1.1.6 Micro Genetic Algorithm. Coello Coello and Pulido present a micro genetic algorithm in [29, 30]. The authors propose using a GA with a small population size and a re-initialization (also referred to as re-hope) process to prevent premature convergence. While the concept of micro-GAs is not new, this is the first application to MOPs. The MOEA executes by randomly creating a small population. The concept of memory is used in two variants, a replaceable and a non-replaceable memory. The non-replaceable portion of the memory does not change throughout the execution of the MOEA and is used to maintain diversity in the algorithm. The replaceable portion is modified after each "cycle" of the MOEA.

Any given cycle of the micro-GA takes a portion of the population from both sections of memory. The following operators are then used: tournament selection, two-point crossover, and uniform mutation. Following each cycle, two nondominated members are chosen from the population and compared to the external memory. If they remain nondominated, they are then included in the active population. This external memory acts as an archive for Pareto front points. The same two members are compared to two members of the replaceable memory and added if they continue to be nondominated. An adaptive grid approach, similar to that contained in [116], is used to keep diversity in PF_{known}. The authors state that they obtain better convergence rates than the NSGA-II and PAES, better values for a specific metric, and better timing but they need to improve the overall results of the micro-GA for constrained MOPs.

3.1.1.7 Niched Pareto Genetic Algorithm 2. To improve the performance of the original NPGA, Erickson, Mayer and Horn modified the NPGA to create the Niched Pareto Genetic Algorithm 2 (NPGA2) [61]. The main goal of this effort was to lessen the noise present in the Pareto domination tournament selection process. A degree of domination term is used as the ranking of an individual to make the Pareto ranking process deterministic. The degree of domination is a count of the total number of population members that dominate the member being checked. If the member is nondominated then the degree of domination (rank) is set to 0. Tournament selection is maintained throughout the algorithm with the ranking mechanism used for determining the winner of the tournament. Ties are resolved using fitness sharing and niche counts as in the original NPGA.

Following the ranking of population members, the deterministic tournament selection routine is used to select members for reproduction. A parameter, k, is used to specify the size of the tournament. k population members are randomly chosen. The member that has the lowest rank is the "winner" of the tournament. If there is no one "winner" of the tournament, a tie breaking scheme is applied. In the tie breaking process, as in the original NPGA, the niche count is used to determine the winner of the tournament and this count is based on the partially filled next generation of population members.

3.1.1.8 Nondominated Sorting Genetic Algorithm II. In 2000, Deb, Agrawal, Pratab, and Meyarivan suggested an improvement to the original NSGA, creating the Nondominated Sorting Genetic Algorithm II (NSGA-II) [46]. The first modification to the original NSGA implements a "better" book-keeping strategy in terms of creating the various nondominated fronts to decrease the execution time of the algorithm. The authors implement a density estimation and a crowding comparison operator as part of this modified algorithm. The density estimation operator estimates the overall population density surrounding a particular population member by calculating the average distance of two points on either side of the member, along each of the objectives. This calculation results in what the authors term a crowding distance which represents the largest cuboid that can be placed around a specific point without including any other points.

The crowding comparison operator seeks to produce a uniformly distributed Pareto front. This has a similar effect to the sharing implemented in the original NSGA except that a niching parameter is no longer necessary. The crowding comparison requires the calculation of each population member's non-domination rank and crowding distance. Once these values are obtained, a partial ordering is completed. This ordering states that if two solutions of differing non-domination ranks are compared, the lower ranked solution is chosen. Otherwise, a comparison of two solutions of the same front results in a choice of the solution with a larger cuboid, i.e., a solution whose neighbors are further away. The remainder of the algorithm is the same as the original NSGA.

3.1.1.9 (1 + 1) - Pareto Archived Evolution Strategy. The (1 + 1) - Pareto Archived Evolution Strategy ((1 + 1)-PAES) is another popular MOEA cited in the literature [115]. (1 + 1)-PAES uses a single parent and single child to search the space. Knowles and Corne introduce what they refer to as a simpler approach to Multiobjective Optimization (MOO). PAES is a (1 + 1) local search based evolutionary strategy using a population of size one and a "history" of previously found solutions. PAES does not use crossover but obtains "good" solutions through the exclusive use of a mutation operator. The authors note that PAES may be a good baseline approach to compare to other population based MOEAs. The motivation for engineering PAES was based on the fact that in many cases a local search algorithm, for example simulated annealing or tabu search, performs much quicker than the population based GA approaches for single objective problems [115]. PAES attempts to bring a local search mechanism to the multiobjective optimization world.

The main objectives of PAES are to conduct local search operations and to handle all Pareto optimal solutions in the same manner. PAES begins by creating a single individual and evaluating it with respect to all of the objective functions. This evaluation function is a dominance based evaluation. The population member is then copied and mutation is applied. The new, mutated population member is compared to the parent and the nondominated member of the two is selected. If neither solution dominates, then the new member is compared with the previously archived solutions. Since this approach restricts

the population to only contain a single parent and child, if one solution still is not favored over the other, the solution in the least crowded area is selected.

The archive keeps a current population of Pareto points found and is used to help guide the selection mechanism by providing the current Pareto front found. The archive allows (1 + 1)-PAES to have some pressure towards finding a better front as time goes on. The archive also has an a priori specified maximum size that allows the user to specify how many solutions are desired. In order to obtain a good distribution of points along the front, a k-dimensional grid crowding mechanism is implemented in the phenotype space, where k is the number of objectives. This grid is used in a different fashion than in other crowding or niching methods. In PAES, once a nondominated solution is found and is ready to be placed into the full archive, it replaces a member in the archive with the highest grid count as long as its own grid count is lower.

3.1.1.10 ($\mu + \lambda$) - Pareto Archived Evolution Strategy. Modifications to the original (1 + 1)-PAES algorithm produced two variants: the $(1 + \lambda)$-PAES and the $(\mu + \lambda)$-PAES, which are both referred to as $(\mu + \lambda)$-PAES [116], where μ is the size of the parent population and λ is the child population size. These varieties of the original PAES MOEA are population based schemes that continue to use only the mutation operator (no crossover).

The selection mechanism is modified as there are λ potential solutions to choose for the next current solution(s). This is accomplished through a fitness assignment that is based on the grid location of the member and a comparison to the archive. Each of the population members are compared to the archive and assigned a dominance value based on whether the member dominates a solution in the archive or is dominated by that solution. The members are then assigned a fitness based on the dominance value assigned. Members of the same dominance value receive a fitness based on the number of population members in that particular grid location.

The archive is updated in the same format as the original PAES MOEA. The λ potential solutions are generated through a mutation of members chosen via a binary tournament selection of current solutions. The $(1+\lambda)$-PAES and the $(\mu+\lambda)$-PAES approaches

are compared to $(1+1)$-PAES, the NPGA, and the NSGA. The authors state that $(1+1)$-PAES appears to be the quickest and most reliable approach [116]. $(1+\lambda)$-PAES does not do as well as the original PAES and $(\mu+\lambda)$-PAES is a little better than this variant. Overall the results indicate that $(1+1)$-PAES performs well when compared to other MOEAs on the limited test suite and metrics used.

3.1.1.11 Memetic PAES. Memetic PAES (M-PAES) is a population based variant of the original PAES algorithm [115] developed by Knowles and Corne to include the use of recombination operators [112]. M-PAES is based on the local search PAES but includes the use of a population along with a crossover operator to recombine local optima. Mutation is used as in the original algorithm. A difference from the original PAES algorithm is the use of a second archive for Pareto front solutions. One of the archives is used as a global archive, referred to as G and the other as a local archive for comparisons referred to as H. At the start of each local search, archive H is emptied and filled with solutions from archive G that do not dominate the current solution c. At this point H is continually used as a comparator while G is updated to reflect the current global front found.

The local search mechanism is identical to the procedure used in PAES, with the exception that the termination criteria is modified. Termination now occurs once the specified maximum number of local search operations occur, *l_opt*, or if the specified maximum number of local search failures, *l_fails*, are achieved. The *#fails* parameter is initialized to 0 and is incremented each time a mutated member is dominated by the current solution and is returned to 0 once a mutated member dominates the current solution, i.e., *#fails* effectively counts the number of mutation moves that are detrimental in between improving moves. Once *#fails* $\geq$ *l_fails*, the stopping criteria has been met.

Recombination occurs through the random selection of two parent population members from the population, prior to the local search mechanism being employed, and the global archive. A child is accepted if it is not dominated by any members of the global archive or in a less crowded region than one or more of its parents. The size of the crowding region is user specified. This occurs until a child is accepted or a specified number of

attempts are exceeded. In the second case, binary tournament selection is used to put a member of the global archive into the population.

3.1.1.12 The Strength Pareto Evolutionary Algorithm. The Strength Pareto Evolutionary Algorithm (SPEA) was developed by Zitzler and Thiele [213]. This algorithm incorporates many of the concepts of previous MOEAs into one MOEA, the SPEA. SPEA keeps an external archive of Pareto optimal solutions, uses Pareto dominance concepts in the fitness assignment scheme and uses a clustering concept to reduce the number of non-dominated solutions that must be stored while maintaining the structure of Pareto front. The fitness assignment scheme is unique in that it is designed to use the archive to determine the fitness of any given solution, whether it is Pareto optimal or not. The selection mechanism uses all of the solutions in the Pareto archive and niching is implemented without distance measurements. A standard binary tournament selection operator is used with a standard mutation operator.

Individuals are evaluated based on the number of Pareto points that dominate them or are equal to them (the authors refer to this as *covering* the points). A different approach to niching is presented in Zitzler et al. [213]. This approach makes use of Pareto dominance to keep niches. The goal of this approach is to maintain a uniform distribution of Pareto front members that cover the same number of individuals as any other Pareto front member. The Pareto archive contains all of the Pareto Optimal points found up to the current point in time. This ensures no loss of Pareto points found and further imposes no restrictions on the size of the archive.

The algorithm begins by assigning each member of the population a strength value, s. This is a real value proportional to the number of population members that it covers and is also the fitness of the individual. In the next step, the individuals are ranked according to the calculated strengths, i.e., the fitness of a population member is calculated by taking the sum of the fitnesses of all the archived Pareto front solutions that cover it and adding one to this value. In the event that all of the Pareto solutions have the same strength value, the fitness is calculated by taking the sum of the number of Pareto points that cover it. If

the population is unbalanced, the Pareto front points that cover the "least" fit individuals are assigned the greatest strength.

The size of the Pareto archive is unrestricted, therefore a clustering method is applied to reduce the size of this set while maintaining the original attributes of the Pareto archive. The general idea of clustering is to partition the archive of size p into q groupings, where $q < p$ and all of the points within any of the q groups have the same characteristics. The SPEA implements a hierarchical clustering method referred to as the average linkage method.

The clustering approach begins by making each element of the initial Pareto archive a cluster. Following this, two clusters are chosen via a distance measurement to combine into one cluster. The distance is calculated as average distance between pairs of individuals across the clusters. At the completion of the cluster approach, the new Pareto archive consists of the centroid members of each cluster. The authors show favorable results compared to other MOEAs.

3.1.1.13 The Strength Pareto Evolutionary Algorithm 2. Zitzler, Laumanns, and Thiele modified the SPEA and released the "improved" SPEA2 in 2001 [211]. In modifying the original SPEA, a fine-grained fitness assignment method, a density estimation technique, and a new archiving method are introduced. The "new" fitness assignment method is similar to that of the NSGA-II, PAES, PESA, and others. The fitness assignment for a population member is based off of the number of individuals the population member dominates and the number that dominated it. Additionally a k-th nearest neighbor mechanism is present to differentiate between individuals which have the same fitness values. Modifications were made to the archival update process to keep the size of the archive constant and because of this prevent the removal of "end points" from the archive.

The authors state that the incorporation of these changes provides better results than the original SPEA over a limited test suite and metric set. When compared to the NSGA-II and PESA, the SPEA2 and NSGA-II yield the best overall performance.

3.1.1.14 Pareto Envelope-Based Selection Algorithm. The Pareto Envelope-Based Selection Algorithm (PESA) was developed by Corne, Knowles and Oates as an MOEA that controls the selection and diversity of solutions via a hyper-grid scheme [34]. PESA introduces some of the ideas presented in the SPEA and PAES from Sections 3.1.1.10 and 3.1.1.12. The PESA MOEA uses a small population size and maintains an external archive of PF_{known} solutions. Niching is accomplished through an implicit hyper-grid division of the phenotype space, and the selection mechanism is based on crowding. Crossover and mutation are both implemented in the PESA algorithm. Additionally two other parameters must be set, the size of the population and the maximum size of the archive.

Population members enter the archive through Pareto dominance checks so that only nondominated points remain in the archive. Crowding is implemented through a hyper-grid structure that divides the phenotype space in hyper-boxes. The number of points within each hyper-box is referred to as the "squeeze" factor and is used for selective fitness calculations and archive updating. Tournament selection is used with the squeeze factor to select members from the archive for directing the search in subsequent generations. The authors obtain favorable results in comparing their algorithm against SPEA and PAES.

3.1.1.15 Pareto Envelope-Based Selection Algorithm II. Corne et al. improved the PESA algorithm and developed the PESA-II [33]. PESA-II differs from PESA in a few areas. First the selective fitness assignment has been modified to assign a value to the hyperbox in the phenotype space instead of directly to each individual. As hyperboxes are chosen, individuals are randomly selected from them. The authors state that this method of selection (referred to as region based selection) is better at distributing the members in the archive more evenly across the Pareto front. All of the other operators remain the same from PESA to PESA-II. The authors compare PESA-II to PAES, SPEA and PESA to illustrate "good" results are obtained with this new selection method over a limited test suite and set of metrics.

3.1.1.16 The Multiple-Objective Genetic Local Search Algorithm. A local search based MOEA referred to as the Multiple-Objective Genetic Local Search Algorithm (MOGLS) was implemented by Jaszkiewicz [100]. This algorithm is exclusively applied

to the Modified Multiobjective Knapsack Problem (See Chapter IV, Section 4.4.1) and compared to other MOEAs applied to this problem. The Genetic Local Search (GLS) algorithm, also referred to as a memetic or hybrid GA, hybridizes recombination operators with other local heuristics. These types of algorithms have been shown to find the best known solutions to a number of problem domains to include the traveling salesman problem, graph coloring problem, and quadratic assignment problem, hence the motivation of the authors to use it for the multiobjective knapsack problem [100].

The author states that the goal of multiobjective metaheuristics is to generate good approximations to the nondominated set. In order to understand the process used in MOGLS, some terminology must be presented. The approximation to the nondominated set is defined as:

Definition 17 (Approximation to the Nondominated Set): *A set A of points (and corresponding solutions) such that $\neg\exists \mathbf{z}^1, \mathbf{z}^2 \in A$ such that $\mathbf{z}^1 \succ \mathbf{z}^2$ i.e., set A is composed of mutually nondominated points.* □

Point $\mathbf{z}^1$ dominates $\mathbf{z}^2$ is represented by $\mathbf{z}^1 \succ \mathbf{z}^2$ and the point $\mathbf{z}^*$, composed of the best attainable objective function values is called the ideal point.

Definition 18 (Ideal Point): $z_j^* = max\ \{f_i(\mathbf{x})\mathbf{x} \in D\}, \qquad j = 1, \ldots, J$. □

Definition 19 (Approximation of the Ideal Point): $\mathbf{z}^{**}(A)$ *is an approximation of the ideal point based on set A, i.e., $z_j^{**} = max\ \{z_j|\mathbf{z} \in A\}, \qquad j = 1, \ldots, J$.* □

The author further defines:

Definition 20 (Range Equalization Factors): $\pi_j = \frac{1}{R_j}, j = 1, \ldots, J$ *where R_j is the approximate range of objective j. The range is calculated by taking the minimum and maximum values from the set D, A, or the nondominated set and can also be estimated through using objective function values in a relaxed problem.* □

Normalized objective function values are defined as objective function values multiplied by range equalization factors.

Definition 21 (Weighted Linear Scalarizing Functions): $s_l(\mathbf{z}, \Lambda) = \sum_{j=1}^{J} \lambda_j z_j = \sum_{j=1}^{J} \lambda_j f_j(\mathbf{x})$. *Each weighted linear scalarizing function has at least one global optimum (maximum) belonging to the set of efficient solutions.* □

Definition 22 (Weighted Tchebycheff Scalarizing Functions): $s_\infty(\mathbf{z}, \mathbf{z}^0, \Lambda) = max_j\{(\lambda_j(z_j^0 - z_j)\} = max_j\{\lambda_j(z_j^0 - f_j(\mathbf{x}))\}$ *where* z^0 *is a reference point and* $\Lambda = [\lambda_1, \ldots, \lambda_J]$ *is a weight vector such that* $\lambda_j \geq 0 \; \forall j$. □

Minimization of the weighted Tchebycheff scalarizing function corresponds to a min-max problem. The author states that for each efficient solution $\mathbf{x}$, there exists a weighted Tchebycheff scalarizing function s such that $\mathbf{x}$ is a global optimum (minimum) of s [100].

Definition 23 (Normalized Weight Vectors): *Weight vectors that meet the conditions* $\forall j \lambda_j \geq 0, \; \sum_{j=1}^{J} \lambda_j$. □

Jaszkiewicz also notes that all weighted Tchebycheff and all weighted linear scalarizing functions have optima in the nondominated set. Further, he states that finding the whole nondominated set is equivalent to finding optima of all Tchebycheff and all weighted linear scalarizing functions. Hence the goal of multiobjective metaheuristics is reformulated as simultaneous optimization of all weighted Tchebycheff or all weighted linear scalarizing functions (it is enough to consider scalarizing functions with normalized weight vectors) [100].

In the application to the MMOKP MOP, MOGLS simultaneously optimizes all weighted Tchebycheff or all weighted linear scalarizing functions with normalized weight vectors by random choice of a scalarizing function optimized in each iteration [100]. Each iteration of MOGLS is defined as a single recombination of a pair of solutions and application of a local heuristic, taking into account the value of the current scalarizing function. MOGLS attempts to improve the value of a randomly selected scalarizing function.

MOGLS proceeds to initially generate a set of feasible solutions. Solutions are then selected based on being best on a scalarizing function and then recombination and mutation operators generate new solutions. A local heuristic is applied to the generated solution and the process repeats. The outcome is an approximation to the nondominated set

that contains all potentially nondominated points. In this method a repair procedure is used that differs from the common procedure of removing items in increasing order of the profit/weight ratio. In the modified procedure, items are removed that locally decrease the weights in the knapsacks at the lowest cost of the current scalarizing functions.

The results of MOGLS are compared to a number other MOEAs by allowing MOGLS to simultaneously optimize weighted Tchebycheff functions representative of the MMOKP objectives. However, the MMOKP MOP formulation is modified to allow the decision variables to take on continuous values between 0 and 1. This new formulation of the MMOKP problem is referred to as the Relaxed multiple-objective 0/1 knapsack problem. As stated in Chapter IV, Section 4.4.1, this formulation is not the correct formulation of the multiobjective knapsack problem. While the author attempts to make a fair comparison of MOGLS to other MOEAs, one must question this comparison as MOGLS is applied to a different formulation of the MMOKP MOP.

3.1.2 Explicit Building Block Manipulating MOEAs. MOEAs that explicitly manipulate BBs are the least common form of EA and MOEA as illustrated by the limited number of publications of this class of MOEA [26, 62, 120]. Explicit BB-based MOEAs explicitly look for "good" BBs throughout execution of the MOEA and manipulate those BBs to generate "good" solutions. Researchers who have designed explicit BB-based MOEAs typically are knowledgeable in the theory of what is happening "behind the scenes" of the MOEA.

Explicit BB-based MOEAs are designed around the Schema Theorem and the concept that "good" BBs are vital to solving optimization problems [201]. A detailed discussion of BBs and the BBH is presented in Chapter II, Section 2.2. This class of MOEAs attempt to find the "good" BBs for the particular MOP at hand and exploit those BBs in order to generate "good" solutions to the problem. These MOEAs typically have a higher complexity in terms of understanding how they operate and hence many researchers do not use them [26]. A discussion of the explicit BB-based MOEAs can lead to new, innovative ideas and the design of new MOEAs with increased performance over existing MOEAs as well as an increased interest in this class of MOEAs.

3.1.2.1 Multiobjective Messy Genetic Algorithm. The Multiobjective Messy Genetic Algorithm (MOMGA) is an explicit BB-based MOEA, created by Van Veldhuizen and Lamont, based off of the concepts of the single objective messy Genetic Algorithm (mGA) [78, 184]. The authors of the MOMGA took the building block ideas of the mGA and extended them into the multiobjective domain. The MOMGA is considerably different from all of the other MOEAs discussed in the literature in that it is the only algorithm to explicitly manipulate BBs. The MOMGA consists of three phases, the initialization, primordial, and juxtapositional phases.

In the initialization phase, the MOMGA completes a total enumeration of all user specified BB sizes in initializing its population. The population is generated through a process of analyzing the specified BB size, generating all of the BBs necessary, and evaluating the objective function values for the strings. Therefore the starting population size is based upon the BB size, string length and cardinality of the alphabet used. BBs are underspecified, meaning that they are partial strings, having some bit values, alleles and loci, missing. The MOMGA is a variable string length MOEA that makes use of competitive templates to evaluate population members missing bits or allele values. The templates are fully specified individuals that do not get modified throughout one iteration of the algorithm. The "steady-state" templates allow for a good comparison basis when evaluating population members. The initial population members are "overlayed" on top of the competitive templates to determine their fitness values. In this overlay process, only bits that are missing from the population member are copied from the selected template for evaluation. The copied bits do not remain with the population member subsequent to its evaluation.

The MOMGA does not implement mutation. Following the initialization phase, the primordial phase executes. The number of generations of this phase is dynamically determined based on the proportion of good BBs in the population. During this phase, binary tournament thresholding selection is used along with a population reduction method to enrich the population with a good BBs.

The last phase, the juxtapositional phase, is used to perform recombination. This phase consists of the use of a cut-and-splice operator along with tournament thresholding

selection. The cut-and-splice operator is a crossover operator for variable length strings and its continued use builds up strings from their underspecified BB size to the fully specified string length. Cut and splice randomly chooses a point to cut a string, then subsequently splices the string with another cut string. Repeating this process allows the population members to grow in length when used in conjunction with a large enough splice probability. During the intermediate evaluations, competitive templates are used to evaluate the underspecified strings. The templates are fully specified population members that are not modified until the completion of the juxtapositional phase. Following the juxtapositional phase, the competitive templates are updated with the best individuals found with respect to each objective function and the process repeats itself utilizing the next user-specified BB size. In the case of an overspecified individual, or one that specifies an allelic value for a particular gene multiple times, a first encountered method is used meaning that the first allele value encountered for a specific locus (gene), when scanning the bit string, is the one used.

During all of the selection routines, Pareto dominance based selection is used. In order to expand the mGA to multiple objective functions, there are k competitive templates used for a k objective function MOP. The template is randomly chosen when evaluating a population member so as to avoid convergence to one of the objective function solution sets. Additionally niching is implemented based off of the method used by Horn in the NPGA. The MOMGA achieves favorable performance when compared to a limited set of other MOEAs.

3.1.2.2 Multiobjective Messy Genetic Algorithm II. The Multiobjective Messy Genetic Algorithm II (MOMGA-II) is designed and implemented through this research effort. The MOMGA-II is the second known explicit BB-based MOEA. The details of this algorithm are discussed in Chapter VI.

Table 3.1 presents a summary of the aforementioned MOEAs, along with the coding method used by each, their Evolutionary Operators (EVOPs), fitness assignment scheme, niching, and population size.

Table 3.1: MOEA Characteristics

Algorithm	**Coding**	**EVOPs**	**Fitness Assignment**	**Sharing and Niching**	**Population**
MMOSGA [107]	Binary	Crossover, implicit mutation through the dominance table, Tournament Selection, External Archive	Standard fitness assignment	None	Randomly initialized; $N = 100$
micro-GA [29, 30]	Binary	2 Point Crossover, uniform mutation, Tournament Selection, Memory (similar to the concept of an archive)	Standard fitness assignment	Phenotypic (adaptive grid)	Randomly initialized; $N = 4$, External memory $N = 100$, Population memory $N = 50$
MOGA [64]	Gray	2 Point Surrogate Crossover and Mutation ($p_c = 1$, $p_m = \frac{1}{0.042}$), Tournament Selection, Mating Restriction Utilized	Linear interpolation using Fonseca's [64] Pareto ranking	Phenotypic (σ_{share} - Fitness)	Randomly initialized; $N = 80$
MOMGA [184]	Binary	"Cut and splice" ($p_{cut} = 0.02$, $p_{splice} = 1$), no mutation, Tournament Selection, External Archive	Tournament ($t_{dom} = 3$)	Phenotypic (σ_{share} - Domination)	Randomly initialized; Deterministic Size

Table 3.1: (continued)

Algorithm	**Coding**	**EVOPs**	**Fitness Assignment**	**Sharing and Niching**	**Population**
MOMGA-II [217, 219]	Binary	"Cut and splice" ($p_{cut} = 0.02$, $p_{splice} = 1$), no mutation, Tournament Selection, External Archive	Tournament ($t_{dom} = 3$)	Phenotypic (σ_{share} - Domination)	Randomly initialized; Deterministic Size
MOGLS [100]	Binary	Crossover, Mutation, use of a Temporary Elite Population	Tchebycheff and weighted linear scalarizing functions	None	Random, $N = 150 - 350$
M-PAES [115]	Binary	Crossover, Mutation, Tournament Selection, External Archives (Local and Global)	Standard fitness assignment	Phenotypic (Crowding)	$N > 1$
NPGA [93]	Binary	Crossover, Mutation ($p_c = 1$, $p_m = \frac{1}{0.042}$), Tournament Selection with comparison set (Pareto ranking)	Tournament ($t_{dom} = 5$)	Phenotypic (σ_{share} - Domination)	Randomly initialized; $N = 100$
NPGA2 [61]	Binary	Crossover, Mutation ($p_c = 1$, $p_m = \frac{1}{0.042}$), Tournament Selection with comparison set (Deterministic Pareto ranking)	Tournament ($t_{dom} = 5$)	Phenotypic (σ_{share} - Domination)	Randomly initialized; $N = 50, 100$
NSGA [176]	Binary	Crossover, Mutation ($p_c = 1$, $p_m = \frac{1}{0.042}$)	"Dummy" fitness using Goldberg's [78] Pareto ranking	Genotypic (σ_{share} - Fitness)	Randomly initialized; $N = 100$

Table 3.1: (continued)

Algorithm	**Coding**	**EVOPs**	**Fitness Assignment**	**Sharing and Niching**	**Population**
NSGA-II [46]	Real Valued	Simulated Binary Crossover (SBX), Real Parameter Mutation, Tournament Selection based on niching	Dominance Based Fitness Assignment, Pareto ranking	Phenotypic (Crowding)	Randomly initialized; $N = 100$
PAES(1+1) [115]	Binary	No Crossover, Mutation, Niching Based Selection, External Archive (Fixed Size)	Dominance Based fitness assignment	Phenotypic (Crowding based on a grid)	$N = 1$
PAES($\mu + \lambda$) [116]	Binary	No Crossover, Mutation, Niching Based Selection, External Archive (Fixed Size)	Dominance Based fitness assignment	Phenotypic (Crowding based on a grid)	$N = \mu$
PESA [34]	Binary	Crossover, Mutation, Tournament Selection from archive, External Archive (Fixed Size and selection based on degree of dominance used in the archive)	Selective fitness assignment	Phenotypic (Crowding based on a hyper-grid)	$N = 10$
PESA-II [33]	Binary	Crossover, Mutation, Region Based Tournament Selection from archive(implicit phenotype crowding), External Archive (Fixed Size)	Selective fitness assignment	Implicit Phenotypic (Through region based selection and grid)	$N = 10$

Table 3.1: (continued)

Algorithm	**Coding**	**EVOPs**	**Fitness Assignment**	**Sharing and Niching**	**Population**
SPEA [213]	Binary	Crossover, Mutation, Tournament Selection, External Archive, Pareto Ranking	Dominance based fitness assignment (based on external archive only)	Phenotypic (density based niching)	$N = 100$
SPEA2 [211]	Binary	Crossover, Mutation, Tournament Selection, External Archive (Fixed Size), Pareto Ranking	Dominance based fitness assignment (based on external archive and current population)	Phenotypic (density based niching)	N = 250-2 objective, N = 300-3 obj, N = 400-4 obj
VEGA [170, 64]	Binary	Crossover, Mutation, Subpopulation based	Standard fitness assignment	None	Randomly initialized; $N = 30$

3.1.3 MOEA Summary. This discussion of popular and highly referenced contemporary and recent MOEAs provides a historical perspective for the reader in terms of MOEA development from the first MOEA, the VEGA to recent MOEA developments like the NSGA-II, NPGA 2, SPEA 2, MOMGA-II, and others. The interested reader is referred to [26], a continuously updated website of MOEA references for new additions.

The question of *Which is the best MOEA to use in solving MOPs?* has no one answer. According to the NFL Theorem [204] there is no correct answer to this question. Hence, a single MOEA cannot be the best MOEA for solving all classes of MOPs. In the case where a researcher tunes an MOEA for one class of problem, or more specifically for one instantiation of that problem class, an MOEA has the potential to perform better than all other MOEAs on that one problem instantiation. However, this does not mean that continued efforts to develop a generic MOEA should not be continued.

The development of any MOEA should employ the use of sound software engineering practices. These practices include the use of a standard structure to code, the code should be well documented and user-friendly as well as robust enough to operate on a number of popular hardware and software environments. Good software engineering practices can reduce the number of coding errors and decrease the difficulty of debugging the code. The MOEA developed in this research effort is the MOMGA-II and these principles were used to guide the development of the code. While it is difficult to follow these practices, the time expended up front leads to drastic savings in debugging time expended later.

A good guideline to use when selecting another researcher's MOEA, or an optimization approach in general, is to make sure that the problem is clearly stated and understood and that the MOEA is fully understood. The discussion of BBs presented in Chapter II as well as this chapter illustrate the usefulness of explicit BB approaches for solving MOPs. The area of explicit BB-based MOEAs is still relatively new and unexplored. An efficient and effective explicit BB-based approach is explored.

3.2 MOEA Metrics

Evaluating the efficiency and effectiveness of any MOEA is a difficult task to accomplish. Efficiency is a measure of resource use, such as memory requirements, CPU utilization, network bandwidth, file storage and more, as well as the total wall clock time required for an algorithm to execute. Effectiveness is a measure of the quality of the solutions that are generated by the MOEA. Metrics are used in an attempt to determine if a difference exists and how much of a difference exists between two measured sets and hence allow researchers the ability to report on the performance of MOEA approaches. Some researchers attempt to compare the results of their MOEA to other MOEAs in terms of analyzing the performance of a specific MOEA across a broad range of MOPs or just one specific problem area. In either case, a set of defined metrics is necessary to compare the performance of the MOEAs. Besides comparing the performance of MOEAs, the use of metrics allows researchers the ability to report the performance of his/her MOEA when applied to a specific application for which no existing data on the performance of other MOEAs exists. This again is a difficult problem and creates the need for a set of metrics to measure the efficiency and effectiveness of an MOEA.

Comparisons of various MOEA approaches and measures of efficiency and effectiveness are desired goals by those who conduct research and development of MOEAs. Standardized metrics for MOEAs were somewhat lacking in the field until recently when Srinivas and Deb [176], Schott [172], Czyżak and Jaszkiewicz [35], Zitzler and Thiele [213], and Van Veldhuizen and Lamont [184, 190] proposed a number of metrics. More recently Knowles and Corne summarized and evaluated many of the metrics in one of the most complete evaluations to date [113, 114]. The metrics proposed by the authors [35, 113, 114, 172, 176, 184, 190, 213] are genotypic and phenotypic based and provide a foundation for comparing the performance of various MOEAs. Prior to the introduction of these metrics, there was no standardized way of comparing MOEAs besides the qualitative comparison of the true Pareto front PF_{true} versus the known Pareto front PF_{known} that the MOEA finds. This standardization of metrics allows for statistical comparisons between various MOEA approaches and associated parameter values.

Some authors have recognized a problem associated with existing metrics [31, 113, 114, 184]. Typically metrics take a multidimensional set of data, PF_{known} , and map it into a reduced dimensional form, many times a scalar value. Thus a single scalar value cannot convey the same amount of information found in the multidimensional Pareto front. In analyzing the metrics discussed in this chapter, one must determine whether or not the associated metric is viable to be applied to the MOP being solved.

Test problems exist in the MOEA community in which PF_{true} is known and hence metrics have been developed to measure the performance of MOEAs applied to these MOPs. However, in many real-world MOPs PF_{true} is often unknown and these same metrics cannot be used. Similar issues arise when applying certain metrics to MOPs of varying characteristics, in terms of the shape of the front and the connectedness. These issues make the analysis of MOEA results sometimes more challenging then the design of an MOEA.

A number of researchers [31, 44, 114, 210] have recognized three main goals to be achieved when analyzing MOEA performance. In particular, the goals of an MOEA should include:

- The distance from PF_{known} to PF_{true} should be minimized.
- A uniform distribution of points across the Pareto front should be found. The points found should be evenly spaced across the front even though all of the points on the front may not be found.
- A large number of solutions should be found across the entire front, in which no area of the front should be lacking unless that is the characteristic of PF_{true} .

Some authors feel that a generic goal would include only the first item as finding the true Pareto front has been identified as a goal of MOEAs [114].

Knowles and Corne presented a thorough analysis of many of the metrics from the current literature, evaluated them in terms of their Pareto Compatibility, pros, and cons. For each of the metrics presented in this chapter that Knowles and Corne analyzed [113, 114], their assessment of the quality of each nondominated set comparison metric (NDSCM) follows. Prior to discussing the metrics, one must understand the details used by Knowles

and Corne to conduct their analysis. Knowles and Corne in part used the outperformance relations from Hansen et al. [87]. These relations express the relationship between two sets of nondominated objective function vectors, A and B, where $\mathbf{ND}(S)$ identifies the nondominated points in S.

Definition 24 (Weak Outperformance): *$A\ O_W\ B \iff \mathbf{ND}(A \cup B) = A$ and $A \neq B$ i.e., A* weakly outperforms *B if all points in B are 'covered' by those in A (where 'covered' means is equal to or dominates) and there is at least one point in A that is not contained in B.* □

Definition 25 (Strong Outperformance): *$A\ O_S\ B \iff \mathbf{ND}(A \cup B) = A$ and $B \setminus \mathbf{ND}(A \cup B) \neq 0$ i.e., A* strongly outperforms *B if all points in B are 'covered' by those in A and there is some point in B dominated by a point in A.* □

Definition 26 (Complete Outperformance): *$A\ O_C\ B \iff \mathbf{ND}(A \cup B) = A$ and $B \cap \mathbf{ND}(A \cup B) \neq 0$ i.e., A* completely outperforms *B if each point in B is dominated by a point in A.* □

Of these relations, it is important to notice that complete outperformance is the strongest of the relations and weak outperformance is the weakest, i.e., $A\ O_C\ B \implies A\ O_S\ B \implies A\ O_W\ B$. Knowles states that these definitions can be used to describe the relationships between approximations to PF_{true} since they are only dependent on standard Pareto dominance relations [114]. He further explains that these are not metrics and therefore provide no information if each set contains points that are not covered or dominated by the other set. Any metric that is not compatible with these relations is not guaranteed to provide meaningful results in isolation. Further, Hansen et al. [87], present the following definitions of compatibility and outperformance relations:

Definition 27 (Weak Compatibility): *A comparison metric R is* weakly compatible *with an outperformance relation O if for each pair of nondominated sets A, B with $A\ O\ B$, R evaluates A as being no worse than B.* □

Definition 28 (Compatibility): *A comparison metric R is* compatible *with an outperformance relation O if for each pair of nondominated sets A, B such that $A\ O\ B$, R evaluates A as being better than B.* □

Knowles uses the outperformance relations to compare various metrics. He categorizes metrics into direct comparative metrics; those metrics that compare two sets using a scalar measure, reference metrics; those metrics that must use a reference set to yield a value, and independent metrics; those metrics that measure a property of a set without requiring a comparison to another set or a reference set. Additionally, Knowles states if a metric induces a complete ordering, which allows one to state set A is better than set B, set B is better than set C, and hence set A is better than set C [114].

Knowles uses Pareto compatibility to state whether a metric is misleading. Pareto compatibility is defined to be the compatibility of a metric with the outperformance relations. He states that the hardest relation to be compatible with is O_W and the easiest is O_C. Knowles further defines two desirable features of a metric; monotony and relativity as [114]:

Definition 29 ((Weak) Monotony): *Given a nondominated set A, adding a nondominated point improves the value of the metric.* □

Definition 30 ((Weak) Relativity): *The evaluation of Z^* is (non)-uniquely optimal, i.e., all other nondominated sets have a strictly inferior evaluation.* □

Compatibility with O_W is necessary and sufficient for ensuring monotony and not necessary but sufficient for ensuring relativity.

3.2.1 Chi-Square-Deviation Metric. Srinivas and Deb [176] present a metric to evaluate the performance of MOEAs. They use a chi-square-like deviation form distribution measure shown in Equation (3.1)

$$\iota = \sqrt{\sum_{i=1}^{q+1} \left(\frac{n_i - \bar{n}_i}{\sigma_i} \right)^2} \tag{3.1}$$

where q is the number of desired optimal points and $(q+1)$-th sub-region is the dominated region, n_i is the actual number of individuals serving the i-th sub-region of the nondominated region, and σ_i^2 is the variance of the individuals serving the i-th sub-region of the nondominated region. The authors state that an algorithm with a good distribution capability is characterized by a low deviation measure [176]. This metric was not evaluated by Knowles but one can see a disadvantage is in the requirement of one to select a number of desired solution points (q). In problems in which one does not know the cardinality of the solution set, this metric is not easily applied. Since in most real-world application MOPs, one does not know the cardinality of the solution set, this metric is not typically useful.

3.2.2 Conservative Distance Convergence Metric. In [46], Deb presents two metrics for adoption. The first is based on the consecutive distances among the solutions of the best nondominated front in the final population. Deb proposes finding a set of 500 uniformly spaced solutions from PF_{true}. Then one takes the points found, PF_{known}, and computes the minimum Euclidean distance of them from the 500 solutions. The average of these distances is referred to as the convergence metric, Υ. Deb suggests using the average, $\bar{\Upsilon}$, and variance, σ_Υ of the convergence metric to evaluate an MOEA's convergence to PF_{true}. He states that even if PF_{known} converges to PF_{true}, the metric only realizes a zero value when all of the points lie on all of the PF_{true} points.

Deb suggests using a second metric, Δ to measure the spread of points along the front. This metric consists of a calculation of the Euclidean distance, d_i of each consecutive point on PF_{known} and then calculating the average of the distances, $\bar{d}$. Using these values, one can compute the spread of solutions along the front, as shown in Equation (3.2)

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{d_f + d_l + (N-1)\bar{d}} \tag{3.2}$$

where d_f and d_l are the Euclidean distances between the extreme solutions and the boundary solutions respectively, and N is the number of solutions. Deb states that for the most widely spaced solutions, the numerator would be zero, yielding a zero value for the metric.

These two metrics have not been analyzed by Knowles. The first metric, the consecutive distance metric requires the use of a comparison set containing points on PF_{true} and hence cannot typically be used. In some cases PF_{true} is known but in many others it is not and hence limits the useability of this metric. Deb's Δ metric is similar to other spacing metrics that exist in the literature and attempts to measure the distribution and uniformness of the spread of solutions in PF_{true} .

3.2.3 Error Ratio Metric. Van Veldhuizen [184] defines Error Ratio (ER) as the error in PF_{known} when compared to PF_{true} .

$$ER \triangleq \frac{\sum_{i=1}^{n} e_i}{n}, \tag{3.3}$$

where n is the number of vectors in PF_{known} and

$$e_i = \begin{cases} 0 & \text{if vector } i, i = (1, \ldots, n) \in PF_{true}\,, \\ 1 & \text{otherwise.} \end{cases} \tag{3.4}$$

For example, $E = 0$ indicates every vector reported by the MOEA in PF_{known} is actually in PF_{true} ; $E = 1$ indicates that none are. Also note that a similar metric [213, 214, 215] measures the "percentage of solutions" in some set (e.g., P_{known} or PF_{known}) dominated by another solution set's members (e.g., P_{true} or PF_{true}) as presented next.

Knowles states that this metric is only weakly compatible with O_C, incompatible with the outperformance relations, induces a total ordering, and violates monotony as well as relativity [114]. Another disadvantage of this metric is in its requirement of a reference set and that the addition of other members into the set being measured may lead to a worse metric value even if they are closer to PF_{true} (but not elements of PF_{true}) than the current members in the set.

3.2.4 Relative Coverage Metric. Zitzler et al. [210] present an MOEA metric which is referred to as a relative coverage comparison of two sets. Consider $X', X'' \subseteq X'$ as two sets of phenotype decision vectors. CS is defined as the mapping of the order pair

(X', X'') to the interval $[0, 1]$ per Equation (3.5).

$$CS(X', X'') \triangleq \frac{|\{a'' \epsilon X''; \exists a' \epsilon X' : a' \succeq a''\}|}{|X''|} \tag{3.5}$$

If all points in X' dominate or are equal to all points in X'', then by definition $CS = 1$. $CS = 0$ implies the opposite. In general, $CS(X', X'')$ and $CS(X'', X')$ both have to be considered due to set intersections not being empty. Additionally, this metric can be used for X = P_{known} or PF_{known} .

Knowles states that this metric is complicated to analyze in reference to the compatibility relations he uses [114]. In general he states that the relative coverage metric is not compatible with O_W but is compatible with O_S and O_C. A disadvantage of this metric lies in the complex analysis sometimes necessary to determine what is indicated by the metric result. In cases where the sets being compared are of differing cardinalities and contain unevenly distributed points along the front, the metric result is somewhat different from the result that one would expect, further indicating the care that must be given when interpreting the result of the relative coverage metric. An advantage of this metric is that it does not rely on a reference set. The complexities associated with interpreting the results of this metric discourage its use.

3.2.5 Maximum Pareto Front Error Metric. Comparing two sets to determine which one contains "better" solutions is a typical occurrence in the MOEA community. One may attempt to determine how far apart and how similar two sets, PF_{known} and PF_{true} are. The maximal Pareto front error metric determines a maximum error band, that when considered with respect to PF_{known} , encompasses every vector in PF_{true} [184]. Put another way, this is the largest minimum distance between each vector in PF_{known} and the corresponding closest vector in PF_{true} . This metric is defined as:

$$ME \triangleq \max_j(\min_i \mid f_1^i(\vec{x}) - f_1^j(\vec{x}) \mid^2 + \mid f_2^i(\vec{x}) - f_2^j(\vec{x}) \mid^2)^{1/2}, \tag{3.6}$$

where $i = 1, \ldots, n_1$ and $j = 1, \ldots, n_2$ index vectors, respectively, in PF_{known} and PF_{true}. A result of 0 indicates $PF_{known} \subseteq PF_{true}$; any other value indicates at least one vector in PF_{known} is not in PF_{true}.

This metric requires the use of a reference set. Knowles states that this metric induces a complete ordering but is not compatible with any of the outperformance relations [114]. It meets the criteria for weak relativity and violates weak monotony. An advantage of this metric is the ease of computation. This metric's disadvantage is that it does not always result in a value that one would expect given the PF_{known} set obtained. For example, a set containing mostly members in PF_{true} but a single or few members relatively far away from PF_{true} may be considered of lesser quality than a set consisting of members very close to PF_{true} but not containing any members in PF_{true}. This possibility makes this metric relatively difficult to interpret and hence it must be used with multiple other metrics in order to lend insight into the results. The potential difficulty of interpreting the results of the maximum Pareto front error metric discourages its use.

3.2.6 Average Pareto Front Error Metric. The average Pareto front error attempts to measure the convergence of an MOEA by using the distance of PF_{known} to PF_{true}. From each solution in PF_{known}, its perpendicular distance to PF_{true} is determined by approximating PF_{true} as a combination of piece-wise linear segments with the average of these distances defining the metric value [31].

This metric was not analyzed by Knowles but exhibits similar properties to the maximum Pareto front error metric. A disadvantage of this metric is its approximation of PF_{true} versus use of the real PF_{true}. The quality of this metric is dependent on the quality of the approximation. Additionally multiple researchers could apply this metric to the same MOEA and MOP and generate different results if they did not use the same approximation. Hence this metric is somewhat easy to understand but requires the reporting of multiple parameters in order to ensure that the results are comparable to other researcher's results. Without the exact approximation, one cannot compare the results of multiple MOEAs over the same MOPs with any confidence in the metric values. Due to the disadvantages and

possibility for incorrect comparisons of this metric value among researchers, this metric is not recommended for use.

3.2.7 Hyperarea Metric. Zitzler and Thiele, propose an MOEA comparative metric which can be termed *hyperarea* [214]. Hyperarea defines the area of objective value space covered by PF_{known} (i.e., the "area under the curve"). For example, a vector in PF_{known} for a two-objective MOP defines a rectangle bounded by an origin and $(f_1(\vec{x}), f_2(\vec{x}))$. The union of all such rectangles' area defined by each vector in PF_{known} is then the comparative measure and is defined as [184]:

$$H \triangleq \{\bigcup_i a_i \mid v_i \in PF_{known}\}, \tag{3.7}$$

where v_i is a nondominated vector in PF_{known} and a_i is the hyperarea determined by the components of v_i and the origin.

Zitzler et al., note that this metric may be misleading if PF_{known} is non-convex. They also assume the MOP's objective space origin coordinates are $(0, \ldots, 0)$, which is not always the case. In addition, they propose a hyperarea ratio metric defined as:

$$HR \triangleq \frac{H_1}{H_2}, \tag{3.8}$$

where H_1 is the hyperarea of PF_{known} and H_2 that of PF_{true}. In a minimization problem, this ratio is 1 if $PF_{known} = PF_{true}$ and greater than one if PF_{known}'s hyperarea is larger than PF_{true}'s.

The hyperarea metric may yield confusing and misleading results when applied to maximization MOPs and non-convex MOPs. This metric was meant for analyzing convex minimization MOPs. Since the properties of many real-world MOPs are unknown, this metric is more useful when applied to MOPs in which the Pareto front is known to be convex. The hyperarea ratio requires the use of a reference set and the conclusions drawn for the hyperarea metric apply to hyperarea ratio.

Knowles states that the hyperarea metric is compatible with O_W provided that the upper boundary of the dominated region is set so that all of the feasible nondominated points are always positive [114]. He states that the advantages of this metric are that it is compatible with the outperformance relations, it is an independent metric, and it can distinguish between different levels of outperformance. The disadvantage of this metric is that it requires the upper boundary of the feasible region to be defined. This boundary directly effects the ordering of the nondominated sets and he states it is difficult to justify any specific setting for this boundary. This metric also has a large computational overhead, hence making an analysis of a PF_{known} set of large cardinality a time consuming process. For PF_{known} sets of high dimensionality or large cardinality, this metric is unusable. Due to the overhead and disadvantages of this metric, it is not recommended for use.

3.2.8 Generational Distance Metric. Van Veldhuizen defines the Generational Distance as a value representing in the average how "far" PF_{known} is from PF_{true} [184]:

$$GD \triangleq \frac{(\sum_{i=1}^{n} d_i^p)^{1/p}}{n}, \tag{3.9}$$

where n is the number of vectors in PF_{known}, $p = 2$, and d_i is the Euclidean distance (in objective space) between each vector and the *nearest* member of PF_{true}. A result of 0 indicates $PF_{true} = PF_{known}$; any other value indicates PF_{known} deviates from PF_{true}. In the case of a disconnected Pareto front, for any p value, regions of the Pareto front can use the same equation and then a weighted sum can be obtained across the entire Pareto front. Figure 3.2 presents example PF_{true} and PF_{known} sets. The generational distance for PF_{known} in Figure 3.2 has: $d_1 = \sqrt{(1.2-1)^2 + (7-7)^2} = 0.200$, $d_2 = \sqrt{(1.5-1.5)^2 + (4-4)^2} = 0.000$, and $d_3 = \sqrt{(3-2)^2 + (2-2)^2} = 1.000$.

$$G = \frac{\sqrt{0.200^2 + 0.000^2 + 1.000^2}}{3} = \frac{1.020}{3} = 0.340$$

Knowles states that this metric induces a total ordering [114]. This metric also requires the use of a reference set in order to determine the distance from the reference set. Generational distance is relatively easy to compute, which makes it desirable to use.

A disadvantage of this metric is its incompatibility with O_W and it cannot be used reliably with sets that are changing in cardinality, hence this metric is not a good measure of the progress of an MOEA from generation to generation. This metric also cannot differentiate between different levels of complete outperformance but when used in conjunction with other metrics can provide useful information. This metric yields useful information as it states how close an MOEA comes to the actual solution of an MOP and hence this metric is recommended for use in conjunction with other metrics and in MOPs where PF_{true} is known.

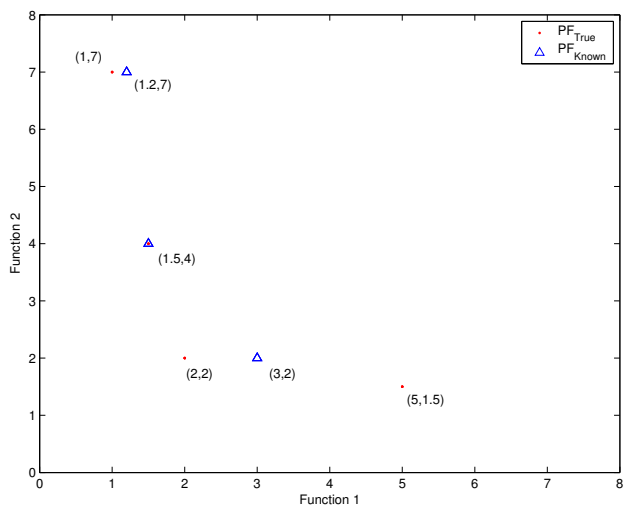

Figure 3.2 Example Minimization MOP PF_{true} and PF_{known}

3.2.9 7-Point Average Distance Metric. Schott defines a "7-Point" average distance measure that is similar to generational distance [172]. In his experiments, neither P_{true} or PF_{true} are known, so he generates seven points in objective space for comparison. Assuming a two objective minimization MOP and an (f_1, f_2) coordinate system with origin at (0,0), first determine the maximum value in each objective dimension. Two equidistantly spaced points are then computed between the origin and each objective's maximum value (on the objective axis). The metric value is then created by averaging the Euclidean distances from each of the seven axis points to the member of PF_{known} closest to each point. In a general 2 objective function minimization MOP $F(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}))$, the

seven points are:

$$\{(0, (\max f_2(\vec{x}))/3),\ (0, 2 * (\max f_2(\vec{x}))/3),\ (0, (\max f_2(\vec{x}))),\ (0, 0),$$
$$((\max f_1(\vec{x}))/3, 0),\ (2 * (\max f_1(\vec{x}))/3, 0),\ ((\max f_1(\vec{x})), 0)\}. \tag{3.10}$$

This metric was not analyzed by Knowles. The 7-Point average distance metric provides a method to attempt and find how close a solution set is to PF_{true} when one does not have knowledge of PF_{true}. The disadvantage of this method lies in the determination of the maximum value within each objective dimension. In non-linear optimization problems, this may not be readily available and hence one may expend a large amount of computational resources in attempting to generate those points. Additionally, calculating the points that are equidistant from the origin to the maximum values may also be intensive. In such cases, one may be better off solving for PF_{true} and then using the generational distance metric. This metric also does not have knowledge of PF_{true} and hence distances are only calculated from the seven axis points to a single member of PF_{known}. Hence this metric does not present a good measure of how far PF_{known} is from PF_{true} but instead an average measure of the distance from a single point in PF_{known} to seven generated points in objective space, not guaranteed to be in PF_{true}. This metric is not recommended for use.

3.2.10 Spacing (Range) Metric. In some operational settings, one desires to measure the spread of points across PF_{known}. Most experimental MOEAs perform fitness sharing (niching or crowding) in an attempt to spread each generational population ($PF_{current}(t)$) evenly along the known front. Since one has knowledge of PF_{known} and its endpoints at the conclusion of the MOEA, a spacing metric can define how well PF_{known} (or PF_{true}) is distributed. Schott proposes such a metric measuring the range (distance) variance of neighboring vectors in PF_{known} [172]. Called *spacing*, he defines this metric as:

$$S \triangleq \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\overline{d} - d_i)^2}\ , \tag{3.11}$$

where $d_i = \min_j(\mid f_1^i(\vec{x}) - f_1^j(\vec{x}) \mid + \mid f_2^i(\vec{x}) - f_2^j(\vec{x}) \mid)$, $i, j = 1, \ldots, n$, $\overline{d}$ is the mean of all d_i, and n is the number of vectors in PF_{known}. A value of zero for this metric indicates all members of PF_{known} are equidistantly spaced. Spacing for PF_{known} in Figure 3.2 has: $d_1 = |1.2 - 1.5| + |7 - 4| = 3.300$, $d_2 = |1.5 - 1.2| + |4 - 7| = 3.300$, $d_3 = |3 - 1.5| + |2 - 4| = 3.500$, and $\overline{d} = \frac{3.300+3.300+3.500}{3} = 3.367$.

$$\begin{aligned} \sum_{i=1}^{n} (\overline{d} - d_i)^2 &= (3.367 - 3.300)^2 + (3.367 - 3.300)^2 + (3.367 - 3.500)^2 = 0.027 \\ S &= \sqrt{\frac{1}{3-1} 0.027} = 0.115 \end{aligned}$$

Some MOPs (e.g., MOP3, MOP4, and MOP6 discussed in Chapter IV) have PF_{true}'s that are composed of two or more Pareto curves. Including the distance between the endpoints of two successive curves may skew Schott's range metric. Thus, for MOPs with this characteristic, the distance corresponding to the "breaks" in the front should be removed from the spacing computation.

Knowles states that this metric is incompatible with O_W, is incompatible with the outperformance relations, and does not meet the criteria for monotony or relativity [114]. However, this metric is useful when used in conjunction with other metrics. If one is aware of the structure of the front, as the structure of PF_{known} is known once the MOEA completes, then one can interpret the result of this metric correctly. Blindly using this metric, as is the case with many of the other metrics, can lead to an incorrect interpretation of the results. This metric requires low computational resources and can easily be generalized to multiple dimensions making this a favorable metric to use.

3.2.11 Overall Nondominated Vector Generation and Ratio Metric. This metric (ONVG) measures the total number of nondominated vectors found during MOEA execution and is defined as [184]:

$$ONVG \triangleq \mid PF_{known} \mid . \tag{3.12}$$

Schott uses this metric (although defined over the Pareto optimal set, i.e., $\mid P_{known} \mid$) [172]. Genotypically or phenotypically defining this metric is just a matter of preference, but note that multiple solutions may map to an identical vector $\mid P_{known} \mid \geq \mid PF_{known} \mid$, hence it is important to identify whether this metric is measuring the genotype or phenotype. Although counting the number of nondominated solutions gives some feeling for how effective the MOEA is in generating desired solutions, it does not reflect on how "far" from PF_{true} the vectors in PF_{known} are. Only a comparison between PF_{true} and PF_{known} as well as a spacing value can illustrate whether or not a particular ONVG value is "good." Without other information, one cannot determine if an ONVG value of 10 or 1000 is "good." This metric used in conjunction with a spacing and distance metric as well as a visualization of the Pareto front can yield a clear analysis of the MOEA results.

Van Veldhuizen defines the ratio (ONVGR) between the cardinalities of PF_{known} and PF_{true} as [184]:

$$ONVGR \triangleq \frac{\mid PF_{known} \mid}{\mid PF_{true} \mid}. \tag{3.13}$$

A value of 1 indicates that the MOEA has found the same number of nondominated vectors as exists in PF_{true}. It is important to recognize that the ratio also requires the use of other metrics in order to make a correct analysis of the results. It is possible for the cardinality of PF_{true} and PF_{known} to be identical yet PF_{known} be nowhere close to PF_{true}. Hence this metric must be used in conjunction with other metrics. The use of ONVG along with spacing, generational distance, and a visualization of the front is a good combination of metrics. ONVG and ONVGR for PF_{known} in Figure 3.2 is $ONVG = 3$ and $ONVGR = \frac{3}{4} = 0.750$.

Knowles states that ONVG induces a complete ordering on the set, but it is not weakly compatible with any of the outperformance relations, nor does it have the property of weak monotony or weak relativity [114]. An advantage of this metric is its ease of computation and the fact that it does not require the use of a reference set. As stated earlier, this metric is useful when used in conjunction with other metrics. Knowles' analysis of the ONVG metric is representative of his discussion of ONVGR.

3.2.12 Progress Measure Metric. Progress measure is a metric originally used in the single objective area to assess the convergence of an EA. This metric was defined by Bäck [13] and it attempts to measure relative, instead of absolute convergence improvement

$$P \triangleq \ln \sqrt{\frac{f_{max}(0)}{f_{max}(t)}}, \tag{3.14}$$

where $f_{max}(i)$ is the best objective function value in the parent population at generation i.

Van Veldhuizen modified this metric so as to present a method for using progress measure to evaluate the performance of MOEAs [184]. This metric is defined as:

$$RP \triangleq \ln \sqrt{\frac{G_1}{G_t}}, \tag{3.15}$$

where G_1 is the generational distance at generation 1, and G_t the distance at generation t.

This metric has not been evaluated by Knowles, but one can see that this metric can yield useful information about the convergence of an MOEA. However a disadvantage of this metric is that it only measures the progress between two specific generations of MOEA execution. In most MOEAs, the overall solution found throughout the entire MOEA execution is presented to the user. This metric only compares the progress between two generations and not the contribution of each generation to the overall solution set presented at MOEA termination. This metric is therefore not recommended for use.

3.2.13 Generational Nondominated Vector Generation Metric. (GNVG) This metric tracks how many nondominated vectors are produced at each MOEA generation and is defined as:

$$GNVG \triangleq \mid PF_{current}(t) \mid \ . \tag{3.16}$$

This metric is useful when used in conjunction with the progress measure for evaluating the MOEA performance from generation to generation. Knowles states that this metric exhibits the same advantages and disadvantages as the ONVG metric it is based off of [114].

3.2.14 Nondominated Vector Addition Metric. This metric evaluates the change in the cardinality of $PF_{known}(t)$ as the generations progress. As the MOEA is executed, one would like the cardinality of $PF_{known}(t)$ to increase and hence generate a PF_{known} set of high cardinality. This metric is then defined as:

$$NVA \triangleq \mid PF_{known}(t) \mid - \mid PF_{known}(t-1) \mid . \tag{3.17}$$

This metric may be useful when used in conjunction with GNVG and progress measure to evaluate the MOEA performance from initialization to termination. This metric can be misleading if one does not use it along with other metrics and conduct a thorough analysis of $PF_{known}(t)$. This is important as the addition of a potential solution to the $PF_{known}(t)$ set may result in the removal of many members of that set. This may occur if the new member dominates existing members. While one does not know how often this may occur, in most cases, an MOEA yields better solutions, that dominate previous solutions, as it progresses. Hence the cardinality of $PF_{known}(0)$ may be larger than that of PF_{known}, yet all of the solutions in PF_{known} may dominate those in $PF_{known}(0)$. Hence this metric only presents a change in the cardinality of the set and not in the quality of solutions and therefore is not recommended for use. Therefore this metric must be used with care and in conjunction with other metrics. Knowles states that this metric exhibits the same advantages and disadvantages as the ONVGR metric as it is based off of it.

3.2.15 $D1_R$. This metric measures the mean distance, over the points in a reference set, of the nearest point in an approximation set [35]. The definition of this metric is:

$$D1_R(A, \Lambda) \triangleq \frac{1}{|R|} \sum_{r \in R} min_{z \in A}\{d(\mathbf{r}, \mathbf{z})\} \tag{3.18}$$

where A is the approximation set, R is the reference set, $d(\mathbf{r}, \mathbf{z}) = max_k\{\lambda_k(r_k - z_k)\}$ and $\Lambda = [\lambda_1, \lambda_2, \ldots, \lambda_K]$, $\lambda_k = 1/R_k$, $k = 1 \ldots K$ with R_k representing the range of objective k in set R.

Knowles states that this metric induces a complete ordering, is weakly compatible with O_W, and incompatible with O_C [114]. The disadvantage of this metric is that it

calculates a weighted average dependent on the reference set. This calculation is similar to Schott's 7-Point average distance metric. An advantage of this metric is that it is easy to compute and it can differentiate between different levels of complete outperformance. This metric is not recommended for use as it requires one to select a good approximation set which is not known a priori for real-world MOPs.

3.2.16 R1 and R1$_R$. The R1 metric calculates the probability that a set A is better than a set B over a set of utility functions and R1$_R$ is identical to R1 when it is used with a reference set [88].

$$R1(A, B, U, p) = \left\{ \int_{u \in U} C(A, B, u) p(u) du \text{ , where} \right. \tag{3.19}$$

$$C(A, B, u) = \begin{cases} 1 & \text{if } u^*(A) > u^*(B) \\ 1/2 & \text{if } u^*(A) = u^*(B) \\ 0 & \text{if } u^*(A) < u^*(B) \end{cases} \tag{3.20}$$

where A and B are two approximation sets, U is a set of utility functions, $u : \Re^K \rightarrow \Re$ which maps each point in objective space into a measure of utility, $p(u)$ expresses the probability density of the utility $u \in U$, and $u^*(A) = max_{z \in A}\{u(\mathbf{z})\}$ and also for $u^*(B)$.

Knowles states that these metrics require a set of utility functions which must be defined [114]. They also use low computational resources and can differentiate between different levels of complete outperformance if given a reference set. These are good metrics to use but are somewhat complex to understand and require the use and determination of utility functions, reducing the attractiveness of the metric. Since the selection of utility functions may not be possible and may be difficult to complete, this metric is not recommended for use.

3.2.17 R2 and R2$_R$. The R2 metric is an improvement over R1 by taking into account the expected values of the utility functions. It calculates the expected difference in the utility of an approximation A with another approximation B [88]. The symbology

is the same as for R1 and R2 is defined as:

$$R2(A, B, U, p) = E(u^*(A)) - E(u^*(B)) = \int_{u \in U} (u^*(A) - u^*(B))p(u)du \tag{3.21}$$

Knowles states that these metrics require that it makes sense to add the values of different utility functions from the set U meaning that each utility function must be appropriately scaled [114]. He further discusses that these metrics can differentiate between different levels of complete outperformance. These are good metrics to use but are somewhat complex to understand and require the use and determination of utility functions, reducing the attractiveness of using them. Since the selection of utility functions may not be possible and may be difficult to complete, this metric is not recommended for use.

3.2.18 R3 and $R3_R$. The R3 metrics are similar to the R2 metrics but instead of calculating the difference of the utility values, the ratio of the best utility values is used [88]. The analysis of this metric is similar to the previous two metrics generated by Hansen and Jaszkiewicz. Again, these are good metrics to use but are somewhat complex to understand and require the use and determination of utility functions, reducing the attractiveness of using them. Since the selection of utility functions may not be possible and may be difficult to complete, this metric is not recommended for use.

3.2.19 Visualization. The visualization of MOEA statistical results is an important issue to address and deserves attention. One of the most basic and simplest ways to analyze the results of an MOEA is to visualize the P_{known} and PF_{known} sets. A simple plot of these sets is useful in determining what a number of the characteristics of the PF_{known} set are. For example, a visualization can illustrate the cardinality of the set, the number of disjoint fronts that appear, the structure of the front and more information. The advantage of conducting a visual analysis and comparison of MOEA results is that all of the data is present to the naked eye for analysis. A researcher can readily see the structure of the results of a visualization of PF_{known} . This is an advantage over some of the metrics discussed as single metric results can be misleading and the use of multiple metrics is important.

Visualization of MOEA results provides an easy mechanism to see the general MOEA performance and is recommended for use. A more detailed analysis using other metrics is necessary to compare the performance of multiple MOEAs. A visual comparison can easily illustrate complete outperformance but cannot as easily distinguish between MOEA results indicating mixed performance (MOEA 1 has some points better than MOEA 2 and vice-versa). Visualization of MOEA results also becomes more difficult if not impossible when analyzing the results of an MOP containing a large number of dimensions. Since three dimensions is typically the maximum that one can easily visualize, MOPs with greater than three dimensions require the plotting of only three of the dimensions at a time or the use of other metrics.

3.3 MOMGA-II Performance Metrics

Many of the metrics discussed in this chapter are described in the genotype or phenotype domains only, but these metrics can be equally applicable in either domain (applied to P_{known} or PF_{known}). Each of the metrics discussed attempts to map multiple points existing on the Pareto front to a single metric value. As is known in other academic fields and mathematics, such a mapping is lossy and hence a single metric typically cannot indicate overall performance. The metrics are referred to as lossy since the mapping of multiple points (the Pareto front) to a single metric value results in a loss of information. Which combination of metrics can provide the best analysis of a solution set? The answer to that question is still highly debated.

The metrics discussed in this chapter have been accepted and adopted by the MOEA community in limited or wide spread use. The summary presented in support of this research effort is meant to provide an overview of currently known MOEA metrics. This metric overview is used to select metrics for evaluating the research conducted in support of the objectives presented in Chapter I.

In order to quantitatively compare the results of the MOMGA-II with other MOEAs, statistical analysis of the experimental results and associated observations are presented in Chapter VI. Since no single metric can represent total MOEA performance, a series of appropriate metrics is used to measure the performance in the phenotype domain. MOEA

metrics should consist of comparisons of **1)** the PF_{known} values to the PF_{true} values to calculate generational distances, **2)** spacing along the PF_{known} Pareto front, and **3)** the range of values for each objective function [191, 210]. In cases where PF_{true} is unknown, it is important that metrics are selected that result in the measurement of the distribution of points along PF_{known} , and the cardinality of PF_{known} . These attributes are important to consider when evaluating the performance of an MOEA. Metrics that address the distribution of points and the cardinality of the known Pareto front are important to provide insight into the performance of an MOEA.

Knowles recommends the use of the hyperarea, R1, R2, and R3 metrics [114]. While Knowles states that these metrics are "good" metrics to use, this statement is highly dependent on the characteristics of the PF_{known} sets that one is measuring. In this dissertation, an analysis is made of MOEA performance as applied to MOPs some of which PF_{true} is known and others where PF_{true} is unknown. Additionally, the characteristics of the PF_{known} sets generated through this effort vary greatly. Some of the characteristics of the PF_{known} sets generated by the MOMGA-II include connected fronts, disconnected fronts, are of large and small cardinality, and are of varying dimensionality in P_{true} and PF_{true} . This makes the selection of metrics challenging. While Knowles makes the statement that the hyperarea, R1, R2, and R3 metrics are recommended for use, this statement should take into consideration the characteristics of the PF_{known} sets generated by the MOEA. For example, Knowles states that the hyperarea metric is unusable for PF_{known} sets of large cardinality. Since some of the PF_{known} sets generated in this research effort exhibit this characteristic, his recommended metric is not applicable. The R1, R2, and R3 metrics have many advantages but they all require the ability to define a set of utility functions. This is a time consuming process and is not always possible. The definition of utility functions is not completed in this research effort and hence these metrics are not applicable.

The metrics selected for use in this effort (Chapter VI), to evaluate and compare the performance of the MOMGA-II are generational distance, spacing, overall nondominated vector generation, and a visual comparison of Pareto front results. Of these, the only metric that requires one to have knowledge of PF_{true} is generational distance. Although other

metrics such as error ratio, max error, hyperarea ratio, progress measure, and others are discussed in this chapter and have been employed for MOEA testing, the four metrics (G, S, ONVG, visualization) used together are quantitatively quite adequate for statistical MOEA comparison. G, S, and ONVG allow a researcher the ability to make statistical statements about the performance of MOEAs and require low computational resources. These metrics along with a visual analysis of the PF_{known} sets provides the necessary insight into MOEA performance as applied to given MOPs with respect to the goals addressed in the beginning of this section.

A software package for calculating all of the discussed metrics is not readily available and hence researchers must either search for existing code or generate their own code. As the use of existing code is preferable, there are numerous statistical packages that may be used to analyze an MOEA's results.

Note that the discretization of the continuous decision variable domain can be critical in generating the MOP's Pareto front. That is, if only a very few decision variable values relate to a particular objective function optima, the statistical chance of generating these values is small assuming a uniform distribution of values. The resulting PF_{known} could thus be a considerable distance from PF_{true} resulting in an unacceptable solution. Some say that this is a deception problem, but in reality, it is a numerical analysis problem.

One also should observe that some authors use straight-line approximations to the continuous Pareto front if PF_{true} is functionally unknown in generating metric values [210]. A better method, which is employed (for MOP-C1 in Chapter VI), is to use high-order polynomial approximations minimizing the error over the discrete PF_{true} segments which gives more realistic results.

3.4 Summary

Throughout the past decade the number of MOEA publications has greatly increased. While many of these publications discuss new approaches and operators, many more discuss the application of an existing MOEA to a real-world or new MOP. This chapter presents a discussion of the most popular and highly referenced contemporary and recent MOEAs.

This serves as a good summary of the what could be considered the best MOEAs in the field. Through an understanding of the literature, one can improve an existing MOEA or develop a new, efficient and effective MOEA. An objective of this research is the latter of the two. The literature review presented illustrates the lack of interest, or understanding of explicit BB-based MOEAs by the MOEA community as a whole. This is an area that can yield great results and is the focus of this research.

The complexity and computational cost of an MOEA somewhat drive its utility in the real-world. While the cost of the fitness function calculation is typically dependent on the hardware of the machine, the cost of the MOEA in terms of execution time, memory utilization and complexity is left to the programmer. The most efficient MOEAs are sure to be used more often than those of lower efficiency, where efficiency includes the memory and processing requirements as well as the required time to execute. The same is true of complexity; typically researchers want to use the least complex approach, that yields the most accurate results. This process has a higher probability of researchers understanding the approach and improving upon it. That is not to say that complex MOEAs are not useful. One example of this is the MOMGA [184] and the MOEA developed through this research, the MOMGA-II. The MOMGA is a complex MOEA that takes an approach no other MOEA researches have used, yet it parallels the performance of other MOEAs and performs better on selected MOPs. The good performance realized by the MOMGA is one of the motivations of conducting research in the area of explicit BB-based MOEAs. The complexity and computational cost are important attributes of an MOEA but are also difficult to compare as many MOEAs make use of different operators, are coded by different researchers and therefore may be difficult to compare. However, the efficiency and effectiveness of these approaches can be analyzed and compared.

Once a researcher has implemented and tested what they feel is a well-engineered, efficient and effective MOEA, the researcher must have a method to evaluate the results obtained from the MOEA. A well-engineered MOEA is one in which a researcher has taken the time to address the issue of complexity, followed good software engineering practices and analyzed available literature to incorporate promising ideas. The latter part of this chapter presents a summary of many metrics used in the MOEA community to evaluate

the MOEA performance or compare and contrast the performance of multiple MOEAs. Evaluations of each of the metrics illustrate that there is no one best metric to use, but through a visual analysis of the characteristics of the MOEA results, one can make an intelligent decision as to which metrics can accurately evaluate the performance of the MOEA tested. Generational distance, ONVG, spacing, and visualization of the Pareto front are selected as the metrics to evaluate the performance of the MOEAs compared.

IV. MOEA MOP Test Suites

Throughout the previous two chapters, a detailed discussion of BBs and the MOEAs that incorporate BB concepts is presented. In the latter part of the last chapter, MOEA metrics are presented to analyze MOEA performance. Having presented the metrics, this chapter addresses the MOPs selected for use in evaluating MOEA performance. Additionally, a discussion of MOPs and associated MOP test suites is presented. This includes a discussion of a few unconstrained MOPs as well as constrained, NP-Complete, and real-world MOPs of varying characteristics. These MOPs are presented to aid researchers in evaluating the performance of MOEAs.

The selection of MOPs for inclusion into MOEA MOP test suites has been discussed in the MOEA literature and problems from standardized MOP test suites are selected [31, 43, 52, 189, 188]. Each MOP selected for use is described in this chapter. The MOP test problems used in this effort are selected from standardized standardized MOP test suites suggested and used by MOEA researchers. The use of standard test problem sets is recommended by Jackson, Boggs, Nash, and Powell to allow for comparisons of results presented by different researchers [97]. Different motivations exist for the selection of test MOPs. One of the motivations is to analyze general MOEA performance over a variety of classes of MOPs. While the selected MOEA may not have the best performance out of any given method over all of the MOPs, there may be a particular class of MOP that it performs the best on compared to the classes tested. Testing an MOEA over a number of classes of MOPs can illustrate the class of MOP that a given MOEA obtains the best performance over. Another motivation of testing an MOEA is to tune it to perform well over a single class of MOP. This is more indicative of the type of testing conducted by MOEA researchers attempting to solve a specific real-world problem. Researchers may choose to incorporate problem domain knowledge and tune their MOEA to a class or instantiation of this problem. These are two motivations for the selection of test MOPs. The motivation of this chapter is on the selection of test MOPs for analyzing the performance of MOEAs over multiple problem classes.

While a few unconstrained test functions from another test suite are discussed, the contribution of this effort is the addition of a real-world and constrained MOPs with integer

based decision variables into a test suite. Additional details of other researchers' test suite MOPs is presented in Appendix B. The specific MOPs selected for use represent MOPs of varying characteristics. The selection of MOPs and MOP test suites is just as important as the selection of an MOEA to apply to a problem. Researchers must ensure that they understand the MOP and the formulation used. This understanding is advantageous in tuning and selecting an MOEA that can perform well on the particular class of MOP selected. A researcher may also attempt to solve an MOP that no prior knowledge of the structure or characteristics of the Pareto front exists. In this case, a researcher can look at all of the available MOEAs for one that performs the best over that class of MOP. While knowledge of the characteristics of the Pareto front may be unknown for an MOP, the mathematical representation of the MOP is known. Using the mathematical formulation of the MOP, a researcher can determine which other MOEAs have performed well when applied to similar problems.

Test suites are integral to the presentation of an unbiased comparison of different MOEA approaches. This chapter is organized as follows: unconstrained test function MOPs are presented in Section 4.1, followed by a discussion of constrained test function MOPs, discrete MOPs and real-world MOPs in Sections 4.2 through 4.5. The constrained test suite MOP section contains the NP-Complete and real-world MOPs that are presented for use by other researchers in evaluating the performance of an MOEA against constrained MOPs formulated with integer based decision variables. Real-world MOPs of interest to industry and the government are also discussed. Test suite generators and other proposals are discussed in Section 4.2 to aid a researcher in creating a good design of experiments when testing the efficiency and effectiveness of their MOEA. These discussions are useful for generating a good test of an MOEA as well as identifying problem domain characteristics that, if addressed, can improve MOEA performance when applied to problems of this class.

4.1 Unconstrained Test Suite MOPs

The purpose of a test suite is to objectively determine the efficiency and effectiveness of an MOEA [43, 52, 189, 188]. The selection of MOPs for testing is a necessity to analyze or compare the performance of a single or multiple MOEA approaches. A researcher's mo-

tivation for applying an MOEA to an MOP from a test suite is to illustrate the performance of that MOEA as applied to the class of MOP selected or to compare the performance of multiple MOEAs as applied to the same MOP. It is also prudent to remember that an MOEA may perform better than other approaches over one or more classes of MOPs, but this does not prove that the MOEA performs better over all classes of MOPs it encounters. The No Free Lunch (NFL) Theorem states that one algorithm cannot outperform every other algorithm over every possible class of test problem [204].

EAs are generally accepted as "good" search methods when applied to problems in which the search space is extremely large, the search space is known to be neither flat nor chaotic, the fitness function is noisy, another tuned method for solving the MOP is nonexistent, or the MOP is not well understood [148]. Problems in which there are a large number of dimensions in the genotype or phenotype spaces may also be good problems for application by an evolutionary approach. It is important to apply an MOEA to problems that meet these criteria. The application of an MOEA to MOPs that do not meet the aforementioned criteria and hence are not challenging enough for an MOEA, likely result in poor performance.

Van Veldhuizen and others have selected MOPs for inclusion in test suites based in part on the guidelines that Whitley suggested [203]. These guidelines were stated in the context of single objective problems but apply to the MOPs also. The guidelines state that test suites should contain problems (MOPs) that [203]:

1. are difficult for simple search strategies to solve
2. are nonlinear, nonseparable, and nonsymmetric, requiring increased computational resources
3. are scalable
4. contain a large number of dimensions

Many of the researchers who have suggested MOP test suites have also suggested that the MOPs within these test suites include functions of varying characteristics: continuous, discontinuous, connected, disconnected, convex, concave, unimodal, multimodal, quadratic, and nonquadratic [74, 138]. Van Veldhuizen included historical MOPs that were

referenced in the literature in his test suite [184]. He created a standardized test suite to use based on MOPs with a variety of desired characteristics. It is important to include multiple classes of MOPs, with varying characteristics in order to test an MOEA's ability to solve problems of these classes.

Five unconstrained MOPs from Van Veldhuizen's standard test suite, which represent both minimization and maximization functions with varying complexity levels and Pareto front (phenotype) characteristics, are selected for use in this research effort. Two objective problems are used for ease of presentation and to provide critical insight to MOEA performance. The five MOPs selected are presented in Table 4.1 and are Schaffer's #1 labeled as MOP1, Fonseca's # 2 labeled as MOP2, Poloni's labeled as MOP3, Kursawe's labeled as MOP4, and Deb's labeled as MOP6 [184]. These MOPs are selected for testing based on the fact that they meet the criteria described above in terms of being good test MOPs for comparing MOEA approaches and are presented in Table 4.1. Additionally, other researchers have used a subset of these five MOPs to test their MOEAs [46, 47]. Other researcher's use of these MOPs allows for a comparison between the algorithm of interest in this research, the MOMGA-II, and other MOEAs that have already been applied to these test MOPs.

An analysis of the MOPs present in existing test suites reveals a generalized problem exists in terms of the dimensionality [43, 44, 50, 52, 53, 54, 184, 188, 189, 192]. Van Veldhuizen, Deb, and others chose MOPs for inclusion based on varying characteristics. However, many of these problems contain only two objective functions and only two or three decision variables, which is not typical of real-world application problems. These MOPs are of low dimensionality and may not be fully testing the capabilities of an MOEA. Additional MOPs are selected for use in this effort. These additional MOP formulations contain constraints, integer based decision variables, and characteristics non-existent in the other selected MOPs.

Table 4.1: MOEA Test Suite Functions [184]

MOP	Definition	Constraints
MOP1 P_{true} connected, PF_{true} convex	$F = (f_1(x), f_2(x))$, where $$\begin{aligned} f_1(x) &= x^2, \\ f_2(x) &= (x-2)^2 \end{aligned}$$	$-10^5 \leq x \leq 10^5$
MOP2 P_{true} connected, PF_{true} concave, number of decision variables scalable	$F = (f_1(\vec{x}), f_2(\vec{x}))$, where $$\begin{aligned} f_1(\vec{x}) &= 1 - \exp(-\sum_{i=1}^{n}(x_i - \frac{1}{\sqrt{n}})^2), \\ f_2(\vec{x}) &= 1 - \exp(-\sum_{i+1}^{n}(x_i + \frac{1}{\sqrt{n}})^2) \end{aligned}$$	$-4 \leq x_i \leq 4;\ i = 1, 2, 3$
MOP3 P_{true} disconnected, PF_{true} disconnected (2 Pareto curves)	Maximize $F = (f_1(x,y), f_2(x,y))$, where $$\begin{aligned} f_1(x,y) &= -[1 + (A_1 - B_1)^2 + (A_2 - B_2)^2], \\ f_2(x,y) &= -[(x+3)^2 + (y+1)^2] \end{aligned}$$	$-3.1416 \leq x, y \leq 3.1416$, $$\begin{aligned} A_1 &= 0.5\sin 1 - 2\cos 1 + \sin 2 - 1.5\cos 2, \\ A_2 &= 1.5\sin 1 - \cos 1 + 2\sin 2 - 0.5\cos 2, \\ B_1 &= 0.5\sin x - 2\cos x + \sin y - 1.5\cos y, \\ B_2 &= 1.5\sin x - \cos x + 2\sin y - 0.5\cos y \end{aligned}$$
MOP4 P_{true} disconnected, PF_{true} disconnected (3 Pareto curves), number of decision variables scalable	$F = (f_1(\vec{x}), f_2(\vec{x}))$, where $$\begin{aligned} f_1(\vec{x}) &= \sum_{i=1}^{n-1}(-10e^{(-0.2)*\sqrt{x_i^2 + x_{i+1}^2}}), \\ f_2(\vec{x}) &= \sum_{i=1}^{n}(\lvert x_i\rvert^{0.8} + 5sin(x_i)^3) \end{aligned}$$	$-5 \leq x_i \leq 5;\ i = 1, 2, 3$
MOP6 P_{true} disconnected, PF_{true} disconnected (4 Pareto curves), number of Pareto curves scalable	$F = (f_1(x,y), f_2(x,y))$, where $$\begin{aligned} f_1(x,y) &= x, \\ f_2(x,y) &= (1+10y) * \\ &\quad [1 - (\frac{x}{1+10y})^{\alpha} - \frac{x}{1+10y}\sin(2\pi q x)] \end{aligned}$$	$0 \leq x, y \leq 1$, $$\begin{aligned} q &= 4, \\ \alpha &= 2 \end{aligned}$$

4.2 Constrained Test Suite MOPs

Constrained numeric MOPs should also be included in any comprehensive MOEA test function suite. Suitable linear and nonlinear constrained MOPs are presented as drawn from the published literature [43, 44, 50, 52, 53, 54, 184, 188, 189, 192]. Many of the constrained problems presented in the literature use linear constraints and have similar characteristics to other constrained MOPs. In order to select constrained problems of varying characteristics, Binh's constrained MOP from Van Veldhuizen [184] is selected as well as a modification of Tanaka's constrained MOP [182]. Table 4.2 presents the formulation for these two MOPs. Binh's MOP contains a single convex Pareto curve, uses linear constraints, and is labelled **MOP-C1**.

Table 4.2: Constrained MOEA Test Suite Functions

MOP	**Definition**	**Constraints**
MOP-C1 Binh(2)	$F = (f_1(x,y), f_2(x,y))$, where $f_1(x,y) = 4x^2 + 4y^2,$ $f_2(x,y) = (x-5)^2 + (y-5)^2$	$0 \le x \le 5,\ 0 \le y \le 3,$ $0 \ge (x-5)^2 + y^2 - 25,$ $0 \ge -(x-8)^2 - (y+3)^2 + 7.7$
MOP-CT Tanaka Test Function Generator	$F = (f_1(x,y), f_2(x,y))$, where $f_1(x,y) = x,$ $f_2(x,y) = y$	$0 < x, y \le \pi,$ $0 \ge -(x^2) - (y^2) + 1 + (a\ cos(b\ arctan(x/y)))$ $a = 0.1$ $b = 16$

A modification of Tanaka's two objective function is proposed by this research effort as **MOP-CT** with a and b tuning parameters [182]. The original Tanaka MOP formulation does not contain the a and b terms in the constraint but instead uses the constants $a = 0.1$ and $b = 16$. Thes original formulation has already been proposed as a test func-

tion [182, 184]. The original formulation of the Tanaka MOP is modified to include the a and b parameters in the constraints. These are tuning parameters that allow one to modify the characteristics of the Pareto optimal set and the Pareto front. The nonlinear constraints can be modified to change the characteristics of the Pareto front from a continuous front of five disconnected sections to multiple continuous disconnected fronts to multiple disconnected fronts where each disconnected section consists of a single point. Each of these possible MOP formulations have a different number of infeasible points between the feasible points and hence varying levels of difficulty. MOP-CT allows a researcher to easily change the characteristics and level of difficulty of a single MOP without having to code an entirely different MOP for testing various MOP characteristics.

MOPs with a continuous Pareto front are typically easier for an MOEA to generate solutions compared to MOPs with multiple disconnected Pareto fronts [31]. This is due to the dispersion in phenotype space of the disconnected fronts. Once an MOEA generates a solution on the Pareto front of an MOP with a continuous front, it may be easy for the MOEA to find other solutions that are within a small perturbation of the decision variable values to this solution in phenotype space. To generate other solutions on the Pareto front may just require a single mutation event if the Pareto front solutions are close together in terms of a distance metric in phenotype space and reflect the same distance characteristic in genotype space. In MOPs with disconnected fronts, it is more difficult for an MOEA to generate a solution on each of the separate Pareto fronts and continue to generate these solutions as the MOEA must find the good BBs required to generate solutions on each of the fronts. In the previous example, if solutions that are close in phenotype space reflect solutions that are a small perturbation away in genotype space, then one can see that disconnected fronts reflect a larger distance between solutions. This larger distance translates to multiple perturbations being required to generate a Pareto front solution on a different portion of the Pareto front starting with a solution from another portion of the Pareto front. This characteristic of a similar distance relationship between the genotype and phenotype spaces is problem dependent but illustrates how the characteristic of a connected or disconnected Pareto front may pose difficulty for MOEAs.

Constrained MOPs present additional difficulties over unconstrained MOPs for MOEAs as potential solutions that are close in phenotype space may be feasible or infeasible and directly effect the search process. Considering the previous example of a similar relationship between the distances of solutions in the genotype and phenotype spaces, one can see that infeasible points between feasible solutions can pose difficulty for the MOEA search process. Small perturbations to feasible solutions may yield infeasible points, whereas a larger perturbation yields a feasible solution. This makes the search somewhat difficult and this is especially true if some of the good BBs necessary to generate solutions on the front also generate infeasible points in the space.

Figures 4.1 through 4.6 reflect six different resultant characteristics of MOP-CT. These variations are generated through varying the MOP constraints and were generated through a total enumeration of the space and a Pareto analysis of the resultant feasible solutions at a specific resolution. In each of these figures, the feasible region is reflected by the shaded area and the *s represent the Pareto front. As the a and b parameters are modified, one can see the effect on the characteristics of the Pareto front and the feasible region. The characteristics of the six variants of the Tanaka function are as follows:

- Standard Tanaka phenotype with $a = .1$ and $b = 16$. The Pareto front represents 5 disconnected curves and is symmetric about the line $x = y$. (Figure 4.1)
- Smaller continuous phenotype regions with $a = .1$ and $b = 32$ with larger infeasible regions between Pareto front points as compared to the standard settings. The Pareto front represents 8 disconnected curves and is symmetric about the line $x = y$. (Figure 4.2)
- Decreased distance between Pareto front regions with, $a = .1$ and $b = 16$ as compared to the standard settings. The Pareto front represents 8 disconnected curves and is symmetric about the line $x = y$. (Figure 4.3)
- Increased distance between Pareto front regions with, $a = .15$ and $b = 32$ with larger infeasible regions between Pareto front points as compared to the standard settings. The Pareto front represents 16 disconnected curves (some are individual points) and is symmetric about the line $x = y$. (Figure 4.4)

- Increased distance between Pareto front regions with, $a = .1(x^2 + y^2 + 5xy)$ and $b = 32$ with deeper infeasible regions between Pareto front points as compared to the standard settings. The Pareto front represents 16 disconnected curves (some are individual points) and is symmetric about the line $x = y$. (Figure 4.5)
- Increased distance between Pareto front regions with, $a = .1(x^2 + y^2 + 5xy)$ and $b = 8(x^2+y^2)$ with deeper non-periodic infeasible regions between Pareto front points as compared to the standard settings. The Pareto front represents 6 disconnected curves (some are individual points) and is not symmetric about the line $x = y$. (Figure 4.6)

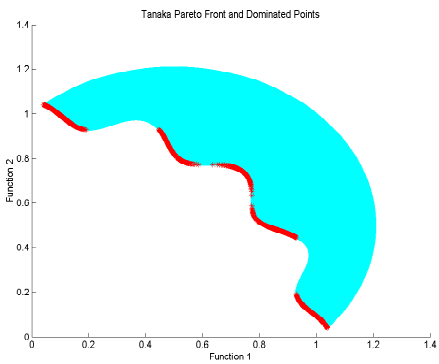

Figure 4.1 MOP-CT (Tanaka), $a = .1, b = 16$, Original PF_{true} (P_{true}) regions

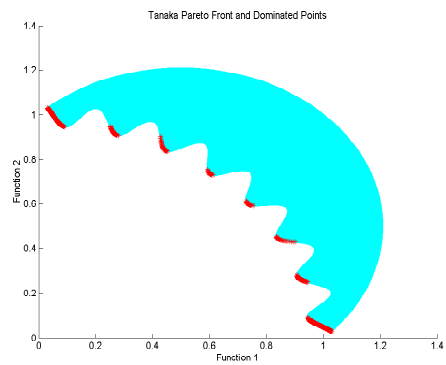

Figure 4.2 MOP-CT (Tanaka), $a = .1, b = 32$, PF_{true} (P_{true}) regions

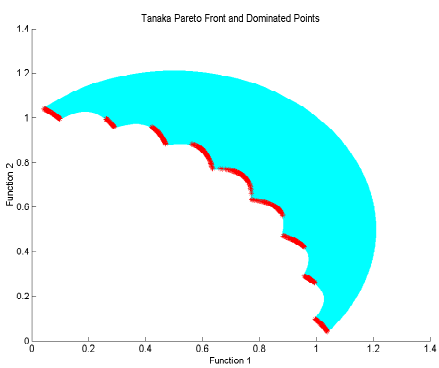

Figure 4.3 MOP-CT (Tanaka), $a = .1, b = 16$, PF_{true} (P_{true}) regions

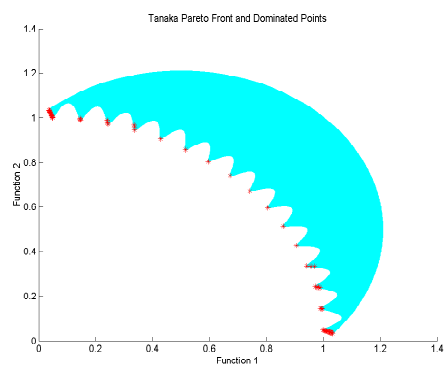

Figure 4.4 MOP-CT (Tanaka), $a = .15, b = 32$, PF_{true} (P_{true}) regions

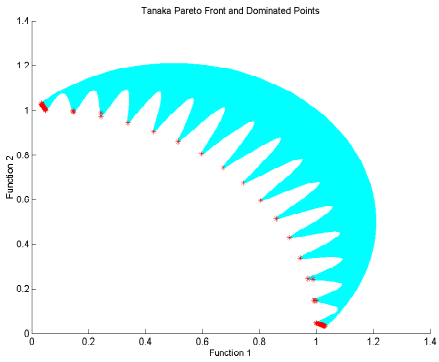

Figure 4.5 MOP-CT (Tanaka), $a = .1(x^2 + y^2 + 5xy), b = 32$, PF_{true} (P_{true}) periodic regions

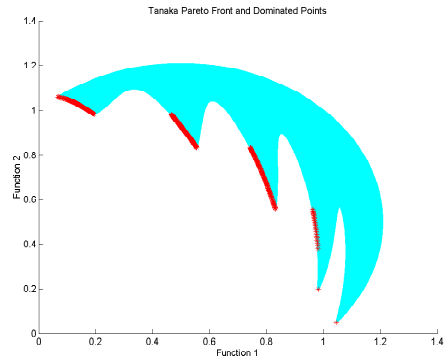

Figure 4.6 MOP-CT (Tanaka), $a = .1(x^2+y^2+5xy), b = 8(x^2+y^2)$, PF_{true} (P_{true})

The center sections of the Pareto front shown in Figure 4.1 are difficult to find numerically because of the near horizontal or vertical slope encountered in this portion of the front. Besides modifying the tuning parameters, increasing or decreasing the resolution of the decision variables may cause the characteristics of the Pareto front presented in Figures 4.1 through 4.6 to change.

In general the "tuning" parameters a and b control the length of the continuous portion of the Pareto front and the number of infeasible points. Increasing the b parameter has the effect of increasing the number of infeasible solutions between the feasible solutions on the front. Increasing the a parameter has the effect of decreasing the continuous portion of the front and in some areas yielding discrete points versus a continuous section. This MOP is a contribution to the area of MOP test suites as a researcher can code one function into their MOEA and vary the characteristics of this function. A researcher thereby gains detailed insight into the operation of their MOEA while expending less effort on finding MOPs of varying characteristics.

Numerous constrained MOPs exist in the literature and each of these can be modified in a way similar to what is done for the Tanaka MOP (MOP-CT) to vary the characteristics of the solution sets P_{true} and PF_{true}. Since many of these problems have similar characteristics or can be tuned to exhibit similar characteristics, each of these problems is not presented. One may select an MOP that reflects the class of MOP that they are most interested in solving and modify the MOP in the same manner as MOP-CT proposed.

This allows one to evaluate MOEA performance over a single class of MOP without the researcher having to identify multiple MOPs of this class but instead to identify one MOP of interest and modify it. It is important that researchers test their MOEAs against problems of the class in which they are interested in solving. Additional real-world constrained problems are presented in Sections 4.4 and 4.5 of this chapter.

The best test suite to use in evaluating MOEA performance is one that contains MOPs of similar characteristics to the MOPs that an MOEA is designed for and is being applied to. MOPs 1, 2, 3, 4, and 6 are used for their varying genotype and phenotype characteristics and MOP-CT, the Tanaka test function generator, can be tuned to yield constrained test problems with varying characteristics. The selection of test problems is difficult and there is no best test suite to use. A researcher must understand the problems used to test an MOEA and ensure that these MOPs are testing the MOEA over the types of problems it is designed for. For example, if a researcher has designed an MOEA for solving a single MOP, then the MOEA should be tested against other MOPs with similar characteristics. While the MOPs described up till this point contain relatively low dimensionality, they are useful for comparing the performance of an MOEA against the true solution. Other, larger MOPs are describes and used for testing the performance of the MOMGA-II.

4.3 Deceptive MOPs

An MOEA must identify and manipulate good BB(s) in order to generate PF_{true}. In order to understand the definition of a deceptive MOP used, one must understand the single objective definitions of deception. In single objective EAs, deceptive functions are discussed in terms of BBs [74, 75, 77, 78]. A single objective deceptive function is defined as one in which the global optima is fairly isolated and the local optima is surrounded by good solutions a hamming distance away [76]. Additionally, the deceptive BB is a maximum distance away from the BB necessary to find the global optima.

A deceptive MOP is defined as:

Definition 31 (Deceptive MOP): *A deceptive MOP is one in which at least one of the k objective functions is deceptive.* □

where a single objective function is identified as deceptive if the global optima is fairly isolated and the local optima is surrounded by good solutions a hamming distance away.

An example of a deceptive maximization MOP is presented. Figure 4.7, generated for this dissertation, illustrates a deceptive function, Function 2, and a non-deceptive function, Function 1. The decision variable x is restricted to be an integer and the equations for this MOP are presented.

$$f_1 = \begin{cases} 0.75 - \frac{3}{48}x & 0 \leq x \leq 12 \\ 1 - \frac{1}{3}(x mod 12) & 12 < x < 15 \\ 1 & x = 15 \end{cases} \tag{4.1}$$

$$f_2 = \left\{ 1 - \frac{1}{15}x \quad 0 \leq x \leq 15 \right. \tag{4.2}$$

Graphically, the two objective functions are shown in Figure 4.7 with each plotted as a single objective problem for ease of illustrating the deceptive function. In function 1, an x-value of 0 contains the deceptive BB that leads to a locally optimal value of 0.75; however, an x-value of 15 contains the BB that leads to the global optima of 1.0. The deceptive BB is of order 1 and is represented by the schema *0**. The schema *0** leads to solutions on the left side of Figure 4.7, solutions that lead to the locally optimal solution of 0.75, whereas the BB necessary to find the global optima is of order 2, and is the schema 11**. The schema 11** generates solutions on the right side of Figure 4.7, solutions that lead to the globally optimal solution of 1.0. Since function two is non-deceptive, a BB of order 1, containing the schema *0** leads to the globally optimal solution of 1.0.

Deb also presents an alternate definition of a deceptive MOP [43]. Prior to discussing his definition of a deceptive MOP, it is necessary to present Deb's definition of a local Pareto optimal set. Deb defines both a local and global Pareto optimal set, where the global Pareto optimal set is equivalent to P_{true}. The local Pareto optimal set is defined as [43]:

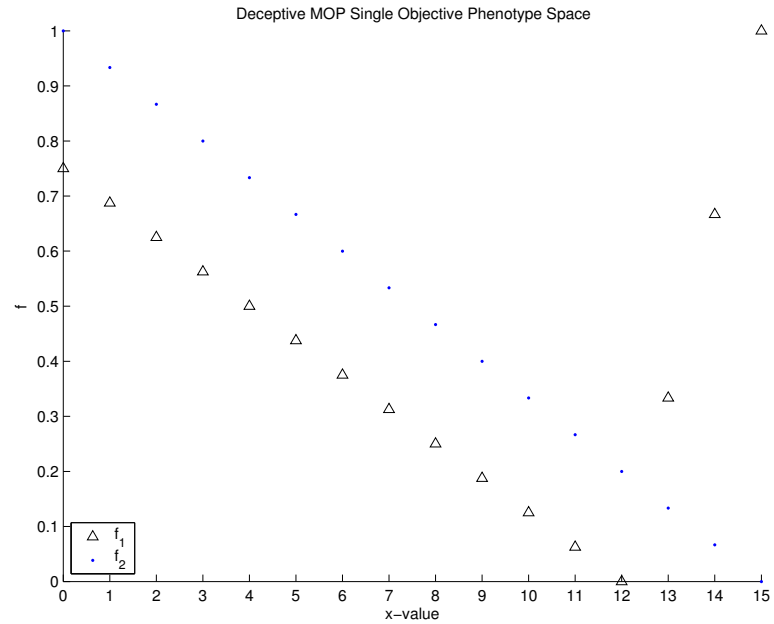

Figure 4.7 Pedagogical Deceptive MOP

Definition 32 (Local Pareto Optimal Set:): *Given some Pareto optimal set $\mathcal{P}$, if $\forall x \in \mathcal{P}$, $\neg\exists y$ satisfying $\| y - x \|_\infty \leq \epsilon$, where ϵ is a small positive number (in principle, y is obtained by perturbing x in a small neighborhood), and for which $F(y) \preceq F(x)$, then the solutions in $\mathcal{P}$ constitute a local Pareto optimal set.* □

This implies that if a set of P_{known} (t) is perturbed slightly, no new nondominated points are found [43]. The definition describes the case where the local front is a physical distance from the true front. The local front is dependent upon the ϵ selected within which solutions are perturbed. Too small an ϵ may result in multiple local fronts being found and too large an ϵ prevents the identification of a local Pareto front.

Deb defines the terms multimodality and deception (known single-objective EA issues) in the multiobjective domain [42]. Deb defines a multimodal MOP as one with multiple local Pareto fronts [42]. This definition is confusing since the term multimodal is defined with respect to single objective optimization when referring to an optimization function containing both local and global minima [184].

Deb defines a deceptive MOP as one in which there are at least two optima (PF_{local} and PF_{true}) and where the majority of the search space favors finding points on PF_{local}.

This concept depends on finding PF_{local}. If PF_{local} does not exist or cannot be found, based on the ϵ chosen, Deb would state that the MOP is not deceptive. However, the existence of PF_{local} is inconsequential to the determination if the MOP is deceptive or not. This determination is based on Definition 31 and the analysis to determine if at least one of the objective functions is deceptive. As long as this criteria is met, the MOP is declared deceptive. This is important as finding the deceptive BBs is necessary to finding the solution to a deceptive MOP. The definition of deception used is consistent with the single objective definition of deception.

Deb states that an MOEA has a hard time finding points on PF_{true} since it gets stuck in the local optima of PF_{local}. Coello Coello et al., state that this may be more an effect of the discretized genotype space than that of the phenotype [31]. This is possible in many MOPs when the discretization process causes the global optima to have fewer local optima near it than another decision variable resolution choice. The issue of discretizing an MOP illustrates the point made in Section 2.5.1 that the discretization process may change the shape of the front or increase the difficulty in finding the front. Additionally, a uniform sampling or discretization of the genotype space does not imply uniform mappings into phenotype space and hence the resultant phenotype space may have all of the solutions concentrated in a single area of the space.

4.4 Discrete MOPs

Discrete MOPs have their own characteristics and may require the use of specialized operators to solve. MOPs containing discrete points on the Pareto front and those containing integer based decision variables can present additional challenges for MOEAs especially when these MOPs are also constrained, as they typically are. The phenotype space of these MOPs is discrete and offers only isolated points, even though when plotted the appearance is given of a continuous front. As only a finite number of solutions exist, only a finite number of corresponding vectors can result.

Nondeterministically polynomial (NP) problems are those problems that a conjectured answer to the problem can be verified in polynomial time but the problem cannot be solved in polynomial time [174]. NP-Hard problems are those problems that are at least

Table 4.3 Possible Multiobjective NP-Complete Functions [31]

NP-**Complete Problem**	**Example**
Traveling Salesperson	Min energy, time, and/or distance; Max expansion
Coloring	Min number of colors, number of each color
Set/Vertex Covering	Min total cost, over-covering
Maximum Independent Set (Clique)	Max set size; Min geometry
Vehicle Routing	Min time, energy, and/or geometry
Scheduling	Min time, missed deadlines, waiting time, resource use
Layout	Min space, overlap, costs
NP-Complete Problem Combinations	Vehicle scheduling and routing
0/1 Knapsack - Bin Packing	Max profit; Min weight

as hard as any other problem in NP. NP-Complete problems are those problems that are NP-Hard and also in NP. Skienna also notes that most NP-Hard problems are also NP-Complete [174].

An MOP test suite of combinatorial problems to include NP-Complete MOPs is a useful concept since some real world MOPs are constrained MOPs utilizing integer based decision variables. Research into solving these NP-Complete MOPs can aid in solving real-world MOPs containing the same characteristics and a NP-Complete test suite is useful to the MOEA community. Table 4.3 outlines a subset of the possible NP-Complete MOPs. Just like other MOPs, NP-Complete MOPs have varying characteristics in their genotype and phenotype domains that make them interesting and useful test problems.

4.4.1 Modified Multiobjective Knapsack Problem. MOPs with integer based decision variables are lacking in current test suites and this makes the Modified Multiobjective Knapsack Problem (MMOKP) a good problem for test suite inclusion. The MMOKP is modeled off of the single objective knapsack problem [174]. In the single objective knapsack problem, a soldier must decide what items to take with him in his knapsack (defined as a rucksack). The soldier has a rucksack to place the items in, where each item has a particular weight and value associated with it and the rucksack has a specific capacity. A value is assigned to each item based on how important it is for the soldier to have that item with him/her in combat. The objective of the soldier is to maximize the value of the

items in the rucksack while meeting the constraint that he or she can only carry a certain amount of weight within the rucksack and that only whole items may be placed within the rucksack. Dependent on the mission, carrying additional water may be of more value than carrying additional ammunition. The multiobjective formulation of this problem uses an arbitrary number of rucksacks where the objective is to maximize the value of each of the rucksacks simultaneously while meeting the constraints.

The modified multiobjective form of this problem, MMOKP, consists of a similar formulation with the exception that there are an arbitrary number of rucksacks [215] as well as associated weights and profits of items with respect to each rucksack. Zitzler refers to this as the extended 0/1 knapsack problem [215] and states that this formulation has been previously used by Sakawa [168]. In reality, this is not an extended version of the knapsack problem as, in the model Zitzler used, each item must be placed into all of the knapsacks or none of the knapsacks. This is not representative of a true formulation of the multiobjective knapsack problem as defined in OR texts [159].

The extended multiobjective knapsack problem is renamed the MMOKP MOP, in this effort, as it does not accurately reflect the traditional multiobjective knapsack problem formulation. In the traditional multiobjective knapsack problem formulation, an item can be placed into only one of the knapsacks and cannot be placed into all of the knapsacks. However the MMOKP formulation constrains each of the items to either be placed into all of the knapsacks or none of the knapsacks, hence each decision variable is referred to as an item type. The MMOKP does, however, reflect some real-world optimization problems. For example, consider the movement of cargo by the military for contingency operations in different parts of the world. In this scenario, there are multiple tanks, artillery, weapons, people and other support items that must be moved. Consider the use of the Air Force to move all of this necessary equipment and personnel for a contingency operation. A number of aircraft (C-17, C-5, C-130 and C-141) are available and used to move the various item types (sets of tanks, equipment, groups of people and additional support equipment). Each of the aircraft have different cargo and fuel storage capabilities as well as different operating costs associated with the number of pilots, co-pilots, load-masters, and other associated support personnel required for operation of the aircraft at the destination and

arrival points. A specific aircraft, the C-17 cargo plane, is selected for movement of the cargo. Additionally it is important to recognize that all of the C-17s used are not located at the same departure point and all of the equipment is not located at the same place as the C-17 used to move the equipment. Therefore, some of the C-17 aircraft are located closer to the starting location of the cargo than others while other C-17 aircraft may be located closer to the destination point of the required cargo. Since the cost of operating each of the aircraft varies greatly as each C-17 is a different distance from the cargo and the contingency location, each of the associated item types to be moved is assigned a weight and value associated with placement into a specific aircraft.

The weight of an item type takes into account the weight of the item type, the amount of fuel required to move the cargo plane from its current location to pick up the cargo, as well as the weight of the fuel required to move the item type in the respective aircraft to its associated destination (how much fuel must be expended to move the aircraft to the cargo location and how much fuel must be expended to move the aircraft to the contingency location). As the item types and aircraft are all located at different locations, the movement of an item type requires a different amount of fuel dependent on the aircraft it is placed in. The method described of defining weights results in a different weight value for each item type dependent on the aircraft it is placed on. The overall weight of all the items placed into an aircraft must not exceed the capacity of that aircraft.

The value of an item type represents how integral that item type or group of people is to the contingency operation along with a measure of the on-time delivery of the item type. For example, assume that it is extremely important for type 1 tanks to arrive at the destination by 2300 hours on Tuesday. A type 1 tank placed into aircraft A, which is destined to complete the cargo movement by 0800 on Tuesday would receive a higher profit than a type 1 tank placed into aircraft B, which does not arrive at the destination until 1 week from Tuesday. Since this example problem requires the movement of as much of each item type as possible, a type 1 tank placed into aircraft A is also placed into all of the aircraft even if the tanks are delivered late. This constraint is dictated by the fact that multiple sets of cargo are required at the destination and they must be sent even if they are received late. Hence one can see that identical items are placed into all of the C-17s

which are represented as knapsacks in the MMOKP. The objective is to simultaneously maximize the value of each aircraft while meeting the weight constraints of the aircraft and the constraint that as much of each item type as possible must be moved. Therefore it is important to place an item into all or none of the knapsacks.

The objective of the MMOKP is to maximize the value of the item types placed into each knapsack while simultaneously satisfying the weight constraint of each knapsack. A commander requests the movement of more item types (items) than can physically fit into the number of planes (knapsacks) available. Therefore one is attempting to determine the set of item types that maximize the profit of each of the airplanes. This is just one example of a potential real-world problem that the MMOKP represents.

In Zitzler's formulation, a weight and profit exists for each item with respect to the specific knapsack it is placed into, i.e., item 21 may be better placed in knapsack 1 versus knapsack 2. The overall goal is to maximize the profit obtained from all of the knapsacks while meeting the weight constraints imposed on each knapsack, where:

$$\begin{aligned} p_{i,j} &= \text{profit of item } j \text{ according to knapsack } i, \\ w_{i,j} &= \text{weight of item } j \text{ according to knapsack } i, \\ c_i &= \text{capacity of knapsack } i \end{aligned}$$

For the MMOKP problem with n knapsacks and m items, the objectives are to maximize

$$f(\mathbf{x}) = (f_1(\mathbf{x}), ...f_n(\mathbf{x})) \tag{4.3}$$

where

$$f_i(\mathbf{x}) = \sum_{j=1}^{m} p_{i,j} x_j \tag{4.4}$$

and where $x_j = 1$ if item j is selected, 0 otherwise [215]. The constraints are:

$$\sum_{j=1}^{m} w_{i,j} x_j \leq c_i \forall\ i \tag{4.5}$$

4.5 Real World MOPs

The classification and analysis of real-world, highly-dimensional genotype and phenotype MOPs allows for the addition of a real-world MOP to the test suite presented in this chapter for MOEAs. The usefulness of this test suite is in determining the characteristics of real-world problems on which an MOEA performs well.

Researchers have conducted a performance analysis of MOEAs utilizing the test suites currently referenced in the literature and mentioned in the this chapter. The problem with this approach is that many of these test MOPs are not realistic when compared to the MOPs encountered in different application areas or in the real-world. A limited test suite that reflects some of the characteristics of MOPs in varying problem areas would be beneficial to the community. The selection of the problem areas and highly dimensional problems would take a thorough analysis of the problem domain aspects and is specific to the objective of a researcher. If a researcher is interested in solving nonlinear real-world problems of a specific application, then the researcher should use MOPs that are indicative of the characteristics of this problem. The classification of every real-world MOP based on the MOPs' varying characteristics and complexity of their fitness functions and constraints is impossible. Too many MOPs exist in the real-world to solve. Some of the application areas include Unmanned Aerial Vehicle (UAV), Unmanned Combat Aerial Vehicle (UCAV), Micro Aerial Vehicle (MAV) Routing and Control, Groundwater Problem, Image Compression, Antenna Design, Target Tracking, Advanced Logistics Problem (ALP), MEMs Applications, and Intrusion Detection.

The work presented is of interest to the Air Force and other organizations. As stated previously, optimization problems exist in any operational setting. Many of the applications and methods that are used to complete tasks are not currently solved to an optimal solution. Generating good solutions to many real-world optimization problems would benefit a wide variety of organizations including the Air Force. A limited number of application areas are presented in this section with the mathematical model included to illustrate the complexity associated with these problems.

Currently, good solutions have been found to many real-world problems but better solutions may exist. The difference between the current known good solution and a better one may be only 1% or may be 50% or more. This can translate to a huge dollar savings depending on the application or a huge savings in the amount of time it takes to complete a task.

Defining accurate models of the real-world application of interest is extremely important. First the level of accuracy or resolution of the solution required must be known (i.e., how many bits to use to represent a decision variable) or else there is no way to compare the results that the MOEA finds to other methods. Additionally the model must be correct and validated in order to ensure the solutions found are feasible and the MOEA is given an opportunity to find the real solutions at the desired resolution. The complexity of the model directly effects the execution time (efficiency) of the MOEA, therefore the least complex model that produces valid results is what must be chosen.

A real-world application, the Advanced Logistics Problem MOP, is selected for inclusion in an MOP test suite. A description of the problem domain is provided to aid the reader in obtaining an understanding of the characteristics of the ALP problem. This problem is selected as it is a real-world problem with a large number of decision variables, it is constrained, and it is a discrete MOP. Discrete MOPs are lacking in other proposed test suites. Since many real-world problems involve the use of discrete decision variables, this is a good MOP to use in a test suite.

4.5.1 Advanced Logistics Problem (ALP). The Advanced Logistics Problem (ALP) involves logistics research in resource allocation. ALP is an effort by the Defense Advanced Research Projects Agency (DARPA) to develop a distributed computing architecture linking current and future logistics information systems to the crisis action planning processes [21]. This is the development of a real-time end-to-end system linking operations and logistics. The goal is to develop a system which gives planners the ability to review multiple plans in relation to real-time up to date data and effectively choose the best plan for the situation. Essentially a choice must be made by a commander, given the operational situation, of which mix of resources is preferred to support the operation.

A mission ready resource (MRR) is a combination of an asset type and its associated resources. This includes aircraft, tanks, support personnel and equipment. These resources are assigned a suitability value for use in a specific plan [199]. An example of an MRR is a C-130 aircraft. This MRR is available in a number of different configurations, where each configuration is a MRR type, and the number of times that an MRR can be used in a given day is also important. This information is used to assign the MRR a suitability value. A combination of MRR types is defined to be a MRR set or resource mix. Different MRR combinations provide different capabilities and suitability values to the commander as well as require different amounts of lift resources to operate. Two of the objectives of this problem are to maximize asset suitability and minimize lift consumption [199, 216].

The ALP problem formulation is based on Swartz's work on the ALP Pilot problem [180]. The mathematical MOP Formulation is given m tasks and n MRR types, and the solution set is an m x n matrix. A matrix element is an integer based decision variable, $x_{i,j}$, that represents the number of MRRs of type j allocated to task i. Assuming that each task is satisfied by exactly one MRR, and that no interactions exist between differing MRR types, then the suitability, S, for all MRRs is defined by [31, 199, 216]:

$$S = \sum_{j=1}^{n} \sum_{i=1}^{m} a_{i,j}\ x_{i,j} \tag{4.6}$$

where $a_{i,j}$ is the suitability of MRR j for Task i and $x_{i,j}$ is the number of MRRs j allocated to task type i.

Since the desired capability for a task is set by the decision maker and defined to be static, Equation (4.7) is an equality constraint. The requirement that all tasks $i = 1, \ldots, n$ must be satisfied at a particular resource level (RL) k is:

$$\sum_{j=1}^{n} x_{i,j} = RLtask_{k,i} \tag{4.7}$$

In this application, the decision variables are allowed to take on any non-negative integer value so long as they do not exceed the specified resource level. Therefore Equation (4.8) is an inequality constraint. The requirement that all of the MRR types, $j = 1, 2, \ldots, n$,

do not exceed their available number at a particular resource level k is:

$$\sum_{i=1}^{m} x_{i,j} \leq RLmrr_{k,j} \tag{4.8}$$

The maximum number of efforts per day for a particular asset is A and A has P configurations corresponding to P MRR types.

It is difficult to determine what the actual logistical footprint is for a given asset set [199]. At the very least, it is clear that for each additional asset deployed, there is a corresponding increase in cost for additional resources such as fuel and supplies. Assuming that consumption is linear and without interaction, the weight consumption, W, and volume consumption, V, for all MRRs are

$$W = \sum_{j=1}^{n} \sum_{i=1}^{m} \beta_j \; x_{i,j} \tag{4.9}$$

and

$$V = \sum_{j=1}^{n} \sum_{i=1}^{m} \lambda_j \; x_{i,j} \tag{4.10}$$

where β_j and λ_j are the weight and volume consumed by a single MRR j.

The form of the *suitability maximizing / lift minimizing MOP* with A asset types, m tasks, n MRR types, at a resource level k, and decision variables $(x_{1,1}, x_{i,j}, \ldots, x_{m,n})$ is to maximize:

$$S = \sum_{j=1}^{n} \sum_{i=1}^{m} a_{i,j} \; x_{i,j} \tag{4.11}$$

minimize:

$$W = \sum_{j=1}^{n} \sum_{i=1}^{m} \beta_j \; x_{i,j} \tag{4.12}$$

and minimize:

$$V = \sum_{j=1}^{n} \sum_{i=1}^{m} \lambda_j \; x_{i,j} \tag{4.13}$$

subject to:

$$\sum_{j=1}^{n} x_{i,j} = RLtask_{k,i} \text{ for } i = 1 \ldots m \tag{4.14}$$

$$\sum_{i=1}^{m} x_{i,j} \leq RLmrr_{k,j} \tag{4.15}$$

$$\text{for } A = 1 \ldots a \text{ and } P_a = \text{ number } MRR \text{ types for } a$$

The number of constraints resulting from Equation (4.14) is equal to the number of tasks. These constraints ensure that the total number of efforts for Task i is exactly the desired capability at that resource level. The maximum value for any decision variable is found by using Equation (4.14) and allocating all task capability to one MRR type. The number of constraints resulting from Equation (4.15) is equal to the number of MRR types. These constraints ensure that no MRR type can be allocated a number of efforts that exceeds the given resource level. These constraints are also used when there are restrictions on the available number of any MRR type, e.g. attrition or changes in asset turn rate. It is important to note that each constraint refers to a single MRR type.

The specific number of tasks in Table 4.4 and MRR types in Table 4.8, along with their task suitabilities, are defined. The suitabilities reflect notional but reasonable values that clearly differentiate the MRR types. The same can be said for the consumption values in Table 4.8. To keep the number of task capability decisions by the decision maker at a reasonable level, five resource levels in Table 4.5 are specified, equating to 15 separate task preference decisions as specified in Wakefield [199]. These preferences are reflected in Table 4.6. When the ratios are applied to their respective resource level values, the result is the capability matrix in Table 4.6. The values are rounded to a whole number so that the sum across tasks is equal to the resource level. The values of the resource levels were chosen to create solution spaces of increasing size. Given three tasks and five MRR types and a resource level of 300 efforts per day, the worst case number of possible resource mixes is approximately 9.72 x 10^{19}. Wakefield's formulation is such that for the target MOP, it is assumed that there is no restriction on the available number of any MRR type; no attrition; and that each asset has one associated MRR type, i.e., one effort per day [199].

These simplifying assumptions are made to provide an opportunity to explore the solution and complexity.

Table 4.4 Tasks

Index	Nomenclature
1	Air-to-Air (AA)
2	Air-to-Ground (AG)
3	Precision Locating (PL)

Table 4.5 Resource Levels

Index	RL (efforts per day)
1	16
2	32
3	75
4	150
5	300

Table 4.6 Desired Task Capability Ratios

	Percent to Task		
Index	AA	AG	PL
1	60	30	10
2	30	60	10
3	25	60	15
4	20	50	30
5	20	30	50

The complete MOP formulation is as follows: Decision variables - Number of MRR j assigned to Task $i = (x_{1,1}, \ldots, x_{i,j})$

Maximize:

$$\begin{aligned} S &= 0.8x_{1,1} + 0.3x_{1,2} + 0.6x_{1,3} + 0.001x_{1,4} + 0.001x_{1,5} \\ &+ 0.4x_{2,1} + 0.8x_{2,2} + 0.6x_{2,3} + 0.001x_{2,4} + 0.001x_{2,5} \\ &+ 0.001x_{3,1} + 0.001x_{3,2} + 0.1x_{3,3} + 0.8x_{3,4} + 0.4x_{3,5} \end{aligned} \tag{4.16}$$

Table 4.7 Desired Capability Matrix

	TASK (efforts per day)			
Index	AA	AG	PB	Decision Space Cardinality
1	10	5	1	630,630
2	10	20	2	159,549,390
3	19	45	11	$\approx 2.56x10^{12}$
4	30	75	45	$\approx 1.48x10^{16}$
5	60	90	150	$\approx 4.37x10^{19}$

Table 4.8 Task Suitability / Lift Consumption Matrix

footnotesize

		Task Suitability			Lift Consumption	
Index	MRR Type	AA	AG	PL	Weight (Short Tons)	Volume (Cubic feet)
1	F_A	0.800	0.400	0.001	20.2	1650.0
2	F_B	0.300	0.800	0 001	28.5	2475.0
3	F_C	0.600	0.600	0.100	35.7	2887.5
4	B_1	0.001	0.001	0.800	19.9	1705.0
5	B_2	0.001	0.001	0.400	22.5	2200.0

Minimize:

$$\begin{aligned} W &= 20.2(x_{1,1} + x_{2,1} + x_{3,1}) + 28.5(x_{1,2} + x_{2,2} + x_{3,2}) \\ &+ 35.7(x_{1,3} + x_{2,3} + x_{3,3}) + 19.9(x_{1,4} + x_{2,4} + x_{3,4}) \\ &+ 22.5(x_{1,5} + x_{2,5} + x_{3,5}) \end{aligned} \tag{4.17}$$

and minimize:

$$\begin{aligned} V &= 1650(x_{1,1} + x_{2,1} + x_{3,1}) + 2475(x_{1,2} + x_{2,2} + x_{3,2}) \\ &+ 2887.5(x_{1,3} + x_{2,3} + x_{3,3}) + 1705(x_{1,4} + x_{2,4} + x_{3,4}) \\ &+ 2200(x_{1,5} + x_{2,5} + x_{3,5}) \end{aligned} \tag{4.18}$$

subject to:

$$
\begin{aligned}
(x_{1,1}, \ldots x_{3,5}) &\geq 0 && (4.19)\\
(x_{1,1}, \ldots x_{3,5}) &\in I(integers) && (4.20)\\
x_{1,1} + x_{1,2} + x_{1,3} + x_{1,4} + x_{1,5} &= RLtask_{m,1} && (4.21)\\
x_{2,1} + x_{2,2} + x_{2,3} + x_{2,4} + x_{2,5} &= RLtask_{m,2} && (4.22)\\
x_{3,1} + x_{3,2} + x_{3,3} + x_{3,4} + x_{3,5} &= RLtask_{m,3} && (4.23)\\
x_{1,1} + x_{2,1} + x_{3,1} &\leq RLmrr_{m,1} && (4.24)\\
x_{1,2} + x_{2,2} + x_{3,2} &\leq RLmrr_{m,2} && (4.25)\\
x_{1,3} + x_{2,3} + x_{3,3} &\leq RLmrr_{m,3} && (4.26)\\
x_{1,4} + x_{2,4} + x_{3,4} &\leq RLmrr_{m,4} && (4.27)\\
x_{1,5} + x_{2,5} + x_{3,5} &\leq RLmrr_{m,5} && (4.28)
\end{aligned}
$$

where m is the Resource Level index for the current problem.

4.6 Summary

MOPs, their characteristics, and real-world applications are all important to understand. An understanding of these MOPs allows a researcher the ability to evaluate an MOEA's performance over a class of problems indicative of those the MOEA is applied to. This chapter presents constrained, discrete, NP-Complete and real-world MOPs for inclusion in an MOP test suite. These constrained MOPs are largely missing from existing test suites. Constrained problems have only recently been considered for testing MOEAs. A constrained test function generator is included in the MOP test suite. This MOP, MOP-CT, is formulated in a manner that makes it relatively easy to modify the characteristics and the difficulty that MOEAs have in attempting to solve this MOP.

Discrete problems are another class of MOP missing from existing MOP test suites. The MMOKP and ALP problems are both included in the MOP test suite. These problems are constrained and have a large number of integer based decision variables. Both of these MOPs are indicative of real-world MOPs. Additionally, a discussion of existing MOP test

suites and MOP test suite generators is included. The MOPs included in the test suite presented in this chapter are important. The application of MOEAs to this test suite can provide good information about the MOP characteristics that yield good results from various MOEAs.

The use of MOP test suites and metrics aids in understanding the effect that BBs have on solving MOPs and in defining a level of MOP difficulty based on the types and sizes of BBs that must be found in order to generate a "good" solution to an MOP. Using the concept of BBs, one can develop equations to determine the population sizes that are necessary to obtain "good" solutions for an MOP utilizing a BB-based approach. The development of an MOEA population sizing equation advances the theoretical contribution of this work to the MOEA community. This equation must be defined to support and retain the use of good BBs in the population. MOEA population sizing is discussed in Chapter VIII. The next chapter discusses the existing MOEA theory and identifies areas in which the current theoretical development is lacking. Additionally, symbolic formulations are presented for various MOEA operators.

V. MOEA Symbolic Formulations

The design of an MOEA and advancement of MOEA theory requires an in-depth understanding of the existing theory and background. Previous chapters discuss MOEA terminology and current MOEA theory. In this chapter, symbolic formulations are presented for MOEA operators. To advance the state of the art and contribute to increasing the efficiency and effectiveness of MOEAs, a researcher must analyze the foundation of MOEAs, determine what leads to good or bad performance, and develop a clear understanding of MOEA operators. Currently, there is only a limited amount of published theoretical analysis of MOEAs, which mostly concentrates on the area of MOEA convergence properties [85, 163, 167, 184]. The advancement of MOEA theory can yield a better understanding of how and why MOEAs work and improve their overall performance. A first step to advance the theory involves the understanding of MOEA operators. To aid researchers in understanding these operators, symbolic formulations of MOEA operators is presented.

This chapter presents generalized symbolic representations for MOEAs and associated MOEA operators. The general concepts presented in Chapters II and III are used to classify MOEAs and present symbolic formulations of their operators. Development of a symbolic representation for MOEAs is necessary to understand and develop efficient and effective MOEAs. This chapter presents new symbolic formulations for MOEAs to serve as a basis for the research conducted and increase the understanding of MOEAs.

5.1 Multiobjective Evolutionary Algorithm Operators

Symbolic formulations of MOEAs and their associated operators are rarely, if at all, discussed. In order to develop a deeper understanding of MOEAs, the symbolic formulations of MOEAs and a subset of their operators are developed and presented, utilizing the notation of Bäck [13], Merkle [132], and Van Veldhuizen [184] for consistency and ease of understanding. The usefulness of symbolic formulations of MOEA operators and algorithm definitions lies in providing a deeper understanding of the MOEAs. Researchers who understand the symbology hopefully use it in the design of their MOEAs. The mapping from

the symbolic notation to the actual MOEA implementation can yield more efficient and effective MOEAs. The single objective symbolic definitions are presented in Appendix A.

An MOEA's foremost difference from a single objective EA is the k objectives that are to be optimized simultaneously. MOEAs typically differ from EAs in their use of a modified selection mechanism to reflect a Pareto based selection and the use of niching and crowding mechanisms to obtain a good distribution of points on the Pareto front [44]. All of these issues can result in additional processing requirements.

Recombination, selection, and mutation are the main operators currently used in MOEAs. Other operators, in addition to the recombination, selection, and mutation operators, may have a substantial effect on the results of the algorithm [46, 48]. Some of the operators that must be analyzed include secondary population storage schemes, subpopulations, niching mechanisms and mating restrictions. This section presents the symbolic formulations for each of these operators. A more in-depth description of MOEA operators is presented in Chapter II, Sections 2.6 through 2.10.

Archiving or storing a secondary population is typical in MOEAs to prevent the loss of "good" solutions generated throughout the search process. Archiving strategies involve the process of identifying nondominated points in the current population of each generation t of an MOEA. A Pareto selection operator is presented in Definition 33. Following recombination and mutation, the Pareto selection operator is responsible for identifying the nondominated points that exist in the current population and are members of the set $PF_{current}(t)$. Pareto selection analyzes the current population of size μ in generation i and selects the nondominated population members for inclusion in the set $PF_{current}(t)$. The $\mathcal{T}_p$ operator determines which population members are nondominated.

Definition 33 (Pareto Selection):

$$s^{(i)} : \mathbb{X}_s^{(i)} \times \mathcal{T}(I, \mathbb{R}^k) \longrightarrow \mathcal{T}\left(\Omega_s^{(i)}, \mathcal{T}_p\left(\left(I^{\mu'^{(i)}+\chi\mu^{(i)}}\right), PF_{current}(t)\right)\right),$$

$$\mathcal{T}_p\left(\left(I^{\mu'^{(i)}+\chi\mu^{(i)}}\right), PF_{current}(t)\right) =$$

$$\{\vec{u}_m \big| \vec{u}_m \preceq \vec{u}_n given \vec{u}_m, \vec{u}_n \in \left(I^{\mu'^{(i)}+\chi\mu^{(i)}}\right)\}.$$

□

One method of storing population members in an archive is to append the nondominated points from the current generation t, $PF_{current}(t)$, into the archive set. The archive contains the nondominated points found from the beginning of MOEA execution through the previous generation $t-1$. After appending the population members, one must determine the points in this combined archive that are nondominated. The nondominated points in the archive are identified as $PF_{known}(t-1)$. This process is repeated at the conclusion of each generation (i.e., $PF_{current}(t) \cup PF_{known}(t-1)$ and Pareto Selection is applied to this combined intermediate set). Definition 33 illustrates the process of identifying the points that belong in the set $PF_{current}(t)$ for generation t. Subsequently, all of the points in $PF_{current}(t)$ are combined with the points in $PF_{known}(t-1)$ and the combined set is analyzed to identify the nondominated points. Definition 34 presents the symbolic formulation for the Pareto front set combination operator (PFSCO).

Definition 34 (Pareto Front Set Combination Operator):

$$PFO := \mathcal{T}\left(\left(PF_{known}(i-1)\bigcup PF_{current}(t)\right), PF_{known}(t)\right)$$

$$PF_{known}(t) :=$$

$$\{\vec{u}_m | \vec{u}_m, \vec{u}_n \in \left(PF_{known}(t-1)\bigcup PF_{current}(t)\right), \vec{u}_m \preceq \vec{u}_n\}$$

□

The PFSCO combines the known Pareto front through generation $t-1$, $PF_{known}(t-1)$ with the current Pareto front for generation t, $PF_{current}(t)$. The new archive set, after application of the PFSCO operator, contains only the nondominated members found through generation t, $PF_{known}(t)$, and this set becomes the new archive set. The PFSCO operator can be used for combining any two sets and identifying the points in the combined set that are nondominated.

At any given point in time, one can analyze the archive to determine the nondominated points that the MOEA has found through generation t, $PF_{known}(t)$. Another archiv-

ing option is to append all of the population members from each generation, $P(t)$ or the current nondominated points from each generation, $PF_{current}(t)$, into an archive for post-processing. At the termination of the MOEA, the archive must be analyzed to determine which of the members are nondominated and belong in the set PF_{known}. Remember that the set PF_{known} contains the best points that the MOEA generated throughout its search process. This archiving method (post-processing) results in a PF_{known} set identical to the previous method of constantly analyzing new population members for inclusion in the archive.

The difference between the two archiving methods lies in the overall memory requirements of archiving. The post processing method typically requires more memory as duplicated points as well as dominated points remain in the archive until the termination of the MOEA. At termination, the duplicate points as well as the dominated points are removed and the nondominated points are presented in the set, PF_{known}. Since the duplicate and dominated points remain in the archive until MOEA termination, this can take up a substantial amount of memory dependent on the archive size. The archiving method that continuously removes duplicate and dominated population members from the archive typically requires less memory as only the nondominated points are present in the archive at any given time. The two archiving methods described require the same amount of memory in the situation where every point generated is a nondominated point and duplicate points are not found. In this example, both archiving strategies would store the same number of points in the archive each generation and and use the same amount of memory resources. Overall, the selection of an archiving method to use is dependent on the resources available and if a requirement exists for knowledge of $PF_{known}(t)$ following each generation as both methods result in an identical solution set, PF_{known}.

5.2 Subpopulation Based Approaches

In comparison to a single objective EA, the greatest difference between an EA and an MOEA is the use of multiple fitness functions and a multiobjective selection mechanism in the MOEA. Multiple options exist for handling the k fitness functions existing in an MOP formulation. In the simplest case, as in the VEGA of Section 3.1.1.1, an MOEA uses

multiple subpopulations of individuals. This is a natural evolution from the aggregated fitness function approach and may be considered a similar type of approach. Each of the k subpopulations maintain a population of individuals that have high quality fitness values with respect to one fitness function. In the VEGA, the same number of individuals, j, of high fitness value for each of the fitness functions, are selected from the results of the previous generation. These subpopulations are then shuffled, followed by crossover and mutation and the process repeats. The selection mechanism for this class of algorithms is modeled off of the VEGA [170]:

Definition 35 (Subpopulation Based Pareto Selection):

$$s^{(i)} : \mathbb{X}_s^{(i)} \times \mathcal{T}(I, \mathbb{R}^k) \longrightarrow \mathcal{T}\left(\Omega_s^{(i)}, \mathcal{T}\left(\bigcup_{j=1}^{k}\left(I^{\mu_j'^{(i)}+\chi\mu_j^{(i)}}\right), I^{\mu^{(i+1)}}\right)\right),$$

where $I^{\mu_j'^{(i)}}$ and $I^{\mu_j^{(i)}}$ are equal to $\frac{I^{\mu'^{(i)}}}{k}$ and $\frac{I^{\mu^{(i)}}}{k}$, i.e., each of the k subpopulations are of the same cardinality. □

One can see that this is almost identical to the single objective selection mechanism. A single objective elitist (individuals with the highest quality fitness values are chosen first) selection mechanism is applied k times, where the number of individuals selected for each fitness function is equal to the offspring population size divided by the number of fitness functions. All of the selected individuals are subsequently combined and form the next population, upon which the evolutionary operators are applied.

Disadvantages exist in this approach. A separation of the fitness functions does not allow for a simultaneous optimization of all of the fitness functions as a Pareto based approach achieves. Additionally, the survival pressure is only exerted on one dimension of the k objective functions within each subpopulation. This may lead to the creation of species within each subpopulation and can yield similar results to optimizing each fitness function individually and then combining the results [170]. Hence a Pareto based approach with the use of archiving is recommended for attempting to solve MOPs.

5.3 Ranking

Rank based fitness assignments were first proposed by Goldberg [74] and have since been used in a number of MOEAs [61, 64, 66, 68, 92, 176, 213, 183]. In general, this form of fitness assignment assigns a rank to each individual based on the number of individuals that dominate it. There are a number of MOEAs that use this method or a slightly modified method for calculating the ranks. Definition 36 presents a general symbolic formulation for ranking, in which a rank is assigned to each individual based on the number of individuals that dominate it, as proposed by Fonseca [64]. This definition can be easily modified to reflect the slight differences in calculating the rank seen in other MOEAs.

Definition 36 (Rank Assignment): *Let* $a_m \in P(i)$ *where* $P(i) := \{a_1(i), a_2(i), \ldots, a_\mu(i)\} \in I^{\mu^{(i)}}$ *and* $m < \mu$ *(*a_m is an individual of the current population with fitness vector $\vec{u}_m$*),* The rank of population member a_m, present in generation i, is:

$$R_i(a_m) = \sum_{n=1}^{\mu^i - 1} \left| \left(\vec{u}_m \preceq \vec{u}_n \right) \right|$$

The rank of a_m is equal to the cardinality of population members in population P_i that dominate member a_m. □

5.4 Niching Based Approaches

Niching operators have been used in a number of MOEAs [47, 64, 69, 93, 94, 176, 184]. Niching operators are used in an attempt to generate a uniform distribution of points across the Pareto surface they form. Niching can be integrated with the fitness function in fitness sharing to penalize solutions that are within a certain distance of each other. The symbolic formulation for fitness sharing with niching, as used in Horn [93], is presented in Definition 37.

Definition 37 (Fitness Sharing and Niching): *If* a *and* $b \in P(i)$ *and* $\mathcal{D}(a, b) \leq \sigma_{share_i}$*, where* $\mathcal{D}$ *is the genotype or phenotype distance between points* a *and* b*, then the fitness is modified such that* $\Phi_i(a) = \Phi_i(a) - X$ *and* $\Phi_i(b) = \Phi_i(b) - X$*, where* X *is determined by the distance* $\mathcal{D}$ *between the two population members.* □

Typically niching is used in conjunction with the selection operator to aid in obtaining a good distribution of points along the Pareto surface. A distance metric $\mathcal{D}$ is used in order to determine how far apart population members reside in relation to other population members. A distance metric can be used in isolation as presented in the fitness sharing definition, Definition 37, or can be used with a niche count, n_i to allow up to a specified number of solutions to reside in a specific area of the space as used by Horn [94].

Definition 38 (Niche Count): *n_i is a count of the number of population members within a portion (niche number i) of a R^n (genotype) or R^k (phenotype) dimensional space. i.e., A count of the number of population members within a specified portion of the $P_{current}$ (t), P_{known} , $PF_{current}$ (t), or PF_{known} space.* □

Use of the niching operator in conjunction with the selection operator restricts which solutions are allowed to pass to the next generation [94]. Niching based selection is defined as:

Definition 39 (Niching Based Selection): *Two nondominated population members a and b are considered for niching based selection. The population member assigned the lowest niche count ($min\,(n_i(a), n_i(b))$) is selected. $n_i(a)$ is the niche count of the niche in which population member a resides. The population member with the lowest niche count is selected.* □

Definition 39 applies to the genotype or phenotype domains. Niching can also be used to reduce the size of the final PF_{known} set that the MOEA generates, in which members are only added to PF_{known} if they would enter an uncrowded niche or they have a lower niche count than at least one member of PF_{known} . If the cardinality of PF_{known} is equal to its maximum size, an element with a larger niche count must be removed from PF_{known} . This is useful in conjunction with a fixed size archive and is defined as:

Definition 40 (Niche Based Archiving): *Let population member $a_m \in I^{\mu^{(i)}}$ having fitness vector $\vec{u}_m$ be considered for placement into PF_{known} . If PF_{known} is restricted in size to PF_{known_max} and $\left|PF_{known}\right| < PF_{known_max}$, $PF_{known} = (PF_{known} \bigcup a_m)$; else*

$PF_{known} = (PF_{known} \bigcup a_m)$ *iff:*

$$\{\vec{u}_m \big| n = \{1, \ldots, \mu^i\}, \vec{u}_n \in PF_{known}, \vec{u}_m \preceq \vec{u}_n\}$$

and:

1. $a_m \in$ *of an empty niche* OR
2. a_m *is placed into a non-empty niche,* N*, given* $n_i(a) < max(n_i)$

and population member b *is removed, where* $n_i(b) = max(n_i)$. □

5.5 Mating Restrictions and Comparison Sets

Restricted mating of MOEA population members is yet another area of interest. The concept is to restrict the mating between individuals within a population unless they are "close" enough or "far" enough away from each other with respect to their genotype or phenotype values [64]. Restricting the mating process creates different "species" within the population. Horn presented the idea of using a portion of the population, t_{dom}, as a comparison set [93]. This comparison set is used in cases where the selection operator must decide between two nondominated individuals. Definition 41 presents this concept.

Definition 41 (Pareto Comparison Set Based Selection): $a_m, a_j, \{p_{c_{t_{dom}}} \subseteq P(i)\} \in I^{\mu^{(i)}}$ *are randomly chosen, where* p_c *is the randomly selected comparison set and* $t_{dom} = \big|\{p_c\}\big| \leq \big|P(i)\big|$. *If* $n = \{1, \ldots, t_{dom}\}, \vec{a}_n \in I^{\mu^{(i)}}$,

$$\vec{u}_m \preceq \vec{u}_n \text{ and } \vec{u}_j \not\preceq \vec{u}_n \text{ then select } a_m$$

Else if

$$\vec{u}_m \preceq \vec{u}_n \text{ and } \vec{u}_j \preceq \vec{u}_n \text{ or } \vec{u}_m \not\preceq \vec{u}_n \text{ and } \vec{u}_j \not\preceq \vec{u}_n$$

Then use Definition 39 to select a_m *or* a_j □

Similar mating restriction methods are presented in the literature and discussed in Coello Coello et al. [31]. Some accomplish this through the use of a grid based structure,

only allowing mating to occur within some area, while others label the population members by sex and only allow sexual reproduction to occur.

5.6 Multiobjective Evolutionary Algorithm Formulation

One can present a generalized symbolic formulation for an MOEA once a symbolic formulation for the operators used in an MOEA is defined. Such a formulation aids researchers in implementing an MOEA correctly by following the predefined description. Additionally the formulation clarifies any misunderstandings of the MOEA operators.

A generic MOEA formulation is presented, using consistent notation with the EA formulation presented in Section A.2:

Definition 42 (Multiobjective Evolutionary Algorithm): *Let*

- *I be a non-empty set (the individual space),*
- *$\{\mu^{(i)}\}_{i\in\mathbb{N}}$ a sequence in $\mathbb{Z}^+$ (the parent population sizes),*
- *$\{\mu'^{(i)}\}_{i\in\mathbb{N}}$ a sequence in $\mathbb{Z}^+$ (the offspring population sizes),*
- *$\Phi : I \longrightarrow \mathbb{R}^k$ (k fitness functions),*
- *$\iota : \bigcup_{i=1}^{\infty}(I^{\mu})^{(i)} \longrightarrow \{\texttt{true}, \texttt{false}\}$ (the termination criterion),*
- *$\chi \in \{\texttt{true}, \texttt{false}\}$,*
- *r a sequence $\{r^{(i)}\}$ of recombination operators*
  $r^{(i)} : \mathbb{X}_r^{(i)} \longrightarrow \mathcal{T}\left(\Omega_r^{(i)}, \mathcal{T}\left(I^{\mu^{(i)}}, I^{\mu'^{(i)}}\right)\right)$,
- *m a sequence $\{m^{(i)}\}$ of mutation operators*
  $m^{(i)} : \mathbb{X}_m^{(i)} \longrightarrow \mathcal{T}\left(\Omega_m^{(i)}, \mathcal{T}\left(I^{\mu'^{(i)}}, I^{\mu'^{(i)}}\right)\right)$,
- *s a sequence $\{s^{(i)}\}$ of selection operators*
  $s^{(i)} : \mathbb{X}_s^{(i)} \times \mathcal{T}(I, \mathbb{R}^k) \longrightarrow \mathcal{T}\left(\Omega_s^{(i)}, \mathcal{T}_p\left(\left(I^{\mu'^{(i)}+\chi\mu^{(i)}}\right), I^{\mu^{(i+1)}}\right)\right)$,
- *$\Theta_r^{(i)} \in \mathbb{X}_r^{(i)}$ (the* recombination parameters*),*
- *$\Theta_m^{(i)} \in \mathbb{X}_m^{(i)}$ (the* mutation parameters*), and*
- *$\Theta_s^{(i)} \in \mathbb{X}_s^{(i)}$ (the* selection parameters*).*

Then the algorithm shown in Figure 5.1 is called a Multiobjective Evolutionary Algorithm.

□

$t := 0;$
initialize $P(0) := \{\mathbf{a_1}(0), \ldots, \mathbf{a}_\mu(0)\} \in I^{\mu^{(0)}};$
while $(\iota(\{P(0), \ldots, P(t)\}) \neq \texttt{true})$ **do**
  recombine: $P'(t) := r^{(t)}_{\Theta_r^{(t)}}(P(t));$
  mutate: $P''(t) := m^{(t)}_{\Theta_m^{(t)}}(P'(t));$
  select:
    if χ
      then $P(t+1) := s^{(t)}_{(\theta_s^{(t)}, \Phi)}(P''(t));$
      else $P(t+1) := s^{(t)}_{(\theta_s^{(t)}, \Phi)}(P''(t) \cup P(t));$
    fi
  $t := t + 1;$
od

Figure 5.1 Multiobjective Evolutionary Algorithm Outline

Definition 42 differs from the single objective definition, Definition 54, in the basic sense that there are k objectives to be solved and therefore there are k fitness functions. Additionally, the MOEA selection operator uses the concept of Pareto dominance and niching to aid in obtaining an even distribution of points across the Pareto front. A generic formulation of an MOEA aids researchers in understanding the basic structure of an MOEA. The formulations of each MOEA operator provides for a more in-depth understanding of MOEAs and their operators.

5.7 Summary

This chapter presents symbolic formulations in the form of definitions of MOEA operators. These definitions are important and aid researchers in correctly implementing MOEA operators. A generic MOEA formulation is presented using the definitions developed earlier in this chapter. The associated definitions aid in developing an in-depth understanding of MOEA operators and a fuller understanding of the operation of MOEAs. Once a researcher reaches this level of understanding, he or she has the "tools" necessary to develop MOEAs as well as their operators and potentially contribute to advancing MOEA

theory. The next chapter builds upon the understanding developed through the previous chapters to develop a new explicit BB-based MOEA. An understanding of the background information presented is crucial in designing, implementing, and testing MOEAs.

VI. MOMGA-II Development

The goal of this research effort is to advance the state of the art with respect to explicit BB-based MOEAs. This goal requires an in-depth understanding of existing explicit BB-based approaches in order to recognize the advantages and disadvantages of such approaches and subsequently improve upon them. Two of the objectives of this effort are addressed: the development of an explicit BB-based MOEA for solving real-world constrained MOPs, and the demonstration that explicit BB-based MOEAs provide insight into difficult MOPs that other approaches may not provide.

The first objective can be satisfied either through the use of existing MOEAs or through the development of an entirely new MOEA. Since many MOEAs exist, choosing an existing MOEA would be of some utility as most of the coding would already be completed. Creating a new MOEA from scratch involves possibly re-creating existing code or implementing new code but has the advantage that inefficiencies in existing approaches can be addressed and improved upon. The second objective is more difficult to realize and involves the testing of various MOPs in order to demonstrate that additional insight is be gained into difficult MOPs through the use of explicit BB-based MOEAs.

6.1 Introduction

Many EAs and one MOEA follow the concepts in the Building Block Hypothesis (BBH) presented in Chapter II. To summarize, the BBH states that BBs exist in a population of individuals. If an EA (MOEA) can identify the "good" BBs and manipulate them through the use of Evolutionary Operators (EVOPS), the EA (MOEA) has an increased probability of generating "good" solutions to the problem. Considerable research has been conducted into the explicit identification of BBs in single objective EAs [36, 77, 78, 111, 132, 142] and recently limited research into their use in MOEAs [184]. These EA and MOEA approaches have shown impressive results and hence conducting additional research into explicit BB-based MOEAs is justified.

In order to meet the objectives of this work, an explicit BB-based MOEA must be used. The only existing explicit BB-based MOEA, the MOMGA, could be used or an

entirely new MOEA could be created. While the MOMGA has been shown to be effective across a wide variety of classes of MOPs [184], the execution time and resource requirements (efficiency) of this MOEA require improvement. Since the efficiency of the MOMGA is not sufficient to solve large scale real-world MOP applications [184], a new explicit BB-based MOEA is created, the Multiobjective Messy Genetic Algorithm II (MOMGA-II).

Considering the objectives of this research, the choice was made to develop the MOMGA-II based upon the concepts and findings from research into the MOMGA. The MOMGA illustrated effective performance when applied to a variety of classes of pedagogical MOPs but is somewhat lacking in its overall efficiency. In order to achieve the level of effectiveness of the MOMGA and improved efficiency in the MOMGA-II for application to real-world MOPs, a thorough literature search revealed a number of possibilities for obtaining efficiency gains in the MOMGA-II. The SO fmGA [77] is chosen as a template to use for realizing those efficiency gains. The best concepts of the fmGA and the MOMGA are incorporated and extended to yield the MOMGA-II. The motivation is to attempt to solve MOPs of classes similar to real-world, Air Force MOP applications efficiently and effectively. Using an efficient and effective explicit BB-based MOEA is crucial to meeting the objectives of this research effort to include solving real-world, Air Force MOP applications, extending the analysis of explicit BB-based MOEAs as applied to MOPs, and gaining insight into difficult MOPs.

This chapter is organized as follows: a discussion of the MOMGA is presented in Section 6.2 to aid the reader's understanding of the differences between the MOMGA and the MOMGA-II. A detailed discussion of the MOMGA-II and the design of experiments follows. The results and analysis of the application of the MOMGA-II to unconstrained and constrained MOPs as well as real-world MOPs appears in Section 6.5. The MOEA BB testing is the final topic addressed in this chapter and is presented in Section 6.6.

6.2 Background of the MOMGA

In order to understand the MOMGA-II, its operators, and its implementation, one must understand the predecessor of this algorithm, the MOMGA. The MOMGA is described in Chapter III, Section 3.1.2.1. The MOMGA is an extension of the single objec-

tive mGA, that explicitly manipulates BBs of a user specified size in constructing "good" solutions. The MOMGA explicitly creates a population of BBs, containing every possible BB of the user specified size. Increasing the BB sizes and string lengths in the MOMGA results in an increase in the initial population size. The main disadvantage of the MOMGA is the exponentially increasing population sizes required for use as the BB size increases. This detriment previously prevented the MOMGA from being applied to large, real-world MOPs. In order to attempt to solve these MOPs, as well as other real-world MOPs, improvements over the original MOMGA are a necessity. The fmGA was designed to combat the large population sizes encountered in the mGA while producing "good" results when compared with the mGA and other single objective EAs and hence these concepts are of use in the MOMGA-II [77].

Designing and implementing an efficient and effective explicit BB-based MOEA involves the integration of various concepts from various algorithms as well as new ideas. To improve on the efficiency of the MOMGA and the fmGA, the code representing the concepts and ideas taken from these approaches was rewritten. In cases where efficient code existed, the existing code was used. A detailed description of the MOMGA is presented to explain the background operators used to create the MOMGA-II.

The MOMGA implements a deterministic process to generate an enumeration of all possible BBs, of a user specified size, for the initial population. This process is referred to as Partially Enumerative Initialization (PEI). Thus, the MOMGA explicitly uses these building blocks in combination to attempt to solve for the optimal solutions in multiobjective problems. While finding the optimal solution is never guaranteed by an MOEA, the MOMGA statistically finds the optimal solutions or solutions close to the optimal solutions in phenotype space to the functions presented in our standard MOP test suite [184, 189, 191, 192]. The pseudocode for the MOMGA is presented in Figure 6.1.

The original messy GA consists of three distinct phases: *Initialization Phase, Primordial Phase, Juxtapositional Phase*. The MOMGA uses these concepts and extends them where necessary to handle $k > 1$ objective functions. In the initialization phase, the MOMGA produces all building blocks of a user specified size. The population size grows exponentially with an increase in the BB size (also referred to as the order of the

```
For n = 1 to o
    Perform Partially Enumerative Initialization
    Evaluate Each Population Member's Fitness (w.r.t. k Templates)
// Primordial Phase
    For i = 1 to Maximum Number of Primordial Generations
        Perform Tournament Thresholding Selection
        If (Appropriate Number of Generations Accomplished)
            Then Reduce Population Size
        Endif
    End Loop
//Juxtapositional Phase
    For i = 1 to Maximum Number of Juxtapositional Generations
        Cut-and-Splice
        Evaluate Each Population Member's Fitness (w.r.t. k Templates)
        Perform Tournament Thresholding Selection and Fitness Sharing
        P_known(t) = P_current(t) ∪ P_known(t − 1)
    End Loop
Update k Competitive Templates (Using Best Value Known in Each Objective)
End Loop
```

Figure 6.1 MOMGA Pseudocode

BB), o, as shown in Equation (6.1). This yields a population size determined by Equation (6.1) [78, 184].

$$N = C^k \binom{l}{k} \tag{6.1}$$

In Equation (6.1) N represents the population size, C is the cardinality of the alphabet that is used in the GA (binary $C = 2$), l is the length in bits of the chromosome and o is the BB size. The MOMGA uses Equation (6.1) to determine the necessary population size in order to create every possible BB of size o. At the end of the initialization phase, the population consists of all of the possible strings of length o, i.e., all of the BBs. Since the population members are underspecified, a mechanism must exist for determining the fitness of the partial strings or BBs. To evaluate the fitness of a BB, a competitive template is used to fill in the unspecified or missing bits, prior to evaluation. This ensures that each evaluation is of a fully specified string, through the BB alone or in conjunction with the competitive template if necessary. In extending the mGA, the MOMGA incorporates k competitive templates, each initially generated randomly, and each later optimized for one of the respective k objectives.

The primordial phase performs tournament selection on the population and reduces the population size if necessary. The population size is adjusted based on the percentage of "high" fitness BBs that exist. In some cases, the "lower" fitness BBs may be removed from the population to increase this percentage. In the juxtapositional phase, BBs are combined through the use of a cut and splice recombination operator. Cut and splice is a recombination (crossover) operator used with variable string length chromosomes. The cut and splice operator is used with tournament thresholding selection to generate the next population. As the MOEA progresses through the generations, the number of specified bits within the chromosomes increases. With this increase in the number of specified bits comes a decrease in the dependence on the competitive template. The process continues for a user specified number of generations to yield strings with high fitness values for each fitness function. The combination of the three phases produces one era [77, 78]. The MOMGA continues for the user specified number of eras or BBs. Observe that an *epoch* refers to the execution of the algorithm through all of the eras.

The competitive templates are initially created randomly. Following each era, the best found individual in the population, for each objective function, becomes the new competitive template for that objective function. The era is incremented and the next BB size is executed using the same process as described above. A more detailed discussion of the MOMGA architecture is presented in [184, 190].

One objective of the MOMGA was to illustrate the fact that explicit BB concepts apply to the multiobjective problem domain. The MOMGA was designed to illustrate the usefulness of BBs in MOEAs and not necessarily to be a robust algorithm. The MOMGA lacks generalizations necessary to easily transition between attempting to solve MOPs of different sizes and decision variable resolutions. In most of the MOMGA code, the number of fitness functions and decision variables are hard coded into the algorithm. Hard coding numerous parameters makes it difficult to apply the MOMGA to MOPs of a different number of decision variables or fitness functions than those coded into the MOEA. Additionally, the MOMGA has only been applied to unconstrained MOPs and hence does not contain constraint handling code or feasibility measures. While the data structure of the MOMGA mirrored that of the mGA, these algorithms were not designed and coded

with parallelization in mind. Although not an objective of the research into the MOMGA, parallelization is an objective of this research effort. These factors along with coding errors present in the mGA/MOMGA motivated a new effort at coding the appropriate operators present in the MOMGA for use in the MOMGA-II. Existing MOMGA code was utilized when possible.

6.3 MOMGA-II - Development

The MOMGA has been shown to yield effective performance when applied to a variety of MOPs and compared to other well-engineered MOEAs [184, 190]. A similar level of effectiveness is desired in the MOMGA-II with increased efficiency. The results of the MOMGA-II are compared to the results of the MOMGA in order to validate the research objectives of this effort. Efficiency improvements over the MOMGA are necessary to decrease the execution time and system resources required. A thorough description of the MOMGA-II and its differences from the explicit BB-based MOMGA is presented in this section.

The main bottleneck in the mGA and the MOMGA is the initialization phase. This phase requires the enumeration of every possible BB, of user specified size, as denoted in Equation (6.1). In the original testing of the MOMGA, only two objective function MOPs were used with bit strings of length 24 and BB sizes one through three. In this configuration, the population sizes required to attempt and solve these MOPs were somewhat manageable but larger than used in most MOEAs, requiring a maximum population size of 16,192 individuals. Using longer string lengths or larger BB sizes leads to enormous, unmanageable population sizes that make other EAs and MOEAs more attractive. As many real-world MOPs require the use of strings longer than 24 bits, the MOMGA is not a feasible MOEA to use when attempting to solve such MOPs. The fmGA was designed to remedy the population size issue encountered with the mGA [77]. Integrating some of the concepts of the fmGA and MOMGA into the MOMGA-II creates an effective and efficient explicit BB-based MOEA.

A probabilistic approach is used in initializing the population of the fmGA. The approach is referred to as Probabilistically Complete Initialization (PCI) [77]. PCI initializes

the population by creating a controlled number of BBs based on the user specified BB size and string length. The fmGA's initial population size is smaller than the mGA (and MOMGA by extension) and grows at a smaller rate as a total enumeration of all BBs of size o is not necessary. These BBs are then "filtered," through a Building Block Filtering (BBF) phase, to probabilistically ensure that all of the desired good BBs from the initial population are retained in the population. The BBF approach effectively reduces the computational bottlenecks encountered with PEI through reducing the initial population size required to obtain "good" statistical results. The fmGA concludes by executing a number of juxtapositional phase generations in which the BBs are recombined to create strings of potentially better fitness. The pseudocode for the MOMGA-II is presented in Figure 6.2.

```
For n = 1 to o
    Perform Probabilistically Complete Initialization
    Evaluate Each Population Member's Fitness (w.r.t. k Templates)
// Building Block Filtering Phase
    For i = 1 to Maximum Number of Building Block Filtering Generations
        If (Building Block Filtering Required Based Off of Input Schedule)
            Then Perform Building Block Filtering
        Else
            Perform Tournament Thresholding Selection
        Endif
    End Loop
//Juxtapositional Phase
    For i = 1 to Maximum Number of Juxtapositional Generations
        Cut-and-Splice
        Evaluate Each Population Member's Fitness (w.r.t. k Templates)
        Perform Tournament Thresholding Selection and Fitness Sharing
        P_known(t) = P_current(t) ∪ P_known(t − 1)
    End Loop
Update k Competitive Templates (Using Best Value Known in Each Objective)
End Loop
```

Figure 6.2 MOMGA-II Pseudocode

The MOMGA-II mirrors the fmGA and consists of the following phases: *Initialization, Building Block Filtering, and Juxtapositional.* The MOMGA-II differs from the MOMGA in the Initialization and Primordial phase, which is referred to as the Building Block Filtering phase. The initialization phase of the MOMGA-II uses PCI instead of the PEI implementation used in the MOMGA and randomly creates the initial population.

The size of the population suggested to obtain "good" results for the fmGA is shown in Equation (6.2) [77].

$$n = \frac{\binom{l}{l'}}{\binom{l-k}{l'-k}} * 2c(\alpha) * \beta^2 * (m-1) * 2^k \ , \tag{6.2}$$

where l is the total number of genes, l' is the string length, o is the BB size, $c(\alpha)$ is used to determine the acceptable amount of error, β^2 is the signal to noise ratio, and m is the number of subfunctions. A detailed description of this equation is presented in [77]. Equation (6.2) provides a means of determining the initial population size to use in the fmGA in order to statistically generate the good BBs for the problem being solved. However the population sizing equation (Equation (6.2)) makes the assumption that the BBs are independent, and does not apply to MOPs unless the objective functions are all independent. In this chapter, a much smaller maximum population size is used in the MOMGA-II as compared to the MOMGA (250 population members for the MOMGA-II as compared to 16,192 for identical MOPs) to illustrate that the MOMGA-II obtains a similar level of effectiveness but with greater efficiency than the MOMGA. A smaller population size requires less system resources and hence is more efficient. Additionally, the evaluation of fewer population members leads to a reduction in execution time. These statements are validated through the MOMGA-II results presented in Section 6.5 of this chapter.

In the PCI initialization of the population in the MOMGA-II, all possible BBs of the user specified size are not created but instead the population is created randomly, with strings of length l. Fully specified population members are generated in which the allelic values are randomly selected as well as the positions (loci) to place these values. The size of the population is specified by the user. Following the generation of the initial population, the population members are all evaluated with respect to each of the objective functions in the same manner as the MOMGA. A random choice of one of the k competitive templates is conducted and overlayed by the bits stored in the population member. This is completed in order to generate a fully specified individual and conduct fitness evaluations on that individual. The competitive templates are handled in the same manner as used in the MOMGA, i.e., they are initially randomly created. After each BB size is executed, the

MOMGA-II updates each of the templates with the best population member found with respect to each individual fitness function.

The BBF phase reduces the lengths of the strings generated in the initialization phase through a systematic process. A user generated filtering schedule specifies the generations to conduct a random deletion of bits and the number of bits to delete from each chromosome. The schedule also specifies the number of juxtapositional generations to execute and can be automated by the MOMGA-II. The MOMGA-II uses the filtering schedule recommended in [77], in which the strings are halved in length with each filtering operation until they are reduced to the specified BB size. This is but one method to use for the schedule and has been shown to perform well [77]. Other scheduling methods exist but an optimal method has not been identified in the literature [108, 132]. The MOMGA-II does not use a mutation operator directly but the BBF operator can be classified as a mutation operator. The BBF operator does not randomly choose a string to mutate based on the probability of mutation, as does a traditional mutation operator. Instead, each string is filtered (mutated) and hence the probability of mutation can be viewed as 100%. Since the BBF operation is sufficiently different from the standard concept of mutation, the probability of mutation of the MOMGA-II is stated as 0%.

The filtering process consists of a random deletion of f bits from each of the population members. Therefore, after the first filtering operation, the strings are reduced in length from l bits to $l - f_i$ bits where f_i specifies the number of bits to remove in filtering step i. Each subsequent filtering operation randomly deletes additional bits. This filtering typically does not delete the same number of bits with each operation.

The filtering operation is alternated with tournament selection. The purpose of using a selection mechanism is to yield a competition between the building blocks. This competition is used to yield a new population containing the best population members found and hence by extension the best BBs. At the conclusion of the BBF phase, the entire population consists of individuals of the specified BB length. The population essentially consists of the best BBs found. The goal of the entire BBF process is to yield a similar result to the primordial phase of the MOMGA, a population consisting of a sufficient number of "good" BBs to be recombined in the juxtaposition phase of the algorithm.

The juxtapositional phase proceeds in the same manner as it did in the MOMGA. During this phase, the building blocks found through the Initialization phase and BBF phase are recombined through the use of a cut and splice operator alternated with tournament selection with thresholding. This process results in a gradual increase in the length of the population members towards becoming fully specified or having an allelic value stored in each locus position. The recombination operator is alternated with tournament selection in order to maintain a population of the best individuals found. Competitive templates are used to evaluate the fitness values of underspecified population members. In the event of overspecification, where a loci location has multiple allele values, a scan conducted of the population member upon which the first value encountered to specify the gene location becomes the allele value for that gene. As the juxtapositional phase continues, the population members gradually increase in length and are less dependent on the competitive templates to fill in unspecified alleles.

Following the juxtapositional phase, the BB size is incremented and the population member with the best fitness value, for each individual fitness function, replaces the respective competitive template. The MOMGA-II executes all three phases for each BB size and presents PF_{known} as the solution set generated. The MOMGA-II can be applied to MOPs with theoretically any number of decision variables and fitness functions, constrained only by the system resources of the platform used. The application of the MOMGA-II to real-world and constrained MOPs is discussed in more detail later in this chapter. The integration of problem domain information is necessary to effectively generate solutions to these MOPs. Additionally, constraint handling approaches are integral to finding good solutions to the constrained MOPs tested.

To evolve the MOMGA-II, the original MOMGA code was used as a guideline. Whenever possible, existing, logically correct, code from the MOMGA was used in the MOMGA-II coding; otherwise, new code was written. Considering that the original fmGA code was unavailable, large portions of the MOMGA-II had to be written from scratch. The MOMGA-II is written in ANSI C and therefore can be executed on many computing platforms. Testing of the MOMGA-II was conducted on systems using the Linux operating system.

The combination of existing code and new code made non-trivial modifications to the operation of the MOMGA in the initialization and primordial phases. A sound software engineering approach was used. This approach involved taking the time to develop a design process for coding and implementing the MOMGA-II. Additionally, debugging code was integrated into the MOMGA-II to aid in locating and correcting coding bugs. Since Objective 5 (see Chapter I) of this research is to analyze parallel concepts as applied to MOEAs, parallelism was kept in mind during the coding process. Much of the existing MOMGA code could not be parallelized without considerable modifications to the data structures. In the coding of the MOMGA-II, data structures were used that support parallelization. This involved the removal of a linked list data structure and replacement with dynamically allocated, multidimensional array data structures. Additionally, the MOMGA was coded explicitly for application to MOPs of low dimensionality in the genotype and phenotype spaces, use of a maximum of three competitive templates, and application to unconstrained MOPs only. The MOMGA-II coding process addressed these issues and generalized most parameters and variables to allow for increased functionality. This included the ability to easily apply the MOMGA-II to MOPs of any number of decision variables or fitness functions without having to rewrite a substantial portion of the code, the ability to specify any number of competitive templates, the addition of multiple constraint handling approaches and many other generalizations that allow for multiple parameter settings constrained only by the hardware limitations of the computing platform chosen.

Initially code optimization was not addressed but logical correctness and a good design to provide increased functionality over the MOMGA and fmGA were paramount. Following the implementation and testing of a working code and the desire for parallelization, code optimization issues were addressed. This is an important fact since a fair comparison of the MOMGA and MOMGA-II is desired to discuss the differences in efficiency and effectiveness of the two MOEAs. The code optimization issues were addressed following the comparison of the MOMGA to the MOMGA-II.

6.4 Design of Experiments

To conduct a thorough evaluation of the performance of any MOEA, a good design of experiments must be conducted prior to testing the algorithm. Two of the objectives of this chapter are to develop an efficient explicit BB-based MOEA for solving real-world applications and real-world Air Force MOPs and to demonstrate that explicit BB-based MOEAs provide insight into difficult MOPs that otherwise could not be found. These objectives must be taken into account when conducting the design of experiments.

To make a fair comparison of the results of the MOMGA-II to the MOMGA and other MOEAs, over a class of MOPs, these algorithms must all be tested over the same MOPs. In comparing the MOMGA-II to the MOMGA, the objective is to illustrate an improved efficiency at a similar or better statistical level of effectiveness. Efficiency is used in the context of system resources and execution time. In comparing the MOMGA-II to other MOEAs, the objective is to illustrate statistically similar, or better, effectiveness of the MOMGA-II.

The guidelines suggested by Jackson et al. [97] and Barr, Golden, Kelly, Resende, and Stewart [16] are used to conduct a good design of experiments. Specifically, the 5 experimentation steps presented by Barr et al. [16] are followed throughout this research effort to ensure that the experiments and results are meaningful. These steps consists of: 1) Define the goals of the experiment (Chapter I); 2) Choose measures of performance and factors to explore (Chapter III); 3) Design and execute the experiment (Chapter VI); 4) Analyze the data and draw conclusions (Chapter VI) and 5) Report the experiment's results (Chapter VI) [16]. The results are analyzed and statistical tests are used to provide useful insight into MOMGA-II performance.

The selection of test MOPs for comparing the MOMGA-II to the MOMGA and other MOEAs, including the MOGA, NSGA, NPGA, SPEA, PAES, and NSGA-II, is a difficult task. A more in-depth discussion of the possible MOPs to test these algorithms against is presented in Chapter IV. In the comparison of the MOMGA-II to the MOMGA, the same MOPs are chosen as were used for testing the MOMGA in Van Veldhuizen's original work [184]. These five MOPs were meticulously selected from numerous MOPs existing in

the literature and represent different classes of MOPs with varying characteristics. While it is impossible to evaluate the performance of any algorithm over every class of problem, the selected MOPs have characteristics that are representative of many real-world application problems. The genotype and phenotype characteristics vary as well as the structure of P_{true} and PF_{true} , which includes connected and disconnected, concave, convex, and a scalable number of decision variables and Pareto curves. These selected MOPs represent both minimization and maximization functions. Additionally, other researchers have recently used a subset of these MOPs to evaluate and compare the performance of their MOEAs to others [29, 43, 46, 116]. Two fitness function MOPs are used for ease of presentation and to provide critical insight to MOEA performance. The five unconstrained MOPs selected from Chapter IV are MOP 1, MOP 2, MOP 3, MOP 4, and MOP 6.

In comparison testing of the MOMGA and MOMGA-II, all of the parameter settings were kept constant across the two algorithms in order to illustrate the improvement in execution time of the MOMGA-II. These parameter values allow for a fair comparison of the MOMGA-II to other MOEAs, which were executed with comparable parameter settings. The parameter settings are discussed in Section 6.5 of this chapter. In order to make statistically accurate statements about the results of the MOEAs tested, various metrics must be used. The metrics used are drawn from the discussion presented in Chapter III, Section 3.2. While numerous metrics exist, a subset of these are selected in order to evaluate the results of the MOEAs tested. In general, one would like to use metrics that are computationally feasible to compute, allow for a clear interpretation of the metric results (are not misleading) and metrics that complement each other in providing the researcher with a good overall idea of the performance of the MOEA on the tested MOP. Error ratio, spacing, generational distance, maximum error, hyperarea ratio, ONVGR and ONVG as well as a visual comparison are selected for use in comparing the MOMGA-II to the MOMGA results.

The five MOPs used in the comparison testing of the MOMGA-II and MOMGA represent unconstrained MOPs with varying characteristics. Additional MOPs are selected to conduct a more thorough analysis of the performance of the MOMGA-II. These MOPs include constrained, discrete, and real-world MOPs. A constrained MOP, MOP-C1 is used,

as well as MOP-CT, a constrained test function generator that allows a researcher to modify parameters and change the characteristics (and difficulty) of the MOP. Four variants of MOP-CT are tested in order to illustrate the performance of the MOMGA-II across a variety of constrained problem characteristics. A real-world MOP, the Advanced Logistics Problem, and the Modified Multiobjective Knapsack Problem are also used to illustrate the performance of the MOMGA-II as applied to discrete MOPs formulated with a large number of decision variables. The three metrics used to compare and contrast results for these MOPs are generational distance, spacing and ONVG as discussed in Chapter III. These three metrics are selected as they complement each other and provide data that can clearly indicate MOEA performance. The use of the three selected metrics in conjunction with a visual analysis of PF_{known} (compared to PF_{true} if possible) provides a clear view of the performance of the MOMGA-II.

The application of the MOMGA-II to the unconstrained test problems is straight forward; however, constrained problem solving poses more of a challenge. In the solving of constrained MOPs, one has multiple options to consider for constraint handling. The simplest option is to ignore the constraints until MOEA completion and then remove all of the infeasible population members. This option has the advantage that the constraints do not need to be continually checked as the MOEA executes. However, the disadvantage of this approach is that there is no guarantee that the MOEA is going to find any feasible solutions. In some constrained MOPs, the majority of the search space is infeasible (as occurs in the ALP MOP) and there is a low probability that feasible solutions are generated using this approach. Additionally, ignoring the constraints until MOEA completion requires the researcher to store every population member generated throughout MOEA execution. Storing the entire population is required since one does not check the constraints until MOEA termination and hence all population members must be stored in order to have a greater probability of generating a feasible solution. This approach can perform well if the MOEA finds BBs that lead to feasible solutions throughout the search process.

Another option for constraint handling is to penalize the fitness values of infeasible population members. Penalty approaches are sometimes difficult to implement as a researcher must determine the amount of penalty to apply to infeasible solutions. Repair

mechanisms are yet another option for attempting to solve constrained MOPs. A repair mechanism mutates infeasible individuals to generate feasible solutions. This approach ensures that the population always contains feasible individuals. A repair approach is used for both of the discrete constrained integer MOPs, the ALP and MMOKP MOPs.

Other researchers have applied their MOEAs to the identical MOPs selected in this effort for testing. In many cases, these MOEAs have been executed with comparable parameter settings and in some cases their statistical results have been published. In cases where statistical results are available, a statistical comparison is made between these MOEAs and the MOMGA-II. In cases where the same MOPs are used but the statistical results are not available, a visual comparison is made between the MOMGA-II results and these MOEAs.

6.5 MOMGA-II Results and Statistical Analysis

Relative comparison performance data for a set of MOEAs and MOPs can be presented in a number of formats. They include but are not limited to individual line and bar graphs of metric values illustrating average, median, standard deviation, maximum and minimum values. Additionally, tables listing explicit values of the metrics or their averages, compact scatter diagrams, or Pareto fronts and Pareto solutions are effective for illustrating MOEA performance. In any case, the use of explicit hypothesis testing should be conducted when possible to enforce the statistical analysis made.

In the analysis of MOEA results, parametric comparisons can be made in situations where normal distributions of the data exist and other criteria is met. Since the results obtained in this research effort are not normally distributed and do not meet the criteria of parametric testing, nonparametric statistical techniques are discussed and used where applicable. As a wide variety of MOPs are tested in this research effort, situations exist where statistical comparisons of the MOMGA-II to other MOEAs are warranted. In other situations, statistical data for the comparison MOEAs is not available but a visualization of the Pareto front is. In such situations, visual comparisons are made as appropriate. In cases where real-world, Air Force specific MOPs are tested, that other MOEAs have not

been applied to, comparison data is not available and hence only the results of the metrics are presented.

6.5.1 Standard MOP Test Suite Experimental Results. In applying the MOMGA-II to the unconstrained MOPs, each experiment was executed ten times. Ten data runs are conducted for the MOMGA-II as the comparison MOEAs were also executed ten times and this allows for a statistical comparison to be made between the MOEAs. Additionally, a visual comparisons is made between the MOMGA and MOMGA-II for these MOEAs. The following sections present the results obtained in applying the MOMGA-II to the five unconstrained MOPs.

The primary objective of testing the unconstrained MOPs is to illustrate that the MOMGA-II obtains statistically similar results (or better) than the MOMGA in terms of effectiveness over the selected metrics. Additionally, it is shown that the MOMGA-II obtains similar or better effectiveness results as compared to the comparison MOEAs, and greatly improved timing results over the MOMGA. As Van Veldhuizen previously compared the results of the MOGA, MOMGA, NPGA, and NSGA to each other, the analysis presented in this section is concerned with comparing the MOMGA-II to each of these MOEAs and hence does not cover the other comparisons that can be made between the remaining MOEAs. The reader is referred to [184] for a more detailed discussion of these comparisons.

In each of the five MOPs tested, the PF_{true} was determined through a total fine-resolution grid enumeration of the search space. A program was constructed and run on powerful, high-performance computer systems to find P_{true} and PF_{true} for each of the five MOPs tested [184]. This program conducted a total enumeration of the MOPs' search space in order to find the true solution to the MOPs at a 24 bit decision variable computational resolution.

The results presented reflect the output of each MOEA using the default parameter values. The statistical comparisons completed are based on the maximum, minimum, median, average, and standard deviation calculations for each metric. Additionally, advanced nonparametric statistics, including the Kruskal-Wallis Hypothesis Test and the pairwise

Wilcoxon Rank Sum Test (Mann Whitney Test) are used to illustrate a statistical difference or indifference between the various MOEA's sampled results.

The MOMGA-II and MOMGA runs consist of executing each MOEA for 3 eras of 20 generations each. A string length of 24 bits, BB sizes of 1-3, a cut probability of 2% and a splice probability of 100% are used in the MOMGA and MOMGA-II. The MOMGA-II differs from the MOMGA by the BBF schedule that dictates the number of generations that BBF must take place. The MOMGA uses the mGA calculation to determine the number of primordial generations to execute, while the MOMGA-II always executes 9 generations of the BBF and tournament selection operations. Hence the MOMGA-II only executed 11 generations of the juxtapositional phase. The MOMGA on the other hand executes between 17 and 19 generations of the juxtapositional phase. The number of juxtapositional generations executed per MOEA is important to note since the generation of the solution strings takes place through the cut and splice operation of the juxtapositional phase. The larger number of juxtapositional generations executed by the MOMGA favors the MOMGA over the MOMGA-II since it yields a more thorough mixing of the BBs. However, the MOMGA-II obtains statistically similar results to the MOMGA with much fewer juxtapositional generations. The results illustrate that the MOMGA-II finds the same "good" building blocks that the MOMGA finds and does this in less overall execution time.

The seven metrics used for comparison of the MOGA, MOMGA, MOMGA-II, NPGA, and NSGA as applied to MOPs 1-6 are error ratio, spacing, generational distance, maximum error ratio, hyperarea ratio, ONVGR and ONVG. Additional metrics are used beyond the generational distance, spacing and ONVG metrics suggested for use in Chapter III as results are available to compare the MOMGA-II and MOMGA with these metrics. It is desired that the error ratio, generational distance, and maximum error be as close to 0 as possible to illustrate that the MOEA finds all of the points contained in PF_{true} . Finding all of the points in PF_{true} would yield a value of 1 for the hyperarea ratio and ONVGR, hence the closer these metrics are to a value of 1 versus 0, the larger the number of PF_{known} points that also exist in PF_{true} . The spacing and ONVG metric are problem dependent and their measures vary with respect to the spacing between points and cardinality of PF_{true} . In

general, a small distance between points on the front is desired as well as the generation of a large number of points on the front. The metrics and the rationale for selecting each of them is discussed in detail in Chapter III.

The results for each of the MOEAs as applied to MOPs 1, 2, 3, 4, and 6 are presented in Figures 6.3 through 6.9, which illustrate a visual comparison of PF_{known} and PF_{true}. Figures 6.13 through 6.17 present the metric values calculated for each MOEA applied to each MOP. The first set of figures, Figures 6.3 through 6.9, represent the combination of the ten PF_{known} results found for the MOMGA-II and MOMGA. Figures 6.13 through 6.17 appear at the end of this section and present the metric results of the five MOEAs (on the x-axis) as applied to the respective MOP selected from the test suites. A dot (.) denotes the metric value (the y-axis) for each of the ten runs executed per MOEA. Error bars are overlayed on each MOEA's data points to illustrate the variance present in the results. These error bars represent one standard deviation about the mean ($\mu \pm \sigma$). The abbreviations M-I and M-II denote the MOMGA and MOMGA-II results, respectively.

Details of the MOGA, MOMGA, NPGA, and NSGA are necessary in order to conduct a fair comparison of the MOMGA-II to these MOEAs. The results for the four comparison MOEAs are taken from Van Veldhuizen [184]. He states that the error ratio metric may be misleading as the MOGA and NPGA use a different binary to real-value mapping than the NSGA and MOMGA (also the MOMGA-II) and are executed on different hardware architectures. Additionally, the calculation of P_{true} and PF_{true} was conducted on a comparable architecture to that used to execute the NSGA and MOMGA (as well as the MOMGA-II). This causes subtle differences in the results of the MOGA and NPGA as compared to the other MOEAs which may in turn manifest itself in an incorrect error ratio for these MOEAs. This may occur since the MOGA and NPGA are susceptible to different round-off or truncation errors as they are implemented using different programming languages on different machine architectures. Van Veldhuizen does not state it, but by extension, the same truncation problem may effect the generational distance and maximum error ratio metrics as they require the explicit computation of distances from vectors in PF_{known} to PF_{true}.

Statistically, the results of the MOMGA-II are compared to other MOEAs through the use of maximum, minimum, average, and median results in conjunction with the associated standard deviation and variances. The statistical results are discussed following the visual comparisons. Mann-Whitney and Kruskal-Wallis hypothesis testing are used together to make statistically significant statements about the results of the MOMGA-II. The combined use of these statistical comparison techniques is integral to the presentation of meaningful results and analyses of these results.

6.5.1.1 MOP 1 Results. Figure 6.3 illustrates the PF_{known} set generated by the MOMGA-II and MOMGA over ten data runs and compared to PF_{true}. The MOMGA-II generates a good distribution of points which covers the entire front. Visually it appears that the MOMGA-II and MOMGA find all of the solutions in PF_{true} and obtain identical results when compared to each other. The statistical results comparing the effectiveness of these algorithms is presented in Figure 6.13 and shows slight differences between the generated values and PF_{true}. Statistical data for comparing the NSGA-II and PAES is unavailable; therefore, a visual comparison between the MOMGA-II and these MOEAs is made.

Figure 6.4 illustrates the visual results of the NSGA-II and PAES as applied to MOP1 for one data run [46]. The NSGA-II and PAES are executed with similar parameter settings to the MOMGA-II which allows for a comparison to be conducted. Both the NSGA-II and PAES are executed with an equivalent population size of 100 for 250 generations. This favors these MOEAs as the MOMGA-II uses a population size of 250 and only 60 generations. The parameter settings indicate that the NSGA-II and SPEA evaluate more population members than the MOMGA-II over the course of execution and hence search more of the space. Since the statistical results are not available, an exact comparison of the MOEAs cannot be made. However, one can see that the MOMGA-II visually performs better than both of these MOEAs. Since the MOMGA-II finds a better spread of solutions across the front and more overall solutions, equivalent or better statistical performance is expected from the MOMGA-II as compared to the NSGA-II and PAES.

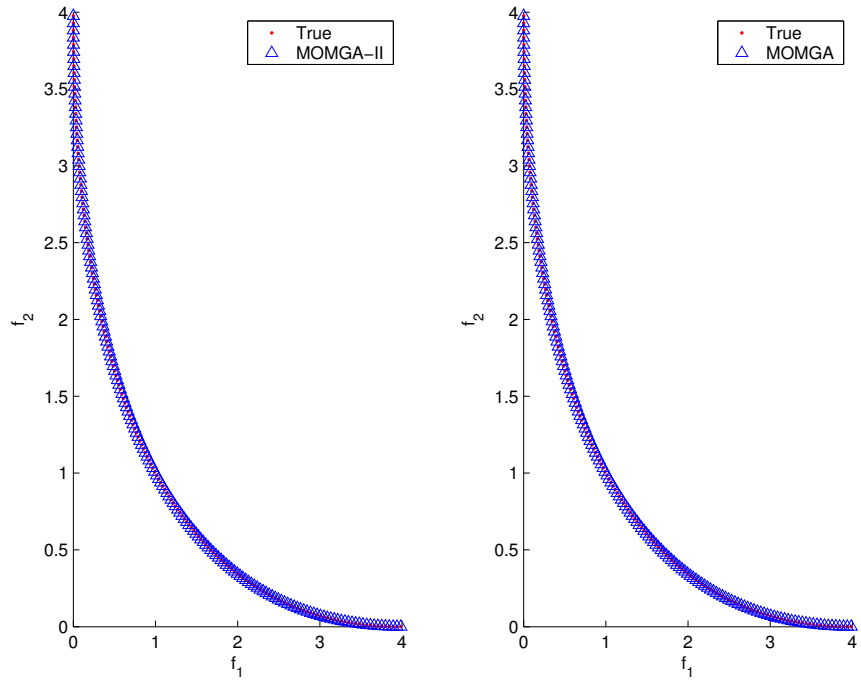

Figure 6.3 MOP1 PF_{known} Comparison

The metric values for each MOEA are presented in Figure 6.13. The results of the MOMGA-II as compared to the MOMGA are favorable. For each metric, with the exception of the error ratio, the two algorithms obtain results in which the mean and standard deviations overlap hence illustrating similar statistical performance in terms of effectiveness. The results of the MOMGA-II generally illustrate better effectiveness over the remaining MOEAs. The NPGA and NSGA typically only generate one or two solutions in PF_{known} each data run, and due to this, the spacing results for these MOEAs must be discounted. Overall, the MOMGA-II and MOMGA are more effective than the other MOEAs based on the selected metrics.

While this MOP formulation only uses one decision variable, some MOEAs have difficulty generating solutions close to PF_{true} . The bounds on the decision variable make MOP 1 difficult for some approaches as the cardinality of PF_{true} is small compared to the overall size of the search space. The MOMGA-II and MOMGA find the good BBs necessary to generate good solutions to this MOP and the MOMGA-II obtains results that compare to the effectiveness of the MOMGA.

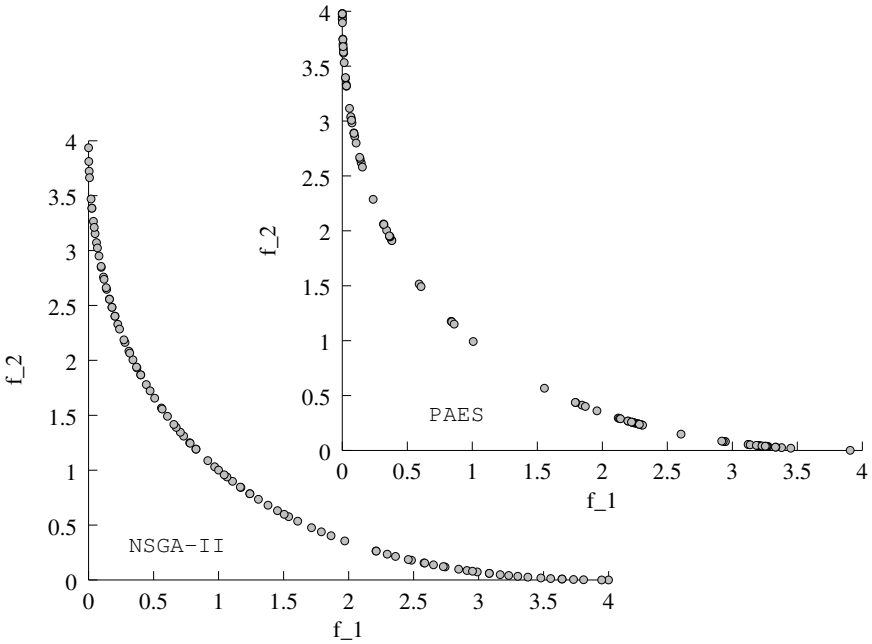

Figure 6.4 MOP1 NSGA-II and PAES Results [46]

6.5.1.2 MOP 2 Results. The visual comparison of the Pareto fronts generated by the MOMGA-II and MOMGA in Figure 6.5 illustrate minor differences in the lower right section of the front. The results for these two MOEAs are almost identical but the MOMGA finds a few more points than the MOMGA-II in the lower right section of the front. Both MOEAs find a good distribution of points across PF_{known} and these points appear to be very close if not identical to those in PF_{true} and cover the entire front.

Statistically the results comparing the effectiveness of these algorithms is presented in Figure 6.14. The statistical data illustrates overlapping error bars between metric values for the MOMGA-II and the MOMGA with the exception of the error ratio, in which the MOMGA-II obtains better results. These overlapping error bars illustrate similar performance in terms of the effectiveness of the MOMGA-II and the MOMGA.

The NPGA is the only other MOEA to find solutions contained in PF_{true} besides the MOMGA-II. The spacing results show similar performance between all of the MOEAs with the exception of the NSGA which does not obtain the same quality of solutions. This observation also holds for the generational distance metric. The MOGA and NPGA reflect a smaller variance in the maximum error and hyperarea ratio metrics while the remaining

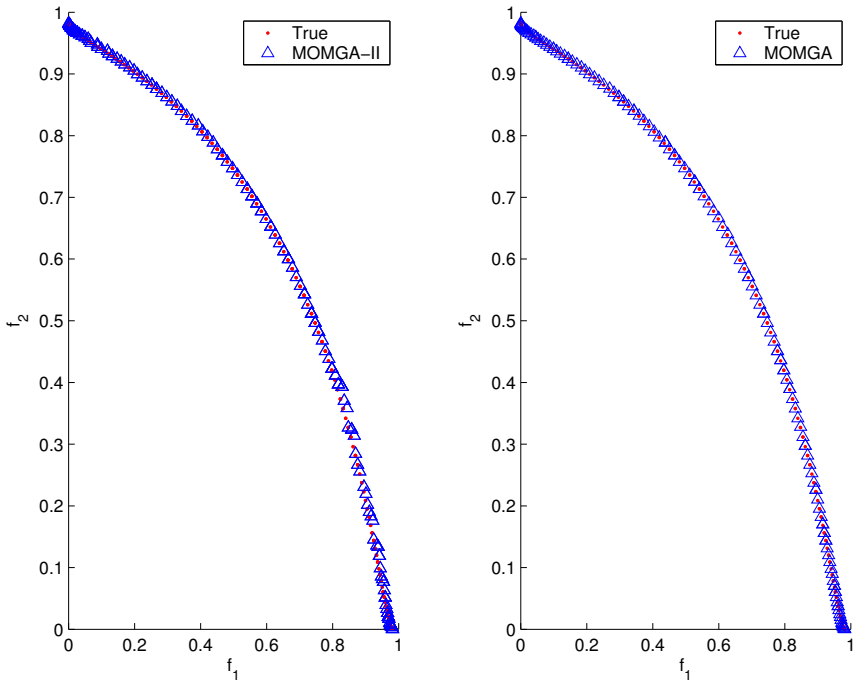

Figure 6.5 MOP2 PF_{known} Comparison

MOEAs all exhibit a somewhat large variance for these metrics. The ONVGR and ONVG performance of most of the MOEAs is similar with the exception of the NSGA which consistently generates fewer solutions. The results again illustrate that the MOMGA-II is as effective as the MOMGA and compares favorably to other MOEAs for MOP 2.

6.5.1.3 MOP 3 Results. MOP 3 is a maximization problem and its associated Pareto front is disconnected. Visually, Figure 6.6 shows the performance of the MOMGA-II and MOMGA as appearing identical. However, slight differences are found in the statistical analysis of the results of each MOEA. The MOMGA-II is effective at finding a good distribution of points across the front and in generating points that cover the front.

The statistical results are presented in Figure 6.15. The MOGA and NPGA are the only two MOEAs to find any solutions that are contained in PF_{true}. The spacing results of the MOMGA-II is typically not as good as that realized by the MOGA, MOMGA, and NPGA. The same observation holds for the generational distance and ONVG metrics in

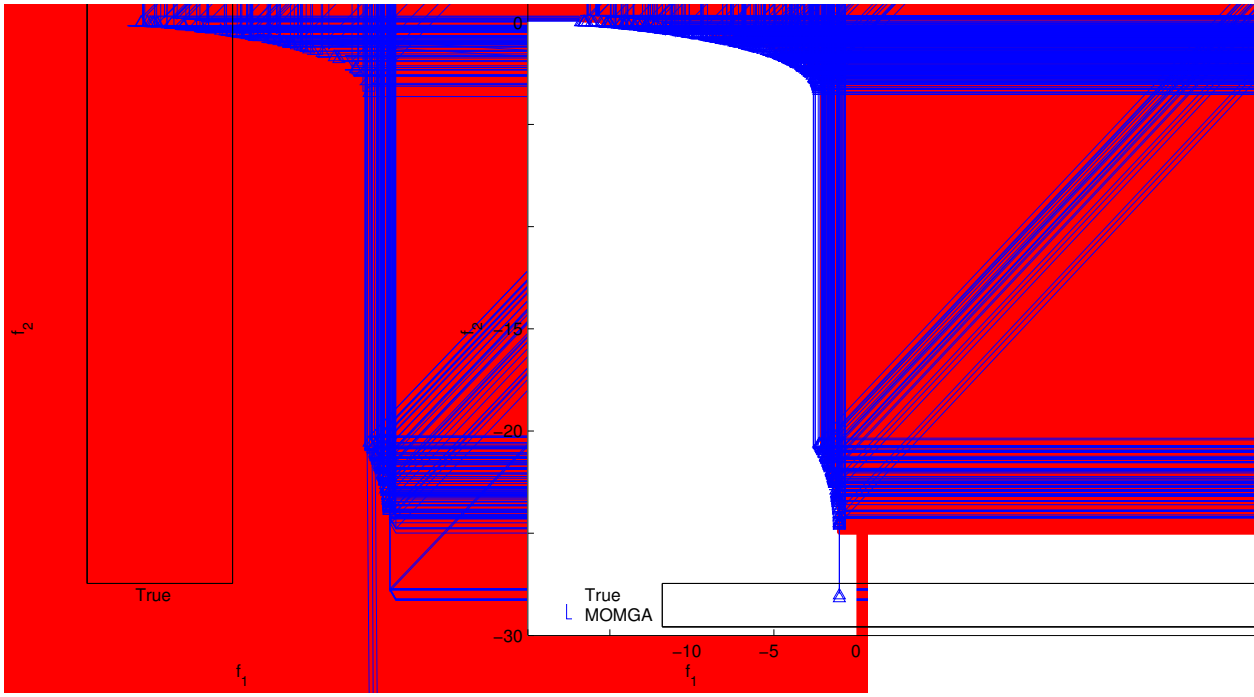

Figure 6.6 MOP3 PF_k

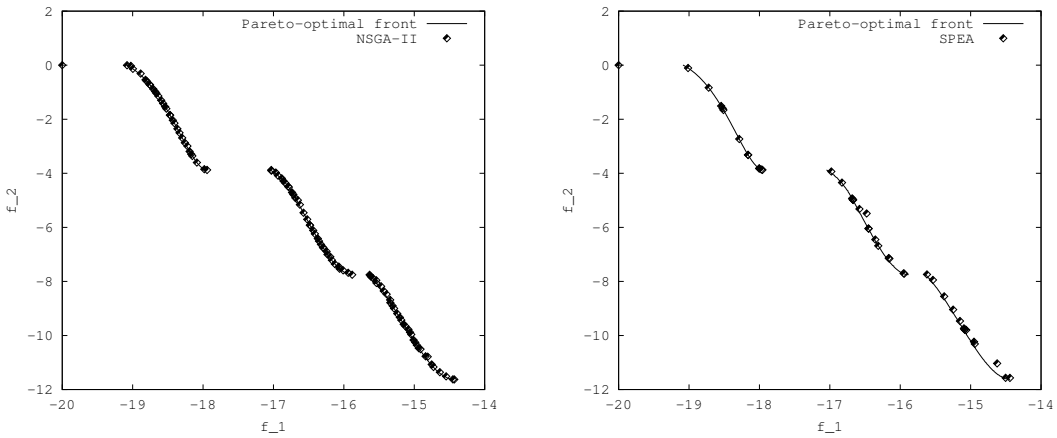

Figure 6.8 MOP4 NSGA-II and SPEA Results [46]

6.5.1.5 MOP 6 Results. Figure 6.9 illustrates identical performance of the MOMGA-II when compared visually to the results of the MOMGA. The MOMGA-II is effective at finding points within PF_{known} and points that are well distributed and cover the entire Pareto front.

The statistical results presented in Figure 6.17 support the visual results shown. The MOMGA-II and MOMGA find a large number of solutions that are elements of PF_{true} with the MOMGA-II finding an overall larger number of solutions in PF_{true}. The only other MOEA to find any solutions contained in PF_{true} is the NPGA which only finds a small number relative to the results of the MOMGA-II. The spacing metric indicates similar performance among the MOMGA-II, MOGA, and MOMGA, with the remaining MOEAs not performing as well. The same trend holds for the generational distance, maximum error and hyperarea ratio. The ONVGR and ONVG results of the MOMGA-II and MOMGA overlap and are much better than those generated by the other MOEAs. These results indicate similar effectiveness of the MOMGA-II and MOMGA and a tendency for better effectiveness of the MOMGA-II as compared to the other MOEAs tested.

The limited statistical results published for the NSGA-II, PAES, and SPEA illustrate that the MOMGA-II also performs "statistically better" than the NSGA-II, PAES, and SPEA on MOPs 2, 3, and 4 in terms of the generational distance metric [47]. The NSGA-II obtains larger mean values than the MOMGA-II does as presented in [47]. Note that the NSGA-II results are based on a variant of the spacing metric using absolute distance; i.e.,

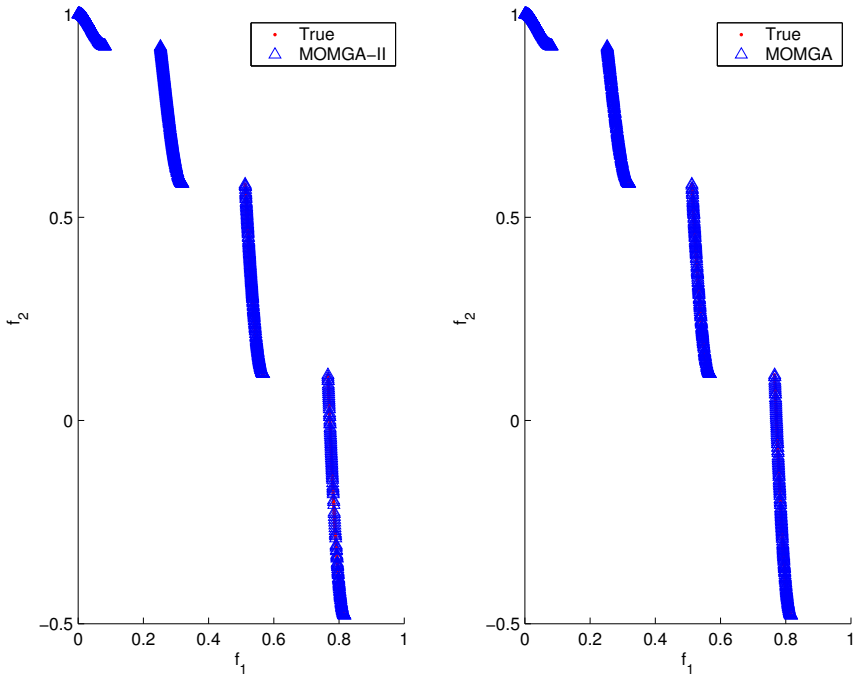

Figure 6.9 MOP6 PF_{known} Comparison

$p = 1$ not 2 thus weighting larger error values (less than one) higher.

$$G \triangleq \sum_{i=1}^{F} \frac{|(d_i - \overline{d})|}{|F|} \tag{6.3}$$

6.5.1.6 Standard MOP Test Suite Statistical Analysis. The visual and statistical results presented indicate that the MOMGA-II is effective at obtaining good results as applied to the selected MOPs. Additionally, the MOMGA-II yields a similar effectiveness on the tested MOPs as compared to the MOMGA and the other comparison MOEAs tested. Since the MOMGA-II was executed with a non-optimized BBF schedule, the results are impressive. The MOMGA-II has found the known Pareto front in less time and with fewer initial building blocks to choose from than the MOMGA while maintaining the same level of effectiveness. The results illustrate that the concepts of PCI and the BBF process can be extended to the MOP Domain.

The variances of the data results presented in Figures 6.13 through 6.17 for each of the MOEAs varies, indicating that a normal distribution of results is not present. There-

fore a nonparametric statistical analysis of the results is necessary. Of the seven metrics presented, three are of particular interest as discussed in Chapter III. The three metrics are Generational Distance, Overall Nondominated Vector Generation, and Spacing. These three metrics are selected for additional analysis through the calculation of nonparametric statistical values.

The other four metrics, for which data is presented, all have disadvantages that preclude them from further analysis. The error ratio metric requires knowledge of PF_{true} and yields a value other than 0 only if exact solutions that are on PF_{true} are found. Error ratio cannot distinguish between cases where solutions contained in PF_{known} are very close to PF_{true} versus cases where solutions in PF_{known} are very far from PF_{true} . Therefore the error ratio metric is not explored in more detail. The maximum Pareto front error measures the largest distance between vectors in PF_{known} and their closest corresponding vectors in PF_{true} . This metric can easily mislead a researcher. Case in point, if a single point in PF_{known} is quite far from PF_{true} , then the overall value for this metric may be poor. This single point of poor quality drastically effects the value of this metric and can mislead a researcher. Therefore this metric is not explored further.

The hyperarea metric presents misleading values when used to analyze a Pareto front that is not convex. Hyperarea measures the area contained under the curve. The Hyperarea ratio metric was proposed to attempt to correct the problem of applying hyperarea to non-convex Pareto fronts. This metric is a ratio of the area under PF_{known} over the area under PF_{true} but still poses problems with non-convex and disconnected Pareto fronts and hence is not explored further. The overall nondominated vector generation ratio metric is the ratio of the cardinality of PF_{known} over the cardinality of PF_{true} . This metric is sensitive to the size of PF_{true} and hence is not further analyzed. The remaining metrics, generational distance, overall nondominated vector generation and spacing are suitable for further analysis.

The generational distance, overall nondominated vector generation, and spacing metrics are selected for further analysis. Since the data is not be normally distributed, parametric mean comparisons are not possible; therefore, non-parametric statistical techniques must be employed. Kruskal-Wallis hypothesis testing is used with $\alpha = 0.1$ to determine

if a statistical difference exists between the results of the tested MOEAs. Kruskal-Wallis testing is useful in comparing the results of multiple MOEAs as no assumption must be made about the distribution of the data. However, the data samples must be independent, there must be at least five values from each data set to be compared and the probability distributions of the data must be continuous in order to use this test [198]. Since these criteria are satisfied, the Kruskal-Wallis test is used. The null hypothesis, H_0, states that the probability distributions of each of the MOEA results are identical when applied to a test MOP. The alternate hypothesis, H_a, states that at least two of the MOEA's probability distributions when applied to a test MOP are different. If the p-value obtained from the Kruskal-Wallis analysis of the data is less than or equal to α ($p \leq \alpha$), then the null hypothesis is rejected. This means that at an α level of 0.1, one is confident that at least two of the MOEA results are distributed differently.

In order to make a statistical statement as to which of the MOEA results are identically distributed, one can apply the Mann-Whitney test, sometimes referred to as the pairwise Wilcoxon rank-sum test. This test allows for pairwise comparisons to be conducted when the data is not normally distributed. The requirements of the Mann-Whitney test are that the samples are randomly selected (and are independent) and that the probability distributions from which the data is selected are continuous [198]. Since all of these requirements are met, the Mann-Whitney test can also be used here. In terms of the Mann-Whitney test, the null hypothesis, H_0, states that the distributions of the data from the two tested MOEAs as applied to the MOP are identical. The alternate hypothesis, H_a, states that the two data distributions are not identical.

In conducting the Mann-Whitney test, one must also select an α level. Using the same approach as Van Veldhuizen [184], the Bonferroni technique states that if x Wilcoxon Rank-Sum tests (Mann-Whitney) are performed with an overall α level of significance, then each individual test can be conducted at a α^* level of significance [131, 149]. Each individual test is defined as having a level of significance equal to $\alpha^* = \alpha/x$. The null hypothesis is rejected whenever the p-value obtained is less than or equal to α^* ($p - value \leq \alpha^*$). Using an overall significance level of $\alpha = 0.2$ and conducting four Mann-Whitney tests, this yields $\alpha^* = 0.2/4 = 0.05$. Even though there are ten possible pairings, one objective of this

research is to compare the effectiveness of the MOMGA-II to these other MOEAs, yielding four pairings. These pairings are: MOMGA-II and MOGA, MOMGA-II and MOMGA, MOMGA-II and NPGA, and MOMGA-II and NSGA.

The three metrics selected for further statistical investigation, generational distance, overall nondominated vector generation, and spacing, are presented in a different format, a bar graph, to highlight the differences among MOEA results. In Figures 6.10, 6.11, and 6.12, the bars represent the mean metric performance, μ, over 10 data runs with the standard deviation (1 σ) error bars shown.

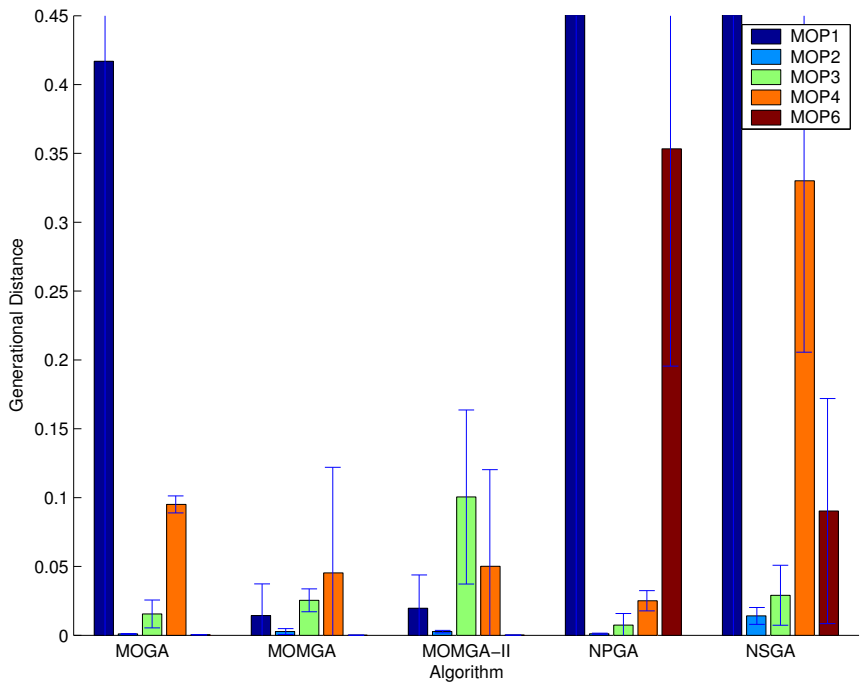

Figure 6.10 Overall Generational Distance Performance

Figure 6.10 presents the generational distance performance of the five MOEAs. The results of the MOGA, MOMGA, and the MOMGA-II as applied to MOP 6 are present but are all minute in comparison to the other MOP results. The results of MOP 6 for the MOGA, MOMGA, and the MOMGA-II are better illustrated in Figure 6.17. The values for MOP 1 for the NPGA (66) and NSGA (11) are not shown (are off of the graph) so as to improve the visualization of the difference between the MOMGA-II and the other MOEAs where smaller values are obtained. Primarily this illustrates that the MOMGA-II

and MOMGA mean and standard deviations overlap for all of the MOPs tested except MOP 3. This illustrates that the same level of effectiveness is achieved by the MOMGA-II and the MOMGA with respect to generational distance.

The results of each MOEA when analyzed with respect to the ONVG metric are presented in Figure 6.11. A similar trend is illustrated. The MOMGA-II and MOMGA mean and standard deviations overlap for all of the MOPs tested with the exception of MOP 3. This means that the MOMGA-II and MOMGA obtain the same level of effectiveness according to the ONVG metric.

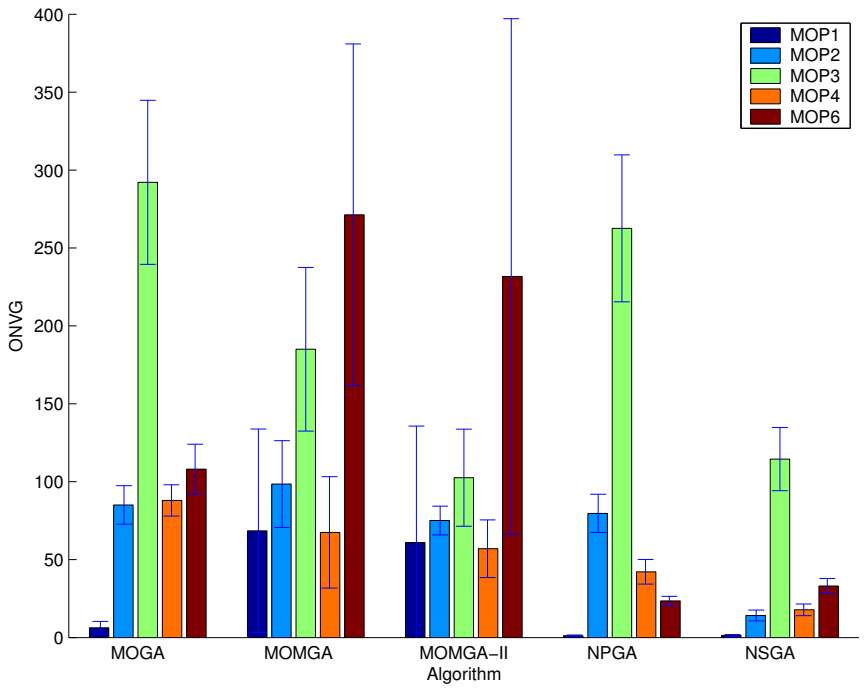

Figure 6.11 Overall ONVG Performance

Figure 6.12 presents the overall MOEA performance with regards to the spacing metric. The MOMGA-II and MOMGA mean and standard deviations also overlap for all of the MOPs tested with the exception of MOP 3. This illustrates that the MOMGA-II and MOMGA obtain the same level of effectiveness for these problems with respect to the spacing metric.

The results from conducting the Kruskal-Wallis Hypothesis testing for each of the three selected metrics as applied to the MOEA results for each of the MOPs are presented

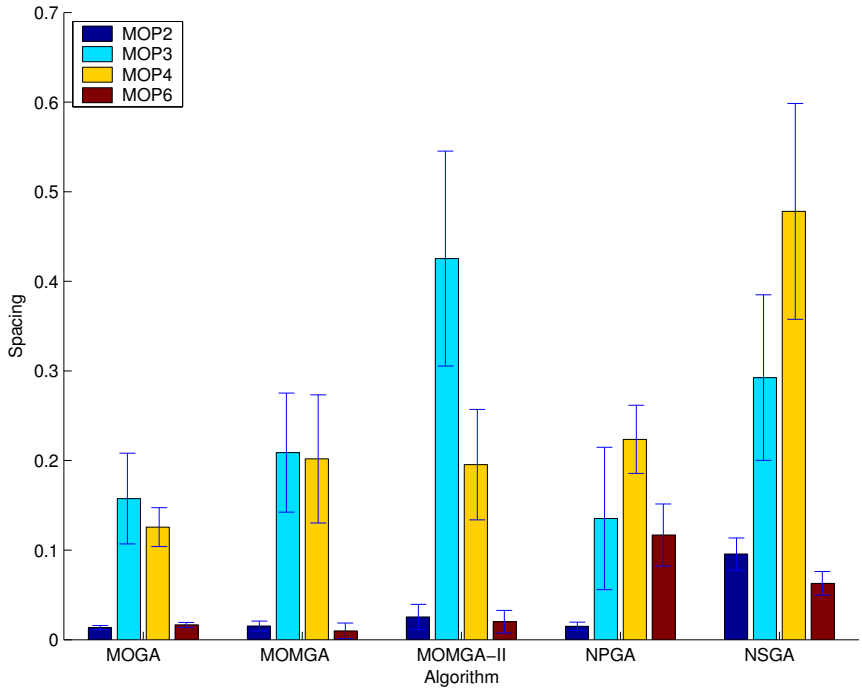

Figure 6.12 Overall Spacing Performance

in Table 6.1. An α level of 0.1 is selected and if $p \leq 0.1$, then the null hypothesis is rejected. The null hypothesis states that with the specified level of confidence, the MOEAs obtain statistically the same distributions of data results for the given metric and MOP.

Table 6.1 Nonparametric Kruskal-Wallis Results for Standard MOP Test Suite

MOP	**Kruskal-Wallis p-value for**		
	Gen Dist	**ONVG**	**Spacing**
MOP1	0.0002	0.0008	< 0.0001
MOP2	< 0.0001	< 0.0001	< 0.0001
MOP3	0.0001	< 0.0001	< 0.0001
MOP4	< 0.0001	< 0.0001	< 0.0001
MOP6	< 0.0001	< 0.0001	< 0.0001

Comparing the p-values in Table 6.1 to an α level of 0.1, illustrates that all of the results presented in the table support rejecting the null hypothesis. This means that in every MOP tested, at least two of the MOEAs obtain statistically different results. In order to determine which of the MOEAs obtain statistically the same or different results, the Mann-Whitney test is applied to every combination of the MOMGA-II and a comparison MOEA. The results of the Mann-Whitney test are presented in Table 6.2.

Table 6.2 Nonparametric Mann-Whitney Results for Standard MOP Test Suite

MOP	**MOEA Pairs**	**Mann Whitney p-value for**		
		Gen Dist	**ONVG**	**Spacing**
MOP1	MOMGA-II - MOGA	0.0355	0.4813	0.0002
	MOMGA-II - MOMGA	0.6305	0.6305	0.9118
	MOMGA-II - NPGA	0.0015	0.0524	0.0232
	MOMGA-II - NSGA	0.0147	0.1051	0.0232
MOP2	MOMGA-II - MOGA	< 0.0001	0.0892	0.0068
	MOMGA-II - MOMGA	0.7394	0.0288	0.0630
	MOMGA-II - NPGA	0.0007	0.3150	0.0630
	MOMGA-II - NSGA	< 0.0001	< 0.0001	< 0.0001
MOP3	MOMGA-II - MOGA	0.0039	< 0.0001	< 0.0001
	MOMGA-II - MOMGA	0.0147	0.0015	0.0010
	MOMGA-II - NPGA	0.0005	< 0.0001	< 0.0001
	MOMGA-II - NSGA	0.0115	0.2176	0.0089
MOP4	MOMGA-II - MOGA	0.0232	0.0007	0.0010
	MOMGA-II - MOMGA	0.9118	0.7394	0.9705
	MOMGA-II - NPGA	0.4359	0.0524	0.1051
	MOMGA-II - NSGA	< 0.0001	< 0.0001	< 0.0001
MOP6	MOMGA-II - MOGA	0.0001	0.0185	0.7394
	MOMGA-II - MOMGA	0.2475	0.2475	0.0355
	MOMGA-II - NPGA	< 0.0001	< 0.0001	< 0.0001
	MOMGA-II - NSGA	< 0.0001	< 0.0001	< 0.0001

In conducting an analysis of the results presented in Table 6.2, whenever $p-value \leq 0.05$, then the null hypothesis is rejected. Rejection of the null hypothesis means that with the specified level of confidence, one rejects the statement that the pair of MOEAs obtains statistically the same distributions of data applied to the respective MOP. Analysis of MOP 1 illustrates that the MOMGA-II and the MOMGA obtain statistically the same results across the three metrics. This statistically reinforces the statement that the two MOEAs obtain the same level of effectiveness. The MOMGA-II, in most cases obtains statistically different (better) results than the other MOEAs for MOP 1.

The Mann-Whitney results for MOP 2 reflect the same trend. The MOMGA-II obtains statistically the same results as the MOMGA for the tested metrics and statistically different (better) results as compared to the other MOEAs. The results of MOP 3 reflect a difference in almost every case between the results of the MOMGA-II and all of the MOEAs. It is noted that the MOMGA-II typically does not perform as well as the other MOEAs on this MOP. This reflects a difficulty of the MOMGA-II in attempting to solve MOP 3. The MOMGA-II's smaller population size as compared to the MOMGA prevents it

from finding all the good BBs in each data run. A larger population size should correct this result but was not tested for MOP 3. MOPs 4 and 6 reflect similar results of the Mann-Whitney testing as do MOPs 1 and 2. The MOMGA-II obtains statistically the same results as the MOMGA and statistically different (better) results than the other MOEAs tested. Overall these results show that statistically, over the tested MOPs and metrics used, the MOMGA-II typically yields the same level of effectiveness as the MOMGA but with a greatly reduced population size. Additionally, the MOMGA-II typically performs as well or better than the other comparison MOEAs.

6.5.1.7 Standard MOP Test Suite Timing Results. The previous section discuss the effectiveness of the MOMGA-II as compared to other MOEAs. Through a statistical and visual analysis of the results of the tested algorithms, one can see that the MOMGA-II performs statistically similar to the MOMGA and similar or better than the selected MOEAs on the selected test MOPs. The results further validate BB concepts in the multiobjective problem domain. Additionally these results illustrate that the concepts of the fmGA are applicable to the multiobjective arena and that a similar level of effectiveness can be obtained through their use. Since the MOMGA-II obtains the same level of effectiveness as the MOMGA for MOPs 1, 2, 4, and 6, the next logical aspect to analyze is the efficiency and timing of the MOMGA-II.

The MOMGA-II is implemented with a similar data structure to the MOMGA implementation to allow for a fair comparison between the MOMGA-II and MOMGA without code optimization. One objective of the MOMGA-II is to develop an efficient and effective explicit BB-based MOEA. The results show that for the same parameter settings as the MOMGA, the MOMGA-II is able to obtain similar results to the MOMGA at an improvement of 60.11 to 97.62% of the median or 83.99 to 99.07% of the average execution time of the MOMGA.

The MOMGA-II and the MOMGA are both executed on the same hardware platform with the same parameter settings in order to conduct a fair comparison of the timing of the two MOEAs. The hardware system used is a SUN Ultra 1 Workstation with a 166.67 MHz CPU and 128 MB of RAM. Table 6.3 presents the maximum, minimum, median, average,

and standard deviation of the execution times of the MOMGA and MOMGA-II across the various MOPs tested. Table 6.4 presents the percentage improvement of the maximum, minimum, median, and average execution times of the MOMGA-II over the MOMGA.

Table 6.3 Timing Comparison (in seconds)

MOMGA					
Multiobjective Problem	Maximum Time	Minimum Time	Median Time	Average Time	Standard Deviation
MOP1	44.60	1.75	2.80	7.09	13.23
MOP2	168.66	2.43	60.09	64.84	58.14
MOP3	193.79	41.49	55.51	80.39	59.85
MOP4	198.29	3.03	180.26	125.03	83.86
MOP6	241.84	1.69	112.27	120.38	113.32
MOMGA-II					
Multiobjective Problem	Maximum Time	Minimum Time	Median Time	Average Time	Standard Deviation
MOP1	1.25	1.06	1.12	1.14	0.07
MOP2	1.52	1.38	1.43	1.44	0.04
MOP3	1.49	1.31	1.42	1.41	0.05
MOP4	1.43	1.28	1.36	1.35	0.05
MOP6	1.17	1.02	1.13	1.11	0.05

As can be seen in Tables 6.3 and 6.4, the MOMGA-II executes in much less time than the MOMGA on the tested MOPs, with a 97.62% or higher median timing improvement on all but MOP 1 over the MOMGA. The average timing improvement of the MOMGA-II ranges from 83.99% to 99.07%. The reason for this great improvement in execution time is that as the size of the BBs increase, the MOMGA must spend more time than the MOMGA-II in the creation and evaluation of the BBs. The MOMGA-II does not create every BB but probabilistically creates "good" BBs through the random deletion of bits from the population members until the correct BB size (string length) is obtained. The results from Figures 6.10 through 6.12 and Tables 6.1 and 6.2 show that the MOMGA-II is able to find "good" BBs since the MOMGA-II statistically obtains similar results to the MOMGA over the tested MOPs.

To further analyze the execution time results obtained, the Mann-Whitney test is used to statistically state that the MOMGA-II and MOMGA timing results are distributed identically or not. The null hypothesis, H_0, states that the distributions of the data from the two tested MOEAs as applied to the MOP are identical. The alternate hypothesis, H_a, states that the two data distributions are not identical. In comparing the timing results

Table 6.4 MOMGA-II Execution Time Improvement Over MOMGA

Multiobjective Problem	Maximum Time %	Minimum Time %	Median Time %	Average Time %
MOP1	97.20	39.43	60.11	83.99
MOP2	99.10	43.21	97.62	97.78
MOP3	99.23	96.84	97.45	98.25
MOP4	99.28	57.76	99.25	98.92
MOP6	99.52	39.64	99.00	99.07

of the two MOEAs, α is selected to be 0.10 and one rejects the null hypothesis whenever $p - value \leq \alpha$. The p-values presented in Table 6.5 are calculated for the average timing comparison of the MOMGA-II and the MOMGA.

Table 6.5 Nonparametric Mann-Whitney Results for Standard MOP Test Suite Timing

MOP	**MOEA Pairs**	**p-value for Average Timing**
MOP1	MOMGA-II - MOMGA	< 0.0001
MOP2	MOMGA-II - MOMGA	< 0.0001
MOP3	MOMGA-II - MOMGA	< 0.0001
MOP4	MOMGA-II - MOMGA	< 0.0001
MOP6	MOMGA-II - MOMGA	< 0.0001

Since each of the p-values is less than an α value of 0.10, one can state with confidence that the null hypothesis is rejected and the MOMGA-II and MOMGA have statistically different distributions of timing data. The same is true of the comparison of the max, min, and median timing values. It is shown here, with statistical confidence that the MOMGA-II obtains a similar level of effectiveness as the MOMGA and obtains statistically better execution times than the MOMGA over the same MOPs.

6.5.1.8 Standard MOP Test Suite Summary. Through the testing presented, the MOMGA-II has been shown to be more efficient in terms of execution time and just as effective as the MOMGA. The MOMGA-II results illustrate a huge improvement in the execution time of the MOMGA-II over the MOMGA ranging from an average improvement of 84% to 99% when applied to a limited MOP test suite. This execution time improvement results from the MOMGA-II's use of the PCI initialization of the population which does not have the computation bottlenecks imposed by PEI used in the MOMGA and the advantages that the BBF phase introduces. Additional focus on input

parameters may yield better performance results as the testing completed used a generic test parameter value set that favors the MOMGA.

Mann-Whitney paired tests of the MOMGA-II and the MOMGA illustrate no statistical difference in the results of the two MOEAs. The results indicate that the MOMGA-II is an effective explicit BB-based MOEA. These results indicate that the objective of designing and implementing an efficient explicit BB-based MOEA for solving MOPs is met. The MOMGA-II is the only known efficient and effective explicit BB-based MOEA to exist. The subsequent sections test the MOMGA-II in more detail to reinforce this statement.

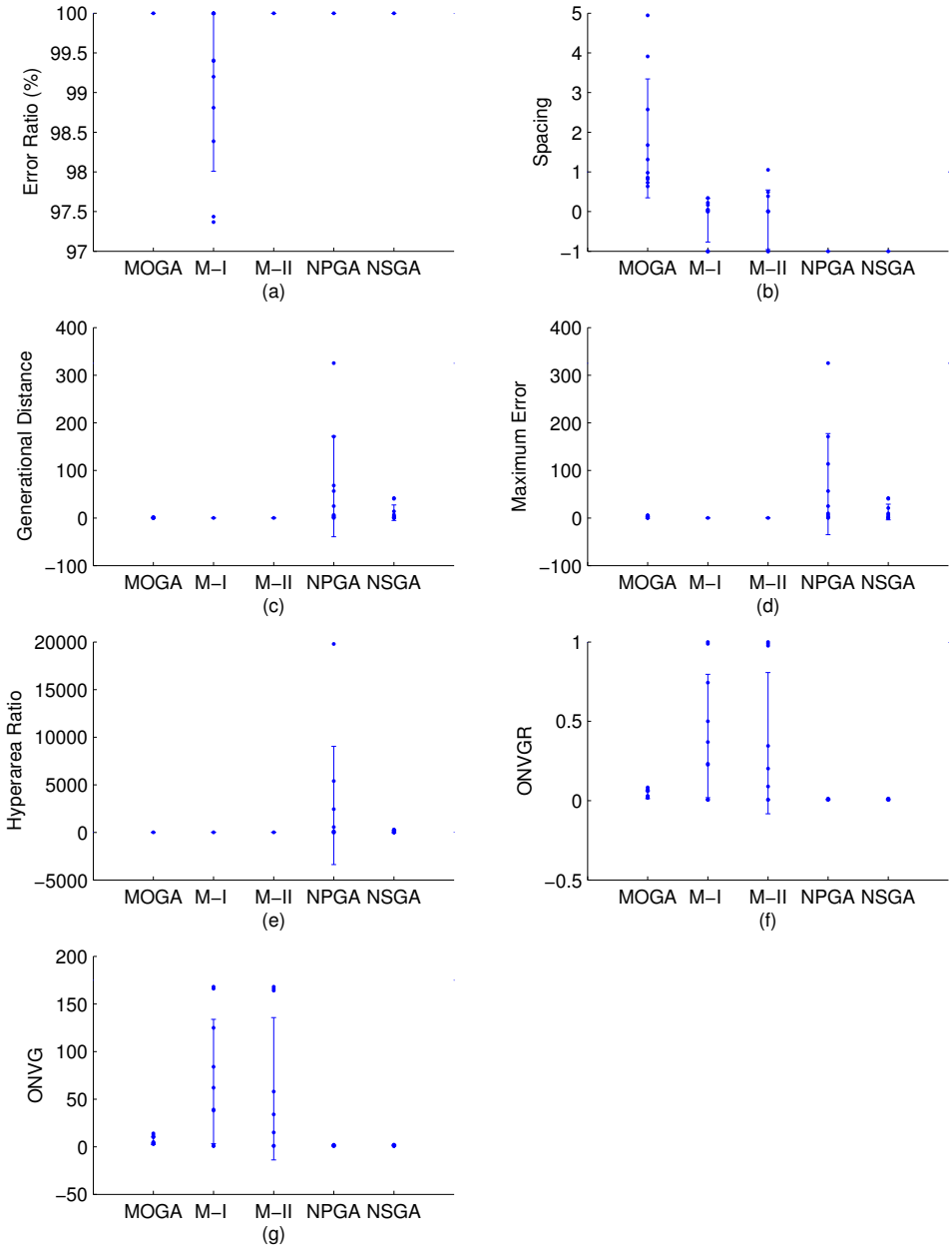

Figure 6.13 MOMGA MOP1 Metrics

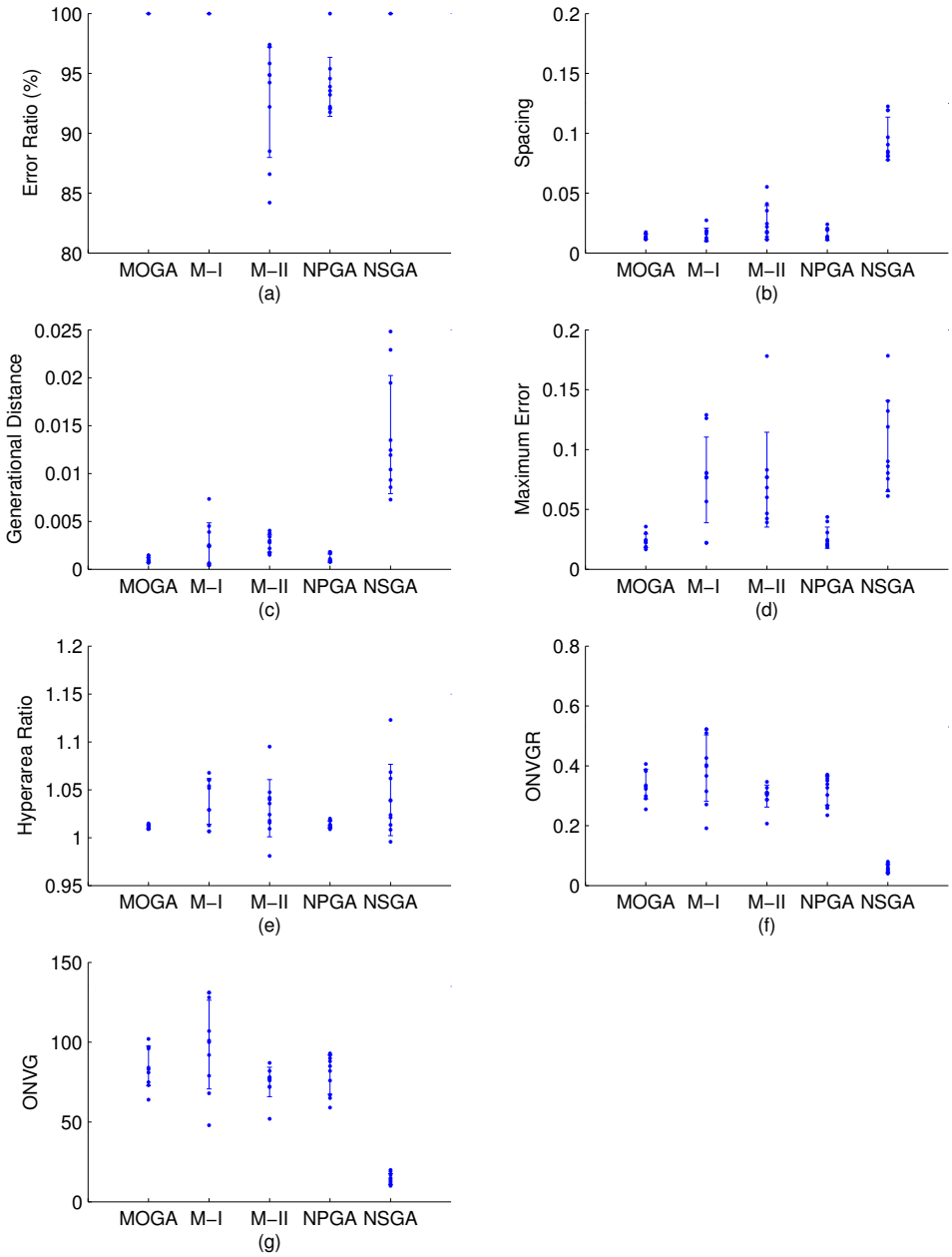

Figure 6.14 MOMGA MOP2 Metrics

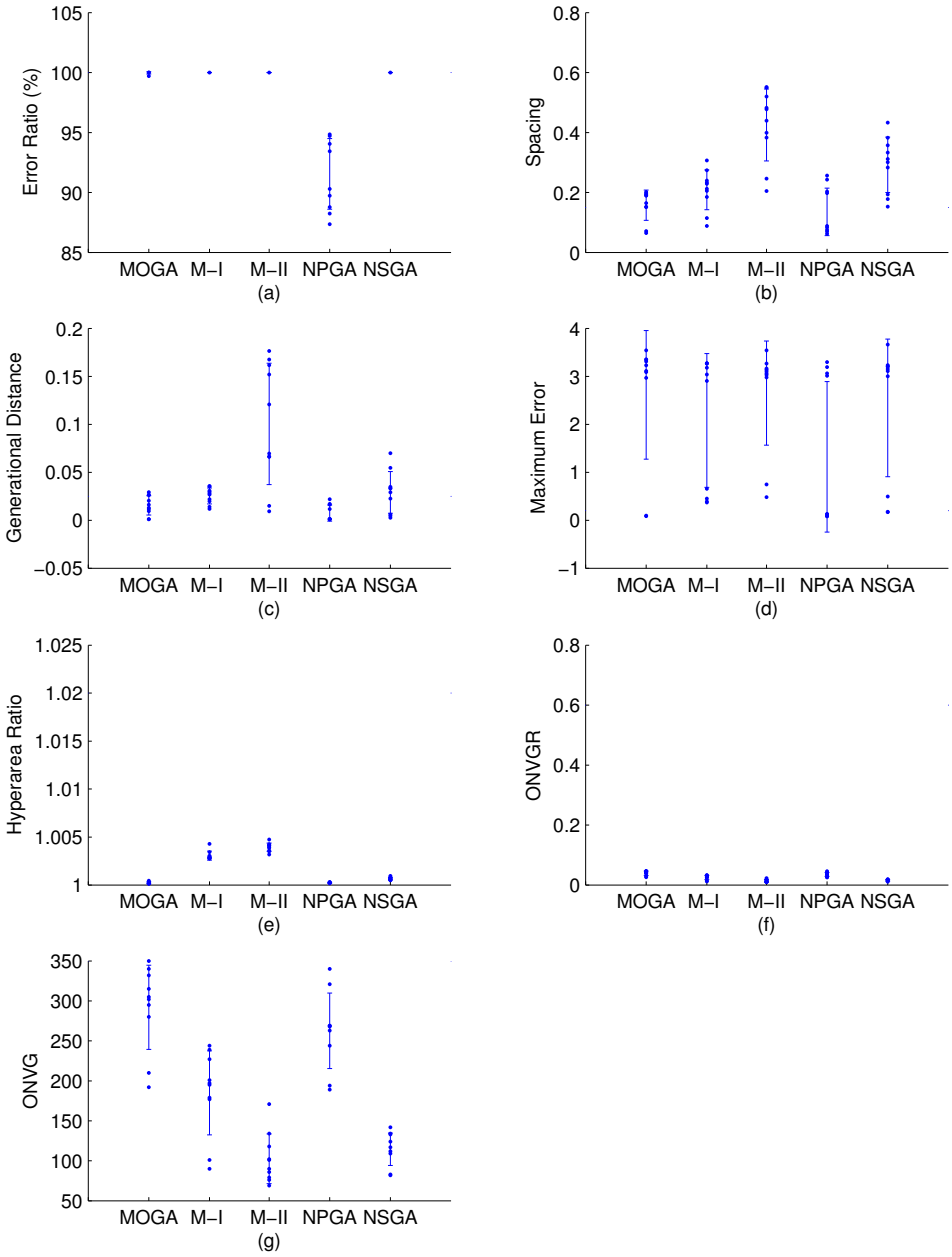

Figure 6.15 MOMGA MOP3 Metrics

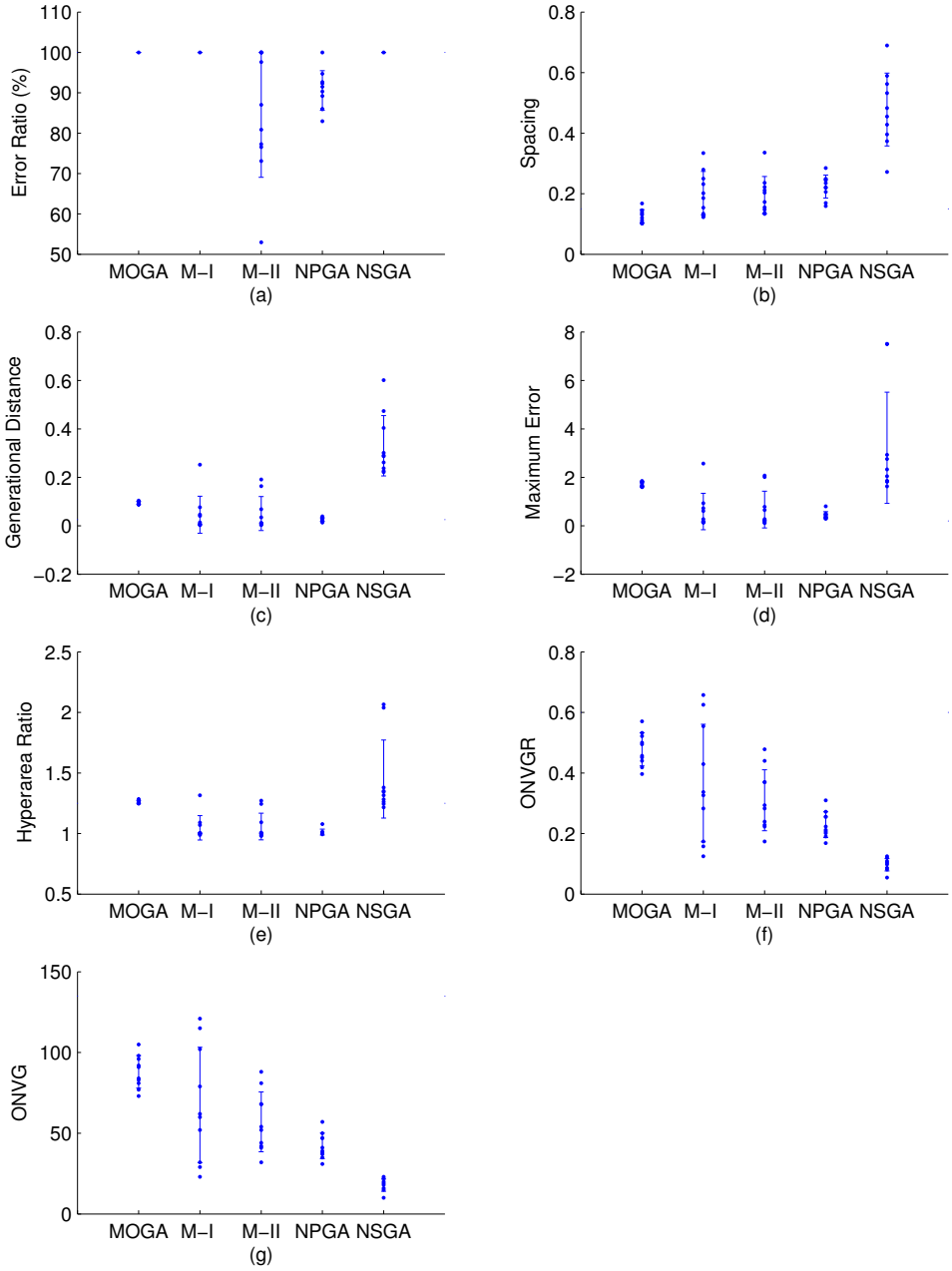

Figure 6.16 MOMGA MOP4 Metrics

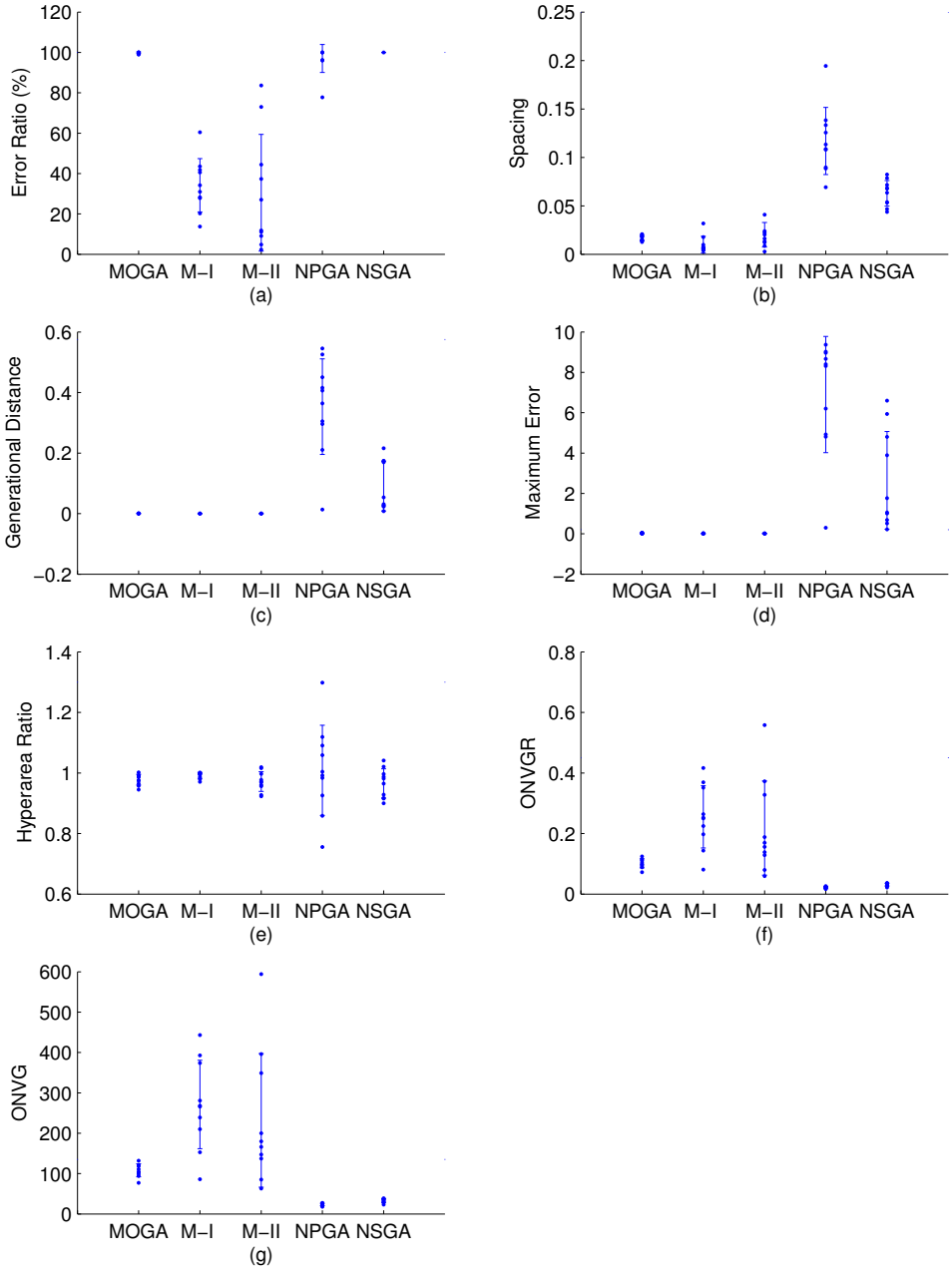

Figure 6.17 MOMGA MOP6 Metrics

6.5.2 Constrained MOP Test Suite Experimental Results. Applying MOEAs to attempt to solve constrained MOPs is an important research area to address. Many real-world MOPs are constrained and improving the performance of an MOEA as applied to constrained MOPs is of high interest to MOEA researchers. Two constrained MOPs are selected from Chapter IV in order to evaluate the performance of the MOMGA-II when applied to a limited constrained MOP test suite. Specifically, MOP-C1 and MOP-CT are selected for testing. Each of these constrained MOPs has varying characteristics that make it a good test MOP. MOP-C1 is a two fitness function constrained MOP, containing a single convex Pareto front. MOP-CT is a tunable constrained MOP based off of the Tanaka MOP [182]. MOP-CT is a modified formulation of the Tanaka MOP, in which the MOP formulation contains variables that are used to change the characteristics of the problem. A researcher can actively modify the constraint parameters of MOP-CT which results in a change to the characteristics of the Pareto front. Modifying the constraint parameters effects the cardinality and shape of the Pareto front, as well as the size and shape of the feasible region. In essence this results in many possible test MOPs that can be generated from MOP-CT.

6.5.2.1 MOP-C1 Results. The results of the MOMGA and MOMGA-II as applied to the constrained test function MOP-C1 are presented in Table 6.6. Values indicate that the MOMGA and MOMGA-II obtain statistically equivalent results for the generational distance metric. Additionally, the Mann-Whitney test implies similar distributions using this metric. The generational distance metric is important since it is a measure of how close PF_{known} is to PF_{true} for each MOEA. The other two metrics, spacing and ONVG illustrate that the MOMGA generates more potential solutions and hence obtains a more uniform distribution of the points along the Pareto front as compared to the MOMGA-II. Statistically the MOMGA results are different and slightly better than the MOMGA-II. As stated in Section 6.5 of this chapter, the better performance of the MOMGA is attributed to the increased number of juxtapositional generations that the MOMGA executes. The execution of additional juxtapositional generations results in an increased mixing of the BBs found. Additionally, the MOMGA uses a much larger population size than the MOMGA-II. The results indicate that the MOMGA-II does not find

all of the required good BBs or mix the BBs enough to generate the same number of solutions as the MOMGA. Increasing the population size (but maintaining a population smaller than the MOMGA) and increasing the number of juxtapositional generations of the MOMGA-II should yield an increased performance comparable to the MOMGA as applied to MOP-C1.

Table 6.6 Constrained MOP

Cardinality					
MOEA	Maximum	Minimum	Median	Average	Stand Dev
MOMGA	1823.00	805.00	1529.00	1418.90	346.60
MOMGA-II	443.00	347.00	396.50	398.00	32.73
Spacing					
MOEA	Maximum	Minimum	Median	Average	Stand Dev
MOMGA	0.2825	0.1150	0.1509	0.1717	0.0600
MOMGA-II	0.8666	0.5392	0.7182	0.7007	0.1048
Generational Distance					
MOEA	Maximum	Minimum	Median	Average	Stand Dev
MOMGA	0.0232	0.0096	0.0140	0.0149	0.0042
MOMGA-II	0.0250	0.0138	0.0190	0.0193	0.0035

Increasing the number of juxtapositional generations allows the MOMGA-II to more effectively search the space for feasible solutions to this MOP. The results generated by the MOMGA-II are based off of the same generic parameter set used in testing MOPs 1, 2, 3, 4, and 6. The results presented illustrate that the MOMGA-II obtains good results that are close to the true solution set for a constrained MOP.

6.5.2.2 MOP-CT Results. MOP-CT is selected for testing due to the MOP formulation that allows a researcher the ability to easily change the characteristics of the MOP solution set. Four different variants of MOP-CT, described in Chapter IV, are tested. Since other MOEAs have not been applied to the variants of MOP-CT presented, the results of the MOMGA-II are only compared to the true solution, PF_{true} .

The four variants of MOP-CT selected for testing are (summarized from Chapter IV):

1. Standard Tanaka phenotype with $a = .1$ and $b = 16$. The Pareto front represents 5 disconnected curves and is symmetric about the line $x = y$. (Figure 4.1)
2. Smaller continuous phenotype regions with $a = .1$ and $b = 32$ with larger infeasible regions between Pareto front points as compared to the standard settings. The

Pareto front represents 8 disconnected curves and is symmetric about the line $x = y$. (Figure 4.2)

3. Increased distance between Pareto front regions with, $a = .1(x^2 + y^2 + 5xy)$ and $b = 32$ with deeper infeasible regions between Pareto front points as compared to the standard settings. The Pareto front represents 16 disconnected curves (some are individual points) and is symmetric about the line $x = y$. (Figure 4.5)

4. Increased distance between Pareto front regions with, $a = .1(x^2 + y^2 + 5xy)$ and $b = 8(x^2+y^2)$ with deeper non-periodic infeasible regions between Pareto front points as compared to the standard settings. The Pareto front represents 6 disconnected curves (some are individual points) and is not symmetric about the line $x = y$. (Figure 4.6)

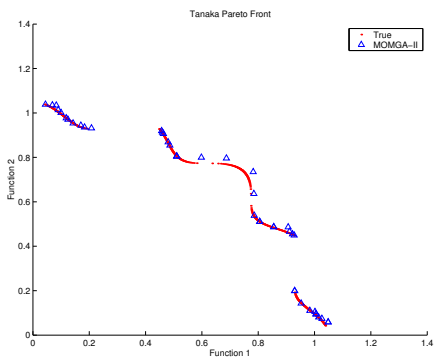

Figure 6.18 MOP-CT, $a = .1, b = 16$, PF_{true} vs PF_{known}

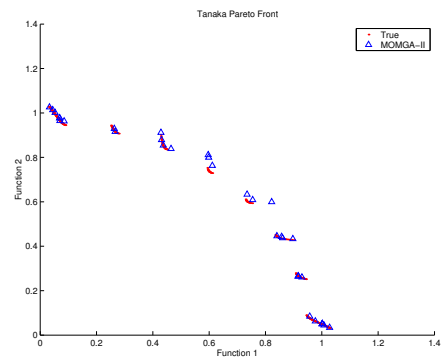

Figure 6.19 MOP-CT, $a = .1, b = 32$, PF_{true} vs PF_{known}

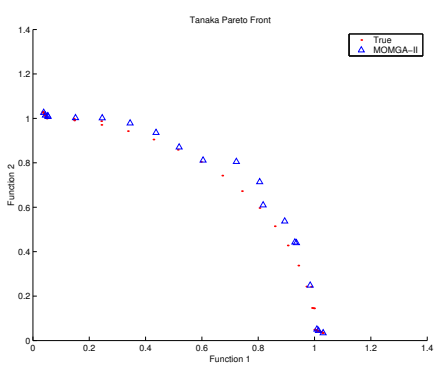

Figure 6.20 MOP-CT, $a = .1(x^2 + y^2 + 5xy), b = 32, PF_{true}$ vs PF_{known}

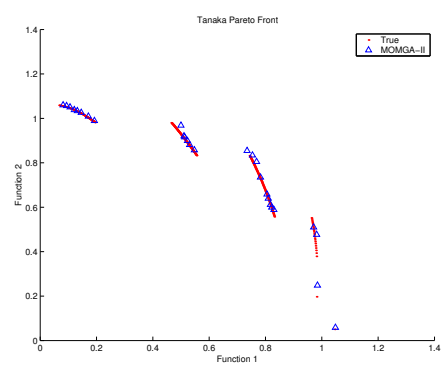

Figure 6.21 MOP-CT, $a = .1(x^2+y^2+5xy), b = 8(x^2+y^2)$, PF_{true} vs PF_{known}

The results of the MOMGA-II, PF_{known}, are visually compared to PF_{true} in Figures 6.18 through 6.21. The same parameter settings is used in the application of the MOMGA-II to MOP-CT variants as described in Section 6.5.1 of this chapter. The parameter set is not optimized and illustrates that the MOMGA-II can generate good solutions to a somewhat difficult constrained MOP without optimizing the parameters. Generating solutions that are within a local area of the true solutions is useful in order to determine the general structure of the Pareto front and optimize MOEA parameter settings based on the structure and characteristics of the Pareto front. Generating solutions within a local area of the true solutions allows a researcher the ability to isolate the areas of interest for further exploration.

The results from the first formulation of MOP-CT are presented in Figure 6.18. The MOMGA-II generates many of the solutions in PF_{true} and obtains a good distribution of points across the front. However, the center section of the PF_{true} is not covered as well as the rest of the true Pareto front. The center section is identified in Chapter IV, Section 4.2 as being a more difficult area of the front for an MOEA to generate solutions. The difficulty in generating solutions on the center of PF_{true} is partly due to the slope of the true Pareto front approaching 0 in this area.

The results from the second formulation of MOP-CT are presented in Figure 6.19. The second formulation of the MOP-CT MOP is a more difficult problem for an MOEA as PF_{true} is partitioned into multiple disconnected regions and a large infeasible region exists. The MOMGA-II obtains better results applied to this formulation of MOP-CT than the previous MOP formulation. The MOMGA-II also generates points on each disconnected portion of the front except the two center sections thereby obtaining a good estimate of the true Pareto front.

The MOMGA-II is also applied to a more difficult formulation of MOP-CT, presented in Figure 6.20. The third instantiation of MOP-CT has an increased distance between points on the true Pareto front and a larger infeasible region as compared to the first two formulations of this problem. The MOMGA-II generates solutions that are close to PF_{true} but has an increased difficulty in generating solutions on PF_{true}. This is attributed to the larger infeasible regions existing between points in PF_{true} and the difficulty

associated with generating discrete Pareto fronts versus those forming a curve or surface. However, the MOMGA-II again generates a good estimate of PF_{true}. Again the center portion of the front poses difficult for the MOMGA-II. Possibly the use of larger BB sizes would improve the effectiveness.

The last tested formulation of MOP-CT is presented in Figure 6.21. The last problem formulation is the most difficult of the four tested. The difficulty is due to the disconnected Pareto front regions, the large non-uniform infeasible regions between Pareto front solutions, and the non-symmetric characteristic of the front. The MOMGA-II performs well when applied to this problem, generating solutions on each disconnected region of the front with the exception of the discrete points on the true Pareto front. Again the MOMGA-II obtains a good estimate of the true Pareto front.

6.5.2.3 Constrained MOP Test Suite Summary. The results of the application of the MOMGA-II to both MOP-C1 and MOP-CT illustrate the robustness of the MOMGA-II in generating solutions that are a good estimate of PF_{true}. The results are obtained from using a generic set of parameter values in testing the MOMGA-II. The constrained MOPs tested have varying characteristics in their genotype and phenotype spaces that make them good test MOPs and may be representative of other MOPs that researchers are interested in attempting to solve. The MOMGA-II generates good results and most of the points generated are on or very close to points in PF_{true}. The visual presentation of PF_{known} illustrates that the MOMGA-II also obtains a good distribution of points across PF_{true} and accurately reflects the overall structure of the true Pareto front. Overall the MOMGA-II performs well when applied to a limited set of constrained MOPs.

6.5.3 Real-World MOP Test Suite Experimental Results. The objective of many MOEA researchers is to develop an efficient and effective MOEA for application to real-world MOPs. The previous sections of this chapter present experiments and analyses illustrating the efficiency and effectiveness of the MOMGA-II when applied to a limited test suite. In this section, the MOMGA-II is applied to a real-world and an NP-Complete constrained MOP formulated with a much larger number of decision variables.

In order to further test the effectiveness of the MOMGA-II, two MOPs indicative of some real-world MOPs are selected from Chapter IV for testing. The Advanced Logistics Problem (ALP) and the Modified Multiobjective Knapsack Problem (MMOKP) are selected and tested. Each of these MOPs are constrained and are difficult problems for an MOEA to solve. The ALP and MMOKP problems are formulated with integer based decision variables. The ALP problem has strict equality constraints and the MMOKP problem has inequality constraints. The results of the MOMGA-II are compared to the true Pareto front in the smallest instantiation of the ALP MOP. A limited number of MOEA researchers have applied MOEAs to the MMOKP MOP.

6.5.3.1 Advanced Logistics Problem Results. The ALP MOP, discussed in detail in Chapter IV, is a real-world Air Force MOP selected for testing. The ALP problem is a constrained, integer based, discrete MOP in which the feasible space is greatly reduced through strict equality constraints present in the MOP formulation. Two instantiations of the ALP MOP are tested; the 60 bit and 120 bit formulations. The difference in the two ALP formulations lies in the range of decision variable values and the constraints used (as detailed in Chapter IV). The number of feasible solutions present in the 60 bit version is $630,630$ out of a total 2^{60} possible points found through an enumeration of the space. The true Pareto front is generated through a total enumeration of the search space. Generating PF_{true} is useful for evaluating the performance of the MOMGA-II. One can see that the number of feasible points in the 60 bit formulation is $5.47 * 10^{-11}\%$ of the total solution space. Problems formulated in a manner in which the feasible space is an extremely small percentage of the overall space are typically difficult MOPs for MOEAs to solve without the use of some type of constraint handling approach.

The application of an MOEA to a class of MOP containing few feasible points creates difficulties that an MOEA must surpass in order to generate any feasible points throughout the search process. A random initialization of an MOEA's population may not generate any feasible points in a constrained MOP. Without any feasible solutions in the population, one must question whether or not the MOEA can even conduct a worthwhile search. In problems where the feasible region is greatly restricted, it may be impossible to create

a complete initial population of feasible solutions randomly. Without feasible population members, any MOEA is destined to fail. Feasible population members contain the BBs necessary to generate good solutions. It is possible for an infeasible population member to contain a BB that is also present in a feasible solution. As it is also possible for mutation to generate a feasible population member from a infeasible population member. However, typically feasible population members contain BBs that are not present in infeasible population members. EVOPs applied to feasible members tend to yield better results than EVOPs applied to infeasible population members. Therefore, it is critical to initialize and maintain a population of feasible individuals. For example, in the initial application of the MOMGA-II to the ALP MOP, the initial population yielded 0 feasible members across 30 data runs and in turn 0 feasible members were generated throughout the entire MOMGA-II execution. In each of the 30 data runs, the parameter settings included a population size of 900, BB sizes 1-4, 40 generations per BB size, cut probability of 2%, splice probability of 100%, and the remaining standard settings for the MOMGA-II. This illustrates the motivation for maintaining a population of feasible individuals.

In general MOEA researchers have not experimented with or studied constrained MOPs in much detail even though many real-world MOPs are constrained. One of the issues that must be addressed in applying an MOEA to a constrained MOP is in the selection of a constraint handling mechanism. The details of many constraint handling mechanisms are discussed in Chapter II, Section 2.8. It is important to note that the best constraint handling approach is problem dependent. Two methods that are used in applying the MOMGA-II to highly dimensional MOPs include: the removal of infeasible population members and the repair of infeasible population members.

The removal of infeasible members is a poor choice of a constraint handling approach in MOP formulations consisting of a greatly restricted feasible region. In such MOP formulations, the majority of the space is infeasible and hence a random generation of the population typically yields few to no feasible members. Remember that the MOMGA-II explicitly manipulates BBs and must identify good BBs in the population in order to effectively generate good potential solutions to an MOP. Therefore, a large percentage, if not all, of the randomly generated initial population of the MOMGA-II must contain

feasible population members to yields good population members. The random generation of a single feasible solution could take a considerable amount of time and hence removal of infeasible members is not a practical constraint handling approach in MOPs with small feasible regions.

In some MOPs the MOP formulation and the constraints allow for the use of a repair mechanism. Repairing population members involves mutating the decision variables in order to generate a feasible population members. Repairing population members may be the best constraint handling method for MOPs with small feasible regions and containing integer based decision variables. The ALP and MMOKP are constrained MOPs formulated with integer based decision variables and hence are candidates for the use of a repair operator.

The multiobjective advanced logistics problem (MOP-ALP) is concerned with allocating appropriate resources for use in various tasks [199]. The ALP MOP is summarized in this section but is discussed in detail in Chapter IV, Section 4.5. In the ALP MOP, resources are categorized into different package types and are assigned to various tasks. Resources are distinguished by their suitability towards tasks ($0 \leq \alpha \leq 1$), weight consumption (β), and volume consumption (λ). The objectives are to maximize overall suitability while minimizing weight and volume consumption. The ALP MOP formulation, consisting of: A asset types, m tasks, n MRR types, at a resource level k, and decision variables $(x_{1,1}, x_{i,j}, \ldots, x_{m,n})$, where x is restricted to a non-negative integer value, requires a researcher to maximize:

$$S = \sum_{j=1}^{n} \sum_{i=1}^{m} a_{i,j}\ x_{i,j}\ , \tag{6.4}$$

minimize:

$$W = \sum_{j=1}^{n} \sum_{i=1}^{m} \beta_j\ x_{i,j}\ , \tag{6.5}$$

and minimize:

$$V = \sum_{j=1}^{n} \sum_{i=1}^{m} \lambda_j\ x_{i,j}\ . \tag{6.6}$$

Subject to:

$$\sum_{j=1}^{n} x_{i,j} = RLtask_{k,i} \text{ for } i = 1 \ldots m \ , \tag{6.7}$$

$$\sum_{i=1}^{m} x_{i,j} \leq RLmrr_{k,j} \ , \tag{6.8}$$

$$\text{for } A = 1 \ldots a \text{ and } P_a = \text{ number MRR types for } a,$$

where $a_{i,j}$ is the suitability of MRR j for Task i and $x_{i,j}$ is the number of MRRs j allocated to task type i.

The equality constraints directly influence the length of the bit string as the constraints represent the largest integer value that any decision variable can take. In the smallest formulation of the ALP MOP tested, the constraints ($RLtask_{k,i}$) are set to 10, 5, and 1. The constraints are discussed in more detail in Chapter II, Section 2.8 and each constraint represents the linear summation of five different decision variables. The constraint values result in decision variables of 4 bits in length since 4 bits are required to represent the largest possible decision variable value of 10. Hence an overall string length of 60 bits is required. The largest instantiation of the MOP ALP tested defines the constraint ($RLtask_{k,i}$) values as 60, 90, and 150. The constraint values result in decision variables of 8 bits in length since 8 bits are required to represent the largest possible decision variable value of 150. Hence an overall string length of 120 bits is required. The 60 bit instantiation of the ALP MOP is completely enumerated for testing purposes in order to compare the results of the MOMGA-II to the true solution to the MOP.

Initially, the application of the MOMGA-II to the ALP MOP did not integrate a constraint handling mechanism. The advantage of this approach is that computational power is not expended to check the constraints after each population member evaluation. The disadvantage of not using a constraint handling approach is apparent once an MOP with a small feasible region is encountered. As discussed, the random generation of the initial population typically yields few feasible population members in MOPs with small feasible regions. Without the use of a constraint handling approach the MOMGA-II, which uses an elitist selection mechanism, promotes population members that have the

best fitness values regardless of their feasibility. Hence, the MOMGA-II attempts to solve the MOP with the best feasible or infeasible BBs generated. In reality, the MOMGA-II attempts to solve the unconstrained MOP formulation and applies the constraints during post-processing. If the true Pareto front for both the unconstrained and constrained MOP formulations are relatively close together, then such a post-processing constraint handling approach can generate good solutions. In other cases, as exists in the ALP MOP, post-processing the constraints does not yield any feasible solutions. This is attributed to the small feasible region and the inability of the MOMGA-II to generate good feasible BBs from the infeasible population members.

In unconstrained optimization, good BBs are defined by Definition 10 (see Chapter II, Section 2.5.2). However, in constrained optimization the definition of a feasible BB is different from that of a good BB. In the context of constrained optimization, a feasible BB is defined as:

Definition 43 (Feasible Building Block): *A feasible multiobjective BB contains a combination of gene values (alleles) that are located in distinct but not necessarily adjacent locations (loci) and hence meets the requirements of Definition 1. There exists a set of alleles for the loci not specified in the BB that yields a feasible population member with respect to the constraints of the MOP formulation.* □

A feasible BB in the MOMGA-II is a BB that when combined with one of the competitive templates yields a feasible population member for evaluation. It is important to note that a feasible BB may become infeasible if a different competitive template or set of allelic values is used to evaluate the BB. This manifests itself in the execution of the MOMGA-II. A BB that is feasible in one iteration of the MOMGA-II may be infeasible in another iteration if a different competitive template is used. A strict feasible BB is defined as:

Definition 44 (Strict Feasible Building Block): *A strict feasible multiobjective BB contains a combination of gene values (alleles) that are located in distinct but not necessarily adjacent locations (loci) and hence meets the requirements of Definition 1. To be classified as a strict feasible BB, the combination of the strict feasible BB with every*

possible set of alleles for the loci not specified must yield a feasible population member with respect to the constraints of the MOP formulation. □

A strict feasible BB may not exist for every MOP formulation.

Without the use of a constraint handling approach, the MOMGA-II's initial population contains no feasible population members in application to the ALP MOP. An infeasible population is a poor starting point for the MOMGA-II as it uses good BBs to generate good results. The MOMGA-II and many other MOEAs were designed using the concept of the BBH. Hence, these MOEAs assume that good BBs are present in the initial population and in later generations. A population of infeasible solutions contains no feasible BBs and may not lead to the generation of any feasible population members over MOEA execution. A constraint handling method is necessary to generate feasible population members and good results for the ALP MOP.

The integration of a constraint handling approach into the MOMGA-II requires some thought. Considering that the two constrained MOPs of interest, the ALP and MMOKP MOPs, are discrete MOPs formulated with integer based decision variables and linear constraints, a repair mechanism is implemented. Since the constraints are linear and the decision variables are integer based, it is relatively easy to repair population members in an MOP of this type. A repair mechanism is used in the form of a mutation operator that mutates a population member until it is feasible.

There are many locations within the MOMGA-II that a repair operator can be integrated. Specifically a repair operator may be used in any or all of the three phases (initialization, BBF and juxtapositional) of the MOMGA-II. Each of the phases are discussed in detail along with the rationale for using a repair operator within each phase. The initialization and BBF phases of the MOMGA-II require feasible BBs to be present in order to increase the probability of generating good feasible solutions. If feasible BBs are not present within the first two phases then the BBF process is wasted. The BBF phase does not identify feasible BBs for the MOMGA-II to generate feasible solutions of high quality. Therefore, the initialization and BBF phases should maintain a feasible popula-

tion in order for the MOMGA-II to effectively identify the good BBs and generate good feasible results.

Another factor that must not be overlooked in the MOMGA-II is the feasibility of the competitive templates. Even if the assumption is made that a feasible template is used, the combination of the competitive template and feasible or infeasible BBs is not guaranteed to yield a feasible population member. Therefore a BB that is feasible with respect to template A in generation t may not be feasible with respect to template B in generation $t+1$. It is important to recognize that the competitive template has a large effect on the population when a large number of loci in the population members are unspecified. As the strings increase in length, in the juxtapositional phase, the competitive template has a lesser effect as more bits become specified in each of the population members. Maintaining a feasible competitive template does not guarantee a feasible population is generated and hence a repair operator is used to ensure feasible potential solutions are generated. During the juxtapositional phase, the BBs identified in the BBF phase are combined. Again the potential exists for the feasible BBs to be combined in a manner that yields an infeasible population member. A repair mechanism is also implemented in the juxtapositional phase.

To avoid the potential of generating few feasible solutions, the simple post-constraint handling method was deemed inappropriate as well as the method suggested by Deb [46, 50]. The approach used in the MOMGA-II's application to the ALP MOP does not rely on constraint handling being coupled with the selection mechanism but instead relies on a repair mechanism. The use of a repair mechanism allows the algorithm to proceed without modification to the existing selection routines.

To improve upon the effectiveness of the MOMGA-II as applied to the ALP MOP and generate feasible solutions, a constraint handling mechanism is used. The initialization of the population and the competitive template is a key issue in the effectiveness of the MOMGA-II. In the initialization phase, the population of the MOMGA-II and the competitive templates are randomly created. The random initialization is used in an attempt to uniformly distribute the population members across the search space. A uniform distribution of the population across the search space increases the diversity of the population in an attempt to identify multiple good BBs that are also feasible. Immediately following the

random initialization, the population and competitive templates are repaired. The repair ensures that the MOMGA-II starts with a feasible population and templates for use in the BBF phase.

The repair mechanism implemented in the MOMGA-II assumes an application to an MOP formulated with integer based decision variables and linear constraints as is the ALP problem. The following procedure is used to repair the population members as applied to the ALP MOP:

1. Combine the population member with a competitive template if necessary.
2. Randomly choose a locus position to start the repair.
3. Randomly choose a direction to move within the string (left or right).
4. Analyze each decision variable. A single bit or combination of bits represents each decision variable.
5. Proceed to scan the bit string in the specified direction and increment or decrement each decision variable by one unit until each constraint is met and the population member is feasible. If the end of the string is reached prior to achieving feasibility, loop back around to opposite end of the string and continue the repair process until the feasibility criteria is achieved. Maintain the decision variable constraints and ensure that each decision variable value is non-negative.

The random repair method described has processing overhead associated with it. The repeated feasibility analysis requires additional processing power but is necessary in order to maintain a feasible population. The advantage of random repair method is in the resulting uniform movement across the search space as the decision variables are modified one at a time. The population member is repeatedly scanned and repairs to prevent a single decision variable from being incremented or decremented by a huge amount in the repair process. A large change to one decision variable causes a large movement in one direction of the search space, while a uniform modification of the decision variables yields more of a local search to find the nearest feasible solution.

Once the repair mechanism terminates, all of the population members are feasible prior to entering the next generation. The population members are repaired prior to entering the BBF phase and during each juxtapositional generation. An analysis of the initial results of the MOMGA-II illustrate that without the repair method, the infeasible results are somewhat close to the feasible solutions in phenotype space. Hence the repair mechanism is only used in the initialization and juxtapositional phases. The repair mechanism is not used after each BBF operation in order to determine if the MOMGA-II could generate effective results without the additional overhead added by the repair process in the BBF phase.

As each generation of the juxtapositional phase executes, the population members are repaired. The population members are not repaired after each specific cut-and-splice operation in order to avoid convergence to a suboptimal solution and reduce the overhead associated with the numerous recombinations that take place within a single generation. However, at the end of each generation of the juxtapositional phase, all of the population members are repaired. The population is stored to an external archive to ensure that the feasible population members generated by the MOMGA-II are not lost. The next juxtapositional generation begins with feasible population members and the process repeats itself. Once termination criteria for the MOMGA-II is met, the results are presented to the researcher. At the conclusion of execution of the three phases of the MOMGA-II, all of the population members are feasible and the competitive templates are updated for the next BB execution. The repair process ensures that subsequent BB size executions begins with feasible competitive templates.

Prior to the implementation of the repair mechanism, the MOMGA-II generated 0 feasible solutions throughout its normal execution. Utilizing a method of throwing away all of the infeasible solutions and continuously generating solutions until a feasible one is found yielded 0 feasible solutions over the course of numerous randomly generated population members. Therefore a repair mechanism was implemented to fix infeasible solutions and give the MOMGA-II the good BBs necessary to find good solutions.

The ALP MOP (see Chapter IV, Section 4.5 for a detailed description of this MOP) has three objectives and yields a discrete PF_{true} set. The ALP MOP is a maximization

and minimization function consisting of 15 decision variables, 3 fitness functions and strict equality constraints. The feasible genotype space and phenotype solution spaces are discrete and are illustrated by the shaded area in Figure 6.22, where the (*)s represent PF_{true} .

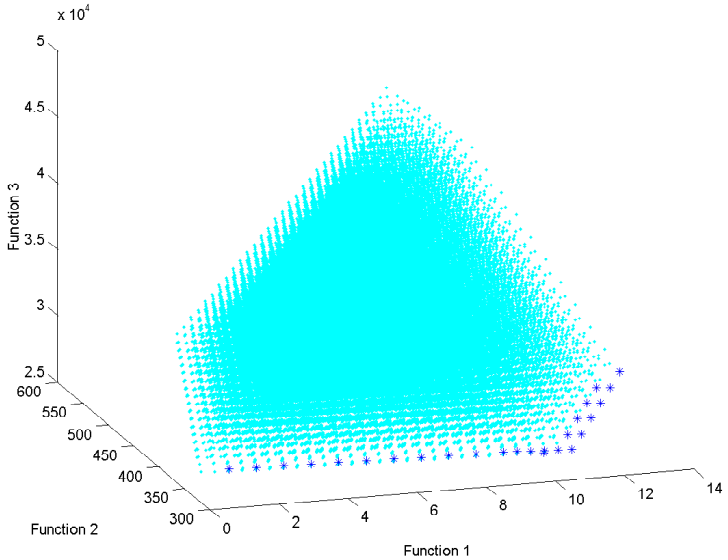

Figure 6.22 MOP-ALP Discrete 3D Integer Search Space

The results of the MOMGA-II with the repair mechanism as applied to the ALP MOP are presented in Figure 6.23. The true Pareto front, PF_{true} , is represented by (*)s and the known Pareto front, PF_{known} , generated by the MOMGA-II is represented with (△)s. One can see that the MOMGA-II with the repair mechanism generates all of the points in PF_{true} ! This illustrates how useful a repair mechanism is when applied to constrained MOPs formulated with integer based decision variables leading to small feasible regions in the space. Since the MOMGA-II always generates PF_{true} , no statistical data is presented. It is apparent that all of the metrics yield their optimal value when applied to the results of the MOMGA-II application to the ALP MOP. In essence the generational distance is 0, the ONVG is 26 and the spacing metric value is identical for PF_{known} and PF_{true} , which constitutes the best results one can achieve for the ALP MOP. No other known MOEAs have been applied to the ALP MOP and hence the MOMGA-II results are only compared to the true solution.

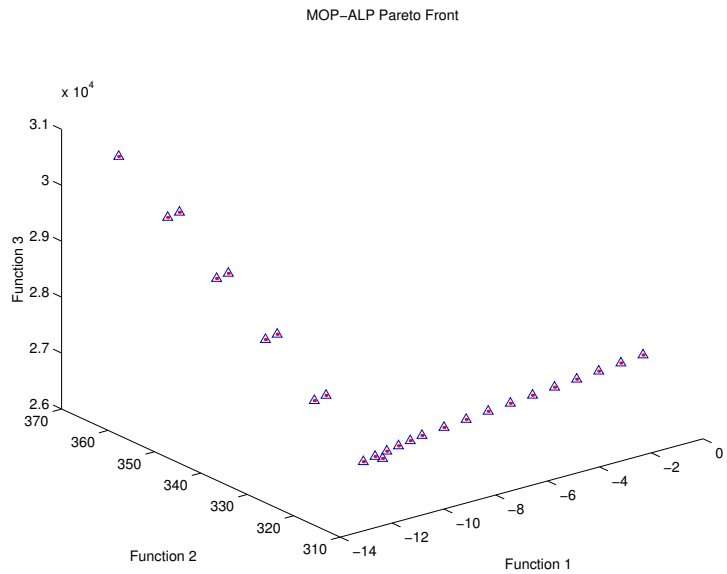

Figure 6.23 True Pareto Front Versus MOMGA-II Pareto Front

Figure 6.23 presents PF_{true} and PF_{known} results for the smallest (60 bit) instantiation of the problem. The results of applying the MOMGA-II to the largest instantiation (120 bit) of the ALP MOP is presented in Figure 6.24. Since the larger formulation of the ALP MOP is too large to determine what PF_{true} is within a reasonable amount of time, only PF_{known} for the MOMGA-II is presented. Again we see that the MOMGA-II clearly finds a Pareto front of the same structure as that in Figure 6.23. This PF_{known} solution set is anticipated to contain most of the points in PF_{true} as it has a similar structure and characteristics of the true solution to the 60 bit problem. Additionally, the MOMGA-II finds a "good" distribution of points across PF_{known} . This reinforces the usefulness of the repair mechanism applied to a discrete MOP.

All the decision variables can only take on a finite number of integer values in this MOP, hence, the resulting fitness landscape also can only take on a finite number of values (although very large). Of course a particular MOEA such as the MOMGA-II may generate infeasible values, but they can be appropriately repaired; i.e., becoming a feasible potential solution. Assuming that in finite time all finite values in the fitness landscape are generated by the MOMGA-II operators with repair, along with nondominated selection continually moving points toward the Pareto front, proper convergence to the complete Pareto front

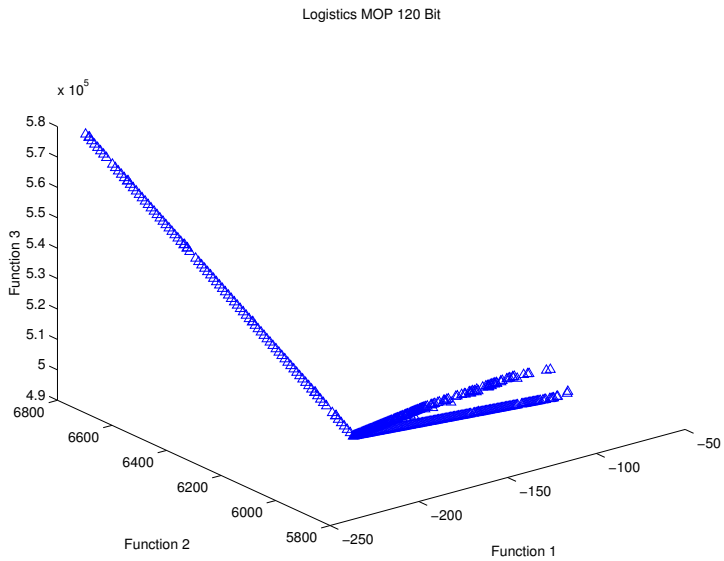

Figure 6.24 MOMGA-II Pareto Front - Expanded Problem

occurs. Note that the Pareto front consists of only a finite number of points for this class of integer problem. Observe that the MOMGA-II applied to the ALP MOP reflects this phenomena.

6.5.3.2 Modified Multiobjective Knapsack Problem (MMOKP). The MOMGA-II is applied to another constrained integer based decision variable MOP, the MMOKP MOP. The MMOKP MOP formulation contains a large number of decision variables. MOEAs are suited to attempt and solve problems of high dimensionality and hence the MOMGA-II is suited for application to the MMOKP [148]. The MMOKP is formulated in 100, 250, 500, and 750 item formulations with integer based decision variables and real-valued fitness functions. A more detailed description of the MMOKP MOP is presented in Chapter IV, Section 4.4.1. While the MMOKP MOP formulation used does not reflect the true multiobjective formulation of the knapsack problem [148] due to the constraint that any item placed into one of the knapsacks must also be placed into all of the knapsacks, the MMOKP MOP remains a good test problem due to the large number of decision variables and the difficulty associated with generating solutions on the Pareto front. Many MOEA researchers have selected this MOP to test their MOEAs [99, 100, 112, 123, 168, 207, 211, 213, 215]. The MMOKP MOP has been se-

Table 6.7 MOMGA-II Parameter Values for MMOKP

Parameter	Value
Eras	10
BB Sizes	1-10
P_{cut}	2%
P_{splice}	100%
string length	100, 250, 500, 750
Total Generations	100

lected for testing due to the difficulty of MOEA approaches in finding good solutions to this problem and to evaluate the performance of an explicit BB-based MOEA approach as applied to this MOP. Since the MMOKP MOP has similar characteristics to other MOPs and real-world MOPs, it is a good test problem to use.

The results of the MOMGA-II as applied to the MMOKP MOP using the default parameter settings for the MOMGA-II are presented. The MMOKP problem specific settings used are presented in Table 6.7. The MOMGA-II results are taken over 30 data runs in order to compare the results of the MOMGA-II to other MOEAs also executed over 30 data runs. The MOMGA-II was run on a SUN Ultra 10 with a single 440 MHz processor, 1024 MB of RAM, and using the Solaris 8 operating system.

The MMOKP is a modification of the of the Multiobjective Knapsack problem (MOKP) and is a difficult problem for an MOEA to solve [100, 215]. The MMOKP has similar characteristics to certain real-world logistics problems and is discussed in detail in Chapter IV, Section 4.4.1. The overall goal is to maximize the profit obtained from each of the knapsacks simultaneously while meeting the weight constraints imposed. The MOP formulation follows for m items and n knapsacks , where

$$\begin{aligned} p_{i,j} &= \text{profit of item } j \text{ according to knapsack } i, \\ w_{i,j} &= \text{weight of item } j \text{ according to knapsack } i, \\ c_i &= \text{capacity of knapsack } i \end{aligned}$$

For the MMOKP problem with n knapsacks and m items, the objectives are to maximize

$$f(\mathbf{x}) = (f_1(\mathbf{x}), ...f_n(\mathbf{x})) \tag{6.9}$$

where

$$f_i(\mathbf{x}) = \sum_{j=1}^{m} p_{i,j} x_j \tag{6.10}$$

and where $x_j = 1$ if item j is selected, 0 otherwise [215]. The constraints are:

$$\sum_{j=1}^{m} w_{i,j} x_j \leq c_i \forall\ i \tag{6.11}$$

where $x_j = 1$ if item j is selected, 0 otherwise. The MMOKP has similar characteristics to those of the ALP problem described in the last section. Both the MMOKP and ALP MOPs are formulated with a large number of decision variables, the decision variables are integer based, and the constraints are linear. Since the MMOKP and ALP MOPs have similar characteristics, one may expect similar issues to arise as compared to the ALP testing. In fact, the application of the MOMGA-II to the MMOKP problem illustrates that initially the similar performance was obtained as compared to the results of the ALP MOP. In the initial application of the MOMGA-II to the MMOKP MOP, a constraint handling approach was not used and the MOMGA-II generated few feasible solutions, entirely of inferior quality when compared to the results of other MOEAs. The initial results necessitating the use of a repair mechanism in order to provide the MOMGA-II an increased probability of identifying good BBs in the population.

The constraints on the MMOKP MOP are formulated such that an item placed into or removed from a knapsack must be placed into or removed from all of the knapsacks and the other constraint is that the weight of each knapsack must not exceed the capacity of the knapsack (see Chapter IV, Section 4.4.1). For example, in a 2 knapsack instantiation of the MMOKP MOP, if the capacity constraint of knapsack 1 is exceeded but the capacity constraint of knapsack 2 is met, at least one item must be removed from both knapsacks until the capacity constraint is met for knapsack 1.

An analysis of the initial results of the MOMGA-II illustrate that without the repair method, the infeasible results generated are far away from the feasible solutions in phenotype space. This is a different result than was realized from the application of the MOMGA-II to the ALP MOP. Hence the repair mechanism is also used in all three phases

of execution. The population members are repaired following the random initialization of the population, during the BBF phase and during the juxtapositional phase. Additionally, the competitive templates are repaired prior to use. The population members are not repaired after each specific cut-and-splice operation of the juxtapositional phase in order to avoid convergence to a suboptimal solution and reduce the overhead associated with the numerous recombinations that take place within a single generation. However, at the end of each generation of the juxtapositional phase, all of the population members are repaired. The population is stored to an external archive to ensure that the feasible population members generated by the MOMGA-II are not lost. The next juxtapositional generation begins with feasible population members and the process repeats itself. Once termination criteria for the MOMGA-II is met, the results are presented to the researcher. At the conclusion of execution of the three phases of the MOMGA-II, all of the population members are feasible and the competitive templates are updated for the next BB execution. The repair process ensures that subsequent BB size executions begins with feasible competitive templates.

The initial repair mechanism selected for use is identical to the repair mechanism used in the application of the MOMGA-II to the ALP MOP. To summarize, the repair mechanism randomly chooses a bit position and a direction to scan the bit string. After the choices of position and direction are made, if a constraint is violated, items are removed from the knapsacks until the weight constraint for each knapsack is met. The items are removed one by one as the string is scanned bit by bit until the constraints are met. While the random repair method performed well for the ALP problem, in the application of the MOMGA-II to the MMOKP MOP the performance of the ransom repair is not very good. The results of the random repair are of lesser quality than those obtained from other MOEAs. Other repair mechanisms that incorporate problem domain knowledge must be considered in order to obtain effective results.

Since other researchers have attempted to solve the MMOKP, one must consider whether or not other repair approaches may have merit. Zitzler and Thiele [215] proposed a greedy approach that is an extension of a single objective repair mechanism used by Michalewicz and Arabas [139]. In this multiobjective greedy repair approach, items are

removed from the knapsacks based on their profit to weight (ptw) ratio. Remember that each item has an associated profit and weight with respect to the knapsack the item is placed into. If the knapsack capacity is exceeded, then items of lowest profit to weight ratio with respect to the knapsack constraint that is violated are removed first. The profit to weight ratio is defined in Equation (6.12).

$$ptw_{i,j} = \frac{p_{i,j}}{w_{i,j}} \tag{6.12}$$

Zitzler et al., state that this repair method performs well when used in their MOEA, the SPEA. Initial testing of this method applied to the MOMGA-II was not as successful and hence a different way of repairing the population members became necessary. Since the MOMGA-II's operators are substantially different from the SPEA operators and the process it follows, it is accepted that a method that performs well when implemented in one MOEA may not perform well when implemented in other MOEAs per the NFL Theorem.

Considering that an item placed into one knapsack must be placed into every knapsack in the MMOKP formulation tested, and that each item yields a different profit to weight ratio based on the knapsack it is placed into, there may be a more effective method of implementing the repair. Zitzler's repair mechanism is a simple approach to extending a single objective repair mechanism to a multiobjective repair mechanism. The problem is that Zitzler's repair assumes that all of the constraints are violated, which may not be the case [215].

A potentially better repair mechanism for use in the MOMGA-II removes items from the knapsacks based only on the knapsack constraint that is violated. If only the knapsack 1 constraint is violated, then the ptw ratio of each item based on knapsack 1 is compared. For example, assume that two knapsacks are used and knapsack 1 violates the constraint. It is possible that item 1 has a $ptw_{1,1}$ (item 1 placed into knapsack 1) of 12 but a $ptw_{2,1}$ (item 1 placed into knapsack 2) of 4 and item 2 has a $ptw_{1,2}$ of 6 and a $ptw_{2,2}$ of 9. Using Zitzler's ptw in Equation (6.12), item 1 would be the first to be removed as it has a lower overall ptw ratio but only knapsack 1 violates the constraint. Hence item 2 should be removed first using the proposed repair mechanism as item 2 has the lowest ptw for

knapsack 1. In cases where the number of constraint violations is greater than 1 but less than n, the repair mechanism randomly selects a knapsack that has a constraint violation and repairs the population member with respect to that knapsack. If all knapsacks violate the constraint, then Equation (6.12) is used to repair the knapsacks. The proposed repair method better reflects the multiobjective nature of the MMOKP MOP. However in testing this method in the MOMGA-II, the new proposed repair method tends to obtain slightly better results in some cases than Zitzler's method but overall the performance is similar to that obtained with Zitzler's repair mechanism.

Another researcher recently has proposed a new approach to repairing population members when using the MMOKP formulation presented in this chapter [100]. In Jaszkiewicz's approach, used in the IMMOGLS MOEA, the ptw ratio of an item takes into account the multiple objectives in a better manner than the other mentioned approaches. Only one ptw ratio is calculated per item, but the ptw ratio takes into account all j knapsacks unlike the previously mentioned methods that calculate a ptw ratio per item, per knapsack. Jaszkiewicz's method sums up the profits over all of the knapsacks and sums up the weight over all of the knapsacks for a single item. The ptw ratio is calculated by determining the ration of the sums and is presented in Equation (6.13).

$$ptw_j = \frac{\sum_i p_{i,j}}{\sum_i w_{i,j}} \tag{6.13}$$

Jaszkiewicz's repair method yields the best performance when used in the MOMGA-II. All of the results presented for the MOMGA-II in this chapter use Jaszkiewicz's repair method. The improved performance of Jaszkiewicz's repair method over other repair methods (applied to the MMOKP using the MOMGA-II) is attributed to a better calculation of the lowest ptw ratio that more accurately takes into account all of the knapsacks simultaneously. The MOMGA-II is applied to the 100 item, 2 knapsack; 250 item, 2 knapsack; 500 item, 2 knapsack; and 750 item, 2 knapsack MMOKP presented in Zitzler [215].

Any proposed change to an MOEA should be tested in order to determine if it performs logically correct and to determine if the effect of the change is worthwhile or as anticipated. The effect of the repair method selected (Jaszkiewicz's repair method),

as applied to the initial population of a single run of the MOMGA-II, is illustrated in Figure 6.25. The black (.)s represent the results of applying the repair mechanism to the infeasible population members. Any feasible population members are not repaired and hence the black (.)s only represent the members that are repaired. One can see that the repaired population members have fitness values that are lower than the initial population members since the repair mechanism removes item types from the knapsack if a constraint is violated. Hence the repaired members move towards the lower left portion of the figure as item types are removed from their knapsacks and hence the fitness values of these repaired members are decremented. Such a result is what one would expect occurs after repairing the population and this validates that the repair mechanism performs correctly. The repaired members are all feasible, there are feasible BBs in the population, and the MOMGA-II can proceed to the BBF phase with feasible BBs present in the population. This type of comparison should always be done by researchers attempting to improve the performance of their MOEAs. It is crucial to validate the performance of a repair operator.

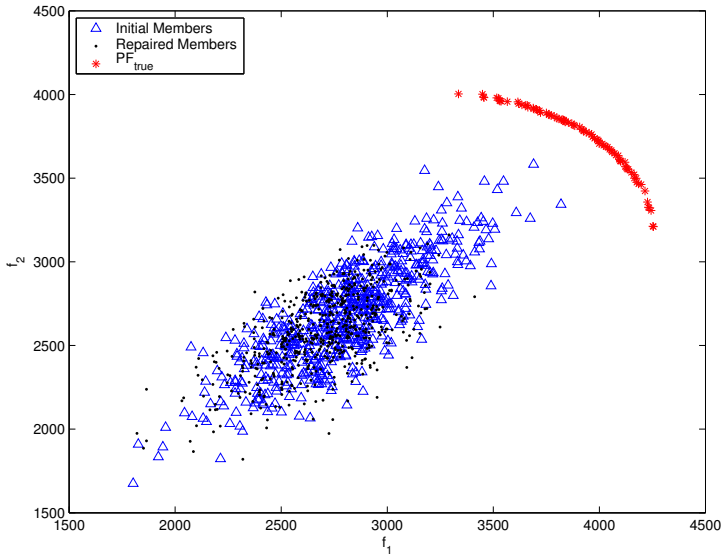

Figure 6.25 Initial Population for 100 Item 2 Knapsack MMOKP

Prior to execution of the MMOKP tests, a modification to the MOMGA-II is completed to increase the selection pressure. The MOMGA-II uses an elitist selection mechanism in which the current Pareto front is passed from one generation to the next each

time that selection occurs. This mechanism is used in the testing presented in this chapter and performs well. The elitist method used maintains $PF_{current}$ in the population from generation to generation but does not guarantee that a member from PF_{known} is not destroyed or removed from the population during a particular generation. Instead, the nondominated members are stored to an external archive each generation and if nondominated at the conclusion of MOEA execution, appear in the final PF_{known} set. Maintaining $PF_{current}$ through the selection mechanism is anticipated to increase the effectiveness of the MOMGA-II as the good BBs present in $PF_{current}$ remain in the population. In order to increase the selection pressure of the MOMGA-II and increase the convergence to a good solution set, this elitist scheme is implemented for the MMOKP. The elitist selection mechanism selects all of the population members that are elements of $PF_{current}$ to be placed in the next generation. If the population is not full, additional members are selected through the use of an elitist based tournament selection mechanism that selects two individuals at random to compete against each other. The tournament selection process repeats until the required population size is achieved.

This new elitist routine is expected to increase the effectiveness of the MOMGA-II. The results of this elitist scheme are presented in Figure 6.26 as (*)s, and the results of the normal tournament selection without elitism scheme is represented by (.)s. In the 100 item, 2 knapsack MMOKP, a slight improvement is noted in the results of this new elitist selection scheme. Since an improvement is realized in the limited testing and anticipated to improve performance, all of the MMOKP tests use this new elitist selection mechanism.

In comparison testing of the MOMGA-II to another MOEA applied to the MMOKP, the SPEA is selected for comparison. The data results of the SPEA are available and hence allow a comparison between the two MOEAs to be conducted. Figure 6.27 presents a graphical representation of the results of the SPEA and the MOMGA-II as applied to the smallest instantiation of the MMOKP. The ($\triangle$)s represent the results (PF_{known} set) of the MOMGA-II and the (.)s represent the results (PF_{known} set) of the SPEA. A small difference exists between the results of the MOMGA-II as compared to the SPEA. The MOMGA-II obtains a limited number of points that dominate those the SPEA generates

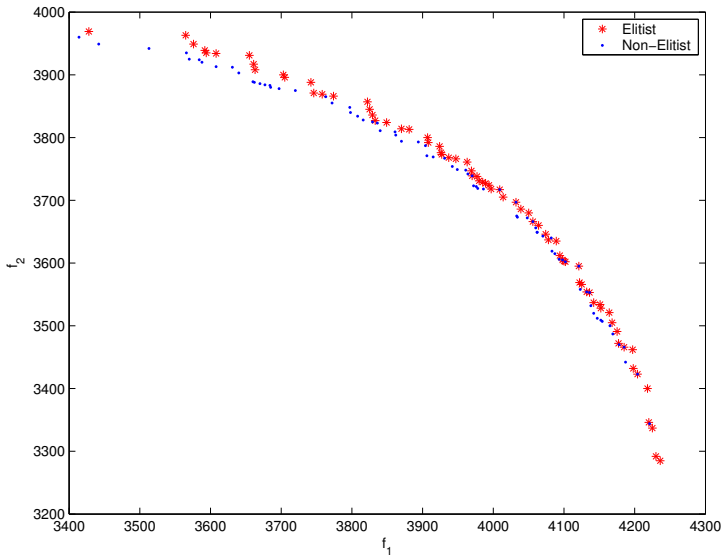

Figure 6.26 Elitist 100 Item 2 Knapsack MMOKP

and vice versa, but overall both algorithms generate numerous identical points. The SPEA appears to obtain a few points that dominate the MOMGA-II results in the upper left end of the Pareto front. Overall the results from both MOEAs are similar with the exception of slightly better performance by the SPEA at the end of the front.

The smallest instantiation of the MMOKP represents a difficult problem for an MOEA, but one in which both MOEAs find good solutions and a good distribution of points across the front. The next instantiation tested is the 250 item, 2 knapsack MMOKP. The MOMGA-II results in better performance than the SPEA across the entire center section of the front, as well as the lower right end of the front, shown in Figure 6.28. However the SPEA obtains points in the upper left end of the front unpopulated by MOMGA-II results. Better performance is realized by the MOMGA-II for most of the Pareto front but the SPEA continues to obtain better solutions in the upper left end.

Increasing the dimensionality of the MMOKP leads to the next instantiation of the MMOKP MOP tested, the 500 item, 2 knapsack formulation. Figure 6.29 shows the results generated by both the MOMGA-II and the SPEA as applied to the 500 item formulation of the MMOKP. The MOMGA-II obtains many more points than the SPEA and the MOMGA-II results dominate those of the SPEA. The results illustrate a considerable

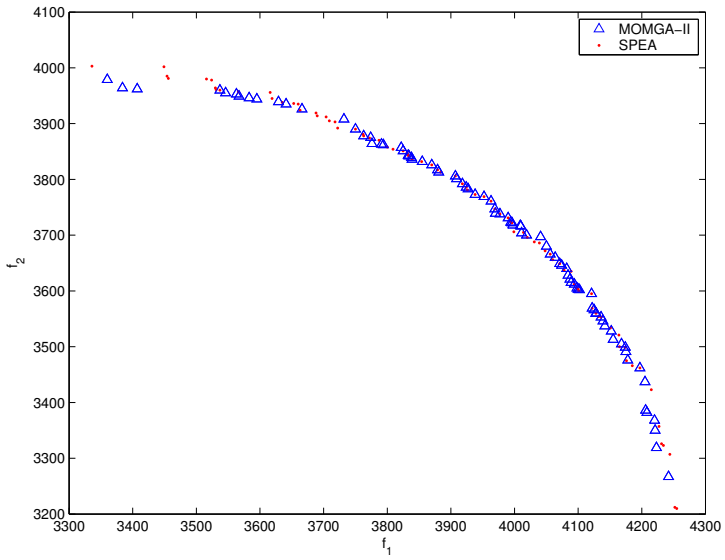

Figure 6.27 100 Item 2 Knapsack MMOKP

distance between the Pareto front generated by the SPEA and the better front generated by the MOMGA-II. All of the points generated by the SPEA with the exception of two are dominated by the MOMGA-II. The MOMGA-II also obtains a good spread of solutions across the front. While the MOMGA-II does not find solutions on one small area of the front, the solutions found by the MOMGA-II dominate those solutions found by the SPEA. This illustrates much better performance by the MOMGA-II on this more difficult MOEA MOP.

The largest instantiation of the 2 knapsack problem is also tested, the 750 item MMOKP. The MOMGA-II results exhibit much better performance across the entire front as compared to the SPEA. Figure 6.30 shows that the results of the MOMGA-II as compared to the SPEA. It is easily seen that the entire Pareto front generated by the MOMGA-II dominates the entire front generated by the SPEA in this 750 item MMOKP MOP. The MOMGA-II does not obtain the same spread of solutions as the SPEA but all of the MOMGA-II solutions are of higher quality (dominate) than those of the SPEA.

Results of calculating metric values for the 100, 250, 500, and 750 item, 2 fitness function MMOKP MOP instantiations are presented in Table 6.8. The ONVG and spacing metrics are used along with the visualizations of the Pareto fronts presented. The selection

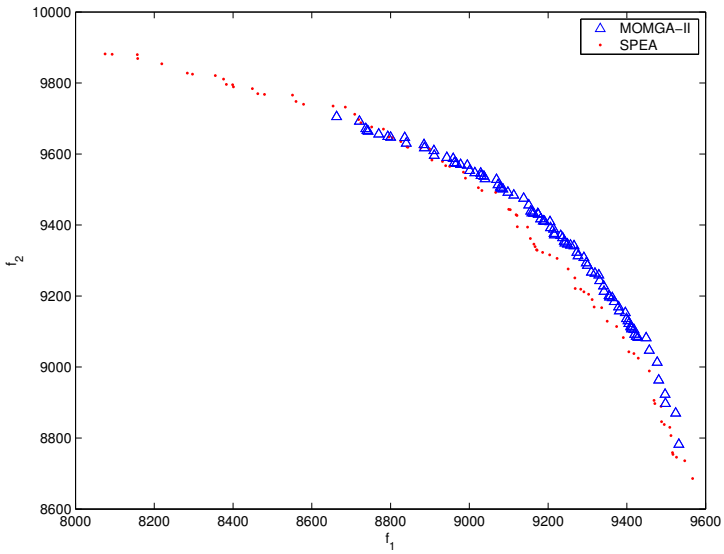

Figure 6.28 250 Item 2 Knapsack MMOKP

Table 6.8 2 Knapsack MMOKP Results

Number of Items	**MOEA**	**ONVG**		**Spacing**	
		Mean	**SD**	**Mean**	**SD**
100	SPEA	49.267	6.291	17.797	3.841
100	MOMGA-II	44.333	8.976	18.331	7.361
250	SPEA	55.567	6.377	26.163	3.956
250	MOMGA-II	41.167	10.986	22.855	9.343
500	SPEA	34.533	5.594	46.798	10.778
500	MOMGA-II	35.733	10.866	24.703	16.146
750	SPEA	34.200	6.408	71.340	20.733
750	MOMGA-II	30.200	12.090	34.096	24.750

of these metrics is discussed in detail in Chapter III, Section 3.2. For each MOEA, the mean and standard deviation results are presented for each metric. The MOMGA-II and SPEA obtain similar performance for the ONVG and spacing metrics as applied to the 100 item knapsack MMOKP. The 100 item knapsack MMOKP is the smallest instantiation of the MMOKP MOP tested and as shown in Figure 6.27, the two MOEAs obtain a similar front.

The results presented in Figure 6.28 illustrate that the MOMGA-II obtains a number of points that dominate those found by the SPEA but the SPEA obtains a better spread of points across the known front. Table 6.8 shows that the SPEA, on average, finds a

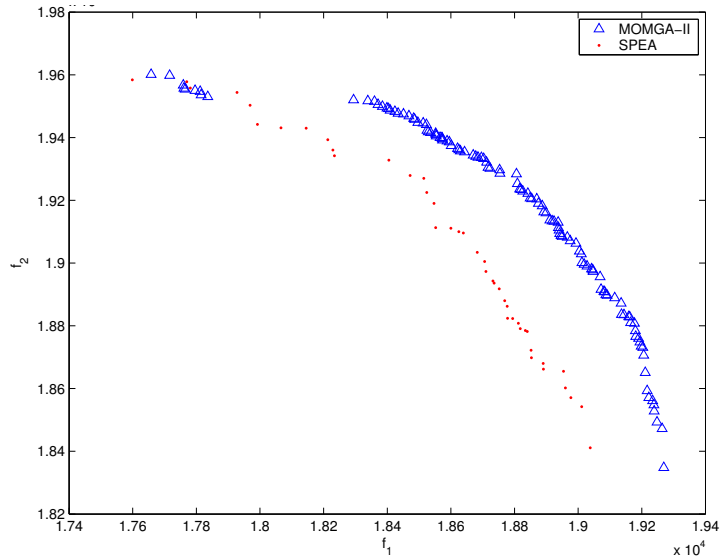

Figure 6.29 500 Item 2 Knapsack MMOKP

larger number of points in PF_{known} and obtains a slightly better spacing value for the 250 item MMOKP MOP. However, the MOMGA-II obtains a much better distribution of points and points of higher quality over most of the front, with the exception of the upper left section shown in Figure 6.28. Overall, the MOMGA-II generates mostly points of equivalent or better quality but the SPEA obtains a better spread of points. Since the quality of the results is typically a driving factor for use of an MOEA, one would deem both the MOMGA-II and the SPEA as performing well on this MOP instantiation.

The 500 item MMOKP MOP is a more challenging instantiation of the MMOKP MOP and the formulation specifies a large number of decision variables. Figure 6.29 illustrates that the solutions found by the MOMGA-II dominate all but two of the solutions found by the SPEA. Additionally, Table 6.8 illustrates that the average number of solutions and the spacing values generated by the results of both MOEAs are comparable and but the SPEA metric results tend to be slightly better than the MOMGA-II. However, the MOMGA-II obtains a good distribution of points across most of the front and the MOMGA-II points dominate those of the SPEA. Overall the MOMGA-II obtains better results on the 500 item MMOKP.

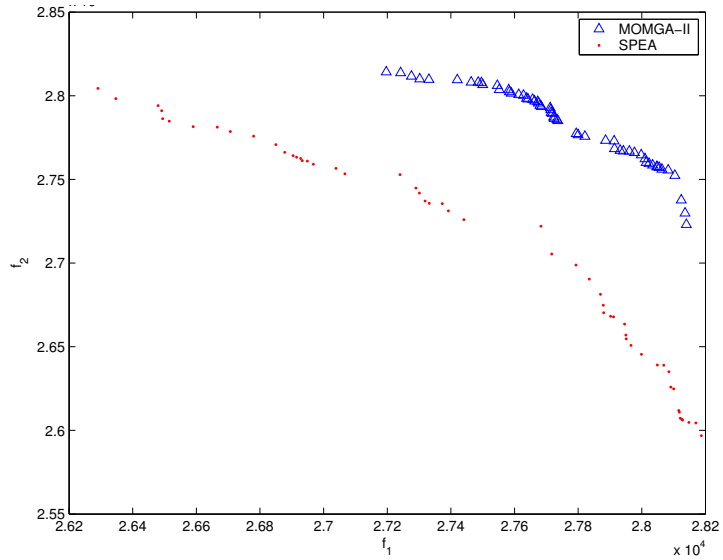

Figure 6.30 750 Item 2 Knapsack MMOKP

The results presented in Table 6.8 show that the MOMGA-II obtains a similar number of vectors on average as the SPEA but obtains a much better spacing value than the SPEA for the 750 item MMOKP MOP. The results are shown graphically in Figure 6.30 which also illustrates that the results of the MOMGA-II dominate all of the points found by the SPEA. Overall the MOMGA-II obtains better performance than the SPEA on this instantiation of the MMOKP MOP. The results presented illustrate a trend that as the number of decision variables increase, the improvement in performance of the MOMGA-II as compared to the SPEA increases, and the MOMGA-II typically generates solutions of higher quality.

Jaszkiewicz proposes a different metric for comparing the results of different MOEAs. His method involves calculating the average range of fitness values of each Pareto front curve or surface and then he compares the average values among the MOEAs [100]. To determine these averages one must find the minimum value generated for each fitness function in PF_{known} for each data run. The minimum values for each fitness function are then averaged across the number of data runs conducted. The same process is used to calculate the average maximum values for each fitness function. For example, in a 2 fitness function MOP, in which 10 data runs were conducted, one must average the 10 minimum values for function 1 and function 2 separately. The same procedure is repeated

for the maximum values. The minimum and maximum values are placed into a table for comparison with other MOEAs.

The results of the IMMOGLS, SPEA, M-PAES, and MOGLS [100] are presented in Table 6.9 and compared to the MOMGA-II. Jaszkiewicz uses this average range metric to state how well the MOEA solutions are spread out over PF_{known}. However, the data presentation format he used does not accurately reflect the spread or concentration of points. In most attempts to solve an MOP by an MOEA, researchers conduct numerous data runs and then combine the PF_{known} sets generated for each run. This combination of the data and subsequent analysis yields the overall PF_{known} solution set generated by an MOEA over a course of data runs. While one run may generate solutions in the lower right portion of the front (consider the characteristic of the MMOKP fronts as an example, Figure 6.30), another may generate solutions exclusively in the upper left portion of the front and the remaining runs may only generate solutions in the center of the front. A researcher solving real-world MOPs is interested in the final overall result and not necessarily in the averages. The averages can be deceiving. In the previous example, most of the data runs generated solutions in the center of the front. Calculating the maximum and minimum average values using the data runs containing results on the endpoints of the Pareto front and combining the results with the two runs that generated solutions in a different area of the front may yield a value closer to the center than the end of the Pareto front. An analysis would then lead one to believe that the MOEA did not generate a good spread of solutions but instead a cluster around the center of the front. In comparison testing of an MOEA that consistently generated solutions only at the two ends of the front, the MOEA would be shown as generating a poor distribution of points across the front. However, the a different MOEA may not have generated any solutions in the center portion of the front and therefore does not obtain as good a spread of solutions but the average range table would show otherwise.

While the average range data presentation format may be useful, it must be used in conjunction with other metrics or with a graphical presentation of the Pareto front in order to avoid misinterpreting the results. As stated earlier, in some cases, a visual representation may be better than the results of a specific metric as metrics are lossy

evaluations and loose information in mapping a Pareto front of multiple data points to a single value. However, Table 6.9 can be useful if used in conjunction with the graphical representation of PF_{known} presented in Figures 6.27 through 6.30.

It is important to realize that the MOGLS MOEA is executed on the relaxed formulation of the MMOKP MOP (see Chapter III, Section 3.1.1.16) and not on the formulation of the MMOKP that the MOMGA-II and the SPEA use. Due to this fact, one must question if Jaszkiewicz's comparison of the MOGLS to IMMOGLS, SPEA, and M-PAES is valid. In Table 6.9 the results of MOGLS are included, but a direct comparison between the MOMGA-II and the SPEA is the main focus. A comparison is made to the SPEA as it achieves the best published performance when applied to the identical formulation of the MMOKP MOP that the MOMGA-II is applied to.

The results presented in Table 6.9 indicate that, for the most part, the MOMGA-II does not generate as effective results as the the other MOEAs when comparing the spread of solutions. This interpretation of the results is not necessarily correct. For example, consider the 2 knapsack, 750 item instantiation of the MMOKP. Table 6.9 indicates that the SPEA has a wider range of values and leads to better solutions but in actuality, Figure 6.30 illustrates that the front generated by the MOMGA-II dominates the entire front generate by the SPEA. Therefore this presentation format is not recommended for use when one Pareto front completely dominates another. The average range metric can mislead a researcher's analysis of the data if one is not careful to see the whole picture.

6.5.3.3 Real-World MOP Test Suite Summary. Testing every class of real-world MOP is not possible. This section presents the results of applying the MOMGA-II to a real-world application MOP, the ALP and a NP-Complete MOP containing an extremely large number of decision variables, the MMOKP MOP. The results of the MOMGA-II are very good when applied to these real-world application MOPs. The MOMGA-II finds PF_{true} for the 60 bit formulation of the ALP and appears to find good solutions for the larger formulation. The MOMGA-II also performs favorably when applied to the MMOKP and compared to other MOEAs. In particular, as the problem size increases, so does the effectiveness of the MOMGA-II in terms of dominating the solutions found by other

MOEAs. Additionally, detailed descriptions of possible repair mechanisms and the best found MOMGA-II repair mechanism to use with MOPs formulated with integer based decision variables is presented. The effect of a limited test of repair mechanisms on the effectiveness of the MOMGA-II as applied to two MOPs is also presented. The MOMGA-II performs well when using repair mechanisms to attempt to solve these discrete constrained MOPs.

Table 6.9: Knapsack Problem Results

Instance	**IMMOGLS**	**SPEA**	**M-PAES**	**MOMGA-II**	**MOGLS**
2-250	[8520.15, 9537.85]	[8407.63, 9460.87]	[8742.50, 9473.05]	[8876.47, 9405.57]	[7332.55, 9883.90]
	[8614.15, 9629.90]	[8809.47, 9747.23]	[8866.95, 9593.00]	[8956.97, 9546.53]	[7747.65, 10093.2]
2-500	[17684.70, 19047.50]	[17697.50, 18900.40]	[18198.50, 19174.70]	[18387.17, 18906.07]	[16148.60, 20029.20]
	[18114.70, 19459.30]	[18455.20, 19460.70]	[18541.80, 19514.00]	[18830.70, 19311.60]	[16766.30, 20444.20]
2-750	[25902.60, 27868.50]	[26374.90, 27924.00]	[27100.30, 28661.80]	[27001.57, 27506.87]	[23728.90, 29938.80]
	[25738.10, 27904.30]	[26152.20, 27720.10]	[26460.30, 28255.90]	[27110.57, 27580.47]	[23458.20, 29883.10]

6.6 MOEA BB Testing Based on Empirical Data

The background information discussed in Chapter II, Section 2.2 presents a mathematical definition of a BB. Identification and manipulation of good BBs is important to generating good solutions to MOPs by any MOEA. The good results presented in this chapter validate this statement as an explicit BB-based MOEA, the MOMGA-II generates good results as applied to a variety of MOPs.

Since one seeks to find the best solution in any optimization problem, identification of the good BB(s) is critical to generating good and hopefully the best solution. In the search for the optimal solution, it is possible that the identification of only one good BB is necessary to generate all of the good solutions on the Pareto front or, the more likely case, that there exists multiple BBs that must be identified in order to generate the multiple solutions on the Pareto front. If the identification of more than one good BB is necessary to find the entire front, then the MOEA must find the multiple good BBs.

Van Veldhuizen [184] illustrated the point that BBs relatively close together in terms of their genotype space may not be as close together in phenotype space. The use of the MOMGA-II as a BB validator is a high priority. This enables a more thorough analysis of various MOEAs to determine why certain MOEAs perform better than others in reference to BBs found. This analysis requires the execution of the MOMGA-II and other MOEAs on an identical set of test suite MOPs. An analysis of each generation of execution of each MOEA is necessary to determine which BBs are found by the various MOEAs and for how long do the good BBs remain in the population. A BB analysis provides additional insight into the inner workings of any MOEA.

The size of the BBs that are necessary to find points on the Pareto front has yet to be addressed in the literature. Assuming a worst case situation that multiple BBs are contained within each point in PF_{true} , the possibility also exists for the good BBs necessary to generate the points in PF_{true} to be of varying sizes. In MOPs, a BB of size 5 may generate the single (optimal) solution to the problem plus a few near-optimal points that are relatively close to the solution in phenotype space. However, a BB of size 3 may generate a single or a few near-optimal points that are close (in phenotype space) but are

not the solution to the optimization problem. Generalizing this discussion, the analysis of BBs and BB relationships illustrates that a BB of size x may generate a single or a few solutions in PF_{true} , and another BB of size y may generate a single or a few different solutions in PF_{true} , but a BB of size z may only generate a single or a few solutions that are sub-optimal to those contained in PF_{true} . Ranking is used to clarify this discussion.

In analyzing the effects of BBs of varying sizes on the final solution set, the concept of ranked Pareto fronts must be understood. Ranking is discussed in detail in Chapter II, Section 2.9 and is summarized in this section. Ranking is the process that identifies the population members that are on the Pareto front, assigns the members a rank value, and removes the members that are on the Pareto front from the population. The ranking process continues to identify the next Pareto front from the population members that remain in the population, assigns the members on the next front a different rank value and repeats the procedure until all members are ranked. In the assignment of ranks, there are multiple methods that can be used. In this section, the members that exist on the best found front are assigned a rank of 0, the members that are part of the second best front are assigned a rank of 1, and so on until all of the population members are ranked. An example of ranked fronts is illustrated in Figure 6.31.

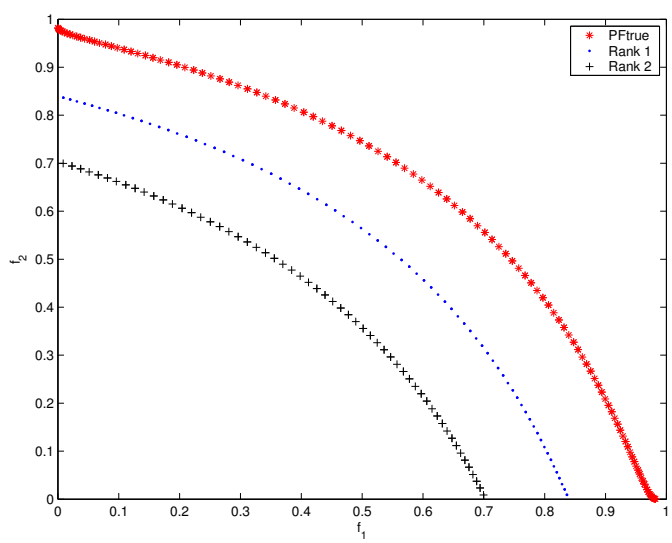

Figure 6.31 Ranked Pareto Fronts

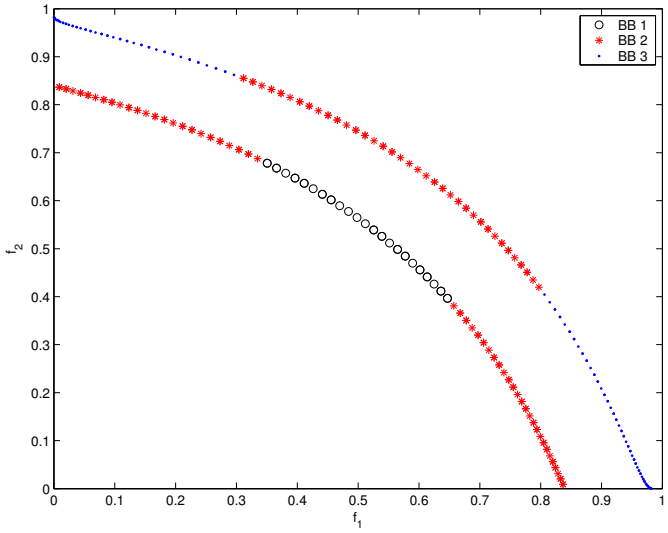

Figure 6.32 BB Sizes and Their Contribution to the Pareto Fronts

If the case exists where multiple BB sizes are required to generate all the solutions on PF_{true}, some of the same BBs may also generate solutions on the next best front. Figure 6.32 illustrates an example where BBs of multiple sizes are required to generate the front of rank 0. The example presented in Figure 6.32 is a specific case, but similar BB characteristics can be found in other MOPs.

In general, the identification of multiple BBs is necessary to generate PF_{true} for many MOPs as multiple solutions exist in PF_{true}. Since multiple solutions existing in PF_{true}, and multiple BBs are typically necessary to generate these solutions, there is a high probability that multiple BB sizes are also required to generate all of the solutions in PF_{true}. The identification of multiple BB sizes by an MOEA results in the generation of multiple fronts of different ranks through the search process.

As good BBs are identified and recombined by the MOMGA-II, solutions on inferior fronts (fronts of rank 1, 2, etc.) are generated as the population progresses towards PF_{true}. Once all of the necessary good BBs are generated, assuming a large enough population size to combat the noise present in the evolutionary process, an MOEA generates all of the points in PF_{true} plus portions of the inferior fronts. Some researchers have identified in explicit statements or through the results they have presented a difficulty for MOEAs in the generating points at the extremes of the front [100, 123, 215]. The extremes are

referred to as the endpoints of the curve or k-dimension surface as dictated by the k objective functions. The difficulty of generating the extreme points of the Pareto front is attributed to the necessary identification of multiple BBs of different sizes. Implicit BB-based MOEAs may only generate BBs of a single size or may not be executed with a population size large enough to statistically generate the multiple good BBs of various sizes necessary to generate PF_{true}. Figure 6.32 illustrates the effect that different BB sizes may have in finding various points in the ranked fronts and in PF_{true}. Notice that in Figure 6.32, only two of the ranked fronts are shown for ease of understanding but multiple fronts may exist, each of possibly varying cardinalities.

Since many MOEA researchers conduct their research efforts with implicit BB-based MOEAs (as described in Chapter III), BB concepts and the effects of the identification of good BBs are not readily noticeable. Through research conducted using the MOMGA-II and the theoretical development of population sizing equations (discussed in Chapter VIII) based on the BBH, the need for different sized BBs to generate PF_{true} becomes apparent.

Many of the existing MOEAs are not effective at finding all of the points on the Pareto front and more explicitly, points at the endpoints or end sections of the Pareto front when applied to test suite and real-world MOPs [100, 123, 215]. While generating any point(s) on the Pareto front may be useful for real-world applications in which potential solutions have not been found, it would be even more useful if a researcher could generate a good distribution of points across the entire front. This has been identified by researchers utilizing MOEAs as an important issue [31, 44, 100, 123].

A question that the MOEA community should answer is: *Why do various MOEAs fail to find the endpoints of the Pareto front or if they do find some of the points, why does this typically occur with larger population sizes?* A pedagogical example is constructed and the results of the MMOKP MOP are analyzed to show insight into the generation of the endpoints of the Pareto front by an MOEA. The pedagogical MOP is formulated utilizing a deceptive objective function along with another function that is not deceptive. Deceptive problems are discussed in Chapter IV, Section 4.3. The equations for this deceptive maximization MOP are presented as Equations (6.14) and (6.15), where x is

restricted to be an integer value.

$$f_1 = \begin{cases} 0.75 - \frac{3}{48}x & 0 \leq x \leq 12 \\ 1 - \frac{1}{3}(x \; mod \; 12) & 12 < x < 15 \\ 1 & x = 15 \end{cases} \tag{6.14}$$

where x is an integer.

$$f_2 = \left\{ 1 - \frac{1}{15}x \quad 0 \leq x \leq 15 \right. \tag{6.15}$$

where x is an integer.

Graphically, the two objective functions are shown in Figure 6.33 with each function plotted as a single objective problem for ease of illustrating the deceptive function. Function 1 is the deceptive function, and function 2 is the non-deceptive function. In function 1, an x-value of 0 contains the deceptive BB that leads to a locally optimal value of 0.75; however, an x-value of 15 contains the BB that leads to the global optima of 1.0. The deceptive BB is of order 1 and is represented by the schema *0**. The schema *0** leads to solutions on the left side of Figure 6.33, solutions that lead to the locally optimal solution of 0.75, whereas the BB necessary to find the global optima is of order 2 and is the schema 11**. The schema 11** generates solutions on the right side of Figure 6.33 that lead to the globally optimal solution of 1.0. Since function two is non-deceptive, a BB of order 1, containing the schema *0** leads to the generation of the globally optimal solution of 1.0.

Van Veldhuizen [184] presented a theorem and proof of deceptive BBs in MOPs. The interested reader is referred to [184] for the proof of this theorem.

Theorem 1 *The orders of deception for the functions composing an MOP are not necessarily equal [184].*

When using an explicit BB-based MOEA, the implications of this theorem are that one must use a BB of the same order as the largest order BB required to solve each of the functions in the MOP. For example, in the deceptive MOP presented, a BB of order 1 is required to generate the global solution for function 1 but a BB of order 2 is required

to generate the global solution for function 2. Therefore a BB of order 2 is required to generate solutions on the Pareto front for this deceptive MOP.

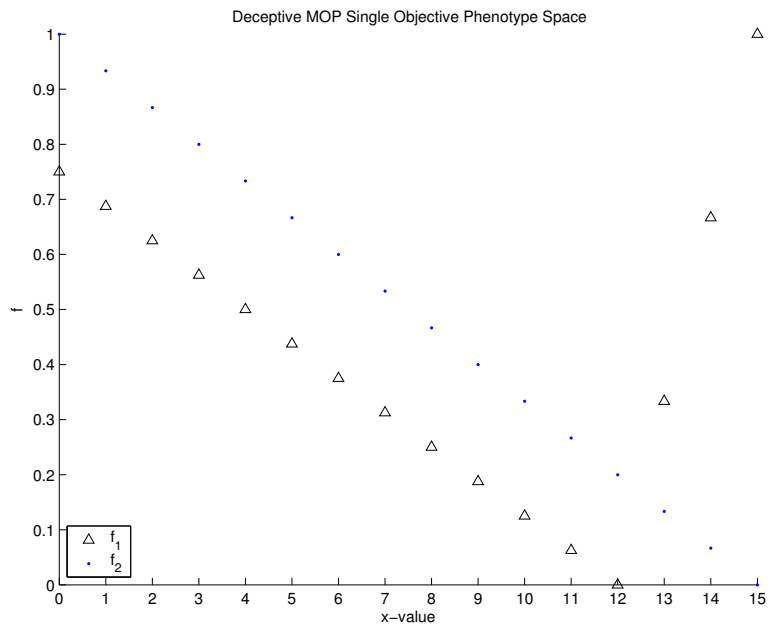

Figure 6.33 Single Objective Values for Deceptive MOP

The genotype space is shown in Figure 6.34 and presents the x values versus their evaluated fitness values. Figure 6.34 illustrates the fact that the true Pareto optimal solutions are the same ones that lead to the global and local optima in this MOP. The Pareto front is shown in Figure 6.35. Notice that there are only two points on the Pareto front of this deceptive MOP. The Pareto front consists of the points at (1.0, 0) and (0.75, 1.0) reflecting the deceptive nature of this MOP. Observe that is is easier to move via an MOEA to the point (0.75, 1.0) than (1.0, 0.0) because of the relatively larger number of points existing between solutions leading to (0.75, 1.0). Also note that there is a larger overall distance between solutions that lead to the point (1.0, 0.0) versus (0.75, 1.0) in the phenotype space. This makes it somewhat harder for an MOEA to find the solution (1.0, 0.0) without identifying the BB 11**.

Since the BB size necessary to solve function 1 is 1 and the BB size to solve function 2 is 2, an overall BB size of 2 is necessary to find all of the points on PF_{true} based on

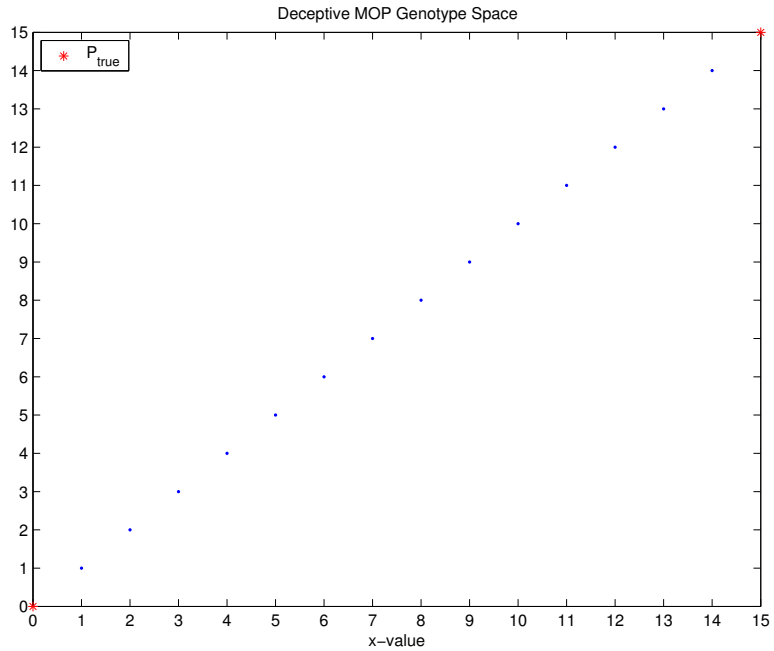

Figure 6.34 Pareto Optimal Solution Paths for Deceptive MOP

Theorem 1. A conjecture is presented as to the relationship of the size of the BB and the location of the points generated when using a BB of this size.

Conjecture 1: A small BB size tends to generate a limited estimate of the structure of the Pareto front for any MOP. A smaller BB size tends to result in a concentration of points neat the center of the Pareto front. To generate additional points on the front, especially near the end sections (endpoints) of the Pareto front, larger BB sizes should be used. The larger BB sizes tend to allow the MOEA to generate points near these sections of the front. The MOEA population sizing Equation (8.26) can thus be employed based upon BB size. □

In analyzing Figure 6.35, there is a larger distance between the points leading to the Pareto front point (1.0, 0.0). This analysis is presented in the following conjecture:

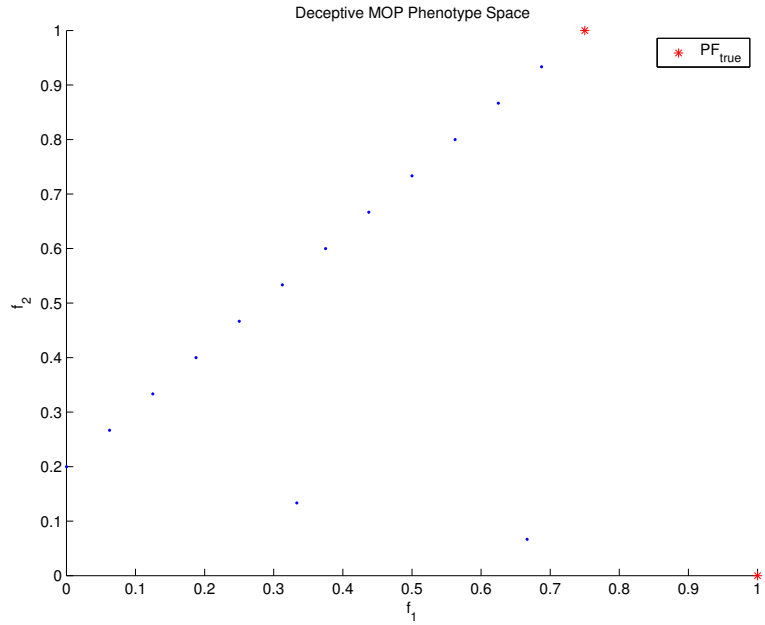

Figure 6.35 Pareto Front Paths for Deceptive MOP

Conjecture 2: The use of larger BB sizes tends to generate points on the Pareto front that exhibit a larger distance from other points in the phenotype space. □

To further explain Conjecture 2, a larger BB size tends to generate points on the Pareto front in areas where there exists a larger distance between points in the phenotype space. Therefore the larger the BB size is in an area of the search space, the further the distance between generational steps from a nondominated point to another; i.e., the more difficult it is to reach points on the Pareto front in that part of the search space.

An MOEA thus tends to move the toward the Pareto front on the "easier" (smaller steps or small BBs) path as generated by the MOEA operators. Of course, there is a lesser probability but not zero that the path to the point (0.0,1.0) is traversed. Therefore this supports Conjectures 1 and 2 in that the use of a larger BB size tends to generate points near the endpoints of the Pareto front. The pedagogical deceptive example presents insight into MOEA search movement to the Pareto front and can be formalized and analyzed based upon BB size requirements per search space region. This provides insight into the largest BB size required to find the solution to a deceptive problem.

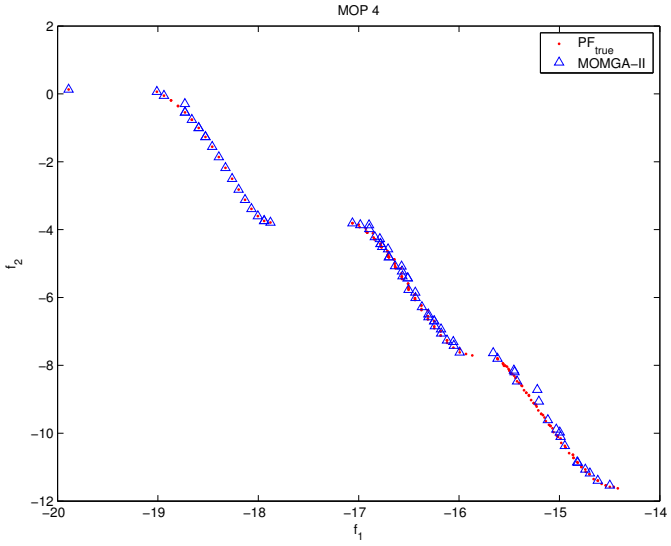

Figure 6.36 MOP4 Pareto Front Generation

Numerous MOPs have been experimented with, using the MOMGA-II and various BB sizes, to determine the applicability of BBs and the conjectures discussed. In Figure 6.36, the results of applying the MOMGA-II with various BB sizes applied to MOP4 is presented. Figure 6.36 illustrates that the majority of points generated from the use of BB sizes 1-3 in the MOMGA-II are on the center sections of the three disconnected fronts. Using a BB size greater than 3 generates more endpoints on the Pareto front, especially those on the lower right portion, the most difficult area to generate points. Again, this is due to larger distances between nondominated points in the search spaces requiring larger BB sizes. A population size larger than the 200 members obtains additional points on the front.

The MOMGA-II is also applied to the MMOKP and generates similar results. In Figure 6.37, the results of the application of the MOMGA-II to the MMOKP MOP using BB sizes 3, 4, and 12 is plotted. All of the BB sizes generate points in the center of the Pareto fronts but as the size of the BBs are increased, the quality of the points found also increases. Additional points, of higher quality, are generated, and the number of solutions generated near the endpoints of the Pareto front increases as the BB size increases. These results again illustrate the fact that the use of larger BB sizes tends to generate points near the endpoints of the front in this MOP. Executing the MOMGA-II with a BB size of 12

generates the best results over the various BB sizes tested (BB sizes 1 through 12). Again, the BB phenomena is due to larger distances between nondominated points in the search space in which the use of a larger BB size tends to overcome this characteristic and the MOMGA-II generates good results. For standard MOEAs, the use of larger population sizes beyond the current 200 may generate additional points near the endpoints of the Pareto front [31, 44]. A similar result is illustrated by Jaszkiewicz [100] in a relaxed version of the MMOKP.

In testing the MOMGA-II against various generic MOPs, it is noticed that as the BB size is increased in the MOMGA-II, additional points are generated near the endpoints of the Pareto front. This parallels the effect seen in implicit BB-based MOEAs where an increase in the overall population size increases the number of endpoints found. Jaszkiewicz [100] found similar results in the application of MOGLS to the multiobjective knapsack problem. In a comparison of SPEA, M-PAES, IMMOGLS and MOGLS (the MOEA developed in Jaszkiewicz's work), the SPEA and IMMOGLS did not generate any points that are as close to the Pareto front as MOGLS, and the points generated are concentrated in the center of the Pareto front. M-PAES found a few points in the center of the front and MOGLS found a number of points distributed across much of the front. However, all of the algorithms had difficulty in generating points near the endpoints of the front and typically converged to points in the center of the front.

6.7 Summary

Throughout this chapter, the MOMGA-II has been shown to yield efficient and effective results when applied to MOPs of varying complexity. The term efficiency of the MOMGA-II is used to describe the resource requirements and execution time of the MOMGA-II as compared to the execution time of the MOMGA. In terms of the memory requirements, the MOMGA-II requires less memory and resources than the MOMGA. The decrease in execution time of the MOMGA-II as compared to the MOMGA is an order of magnitude or more, hence resulting in a more efficient, explicit BB-based MOEA. The MOMGA-II performs statistically similar or statistically better when compared to other

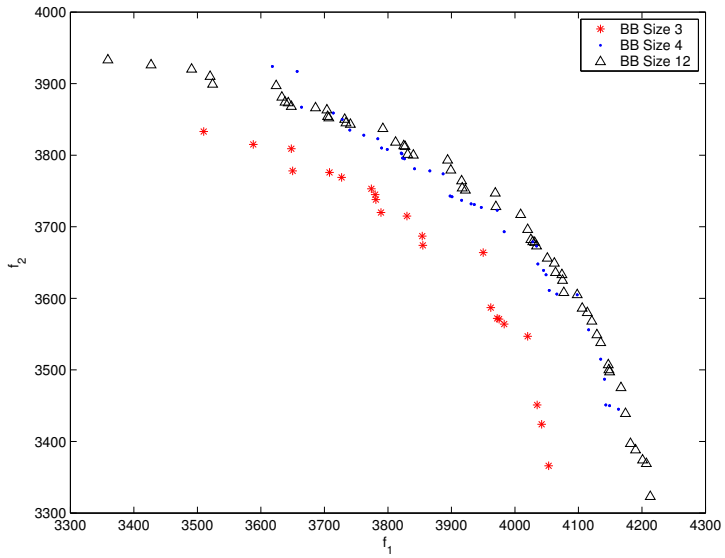

Figure 6.37 100 Item MMOKP BB Analysis

popular MOEA approaches and hence validates the use of this explicit BB-based MOEA. These results are presented in Sections 6.5.1.1 through 6.5.1.7.

Through testing classes of MOPs with similar characteristics to real-world MOPs, the ALP and the MMOKP MOPs (Sections 6.5.3.1 through 6.5.3.2, the results of the MOMGA-II illustrate that problem domain knowledge is integral, in some cases, to generating good solutions. The MOMGA-II is also the first explicit BB-based MOEA designed to attempt and solve real-world and constrained MOPs. In these problems, knowledge is incorporated in terms of a repair function that maps infeasible solutions to feasible solutions and hence generates the good BBs necessary to generate good solutions. In expansion of the testing, the MOMGA-II has the ability to effectively solve constrained real-world, Air Force MOPs of high complexity. Moreover, the various MOP examples indicate that MOEA movement towards the Pareto front is based upon BB size and hence explicit BB-based MOEAs can yield better results than implicit approaches. The MOMGA-II provides additional insight into difficult MOPs through the explicit manipulation of BBs. The MOMGA-II provides insight that other implicit BB-based MOEAs do not, that the use of larger BB sizes and multiple BB sizes tends to generate more solutions near the endpoints of the Pareto front. The identification of BB sizes as a factor in the generation of good potential solutions

to MOPs is a new contribution to the MOEA field, and one not previously realized from implicit BB-based MOEA research.

VII. Parallel MOEA Development

7.1 Introduction

The previous chapters of this effort present MOEA background material, the motivation for using MOEAs and the development of an explicit BB-based MOEA. The use of an MOEA may require the evaluation of hundreds of thousands of population members, if not more, in a single run. Once convinced of an MOEA's effectiveness, researchers typically attempt to increase MOEA execution efficiency (how "quickly" or "cheaply" an MOEA generates good potential solutions to an MOP and with what level of resource requirement) of the MOEA. The desire to reduce execution time, resource expenditure, or increase the quality of solutions generated leads to the consideration of parallel and distributed processing. The objective of this chapter is to describe parallel concepts for MOEAs, present innovative migration and replacement schemes for use in parallel MOEA paradigms, and develop the first explicit BB-based parallel MOEA.

A major computational bottleneck in many contemporary applications of MOEAs to real-world MOPs is the calculation of complex nonlinear MOP function evaluations. Just as in single objective optimization problems, time consuming fitness function calculations are often executed in less wall-clock time by decomposing the computational workload across several processors. The fitness function bottleneck implies that parallelizing an MOEA, to create a parallel MOEA (pMOEA), may improve computational efficiency in terms of execution time. A pMOEA may be able to evaluate additional solutions, as compared to a serial MOEA applied to the same MOP, within the same or reduced execution time. This may be of particular interest to MOEA researchers as identifying a (possibly large) set of "good" objective vectors (genotype) is often the primary goal driving the MOEA search. A pMOEA might then be the preferred implementation for solving complex real-world applications where (multiple) objective function evaluations generate a computational bottleneck.

It is somewhat surprising that additional research into pMOEAs has not yet been completed. This is surprising as MOEAs are inherently parallel in their independent manipulation of a population of individuals. Additionally, parallel and distributed machines

with multiple processors are more readily available to the general public as well as to researchers, than in the past. While advances are made in parallel processing and computers in general, only a small percentage of current MOEA publications either explicitly address parallelism issues or attempt pMOEA execution [31]. Of the known publications solving engineering design and numerical optimization problems with pMOEAs, few discuss algorithmic development issues and for the most part ignore the parallel aspects of their MOEAs [31:p. 314]. In general, these publications do not discuss the rationale for pMOEA employment, algorithmic settings and structure, test problem selection, metrics, or issues of importance to a pMOEA that makes it different from a parallel Evolutionary Algorithm (pEA). However, analyzing these publications yield several key insights into current pMOEA state-of-the-art and highlights areas in which useful research may be focused. The most meaningful findings from analyzing current pMOEA publications are that the following are given little to no attention:

- General pMOEA structure and paradigm suitability for a given MOP domain,
- pMOEA computational hardware architecture considerations,
- MOP test problem selection and pMOEA performance metrics, and
- pMOEA development and implementation issues.

The objective of this chapter is to describe important pMOEA development issues, propose options for satisfactorily resolving them, present a detailed background discussion of the three main parallel MOEA paradigms, present migration and replacement options for use with parallel paradigms in MOEAs, discuss various practical considerations, and present results from the application of a pMOEA, the parallel MOMGA-II to MOPs. Known research approaches and various insights gained from analyzing these approaches are discussed. Through this discussion a template is evolved for generating a new pMOEA or extending an existing MOEA to become a pMOEA. This results in a generic design plan with a list of parameter considerations researchers should consider in attempting to develop efficient and effective pMOEAs, regardless of the problem domain. The term efficiency is used in the context of the use of system resources to include processor utilization, memory requirements, network bandwidth and other resources as well as the wall-clock execution

time of the pMOEA. Effectiveness is used in the context of the quality of the solution set generated by the pMOEA (discussed in Chapter I).

The remainder of this chapter is organized as follows. First, fundamental background information and basic underlying pMOEA philosophy are briefly presented, to include basic pMOEA objectives, paradigms and design issues. Key analyses and issues derived/extended from known pMOEA implementations are discussed next. General pMOEA development issues, along with identification of specific design considerations when implementing the pMOEA paradigms follows. This pMOEA testing includes the use of a pedagogical and NP-Complete, discrete, constrained MOP for demonstrating, testing and analyzing selected pMOEA concepts. As much material is available describing generic parallel processing techniques & implementations, incorporation into EAs and hardware/software configuration issues [2, 3, 4, 5, 6, 7, 13, 20, 31, 119, 179], this chapter focuses only on exploring and analyzing possible benefits of pMOEA development and instantiation. The development of a "generic" pMOEA formulation is detailed along with the advantages of such an algorithm. Experimental results are presented over a limited MOP test suite.

7.2 Fundamental Background

This section addresses pMOEA concepts and background material essential to provide a better understanding and appreciation of the associated concepts. A new pMOEA notation is presented to aid the reader in developing an understanding and relationship between the MOEA and pMOEA concepts presented in this research effort. An overview of the main parallel MOEA paradigms is followed by design considerations.

7.2.1 pMOEA Notation. A clear and consistent notation must be used when discussing MOEAs and pMOEAs. As no clear notation exists for discussing the solution sets that an pMOEA may generate, such a notation is required to ensure that researchers can understand and compare various MOEA and pMOEA approaches. This notation aids in understanding the process that takes place in generating possible solutions to an MOP. Chapter II, Section 2.6, discusses such a notation that has been developed for use with

MOEAs. Considering that pMOEAs may have increased complexity in the algorithmic process or the handling of the solution sets, a precise notation is used to explicitly identify the various Pareto solution sets and the time at which the Pareto solution sets exist. Time is defined to be any stage within the algorithmic process, a generation in the execution of the pMOEA, or more specifically, a particular stage of execution within a specified generation.

A brief summary of serial MOEA notation (from Chapter II, Section 2.6) is provided as necessary background for the pMOEA notation developed in this section. At any given generation, t, the current Pareto front set, $PF_{current}(t)$ contains a subset of the population. The $PF_{current}(t)$ set may vary in cardinality from generation to generation as newly generated, nondominated population members are placed into $PF_{current}(t)$ and dominated members are removed from $PF_{current}(t)$. Since $PF_{current}(t)$ is not a cumulative set across the entire MOEA execution, but just the nondominated solutions from the current generation, the cardinality of $PF_{current}(t)$ is less than or equal to the size of the population in generation t of the pMOEA execution. The cumulative Pareto front set, $PF_{known}(t)$, represents the nondominated members generated from the first generation through generation t of MOEA execution. Note that $PF_{current}(t)$ typically differs from $PF_{known}(t)$. As vectors from $PF_{current}(t)$ are collected each generation, some of them dominate existing vectors in $PF_{known}(t)$ and hence the perception of the Pareto front changes during MOEA execution. PF_{known} represents the nondominated members generated across the entire MOEA execution. Typically an archive is used to aid in the process of identifying and storing the nondominated members found each generation. The goal of the search process is for the MOEA to find the entire true solution set (PF_{true}). Typically this set is unknown for the specific MOP being considered. PF_{true} is implicitly defined by the functions composing the MOP model and the decision variable resolution; it is fixed and does not change.

Certain parallel paradigms, such as the island and diffusion paradigm, execute communication events at different points throughout pMOEA execution. These paradigms use multiple processors in an attempt to generate more effective solutions to an MOP as compared to the serial MOEA. In such paradigms, each processor may find unique solutions to the MOP and hence the notation used must allow for representation of the Pareto sets

resident on each processor. Thus, the current Pareto front set found by a given processor is represented by $PF_{current}\binom{p}{t}$ and a current Pareto optimal set exists and is defined as $P_{current}\binom{p}{t}$ where p represents the processor ID and t the generation number. Corresponding Pareto optimal solution sets exist for each of the Pareto front sets defined but are not explicitly stated. The set containing the nondominated points found through generation t is represented by $PF_{known}\binom{p}{t}$ and the final nondominated set generated by each processor is represented in a similar fashion as $PF_{known}\binom{p}{}$.

In some pMOEAs, migration events occur in which select population members are sent to, received from, or exchanged with other processors. In such cases the Pareto front sets are defined prior to and following each migration event as the addition or replacement of population members may cause the Pareto sets to change. The current Pareto sets on any given processor are represented by $PF_{current}\binom{p}{t_bm_x}$ prior to the migration event and $PF_{current}\binom{p}{t_am_x}$ after migration occurs. The notation $\binom{p}{t_bm_x}$ denotes processor p prior to the xth migration event during generation t. Once migration occurs it is possible that some of the received members dominate some of the current members in $PF_{current}\binom{p}{t_bm_x}$. The current population, including the received population members is analyzed to determine which of the population members are nondominated. The notation $\binom{p}{t_am_x}$ denotes processor p following the xth migration event during generation t. Subsequent to the final migration event in generation t, the current nondominated set for processor p is represented by $PF_{current}\binom{p}{t}$, consistent with the notation used for serial MOEA Pareto sets. This distinction is necessary as each processor may conduct multiple migration/replacement events at a given generation, dependent upon its neighborhood size and number of memberships in different neighborhoods. A neighborhood represents the local processors (either in a logical or physical sense), that a processor is restricted to communicate. The notation just described applies to the known Pareto front sets also. At pMOEA termination, $PF_{known}\binom{p}{}$ represents the known Pareto front set found on each processor, $PF_{known}\binom{}{t}$ represents the cumulative known Pareto front found by all pMOEA processors combined for generation t, and PF_{known} represents the overall Pareto front set found across all processors at MOEA termination.

The pMOEA notation described can be employed in the development of a specific computational pMOEA (algorithm and data structures) to help ensure the design is implemented with a large degree of confidence the code is correctly written. The notation can also be used to aid in mathematically proving various pMOEA properties including convergence to the Pareto front. The pMOEA notation described can aid a researcher in the design, development, and implementation of a pMOEA. This notation is useful in aiding one's understanding of the migration process and the effects it has on the Pareto solution sets. Using a clear and defined notation helps to prevent confusion in this process and contributes to the pMOEA field.

7.2.2 pMOEA Motivation and Paradigms. As discussed in Chapter II, Section 2.1 single objective EAs and other deterministic and stochastic search techniques may not be as effective as MOEAs in the attempted solving of MOPs. The motivation for parallelizing MOEAs is to increase the efficiency, effectiveness, or both characteristics of an MOEA. Prior to using a pMOEA, one must ensure that an MOEA is as good as or is a better choice than other search techniques for the MOP of interest. Once this is established, a researcher must determine if a pMOEA presents advantages over the serial approach. In general, pMOEAs are most useful when a researcher is attempting to solve an MOP consisting of computationally expensive fitness functions or requires a higher quality solution than a serial approach generates. It is important to note that parallel MOEAs do not guarantee more effective results compared to serial MOEAs but the probability exists of generating better solution sets as more of the search space can be typically covered in a parallel implementation.

Increasing the efficiency of an MOEA through a pMOEA approach involves using parallel decomposition techniques. The calculation of the fitness functions is typically the most costly operation in a pMOEA [6]. The decomposition of the fitness function calculations across many processors may decrease the wall clock time required to execute the pMOEA. Increasing the effectiveness of a MOEA through a parallel approach typically involves the execution of an MOEA on multiple processors. This allows a researcher

to search more of the space than a serial MOEA and hence increase the possibility of generating better potential solutions to the MOP.

The motivation for applying parallel concepts to MOEAs is typically to increase the efficiency or effectiveness of the search process and hence improve the performance of the MOEA. The reader is referred to Chapter II for a general discussion of the motivation for applying MOEAs to MOPs. Decreasing the time required to find solutions to an MOP and decreasing the computer system resources required for the search process are part of increasing the efficiency of the MOEA. Increasing the effectiveness involves finding solution sets of "better" quality, solution sets of higher cardinality, and solution sets containing a "better" distribution of points along the front.

Design characteristics of pMOEAs include the concurrent search for multiple solutions, ease of mapping serial MOEAs to parallel MOEAs, reducing wall clock execution time, hybrid interfacing to other search techniques for complex MOPs, and achieving overall better performance in terms of efficiency and effectiveness [6]. In considering MOEA parallelization, a researcher must understand the various parallel paradigms, the MOEA, and the MOP in order to select a paradigm that appears to have the greatest potential for increasing the performance of the MOEA as applied to the specific MOP of interest. For example, in MOPs containing complex, time consuming fitness functions, the decomposition of the population across a number of processors for evaluation may provide the best increase in performance of the MOEA if the researcher is interested in decreasing execution time. This is just one example but illustrates the need to understand the pMOEA and characteristics of the MOP. Architectural characteristics may or may not be of interest to the implementation of a pMOEA. These may include numerical processing, communications, network topology, processor speed, memory access, I/O, etc.

In order to fully understand MOEA parallelization, one must first identify the basic EA components lending themselves to parallelization [6, 20]. Parallelizing the objective functions is a simple and potentially useful idea but one that inherently only decreases execution time and does not effect effectiveness. Some pMOEAs are potentially more effective than their serial counterpart, but effectiveness gains typically are not realized without cost, which may be an increased execution time. Researchers must consider these

tradeoffs in making a decision as to which parallel paradigm to employ. Understanding parallel concepts and their potential usefulness in MOEAs requires a discussion of MOEA algorithms, MOEA operators, and resulting generic MOEA code. Figure 7.1 presents generic MOEA pseudocode reflective of the more common MOEAs found in the literature and discussed in Chapter III.

```
Perform Population Initialization (Size P)
Compute Each Population Member's Fitness
                        (w.r.t. k functions)
Loop
    Perform Clustering/Niching/Crowding
    Execute EVOPs)
    Compute Each Population Member's Fitness
                        (w.r.t. k functions)
    Conduct Selection
    Generate PF_current (t); Update PF_known (t)
End Loop
Conduct Local Search (If Specified)
Generate PF_known ; Present As Final Solutions
```

Figure 7.1 Generic MOEA Pseudocode [31]

The relatively inexpensive cost of Commercial Off The Shelf (COTS) parallel computing platforms, has increased the availability of these systems to researchers. With the increased availability of parallel processing platforms comes an increased use of these resources by those with varying levels of knowledge and understanding of the associated parallel processing concepts. This chapter can aid the inexperienced as well as experienced researcher in the implementation of pMOEAs. The reduced cost of these resources and availability of numerous, yet different, hardware and software platforms leads to many possible configurations and implementations. This includes the use of homogeneous supercomputers, where each processor has identical resources available to heterogeneous COTS clusters of PCs, where the processor speeds and resources available may vary based on each node [18, 119]. The characteristics of these systems are listed in Table 7.1.

This chapter addresses generic pMOEA concepts and their application to homogeneous parallel systems. Homogeneous systems are assumed to have identical resources available. These resources include processors, backplane, memory, storage capability, net-

Table 7.1 Parallel Processing System Characteristics

	System	Attributes
Homogeneous	Cluster of PCs	– Homogeneous COTS PCs – Homogeneous COTS communications backbone
	Supercomputer	– Specialized hardware/software – Homogeneous CPUs, RAMs, caches, memory access times, storage capabilities and communications backbones
Heterogeneous	Cluster of PCs	– Heterogeneous COTS PCs – Heterogeneous CPUs, RAMs, caches, memory access times, storage capabilities and communications backbones

work backbone, operating system, software and more. The parallel concepts discussed can be extended to other configurations (heterogeneous) with varying hardware and software resources available through load balancing techniques and other concepts [18, 119]. There are many possibilities for heterogeneous implementations, hence an in-depth knowledge of the system is required and they are not discussed in detail.

Parallel paradigms can be used to decompose a problem (task and/or data) and in turn decrease execution time. These paradigms can also be used to attempt to search more of the solution space, potentially finding "better" solutions in the same amount of time as a serial implementation. With the general performance objectives of increasing the efficiency and effectiveness of an MOEA, generic pMOEA architectures are now addressed.

The three major pMOEA computational paradigms are considered. They are the "Master-Slave," "Island" and "Diffusion" paradigms [6, 20, 31]. A combination of any or all of these three paradigms may also be considered and is referred to as a hybrid approach. Each paradigm may be implemented in either a synchronous or asynchronous fashion and each has its own particular considerations. Synchronous implementations are defined as utilizing "same-generation" populations where some sort of inter-processor communication synchronizes all processes at the end of each generation. Asynchronous implementations can reduce processor idle times and may be most useful in situations where varying processor speeds, memory, or hardware exist. In an asynchronous implementation, communications occur at varying times and there is no guarantee that the destination

processor actually receives all of the information sent unless additional communications are present to guarantee delivery of the messages.

In recent publications, Cantú-Paz [19, 20] and Alba [6] present a good summary of existing single objective parallel EA approaches and useful background information. Cantú-Paz provides an in-depth discussion of the effects of various migration and replacement policies in single objective parallel EAs. While Cantú-Paz addresses two different migration and replacement strategies, many others exist. His effort concentrates on four different combinations of migration and replacement strategies (in the format migration strategy-replacement strategy); random-random, random-best, best-random, best-best. Many other possibilities exist and hence his work is not complete in this area. The research effort presented in this chapter addresses migration and replacement strategies also, but considers these strategies in a multiobjective optimization and attempts to present a more complete list of innovative possibilities. Alba also provides another perspective to single objective parallel EAs but concentrates more on a historical perspective and discusses additional hybrid approaches without presenting new concepts or results.

Coello Coello et al., present what the authors state is a summary of the existing pMOEA approaches currently available at the time of publication of their book in 2002 [31]. While the discussion of pMOEAs is the most comprehensive to date, their work is only a brief overview of parallel concepts and their application to MOEAs. Their discussion does not address parallel paradigms in detail nor are the possibilities for migration and replacement strategies in pMOEAs discussed. The migration and replacement strategies are integral to the use of the island and diffusion paradigms and hence must be addressed. However, Coello Coello et al. do present a thorough summary of the existing pMOEA papers and categorize these publications based on the paradigm the authors implemented.

The deficiencies present in Coello Coello et al. motivate the presentation of research material contained in this chapter. Background information is provided to aid the reader in understanding the concepts discussed. The three main paradigms for use in pMOEA development are presented.

7.2.2.1 Master-Slave pMOEA Paradigm. The Master-Slave paradigm is one of the easiest to understand conceptually. In this paradigm, a master processor exists to control the execution of the algorithm and the evolutionary operators (EVOPs) while a number of slave processors are present to conduct the fitness evaluations in parallel. The master processor executes the EVOPs and other miscellaneous pMOEA overhead functions including computing the Pareto front and partitioning the population across the slave processors. The search space exploration of this paradigm is conceptually identical to that of a serial MOEA. In other words, the number of processors being used is independent of the quality of the solution sets found, but does effect the overall execution time. The master-slave paradigm is illustrated in Figure 7.2. The master processor distributes population members, controls when/where fitness function evaluations are performed and stores the returned fitness values from the slave nodes [20, 31, 81, 82]. The master processor may also compute fitness function calculations, but this is only typically done when the population cannot be evenly distributed across the slave nodes.

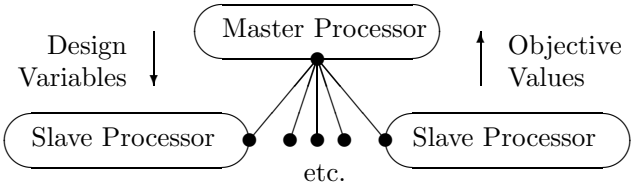

Figure 7.2 Master-Slave pMOEA Paradigm [31]

Since the usefulness of this paradigm lies in the increased efficiency of the pMOEA, it is important to realize that the fitness function calculations need to be fairly complex and time consuming in order to realize any computational speedup. The efficiency is increased through a decrease in the execution time of the pMOEA as compared to the serial MOEA. Speedup is defined as the ratio of the best serial MOEA run time over the pMOEA run time [119]. In some cases, a pMOEA could require additional execution time over a serial MOEA if the time required to communicate the population members to and from the slave processors is longer than the time required to compute the fitness values on the slave processor. This is usually encountered with simplistic objective functions where the communication time overwhelms the computation time or in cases where a larger number of

slave processors is used than there exists work to be partitioned and hence a poor speedup is realized.

Distributing the k fitness function evaluations over a number of slave processers can be implemented in three main ways:

1. An even distribution of the population across the slave processors. Each slave calculates each of the fitness functions for all of the population members received.
2. An even distribution of the fitness functions across the slave processors. Each of the slave processor calculates one fitness function value for the entire population.
3. A decomposition of the fitness functions so that each fitness function is distributed across multiple processors.

The first method of distributing the fitness functions involves evenly distributing the population members across the slaves processors. Each slave processor calculates each of the fitness values for all of the population members received. If the population cannot be evenly distributed the master processor may evaluate the fitness functions for a limited number of individuals. Since each of the slaves computes the identical fitness functions, for an identical number of population members, each slave usually completes the fitness function evaluations within the same amount of wall-clock time. This method is the simplest of the three and does not involve an in-depth analysis of the fitness function, except to determine that their complexity and execution time requirements lend themselves to parallelization. Examples of this method exist in Cantú-Paz [20] and other researcher's work [6, 81, 82, 145]

Master-slave pMOEA execution time is easily modeled based on an SOEA master-slave base case [20]. First, EVOP execution (selection, crossover, and mutation) time is ignored as the execution time required for these operations is typically less than that of fitness function computations in this paradigm. Let T_{c_p} represent the time required to communicate between processors prior to calculating the fitness function (the entire population must be transmitted) and T_{c_a} is the time required to communicate between processors after the fitness function evaluations. These values are typically different as the entire population member must be communicated to the slave processors but only the

fitness function values must be returned to the master processor following evaluation. The time required to execute the first method of distributing the fitness function evaluations is estimated in Equation (7.1)

$$T_{MS} = G \times \left(P(T_{c_p} + T_{c_a}) + \frac{n \sum_{i=1}^{k} T_{f_i}}{P} \right) . \tag{7.1}$$

where P is the number of processors used, n is the total population size, $\sum_{i=1}^{k} T_{f_i}$ is the time required to evaluate one individual for all k fitness functions, and G is the number of generations. This equation may be used to predict performance bounds on various architectures.

The second method of distributing the fitness function evaluations involves assigning the k fitness functions across k processors. Since each processor is required to calculate only one fitness function value for the entire population, each population member must be communicated to each processor. This method may yield varying computational loads among the slaves as each fitness function may require a different amount of wall clock time and be of varying levels of complexity. Let T_c represent the time required to broadcast the entire population to each of the processors and T_{c_a} is the time required to communicate the fitness function values. The time required to execute the second method of distributing the fitness function evaluations is estimated in Equation (7.2).

$$T'_{MS} = G \times (T_c + PT_{c_a} + n * (max(T_{f_i}))) . \tag{7.2}$$

where P processors are used, n is the total population size, $max(T_{f_i})$, where $i = 1, \ldots, k$ is the time required to evaluate the most complex fitness function and G is the number of generations. This equation may be used to predict performance bounds on various architectures.

The third and final method of distributing the fitness function evaluations discussed involves decomposing each of the fitness functions across multiple processors. This decomposition is only useful in situations where the high level of complexity and associated execution time of each fitness function warrants this type of decomposition. The fitness

functions are partitioned in such a manner that each fitness function is distributed across multiple processors. This method may decrease execution time as long as each slave processor takes longer to evaluate each of the population members than to communicate with the master processor. Since fitness functions are decomposed across multiple processors no guarantee exists that each slave processor is assigned equal computational loads; hence, some slaves may be idle for longer periods of time than others.

Real-world problem domains such as Computational ElectroMagnetics or Fluid Dynamics (CEM or CFD) often partition the fitness function evaluations among slave processors. Example parallel codes incorporating load balancing already exist for CEM and CFD applications. These parallel codes could be of great use in efficiently solving CEM and/or CFD optimization problems with a pMOEA, assuming that CEM and CFD problems lend themselves to a multiobjective MOP formulation [126, 127, 128, 155]. pMOEAs solving these MOP types are often interfaced to a specialized problem domain dynamic simulation model written by scientists or engineers from the specific discipline. In general, such interfacing is not very difficult [23].

The time required to execute the third method of distributing the fitness function evaluations is estimated in Equation (7.3).

$$T''_{MS} = G \times (T_c + T_{com} + PT_{c_a} + n * (max(T_{f_i j}))) \ . \tag{7.3}$$

where T_c is the time required to broadcast the entire population to each of the processors, T_{com} is the time required to combine the decomposed fitness function values, T_{c_a} is the time required to communicate between processors after the fitness function evaluations, P processors are used, n is the total population size, $max(T_{f_i j})$ is the time required to evaluate the most complex fitness function where $i = 1, \ldots, k$ and j is the number of partitions each fitness function i is decomposed into, and G is the number of generations. This equation may be used to predict performance bounds on various architectures.

Efficiency is the master-slave paradigm's main objective, and hence, the actual method of fitness function decomposition used is important to achieve the highest efficiency levels. A particular method may be preferred based on available computational

resources and the level of complexity of each of the fitness functions that are present in the MOP being solved.

7.2.2.2 Island pMOEA Paradigm. "Island" paradigm pMOEAs are based on the phenomenon of natural island populations evolving in relative isolation from other islands, such as might occur within some oceanic island chain. These pMOEAs are sometimes called "distributed" as they are often implemented on distributed memory computers; they are also called multiple-population or multiple-deme [6]. This paradigm is illustrated in Figure 7.3. In Figure 7.3 the communication channels for migration of selected individuals is illustrated in a ring fashion but specific paths are assigned as part of the pMOEA's design strategy and are then mapped to some physical communication backbone of a parallel platform. Communication may occur in logical or physical geometric structures such as rings, meshes, toruses, triangles, and hypercubes.

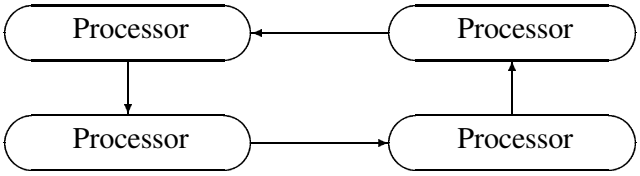

Figure 7.3 Island pMOEA Migration Paradigm [31]

Each processor (or island) in this paradigm, executes an MOEA simultaneously with other processors. Each of the processors has an independent or separate (sub)population (or deme) resident on it. The MOEA, and its associated parameter settings, executing on each island may be identical to the MOEA executing on other islands or may be entirely different. This characteristic of an island paradigm pMOEA mimics the differences noted between inhabitants of oceanic island chains. Each island evolves in isolation of the other islands for a portion of the pMOEA run but individuals occasionally migrate between one particular island and its neighbors based on some criteria and neighborhood structure [20]. Identical or different EVOPs may operate on each island, strongly implying each population is searching very different regions of the overall solution space. One goal of this process is to search a larger portion of the landscape, in the same amount of time,

as a serial MOEA. This additional search capability increases the possibility of generating more effective results.

Using different random number generators, random number seeds, and MOEA parameter values further strengthens covering more of the landscape. The island paradigm requires identifying suitable migration policies to include how often migration occurs (i.e., how many generations between migration events), how many population members migrate, how the population members are selected for migration and how the population members are selected for replacement on the receiving processor [20]. The EVOPs executing on each island control the mixing of BBs present in the population. The migration events work to distribute the "best" population members which typically contain the "best" BBs across the islands. Each MOEA processes the good BBs and may generate better solutions than a serial MOEA. While the generation of better results by a pMOEA is not guaranteed, the additional search capability of multiple processors and mixing of the BBs through the different EVOPs increases the probability of generating different population members compared to a serial MOEA.

Variants of the island paradigm are possible. These variants may use different communication strategies, different population representations, different decision variable resolutions on each island or even different search methods on each island. An island implementation is sometimes termed *course-grained* parallelism because each island (processor) contains a large number of individuals. Considering that many variants of the island paradigm are possible, it is not feasible to address each possibility. However, one can group variants of the island paradigm into four major groupings. The four major variants of the island paradigm are:

1. Homogeneous MOEAs and/or parameters.
2. Heterogeneous MOEAs and/or parameters.
3. A combination of different optimization approaches, not necessarily multiobjective.
4. Partitioning of the genotype or phenotype region so as each island can search a different, specified area of the landscape.

The first and second island paradigm approaches are relatively easy to understand. The first involves the use of the same MOEA with the same parameter settings executing on various processors with intermittent communications occurring. This approach is conceptually the simplest and possibly the easiest of the four variations to implement and appears in a number of publications [56, 124]. The second variant of the island paradigm involves the use of different parameter values on each of the islands. Each of the islands may execute an identical MOEA (with different parameter values as in Schnecke and Vornberger [171]) or entirely different MOEAs on each island.

The third variant of the island paradigm is the combination of different optimization approaches. These optimization approaches do not necessarily have to be of multiobjective nature. One of the methods used to solve an MOP is the use of a single objective approach executed multiple times. An example pMOEA using this approach combines single and multiobjective EAs to attempt and solve an MOP. In this approach, DCMOGA: Distributed Cooperation model of Multi-Objective Genetic Algorithm, Okuda, Hiroyasu, Miki, and Watanabe execute different single objective EAs and an MOEA on multiple islands [150]. Periodic migrations occur between the islands and at the termination of the pMOEA, PF_{known} is presented to the researcher.

The fourth variant of the island paradigm involves partitioning the genotype or phenotype space amongst the islands. Periodic migrations allow each island to maintain a population within the bounds it is assigned in decision variable or fitness function space. Deb, Zope, and Jain [45, 55] implement this approach in a pMOEA. The authors approach is based on the idea of isolating each processor to search a specific phenotype region but allows each processor to devote some effort to searching the entire space. The authors further implement a biasing mechanism to bias the search and in turn force each island to search only a particular region of the space [55].

Given that within each generation T_{ce} is the time for all islands to complete execution, T_{mig} is the time to complete neighborhood migration, T_{coll} the time to collect/compute the overall Pareto front/Pareto optimal set, and G is the number of generations, the island

paradigm's meta level running time, T_I, may be *estimated* as presented in Equation (7.4).

$$T_I = G \times (T_{ce} + T_{mig} + T_{coll}) \tag{7.4}$$

7.2.2.3 Diffusion pMOEA Paradigm. Like the master-slave paradigm, the "diffusion" paradigm deals with one conceptual population, where each processor holds only one to a few individuals. This is sometimes referred to as fine-grained parallelism [6]. A neighborhood structure is imposed on the processors and EVOPs occur only within those neighborhoods. These neighborhoods may be overlapping static neighborhoods or dynamically may change over a number of generations. The neighborhoods may be a square, rectangle, cube, or other shape depending upon the number of dimensions associated with the diffusion algorithm topological design. As "good" solutions arise in different areas of the processor pool, they then spread or diffuse slowly throughout the entire population due to the overlapping or dynamically changing neighborhoods [20, 31].

The communication costs may be very high within a neighborhood as the pool of individuals to select from are distributed across a number of processors. This paradigm is illustrated in Figure 7.4. Observe the example shown is implemented on a logical mesh with a square neighborhood of four processes. The overall logical grid communication structure is mapped to some physical communication backbone. Other examples of possible logical neighborhood structures include a ring, a square, a torus or a triangle, each reflecting some associated number and arrangement of neighbors within a multi-dimensional grid.

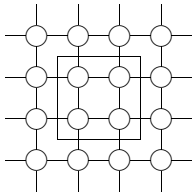

Figure 7.4 Diffusion pMOEA Migration Paradigm [31]

This paradigm typically involves more communication than the other paradigms and hence may be best suited to specific situations where the communication overhead does not degrade the overall performance of the pMOEA. For example this paradigm is suited

to shared memory architectures where communication is unnecessary as each processor has access to the memory location where the entire population exists at high communications speeds. This paradigm also may require additional parameter settings over the other paradigms and may offer many more alternatives to the communications process. One can control the communication within a neighborhood as well as the communication between neighborhoods differently, possibly offering more capability to the researcher, but at additional parameter tuning. An alternative to the three main paradigms discussed is a hybrid paradigm or a combination of any or all of the three main paradigms. This paradigm is what Cantú-Paz terms the class of hierarchical hybrids; at a high level of abstraction these are multiple-deme algorithms with each deme executing a particular MOEA instantiation [20:pp. 126-128]. He proposes three island paradigm hybrids (abstracted to the pMOEA domain) where:

1. Each island contains a diffusion pMOEA,
2. Each deme is a master-slave pMOEA and
3. Each island is composed of an island pMOEA.

Another innovative computational pMOEA design incorporates a co-evolutionary hierarchical pMOEA. A tree or graph search structure is employed in an attempt to find better pMOEA algorithmic structures for a given problem domain. In this paradigm, the leaves of the tree structure are various MOEA instantiations with associated parameter values. As their concurrent execution completes, the next level of the tree evaluates the performance of the lower leaf nodes and selects new parameter values based on those results; they may also deterministically or stochastically add other algorithmic constructs to improve their local MOEA instantiation. This development process continues through the tree's levels to the root node [95, 200]. Assuming a binary tree structure with n MOEAs executing at the lowest level and l levels, this paradigm is illustrated in Figure 7.5. Note that each leaf could have any number of children, and although the complexity would be much higher any particular node could itself be a pMOEA.

7.2.2.4 Related pMOEA Design Issues. Regardless of the pMOEA paradigm implemented, several issues must be addressed in order to ensure that equivalent or po-

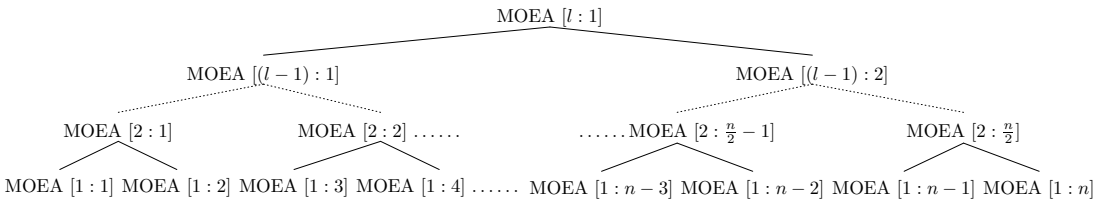

Figure 7.5 Example Hierarchical Paradigm

tentially improved performance is achieved through the parallelization of an MOEA. In designing and implementing a new pMOEA from scratch, guidelines and issues discussed can help to ensure major problems are dealt with prior to implementation of the code. In considering the parallelization of an existing MOEA, an evaluation of the code must be conducted in order to determine if the algorithm lends itself to parallelization or if a rewrite of the code may be more beneficial.

In the master-slave paradigm, the pMOEA developer must account for the possible effects (both good and bad) of data and fitness function decomposition. An MOEA's underlying data structures may effect effective and efficient parallelization. In other words, how and where necessary data is stored, its quantity, to where (and when) it must be communicated, the parallel communication libraries selected, the underlying hardware used, and the implementation of the parallel communication operations directly effects the overall performance of the pMOEA. For example, consider a generic master-slave pMOEA implementation evaluating k fitness functions where each slave processor evaluates only one fitness function. Given these conditions perhaps only a subset of the components of the underlying data set (e.g., population, decision variables, input data) are needed by each processor, but if each processor computes a completely different fitness function, and a shared memory system is not used,[1] the entire data set may need to be sent to each processor for evaluation. In this example it may be more efficient to have each processor evaluate all of the fitness function values for a portion of the population. Another issue encountered with the use of the Message Passing Interface (MPI) as the underlying communications library is that data to be sent in a single message must be present in contiguous memory locations otherwise multiple messages are required [156]. When sending a large population of individuals, it is more efficient to send a single large message versus sending many

small messages. This illustrates how critical it is to understand the underlying hardware constraints of the system selected and the method in which parallelization is implemented. In real-world design problems this associated data set may be quite large. In many cases, the communications between processors is a considerable portion of the execution time of a pMOEA. Therefore, reducing the amount of communications required may speed up overall algorithm execution.

Inhibiting the ability to effectively and efficiently parallelize an MOEA is its inherently sequential nature due to its EVOPs' attributes. Each EVOP typically requires access to the entire population in order to "effectively" conduct the operation in a Master-Slave design. However, fitness function computations typically do not have that limitation. It is instructive to consider how parallelizing multiple fitness function computations might be accomplished and what implications may subsequently arise. Careful consideration reveals this task allows parallelism in one of three ways. Various options include, each function's evaluation might be assigned to different physical processors, (sub)populations might be assigned for evaluation of all k functions on different processors, or each individual's evaluation for one of the k objective functions might be assigned across several processors. The following issues are identified for consideration given these Master-Slave variations.

1) Assign each function evaluation to a processor

When implementing the first option one must consider the possibility that each fitness function's execution time may be substantially different. This is easily illustrated in a heterogeneous environment where processor speeds vary greatly between different machines. In this example, randomly assigning each of the k function evaluations to processors may not be the most efficient strategy if one function's evaluation takes several times longer than any of the others (i.e., $T_c(f_1) >> T_c(f_i) >> \ldots >> T_c(f_k)$). Static or dynamic load balancing may help improve the efficiency in this example but additional overhead is required [119]. Consideration of the communications backbone is essential to implementing an efficient pMOEA. If a high-speed communications backbone is present in a shared memory system, the requirement that each processor use the entire population can be efficiently executed. However, this same requirement on a non-shared memory system with a low-speed communications backbone most likely cannot be efficiently executed.

2) Assign subpopulations to processors

When implementing the second option, equal fractions of the population are assigned to different processors where all k objective functions are evaluated. Here, identical numbers of individuals are evaluated via identical objective functions. If communications are not a large fraction of each processors' computational workload, then this is likely an efficient parallelization method. However, note that in some design problems objective function evaluation cost may widely vary.

3) Parallelize each function evaluation.

The last option may be useful when evaluating extremely expensive objective functions. In this case, each individual's evaluation of each function is split among processors. As stated before, some form of load balancing may be required to obtain efficient use of processing resources as each fitness function and/or each individual's evaluation may consume differing amounts of computational resources. Problem domains such as Computational ElectroMagnetics or Fluid Dynamics (CEM or CFD) may find this option useful. Example parallel codes with load balancing already exist and they can be of use in efficiently solving CEM/CFD optimization problems using a pMOEA [126, 127, 128, 155].

Island paradigm pMOEAs encounter similar issues related to the efficiency of the pMOEA. The overall goal in a non-shared memory system is to reduce the amount of communications that are required as each communication event takes time to complete. In the island paradigm, communications are required but possibly not as often as in the master-slave paradigm. The island paradigm occasionally migrates individuals in order to maintain diversity in the population and propagate the "good" BBs to other processors. A tradeoff must be made between the amount of communications deemed necessary to find "good" potential solutions and the overall pMOEA execution time that a researcher can tolerate.

Communication in the island paradigm involves how often one communicates but a more basic issue is which population members to communicate. Single objective island paradigms typically send the single best population member or the top few (usually a small number) population members. In pMOEAs using the island paradigm, there typically is

no single best or top few. Instead, a multiobjective approach using Pareto dominance generates a Pareto front which contains the best found population members. Considering that $PF_{current}\binom{p}{t_bm_x}$ typically varies in size, this increases the difficulty of implementing an island pMOEA. Another related issue is the selection of members to replace on the receiving processor. In single objective EAs, the migration of the best member may lead to the replacement of the worst member on the destination processor. In pMOEAs, the migration of a set can lead to a number of possibilities in terms of replacement. There typically is no single worst member but a set of worst members if one considers ranking the Pareto fronts. Hence, the replacement policy is not straight forward either. Currently the literature offers little guidance in the area of migration and replacement in pMOEAs [26].

Related to this communication issue is the selection of a population size to use on each island processor. If the goal is to reduce the execution time as compared to the serial implementation, one may choose to use a population size of p/n where p is the serial population size and n is the number of nodes. Alternatively, one may allow the pMOEA to execute within a similar time to the serial MOEA and hence allow a population of p on each processor. This increases the probability of finding "better" solutions as more of the landscape is searched in the same execution time as the serial MOEA. Finally, the overall solution quality may be of primary concern and a population size of much greater than p may be used on each processor. Each of these possibilities may also be dependent on the selection of appropriate migration and replacement policies, intervals, neighborhood sizes, and definition of neighbors. The policy dictates which population members are migrated and which are replaced on each processor while the interval dictates how often these events occur. The neighborhood size and definition of neighbors dictates how many migration and replacement events occur each generation and among whom. The larger the neighborhood, the greater the number of communication events, and typically the longer the algorithm takes to execute.

At first glance a researcher might interpret the execution time of the diffusion paradigm as less than the execution time of a serial MOEA since an overall population size of p is typically used with only a few population members per processor. However, this is highly dependent on the underlying communications and memory architecture. Since the EVOPs

are conducted only within each neighborhood, a large amount of communication is necessary to execute this paradigm. Additionally, the definition of the neighborhood may be critical to the performance of this paradigm as well as the number of population members present on each processor. The complexity and number of parameters associated with the diffusion paradigm may be the reason for the low popularity of this paradigm [26].

Another issue discussed in the pEA literature and applicable to pMOEAs concerns reports of superlinear speedups. This issue is present in the pMOEA community as well [206]. Claims of superlinear speedup may cause controversy. Superlinear speedup occurs when the parallel algorithm execution time is reduced by a factor greater than the number of processors used. Cantú-Paz addresses this issue in some detail [20:pp. 114-117]. It appears many researchers report superlinear speedups when parallelizing their EAs and one might expect pMOEA researchers to do the same. The issue lies within the comparison of the serial EA's effort expended versus the parallel EA's. For example one can attribute superlinear speedup to the fact that a pMOEA finds better solutions by not examining the same number of points as a serial MOEA, or through the use of a smaller or different phenotypical region for parallel search.

The typical argument against superlinear speedups is summarized by Cantú-Paz who argues the main reason to distrust superlinear claims is that if one was to execute all tasks of a parallel program using threads on a single processor, the total execution time cannot be less than that of a serial program performing the same computations [19]. He states the assumption is that the serial and parallel programs execute the exact same tasks which is typically not true in parallel EAs or pMOEAs. One must then cast superlinear speedups with the statement that identical areas of the search space may not have been searched by both algorithms.

For example, the master-slave paradigm's search process is the same as an associated serial MOEA. The number of fitness evaluations is equal and chromosomes created are identical. Depending upon the fitness function's complexity and number of processors used, one may then obtain a close to linear speedup performance. However, an island or diffusion paradigm's implementation requires statistically uncorrelated random number generators and the resulting landscapes are completely different than that of a serial version. As

such, the pMOEA may find a "better" or equivalent solution quicker or even slower than the serial version. One must be careful in stating claims of superlinear speedup without describing the actual portion of the search space covered by each algorithm.

These are some of the major design issues encountered in the development of a pMOEA. The underlying hardware, software, and data structures are all integral components in designing an efficient and effective pMOEA. Numerous minor issues must be dealt with in the development of a pMOEA. A discussion of such issues is beyond the objectives of this effort. This summary is presented to aid a researcher in addressing many of the important issues in pMOEA development.

7.3 pMOEA Analyses and Issues

This section presents a largely qualitative analysis of the currently known pMOEA research. High level topics are addressed, first highlighting several issues that are given little attention or even ignored in the literature. The section concludes with a broad discussion of the major factors to consider when developing, implementing and analyzing pMOEAs, as well as supporting claims of important issues currently unexplored in the pMOEA field.

7.3.1 pMOEA Observations. An active research community interest in pMOEAs is not currently indicated. In fact, very few pMOEA publications exist in over 1000 contemporary MOEA publications [26, 31, 62, 110, 120, 151, 181]. This indicates that MOEA researchers currently are not pursuing pMOEA techniques. This may be due to the complexity of parallel and distributed computation, a limited understanding of parallel processing techniques as well as the difficulty with debugging parallel code.

As is the case with their serial MOEA analogues, pMOEAs have been applied in solving various real-world engineering problems. For example, Coello Coello et al., include pMOEA solutions to a variety of MOPs [31]. Many of these approaches use the master-slave computational parallel paradigm because of the complex objective functions. However, Cantú-Paz cites that island paradigm implementations are the most popular parallel SOEAs [20:p. 49]. The same fact exists for pMOEAs. Almost half as many island

implementations are identified as the master-slave and diffusion combined. This may be due to the island paradigm's ease of implementation as serial MOEA code is easily integrated to create a pMOEA. This is especially true in the case of the island paradigm where little to no inter-processor communication may be selected.

Some of the real-world application areas of pMOEAs include the following master-slave pMOEA implementations: microprocessor cache memory design [178], airfoil design [104, 126, 127, 128, 155], computational fluid dynamics [129], mobile phone network design [137], and X-Ray plasma. Some of the real-world application areas using the island paradigm pMOEA include: vehicle scheduling [15], airfoil designs [157, 158], rotor blade design [124], aerodynamic structural designs [56], and facility layout problems [171]. The diffusion paradigm has also been applied to real-world MOPs, some of these application areas include: route planning problems [98] and sensitivity analysis problems [161].

The known pMOEA papers generally do not address many details of the development and implementation of the actual pMOEA. Typically there is little discussion of why the pMOEA or parallel paradigm is selected for attempting to solve a specific MOP. In general, little topology, migration, or selection details are presented. Without an adequate discussion of the pMOEA used and the MOP being solved, one cannot evaluate the quality or performance of the pMOEA presented.

A few publications exist in which pMOEA researchers have presented some of these concepts in a clear manner. Okuda [150] presents an island paradigm pMOEA, utilizing $k+1$ islands, where k is the number of fitness functions in the MOP. In this implementation, k of the islands have a single objective EA executing to solve one of the k fitness functions as a single objective problem. The remaining island uses an MOEA to solve the fitness functions simultaneously. The communications between the islands involves the passing of the best population member with respect to the fitness function solved on that island to the MOEA or back to the single objective EA. The results of this method are promising but additional testing is necessary to evaluate the performance of this pMOEA.

In an application to the Vehicle Routing Problem, Jozefowiez [105] proposes a parallel technique for MOEAs which they refer to as *Elitist Diversification*. This innovative

technique involves maintaining multiple archives. A standard elitist archive is maintained, as typical in an MOEA, of $PF_{current}$. Other archives are maintained in an elitist fashion with respect to one of the objective functions. For example, in the case of a two objective maximization MOP, an archive is maintained containing the Pareto front when calculated as a maximization of both fitness functions, another as calculated as a maximization of f_1 and a minimization of f_2 and a last archive as a minimization of f_1 and a maximization of f_2. Solutions from all of the archives are included in the population with the hope that the archives push the population into unexplored areas of the landscape. This is implemented in an island paradigm in which periodic communications of the standard elitist archive takes place. The authors present other modifications of this scheme. The results show that this elitist diversification method improves the diversity along the Pareto front in some cases but also increases the execution time of the MOEA.

Another pMOEA island paradigm idea involves isolating each processor to solve a different physical region of the Pareto front. As one does not know where the Pareto front is prior to execution, this approach can only be implemented once the structure of the Pareto front has been identified. This has been referred to as a guided domination concept [45, 55]. The concept deserves attention and appears to be simplistic but in actuality may not be feasible. Partitioning the Pareto front into sections and distributing those sections to a number of processors attempts to allow each processor to search in a certain area of the landscape. The problem encountered is in how to ensure that the processors only search and generate points in their restricted area of the landscape. One can force the processors to generate points until enough are found in the restricted space to enter the next generation, but for certain MOPs, this process may take infinite time if it is even possible. There is no guarantee that selecting two points on a certain region of $PF_{current}(t)$ and performing recombination yields another point within that region. Another approach, possibly more efficient, may be to restrict each processor to a region of the genotype space based on $P_{current}(t)$. In this implementation, it may be possible to restrict each processor to utilizing genotype variables within a specific solution region.

A variant of the island paradigm, that at first glance appears to be an easy way of implementing a pMOEA is to isolate each processor to search a specific, non-overlapping

region of the phenotype space [45, 55]. The entire phenotype space must be covered and each portion of the space must be assigned to each of the processors a priori. A difficulty with this approach is that each island most likely generates phenotype values outside its constrained phenotype region. One way to handle this is to continually force each processor to generate points until the required number are found within its assigned region. For certain MOPs, this process may take a long time, and it is a waste of processing power and resources. Individuals could migrate to the appropriate processor or just be deleted from the population, but this may require phenotype sorting and additional communication overhead. The "standard" island paradigm, utilizing each processor to identify the complete Pareto front, appears to be a more efficient method even though neither method is guaranteed to generate PF_{true} .

A specific example of this variant assumes a convex Pareto front. It is based on the idea of isolating each processor to search a specific phenotype region but allows each processor to devote some effort to searching the entire space. Deb et al., proposes what the authors refer to as, a guided domination concept. The guided domination concept is defined using a weighted function of the objectives as presented in Equation (7.5) [55].

$$\Omega_i(f(x)) = f_i(x) + \sum_{j=1, j\neq x}^{M} a_{ij} f_j(x), i = 1, 2, \ldots, M \tag{7.5}$$

where a_{ij} is the amount of gain in the j-th objective function for a loss of one unit in the i-th objective function [55]. Deb further defines a different domination concept for minimization problems as:

Definition 45 (Guided Domination Concept): *A solution $x^{(1)}$ dominates another solution $x^{(2)}$ if $\Omega_i(f(x^{(1)})) \leq \Omega_i(f(x^{(2)}))\ \forall\ i = 1, 2, \ldots, M$ and the strict inequality is satisfied for one objective [55].* □

Using appropriate weights determined from known points on the Pareto front, Definition 45 enlarges the dominated region of the phenotype space as shown in Figure 7.6. The lines appearing in Figure 7.6 represent the weighted functions as they pass through

the objective space and cross at the Pareto front point A. The hashed region represents the region in objective space dominated by point A.

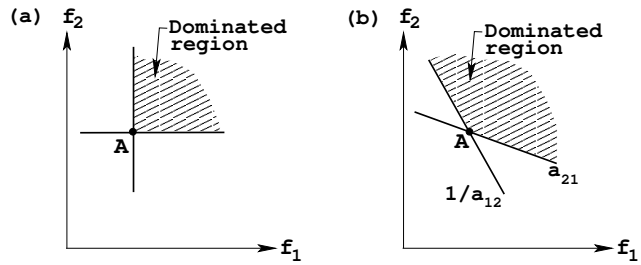

Figure 7.6 (a) Dominated Region (Standard Definition) and (b) Dominated Region (New Definition) [55]

Deb et al., state that the purpose of this is to bias the search through the weighted functions and in turn search only a particular region of the space. This bias is towards the center of where the weighted functions cross in objective space.

The new definition permits processor state-space domain overlap. Thus, the returned Pareto front may not be totally nondominated. Therefore, each processor using this new definition only finds the "real" convex Pareto front in a region based upon the old non-dominated definition. The other points found above and below this convex Pareto front in the entire search space are dominated in the real sense, but not in this new sense. In order to successfully accomplish this, one must have knowledge of a limited number of vectors on the known Pareto front. The number of vectors selected that allocate the overlapping regions of processor search is the number of processors required.

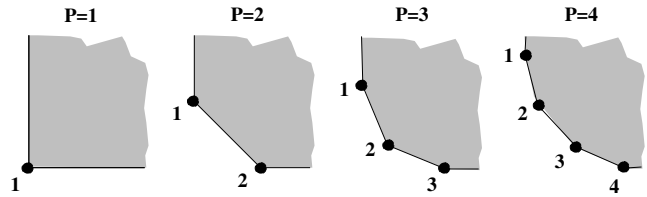

Figure 7.7 Generic Objective Function Processor Allocation ([55])

Using the Ω transformation, all regions of the convex Pareto front are accounted for with what one hopes is an "equal slice" of the front. With known convex examples, the concept deserves attention, but it appears to be somewhat simplistic. In actual real-world

problems, it may not even be feasible due to Pareto front structure. Even for convex fronts, problems occur in determining how to find initial members of the front to ensure the processors generate vectors in their restricted search region. This is the intent of the revised domination definition. Figure 7.7 presents a generalized allocation for two objective functions, given 1 to 4 processors. Observe there is no guarantee that selecting two or more members of the front on a given processor and in a certain region, yields another vector within that region through the use of EVOPs. This is only one variant of the island paradigm but illustrates the difficulty of attempting to assign regions of the phenotype space to specific processors for multiobjective optimization.

Many of the mentioned details are missing from pMOEA publications. This may be due to the fact that there is no outline suggested by researchers of which pMOEA attributes and settings are important to discuss. Additionally, none of the existing publications in the literature state what is required to design a "good" pMOEA [26, 31, 62, 110, 120, 151, 181]. However, this does not justify the limited details and discussions that appear in most pMOEA publications. No theoretical or practical studies are yet known comparing the efficiency and effectiveness of the three major pMOEA paradigms or discussing the possible parameter settings for the most popular paradigm, the island paradigm. Due to its popularity, one might expect to find a detailed comparison of the possible migration and replacement strategies but this type of study currently does not exist. This contribution is essential to support a decision as to which parallel paradigm a researcher can chose for implementation. These concepts are addressed in the following sections.

It is important to recognize that improvements are constantly being made to the speed and capabilities of computer systems. With these improvements in capabilities comes the potential to improve the performance of the various pMOEAs. Additionally, pMOEA researchers must use high performance computing and parallel EA experts to increase their understanding and ability to integrate parallel concepts with MOEAs. This integration of knowledge along with experts from the problem domain may lead to the generation of better solutions than currently exist for real-world application MOPs.

7.3.2 pMOEA Suitability Issues. As indicated in the previous section, the limited number of existing pMOEAs currently yield good results when applied to various problem domains. However, in general, the authors of these pMOEAs present little or no discussion regarding the suitability of the application of their pMOEA to particular MOPs [26, 31, 62, 110, 120, 151, 181]. More discussion of the characteristics of the MOP that make it suitable to use an pMOEA is necessary. This includes, but is not limited to, a discussion of the search space, the complexity of the fitness functions, the underlying data structures, and reasons why a pMOEA is anticipated to perform as good as or better than a serial MOEA. This information is critical to increasing the level of understanding of pMOEAs and MOP characteristics that make these MOPs suitable for parallel processing.

As stated in the EA and MOEA literature, a researcher should have a thorough understanding of the problem being solved and the algorithm selected in order to attempt to efficiently and effectively solve the problem [13, 31, 44, 78, 184, 190]. One must make a determination if any gains are anticipated to be realized from the use of a parallel approach as compared to the serial approaches that exist as additional overhead may be encountered with the added complexity of parallel processing. For example, it is fairly easy to realize that a master-slave pMOEA applied to an MOP incorporating complex fitness functions that require hours to calculate on a specific machine can realize performance improvements over a serial MOEA. This determination requires an understanding of the algorithm, problem and underlying hardware and software platforms chosen for execution. Dependent on a researcher's limitations in terms of time, resources, and goals sometimes makes this a complex determination.

Considering that researchers agree that an understanding of the problem and algorithm domains is critical to implementing an efficient and effective algorithm [13, 31, 44, 78, 184, 190], the following sections attempt to develop a better understanding of pMOEAs through a discussion of the various parallel paradigms. This discussion concentrates on the island paradigm due to its popularity and relative ease of implementation. The contribution made is viewed as an aid to help advance researchers' understanding of parallelism in MOEAs and their ability to make intelligent decisions regarding the effect of various parallel concepts in solving MOPs.

7.3.3 pMOEA Hardware and Software Architecture Issues. As numerous parallel computing platforms exist, it is not possible to discuss all of the available or possible hardware and software combinations. Researchers should consult the experts relating to the specific capabilities of any hardware or software system. The research presented discusses generalized techniques that are applicable to various systems. While each of these systems may or may not have specialized techniques that provide the ability to increase performance in any one aspect of the pMOEA execution, the generalized discussion presented is necessary to provide a high level understanding of issues that may be encountered.

The choice of implementation languages from a programming standpoint, such as FORTRAN, High-Performance FORTRAN (HPF), C++, ANSI C, C#, XML, to name a few, is based on the researcher's knowledge, availability of existing codes, or requirements of the systems available for use [6, 18, 119, 125]. Parallel communications are implemented through communication libraries such as the Message Passing Interface (MPI) [153, 156] or other specialized software such as the Parallel Virtual Machine (PVM) [72] or Open-MP [24]. Either includes communication routines that are readily incorporated into existing MOEA implementations. Both are portable across a wide variety of computer architectures with multiple processors (either homogeneous or heterogeneous), or symmetric multiprocessors (SMP) – multiple processors per node using the same local memory. Note that MPI, Open-MP, and PVM are a few of the many possible methods of controlling pMOEA communication. Others include JAVA and C++ sockets, JAVA, JAVA RMI (Remote Method Invocation), DCOM (Distributed COM), and CORBA (Common Object Request Broker), all of which use multi-threading middleware communication techniques in a distributed computational environment [125].

Standard models of parallel computer architectures include single instruction, single data stream (SISD); single instruction, multiple data stream (SIMD); and multiple instruction, multiple data stream (MIMD) [119]. SISD architectures are standard single processor computers. The SIMD architecture requires each processor to execute only a given broadcast instruction on different local data. This architecture for example is useful for a diffusion paradigm implementation. MIMD architectures are useful for execution of different EAs and MOEAs with different data. Most MIMD homogeneous architectures

and associated compilers reflect a distributed memory structure, this being the "more" generic MOEA environment for parallel implementation of the master-slave, island, or hybrid paradigms. Also, computational platform selection should address the issue of shared memory in SMP architectures versus distributed memory. Moreover, Internet implementations may also be considered for pMOEA implementations [12].

Several disparate MIMD architectures are used in current pMOEA implementations. For example, heterogeneous Sun workstations (80-120 processors), IBM SP-2s (8, 16, 32 and 64 processors), SGI Origin 2000s (8 processors), homogeneous LINUX boxes (24 processors), and Beowulf clusters, Cray T3Ds (32, 64 and 128 processors), a 16-node Motorola MPC 601, a 16-node SGI Power challenge, a heterogeneous DEC Alpha system (8 and 18 nodes), Sun Ultra workstations (2, 4 and 6 processors and 2, 3, 4, 6, 9 and 12 processors), a 32-node Multi-cluster T-800, and dual Pentiums to name a few [31]. Each of these systems may be efficiently used by multiple paradigms or only a single paradigm. This determination is made based on the details of the underlying hardware, software and requirements of the parallel paradigm. The computational capabilities of any system may have a large effect on pMOEA performance. Those capabilities include but are not limited to processor speed, cache and local memory capacities, local and global memory access speeds and communication backbones. Commonly used high speed backbones include Ethernet, Fast-Ethernet, Gigabit-Ethernet, Myrinet, Wulfkit, and the Fiber Distributed Data Interface (FIDDI) [18, 119]. It is also noted that the physical or logical architecture of the connection of the processing nodes may include rings, meshes, and hypercubes to name a few, each with advantages and disadvantages. The associated communication software may be tuned or optimized for the given hardware configuration and hence any given parallel library may yield drastically different performance given its implementation and underlying hardware and hence this must be taken into consideration.

7.3.4 pMOEA Test Function Issues. The selection of test functions for pMOEAs encounters the same issues discussed in Chapter IV. A wide variety of MOP test functions with varying characteristics should be used when evaluating the general performance of an pMOEA. Additionally, if the pMOEA is only going to be applied to a specific problem

domain, then its performance when applied to MOPs with similar characteristics must be evaluated. The main objective of testing an pMOEA implementation is to evaluate, compare, classify, and improve algorithm performance. In evaluating the performance of various algorithms, this testing should focus on the pMOEA's general effectiveness and efficiency. Specific MOP efficiency and effectiveness performance can also be summarized to determine the classes of MOPs for which various pMOEAs or pMOEA operators and settings achieve good performance.

Valid tests may include pedagogical or MOP test functions, a pMOEA test suite, combinatoric multi-objective problems and/or real-world problems. Before construction, the literature should be consulted as many researchers have spent considerable time on constructing, evaluating, and analyzing various MOP test suites [43, 46, 47, 52, 184, 209]. These MOPs as well as others can also be evaluated to determine if they possess characteristics that make them easier to solve with a parallel versus a serial approach.

General test suite guidelines are discussed in Chapter IV. While pMOEA test suites have not been explicitly defined in the literature, it is reasonable to state that various MOEA test function MOPs are valid to use for testing pMOEAs. MOEA approaches are useful when the landscape of the MOP is multimodal and a good search technique for the MOP is unknown. The same statement can be made of pMOEAs. Typically, pMOEAs can be considered useful for solving problems that have large search spaces and take a considerable amount of time to find good solutions; otherwise, an pMOEA may not be the best approach to solving this class of problem. Good problems to apply pMOEAs to include those with a large number of decision variables and fitness functions. This means that applying pMOEAs to small, simplistic problems may not illustrate the true performance of the algorithm. For example, in his study of parallel SOEAs, Cantú-Paz was forced to construct artificial test functions since solving his initially selected functions in parallel did not provide enough computational loading to provide meaningful results [20]. The use of existing MOP test suites may be too simplistic to show efficient and effective results when utilizing a pMOEA. The key is that one must apply a pMOEA to a problem domain for which it is well suited.

Researchers should select functions that are both "easy" and "hard" for pMOEAs to solve. This helps to identify the strengths and weaknesses of pMOEAs and possibly the particular parallel paradigm used. Selecting appropriate test functions for testing and comparing pMOEAs is a difficult task. Numerous factors must be considered. The MOEA community has largely agreed on utilizing a variety of problems from two proposed test suites [31, 44, 52, 184, 209]. These researchers are pushing each other to extend the state-of-the-art in this area and their efforts are continuing. Many of these same researchers agree these test suites, composed of MOPs exhibiting various genotypical and phenotypical characteristics, are useful in making generalized statements about MOEA performance in solving MOPs. It is also useful to note that if the test's goal is to illustrate pMOEA performance on a specific type or class of problem, then the test suite should only be composed of MOPs from that class. These test guidelines and objectives are also applicable in pMOEA comparisons.

Based on the preceding discussion several MOEA functions appear appropriate for initial pMOEA experimentation. Initially selected are an unconstrained MOP, MOP 4 (see Chapter IV, Section 4.1), and a constrained MOP, the MMOKP (see Chapter IV, Section 4.4.1), in order to evaluate the performance of the pMOEA when applied to problems that are difficult for MOEAs to solve.

7.3.5 pMOEA Metric/Parameter Issues. The need to compare various pMOEA approaches exists as well as the need to evaluate the performance of a pMOEA applied to a particular class of MOP. There is no one best metric to use but a combination of metrics is useful in evaluating the performance of an pMOEA. A major focus of some MOEA researchers' efforts is the identification of suitable metrics to evaluate the efficiency and effectiveness of MOEAs. A number of authors [31, 44, 113, 114, 184, 190, 207, 210] have proposed the use of specific metrics to analyze and compare general MOEA performance. These metrics are also applicable to evaluating the performance of pMOEAs. While in isolation, these metrics do not depict the overall performance of an pMOEA, but in combination, they can. The problem with utilizing a single metric to evaluate the performance of an pMOEA is that this metric typically maps the large number of points found on the

Pareto front (PF_{known}) to a single point, the metric value. This mapping is lossy in that a single value cannot accurately evaluate the entire performance of an pMOEA.

A large number of metrics exist for evaluating MOEAs and these metrics naturally extend to pMOEAs. These include relative and exact metrics dependent upon knowledge of PF_{true}. These metrics analyze the cardinality of PF_{known} and its associated distribution. Certain metrics may be identified as better than others in specific applications. Chapter III, Section 3.2 presents a more detailed discussion of various MOEA metrics from the literature.

Various metrics can be employed to determine a given pMOEA's relative performance. As the known pMOEA publications give no recommendations on what specific metrics to use in analyzing pMOEA performance or in comparing various pMOEA instantiations, MOEA metrics are obviously useful and relevant. Additional metrics may provide a more complete evaluation of the parallel performance of the pMOEA. Several general metrics already exist and are in common use by parallel computation researchers to aid in measuring and judging some parallel implementation's performance.

For example, the speedup metric (S) captures the relative benefit of solving a problem in parallel. T_s then denotes the execution time of the fastest known serial (one-processor) implementation; note Kumar's assumption that one may not know what the fastest implementation truly is [119]. Parallel run time is denoted by T_p and is then the execution time for a given parallel implementation assuming identical processors and identical input sizes (search space size or number of evaluations). The speedup metric is defined as $S = T_s/T_p$ [119:p. 118].

Scaled speedup, efficiency, cost, scalability and isoefficiency are also important metrics in evaluating parallel performance. Scaled speedup is defined as the speedup obtained when the size of the given problem is increased linearly with respect to the number of processors [119:p. 144]. Efficiency is a measurement of the fraction of time a processor is conducting work; it is calculated as the ratio of speedup over the number of processors (p) [119:p. 120]. A pMOEA's cost for solving some MOP is defined as the product of parallel execution time and the number of processors used [119:p. 120]. A cost-optimal pMOEA

then occurs when the parallel cost of solving an MOP is proportional to the execution time of the fastest-known sequential algorithm on a single processor [119:p. 120]. Scalability is used to analyze the pMOEA's ability to increase its speedup as the number of processors available is increased [119:p. 128]. Finally, the isoefficiency metric is a determination of how well a pMOEA can maintain a constant efficiency and increasing speedup as the number of employed processors increase [119:p. 131]. All of these metrics are important when analyzing overall pMOEA performance. The pMOEA community should incorporate these metrics in their analyses to support any conclusions that a pMOEA achieves increased statistical performance over a serial MOEA. Other relevant performance metrics and evaluation techniques might be found in the parallel and distributed programming literature.

Looking to the existing pMOEA publications for metric recommendations yields little information. Few of the papers report performing formal parallel experiments and analyses and only two papers present a speedup graph charting their results [83, 206]. Better understanding of the various paradigms' effectiveness and efficiency can only come through "good" experimentation and analysis of results.

In analyzing pMOEA results, metrics specific to the parallel paradigm implemented may be required. For instance, the speedup metric applies to any of the paradigms, but if an island paradigm is used, one may wish to measure each deme's performance, the migration scheme's effectiveness, or some other measure specific to the paradigm. The process of pMOEA migration and replacement strategy selection is largely unaddressed and hence is not analyzed. Although Cantú-Paz presents a thorough analysis of several migration and replacement schemes for parallel SOEAs, he does not address their extension to the multiobjective arena [20]; effective SOEA/pMOEA schemes may be radically different.

As the pMOEA field matures, so must the reporting measures used by researchers. In analyzing the current pMOEA publications, few detail the parameter values used (crossover and mutation rate, selection methodology, etc.). Only a limited number of papers present this information [89, 105]. Table 7.2 proposes some key computational reporting parameters for those researchers interested in exploring pMOEAs. Although *all* are perhaps not always relevant, some subset of these parameters is certainly necessary for a fuller

Table 7.2 Key Parallel MOEA Characteristics

Key Characteristic	Description
Computer	Machine name (e.g., IBM SP3)
CPU	CPU Type(s) (e.g., Pentium IV 2.4 GHz)
# Nodes	Number of CPUs (e.g., 8, 16, 32, . . . , 256, . . .)
Memory	Memory per Machine (e.g., 256 GB)
Operating System	Name/Version (e.g., Red Hat Linux v7.3)
Communication Network	Network (e.g., Ethernet, Fast Ethernet, Gigabit Ethernet)
Communication Library	Library (e.g., MPICH v1.12)

understanding of the particular pMOEA paradigm selected (i.e., master-slave, island, or diffusion), for repeatability of previous results, and for comparison purposes.

The reporting of an MOEA's speedup may lead to some controversy as superlinear speedups may be presented. Section 7.2.2.4 of this chapter discusses this issue in more detail. A researcher must address superlinear speedups if the situation is encountered and be able to explain how such a speedup is achieved. In many cases, a superlinear speedup may be attributed to the fact that identical areas of the search space may not have been searched by both the MOEA and the pMOEA.

7.4 pMOEA Development

Based upon previous discussions, the process of pMOEA development is now addressed in some detail, whether one is designing a pMOEA from scratch or using an existing MOEA as a template. Initially a researcher must determine if a pMOEA is suitable to apply to the particular MOP and determine if conditions exist in which a pMOEA may perform "better" than other alternatives.

Given that a pMOEA then appears useful, the engineering-based algorithmic development process illustrated in Figure 7.8 is suggested [31, 194]. The figure presents a logical flow for an overall design strategy. The process involves a detailed study of the problem domain, the algorithm domain, and an integration of the two domains.

Studying the problem domain involves examining how problem data is input and output, the various constraints on solution states and the set of candidate solutions. If

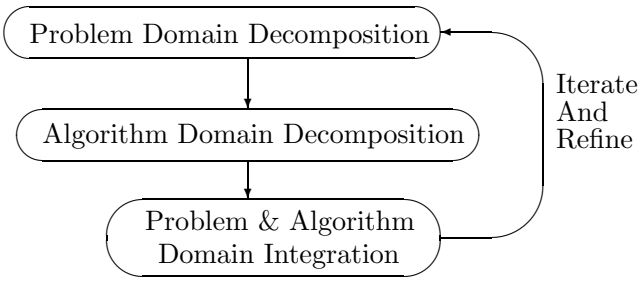

Figure 7.8 Problem-Algorithm Domain Interaction [31, 194]

applicable, one must also consider whether or not candidate solutions meet constraint criteria (feasibility), extracting one or more promising feasible candidate solutions (selection), defining acceptable solutions (the solution set), and whether or not members of that set reflect the selected optimization criteria (the objective functions) [31].

Analysis of the algorithm domain is concerned with input set(s) of candidates, output set(s) of solutions, and partial-solution set(s). One should determine if the MOP is most suitably attempted to be solved by a pMOEA or another approach. Any existing algorithm (with single or multiple objective functions) already being used to solve the MOP should also be considered. Additionally, one must analyze the various pMOEA operators to determine those that appear to offer the best performance on the particular class of MOP.

Integration of the two domains must then follow and the end result is a direct mapping of low-level designs to some chosen compiler language. As the possibility for error exists in any implemented code, it should be thoroughly debugged. Refinements of the code may be required in order to reach an efficient and effective implementation, which is the overall goal.

In implementing an pMOEA, one must consider the data structure to be used. Typically it is also important to consider the parallel paradigm to be employed and when/where that data is needed during execution. This is integral to implementing an efficient and effective pMOEA. The parallel system's architecture and communications backbone are also of concern as they can greatly impact the data transfer rates and may be more efficiently used by one pMOEA paradigm over another. Some pMOEAs may benefit from using one of the available static or dynamic processor scheduling and load balancing techniques,

e.g. [60, 119]. As pMOEAs are increasingly applied to real-world scientific and engineering problems where fitness function calculation time is large, these scheduling heuristics become more and more important [18].

A researcher is also well advised to consider the platform they plan to use prior to implementing the pMOEA. This is important as it may drive the use of one programming or parallel library over another. Additionally, one may find that their system of choice may not be the best system to use. An example of this is illustrated when examining the current utilization trends of a targeted parallel system for pMOEA execution. Many high performance computing centers limit the number of processors or amount of wall-clock time a particular process may consume, dependent on the number of users utilizing the system and the priority of the currently running projects. Hence a researcher may have more resources or processing time available on a local cluster as compared to a supercomputer. Even though a supercomputing facility may have a larger quantity of resources and more processing power, there may be an order of magnitude more users that are currently using the resources as compared to a slower local cluster. Therefore one may realize a quicker turn around time from submission of jobs to the local cluster than the supercomputer even though the supercomputer uses less wall clock time. Additionally, researchers must determine whether or not their code must be portable and hence they may be required to use a "generic" parallel library versus a specific library.

In general researchers have four major development options for implementing a pMOEA:

1. **Parallelize an Existing MOEA.** If researchers have access to working MOEA code, an understanding of algorithmic details and parallel processing techniques, parallelizing the existing MOEA may be the best option. The process of efficiently parallelizing an MOEA involves knowledge of parallel routines/libraries and the major pMOEA paradigms.

2. **Use Existing pMOEA Code.** Researchers may be interested in utilizing existing pMOEA code in order to reduce algorithm development time or a specific pMOEA may already be identified as performing well on a particular class of MOP. One must

understand the problem and algorithm to make the determination that this is a good choice.

3. **Develop a New pMOEA.** Developing a "new" pMOEA presents the possibility of implementing an pMOEA with increased efficiency and effectiveness over other pMOEAs. Developing a new pMOEA also allows for incorporating new search concepts and new operators; or combining existing ones in new ways.

4. **Extend an Existing Parallel EA.** Extending an existing, proven parallel EA to a pMOEA may appear relatively easy, but in actuality, this process may be difficult. Modifying a parallel EA to use additional objective functions may not be difficult; however, modifying migration and replacement schemes as well as other EVOPs are not trivial tasks.

7.4.1 pMOEA Implementation Issues.

7.4.1.1 Master-Slave Implementation. The master-slave paradigm may be the simplest to understand. Generally used for partitioning fitness function evaluations among several processors (see Figure 7.2), this paradigm appears especially useful for computationally expensive evaluations (such as those often found in CFD or CEM MOEA problems and where parallel codes already exist [126, 127, 128, 155]). Parts of the evaluations may also be performed by separate slave processors (i.e., sub-problems) with final values being coordinated/computed by the master processor.

A researcher must first determine that the fitness function calculations are suitably complex, which can be done through comparing the wall-clock time (estimated or actual) required to compute fitness function values for the entire population against the communication time between processors. The main issue is ensuring that each processor receives enough "work" so that communication costs do not overwhelm computational costs and that an improvement is realized over the execution of the serial MOEA. Using Equation (7.1), this is denoted as ensuring in a given generation that

$$PT_c << \frac{n \sum_{i=1}^{k} T_{f_i}}{P} \ . \tag{7.6}$$

In other words, additional processors are useful only to the point that each requires more computational time versus communications time. Figure 7.9 illustrates this situation in a two-node island implementation with a number of farming (slave) nodes used to evaluate the fitness functions. The results of a parallel implementation of the fmGA (pfmGA), as applied to the Protein Structure Prediction Problem, are used to generate Figure 7.9 [41]. Figure 7.9 illustrates a great improvement in processing time when increasing the number of farming nodes from zero to one. This is due to the large amount of computational effort necessary to complete the fitness function calculations. As the number of farming nodes is increased to two and four, minor improvements are made as the communication time begins to approach the computational time. However, increasing the farming nodes from four to eight is detrimental and results in a worse execution time than just utilizing a single farming node. With the addition of more slave nodes, each slave evaluates fewer population members, the computation time per slave drops and the communication or overhead time becomes a larger percentage of the total execution time. At this point the communication time has overwhelmed the computational time and hence worse timing results are realized. The last case results in an underutilization of the farming or slave nodes.

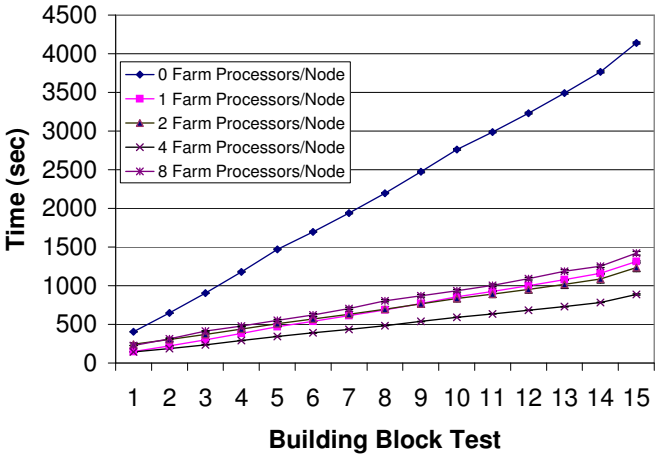

Figure 7.9 Master-Slave Processor Comparison [41]

Generational synchronization should not be a large issue in this paradigm as the typical approach evenly distributes population members across the slaves. Each receives

the necessary chromosomes along with any other necessary parameters for evaluation. The fitness functions are returned to the master node once the slave completes all of the evaluations. This results in relatively small data sets to pass between the master and slaves. Synchronization may be an issue in heterogenous implementations or if some chromosomes require varying evaluation times.

7.4.1.2 Island Implementation. The island paradigm can be implemented in a number of ways but is generally used to execute several MOEAs simultaneously (see Section 7.2.2.2). This is useful for concurrently examining different areas of the search space or a larger amount of the search space in the same wall-clock time as the serial implementation. This also allows for investigating the effects of varied algorithmic parameters (e.g., population size, EVOPs), and to determine how the selected migration policy effects the MOP solution process.

One extreme in terms of communications is represented by separate MOEAs simultaneously executing on some number of processors with no communication occurring between the nodes until each MOEA terminates the search process and its respective $PF_{known}\left(^{p}\right)$ set is transmitted to process "0"; these $PF_{known}\left(^{p}\right)$ sets are then combined to arrive at the final known front generated, PF_{known}. The other extreme occurs when each processor performs a one-to-all broadcast of each population member's chromosome and fitness values as each generation completes.

A more typical implementation involves periodic migration of individuals among selected processors throughout the algorithm execution [20]. This can conceptually be viewed as one overall pMOEA population divided among numerous processors. The same size population as some serial MOEA can reside on each processor to keep pMOEA run time approximately the same. This is not a requirement, however, and population size can be set to whatever number is desired by the researcher. The main concept around which the island paradigm revolves is the periodic migration of population members between selected islands to mix the best BBs found and provide some diversity. Migration and replacement schemes have been discussed in numerous parallel EA papers; the interested reader is referred to Cantú-Paz' thorough study for more information [20].

Implementing pMOEA migration and replacement strategies is far more complex than for parallel EA instantiations, which may explain why they are not currently discussed in any detail in the known literature. While these topics may seem conceptually easy, in actuality a number of issues arise making them quite different for pMOEAs. A discussion of all possible SOEA schemes is not presented due to the large number of possibilities and since there are MOEA-specific issues making some irrelevant.

Numerous pMOEA migration strategies and subsequent replacement strategies are available. A variety of these are structurally different from one another and are classified as separate schemes. Table 7.3 presents a concise listing of new, innovative migration schemes holding the greatest potential and some of their attributes; analogous replacement schemes are presented later in Table 7.4.

Table 7.3 pMOEA Migration Schemes

	Scheme	**Attributes**	**Selection Pressure**
Non-Uniform	Elitist (random)	Migrate a random sample of x % of individuals from $PF_{current}\binom{p}{t_bm_x}$ points or of each ranked front	Relatively High
	Elitist (niching)	Migrate a x % "uniform" distribution of individuals that are members of $PF_{current}\binom{p}{t_bm_x}$	High
	Elitist (front)	Migrate the entire $PF_{current}\binom{p}{t_bm_x}$ set	High
	Elitist (front+)	Migrate the entire $PF_{current}\binom{p}{t_bm_x}$ set plus x randomly chosen population members or x population members from the ranked fronts	High
Uniform	Random	Migrate x randomly selected population members	Low
	Elitist (random)	Migrate x individuals from $PF_{current}\binom{p}{t_bm_x}$, choosing some randomly or from the ranked Pareto fronts if necessary	Relatively High
	Elitist (niching)	Migrate x "uniformly" distributed individuals from $PF_{current}\binom{p}{t_bm_x}$, choosing individuals from the ranked Pareto fronts if necessary	High

Note that both migration and replacement focus on each islands' current Pareto front ($PF_{current}\binom{p}{t}$), as identifying an approximation of PF^*. As each element of $PF_{current}\binom{p}{t}$ contains the local optima and the good BBs, these elements are the focus of exchange amongst processors. Note these strategies apply to each processor used.

A number of parameters may be specified in an island paradigm (see Section 7.2.1). Two parameters of high interest are the number of population members to migrate (x) and the destination processor(s) (p_{dest}) to which the x members are migrated. The number of

members to migrate is of interest as migrating too many individuals negatively impacts the overall communication cost. Additionally, migrating too many individuals may force receiving processors to converge to identical solution sets. A tradeoff must be made between exploration and exploitation, and it is typical to migrate a low percentage of the total number of members in the population.

The selection of the destination processor(s) for immigration is also of high interest. A total broadcast involves every processor migrating individuals to every other processor but can potentially overwhelm the communication backbone as well as force all processors to search the same area of phenotype space. Conversely, not employing migration may defeat the purpose of implementing a pMOEA. Not surprisingly, just as is the case with several EA parameters, there currently exists no "best" method for choice of p_{dest}.

Assume that a given processor is assigned a neighborhood of size N. At the specified migration interval, each processor migrates selected individuals to its N "closest" neighbors. For simplicity, each processor communicates with its "closest" neighbors assuming a grid type architecture. In reality, the neighborhood can contain any of the processors in the system, and the neighborhood can be static as well as a dynamically changing list of neighbors of any structure. As processors might be arranged in various logical topologies (e.g., ring, mesh, hypercube, etc.) or arranged according to a specific physical hardware configuration, the definition of the "closest" members is left open.

The migration schemes are decomposed into two categories, non-uniform and uniform. The *uniform migration schemes* are labelled as such since a constant number, x, of the population members are migrated with each migration event whereas the *non-uniform migration schemes* may involve a migration of a varying number of members with each migration event. The uniform schemes are advantageous in that the amount of communication incurred by the algorithm is consistent and predictable but the non-uniform, unpredictable communications has advantages in relation to anticipated efficiency and effectiveness of the pMOEA.

The simplest migration scheme is a purely *random* one. This involves randomly selecting x population members from a given processor for migration to the N neigh-

bors. This scheme provides low selection pressure as no guarantee exists that members of $(PF_{current}\left({}^{p}_{t_bm_x}\right))$ are migrated.

Having dealt with the obvious choice, more thought is required to define and implement effective migration strategies. SOEA optimization offers the choice of choosing the top x population members or a top x percentage of the population for migration; these methods are called elitist [20]. Migrating the best individuals has an associated high selection pressure as this method could cause other processors to converge to identical solution sets, especially if the destination processor lacks any "better" members. Selection pressure is defined as the ratio of the probability that the best member of the population is selected over the probability that an average member of the population is selected [74].

The concept of "best" takes on a different meaning in MOEAs. There generally is no one best individual unless only one member exists on the Pareto front. Selecting the top x or $x\%$ of the population for migration is somewhat meaningless within an MOEA as all the members of $PF_{current}(t)$ are equally optimal in terms of Pareto dominance criteria. A simple way of handling migration is to randomly choose x members from $PF_{current}\left({}^{p}_{t_bm_x}\right))$ and migrating them where desired. This is an *elitist random* migration method as a portion of the best members are selected for migration. However, this method has the potential to migrate members in a concentrated area of $PF_{current}\left({}^{p}_{t_bm_x}\right)$ and hence not provide a good representation of the current optimum or all of the good BBs. Additionally, a problem arises if the cardinality of the number of solutions to migrate is larger than the number of solutions within the current Pareto front set $(|x| > |PF_{current}\left({}^{p}_{t_bm_x}\right)|)$.

In this case one can send fewer solutions than are requested, by sending all of the members of $PF_{current}\left({}^{p}_{t_bm_x}\right)$, migrate $PF_{current}\left({}^{p}_{t_bm_x}\right)$ plus a random selection of other population members from the entire population, or migrate all of the members of $PF_{current}\left({}^{p}_{t_bm_x}\right)$ plus a random choice of members existing on the next ranked Pareto front until x members are found. The ranked Pareto fronts can be determined using one of the methods discussed in Chapter II, Section 2.9. Migrating members from multiple fronts is anticipated to have the quickest convergence due to the method's "strong" elitism. However, a possible disadvantage lies in the computational overhead required to rank the Pareto fronts until x members are identified. Ranking adds to the complexity of migration

as determining each rank using Pareto optimality is generally of order $O(k * n^2)$ [31]. For the proposed elitist random method, if all x members cannot be found in $PF_{current}\binom{p}{t_bm_x}$, the rest are selected either randomly or from the remaining ranked Pareto fronts.

Another possible elitist migration method, *elitist niching*, uses niching in conjunction with individual selection. As opposed to randomly selecting x individuals, niching allows for the possibility of migrating a more uniform distribution of Pareto front members and BBs, not just a localized one from $PF_{current}\binom{p}{t_bm_x}$. This is accomplished by selecting a "uniform" distribution of x individuals from the local Pareto front $PF_{current}\binom{p}{t_bm_x}$. If $|x| > |PF_{current}\binom{p}{t_bm_x}|$, the remaining members are selected from the ranked Pareto fronts in the same manner.

Another straight forward and conceptually "easy" elitist migration method is migrating the entire local Pareto front ($PF_{current}\binom{p}{t_bm_x}$), *elitist front*, from each process. This is a non-uniform migration method as there is no way to determine $PF_{current}(t)$'s size a priori as it varies both by process and by generation. This method has the greatest selection pressure. A modified version of this scheme (*elitist front +*) involves migrating the entire local Pareto front, and additionally migrating x members randomly chosen or selected from the ranked fronts. This method has the advantage of migrating an elitist set of individuals and some additional individuals to maintain diversity. The disadvantage of this method is the possibility of high communications costs as $PF_{current}\binom{p}{t_bm_x}$ can vary in cardinality. In an attempt to prevent premature convergence, one potential solution is to use the elitist method with niching (*elitist niching*) which selects a "uniform" distribution of individuals across $PF_{current}\binom{p}{t_bm_x}$. A random choice of Pareto front members (*elitist random*) may possibly choose points from a local area of the front but has the least processing overhead as niching calculations do not need to be conducted. Additionally, niching complexity is high due to the fact some density distribution measure must be calculated for all front members in order to select an "even" distribution. This complexity may prevent the use of this scheme. However, its potential benefits may outweigh the additional computational overhead.

Regarding the random and niching methods (either uniform or non-uniform), note that randomly selecting individuals has the least processing overhead as niching calcula-

Table 7.4 pMOEA Replacement Schemes

Scheme	**Attributes**	**Selection Pressure**
Random	Randomly replace x population members	Low
None	No replacement, yields an increasing population size	Medium
Elitist (random)	Maintain $PF_{current}\left(\begin{smallmatrix}p\\t_bm_x\end{smallmatrix}\right)$ and randomly replace members that are $\notin PF_{current}\left(\begin{smallmatrix}p\\t_bm_x\end{smallmatrix}\right)$	Relatively High
Elitist (ranking)	Rank all of the Pareto Fronts and replace members from the "worst" ranked fronts with those received	High
Elitist (100% ranking)	Combine the migrated members with the current population, rank all of the Pareto Fronts and remove members from the "worst" ranked fronts	High

tions are not necessary. Niching complexity is high since some density distribution measure must be calculated for all local Pareto front members in order to select a "uniform" distribution. Although this complexity may preclude a niching scheme's implementation, its potential benefits may outweigh the additional computational overhead.

As multiple migration schemes exist, so also do replacement schemes. Replacement schemes also differ from those employed in parallel SOEAs. In general, an MOEA replacement strategy assumes population size must remain constant each generation. Without this requirement, the population size on each processor can grow fairly quickly. Even a partial replacement scheme, where some number y ($y < x$) of population members are replaced with those migrated and the rest of the migrated members are added to the population, the overall population size still increases as would computational cost. The working assumption is that each processor's population size remains constant and therefore x population members are "thrown away" each migration interval. Table 7.4 presents a concise listing of replacement schemes holding the greatest potential and some of their attributes.

Again, assume each processor has a neighborhood of size N. At the specified replacement interval each processor replaces selected individuals with those received from its N neighbors. Just as it is for migration, the simplest replacement scheme is a *random* one in which immigrants randomly replace members in the target population. This scheme has the lowest selection pressure as no guarantee is made regarding the quality of solutions being replaced, similar to that presented in [20]. For example, a solution might not

be a member of the resulting Pareto front, $PF_{current}\left(\substack{p\\t_am_x}\right)$, on the receiving processor, yet could potentially replace a solution on that front. To actually determine that fact requires combining the receiving population and the immigrants, and then ranking this newly formed population. This situation needs to be addressed in an efficient manner but in order to find out the situations existence some overhead processing must be completed. However, if this situation is not dealt with, this may lead to replacing better individuals with worse individuals and over time this can lead to the population's divergence instead of convergence to good potential solution(s).

Replacement strategies differing from random schemes may lead to increased execution times. The issue of pMOEA replacement strategies really comes down to the issue of how to rank population members in an efficient manner in order to implement an "elitist" replacement strategy. Generally speaking, since all processors perform migration, $PF_{current}\left(\substack{p\\t_bm_x}\right)$ is already calculated. However, replacement then most likely changes $PF_{current}\left(\substack{p\\t_bm_x}\right)$ for each target processor to $P_{current}\left(\substack{p\\t_am_x}\right)$. In other words, no guarantee exists that $PF_{current}\left(\substack{p\\t_bm_x}\right) = P_{current}\left(\substack{p\\t_am_x}\right)$, where the subscript t_bm is prior to and t_am is after replacement. Thus, the question exists of whether to determine a combined front consisting of $PF_{current}\left(\substack{p\\t_bm_x}\right)$ with the received members or to keep members belonging to $PF_{current}\left(\substack{p\\t_bm_x}\right)$ and replace others. The following replacement schemes are all elitist as they rely on some type of ranking or replacement to ensure the "best" members of the population(s) are kept for future generations.

One possible strategy is to keep all the members of $P_{current}\left(\substack{p\\t_bm_x}\right)$, determining which of the remaining population is not Pareto optimal with respect to the immigrants, and randomly replacing them with the immigrants (*elitist random*). A disadvantage is the computation of solutions' objective vectors that are dominated by the immigrants'. Additionally, what if this set, termed DOM, is smaller than the immigrant set, i.e., $|DOM| < -P_{current}\left(\substack{p\\t_am_x}\right)|$? This method does not guarantee the "worst" members are removed but does ensure all members in $P_{current}\left(\substack{p\\t_bm_x}\right)$ are retained.

A method with increased pressure replaces members constituting the "worst" non-dominated front (*elitist ranking*) (as determined in the manner of the NSGA/NSGA-II) [46, 176]. This method has increased computational complexity when used as the

migration strategy, and as before, members must be randomly chosen from some front(s) if the number of members to be replaced does not evenly match the number of individuals represented by the replaced front(s). For example, assume a processor contains four separate fronts with $|F_0| = 12$, $|F_1| = 8$, $|F_2| = 6$ and $|F_3| = 4$. Unless $x = 4$, 10, 18, or 30, some random replacement(s) is then required. This method provides the greatest selection pressure as the "worst" members are continually removed from the population; however, this method likely incurs the greatest amount of resource consumption as computing the nondominated fronts may be time consuming. A slight modification of this method is the *elitist 100 % ranking* in which the received members are combined with the current population members and then ranked. The best population members are kept and the remaining members, from the worst fronts, are thrown away. This offers a slightly higher selection pressure.

pMOEAs may also employ various dynamic migration and replacement schemes. Dynamic methods have a possible advantage if one has some knowledge of the fitness landscape or about how the search should progress. The presented migration/replacement schemes may also be combined in multiple ways. The decision on which schemes to pair may most often be driven by their computational cost, but one must trade this cost off against the resulting effectiveness. Other EVOPs, such as mutation and crossover, can be defined *a priori* for each island [20] or can dynamically change [1]; different islands may also use static or dynamic settings.

As with many search algorithms a local search operation, like hill-climbing, may also be used. This may increase the quality of derived solutions (i.e., PF_{known}). The periodic use of local search operators throughout algorithm execution may be beneficial; it remains to the researcher to tune their implementation for the particular application.

7.4.1.3 Diffusion Implementation. The diffusion paradigm appears the least discussed paradigm in the pEA/pMOEA literature [20, 31]. This is possibly due to its associated complexity and potentially increased communication costs (as compared to the other paradigms; see Figure 7.4). This paradigm uses one conceptual population distributed across a number of processors, each typically holding only a few individuals.

A topological and neighborhood structure is imposed on the processors where EVOPs operate only within some specified neighborhood. Due to most EVOPs' serial nature much communication is necessary to execute them as they require knowledge of an entire neighborhood's population.

Although no proof is offered, it is suggested the diffusion paradigm allows formation of local niches. Each represents different ways of trading off the k objectives. However, note that few fine-grained pMOEAs are found in the literature [31], [20:p.140]. Much work remains to be accomplished before a "good" understanding of these algorithms' strengths and limitations are known. This paradigm appears to be a good choice when paired with shared memory systems and symmetric multiprocessors (SMP) systems (systems containing multiple homogeneous processors within one system unit).

A basic diffusion paradigm includes the pMOEA executing on a number of processors in parallel, where the communications are restricted to occur only between processors in a specified neighborhood. The communications can be based on a static neighborhood structure or a dynamically changing structure. Whatever the case, the neighborhoods overlap to some degree to allow for a slow diffusion of the "good" population members throughout the entire processor pool. The only constraint is that the possibility must exist for a population member to reach all of the possible destination processors in the pool over a reasonable number of generations. The destination processors must be reachable within a small number of generations in reference to the total number of generations during pMOEA execution. Like the other paradigms, this process continues to execute until some specified stopping criteria.

Shared memory systems are ideal for diffusion pMOEA implementations as each processor has access to the entire population through a very high speed backplane and is likely orders of magnitude faster than any current network communications structure. The disadvantage is that shared memory systems typically require using propriety communication calls (OpenMP) not portable to other environments in order to achieve the desired level of performance. This may discourage researchers wanting efficient, portable pMOEA codes.

Implementing a diffusion pMOEA on a shared memory system is fairly straightforward. Using an SMP-based cluster is a bit more complicated and should likely use a neighborhood size equal to the number of resident processors in each SMP unit. This allows one to realize the superior communications backbone within an SMP unit. When utilizing a cluster of SMP nodes, where multiple SMP nodes are networked together and used in the execution of the pMOEA, the communications becomes more complicated. This case is similar to that of utilizing a heterogeneous system where the communication times differ from unit to unit as the communications times between processors within an SMP unit are much quicker then those between different SMP units. One possible implementation involves a hierarchial paradigm as a neighborhood structure is imposed on the overall cluster of workstations, an island paradigm type of communication mechanism is used between SMP units and a diffusion paradigm is used within any given SMP unit. Other variants of this general system design are certainly possible. This example is provided only for an understanding of how one might implement an efficient diffusion pMOEA on a non-shared memory based system, however, the concentration of this research is on homogeneous systems.

One possible diffusion implementation operates by dividing the population into small groups of individuals and assigning those groups to different processors. Another implementation involves a diffusion paradigm operating as part of a hierarchial pMOEA. With a neighborhood structure imposed on the overall SMP cluster, an island paradigm treats each SMP unit as a deme, and a diffusion paradigm operates within any given SMP unit. Other variants of this general system design are certainly possible. These examples are provided only for an understanding of how one might implement an efficient diffusion pMOEA on a non-shared memory based system.

7.4.1.4 Parallel Niching. As discussed by numerous researchers, the use of niching is highly suggested in any (p)MOEA; how best to implement parallelized niching is a currently open question with no single answer (see Chapter II, Section 2.10). Parallel niching is addressed as it is another area of discussion lacking in the current literature,

even though the concept is acknowledged as being of high importance in obtaining high-performance (p)MOEAs.

Niching, crowding, clustering and σ_{share} are terms widely used in the MOEA community when discussing the concept of finding a "good" distribution of points along the Pareto front or among the Pareto optimal solution set. Many methods exist for accomplishing MOEA niching and are discussed in detail in Chapter II, Section 2.10. The concept of parallel niching, however, is fairly new and is not adequately explored or discussed in the current literature. The underlying question is then how to parallelize niching and what, if anything, is gained by doing so.

The typical MOEA niching implementation occurs in the phenotype domain. Niching in either the phenotype or genotype domain involves analyzing some Pareto front or Pareto optimal solution set and attempting to achieve uniform knowledge of the set, i.e., avoiding a clustering of points in any one particular area. The phenotype domain is predominately used since most MOEA researchers desire a uniform distribution of points along the (known) Pareto front. Niching involves removing points from a given population that lie within some user-defined distance of others. New points are then generated in the hopes of filling in an unfilled region, i.e., gaining knowledge of a previously or unsatisfactorily unexplored area. Note that points cannot be generated directly by a process of analyzing an area of the front, selecting a region of the front that is unfilled, specifying objective function values in that region and then performing a reverse mapping to find the corresponding decision variable values. One typically does not know the reverse mapping for objective functions of any difficulty. Niching in the genotype domain involves analyzing some Pareto optimal solution set and then generating new points based on movements away from identified points, i.e., clustering.

Since niching is basically a serial process, it is difficult to efficiently parallelize. Note that a master-slave pMOEA can perform niching in the usual manner but the other paradigms are different. An island pMOEA offers several niching options. For example, a separate niching operator can execute on each island. There may be additional utility in executing a variety of niching operators across the different processors. These methods do not require any extra communication; however, one may well wish to transmit

local front info between (some set of) processors in an attempt to niche based on global knowledge. Again, note that pMOEA niching might be effective in either (or both) the genotype/phenotype domains.

Another parallel niching possibility divides the known front into a number of static regions equal to the number of available processors. After some number of generations, generate the currently known global front (by combining the current known fronts from each processor) and then redistribute the global population according to the restricted search regions for each processor. The goal is for each processor to concentrate their search in a specific region of the landscape. This method appears to be fairly straight forward but in fact that is only true for those members on $PF_{current}(t)$. The dominated members do not map as easily to the regional structure imposed and require a ranking method to be used in order to determine the mapping of each processor. A ranking of the entire population allows for a determination of which region each dominated member belongs. Additionally, this method requires processing and communication overhead as the entire population of each processor is redistributed amongst the processors. Another potential issue encountered in this scheme is the possibility for all of the combined population members to be allocated to a single processor with the remaining processors receiving zero population members if a static regional scheme is specified a priori. A simple solution to this is to allocate the population members based on where they are currently in the landscape, i.e., use a dynamically changing regional structure. This method may be more useful when employing genotypic-based niching. An interesting way of handling this is to partition the genotype space into regions and allow the processors assigned to regions to explore +/- ϵ amount outside of the region. This allows for the processors to find solutions outside of the currently known areas of goodness.

In order to minimize the communications required to implement parallel niching, neighborhood constraints can be imposed on the generation of new population members. This means that a given processor is restricted to generate solutions within a particular region. Each processor then must continue generating solutions until enough are generated in the specified region, throwing the others away. This means that the generation of specific areas of the front to analyze is restricted to the processors within a given neigh-

borhood, similar to the concept of the diffusion paradigm. This implementation effectively reduces the overall communications that are necessary and allows for more of a "parallel" searching of various areas of the front but may increase the overall execution time as each processor must continue generating solutions until it generates the specified number in the region. Another viable method reducing communications overhead is to communicate only every epoch (one epoch corresponds to a specified number of generations) thereby reducing overall communications load yet still achieving global niching. Dynamic niching is another promising scheme involving creating variable-sized niches (possibly) changing over time. Additional constraints can be imposed so as to focus more on exploring or exploiting during certain pMOEA phases.

When utilizing parallel niching in a pMOEA, one must balance requirements for a satisfactory representation of the Pareto front, the Pareto optimal solution set, and the time required to execute the algorithm. Parallel niching may improve pMOEA performance but may also overwhelm the communications backbone if not implemented well. The concepts presented can be applied in either the island or diffusion paradigms with some forethought. As parallel niching can require a large amount of communications, the underlying hardware and communications backbone must be considered when determining how to implement parallel niching. The best choice for implementation may be the use of a local niching operator or niching operators using different parameter values across the processors. On the other hand, especially when implemented on some type of shared memory system, the diffusion paradigm appears well-suited to handle a parallel niching operation where each processor has fast access to the entire population and can thus perform global niching fairly efficiently.

7.4.1.5 Parallel Archiving. Another area of great interest is that of archiving strategies. This is an area that has the potential to increase the efficiency of a pMOEA if intelligently addressed. The concept of archiving is to maintain a subpopulation or an external archive of the "best" found members from the pMOEA. In a pMOEA the issue arises of how to maintain the archive, how many to maintain, and how often to communicate the members between the archives. One method of archiving in a pMOEA is to

have each processor maintain its own archive and at the conclusion of the search process have one of the processors combine the archives and present the final solution to the user. More complicated strategies exist and may increase the effectiveness of the pMOEA, dependent on its particular use of the archive. For those MOEAs that maintain an archive and actively use that archive throughout the search process, a communication between the archives may be beneficial. A potential issue lies in combining the archives each generation which may lead to pre-mature convergence of each processor. In terms of the actual implementation, synchronization is required to ensure that multiple processors do not try and update the archive at the same time. This may increase the execution time of the algorithm, but in certain MOEAs, it has the potential to increase the effectiveness.

7.4.2 Parallel MOEA Theory. In the serial EA world, many researchers have published theories of EAs regarding a variety of characteristics. Such theories encompass EA characteristics such as representations, operators, algorithm convergence, equivalence, etc. For parallel EAs, some extensions have been made regarding population sizing [20]. Specific theories associated with diffusion and island paradigms relating to selection pressure and neighborhood size exist for EAs . For pMOEAs, little has been done except for the population sizing work presented in Chapter VIII. The development of formal modeling insight as to the impact of pMOEA operators and parameter values as well as the impact of computational architectures across classes of MOPs is useful.

7.5 A "Generic" pMOEA

The previous sections discuss major issues in pMOEA development and testing. A case study applying these concepts to a working serial MOEA and a thorough analysis of the results is presented. In particular, the creation of a generic pMOEA (in that it can be executed in a master-slave, island, or diffusion mode) is described as evolved from the working serial MOEA. Note this generic pMOEA's evolution is an excellent example of the design strategy suggested in Section 7.4.

Pseudocode for a generic pMOEA implementation of each of the parallel paradigms is presented. This pseudocode is meant to provide insight into creating an pMOEA using any

```
Randomly Generate The Population
    Randomly Generate Population Of Size P On Processor 0
Evaluate Each Population Member's Fitness
    Send P/n Population Members To Each Processor From Process 0
    Each Processor Conducts k Fitness Evaluations For Each Of The P/n Population Members
    Each Processor Sends k * P/n Fitness Values To Process 0
    Process 0 Determines PF_current (t), Updates PF_known And Assigns Rank If Necessary
Perform Clustering / Niching / Crowding on Processor 0
Execute Evolutionary Operators (Crossover, Mutation) on Processor 0
Evaluate the New Populations' Fitness
    Send P/n Population Members To Each Processor From Process 0
    Each Processor Conducts k Fitness Evaluations For Each Of The P/n Population Members
    Each Processor Sends k * P/n Fitness Values To Process 0
    Process 0 Determines PF_current (t), Updates PF_known And Assigns Rank If Necessary
Conduct Selection on Processor 0
Repeat Until Termination Criteria is Met
Conduct Local Search on Processor 0 if Specified
Processor 0 Generates and Presents PF_known as the Solution
```

Figure 7.10 Generic Master-Slave pMOEA Pseudocode

of the three aforementioned paradigms. This pseudocode is at a fairly high level as exact implementation details can vary based on one's selection of specific EVOPs, migration and replacement schemes, hardware, parallel libraries, etc.

```
Randomly Generate The Population
    Randomly Generate Population Of Size P' On Each Processor
Evaluate Each Population Member's Fitness
    Evaluate Each Population Member's Fitness on each processor
    Each Processor Determines PF_current (t), Updates PF_known And Assigns Rank If Necessary
Perform Clustering / Niching / Crowding on each Processor
Execute Evolutionary Operators (Crossover, Mutation) on each Processor
Evaluate Each Population Member's Fitness
    Evaluate Each Population Member's Fitness on each process
    Each Processor Determines PF_current (t), Updates PF_known And Assigns Rank If Necessary
Conduct Selection on each Processor
Migrate Population Members Among Processes According To Migration Scheme
Repeat Until Termination Criteria is Met
Conduct Local Search on each Processor if Specified
Processor 0 Combines and Presents PF_known as the Solution
```

Figure 7.11 Generic Island pMOEA Pseudocode

In the general pMOEA formulation, the following variables are of importance: p = process ID, n = number of processes, N = neighborhood size, P = population size in the master-slave paradigm, and P' is the population size in the island or diffusion paradigm, which may be P or $\frac{P}{n}$. We use the term process instead of processor since multiple processes can execute on a single processor; for example in a SMP platform. The

underlying assumption, made for simplicity and generality is that each process is running on its own dedicated processor and that the communications backbone is homogeneous in terms of the available bandwidth to each process. One may choose to implement an pMOEA on a heterogeneous cluster or on a system where the communications backbone does not provide equal amounts of bandwidth among the processors. In this case or modifications of it, an intelligent load balancing technique is necessary to achieve the same performance as a homogeneous technique yields.

```
Randomly Generate The Population
    Randomly Generate Population Of Size P' On Each Processor
Evaluate Each Population Member's Fitness
    Evaluate Each Population Member's Fitness on each processor
    Each Processor Determines PF_current(t), Updates PF_known And Assigns Rank If Necessary
Perform Clustering / Niching / Crowding on each Processor or Within the Neighborhood (N)
Execute Evolutionary Operators (Crossover, Mutation) Within Neighborhood (N) Processors
Evaluate Each Population Member's Fitness
    Evaluate Each Population Member's Fitness on each process
    Each Processor Determines PF_current(t), Updates PF_known And Assigns Rank If Necessary
Conduct Selection Within Neighborhood (N) Processors
Diffuse Population Members Among Processes According To Diffusion Scheme
Repeat Until Termination Criteria is Met
Conduct Local Search on each Processor if Specified
Processor 0 Combines and Presents PF_known as the Solution
```

Figure 7.12 Generic Diffusion pMOEA Pseudocode

7.5.1 Engineering the pMOMGA-II. Goldberg et al., indicated that too much attention was being paid to "neat" GA genotype encodings [78]. They proposed a scheme where genotypes can exhibit redundancy, over- and under- specification, and changing structure and length, believing this GA modification forms tighter and more useful building blocks than those formed by standard GAs. Their resultant mGA proved successful in optimizing *deceptive functions*; these functions mislead GA search toward some local optimum when the global optimum actually lies elsewhere [78]. The mGA initializes a population of building blocks via a deterministic process producing all possible building blocks of a specified size and thus deserves its reputation as a (potentially) computationally expensive algorithm. Additionally, note the mGA stored much of its information as unsigned integers and performed operations upon population members by bit shifting.

The fast messy GA (fmGA) was proposed to reduce mGA complexity via probabilistic building block initialization schemes creating a controlled number of building block clones of specified size. These clones are then filtered ensuring that (in a probabilistic sense) all desired building blocks exist in the initial population. Goldberg et al. claim this variant is as effective as the mGA but without the associated initialization bottleneck [77]. Later, the fmGA was expanded to create a parallel fmGA (pfmGA), where a number of issues associated with EA parallelization were considered, addressed and tested on a real-world problem, the Protein Structure Prediction Problem [36, 71, 134, 142]. Initial testing of the pfmGA was done on a Intel Paragon supercomputer. The processors were i860's and the operating system was the Paragon OSF/1 Release 1.0.4 which provided message passing capabilities. All of the tests used the same EVOP parameters, string length of 240 bits, cut probability = 2%, splice probability = 100%, 107 total generations and a population size of 32 members per processor [71]. This pfmGA used the MPI library to allow for interoperability on different parallel platforms; both an island and master-slave versions were implemented. This research was recently expanded to use newer IBM SP2 and SP3 supercomputers as well as a heterogeneous network of SUN Cluster of Workstations (COWS) and a heterogeneous Pile of PCs (PoPCs) in solving the protein structure prediction problem [36, 142]. These experiments obtained satisfactory speedup results; a large efficiency improvement was also noted as compared to the serial fmGA implementation. Note also that the pfmGA used character-based information storage thus using more memory.

Van Veldhuizen's original research investigated the use of building blocks in solving MOPs; he created the Multi-Objective mGA (MOMGA) for that purpose by heavily modifying the original mGA code [189]. Good performance resulted from applying the MOMGA against a validated MOP test function suite. The MOMGA also compared favorably to or outperformed three other well-engineered MOEA variants in wide use at the time. The MOMGA was not parallelized at this time because of data structure issues. The MPI communication library requires contiguous blocks of memory to operate in an efficient manner and the MOMGA used the mGA's original linked-list structure. The next algorithmic evolutionary step is presented in Chapter VI in which the MOMGA-II is cre-

ated by importing the probabilistic building block initialization scheme from the fmGA into the MOMGA. The MOMGA-II differs from the fmGA in that multiple objective functions are solved simultaneously and its selection mechanisms are based on the concepts of Pareto dominance and niching. The MOMGA-II is shown to have good effectiveness when compared to other MOEAs and also performs well when applied to real-world applications. These results motivated the parallelization of the MOMGA-II (or pMOMGA-II).

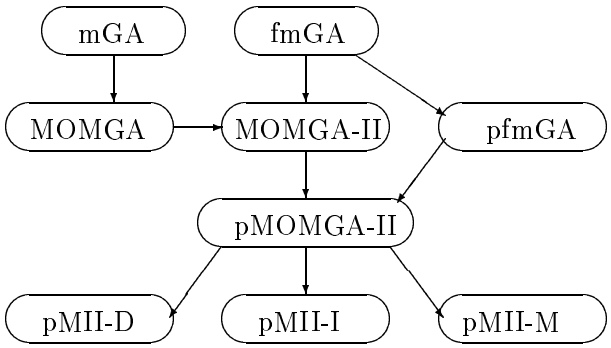

Figure 7.13 pMOEA Evolution

Note that during this algorithmic development process, several problems with the original mGA, fmGA and pfmGA codes were identified and corrected, and the resulting programs (MOMGA-II and pMOMGA-II) optimized. Both programs use unsigned integers to store population information, but population members are stored contiguously in memory so that MPI can be used to pass required information. These two codes are highly optimized as compared to their predecessors in that many extraneous function calls were removed, memory is allocated at program start (instead of reallocated during program execution), contiguous memory blocks are created for the entire population, and several memory leaks were plugged. Additionally, the numbers of decision variables/objective functions and operations with them, were generalized; the original codes used separate functions to handle the various cases. The (p)MOMGA-II development process is detailed to illustrate the iterative algorithmic development process depicted in Figure 7.13.

7.5.2 "Genericizing" a pMOEA. Increasing the MOMGA-II's efficiency and potential effectiveness are major goals in creating the pMOMGA-II. However, it is also

desired to thoroughly test the three parallel paradigms as applied to a single MOEA variant. In other words, use a basic MOEA to create three different parallel versions. Thus, the resulting algorithm is generic in the sense it may be used to implement any of three major pMOEA paradigms as well as a serial one, with an eye towards future expansion. This pMOEA is also generic in that it can execute on any parallel architecture hosting an ANSI 'C' standard compiler.

A generic pMOEA has an additional advantage in the experimental design and execution process, as algorithmic differences are minimized. Of course, any serious pMOEA testing must address not only MOEA-specific issues but must provide meaningful parallel statistics as well. This rigorous testing has not yet occurred with pMOEAs. A thorough analysis of the competing parallel paradigms, while also considering parallel performance issues, should give valuable insights into pMOEA effectiveness and efficiency.

7.6 pMOMGA-II Testing

Testing a pMOEA requires a thorough analysis of the algorithm and possible MOPs to use. One typically wants to test an MOEA on a particular class of problems and may base the choice of MOP test suite on MOPs containing similar characteristics to the real-world problem being solved. "Good" performance on test suite problems does not guarantee that the same level of performance can be obtained on the real-world problem, but it provides some level of confidence. This section presents testing of a pMOEA, the MOMGA-II, as applied to two MOPs, one a standard test suite problem, and the other a large scale constrained MOP.

7.6.1 Experimental Design. Based in part on the discussion presented in Section 7.3.4, two MOPs are selected for use in these experiments. A goal in these experiments is to determine the efficiency and effectiveness of differing parallel paradigms on different problems. Predictions of performance are later offered and results can then be compared. Thus, MOPs must be selected that appear suitable for each paradigm. In this chapter, generalized test suite MOPs are selected to make statements as to the performance of the various parallel paradigms when applied to MOPs in general.

The results of a master-slave implementation are dependent on the MOP being solved and the complexity of the fitness functions. Many references exist and results are presented illustrating the efficiency gains possible through a parallelization of the fitness functions [20, 142]. Results obtained in the single objective EA area apply here to MOEAs and hence this testing concentrates on migration and replacement schemes in pMOEAs. Figure 7.9 presents an example of the speedup achievable as applied to a specific problem.

For testing, two MOPs are chosen and executed using the pMOMGA-II. This is completed to illustrate the effect that migration and replacement strategies have on the overall effectiveness of a pMOEA. These two MOPs contain a different number of decision variables and different characteristics in terms of their associated genotype and phenotype spaces. The first MOP selected is Kursawe's MOP, labeled as MOP4 [184], a highly used MOP in comparing the performance of MOEAs presented in Chapter IV, Section 4.1. This MOP is chosen for its characteristic of having a disconnected P_{true} , and PF_{true} set, being scalable and nonuniform. The second MOP selected for testing is the combinatoric MMOKP MOP, as discussed in Chapter IV, Section 4.4.1. The MMOKP MOP is formulated to contain a large number of discrete, integer based decision variables. The 100 and 250 item instantiations are tested.

Since the objective of this testing focuses on island paradigm performance and the migration and replacement schemes, it is somewhat irrelevant as to which specific MOEA is chosen for testing. Since the MOMGA-II is intelligently engineered (see Chapter VI), it is parallelized and used for testing the associated concepts. In the testing presented, the following parameters are kept constant: cut probability 0.2%, splice probability 100%, and mutation 0.0%, neighborhood size = 2 processors. The pMOMGA-II is an explicit building block-based algorithm and building block sizes were chosen a priori to execute. The population size, building block sizes, and number of generations executed varied with each MOP tested. The reader is referred to Chapter VI for more details on the MOMGA-II.

In order to validate the proposed island paradigm implementation and pMOEA migration and replacement strategies, a cluster of 32 Pentium-III dual node symmetric multiprocessors (SMP) yielding a total of 64 processors connected via a 100 Mbps Ethernet backplane through a 100 Mbps switch is used. The underlying hardware includes 1 GHz

processors connected to 1 GB of RAM through a 133 MHz Front Side Bus (FSB). The Red Hat Distribution, version 7.1 (kernel version 2.4.17), of the Linux operating system is used along with version 1.2.2.1 of MPICH for the parallel communications. MPICH was compiled with the gnu gcc compiler, version 2.96-85.

7.6.2 Experimental Results and Analysis. The island paradigm is typically used to attempt and increase the effectiveness of the pMOEA over the serial MOEA with a similar execution time. The first test presented uses 1, 2, 4, 8, and 16 processors in an attempt to solve MOP 4. The population size is identical on each processor, 25 members, to ensure that each processor receives enough work to complete and thus maintains a similar execution time to the serial MOEA execution. The resulting PF_{known} sets are presented in Figure 7.14.

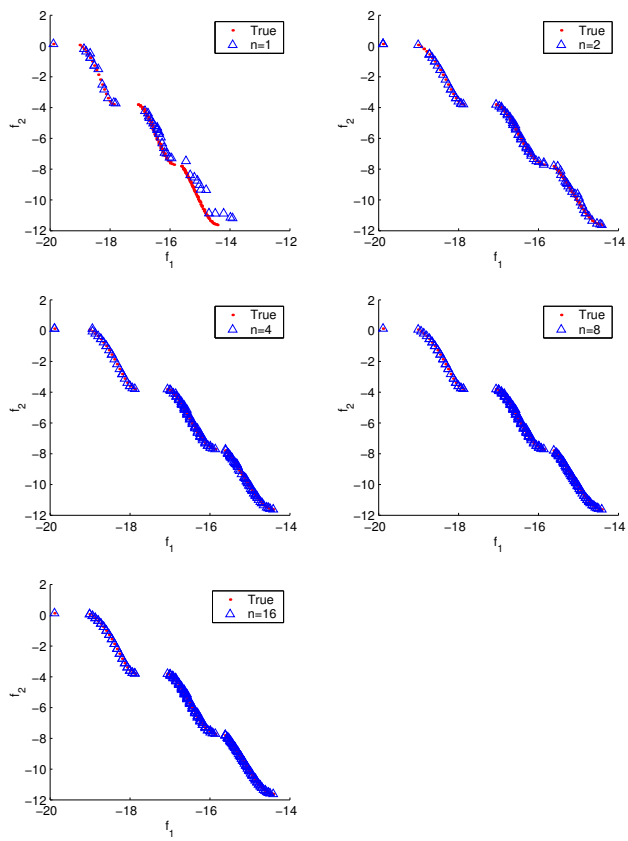

Figure 7.14 Parallel MOP4 Processor PF_{known} Comparison

Figure 7.14 shows that increasing the effectiveness of the pMOEA is possible over the serial MOEA through the addition of processors in order to search more of the space. As additional processors are used, the pMOEA finds a larger number of solutions in PF_{known}, obtains a better distribution of points across the Pareto front, and finds a larger number of solutions in PF_{true}. The metric values for using different numbers of processors are presented in Figure 7.18 at the end of this chapter.

As the number of processors increases, the variance of the metric values typically decreases as shown in Figure 7.18. This is due to the additional search process conducted through the use of more than one processor. As additional processors are used, some of these processors generate identical solutions as others, but many of the processors generate solutions that other processors did not generate. Since an identical population size is maintained on each processor, these results illustrate that the additional processors are able to search different areas of the space and generate solutions not found by other processors.

A subset of the various migration and replacement schemes proposed in Section 7.4.1.2 of this chapter are implemented and applied to MOP 4 in order to compare the results. Since it is possible to implement variants of the migration and replacement schemes proposed, only three schemes are selected for testing. To illustrate the advantage of a parallel implementation, the overall population size in Figure 7.15a is kept constant. The serial MOEA with a population of 50 members is compared to the parallel MOEA using 2 processors and a combined population of 50 members, each of these tests used strings of length 24 bits and executed 90 generations per run. The migration interval is set to every 3 generations and 5 members are chosen for migration and replacement with each event. The combined population size of the parallel implementation was kept equal to the serial tests to address the question of which implementation performs "better." One can see in Figure 7.15a the parallel implementation obtains a good distribution of points along the front and more overall points and a better distribution of points in the lower right portion of the front. This is important as the lower right portion of the front is a more difficult area to generate solutions.

In Figure 7.15b, two of the migration schemes are compared. The method of migrating $P_{current}\left({}^{p}_{t_pm_x}\right)$ is compared to that of migrating $P_{current}\left({}^{p}_{t_pm_x}\right)$ plus a few random

members. It is shown that the migration of the random members along with the current Pareto front provides a better distribution of points across the front.

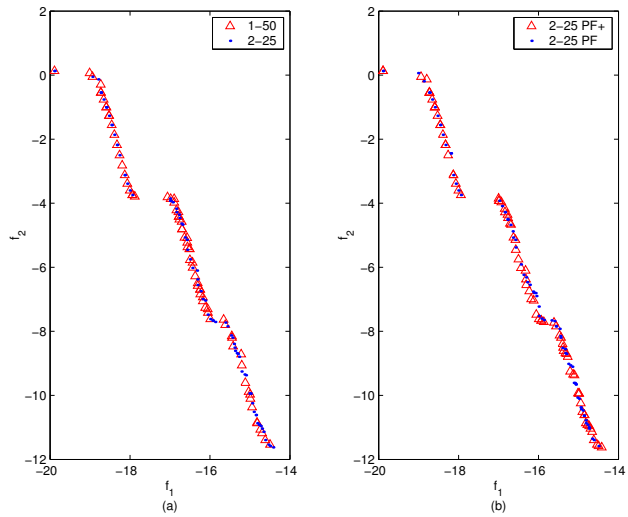

Figure 7.15 Parallel MOP4 PF_{known} Comparison

The metric values are presented in Table 7.5 (at the end of this chapter) for each of the selected migration and replacement schemes shown in Figure 7.15. Table 7.5 shows that the results of the serial and pMOMGA-II execution are similar for generational distance, ONVG and spacing. This is attributed to the fact that a serial MOEA can effectively find solutions to this MOP and hence a more difficult MOP for an MOEA is necessary to test a pMOEA's performance.

In the testing of the MMOKP MOP, a comparison is presented between the various migration schemes, in particular two elitist schemes are compared to the random scheme. The population size is kept at 100 members per processor, 2 processors are used and the 2 objective, 100 item MMOKP MOP is executed with 100 bits per population member and 100 generations per building block size. The migration interval is set to every 3 generations and 5 members are chosen for migration and replacement with each event. Figure 7.16a shows that the migration of $P_{current}\left({}^{p}_{t_pm_x}\right)$ finds more points and a better distribution of points along the front with the exception of the lower right portion of the front. Figure 7.16b shows that the migrating of $P_{current}\left({}^{p}_{t_pm_x}\right)$ plus some random members performs better than that of migrating just $P_{current}\left({}^{p}_{t_pm_x}\right)$ in terms of the cardinality of

the solution set and the distribution of points across the front. Figure 7.16c shows that the migration of $P_{current}\left(\begin{smallmatrix} p \\ t_pm_x \end{smallmatrix}\right)$ plus some random members performs better than that of just migrating random members, even in the lower right.

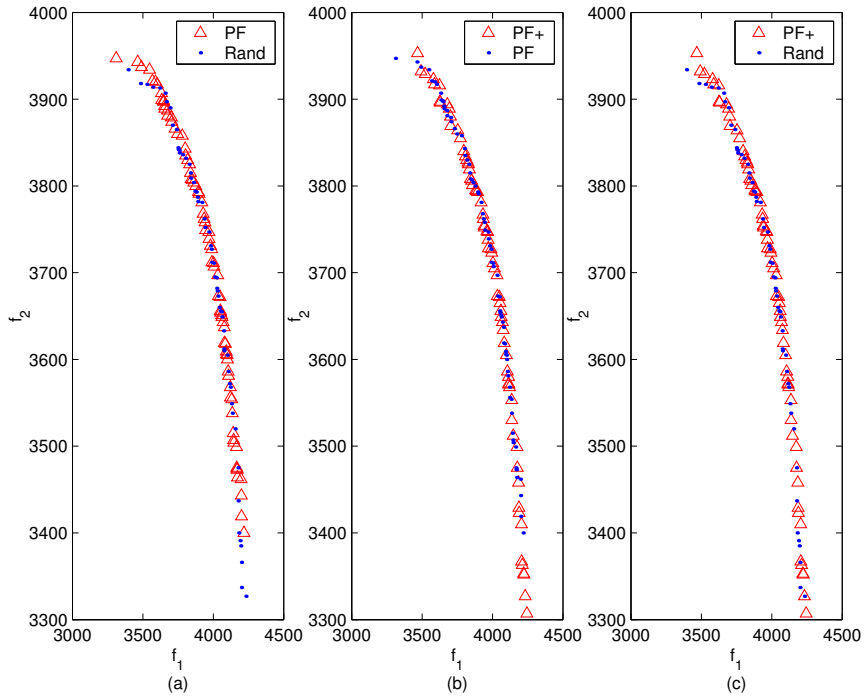

Figure 7.16 MMOKP PF_{known} Comparison

In analyzing the results of Figures 7.15 and 7.16, one can see that the difference in performance between the various migration and replacement schemes tested is relatively small. This is due to the problem complexity associated with these formulations. A larger MOP formulation is required to take full advantage of the possible effectiveness of a parallel island implementation.

The metric values are presented in Table 7.6 (at the end of this chapter) for each of the selected migration and replacement schemes shown in Figure 7.16. Table 7.6 shows that the results of the serial and pMOMGA-II execution are similar for ONVG and spacing. However, the migration of $P_{current}\left(\begin{smallmatrix} p \\ t_pm_x \end{smallmatrix}\right)$ plus some random members does generate a better distribution of points across the front and at the endpoints. This MOP instantiation contains a larger number of decision variables than MOP 4 but is not difficult enough to illustrate the usefulness of a pMOEA island paradigm implementation.

Figure 7.17 illustrates the results of the 250 item MMOKP MOP, with 2 fitness functions. In the attempted solving of this MOP formulation, the population size is kept at 100 members per processor, 2 processors are used and the 2 objective, 250 item MMOKP MOP is executed with 250 bits per population member and 100 generations per building block size. One can see that Figure 7.17 illustrates a much greater difference in the performance of the two migration and replacement schemes. The migrating of $P_{current}\left(\begin{smallmatrix} p \\ t_pm_x \end{smallmatrix}\right)$ plus some random members performs much better than that of just migrating random members. The migrating of $P_{current}\left(\begin{smallmatrix} p \\ t_pm_x \end{smallmatrix}\right)$ obtains a larger number of points in PF_{known}, a better distribution of points and entirely dominates the PF_{known} set generated by migrating only random members. This further strengthens the statement that one must use a parallel implementation only if the problem is suited for it and the complexity requires it. Otherwise one risks finding equivalent or worse solutions than a standard serial implementation may yield.

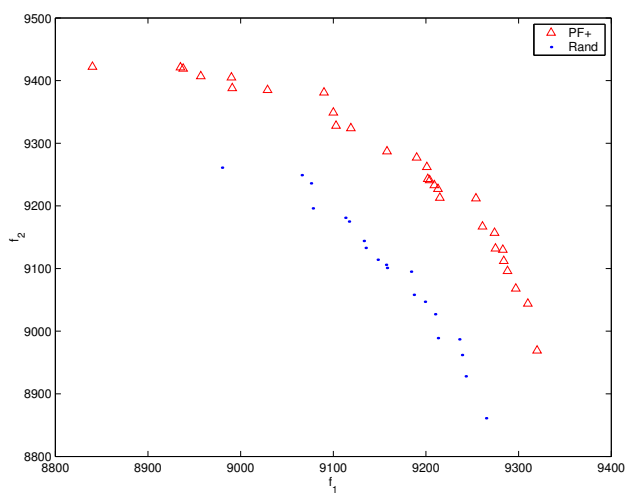

Figure 7.17 MMOKP PF_{known} Comparison

Overall, for both MOPs, the method of migrating $P_{current}\left(\begin{smallmatrix} p \\ t_pm_x \end{smallmatrix}\right)$ alone or with a few random members obtains more points on the Pareto front than does that of the serial method or the random migration method. In all of the migration methods applied to MOP4, the average results support the use of the Elitist (front +) migration scheme but the ANOVA testing indicates similar performance. The reason for this is that MOP4 is a difficult MOP but one that can be solved with many serial MOEAs. This, along with

the small population sizes used, does not lend itself to be the best MOP to test parallel concepts.

When utilizing a small population size, once the population converges to a solution set, eventually all of the processors tend to converge to a similar set over the course of the algorithm. It is recommended for future testing to execute the tests for fewer generations so that each of the processors may converge to a solution but not force an overall convergence to the same points on the Pareto front. In hindsight, fewer generations should have been used to allow for a better analysis of which processor generated each solution and hence better illustrate the effect migration has on a pMOEA. This analysis provides for a comparison of the combined front across all of the processors, generation by generation, so as to determine at what point has the algorithm generated the best overall combined solution, and at what point are the processors just exchanging good population members but not generating any of better quality. Scalability is important; although a small number of processors are employed here, the scalability of the inherently parallel island paradigm across many processors is anticipated to result in similar characteristics in terms of the exploration.

As a comparison to the results obtained for MOP4, the 250 item MMOKP MOP illustrates a definite difference in performance between the migration and replacement schemes. This further illustrates that applying parallel MOEA concepts to a difficult problem for an MOEA to solve yields better performance than the serial MOEA. Hence, selection of the MOP to solve, and the specific parallel concepts is crucial to obtaining good results from the pMOEA.

The results illustrate that through a simple island implementation one can obtain better results in terms of the cardinality of PF_{known}, the distribution of points along PF_{known} and the overall results when compared to the serial MOEA utilizing the same parameters. The reason for this is that the pMOEA allows for the exploration of different regions of the landscape, the generation of different building blocks, and hence the generation of better overall results. Hence utilization of an pMOEA as applied to MOPs that are challenging to serial MOEAs has the potential to dramatically increase the efficiency and effectiveness of the results. Additionally, migrating the current Pareto front alone or with

some random members tends to outperform that of the random migration scheme in the cardinality of PF_{known} , the distribution of points along PF_{known} and in the overall results. Additional testing is necessary to further analyze the concepts presented when applied to other MOPs.

These results gain insight into the fact that pMOEAs may only be more effective than MOEAs when applied to large, complex MOPs where serial MOEAs do not generate the quality of results required or when the pMOEA is executed with fewer generations. In the MMOKP MOP, there is a larger difference between the results obtained for the serial implementation and that of the parallel implementations but again this is a fairly small problem to test. The concentration in this test was to determine if a difference exists between the various migration schemes. The results are somewhat similar between each of the tested migration methods on small MOPs (MOP) but a noticeable difference is seen when applied to larger MOPs (MMOKP MOP with 250 items). Further analysis with more difficult MOPs and fewer overall generations is suggested in order to develop further insight into the relationship between effectiveness and efficiency of pMOEAs implementations.

7.7 Summary

This chapter discusses a high level overview of pMOEA state-of-the-art and presents several meaningful contributions and innovations. A thorough discussion of the major pMOEA paradigms (master-slave, island, diffusion) is included and observations made regarding analysis of the current literature. The discussion presented is important as it aids researchers in understanding pMOEA. An in-depth understanding of these concepts can lead to the development of efficient and effective pMOEAs. Specifically, a major contribution to the MOEA field is the innovative concepts for pMOEA migration and replacement presented. Another major contribution lies in the first presentation of pMOEA niching and archiving concepts as well as a generic pMOEA formulation.

A previous MOEA notation is extended into the pMOEA domain to enable precise description and identification of various sets of interest. Taken together, this analysis and original pMOEA development serve as a pedagogical framework and example of the necessary process to develop an efficient and effective pMOEA. Various recommendations

for creating pMOEA abstractions, designs and implementations and extending the field's knowledge through further research are made. This aids the community in achieving the goal of a fuller understanding of pMOEAs and appropriate contexts for comparing their performance.

Testing some of the concepts presented in this chapter led to a number of conclusions. The application of a master-slave paradigm to an MOP is most useful when the fitness function calculations are time consuming and exceed the communication time required to transmit population members and fitness values. There is a point where the addition of slave processors is counterproductive and decreases the execution time performance of the pMOEA.

Island paradigm pMOEAs present a number of implementation issues that center around the selection of population members for migration and replacement. This chapter presents the first discussion of the migration and replacement schemes as well as testing to validate the performance of a limited set of these schemes as well as the development of the first explicit BB-based pMOEA, the pMOMGA-II. One conclusion made from this portion of the research is that an MOP must be complex and difficult for a serial MOEA in order to realize any real advantage with an island paradigm pMOEA. For example, the testing of the pMOMGA-II as applied to MOP 4 and the 100 item MMOKP reflected little difference between the results of the serial and parallel implementations. This is attributed to the ability of a serial MOEA to effectively find solutions to the MOP. However, the application of the MOMGA-II to the 250 item MMOKP MOP illustrates that a noticeable difference exists. The differences between the various migration schemes tested illustrates that the method of migrating $P_{current}\left(\begin{smallmatrix} p \\ t_pm_x \end{smallmatrix}\right)$ alone or with a few random members obtains more points and typically a better distribution of points across the Pareto front than does that of the serial method or the random migration method. These results illustrate that the application of a pMOEA to an MOP of sufficient difficulty (an MOP that MOEAs cannot easily generate solutions for) can lead to the generation of a PF_{known} set of higher cardinality and a better distribution of points across PF_{known}. The objective of this chapter is met through the presentation of pMOEA concepts, the development of innovative

migration and replacement schemes for pMOEAs, and the development of the first explicit BB-based pMOEA, the pMOMGA-II.

The pMOEA computational paradigms span a wide spectrum and thus each must be evaluated to determine their effectiveness and efficiency across a wide spectrum of applications and implementations on various software and hardware architectures. Moreover, if possible, a theoretical foundation should continue to evolve for MOEAs and pMOEAs in order to provide a foundation for the selection of a computational architecture with associated pMOEA operators and parameter values. This work attempts to motivate the direction of such efforts.

It is obvious that other pMOEA developers, with varying levels of experience, may implement different approaches. These new (or equally similar) approaches are equally important to further the pMOEA field. Well-designed studies of this nature can only aid in further understanding critical pMOEA issues, seeing the impact of differing paradigms on pMOEA efficiency and effectiveness, and gaining a better grasp of algorithmic (class) performance through careful experimentation and analysis.

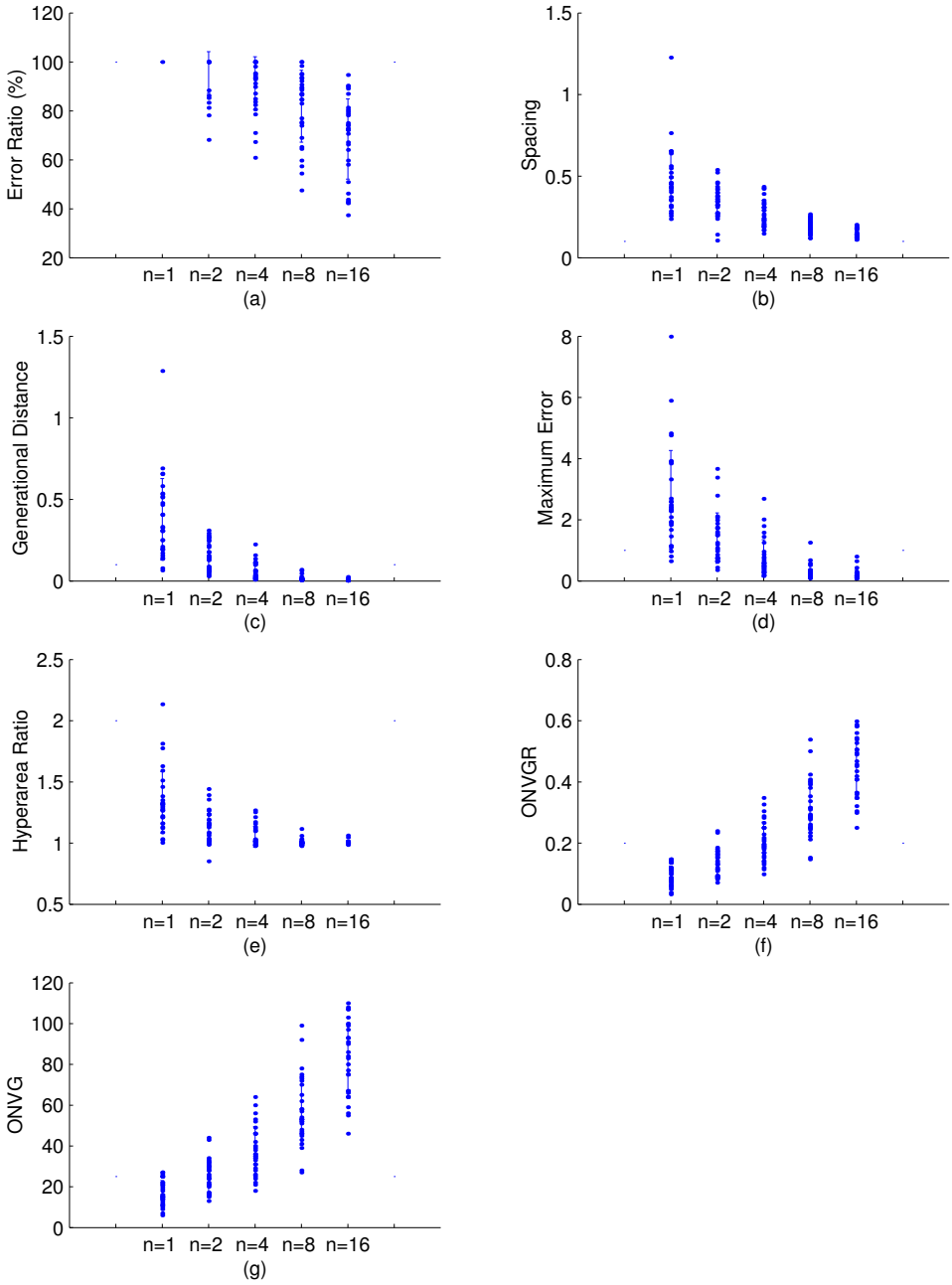

Figure 7.18 pMOMGA-II MOP4 Metrics

Table 7.5: Parallel MOP 4 Results

Number of Processors	**Pop Size Per Processor**	**Mig/Rep Scheme**	**Generational Distance**		**ONVG**		**Spacing**	
			Mean	**SD**	**Mean**	**SD**	**Mean**	**SD**
1	50	N/A	0.183	0.131	23.233	7.908	0.370	0.154
2	25	PF+	0.254	0.225	20.533	6.801	0.441	0.200
2	25	PF	0.171	0.100	22.967	8.143	0.374	0.117

Table 7.6: Parallel 100 Item, 2 Knapsack MMOKP Results

Number of Processors	**Pop Size Per Processor**	**Mig/Rep Scheme**	**ONVG**		**Spacing**	
			Mean	**SD**	**Mean**	**SD**
2	100	Rand	34.800	5.561	22.557	5.230
2	100	PF+	32.233	5.083	22.155	4.676
2	100	PF	30.900	5.095	23.502	4.971

VIII. MOEA Population Sizing

Many researchers in the Multiobjective evolutionary algorithm (MOEA) field have recognized the need for additional theoretical contributions to the field's underlying foundations [31, 44, 184]. In Chapter III, the breadth of theoretical MOEA contributions is presented and summarized. Additional knowledge is therefore needed for further insight into MOEA execution and results. One very important MOEA theoretical area that requires additional development is that of MOEA population sizing. For example, many MOEA researchers currently experiment with a few population sizes and only select the size that tends to perform the best [26, 62, 63, 109, 120, 175, 202].

Thus, in order to increase the probability of generating the true computational Pareto front for an MOP, theoretical analyses of MOEA population sizing is addressed in this research effort. A new generic MOEA population sizing formulation, based upon Building Block (BB) size, with the associated statistical theory is developed along with explicit underlying MOEA BB phenomena. Examples drawn from Chapter VI then illustrate the utility of this innovative model. We have used the single objective population sizing approach as the foundation of our model development. Therefore a brief discussion of Goldberg, Deb, and Clark's original single objective population sizing work is presented first [75].

8.1 Single Objective Population Sizing

The Building Block Hypothesis (BBH) [13, 74, 201] is considered by many to be the basis for understanding GAs and by extension, many MOEAs. The BBH, described in Chapter II, is also the basis for much of the population sizing work that has been completed to date for single objective EAs [75, 77]. Currently, no population sizing efforts have been openly published in the literature for MOEAs. An understanding of the BBH is integral to developing associated MOEA population sizing theory. The development of such theory is important to advance the understanding of MOEAs and determine a method for finding a population size that provides "good" results with a certain level of confidence.

Considering that single objective EAs have existed for many years prior to the implementation of the first MOEA, it is a worthwhile investment of time to understand the theory that has been developed for population sizing in these EAs. One of the most prominent research efforts that has contributed a great amount to the theoretical analysis of single objective GAs is that of David Goldberg and his students. In 1991, Goldberg et al. [75] published a paper on the theory of GA population sizing applied to both implicit and explicit BB-based GAs. This research was completed to understand what population size is necessary to use in order to generate a "good" solution(s) to optimization problems and in turn increase the probability of the GA finding, and converging, to the best solution(s) to the problem.

The single objective (SO) research recognized the fact that GAs with relatively small population sizes typically found good solutions more by chance than through an actual manipulation and discrimination of the best BBs. These results are problem dependent and due to the structure of the fitness landscape. Additionally, Goldberg et al. [75] note that at larger population sizes, a GA can reliably discriminate between various BBs of different quality. Effective use of these concepts along with an understanding of and intelligent employment of the various selection mechanisms can yield a very powerful GA.

The population sizing equations developed in [75] can lead to relatively large population sizes in certain cases, but this is necessary to achieve a high probability of generating a "good" solution to the problem [133]. The population sizing equation generated is based on the comparison of identically sized BBs and the idea that BBs are integral to finding the best solution to an optimization problem and centered on the BBH. If one can find the best BBs that exist for the problem, manipulate them through the standard evolutionary operators and perform selection, a high probability exists of finding the global optimum solution to the problem [75]. A much higher probability exists of generating good solutions through the use of a population sizing equation versus blindly choosing an initial population size to start with. Selection is important as an MOEA must be able to distinguish between various population members and select the better of those compared.

We acknowledge that there are many sources of noise to be found in the execution of any EA or MOEA and that the noise varies from one algorithm to another. Noise can

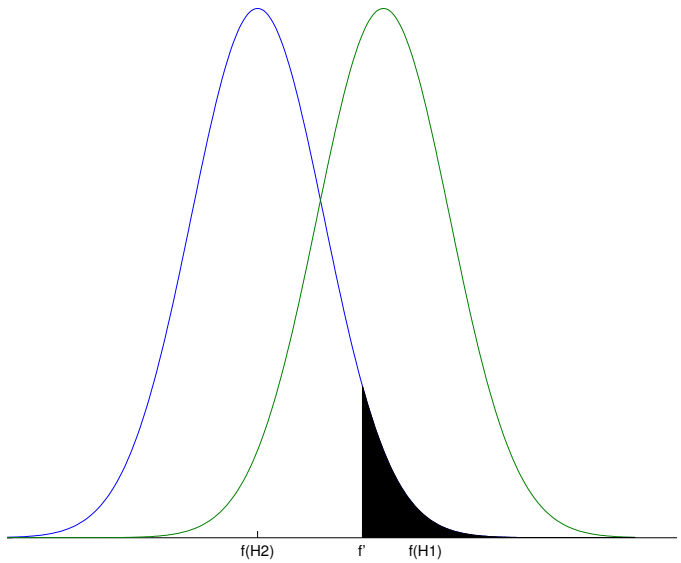

Figure 8.1 Distributions of the mean fitness values of two BBs, H_1 and H_2.

manifest itself in many areas, to include the GA operators, from selection to recombination, noise inherent in the problems themselves, and so on. Noise manifests itself in the genotype space and prohibits the evolutionary search from finding the good BBs and generating the good solutions due to limited genotype resolution. Goldberg based his population sizing research on the requirement that the probability of error in BB decisions be below some threshold α in each generation. The error threshold accounts for the noise inherently present in an EA.

Finding the good BBs present in the initial population is crucial to generating good solutions to the problem and potentially the global solution. The fitness value of any single BB is dependent on the selection of the remaining bits necessary to generate a fully specified potential solution. Since the evaluation of each BB is dependent on additional alleles, one cannot compute a single evaluation of a BB and expect with any confidence that the value calculated is indicative of the mean fitness value for the respective BB. In order to compare BBs and evaluate the quality of a BB, one must determine the mean fitness value of a BB across a number of different combinations of bits that lead to fully specified individuals. The mean fitness values are important as the selection mechanism

used (tournament selection) compares the fitness values of two individuals at a time in order to determine which of the two is better.

In the single objective case, one begins by looking at the mean fitness values of two competing BBs, H_1 and H_2. According to the Central Limit Theorem (CLT) [198], even if one does not know if the fitness values of the BBs are normally distributed, one can state that the distribution of the means of the sample fitness values of the BBs is approximately normal if the sample size is large enough. The objective is to determine the sample (population) size necessary in order to allow one to determine which of two BBs is better with a level of confidence. In Figure 8.1, the distribution of the means over a number of samples of the fitness values of two building blocks are plotted, where any mean fitness value to the right of another is better, considering this is a maximization problem.

In Figure 8.1, BB H_1, whose distribution (and mean) is shown to the right of BB H_2 is better in terms of its associated mean fitness value. The two distributions overlap which makes it difficult to discern which BB is better in the overlapping areas of the distributions. If we chose a particular fitness value, f', both of the BB distributions contain values above this threshold fitness value [75] for a portion of their distribution. [1] In this case, an error is made by choosing H_2 over the overall better BB H_1 in the shaded region of Figure 8.1. Building Block H_1 is referred to as the overall better BB since the mean of BB H_1's distribution is higher (better) than that of BB H_2 and assuming that the problem is not deceptive. A deceptive problem is one in which the deceptive BB is a maximum distance away from the BB necessary to find the global optima.

From Figure 8.1, one can calculate the probability that the worse schema, H_2, is better than a particular fitness value, f', by finding the shaded area under the curves. One can further calculate the overall probability that the sample fitness of H_2 is better than the sample fitness of H_1 by summing the probabilities for all f' which is in fact the convolution of the two distributions. Since the distribution of the means shown are normal and we know that the difference (convolution) of two normals is in fact normal [198]; therefore the mean of the difference is the difference between the means of the two normal

[1] f' is a particular fitness value and not the derivative of f.

distributions and the variance of the difference is the sum of the individual variances. Any linear combination of normal distributions is distributed normally [198]. This means that the underlying distribution of the fitness values of the BBs is inconsequential to this analysis.

The signal difference, the difference of the mean fitness values, of two competing BBs is defined as the random variable $d \triangleq f_{H_1} - f_{H_2}$. The mean variance of two BBs is $\sigma_d^2 \triangleq (\sigma_{H_1}^2 + \sigma_{H_2}^2)$ which is the variance of the mean fitness values. [2] Even though it is not stated directly in [75], this equation makes the implicit assumption that the two BBs H_1 and H_2 are independent. The assumption of independence of BBs is valid if there is no epistatic relationship between the BBs. In cases where the bit length of the BBs is much less than the fully specified string length, the probability of an epistatic relationship may be reduced. However, it is noted that this is problem dependent. The probability of making an error in any single comparison of the two BBs is computed through determining the probability α such that $z^2(\alpha) \triangleq d^2/(2\sigma_M^2)$ where $z(\alpha)$ is based on a one sided normal distribution. To clarify this, $z(\alpha)$ is the tail deviate value at an error probability that one deems acceptable in the empirical population sizing calculation.

In looking at Figure 8.1, the amount of overlap between the two distributions is considerable and increases the probability of making an error in comparing the two BBs. In order to reduce this probability of making an error, one should decrease the amount of overlap between the distributions. Looking to statistics, we know that as we increase the number of samples taken, the variance of the mean fitness value distributions decreases. The mean of the two distributions does not change, but the variance about the mean decreases as the number of samples is increased.

Through this process of increasing the number of samples taken, one can reduce the probability of making an error in choosing the "better" BB as shown in Figure 8.2. In order to reduce the probability of error, one must reduce the overlap between the distributions as much as possible. The number of required samples to reduce the overlap between the

[2] Note that σ_d^2 is used instead of σ_M^2 as in Goldberg [75]. However, $\sigma_d^2 \triangleq 2\sigma_M^2$.

distributions is problem dependent, but can be determined through an analysis of sample data.

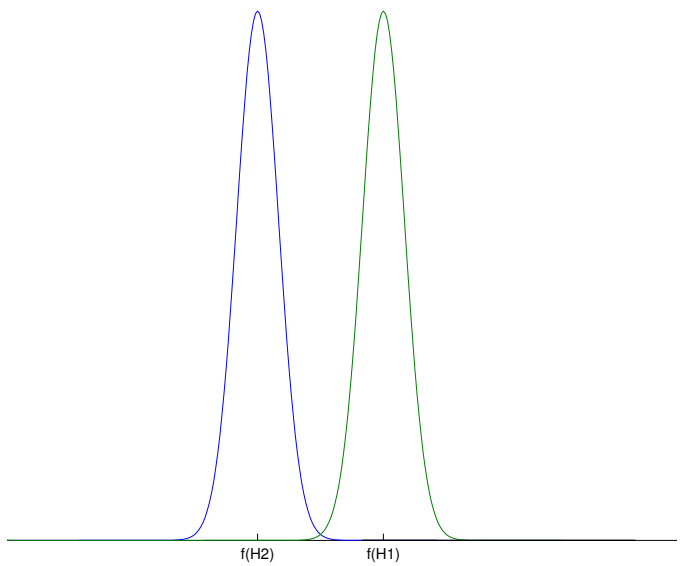

Figure 8.2 Distributions of the mean fitness values of BBs, H_1 and H_2 with an increased number of samples.

To derive the population sizing for single objective GAs, the variance of the population mean is computed as the variance of a single sample divided by the total number of samples. In this discussion, a single sample is the results obtained from a single run of the algorithm. Since the likely number of samples in a randomly generated population of size n is the population size divided by the number of competing schema, κ, that the two BBs compete with, to have an error rate of α one obtains the following equation:

$$z^2(\alpha) = \frac{d^2}{2\sigma_M^2/n'} \ , \tag{8.1}$$

where $n' = n/\kappa$. From this one can obtain a fairly generalized estimate of the population size required to obtain accurate results and converge within a reasonable amount of time. Manipulating Equation (8.1) yields Equation (8.2):

$$n = 2 * z^2 * \frac{\sigma_M^2}{d^2} \ . \tag{8.2}$$

Equation (8.2) assumes that the noise encountered in the GA comes entirely from the variance in the fitness values within the population. To improve the population sizing equation to account for other sources of noise, the equation is modified to include additional multiplicative terms. To account for i multiple sources of noise, a noise coefficient n_i, is presented in Equation (8.3):

$$\epsilon_{n_i}^2 \triangleq \frac{\sigma_{n_i}^2}{\sigma_M^2} \ , \tag{8.3}$$

where the summation of the noise coefficients is presented as:

$$\epsilon_T^2 \triangleq \sum_i \epsilon_{n_i}^2 \ , \tag{8.4}$$

and assumes independence between the sources of noise. To account for the effects of varying the string length, l, the number of BBs that fit into a string of length l is m, where $m = l/o$. The number of BBs of order o is χ^o where χ is the cardinality of the alphabet being used (often times this is binary where $\chi = 2$), and o is the order of the BB or the specified BB size. Including the total number of BBs of a specified order into the equation, the new population sizing equation, to include i sources of noise and the number of BBs is thus Equation (8.5):

$$n = 2 * z^2 * \frac{\sigma_M^2}{d^2} * m * (1 + \epsilon_T^2) * \chi^o \ . \tag{8.5}$$

The terms are multiplicative because of the possible large detrimental effects that the noise introduces into the algorithm and the incorrect choices that may be made in comparing two BBs. Generation of values for the various terms is based upon assumptions and experimental testing and usually results in a large (conservative) value of n [133].

8.2 Multiobjective Population Sizing

Proper population sizing for MOEAs is a problem area that many have identified as of interest to further the theoretical development of MOEAs [31, 44]. This is an area that has not been addressed in the current literature as discussed in Chapter III. While the single objective population sizing development discussed in the previous section can lead to

extremely large population sizes, the theoretical basis for using a population that large is to ensure probabilistically that the necessary "good" BBs are in the initial population. The population sizing equation also attempts to ensure that enough of these BBs are generated so that the good BBs are not all destroyed through the noise inherent in the evolutionary process and operators. These BBs must be present in the initial population in order to be selected, recombined, and generate better solutions, hopefully finding the global optima.

In this section, a novel population sizing equation is developed for use by the MOEA community. This equation is valid for implicit and explicit BB-based MOEAs. The population sizing equation presented at the end of this section is an estimate of the population size necessary to obtain good solutions to the MOP, and it is also the first known theoretical work into the sizing of MOEA populations necessary to obtain "good" solutions. All of the BB comparisons are conducted over BBs of an identical size.

The development of a multiobjective population sizing equation is not straight forward. The problem encountered is that a one dimensional fitness comparison of competing BBs cannot be made, but instead, a multidimensional comparison of BBs, where the dimensionality is equal to the number of fitness functions must be completed. In any multiobjective formulation, a fitness value vector exists for the evaluation of each of the two BBs, H_b, containing k components, one for each of the k fitness functions. The fitness values observed are represented by fitness vectors $\mathbf{f}$ containing elements $f_k(H_b)$, where k represents the fitness function and b represents the specific BB (H_b represents the specific BB; 1 or 2). Each of the two BBs is evaluated with respect to each of the fitness functions thus yielding a fitness value vector. The BBs are evaluated over multiple samples to generate a mean fitness value vector. By the CLT the mean fitness value vectors of each BB are distributed as multivariate normal distributions.

Pareto dominance is used as the criterion for stating that one BB's mean fitness vector is "better" than another. In using Pareto dominance criteria to determine whether or not one BB's mean fitness vector is "better" than another, three possibilities exist. The first possibility is that BB H_1's mean fitness vector dominates that of H_2. The second possibility is that BB H_2's mean fitness vector dominates that of H_1. The last possibility is that neither BB H_1's or H_2's mean fitness vectors dominate, and hence they are both

nondominated BBs. An error is made if the incorrect, or dominated, BB is chosen over the correct, or nondominated, BB in the first two possibilities. In the last possibility, a choice of either BB is not specified as an error in this research as both BBs are nondominated. This is reflected in the tournament selection routine used by many MOEAs. Given a choice of two nondominated BBs, one is typically randomly selected to become part of the next population. Another variant of this would be to allow both BBs to be selected for inclusion in the subsequent population. The definition of error must reflect how the selection mechanism is implemented in the MOEA.

For example, if the mean fitness vector of H_1 is "better" than the mean fitness vector of H_2 then one would like to select population members representing H_1 more often than those representing H_2. In general, *the issue at hand is determining how to increase the probability that strings are selected that represent the nondominated BB H_1 more often than those that represent the dominated BB H_2?* To address this, the probability of making an "error" or the number of times that one selects population members containing H_2 over those containing H_1 on a single trial, must be minimized.

Another objective of this chapter is to present a process for researchers to follow that allows one to determine the MOEA population size. This process requires one to either take random samples of each BB to estimate the true mean fitness values (the disadvantage is that one does not know which BBs are the good BBs are in any real-world MOP), have theoretical insight into the MOP through problem domain knowledge that allows for a good estimate of the true mean fitness values (this is the method used by Goldberg but one does not know if their estimate is a good estimate of the true mean fitness values [77]), or apply this population sizing approach to a pedagogical MOP in which the true mean fitness values are known. It is assumed that the sample size and number of samples is large enough so that the distribution of the means of the sample fitness values of the population is approximately multivariate normal.

The multivariate normal density function, for a k-dimensional normal density, for the random vector $\mathbf{X} = [X_1, X_2, \ldots . X_k]'$ has the form [103]:

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{k/2}|\boldsymbol{\Sigma}|^{1/2}}\, e^{-(\mathbf{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})/2} \tag{8.6}$$

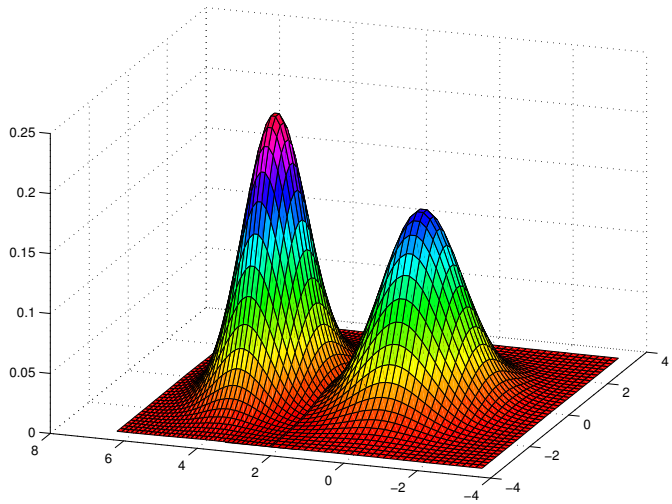

Figure 8.3 Distributions of the mean fitness values of BBs, BB H_1 and H_2 for a two dimensional MOP

where $-\infty < x_i < \infty, i = 1, 2, \ldots, k$. This k-dimensional normal density function is denoted as $N_k(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Two objective function ($k = 2$) MOPs are used for ease of visualization and understanding. The bivariate case has the form [103]:

$$\begin{aligned} f(x_1, x_2) &= \frac{1}{2\pi\sqrt{\sigma_1\sigma_2(1-\rho_{12}^2)}} \\ &* \quad exp\left\{-\frac{1}{2(1-\rho_{12}^2)}\left[\left(\frac{x_1-\mu_1^2}{\sqrt{\sigma_1}}\right)^2 + \left(\frac{x_2-\mu_2^2}{\sqrt{\sigma_2}}\right)^2\right.\right. \\ &\qquad \left.\left. - \; 2\rho_{12}\left(\frac{x_1-\mu_1^2}{\sqrt{\sigma_1}}\right)\left(\frac{x_2-\mu_2^2}{\sqrt{\sigma_2}}\right)\right]\right\} \end{aligned} \tag{8.7}$$

The distributions of the mean fitness values of two BBs selected for comparison, H_1 and H_2, when using two objective functions ($k = 2$) is presented in Figure 8.3. In Figure 8.3, each of the distributions of the mean fitness values for the BBs has a bivariate normal distribution. The area of overlap between the distributions contains the fitness values where an error may occur, an incorrect determination of which BB is better.

The sample mean of the fitness values are represented by the k element vectors $\bar{\mathbf{X}}_b$. The components of the sample mean, $\bar{X}_{kb}$, can be used as estimates for the components of the actual mean, μ_{kb}, since the random variables are taken from the sample data. A k

element sample fitness variance vector, $\mathbf{S}_b^2$, for each BB, b, containing each of the sample fitness variance values, S_{kb}^2, is calculated from the sample mean values. The sample variance values are estimates of the population variance values, which are represented by σ_{kb}^2. It is necessary to define additional symbology for deriving the population size without the use of sample data, F_{kb} is the random variable that has a probability distribution based on fitness function k and BB b. μ_{kb} represents the population mean of the random variable F_{kb}. f_{kb} is the observed value identified through testing and $\bar{f_{kb}}$ is the sample mean of this variable. $\mathbf{F_b}$ is the vector representing the random variables F_{kb} and $\mathbf{f}(b)$ is the vector representing the observed variables f_{kb} for each BB.

To calculate the overall probability that the sample fitness vector of the second BB is mistakenly determined to be "better" than the first BB, requires a complex calculation. The term better is used in the sense of Pareto dominance. This calculation involves finding the probability at any given fitness vector value $\mathbf{f}'$ that the sample fitness $\mathbf{f}(H_2)$ dominates $\mathbf{f}(H_1)$. This is a calculation of the probability of the worse BB dominating the "better" BB for a particular fitness vector, $\mathbf{f}'$. The integral of all the probabilities, for all values of $\mathbf{f}'$, yields the overall probability that the worse BB is actually better. A difference calculation, D, is used to distinguish between the two BBs. This difference, D, is the difference between the mean fitness values of the two BBs over a number of samples. As the number of samples is increased, the standard deviation of the difference between the sample means of the distributions becomes smaller, there is less overlap in the tails of the multivariate normal distributions (for the two BBs being compared), and the level of confidence in the choice of which BB is "better" increases. This is shown in Figures 8.1 and 8.2 for the single objective case for simplicity in understanding where the overlap occurs. A smaller amount of overlap in the distributions means a smaller probability of making an error in choosing which is the "better" BB.

The mean variance of the two BBs, σ_d^2 is needed to calculate the overall population size for an MOEA. The signal difference, D, is a random variable and is defined as the difference in the means of the fitness values of the two BBs compared. If the assumption of independence of the BBs and the fitness functions is made, D is distributed normally

with the estimated means and variances specified:

$$D = (F_{11} + F_{21}) - (F_{12} + F_{22}) \tag{8.8}$$

In the generalized case of k fitness functions, D, can be represented as:

$$D = \left(\sum_k F_{k1}\right) - \left(\sum_k F_{k2}\right) \tag{8.9}$$

Johnson and Wichern [103:p. 165] present a theorem of the normal distribution, where $\mathbf{X}$ is a random vector having a multivariate normal distribution.

Theorem 2 *If* $\mathbf{X}$ *is distributed as* $N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$*, then any linear combination of variables* $\mathbf{a}'\mathbf{X} = a_1X_1 + a_2X_2 + \ldots + a_pX_p$ *is distributed as* $N(\mathbf{a}'\boldsymbol{\mu}, \mathbf{a}'\boldsymbol{\Sigma}\mathbf{a})$*. Also, if* $\mathbf{a}'\mathbf{X}$ *is distributed as* $N(\mathbf{a}'\boldsymbol{\mu}, \mathbf{a}'\boldsymbol{\Sigma}\mathbf{a})$ *for every* $\mathbf{a}$*, then* $\mathbf{X}$ *must be* $N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ *[103].*

In the bivariate case,

$$\mathbf{F} = \begin{bmatrix} F_{11} \\ F_{12} \\ F_{21} \\ F_{22} \end{bmatrix} \sim N_p(\boldsymbol{\mu}_{\boldsymbol{F}}, \boldsymbol{\Sigma}_{\boldsymbol{F}}) \ . \tag{8.10}$$

The marginal distributions can be written as:

$$F_{11} \sim N(\mu_{F_{11}}, \sigma_{F_{11}}) \tag{8.11}$$

$$F_{12} \sim N(\mu_{F_{12}}, \sigma_{F_{12}}) \tag{8.12}$$

$$F_{21} \sim N(\mu_{F_{21}}, \sigma_{F_{21}}) \tag{8.13}$$

$$F_{22} \sim N(\mu_{F_{22}}, \sigma_{F_{22}}) \tag{8.14}$$

Using Theorem 2, D can alternatively be written as:

$$D = \mathbf{a}'\mathbf{F} \tag{8.15}$$

where $\mathbf{a}'$ is the coefficient vector of the linear combination of D. The coefficient vector is presented in Equation (8.8). Therefore, for the bivariate formulation shown in Equation (8.8), the coefficient vector is:

$$\mathbf{a} = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \tag{8.16}$$

Equation (8.8) presents a linear combination of variables in the form presented in Theorem 2. Therefore, using Theorem 2, Equation (8.8) can be alternatively written as:

$$D \sim N(\mathbf{a}'\boldsymbol{\mu}, \mathbf{a}'\boldsymbol{\Sigma}\mathbf{a}) \tag{8.17}$$

The covariance matrix is represented by $\boldsymbol{\Sigma}$. Considering that dependencies can exist between fitness functions and BBs, multiple covariance matrices are possible. For simplicity of understanding, the four categories of covariance matrices $\boldsymbol{\Sigma}_{\mathbf{k_1}-\mathbf{k_2},\mathbf{b_1}-\mathbf{b_2}}$ are presented for a two objective case. The process of generating the covariance matrices described is applicable to k objectives. In these matrices, $\mathbf{k_1} - \mathbf{k_2}$ represents a dependency between the fitness functions specified by $\mathbf{k_1}$ and $\mathbf{k_2}$ and $\mathbf{b_1} - \mathbf{b_2}$ represents a dependency between the BBs specified by $\mathbf{b_1}$ and $\mathbf{b_2}$. The covariance matrices are $2k$ x $2k$ matrices, where k is the number of fitness functions and the comparison is always made between two selected BBs. In the simplest case, of only two fitness functions, assuming independence of BBs and independence of the fitness functions, all of the covariance terms are 0, and the covariance matrix is:

$$\boldsymbol{\Sigma}_{(\mathbf{0},\mathbf{0})} = \begin{bmatrix} \sigma^2_{F_{11}} & 0 & 0 & 0 \\ 0 & \sigma^2_{F_{12}} & 0 & 0 \\ 0 & 0 & \sigma^2_{F_{21}} & 0 \\ 0 & 0 & 0 & \sigma^2_{F_{22}} \end{bmatrix} \tag{8.18}$$

If the BBs are independent but both of the fitness functions illustrate a dependency, then the covariance matrix is:

$$\mathbf{\Sigma}_{(\mathbf{1-2},\mathbf{0})} = \begin{bmatrix} \sigma^2_{F_{11}} & 0 & Cov(F_{11}, F_{21}) & Cov(F_{11}, F_{22}) \\ 0 & \sigma^2_{F_{12}} & Cov(F_{12}, F_{21}) & Cov(F_{12}, F_{22}) \\ Cov(F_{21}, F_{11}) & Cov(F_{21}, F_{12}) & \sigma^2_{F_{21}} & 0 \\ Cov(F_{22}, F_{11}) & Cov(F_{22}, F_{12}) & 0 & \sigma^2_{F_{22}} \end{bmatrix} \tag{8.19}$$

If both BBs exhibit a dependency but the fitness functions are independent, then the covariance matrix is:

$$\mathbf{\Sigma}_{(\mathbf{0},\mathbf{1-2})} = \begin{bmatrix} \sigma^2_{F_{11}} & Cov(F_{11}, F_{12}) & 0 & Cov(F_{11}, F_{22}) \\ Cov(F_{12}, F_{11}) & \sigma^2_{F_{12}} & Cov(F_{12}, F_{21}) & 0 \\ 0 & Cov(F_{21}, F_{11}) & \sigma^2_{F_{21}} & Cov(F_{21}), F_{22}) \\ Cov(F_{22}, F_{11}) & 0 & Cov(F_{22}, F_{21}) & \sigma^2_{F_{22}} \end{bmatrix} \tag{8.20}$$

If both the BBs and the fitness functions exhibit a dependency, then the covariance matrix is:

$$\mathbf{\Sigma}_{(\mathbf{1-2},\mathbf{1-2})} = \begin{bmatrix} \sigma^2_{F_{11}} & Cov(F_{11}, F_{12}) & Cov(F_{11}, F_{21}) & Cov(F_{11}, F_{22}) \\ Cov(F_{12}, F_{11}) & \sigma^2_{F_{12}} & Cov(F_{12}, F_{21}) & Cov(F_{12}, F_{22})) \\ Cov(F_{21}, F_{11}) & Cov(F_{21}, F_{12}) & \sigma^2_{F_{21}} & Cov(F_{21}, F_{22}) \\ Cov(F_{22}, F_{11}) & Cov(F_{22}, F_{12}) & Cov(F_{22}, F_{21}) & \sigma^2_{F_{22}} \end{bmatrix} \tag{8.21}$$

This covariance matrix formulation can easily be extended to any number of fitness functions and any dependency existing between any combination of the respective fitness functions. A positive covariance indicates that a higher than average value of one variable (BB) is compared to a higher than average value of another variable (BB). A negative covariance indicates that a higher than average value of one variable (BB) is compared to a lower than average value of another variable (BB) [198]. In either situation, this illustrates a dependence between the two BBs and fitness functions that may require a larger or smaller overall population size, respectively.

Using the appropriate covariance matrix, coefficient values, and Theorem 2 as presented previously, the equation for σ_d^2 is (where $\sigma_d^2 = 2\sigma_M^2$):

$$\sigma_d^2 = \mathbf{a}'\mathbf{\Sigma}_{\mathbf{F_k},\mathbf{H_b}}\mathbf{a} \tag{8.22}$$

where $\mathbf{a}'$ is a vector representing the coefficients of the D terms shown in Equation (8.16). To provide a level of confidence in the calculations and to calculate the probability of making an error, the confidence interval (CI) for the mean variance of the BBs, σ_d^2, is calculated based on the sample data. The CI provides an estimated range of population sizes necessary to handle the variance in the fitness values with respect to the BBs and provide some level of confidence that H_1 is correctly chosen over H_2 in the previous example.

What α level should be used for the CIs? The answer to this question depends on the level of confidence that one wants to achieve in their results. α is the probability of one falsely rejecting the null hypothesis. In this case, the Null Hypothesis H_0 is: BB 1, H_1, is not dominated by BB 2, H_2. The alternative hypothesis, H_a is: that H_2 dominates H_1. The overall error being calculated is that of rejecting the null hypothesis when in fact it is true, i.e., concluding that H_2 dominates H_1 when in fact it does not.

Once the CI for σ_d^2 is calculated, the upper confidence limit is selected. This upper confidence limit, $\hat{\Theta}_U$ [198] is used in the subsequent population sizing equation as the value for σ_d^2. With a probability of $1-\alpha$, one can be confident that the appropriate σ_d^2 value is used. Since the likely number of samples in a randomly generated population of size n is the population size divided by the number of competing schema, κ, that the two BBs compete with, to have an error rate of α one obtains the following equation:

$$z^2(\alpha) \triangleq \frac{d^2}{\sigma_d^2/n'} \ , \tag{8.23}$$

where $n' = n/\kappa$. From this, one can obtain a fairly generalized estimate of the population size required to obtain accurate results and converge within a reasonable amount of time.

The probability of making an error in a single comparison of two BBs is computed in determining the probability α where α is an acceptable error probability for use in

the population sizing of MOEAs. Since D is distributed normally, the $z(\alpha)$ value can be calculated, and hence, the probability of making an error on any single comparison of each BB can be stated. To enhance the multiobjective population sizing equation, the CI for σ_d^2 is calculated (the CI can also be used in the SO population sizing equation). The confidence interval provides one with a level of confidence that the population sizing equation yields a correct population size to use.

The CI provides a more conservative population size because the observed value, S^2, is typically inflated with the CI. In actuality, the variance may be smaller and therefore more population members are calculated than may be necessary. The required sample size is an increasing function of the variance of the BBs. Once the CI is found for σ_d^2, this value is used in the population sizing equation for MOEAs.

Observe that the assumptions for the single objective population sizing equation discussed in Section 8.1 are used to similarly derive the MOEA population sizing equation. To improve the population sizing equation to account for other sources of noise, the equation includes an additional term(s). Noise can manifest itself in many areas, to include the GA operators, from selection to recombination, noise inherent in the problems themselves, and so on. To account for i multiple sources of noise, a noise coefficient n_i, is presented in Equation (8.24):

$$\epsilon_{n_i}^2 \triangleq \frac{\sigma_{n_i}^2}{\sigma_M^2} \tag{8.24}$$

where the summation of the noise coefficients is presented as:

$$\epsilon_T^2 \triangleq \sum_i \rho_{n_i}^2 \tag{8.25}$$

Additional terms need to be defined in the population sizing equation to account for the effect of varying the string length. Strings of length l, with alphabets of cardinality χ, where χ often times is 2 (binary), are used. Assume that smaller BBs sizes are required to find the global optima ($o \ll l$) and that these BBs are better than other competing BBs in the population (i.e., the problem is not deceptive). The number of BBs that fit into one string of length l is m, where $m = l/o$. The number of competitors between BBs,

considering the highest order BBs, is χ^o. The focus is on the highest order BBs to remain conservative (larger) in the population sizing equation. Using these multiplicative terms in the same manner as the SO population sizing equation, the population sizing equation for MOEAs is derived in Equation (8.26):

$$n \triangleq z^2 * \frac{\sigma_d^2}{d^2} * m * (1 + \epsilon_T^2) * \chi^o \tag{8.26}$$

The parameter d is based on the difference in the distributions and can be calculated in the same manner as was done in Section 8.1. If the BBs are close together in fitness space, one may want a smaller width value so that more of the BBs are found in the population, but this increases the population size. If the good BBs are dispersed more so in fitness, then a larger value for d may be appropriate since one can accept fewer BBs due to how they are distributed in fitness space. This can be viewed in terms of a standard deviation and related to the variance of the fitness functions.

To determine the population size necessary to obtain a good solution set, a Pareto front, estimates for the means, the variances $\sigma_{kH_b}^2$, the confidence interval of the mean variance of the BBs σ_d^2, the selection of an appropriate α level and distance parameter are necessary to determine a good starting population size for an MOEA. If good estimates of these parameters are unknown, one can use observed values for each of the associated parameters to generate the population size required to yield good solutions.

The covariance terms have a direct effect on the population sizing equation as the MOEA must be able to distinguish between the BBs in determining which is "better." For example, if a higher than average value of H_2 is compared with a lower than average value of H_1, from Figure 8.1, then this is a region where the distributions may overlap and hence there is a large probability of making an error. A negative covariance increases the overall population size. This intuitively makes sense as the harder it is to distinguish between BBs, the larger the required population size necessary to achieve an acceptable error level. Convergence properties of an MOEA using this population sizing equation to determine the initial population size are expected to be empirically advantageous.

8.3 MOEA Population Sizing Reality

In order to determine the effect of this population sizing equation, a pedagogical example is presented. In this example, using the parameter values from the testing documented in Chapter VI, the population size N is calculated using Equation (8.26). Tables 8.1 and 8.2 present the population sizes required for the string lengths and BB sizes used in testing MOPs 1, 2, 3, 4, and 6 and MMOKP. In the testing of the standard MOPs, strings of length $l = 24$ and BBs of size $o = 1 - 3$ were used. The remaining parameters values, necessary to calculate the population size are taken from Merkle, Gates, and Lamont [133].

In Merkle, et al. [133], an approximate error level of $\alpha = 0.01$ was selected yielding $z^2(\alpha) = 6$, and $\sigma_d^2/d^2 = 0.25$. These values were calculated for a single objective Protein Structure Prediction problem, but they can be considered somewhat low approximations of the values one may expect to encounter in applying MOEAs to this problem. In many MOPs, the maximum required BB size may be larger than that used in single objective problems. This larger BB size additionally leads to the calculation of a larger overall population size based on the population sizing equation. Therefore, the utilization of larger BB sizes and σ_d^2/d^2 calculated from limited experimentation with an MOP generally yields a larger overall population size.

The population sizes required for some of the MOPs used in Chapter VI provides insight into the values of the initial population sizes required. The error level is specified by the user and hence a value of 6 for $z^2(\alpha)$ is realistic. Additionally, as the dependence between fitness functions and BBs increases, so may the value for σ_d^2/d^2, thereby increasing the overall population size. The resultant population sizes calculated using Equation (8.26) and the specified parameters from [133], for each of the associated BB sizes, is presented in Tables 8.1 and 8.2.

The initial population size used in Chapter VI for MOPs 1-6 is 250. To determine the population size to use in experimentation, one selects the largest population size presented in Table 8.1 (552 members) as this is the required population size for BBs of size 3. In comparing the results of the population sizing equation (552 members) to the population size used (250 members), the population sizing equation requires a larger number of mem-

bers to be used in order to generate good solutions. The difference between the required size of 552 members and the size used of 250 members illustrates that in this example, a smaller initial population size than that calculated is sufficient to generate good solutions. This is due to the upper limit of the confidence interval used in the sizing equation. This upper limit may yield a larger population size than necessary in some cases. Additionally, it is shown in Chapter VI that the MOMGA-II using a population size of 250 performed statistically similar to or better than most other MOEAs. An increase of that population size is expected to have a limited impact on the results of the MOMGA-II. The results using 552 members is expected to be statistically similar to the MOMGA-II results using 250 members in most cases for MOPs 1, 2, 3, 4, and 6.

Table 8.1 Estimate of MOEA Population Size for MOPs 1-6 ($l = 24$)

o	**N**
1	138
2	276
3	552

The population sizes necessary to obtain good empirical results for the MMOKP grow rapidly as the BB size is increased. The population sizes presented in Table 8.2 are calculated using the same parameter values as Table 8.1, with the exception that the string length is increased to 100 for the 100 item MMOKP. In testing the MOMGA-II's performance over the 100 item MMOKP, a population size of 400 members was used and a maximum BB size of 12. In comparing the 400 members used in experimentation with the MOMGA-II to the recommended population size presented in Table 8.2 of 1,216,512 one can see that the population size used was inadequate. The recommended population size is orders of magnitude greater.

If the population size was modified to reflect the size present in Table 8.2, the MOMGA-II is expected to perform statistically better than it did for this MOP. The disadvantage is that the execution time of the algorithm would be increased substantially, potentially resulting in an infeasible test due to time and resource constraints. The resulting population size from Equation (8.26) may not always be feasible to implement but it

Table 8.2 Estimate of MOEA Population Size for MMOKP ($l = 100$)

o	N
1	138
2	1188
3	2376
4	4752
5	9504
6	19008
7	38016
8	76032
9	152064
10	30412
11	608256
12	1216512

is a guideline to illustrate the sizes necessary to obtain good results. Additionally, if the recommended size exceeds a number that the researcher can feasibly use, due to time or resource constraints, the researcher can choose a more favorable population size in terms of their constraints. Once a size is selected, the researcher can calculate an estimate of the probability that this population size may yield good results through Equation (8.26).

Tables 8.1 and 8.2 illustrate the large population sizes that may result from utilization of the multiobjective population sizing equation (Equation (8.26)). The population sizes calculated and presented in Tables 8.1 and 8.2 may be lower estimates than one may expect given the parameter values were calculated from a single objective test. Note that the population sizes actually used are much smaller than those derived from the population sizing equation. This is expected as the population sizing equation presents a conservative (large) estimate of the population size required to obtain good solutions and since multiobjective problems may require the use of larger BB sizes.

The resulting population sizing equation derived as Equation (8.26) is the first known population sizing equation for MOEAs. This equation is a useful model for MOEA researchers in determining the population size necessary to obtain good solutions. This model is also useful to determine with what probability a different population size may generate good results. This population sizing equation and knowledge required to derive it provides a more complete understanding of MOEAs.

8.4 Summary

The population sizing model of MOEA populations is discussed in this chapter. A novel MOEA population sizing equation is developed based on a sound statistical foundation including previous SO population sizing efforts. This equation can lead to conservative population sizing values because of the statistical assumptions. This model is applicable to both explicit and implicit BB-based MOEAs. Even though the BB size symbolic notation is identical, in MOEAs, the extensive variation of BB sizes and associated cardinality is large. As observed in Chapter VI examples, BBs of a large size may be required to make the associated large steps in phenotype space in order to move towards the Pareto front. These BBs of large sizes and the large number of required BBs results in large empirical population sizes. The results of this chapter indicate that such population sizes are quite large in order to statistically converge to the Pareto front.

The symbolic form of the MOEA population sizing equation is quite similar to the SO population sizing equation. The main differences lie in the determination of the σ_d^2 term from the multivariate normal distribution, which accounts for the multiple fitness functions as well as the possible dependence between both the BBs and fitness functions. The example BB size discussion presented in Chapter VI, and the empirical population sizes employed, indicate that a bounding population sizing value would be useful. Such a population sizing equation (Equation (8.26)) is presented in this chapter with calculated population sizes for various MOPs. Analysis indicates that when the empirical population size is orders of magnitude less than that of the theoretical population size, convergence to the Pareto front, PF_{true} , is limited. The population sizing research presented also allows for the consideration of dependencies between BBs and objectives in the determination of a good population size to use. Additionally, a researcher can use the population sizing equation to determine the level of confidence an MOEA will generate good solutions to an MOP using a population size that differs from the result of the population sizing equation.

IX. Conclusions and Documentation

9.1 Introduction

This chapter summarizes the major contributions resulting from this research effort. The discussion presented in this dissertation should be of great interest and use to the MOEA community as well as other researchers outside of the MOEA community. A number of contributions have been made throughout this effort and each of these contributions is important to advancing the MOEA state-of-the-art.

Research into MOEAs is extremely important. MOEA researchers can leverage their knowledge of MOEAs, to develop new, efficient, and effective MOEAs that perform "better" than current methods. The importance of MOEAs lies in the ability of researchers to effectively use MOEAs to generate good solutions and potentially solve problems of various classes. In particular, the Air Force can use the results of this effort to find possibly better solutions to current MOPs of interest. An example of this is the Advanced Logistics Problem, for which the MOMGA-II generated very good results, as shown in Chapter VI. Additionally, the results of the MOMGA-II as applied to the NP-Complete Modified Multiobjective Knapsack Problem illustrates that the MOMGA-II found good solutions to a discrete MOP containing a large number of integer based decision variables. The ALP and MMOKP MOP results are useful in that they illustrate the good performance of the MOMGA-II over problems of this class. The MOMGA-II can be used by Air Force specialists in attempting to solve real-world Air Force MOPs. The thoughts and ideas presented in this document are original or referenced from the appropriate authors. This dissertation serves as an original work, meant to extend the state-of-the-art of MOEAs and pMOEAs.

The five objectives and the research goal of this effort are met. Each of the objectives, with references to the chapters that detail the respective objective, are summarized. This research is a contribution to the MOEA community and advances the state-of-the-art in terms of MOP analysis through the use of BBs, BB-based MOEAs, parallel processing integration in MOEAs, and the theoretical development of MOEA population sizing.

9.2 Contributions

One question typically asked of a research effort is *Did the research effort meet the intended objectives?* The answer to this question is yes. The goal and objectives of this research are met as supported by this dissertation. This research yields contributions to the MOEA community concentrated in the area of explicit BB manipulation in MOEAs.

There are five main contributions resulting from this effort. Some of these contributions are major and others are minor but nevertheless, all of the contributions of this dissertation aid in advancing the state-of-the-art in MOEAs. Symbolic formulations of MOEA operators are presented to aid researchers in understanding MOEA operators. A new explicit BB-based MOEA, the MOMGA-II is designed, implemented, and tested. The development of the MOMGA-II is a contribution to the MOEA field and provides researchers insight into MOPs that implicit BB-based MOEAs do not provide. Through applying the MOMGA-II to test suite MOPs, a limited number of constrained MOPs formulated with integer based decision variables are identified for future use by other researchers. Another contribution to the MOEA field is the presentation of migration and replacement schemes to use in pMOEAs as well as the testing of these schemes. As more researchers are gaining interest in the use of pMOEAs and concentrating much of their effort on island paradigm pMOEAs, the presentation of migration and replacement schemes aids in the development process. The advancement of the theory of MOEA population sizing, with the development of an MOEA population sizing equation is another major contribution resulting from this research effort. The objectives of this research effort are now discussed.

9.2.1 Research Goal. The goal of this research effort, *to advance the state-of-the-art with respect to explicit Building Block Based MOEAs*, is met. Through the research results presented, and meeting all five of the stated objectives (from Chapter I), this dissertation effort advances the state-of-the-art with respect to explicit BB-based MOEAs. All of the results, analyses and discussions presented contribute to the MOEA community and advance the state-of-the-art with respect to explicit BB-based MOEAs. The goal is to advance the field of MOEAs in the specific areas of BB theory, implementation

details, defining and utilizing multiobjective building blocks, addressing the under used concept of pMOEAs, and to present a clear and concise method for MOEA researchers to intelligently use parallel processing concepts in their algorithms. This research has shown the importance of BBs in MOEAs and the insight that can be found through the utilization of BB-based MOEAs. This is an insight that other MOEAs could not have provided due to their implicit handling of BBs.

9.2.2 Symbolic Formulations. Symbolic formulations of an MOEA and its operators is presented in Chapter V. These formulations serve to aid researchers in developing a clear understanding of an MOEA and its associated operators. The formulations presented follow a standardized notation for ease of understanding. Additionally, these formulations aid in developing a high level understanding of MOEAs and an in-depth understanding of MOEA operators. This level of understanding aids researchers in validating MOEA implementations and in developing new, innovative MOEAs and associated MOEA operators. The use of a standard notation for the detailed definitions of the MOEA operators and MOEA algorithm is something not present in other approaches. The first objective, *to present a symbolic formulation of MOEA operators and algorithm details*, is satisfied.

The symbolic formulations presented also aid researchers interested in implementing a generic MOEA and its operators. The use of this standardized symbolic MOEA formulation encourages researchers to complete the analysis of other MOEAs using the notation presented. The presentation of standardized MOEA definitions, using the formulations presented, aids in comparing the structure of MOEAs and understanding the process MOEAs follow to generate good solutions.

9.2.3 Explicit BB-Based MOEA Development. The second objective, *to develop an explicit BB-based MOEA for solving constrained and real-world MOPs*, resulted in the development of an explicit BB-based MOEA, the MOMGA-II, in Chapter VI. To illustrate the importance and utility of explicit BB manipulation in the analysis and attempted solving of MOPs, the MOMGA-II is developed. The development of the MOMGA-II is of great interest to the MOEA community as it further establishes the utility of explicit BB-based MOEAs. The MOMGA-II generates potential solutions to real-world and constrained

MOPs is orders of magnitude less execution time than previous explicit BB-based MOEAs and achieves a similar statistical level of effectiveness while requiring less resources over the selected test MOPs. The results presented in Chapter VI, Section 6.5 illustrate that the MOMGA-II requires less resources and achieves similar or better statistical performance in comparison to the MOMGA over a limited test suite.

The selection of MOPs, to test any MOEA, is an extremely important choice. Many MOP test suites exist in the literature, composed of unconstrained problems of varying characteristics as discussed in Chapter IV, but most do not contain constrained, discrete, NP-Complete, or real-world test problems with integer based decision variables. MOPs are presented for testing and comparing the performance of various MOEAs, including the MOMGA-II, over different classes of problems.

Many real-world MOPs are constrained and are formulated with discrete decision variables. Both the Advanced Logistic Problem and MMOKP are constrained, discrete optimization problems with integer based decision variables. The MMOKP is an example of an MOP containing hundreds of decision variables and the ALP MOP is an approximation to a real-world problem of similar characteristics. Additionally, a constrained test function generator is presented that allows a researcher to vary the characteristics of the MOP by modifying two variables in the MOP formulation. The results of the MOMGA-II as applied to various MOPS is presented and a thorough statistical testing procedure is used to evaluate the performance of the MOMGA-II. A number of constraint handling mechanisms are implemented in the MOMGA-II with a concentration on exploring repair mechanisms for discrete integer MOP formulations. The MOMGA-II is the first explicit BB-based MOEA designed to attempt and solve real-world and constrained MOPs. Various metrics are presented and used to compare the performance of the MOMGA-II to other MOEAs and to the true solution (when known).

The MOMGA-II achieves a competitive level of effectiveness when compared to selected implicit BB-based MOEAs. The MOMGA-II typically obtains results that indicate statistically similar effectiveness when compared to other contemporary and recent implicit BB-based MOEAs over a limited test suite. Different visualization techniques are used to clearly present the results to the reader. While the visualization of data may appear to

be inconsequential, this is an important issue in a field where multidimensional data is commonplace. While the MOMGA-II is not statistically better than all other MOEAs for all problems, this is expected as the NFL Theorem dictates. However the MOMGA-II performs very well when applied to the ALP problem and the MMOKP MOP, both reflective of other real-world, constrained application MOPs.

The MOMGA-II performs well when applied to all of the MOPs discussed and when compared to other MOEAs. The testing of the MOMGA-II presented in this research effort illustrates that good performance can be achieved with the generic parameter settings specified for the MOMGA-II. Additionally, problem domain knowledge can be easily integrated into the MOMGA-II, as was done through the integration of repair mechanisms, to improve the performance as applied to MOEA difficult MOPs. The MOMGA-II was developed for application to real-world and constrained MOPs. Hence, the code was generalized for somewhat easy application to various MOP formulations. The good performance of the MOMGA-II makes it an attractive tool to use in conjunction with other tools for attempting to solve real-world MOPs. Attempts to solve real-world MOPs should make use of a variety of optimization approaches and the MOMGA-II is a good tool to be used to attempt to solve MOPs by itself or in conjunction with other optimization approaches. This illustrates that the MOMGA-II meets objective 2.

9.2.4 BB Insight. BBs are defined in Chapter II. This research effort recognizes and identifies that there is an effect of explicitly manipulating BBs in attempting to solve MOPs, a major contribution to the MOEA field. This effort presents the first explicit analysis of the effect of BB sizes on MOP results. The results illustrate that BBs and explicit BB-based MOEAs provide insight into difficult MOPs that implicit BB-based MOEAs do not.

It is also shown that in general BB sizes are directly related to the ability of an MOEA to generate points on the Pareto front (see Chapter VI). The BB sizes also directly relate to the location of the points generated by the MOEA. This is an important observation as many researchers have been unable to explain the reason that MOEAs generally do not find the endpoints of the Pareto front, be it a curve or a surface. While it is easy to state

that the reason is due to the complexity of the MOP, in Chapter VI (Section 6.6) it is shown that this effect is generally due to the inability of other MOEAs to generate the mix of BB sizes that tend to generate the endpoints of the Pareto front. The use of larger BB sizes and multiple BB sizes tends to generate more solutions near the endpoints of the Pareto front as illustrated by the results of the MOMGA-II. This further illustrates the usefulness of a BB-based MOEA where BB sizes can be explicitly specified and used in the MOEA.

The visualization of BBs is presented to emphasize what constitutes a BB and to show the effect BBs have on finding the potential solutions to an MOP. Chapter VI illustrates the usefulness of BBs in MOEAs and supports meeting objective 3, *to demonstrate that explicit BB-based MOEAs may provide insight into solving difficult MOPs.* The results of Chapter VI are of interest to the MOEA community as such an analysis of the use of BBs in attempting to solve MOPs has not been addressed in the literature.

9.2.5 Parallel MOEAs. The fourth objective, *to describe parallel concepts for MOEAs, present innovative migration and replacement schemes for use in parallel MOEA paradigms, and develop the first explicit BB-based parallel MOEA*, is met. This objective is met through the detailed discussion of pMOEA concepts, the development of innovative migration and replacement strategies for pMOEA paradigms, and the development of the parallel MOMGA-II in Chapter VII. A detailed discussion of the three main parallel MOEA paradigms is presented and aids researchers unfamiliar with parallel processing concepts to understand the most popular and possibly most useful ways of using parallelism in MOEAs. Variations of the three main parallel MOEA paradigms are discussed. A pMOEA is defined and associated concepts and ideas from pEAs are extended for utilization in pMOEAs. As an in-depth knowledge of both MOEAs and parallel processing is required to intelligently develop an efficient and effective pMOEA, a detailed discussion of the process necessary for parallelizing an existing serial MOEA is presented as a guideline for researchers to use. The detailed discussions of pMOEA concepts are integral to understanding pMOEAs and developing new, innovative, efficient and effective pMOEAs.

The first detailed discussion of innovative pMOEA migration and replacement strategies is presented. This is a major contribution as none of the existing pMOEA publications reviewed discuss the details of how to implement migration and replacement strategies even though the effective use of these strategies is integral to achieving good performance in pMOEAs. The explicit use of BBs in a pMOEA is detailed and the effects of parallelization are discussed. The MOMGA-II is selected for parallelization and tested on a limited suite of MOPs. The results of the parallel MOMGA-II testing illustrate performance gains that can be realized through effective use of parallel concepts in MOEAs. A comparison of the strategies is conducted and results are presented in Chapter VII (Section 7.4.1). These details are important to identify and discuss in order to design more efficient and effective pMOEA implementations.

9.2.6 MOEA Population Sizing. The theoretical development of MOEA population sizing is an area upon which MOEA research has not been published. The sizing of MOEA populations is integral to the generation of "good" solutions. While researches typically agree with this statement, the literature review conducted in Chapter III found no theoretical developments related to MOEA population sizing. In Chapter VIII (Section 8.2) the first generic MOEA population sizing equation, Equation (8.26), based on a solid statistical foundation is presented and is a major contribution to the MOEA field. This research is beneficial to the MOEA community and advances the state-of-the-art as other researchers have not addressed this important issue. The development of a population sizing equation serves to aid researchers in understanding the population sizes necessary to ensure the generation of good results with a particular level of confidence. While in many cases use of the calculated population sizes may be prohibited by system resources and execution time requirements, such a development may be useful in pMOEAs or lead to further developments in the sizing of MOEA populations. The discussion of the process used to develop the population sizing equation is important as it may aid researchers in obtaining better results to the attempted solving of real-world MOPs. Additionally, the concepts of BB and fitness function dependence are important to recognize as well as the effect they have on the application of MOEAs to solving real-world MOPs. The sizing

equation presented is applicable to implicit and explicit BB-based MOEAs and satisfies the fifth objective, *to enhance MOEA theory by deriving MOEA population sizing theory.*

9.3 Future MOEAs and their Operators

What does the future hold for MOEAs and MOEA research? This is a question to which many of us would like to know the answer, but the answer is unknown. The future of MOEAs can be decomposed into a few areas, that of new operators mimicking Biology [107], new MOEAs (those that take entirely different approaches), and improvement of existing MOEAs. The generation of new EA/MOEA operators is not a likely occurrence unless a movement is made to more closely mimic biology. A more likely scenario is the slight variation of existing operators. There is much room for improvement and theoretical analysis of current operators. As various representations for EAs have been around for some time now, most of the community has concentrated their attention towards binary based EAs. In upcoming years, there should be additional work in the area of understanding why MOEAs work and thereby the ability to generate new operators that can take advantage of the theory associated with the workings of these MOEAs.

Another important question is *Are we as a community generating better, innovative MOEAs that can guide us towards solving larger, harder, more complex problems or are we generating new MOEAs just so each of us have one to call our own?* Some of the recent papers that have been published on new MOEAs are leaning towards the latter, with few theoretical contributions, comparisons with existing MOEAs, and conclusions based on limited testing of why this MOEA is better than that one. The community as a whole must move towards focusing on theoretical aspects of new MOEAs not just the creation of these algorithms with no theoretical basis. The future holds large opportunities for the presentation of theoretical understanding of MOEAs and their associated operators. The introduction of problem domain information into MOEA operators has not been a main focus of researchers in the past. While some researchers have incorporated problem domain knowledge into their operators, this is an area that could be expanded. The integration of problem domain knowledge into MOEA operators has the potential to provide effectiveness improvements in MOEAs.

An expansion of the application of the MOMGA-II to the MOPs presented in this effort is considered for future work. Additional testing of the MOMGA-II and other MOEAs as applied to MOPs of differing characteristics is planned in the future. In particular, since the MMOKP MOP used in this effort does not accurately reflect the classical multiobjective knapsack problem, the entire MOEA community should consider using the classical multiobjective knapsack problem as an additional test suite MOP for testing MOEAs. A study of the classical multiobjective knapsack problem and the use of different constraint handling mechanisms may be useful to the MOEA community especially since there exists a large amount of literature related to attempts to solve the classical multiobjective knapsack problem. Continuing down the same path, MOEA researchers should look to the Operations Research community and their experiences with attempting to solve constrained MOPs. The knowledge gained from this type of study and the different approaches and experiences of those in the Operations Research field could benefit the entire MOEA community. Additionally, the testing of Operations Research test suite problems expands the nature of the MOP test suites used by MOEA researchers. This expanded testing may illustrate the applicability of MOEAs to solving MOPs from other domains not currently addressed in the MOEA MOP test suites. The evaluation of MOEA performance as applied to Operations Research problems may lead to the development of new innovative MOEA approaches and the development of more robust MOEAs with increased efficiency and effectiveness.

A generic MOEA that incorporates all of the different MOEA operators is a contribution to the community and furthers researchers' understanding of MOEAs. Such a development may lend more insight into the operation of various operators as well as providing a test bed to easily compare and contrast different MOEA operators. Additionally, a generic MOEA would allow a researcher to choose from multiple operators when attempting to solve an MOEA. A generic MOEA, developed through contributions from the MOEA community, would be a useful tool. With the scrutiny of the many researchers who contribute and analyze the MOEA, the potential exists for the development of a well-engineered code. Additionally, with numerous researchers providing additional analysis of the MOEA, errors in the code may be more readily identified and corrected. This is an

advantage over the development of an MOEA by a single researcher who may not recognize coding errors he or she has made. Overall this effort may lead to the development of a powerful optimization tool.

Future work is planned to further analyze the parallel MOMGA-II using the criteria stated. A comparison of the migration and replacement schemes as applied to larger MOPs continues as well as an analysis of the pMOMGA-II's scalability and other associated parallel issues. It is also important to evaluate the performance of various parallel niching and archiving concepts proposed in this effort. Additionally, other existing MOEAs (SPEA2 [211] and the NSGA-II [46]) are being considered for extension to pMOEAs; the goal is creation of other generic pMOEAs. The SPEA2 and NSGA-II are two recent MOEA implementations that use different operators than the MOMGA-II, do not explicitly employ building block concepts, and do employ different ranking, fitness assignment and sharing methodologies. Rather than concentrating on comparing many algorithms more useful results may arise from the comparison and understanding of a select few. Studies of this nature aid the community in further understanding pMOEA issues, the impact of the different parallel paradigms on pMOEA efficiency and effectiveness, and a better grasp of algorithmic (class) performance by using various metrics to present and explain results.

Attempts to improve existing MOEAs or generate new MOEAs continues. A closer look at the MOEAs presented in Table 3.1 illustrates the fact that most of the MOEAs are similar to each other. With the exception of the MOMGA/MOMGA-II, the remainder of the MOEAs are all variations of each other, based off of similar operators and implement similar methods of ranking, niching, etc. Be that the case, future research should address new and innovative concepts that are considerably different from those that currently exist.

Understanding the research that other researchers are conducting is important. The collaboration of researchers to implement new ideas and analyze existing research contributions is useful to advance the field of MOEAs. One of the best methods of obtaining this information is through the reading publications from the many conferences that take place each year and analyzing MOEA code. Coello Coello [26] maintains a web page listing the majority of MOEA publications made available and offers many of them for download as well as the code for MOEAs designed by researchers in the community. Coello Coello's

compendium of publications and software is useful to the MOEA community. Additionally, more researchers should consider placing their software at this location for others to download and understand. This helps the MOEA community become aware of other researcher's approaches and advance the development and understanding of MOEAs.

This dissertation presents a first look at the theory of population sizing in MOEAs. The conservative nature of the population sizing theory illustrates a need for additional testing. One may be able to forecast the population sizes required for specific classes of MOEAs based on the characteristics of the MOPs and the use of BBs. Additional MOEA theoretical development is welcomed for explicit BB-based MOEAs and BBs in MOPs. Promising results are presented in Chapter VIII, but there is much to accomplish.

Pareto dominance is used as the criterion for stating that one BB's mean fitness vector is "better" than another. In this effort an error is made if the incorrect, or dominated, BB is chosen over the correct, or nondominated, BB. If both BBs are nondominated then a choice of either BB is not defined as an error in this effort as both BBs are nondominated. Future research should explore the effect of randomly choosing a nondominated BBs on the overall population size required. This type of effort may possibly yield a better tiebreaking approach. Convergence and rate of convergence to the Pareto front by explicit BB-based MOEAs is an area left unexplored. Completing proofs of the conjectures in Chapter VIII is a start as well as adding to the theory of sizing MOEA populations and pMOEA populations.

The design of new, explicit BB-based MOEAs may not be necessary but improvements to the MOMGA-II and use of the MOMGA-II with other approaches for attempting to solve MOPs may be beneficial to the community. The MOMGA-II has the ability to yield results that illustrate the usefulness of BBs and explicitly show what the BBs are, something that other MOEAs cannot. Continuing research into explicit BB-based MOEAs may produce findings or conjectures that move the MOEA community closer to understanding the how and why MOEAs work.

Additional variations on many of the MOEAs presented in this document undoubtedly is in the future. The fruitfulness of these efforts is not known. The concept of creating

a general MOEA that can solve any MOP may not be possible but is worth the attempts made by researchers. The research conducted in this quest may lead to the generation of more efficient and effective MOEAs, their operators and additional theoretical development into MOEAs and pMOEAs. The advancement of MOEA theory may yield the best results as researchers understand more of the how and why MOEAs "work."

9.4 Publications

The research completed in support of this document has led to 21 publications [8, 9, 10, 11, 37, 38, 39, 40, 41, 106, 107, 143, 144, 145, 195, 196, 216, 217, 218, 219, 220] in three different problem domain areas, the general MOEA area and the applications of intrusion detection and Bioinformatics (Protein Structure Prediction). Of these publications, 2 are peer reviewed Journal articles [9, 195], and the remainder consist of 15 peer reviewed and 4 non-reviewed publications. One additional peer reviewed conference submittal is in-review. These publications are of interest to the Air Force as they illustrate an efficient and effective means of tackling MOPs that when used in conjunction with other approaches yields a powerful tool for attempting to solve MOPs. Other real-world applications of high dimensionality that MOEAs may yield good results when combined with other optimization approaches include Unmanned Aerial Vehicle routing and mission planning problems, Microelectronic chip layout problem, Intrusion Detection Problem, and Satellite scheduling problems.

Appendix A. EA Mathematical Formulation Background

The formulation of mathematical equations and definitions of EA operators aids in a researcher's understanding of the operators themselves. This understanding is necessary to ensure that EA operators are implemented correctly and aid in the development of associated operator theory. The mathematical formulations presented are summarized from Merkle [132, 135] and Bäck [13]. These formulations are used as necessary background for the multiobjective operator formulations developed in this research effort. Additionally, a discussion of a generic MOEA is included at the end of this appendix.

A.1 Evolutionary Operators

In order to utilize a consistent notation, the work of Merkle [132, 135] and Bäck [13] is summarized. Merkle [132] states that many authors describe evolutionary operators as directly mapping populations into populations, with the mapping being "controlled" by the parameters of the operator. In the following definitions the set of mappings from a set $\mathcal{S}_1$ to a set $\mathcal{S}_2$ is denoted $\mathcal{T}(\mathcal{S}_1, \mathcal{S}_2)$.

The first definition is of a *population transformation*. This transformation, shown in Figure A.1, does not restrict the input and output populations to be of the same size.

Definition 46 (Population transformation): *Let I be a non-empty set (the individual space), and $\mu, \mu' \in \mathbb{Z}^+$ (the* parent *and* offspring population sizes*, respectively). A mapping $T : I^{\mu} \longrightarrow I^{\mu'}$ is called a* population transformation*. If $T(P) = P'$ then P is called a* parent population *and P' is called an* offspring population*. If $\mu = \mu'$, then they are called simply the* population size*.* □

The population transformation resulting from the application of an evolutionary operator, to include the offspring population size, often depends on the outcome of a random experiment. This dependence motivates the concept of a *random population transformation* shown in Figure A.2.

Definition 47 (Random population transformation): *Let I be a non-empty set (the individual space), $\mu \in \mathbb{Z}^+$ (the parent population size), and Ω a set (the* sample space*).*

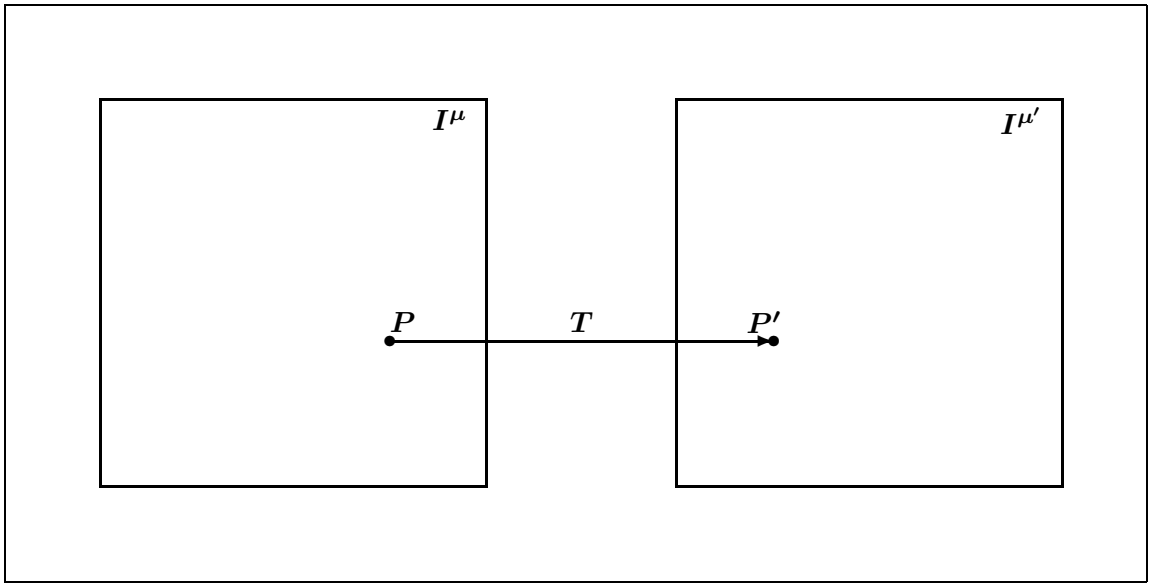

Figure A.1 The population transformation T deterministically maps the parent population P (of size μ) to the offspring population P' (of size μ') [132].

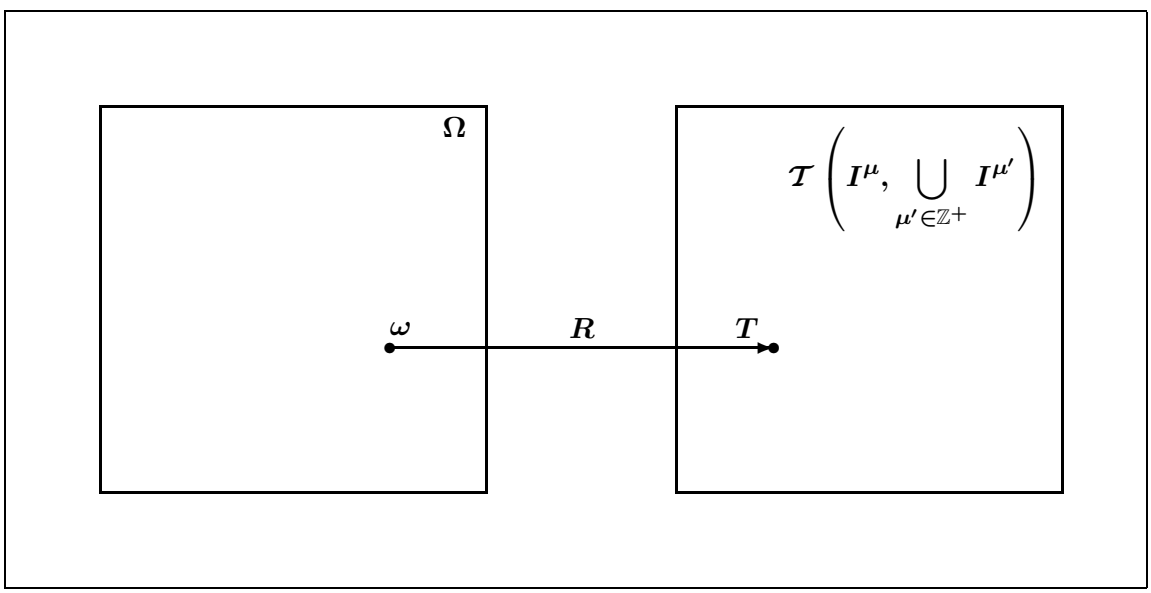

Figure A.2 The random population transformation R maps the random event ω (with sample space Ω) to the population transformation T, which maps parent populations of size μ (which is independent of ω) to offspring populations of some fixed size $\mu' \in \mathbb{Z}^+$ (which may depend on ω) [132].

A random function

$$R : \Omega \longrightarrow \mathcal{T}\left(I^{\mu}, \bigcup_{\mu' \in \mathbb{Z}^{+}} I^{\mu'}\right)$$

is called a random population transformation [132]. □

Merkle [132] further states that the distribution of population transformations resulting from the application of an evolutionary operator may depend on one or more parameters of the operator. Each evolutionary operator maps its parameters to a random population transformation shown in Figure A.3.

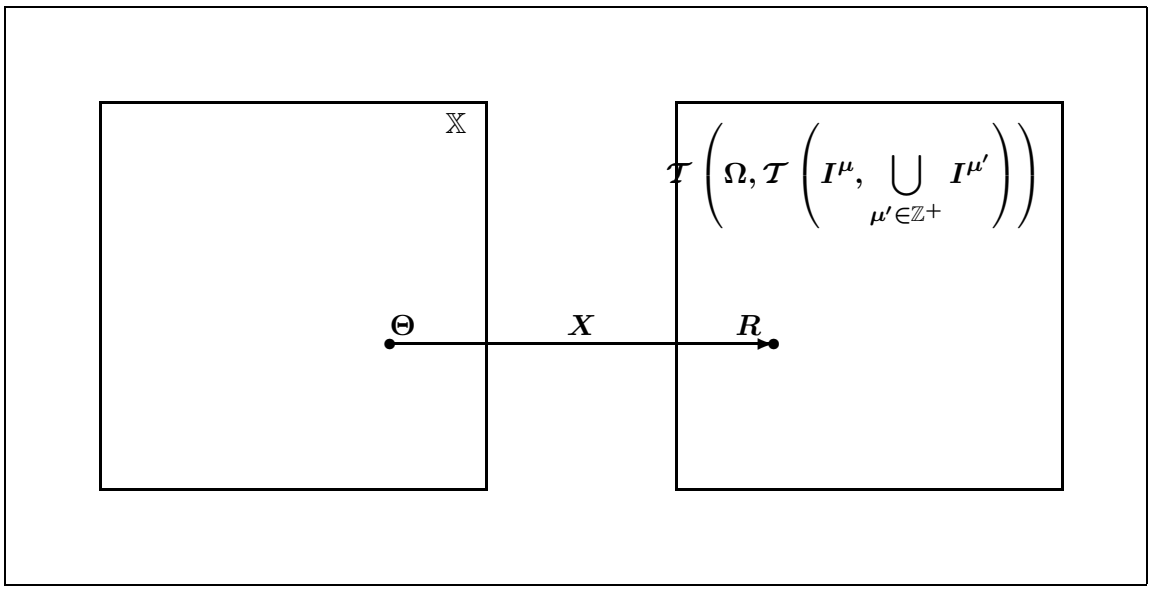

Figure A.3 The evolutionary operator X maps the exogenous parameter(s) Θ to the random population transformation R. The underlying sample space of R is Ω. Each of the possible population transformations acts on populations of size μ. The offspring population size $\mu' \in \mathbb{Z}^{+}$ may depend on Θ as well as the random event $\omega \in \Omega$ [132].

Definition 48 (Evolutionary operator): *Let I be a non-empty set (the individual space), $\mu \in \mathbb{Z}^{+}$ (the parent population size), $\mathbb{X}$ a set (the* parameter space*), and Ω a set (the* sample space*). A mapping*

$$X : \mathbb{X} \longrightarrow \mathcal{T}\left(\Omega, \mathcal{T}\left(I^{\mu}, \bigcup_{\mu' \in \mathbb{Z}^{+}} I^{\mu'}\right)\right) \tag{A.1}$$

is called an evolutionary operator*. The set of evolutionary operators in the form of Equation (A.1) is denoted* $\mathcal{EVOP}(I, \mu, \mathbb{X}, \Omega)$ *[132].* □

The notation X_Θ is used to represent the random population transformation $X(\Theta)$. The population transformation $X_\Theta(\omega)$ is also denoted X_Θ to maintain consistency with the notation of Bäck and Schwefel [13, 173]. In particular, the offspring population $[X_\Theta(\omega)](P)$ is denoted $X_\Theta(P)$. Finally, if X has no parameters, i.e. $X \in \mathcal{EVOP}(I, \mu, \{\}, \Omega)$ then the offspring population is denoted $X(P)$ [132].

Evolutionary algorithms and their operators are based off of biological concepts. Most EAs use variants of the *recombination*, *mutation*, and *selection* evolutionary operators. Recombination operators are the most general of the three. In general, recombination takes multiple individuals from the population to act as "parents" to yield new individuals (children). Merkle [132] presents the following definition for recombination:

Definition 49 (Recombination operator): *Let* $r \in \mathcal{EVOP}(I, \mu, \mathbb{X}, \Omega)$*. If there exist* $P \in I^\mu$*,* $\Theta \in \mathbb{X}$*, and* $\omega \in \Omega$ *such that at least one individual in the offspring population* $r_\Theta(P)$ *depends on more than one individual of* P *then* r *is called a* recombination operator*.* □

Mutation operators are used to maintain diversity in the population for a GA[1]. A low mutation rate is typically used so as to preserve "good" material (building blocks) that are present in the population. Mutation is dependent on only one population member and typically results in the modification of one to just a few bits within that population member. Merkle [132] defined mutation in a general way that does not assume the parent and child population sizes are identical:

Definition 50 (Mutation operator): *Let* $m \in \mathcal{EVOP}(I, \mu, \mathbb{X}, \Omega)$*. If for every* $P \in I^\mu$*, every* $\Theta \in \mathbb{X}$*, and every* $\omega \in \Omega$*, each individual in the offspring population* $m_\Theta(P)$ *depends on at most one individual of* P *then* m *is called a* mutation operator*.* □

Selection operators are used to choose members of one population for mating and inclusion into the subsequent population. Many different selection mechanisms exist including

[1] In other EAs the mutation operator(s) may act as the primary exploration operator(s)

tournament selection, roulette wheel, and elitism to name a few. The selection of a population member for placement into another population is typically based on the fitness of the population members compared. The following definition illustrates the selection operator [132]:

Definition 51 (Selection operator): *Let* $s \in \mathcal{EVOP}(I, \mu, \mathbb{X} \times \mathcal{T}(I, \mathbb{R}), \Omega)$*. If for every* $P \in I^{\mu}$*, every* $\Theta \in \mathbb{X}$*, and every fitness function* $\Phi : I \longrightarrow \mathbb{R}$*,* s *satisfies*

$$\mathbf{a} \in s_{(\Theta,\Phi)}(P) \Longrightarrow \mathbf{a} \in P \quad ,$$

then s *is called a* selection operator *[132].* □

A.2 Evolutionary Algorithm Notation and Formulations

Each evolutionary algorithm has an associated non-empty set I, the *individual space* of the algorithm. Each *individual* $\mathbf{a} \in I$ represents a candidate solution to the optimization problem. Merkle states that the representation scheme is formally defined by the *decoding function* [132].

Definition 52 (Decoding function): *Let* I *be a non-empty set (the* individual space*), and* $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ *(the* objective function*). If* $D : I \longrightarrow \mathbb{R}^n$ *is total, i.e. the domain of* D *is all of* I*, then* D *is called a* decoding function *[132].* □

The range of D determines the subset of $\mathbb{R}^n$ actually available for exploration by the evolutionary algorithm [132].

The fitness of an individual is an indication of the quality of the candidate solution represented by the individual. The *fitness function* is a mapping which provides this information. It is the fitness function which the evolutionary algorithm actually attempts to optimize and Merkle defines as [132]:

Definition 53 (Fitness function): *Let* I *be a non-empty set (the individual space),* $D : I \longrightarrow \mathbb{R}^n$ *(the decoding function),* $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ *(the objective function), and* $T_s : \mathbb{R} \longrightarrow \mathbb{R}$ *(the fitness scaling function). Then* $\Phi \triangleq T_s \circ f \circ D$ *is called a* fitness function *[132].* □

The objective function is determined by the mathematical function that one is trying to optimize. Since optimization problems can typically be expressed in a mathematical function, an EA is suited to find acceptable or feasible solutions to many problems[2] that exist today. The specification of the decoding function D and the fitness scaling function T_s are design issues. An important design criteria for the scaling function is that it preserve the partial ordering induced on the individual space by the decoding and objective functions [132].

Bäck's framework defines a collection of individuals as μ and in terms of population transformations, the μ individuals are represented by I^μ. The actual population transformation is represented by the following relationship: $T : I^\mu \rightarrow I^\mu$, where $\mu \in \mathbb{N}$ [13]. In some EAs, the resultant population size as the algorithm executes is not identical to the size of the initial population or even the population size from the previous phase; therefore, the population transformation should be represented as $T : I^\mu \rightarrow I^{\mu'}$, indicating succeeding populations may contain the same *or* different numbers of individuals. This framework also represents all population sizes, evolutionary operators, and parameters as *sequences* [132, 184]. This is due to the fact that different EAs use these factors in slightly different ways. The general algorithm thus recognizes and explicitly identifies this nuance. Looking to Bäck, Merkle and Van Veldhuizen for guidance, we see that an EA is defined as [13, 132, 135, 184]:

Definition 54 (Evolutionary Algorithm): *Let*

- *I be a non-empty set (the individual space),*
- *$\{\mu^{(i)}\}_{i\in\mathbb{N}}$ a sequence in $\mathbb{Z}^+$ (the parent population sizes),*
- *$\{\mu'^{(i)}\}_{i\in\mathbb{N}}$ a sequence in $\mathbb{Z}^+$ (the offspring population sizes),*
- *$\Phi : I \longrightarrow \mathbb{R}$ (a fitness function),*
- *$\iota : \bigcup_{i=1}^{\infty}(I^\mu)^{(i)} \longrightarrow \{\texttt{true}, \texttt{false}\}$ (the termination criterion),*
- *$\chi \in \{\texttt{true}, \texttt{false}\}$,*

[2]The reader is reminded that EAs are typically suited to solve NP-Complete problems and those problems that are currently impossible to solve on available computing platforms in a reasonable amount of time and in which optimized methods to find the global optima do not exist or are too time consuming to be feasible.

- *r a sequence $\{r^{(i)}\}$ of recombination operators*
  $r^{(i)} : \mathbb{X}_r^{(i)} \longrightarrow \mathcal{T}\left(\Omega_r^{(i)}, \mathcal{T}\left(I^{\mu^{(i)}}, I^{\mu'^{(i)}}\right)\right)$,
- *m a sequence $\{m^{(i)}\}$ of mutation operators*
  $m^{(i)} : \mathbb{X}_m^{(i)} \longrightarrow \mathcal{T}\left(\Omega_m^{(i)}, \mathcal{T}\left(I^{\mu'^{(i)}}, I^{\mu'^{(i)}}\right)\right)$,
- *s a sequence $\{s^{(i)}\}$ of selection operators*
  $s^{(i)} : \mathbb{X}_s^{(i)} \times \mathcal{T}(I, \mathbb{R}) \longrightarrow \mathcal{T}\left(\Omega_s^{(i)}, \mathcal{T}\left(\left(I^{\mu'^{(i)}+\chi\mu^{(i)}}\right), I^{\mu^{(i+1)}}\right)\right)$,
- $\Theta_r^{(i)} \in \mathbb{X}_r^{(i)}$ *(the* recombination parameters*),*
- $\Theta_m^{(i)} \in \mathbb{X}_m^{(i)}$ *(the* mutation parameters*), and*
- $\theta_s^{(i)} \in \mathbb{X}_s^{(i)}$ *(the* selection parameters*).*

Then the algorithm shown in Figure A.4 is called an Evolutionary Algorithm [132]. □

```
t := 0;
initialize P(0) := {a_1(0), ..., a_μ(0)} ∈ I^{μ^(0)};
while (ι({P(0), ..., P(t)}) ≠ true) do
    recombine: P'(t) := r^(t)_{Θ_r^(t)}(P(t));
    mutate: P''(t) := m^(t)_{Θ_m^(t)}(P'(t));
    select:
        if χ
            then P(t+1) := s^(t)_{(θ_s^(t),Φ)}(P''(t));
            else P(t+1) := s^(t)_{(θ_s^(t),Φ)}(P''(t) ∪ P(t));
        fi
    t := t+1;
od
```

Figure A.4 Evolutionary Algorithm Outline

The clear and concise definition of applicable EA terms and operators, along with a consistent mathematical formulation is presented. This provides the necessary background for understanding the multiobjective terminology and operators as well as a mathematical formulation.

A.3 Generic MOEA

A further motivation of conducting research into implicit and explicit BB-based MOEAs is to understand what constitutes a generic MOEA. The NFL Theorem [204] is valid and states that it is not possible to find a single algorithm, that outperforms every other algorithm, in the attempted solving of all classes of optimization problems. However an understanding of a generic MOEA aids researchers in their quest to solve MOPs. The development of a generic MOEA lends insight into workings of MOEAs in general. Through the development process, a researcher gains additional understanding of what comprises an MOEA and an understanding of the various operators and other mechanisms present in many of the aforementioned MOEAs. A generic MOEA should include a wide variety of operators, possibly all of the operators presented in the previous summary of MOEAs.

The performance of a generic MOEA could parallel the performance of each of the specific MOEAs discussed in this chapter as it would contain the same operators. This has the added benefit of allowing researchers to easily test new ideas with any combination of implemented operators as applied to different MOEA problem classes. The usefulness of this effort lies in the possibility of determining a guideline for which operator and parameter settings provide good solutions to different classes of MOPs. A generic MOEA also provides researchers the ability to test the combination of different operators found in other MOEAs on the same test bed MOEA. As researchers are continually developing new operators or modifications of existing operators, a generic MOEA would allow researchers to make a somewhat easy comparison of the general effect of new or modified operators. Currently researchers must compare their new or modified operator to other operators present in entirely different MOEAs or implement the comparison operators in his/her MOEA. A generic MOEA, that the MOEA community as a whole contributes new and modified operators to, allows for a single test bed and comparison. The contribution by numerous researchers to this MOEA would provide a continually evolving MOEA that could advance the state-of-the-art in the MOEA field as well as provide an easy mechanism to compare and contrast the performance of a variety of operators as applied to various classes of MOPs.

Appendix B. Additional MOP Details

Chapter IV presents a discussion of MOPs selected for inclusion in an unconstrained, constrained, NP-Complete and real-world MOP test suite. This appendix presents a more detailed discussion of the selected unconstrained MOPs as well as details of other MOP test suites and test function generators.

B.1 Unconstrained MOPs

Previous research into MOEA test suites and MOEA test suite generators was lacking until the late 1990s when Van Veldhuizen [189, 184, 188] discussed the issue of using standardized MOEA test suites to compare MOEAs. He was one of the first researchers to address the issue of MOP characteristics, their associated search spaces, and criteria for selecting an MOP as a good test problem. At the time of his research, he identified the problem with MOEA testing, MOEAs were not tested against MOPs with large search spaces or high dimensionality in the genotype or phenotype spaces and hence this was motivation for some of the research he conducted [184]. Other researchers have presented test suite MOPs and MOP test suite generators since then [43, 50, 52, 53, 44, 54, 189, 184, 188, 192]. Researchers use test suites to analyze the efficiency and effectiveness of MOEA approaches. The efficiency is the computational resources used and execution time whereas the effectiveness is the quality of the results obtained from the MOEA. The latter is typically addressed in the analysis of the results of an MOEA.

Unconstrained MOPs were identified from the literature and summarized in [184]. A subset of the 30 identified unconstrained MOPs were selected for use in a standardized MOEA test suite by Van Veldhuizen. All 30 of the MOPs he identified were not included in his test suite as many exhibit similar characteristics or do not meet the guidelines suggested for test suite problems and hence do not contribute to the proposed test suite. Each of the MOPs selected has different characteristics in terms of their genotypical and phenotypical solution sets. The main genotype characteristics analyzed were: connected, disconnected, symmetric, scalable, and the solution type, number of functions and number of constraints. The geometry, connected, disconnected, concave and/or convex properties

of the phenotype space were also looked at. Consideration of all of the aforementioned characteristics led to the development of the test suite presented in his work. Another development of a test suite reflects similar functionality [210]. Some of their "generated" test functions have similar characteristics to MOPs from Van Veldhuizen's test suite.

Those MOPs chosen for inclusion into Van Veldhuizen's test suite were selected based on an analysis of their genotypical and phenotypical characteristics previously described. The MOPs he included allow for a good comparison of different MOEAs when applied to a limited test suite. His selected test suite MOPs have varying characteristics in terms of their genotype and phenotype structures as well as P_{true} and PF_{true}.

Schaffer's MOP allows for the determination of an analytical solution for PF_{true}. The resultant PF_{true} set is a convex curve and its P_{true} set is a line in the genotype space. Fonseca's MOP was selected due to the problem formulation and the ability to add decision variables (scalability) without modifying the location or shape of PF_{true}. The resultant PF_{true} set is a concave curve. Poloni's MOP is a maximization problem and can be viewed as a more difficult MOP due to the characteristic of a two part disconnected Pareto front. Kursawe's MOP is nonsymmetric and contains three disconnected curves on the Pareto front. Deb's MOP is selected for its attribute of having four disconnected curves on the Pareto front. These five MOPs were selected for their varying characteristics and level of MOEA difficulty. This standardized test suite allows one to compare the results of their MOEA to other MOEAs. Without a standard test suite, one cannot compare the performance on an MOEA to other MOEAs.

These MOPs, selected by Van Veldhuizen, exhibit varying complexity levels and Pareto front (phenotype) characteristics such as concave, convexity, connected, disconnected, scalable, uniform and non-uniform distribution of points. Thus, each of the MOPs in his standard test suite were very carefully selected from a large variety of well known MOPs.

B.2 Constrained MOPs

Another constrained MOP (**MOP-CO**) with six constraints is that proposed by Osyczka and Kundu [152] with two objective functions and six genotype variables as follows:

Minimize $F = (f_1(\vec{x}), f_2(\vec{x}))$, where

$$\begin{aligned} f_1(x) &= -(25(x_1-2)^2 + (x_2-2)^2 + (x_3-1)^2 + (x_4-4)^2 + (x_5-1)^2) \\ f_2(x) &= x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2 \end{aligned}$$

Subject to

$$\begin{aligned} c_1(x) &= x_1 + x_2 - 2 \geq 0, \\ c_2(x) &= 6 - x_1 - x_2 \geq 0, \\ c_3(x) &= 2 + x_1 - x_2 \geq 0, \\ c_4(x) &= 2 - x_1 + 3x_2 \geq 0, \\ c_5(x) &= 4 - x_3 - 3)^2 - x_4 \geq 0, \\ c_6(x) &= (x_5 - 3)^2 + x_6 - 4 \geq 0, \\ & 0 \leq x_1, x_2, x_6 \leq 10, 1 \leq x_3, x_5 \leq 5, 0 \leq x_4 \leq 6 \end{aligned} \tag{B.1}$$

The PF_{true} is shown in figure B.1 reflecting six regions representing the intersection of specific constraints. Maintaining subpopulations at the different intersections is a difficult MOEA problem. The P_{true} solution values are $x_4 = 0$ and $x_6 = 0$ with the remaining variable values associated with each region presented in table B.1.

Table B.1: MOP-C5 P_{true} solution values, [152]

Region	x_1	x_2	x_3	x_5
AB	5	1	$(1, ..., 5)$	5
BC	5	1	$(1, ..., 5)$	1
CD	(4.06,...,5)	(0.68, ...,1)	1	1
DE	0	2	$(1, ..., 3.73)$	1
EF	$(0, ..., 1)$	$(2, ...1)$	1	1

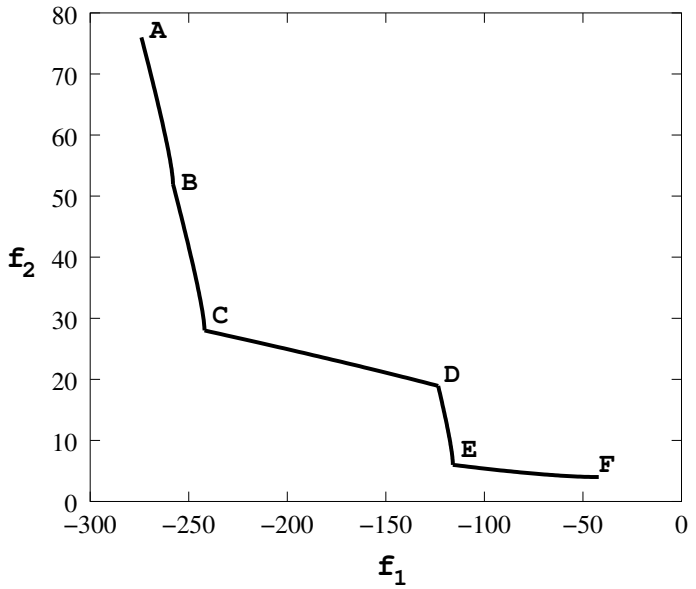

Figure B.1 MOP-C5 connected PF_{true} regions [152]

B.3 Test Function Generator

The concept of a creating a test function generator has also been identified in the MOEA literature by Deb [53]. Deb has presented work on a modified test suite and a test suite generator. The concept of this approach is to create a test suite of modifiable MOPs such that the engineer or scientist can change the complexity of each of the MOPs via one of the function parameters [52] (This is the same concept as the Tanaka function formulation presented). In [43], Deb presents generic two-objective functions as a potential test suite, shown in Equation (B.2). He selected these functions so that modifications to the functions in turn change the characteristics of the resulting Pareto front and can test the performance of the MOEA across MOPs of varying complexity. He states that convexity as well as discontinuity can be affected via the function h, convergence may be affected by modifying the g function to be multi-modal, deceptive, or contain other characteristics and diversity may be affected by choosing a non-linear or multi-dimensional function for f_1. He further presents an analysis of the parameter interactions present in these modifiable

functions.

$$\begin{aligned} \text{Minimize } f_1(\vec{x}) &= f_1(x_1, x_2, \ldots, x_m), \\ \text{Minimize } f_2(\vec{x}) &= g(x_{m+1}, \ldots, x_N) h(f_1(x_1, \ldots, x_m), g(x_{m+1}, \ldots, x_N)). \end{aligned} \tag{B.2}$$

Deb's test suite generator presented in Equation (B.2) is useful to the MOEA community but is somewhat complex to understand [43]. One must be very knowledgeable of functions in order to select a f, g and h function. Deb presents a number of functions to use for f, g, and h and which combinations of these functions yield MOPs of varying characteristics [43]. Problems have been identified with the test functions that Deb proposes. In some of Deb's test problems, the difficulty in finding PF_{true} is due to the specific discretization of the genotype space chosen versus the fact that PF_{local} exists [184]. While this problem exists, this does not mean that Deb's test suite in not useful. In fact, the functions that he presents can be used to evaluate MOEA performance. However it is important that one understand the MOPs used to test MOEA performance in order to make the correct analysis of the MOEA performance realized.

Deb proposes the test functions present in Table B.2 for use in an MOP test suite [43, 46].

Table B.2: Deb's MOEA Test Suite Functions [42, 43]

MOP	**Definition**	**Constraints**
ZDT1 convex PF_{true}	$F = (f_1(x_1), f_2(\vec{x}))$, where $f_1(x_1) = x_1,$ $f_2(\vec{x}) = g(1 - \sqrt{(f_1/g)})$ $g(\vec{x}) = 1 + 9\sum_{i=2}^{m} x_i/(m-1)$	$m = 30; 0 \leq x_i \leq 1$

Table B.2: (continued)

MOP	Definition	Constraints
ZDT2 nonconvex PF_{true}	$F = (f_1(x_1), f_2(\vec{x}))$, where $$\begin{aligned} f_1(x_1) &= x_1 \\ f_2(\vec{x}) &= g(1-(f_1/g)^2) \\ g(\vec{x}) &= 1+9\sum_{i=2}^{m} x_i/(m-1) \end{aligned}$$	$m = 30; 0 \leq x_i \leq 1$
ZDT3 noncont PF_{true}	$F = (f_1(x), f_2(\vec{x}))$, where $$\begin{aligned} f_1(x_1) &= x_1 \\ f_2(\vec{x}) &= g(1-\sqrt{(f_1/g)}-(f_1/g)\sin(10\pi f_1)) \\ g(\vec{x}) &= 1+9\sum_{i=2}^{m} x_i/(m-1) \end{aligned}$$	$m = 10, 0 \leq x_i \leq 1$
ZDT4 local PF_{true}	$F = (f_1(x), f_2(\vec{x}))$, where $$\begin{aligned} f_1(x_1) &= x_1 \\ f_2(\vec{x}) &= g(1-\sqrt{(f_1/g)}) \\ g(\vec{x}) &= 1+10(m-1)+\sum_{i=2}^{m}(x_i^2-10\cos(4\pi x_i)) \end{aligned}$$	$m = 10, 0 \leq x_1 \leq 1$
Deceptive MOP deceptive, unitation	$F = (f_1(x), f_2(\vec{x}))$, where $$\begin{aligned} f_1(x_1) &= 1+u(x_1) \\ f_2(\vec{x}) &= g/f_1 \\ g(\vec{x}) &= \sum_{i=2}^{m} v(u(x_i)) \end{aligned}$$	$m = 11, 0 \leq x_i \leq 1$ $v(u(x_i)) = 2 + u(x_i)$ if $u(x_i) < 5$ $= 1$ if $u(x_i) = 5$

Table B.2: (continued)

MOP	Definition	Constraints
ZDT6 nonuniform PF_{true}	$F = (f_1(x), f_2(\vec{x}))$, where $\begin{aligned} f_1(x_1) &= 1 - \exp(-4x_i)\sin^6(6\pi x_1) \\ f_2(\vec{x}) &= g(1 - (f_1/g)^2) \\ g(\vec{x}) &= 1 + 9((\sum_{i=2}^{m} x_i)/(m-1))^{0.25}) \end{aligned}$	$m = 10, 0 \leq x_i \leq 1$

Deb also proposes another test function generator for constrained problems [50]. The constrained test function generator is presented in Equation (B.3) and results in what he refers to as MOPs CTP2-7.

$$\begin{aligned} \text{Minimize } f_1(x) &= x_1 \\ \text{Minimize } f_2(x) &= g(x)\left(1 - \frac{f_1(x)}{g(x)}\right) \\ \text{Subject to} c(x) &= cos(\theta)(f_2(x) - e) - sin(\theta)f_1(x) \geq \\ & \quad a|sin(b\pi(sin(\theta)(f_2(x) - e) + cos(\theta)f_1(x))^c|^d \end{aligned} \tag{B.3}$$

Deb restricts x_1 to be in the interval $[0, 1]$ and bounds the other variables based on the selection of the $g(x)$ function. The six constraint parameters, θ, a, b, c, d, and e, are used for tuning the MOP [50, 48]. Deb varies the constraint parameters to yield MOPs of varying characteristics.

A problem exists with Deb's test suite generator, as well as many of the MOP test suites that exist. Many of the MOPs contain a low dimensionality in terms of the number of decision variables and objective functions. Low dimensionality MOPs present easy visualization of the Pareto Optimal solutions and the Pareto front but may not allow the MOEA the opportunity to perform as well as it could considering the restricted size of the phenotype space. Again, this is because an MOEA is assumed to be most effective for high-dimensionality problems, in the genotype and/or phenotype spaces. This illustrates the need for a test suite containing problems of high-dimensionality or increased complexity

in terms of the number of decision variables or objective functions or both. Additionally, there has not been a great deal of testing on discrete, high dimensional, or constrained MOPs. A test suite containing problems with these characteristics is proposed and used in this dissertation. Additionally, the use of real-world problems allows one to have some confidence that a particular instantiation of an MOEA will perform well on a certain class of real-world problem.

Deb's problems are presented for completeness but are not used for testing purposes in this effort. Many of his problems are similar to the MOPs selected for testing in this effort. The answer to the question if one best test suite exists, is a definite no.

Bibliography

1. Aguirre, Hernán E. and Kiyoshi Tanaka. Parallel Varying Mutation Genetic Algorithms. *Proceedings of the 2002 IEEE World Congress on Computational Intelligence*, 795–800. Piscataway, NJ: IEEE Service Center, May 2002.

2. Akl, Selim G. *The Design and Analysis of Parallel Algorithms*. Englewood Cliffs, NJ: Prentic Hall, 1989.

3. Alander, Jarmo T. An Indexed Bibliography of Genetic Algorithms: Years 1957-1993. Technical report, University of Vaasa, Department of Information Technology and Production Economics, Vaasa, Finland, 1994. Technical Report Report Series No. 94-1.

4. Alba, E. and J.M. Troya. Analyzing Synchronous and Asynchronous Parallel Distributed Genetic Algorithms. *Future Generation Computer Systems*, 17(4):451–465, 2001.

5. Alba, E. and J.M. Troya. Improving Flexibility and Efficiency by Adding Parallelism to Genetic Algorithms. *Statistics and Computing*, 12(2):91–114, 2002.

6. Alba, Enrique and Marco Tomassini. Parallelism and Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):443–462, 2002.

7. Alexandrov, V N and G M Megson. *Parallel Algorithms for Knapsack Type Problems*. New Jersey: World Scientific Publishing Company, 1999.

8. Anchor, Kevin P., Jesse B. Zydallis, Gregg H. Gunsch, and Gary B. Lamont. A Multi-Objective Evolutionary Approach for Detecting Computer Network Attacks. *Proceedings of the Second Workshop on Multiobjective Problem Solving from Nature(MPSN-II)*, 19–20. Granada, Spain, September 2002.

9. Anchor, Kevin P., Jesse B. Zydallis, Gregg H. Gunsch, and Gary B. Lamont. Detecting Computer Network Attacks Using a Multiobjective Evolutionary Programming Approach. *Journal of Information Warfare*, 2002. In Press.

10. Anchor, Kevin P., Jesse B. Zydallis, Gregg H. Gunsch, and Gary B. Lamont. Extending the Computer Defense Immune System: Network Intrusion Detection with a Multiobjective Evolutionary Programming Approach. Timmis, Jonathan and Peter J. Bentley, editors, *First International Conference on Artificial Immune Systems (ICARIS'2002)*, 12–21. University of Kent at Canterbury, UK, September 2002.

11. Anchor, Kevin P., Jesse B. Zydallis, Gregg H. Gunsch, and Gary B. Lamont. Different Multi-Objective Evolutionary Programming Approaches for Detecting Computer Network Attacks. *Second International Conference on Evolutionary Multi-Criterion Optimization*. Springer-Verlag. Lecture Notes in Computer Science, 2003 In-Press.

12. Arenas, M. G., Pierre Collet, A. E. Eiben, Márk Jelasity, J. J. Merlo, Ben Paechter, Mike Preuβ, and Marc Schoenauer. A Framework for Distributed Evolutionary Algorithms. Merelo Guervós, Juan Julián, Panagiotis Adamidis, Hans-Georg Beyer, José-Luis Fernández-Villaca nas, and Hans-Paul Schwefel, editors, *Parallel Problem*

Solving from Nature—PPSN VII, 665–675. Granada, Spain: Springer-Verlag. Lecture Notes in Computer Science No. 2439, September 2002.

13. Bäck, Thomas. *Evolutionary Algorithms in Theory and Practice*. New York: Oxford University Press, 1996.

14. Bäck, Thomas, David B. Fogel, and Zbigniew Michalewicz, editors. *Handbook of Evolutionary Computation*. Institute of Physics Publishing and Oxford University Press, 1997.

15. Baita, Flavio, Francesco Mason, Carlo Poloni, and Walter Ukovich. Genetic Algorithm with Redundancies for the Vehicle Scheduling Problem. Biethahn, J. and Volker Nissen, editors, *Evolutionary Algorithms in Management Applications*, 341–353. Berlin: Springer-Verlag, 1995.

16. Barr, Richard S., Bruce L. Golden, James P. Kelly, Mauricio G. C. Resende, and Jr. William R. Stewart. Designing and Reporting on Computational Experiments with Heuristic Methods. *Journal of Heuristics*, 1:9–32, 1995.

17. Ben-Tal, Aharon. Characterization of Pareto and Lexicographic Optimal Solutions. Fandel, G. and T. Gal, editors, *Multiple Criteria Decision Making Theory and Application*, volume 177 of *Lecture Notes in Economics and Mathematical Systems*, 1–11. Berlin: Springer-Verlag, 1980.

18. Buyya, Rajkumar, editor. *High Performance Cluster Computing: Architectures and Systems*, volume 1. NJ: Prentice Hall, 1999.

19. Cantú-Paz, Erick. Migration Policies, Selection Pressure, and Parallel Evolutionary Algorithms. Technical report, Department of Computer Science, University of Illinois, Urbana, IL, June 1999. IlliGAL Report No. 99015.

20. Cantú-Paz, Erick. *Efficient and Accurate Parallel Genetic Algorithms*. Boston: Kluwer Academic Publishers, 2000.

21. Carrico, Todd M. *ALP Overview: Background*, Last Download: April 06, 2000. Http://www.darpa.mil/iso/alp/Public_Access/Overview/index.htm.

22. Caswell, David J. *Active Processor Scheduling Using Evolutionary Algorithms*. Master's thesis, Air Force Institute of Technology, Wright Patterson AFB, December 2002. AFIT/GCS/ENG/02-36.

23. Caswell, David J. and Gary B. Lamont. Wire-Antenna Geometry Design with Multiobjective Genetic Algorithms. *Congress on Evolutionary Computation (CEC'2002)*, volume 1, 103–108. Piscataway, New Jersey: IEEE Service Center, May 2002.

24. Chandra, Rohit, Ramesh Menon, Leo Dagum, David Kohr, Dror Maydan, and Jeff McDonald. *Parallel Programming in OpenMP*. Morgan Kaufmann, 2000.

25. Chipperfield, A. J. and P. J. Fleming. Gas Turbine Engine Controller Design using Multiobjective Genetic Algorithms. Zalzala, A. M. S., editor, *Proceedings of the First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems : Innovations and Applications, GALESIA'95*, 214–219. Halifax Hall, University of Sheffield, UK: IEEE, September 1995.

26. Coello Coello, Carlos A. *List of References on Evolutionary Multiobjective Optimization*, Last Download: October 09, 2002. Http://www.lania.mx/ ccoello/EMOO/EMOObib.html.

27. Coello Coello, Carlos A. and Alan D. Christiansen. MOSES : A Multiobjective Optimization Tool for Engineering Design. *Engineering Optimization*, 31(3):337–368, 1999.

28. Coello Coello, Carlos A., Alan D. Christiansen, and Arturo Hernández Aguirre. Using a New GA-Based Multiobjective Optimization Technique for the Design of Robot Arms. *Robotica*, 16(4):401–414, July–August 1998.

29. Coello Coello, Carlos A. and Gregorio Toscano Pulido. A Micro-Genetic Algorithm for Multiobjective Optimization. Zitzler, Eckart, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, 126–140. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

30. Coello Coello, Carlos A. and Gregorio Toscano Pulido. Multiobjective Optimization using a Micro-Genetic Algorithm. Spector, Lee, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, 274–282. San Francisco, California: Morgan Kaufmann Publishers, 2001.

31. Coello Coello, Carlos A., David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. New York: Kluwer Academic Publishers, May 2002.

32. Cohon, Jared L. and David H. Marks. A Review and Evaluation of Multiobjective Programming Techniques. *Water Resources Research*, 11(2):208–220, 1975.

33. Corne, David W., Nick R. Jerram, Joshua D. Knowles, and Martin J. Oates. PESA-II: Region-based Selection in Evolutionary Multiobjective Optimization. Spector, Lee, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, 283–290. San Francisco, California: Morgan Kaufmann Publishers, 2001.

34. Corne, David W., Joshua D. Knowles, and Martin J. Oates. The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization. Schoenauer, Marc, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, Juan Julian Merelo, and Hans-Paul Schwefel, editors, *Proceedings of the Parallel Problem Solving from Nature VI Conference*, 839–848. Paris, France: Springer. Lecture Notes in Computer Science No. 1917, 2000.

35. Czyzak, P. and A. Jaszkiewicz. Pareto Simulated Annealing—A Metaheuristic Technique For Multiple-Objective Combinatorial Optimization. *Journal of Multi-Criteria Decision Analysis*, 7:34–47, 1998.

36. Day, Richard O. *A Multiobjective Approach Applied to the Protein Structure Prediction Problem*. Master's thesis, Air Force Institute of Technology, Wright Patterson AFB, March 2002. AFIT/GCS/ENG/02M-05.

37. Day, Richard O., Jesse B. Zydallis, and Gary B. Lamont. Competitive Template Analysis of the Fast Messy Genetic Algorithm When Applied to the Protein Structure Prediction Problem. *2002 International Conference on Computational Nanoscience and Nanotechnology*, 36–39. San Juan, Puerto Rico, April 2002.

38. Day, Richard O., Jesse B. Zydallis, and Gary B. Lamont. Solving the Protein Structure Prediction Problem Through a Multiobjective Genetic Algorithm. *2002 International Conference on Computational Nanoscience and Nanotechnology*, 32–35. San Juan, Puerto Rico, April 2002.

39. Day, Richard O., Jesse B. Zydallis, Gary B. Lamont, Steven R. Michaud, and Ruth Pachter. Genetic Algorithm Approach to Protein Structure Prediction With Secondary Structures. Giannakoglou, K.C., D.T. Tsahalis, J. Periaux, K.D. Papailiou, and T. Fogarty, editors, *Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, Proceedings of the EUROGEN 2001 Conference*, Theory and Engineering Applications of Computational Methods, 311–316. Athens, Greece: International Center for Numerical Methods in Engineering (CIMNE), September 2001.

40. Day, Richard O., Jesse B. Zydallis, Gary B. Lamont, and Ruth Pachter. A Genetic Algorithm Approach to Solving the Protein Structure Prediction Problem. *Proceedings of the Sixth International Conference on Knowledge-Based Intelligent Information & Engineering Systems*, 477–481. Crema, Italy, September 2002.

41. Day, Richard O., Jesse B. Zydallis, Gary B. Lamont, and Ruth Pachter. Fine Granularity and Building Block Sizes of the Parallel Fast Messy Genetic Algorithm. *Proceedings of the 2002 IEEE World Congress on Computational Intelligence*, 127–132. Piscataway, NJ: IEEE Service Center, May 2002.

42. Deb, Kalyanmoy. Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems. Technical Report CI-49/98, Dortmund: Department of Computer Science/LS11, University of Dortmund, Germany, 1998.

43. Deb, Kalyanmoy. Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems. *Evolutionary Computation*, 7(3):205–230, Fall 1999.

44. Deb, Kalyanmoy. *Multi-Objective Optimization Using Evolutionary Algorithms*. Chichester, UK: John Wiley & Sons, 2001.

45. Deb, Kalyanmoy. Multi-Objective Evolutionary Algorithms: State-of-the-art Techniques and Challenges. Tutorial for the 2002 World Congress on Computational Intelligence (WCCI 2002), May 2002.

46. Deb, Kalyanmoy, Samir Agrawal, Amrit Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. KanGAL report 200001, Indian Institute of Technology, Kanpur, India, 2000.

47. Deb, Kalyanmoy, Samir Agrawal, Amrit Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. Schoenauer, Marc, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, Juan Julian Merelo, and Hans-Paul Schwefel, editors, *Proceedings of the Parallel Problem Solving from Nature VI Conference*, 849–858. Paris, France: Springer. Lecture Notes in Computer Science No. 1917, 2000.

48. Deb, Kalyanmoy and Tushar Goyal. Controlled Elitist Non-dominated Sorting Genetic Algorithms for Better Convergence. KanGAL report 200004, Indian Institute of Technology, Kanpur, India, 2000.

49. Deb, Kalyanmoy and Tushar Goyal. Multi-Objective Evolutionary Algorithms for Engineering Shape Design. KanGAL report 200003, Indian Institute of Technology, Kanpur, India, 2000.

50. Deb, Kalyanmoy and T. Meyarivan. Constrained Test Problems for Multi-Objective Evolutionary Optimization. KanGAL report 200005, Indian Institute of Technology, Kanpur, India, 2000.

51. Deb, Kalyanmoy, A. Patrap, and S. Moitra. Mechanical Component Design for Multi-Objective Using Elitist Non-Dominated Sorting GA. KanGAL report 200002, Indian Institute of Technology, Kanpur, India, 2000.

52. Deb, Kalyanmoy, Amrit Pratap, and T. Meyarivan. Constrained Test Problems for Multi-objective Evolutionary Optimization. Zitzler, Eckart, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, 284–298. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

53. Deb, Kalyanmoy, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable Test Problems for Evolutionary Multi-Objective Optimization. Technical Report 112, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 2001.

54. Deb, Kalyanmoy, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable Multi-Objective Optimization Test Problems. *Congress on Evolutionary Computation (CEC'2002)*, volume 1, 825–830. Piscataway, New Jersey: IEEE Service Center, May 2002.

55. Deb, Kalyanmoy, Pawan Zope, and Abhishek Jain. Distributed Computing of Pareto-Optimal Solutions. KanGAL report 2002008, Indian Institute of Technology, Kanpur, India, 2002.

56. Doorly, D. J., J. Peir, and J-P. Oesterle. Optimisation of Aerodynamic and Coupled Aerodynamic-Structural Design Using Parallel Genetic Algorithms. *6th AIAA/NASA/USAF Multidisciplinary Analysis & Optimization Symposium*. Monterey, California, September 1996.

57. Ecker, Joseph G. and Michael Kupferschmid. *Introduction to Operations Research*. Florida: Krieger Publishing Company, 1991.

58. Ehrgott, Matthias. Approximation Algorithms For Combinatorial Multicriteria optimization problems. *International Transactions in Operational Research*, 7:5–31, 2000.

59. Ehrgott, Matthias and Xavier Gandibleux. An Annotated Bibliography of Multiobjective Combinatorial Optimization. Technical Report 62/2000, Fachbereich Mathematik, Universitat Kaiserslautern, Kaiserslautern, Germany, 2000.

60. El-Rewini, Hesham, Theodore G. Lewis, and Hesham H. Ali. *Task Scheduling in Parallel and Distributed Systems*. Prentice Hall, 1994.

61. Erickson, Mark, Alex Mayer, and Jeffrey Horn. The Niched Pareto Genetic Algorithm 2 Applied to the Design of Groundwater Remediation Systems. Zitzler, Eckart, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, 681–695. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

62. Fogel, David B., Mohamed A. El-Sharkawi, Xin Yao, Garry Greenwood, Hitoshi Iba, Paul Marrow, and Mark Shackleton, editors. *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*. IEEE Press, 2002.

63. Fonseca, Carlos, Jong-Hwan Kim, and A. Smith, editors. *Proceedings of the 2000 Congress on Evolutionary Computation CEC00*. La Jolla Marriott Hotel La Jolla, California, USA: IEEE Press, 6-9 July 2000.

64. Fonseca, Carlos M. and Peter J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. Forrest, Stephanie, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, 416–423. University of Illinois at Urbana-Champaign, San Mateo, California: Morgan Kauffman Publishers, 1993.

65. Fonseca, Carlos M. and Peter J. Fleming. An Overview of Evolutionary Algorithms in Multiobjective Optimization. Technical report, Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, U. K., 1994.

66. Fonseca, Carlos M. and Peter J. Fleming. Multiobjective Genetic Algorithms Made Easy: Selection, Sharing, and Mating Restriction. *Proceedings of the First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, 42–52. Sheffield, UK: IEE, September 1995.

67. Fonseca, Carlos M. and Peter J. Fleming. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, 3(1):1–16, Spring 1995.

68. Fonseca, Carlos M. and Peter J. Fleming. Multiobjective Optimization. Bäck, Thomas, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*, volume 1, C4.5:1–C4.5:9. Institute of Physics Publishing and Oxford University Press, 1997.

69. Fonseca, Carlos M. and Peter J. Fleming. Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms—Part II: A Application Example. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 28(1):38–47, 1998.

70. Fonseca, Carlos M. and Peter J. Fleming. Multiobjective Optimization. Bäck, Thomas, David B. Fogel, and Zbigniew Michalewicz, editors, *Evolutionary Computation*, volume 2, 25–37. Institute of Physics Publishing, 2000.

71. Gates, George H. *Predicting Protein Structure Using Parallel Genetic Algorithms*. Master's thesis, Air Force Institute of Technology, Wright Patterson AFB, March 1994. AFIT/GCS/ENG/94D-03.

72. Geist, Al, Adam Beguelin, and Jack Dongarra. *PVM:Parallel Virtual Machine: A User's Guide and Tutorial for Network Parallel Computing*. MIT Press, 1994.

73. Glover, Fred and Manuel Laguna. Tabu Search. Reeves, C., editor, *Modern Heuristic Techniques for Combinatorial Problems*. Oxford, England: Blackwell Scientific Publishing, 1993.

74. Goldberg, David E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, Massachusetts: Addison-Wesley Publishing Company, 1989.

75. Goldberg, David E., Kalyanmoy Deb, and James H. Clark. Genetic Algorithms, Noise, and the Sizing of Populations. Technical report, Department of General Engineering, University of Illinois, Urbana, IL, December 1991. IlliGAL Report No. 91010.

76. Goldberg, David E., Kalyanmoy Deb, and Jeffrey Horn. Massive Multimodality, Deception, and Genetic Algorithms. Technical report, Department of General Engineering, University of Illinois, Urbana, IL, April 1992. IlliGAL Report No. 92005.

77. Goldberg, David E., Kalyanmoy Deb, Hillol Kargupta, and Georges Harik. Rapid, Accurate Optimization of Difficult Problems Using Fast Messy Genetic Algorithms. Forrest, Stephanie, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, 56–64. San Mateo, CA: Morgan Kaufmann Publishers, July 1993.

78. Goldberg, David E., Bradley Korb, and Kalyanmoy Deb. Messy Genetic Algorithms: Motivation, Analysis, and First Results. *Complex Systems*, 3:493–530, 1989.

79. Goldberg, David E. and Kumara Sastry. A Practical Schema Theorem for Genetic Algorithm Design and Tuning. Spector, Lee, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, 328–335. San Francisco, California, USA: Morgan Kaufmann, 7-11 July 2001. ISBN 1-55860-774-9.

80. Goldberg, David E., Kumara Sastry, and Thomas Latoza. On The Supply Of Building Blocks. Spector, Lee, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, 336–342. San Francisco, California, USA: Morgan Kaufmann, 7-11 July 2001. ISBN 1-55860-774-9.

81. Golovkin, Igor E., Sushil J. Louis, and Roberto C. Mancini. Parallel Implementation of Niched Pareto Genetic Algorithm Code for X-ray Plasma Spectroscopy. *Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference*, 222–227. Las Vegas, Nevada, July 2000.

82. Golovkin, Igor E., Sushil J. Louis, and Roberto C. Mancini. Parallel Implementation of Niched Pareto Genetic Algorithm Code for X-ray Plasma Spectroscopy. *Congress on Evolutionary Computation (CEC'2002)*, volume 2, 1820–1824. Piscataway, New Jersey: IEEE Service Center, May 2002.

83. Golovkin, Igor E., Roberto C. Mancini, Sushil J. Louis, Richard W. Lee, and Lewis Klein. Multi-criteria Search and Optimization: an Application to X-ray Plasma Spectroscopy. *2000 Congress on Evolutionary Computation*, volume 2, 1521–1527. Piscataway, New Jersey: IEEE Service Center, July 2000.

84. Hanne, Thomas. Concepts Of A Learning Objected-Oriented Problem Solver LOOPS. *Proceedings of the 12th International Conference on Multiple Criteria Decision Making*, 330–339. Springer-Verlag, 1995.

85. Hanne, Thomas. On The Convergence Of Multiobjective Evolutionary Algorithms. *European Journal of Operational Research*, 117(3):553–564, 1999.

86. Hanne, Thomas. Global Multiobjective Optimization with Evolutionary Algorithms: Selection Mechanisms and Mutation Control. Zitzler, Eckart, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, 197–212. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

87. Hansen, Michael Pilegaard. *Metaheuristics For Multiple Objective Combinatorial Optimization*. Ph.D. thesis, Institute of Mathematical Modelling, Technical University of Denmark, March 1998.

88. Hansen, Michael Pilegaard and Andrzej Jaszkiewicz. Evaluating The Quality Of Approximations To The Non-Dominated Set. Technical Report IMM-REP-1998-7, Technical University of Denmark, March 1998.

89. Hiroyasu, Tomoyuki, Mitsunori Miki, and Shinya Watanabe. The New Model of Parallel Genetic Algorithm in Multi-Objective Optimization Problems—Divided Range Multi-Objective Genetic Algorithm—. *2000 Congress on Evolutionary Computation*, volume 1, 333–340. Piscataway, New Jersey: IEEE Service Center, July 2000.

90. Holland, John H. Outline for a Logical Theory of Adaptive Systems. *Journal of the Association for Computing Machinery*, 9(3):297–314, 1962.

91. Holland, John H. *Adaptation in Natural Artificial Systems*. Ann Arbor: University of Michigan Press, 1975.

92. Horn, Jeffrey. Multicriterion Decision Making. Bäck, Thomas, David Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*, volume 1, F1.9:1 – F1.9:15. IOP Publishing Ltd. and Oxford University Press, 1997.

93. Horn, Jeffrey and Nicholas Nafpliotis. Multiobjective Optimization using the Niched Pareto Genetic Algorithm. Technical Report IlliGAl Report 93005, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 1993.

94. Horn, Jeffrey, Nicholas Nafpliotis, and David E. Goldberg. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. *Proceedings of the First IEEE Conference*

on Evolutionary Computation, IEEE World Congress on Computational Intelligence, volume 1, 82–87. Piscataway, New Jersey: IEEE Service Center, June 1994.

95. Hu, Xiaohui and Russell Eberhart. Multiobjective Optimization Using Dynamic Neighborhood Particle Swarm Optimization. *Congress on Evolutionary Computation (CEC'2002)*, volume 2, 1677–1681. Piscataway, New Jersey: IEEE Service Center, May 2002.

96. *Institute for Operations Research and the Management Sciences*, Last Download: January 7, 2003. Http://www.informs.org.

97. Jackson, Richard H. F., Paul T. Boggs, Stephen G. Nash, and Susan Powell. Guidelines for Reporting Results of Computational Experiments. Report of the Ad Hoc Committee. *Mathematical Programming*, 49:413–425, 1991.

98. Jakob, W., M. Gorges-Schleuter, and C. Blume. Application of Genetic Algorithms to task planning and learning. Männer, R. and B. Manderick, editors, *Parallel Problem Solving from Nature, 2nd Workshop*, Lecture Notes in Computer Science, 291–300. Amsterdam: North-Holland Publishing Company, 1992.

99. Jaszkiewicz, Andrzej. On The Performance Of Multiple Objective Genetic Local Search On The 0/1 Knapsack Problem. A Comparative Experiment. Technical Report RA-002/2000, Institute of Computing Science, Poznan University of Technology, Poznań, Poland, July 2000.

100. Jaszkiewicz, Andrzej. On the Performance of Multiple-Objective Genetic Local Search on the 0/1 Knapsack Problem—A Comparative Experiment. *IEEE Transactions on Evolutionary Computation*, 6(4):402–412, August 2002.

101. Jiménez, Fernando and José L. Verdegay. Constrained Multiobjective Optimization By Evolutionary Algorithms. *Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems (EIS'98)*, 266–271. University of La Laguna, Tenerife, Spain, 1998.

102. Jin, Yaochu, Tatsuya Okabe, and Bernhard Sendhoff. Adapting Weighted Aggregation for Multiobjective Evolution Strategies. Zitzler, Eckart, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, 96–110. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

103. Johnson, Richard A. and Dean W. Wichern. *Applied Multivariate Statistical Analysis*. Upper Saddle River, NJ: Prentic Hall, fourth edition, 1998.

104. Jones, Brian R., William A. Crossley, and Anastasios S. Lyrintzis. Aerodynamic and Aeroacoustic Optimization of Airfoils via a Parallel Genetic Algorithm. *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, AIAA-98-4811*. AIAA, 1998.

105. Jozefowiez, Nicolas, Frédéric Semet, and El-Ghazali Talbi. Paralllel and Hybrid Models for Multi-Objective Optimization: Application to the Vehicle Routing Problem. Merelo Guervós, Juan Julián, Panagiotis Adamidis, Hans-Georg Beyer, José-Luis

Fernández-Villaca nas, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature—PPSN VII*, 271–280. Granada, Spain: Springer-Verlag. Lecture Notes in Computer Science No. 2439, September 2002.

106. Kadrovach, Anthony, Jesse B. Zydallis, and Gary B. Lamont. Use of Mendelian Pressure in a Multi-Objective Genetic Algorithm. *Proceedings of the 2002 IEEE World Congress on Computational Intelligence*, 962–967. Piscataway, NJ: IEEE Service Center, May 2002.

107. Kadrovach, B. Anthony, Steven R. Michaud, Jesse B. Zydallis, Gary B. Lamont, Barry Secrest, and David Strong. Extending the Simple Genetic Algorithm into Multi-Objective Problems via Mendelian Pressure. *2001 Genetic and Evolutionary Computation Conference. Workshop Program*, 181–188. San Francisco, California, July 2001.

108. Kargupta, Hillol. *SEARCH: Polynomial Complexity, And The Fast Messy Genetic Algorithm*. Ph.D. thesis, Department of Computer Science, University of Illinois, Urbana, IL, May 1995. IlliGAL Report No. 95008.

109. Kim, Jong-Hwan, editor. *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*. COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea: IEEE Press, 27-30 May 2001.

110. Kirley, Michael. MEA: A Metapopulation Evolutionary Algorithm for Multi-objective Optimisation Problems. *Proceedings of the Congress on Evolutionary Computation 2001 (CEC'2001)*, volume 2, 949–956. Piscataway, New Jersey: IEEE Service Center, May 2001.

111. Knjazew, D. and D. E. Goldberg. Large-Scale Permutation Optimization with the Ordering Messy Genetic Algorithm. Schoenauer, Marc, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, Juan Julian Merelo, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VI 6th International Conference*. Paris, France: Springer Verlag, September 16-20 2000. LNCS 1917.

112. Knowles, Joshua and David Corne. M-PAES: A Memetic Algorithm for Multiobjective Optimization. *2000 Congress on Evolutionary Computation*, volume 1, 325–332. Piscataway, New Jersey: IEEE Service Center, July 2000.

113. Knowles, Joshua and David Corne. On Metrics for Comparing Nondominated Sets. *Congress on Evolutionary Computation (CEC'2002)*, volume 1, 711–716. Piscataway, New Jersey: IEEE Service Center, May 2002.

114. Knowles, Joshua D. *Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization*. Ph.D. thesis, The University of Reading, Department of Computer Science, Reading, UK, January 2002.

115. Knowles, Joshua D. and David W. Corne. Assessing the Performance of the Pareto Archived Evolution Strategy. Wu, Annie S., editor, *Proceedings of the 1999 Genetic and Evolutionary Computation Conference. Workshop Program*, 123–124. Orlando, Florida, July 1999.

116. Knowles, Joshua D. and David W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

117. Knowles, Joshua D., David W. Corne, and Martin J. Oates. On the Assessment of Multiobjective Approaches to the Adaptive Distributed Database Management Problem. Schoenauer, Marc, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, Juan Julian Merelo, and Hans-Paul Schwefel, editors, *Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature (PPSN VI)*, 869–878. Berlin: Springer, September 2000.

118. Koopmans, Tjalling C. Analysis of Production as an Efficient Combination of Activities. Koopmans, Tjalling C., editor, *Activity Analysis of Production and Allocation, Cowles Commision Monograph No 13*, 33–97. New York, NY: John Wiley and Sons, 1951.

119. Kumar, Vipin, Ananth Grama, Anshul Gupta, and George Karypis. *Introduction to Parallel Computing: Design and Analysis of Algorithms*. New York: Benjamin/Cummings Publishing Company, Inc., 1994.

120. Langdon, W. B., E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, editors. *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*. New York: Morgan Kaufmann Publishers, 9-13 July 2002.

121. Laumanns, Marco, Günter Rudolph, and Hans-Paul Schwefel. Mutation Control and Convergence in Evolutionary Multi-Objective Optimization. *Proceedings of the 7th International Mendel Conference on Soft Computing (MENDEL 2001)*. Brno, Czech Republic, June 2001.

122. Laumanns, Marco, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. On the Convergence and Diversity-Preservation Properties of Multi-Objective Evolutionary Algorithms. Technical Report 108, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, May 2001.

123. Laumanns, Marco, Lothar Thiele, Eckart Zitzler, and Kalyanmoy Deb. Archiving with Guaranteed Convergence and Diversity in Multi-Objective Optimization. Langdon, W.B., E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M.A. Potter, A.C. Schultz, J.F. Miller, E. Burke, and N. Jonoska, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, 439–447. San Francisco, California: Morgan Kaufmann Publishers, July 2002.

124. Lee, Jongsoo and Prabhat Hajela. Parallel Genetic Algorithm Implementation in Multidisciplinary Rotor Blade Design. *Journal of Aircraft*, 33(5):962–969, September-October 1996.

125. Mahmoud, Qusay H. *Distributed Programming with Java*. Greenwich, CT: Manning Publications Company, 2000.

126. Mäkinen, R., P. Neittaanmäki, J. Periaux, M. Sefrioui, and J. Toivanen. Parallel Genetic Solution for Multiobjective MDO. Schiano, A., A. Ecer, J. Périaux, and N. Satofuka, editors, *Parallel CFD'96 Conference*, 352–359. Capri: Elsevier, 1996.

127. Mäkinen, R., P. Neittaanmäki, J. Périaux, and J. Toivanen. A Genetic Algorithm for Multiobjective Design Optimization in Aerodynamics and Electromagnetics. et al., K. D. Papailiou, editor, *Computational Fluid Dynamics '98, Proceedings of the ECCOMAS 98 Conference*, volume 2, 418–422. Athens, Greece: Wiley, September 1998.

128. Makinen, R., J. Periaux, and J. Toivanen. Multidisciplinary Shape Optimization in Aerodynamics and Electromagnetics Using Genetic Algorithms, 1999.

129. Marco, N., S. Lanteri, J.-A. Desideri, and J. Périaux. A Parallel Genetic Algorithm for Multi-Objective Optimization in Computational Fluid Dynamics. Miettinen, Kaisa, Marko M. Mäkelä, Pekka Neittaanmäki, and Jacques Périaux, editors, *Evolutionary Algorithms in Engineering and Computer Science*, chapter 22, 445–456. Chichester, UK: John Wiley & Sons, Ltd, 1999.

130. Mariano, Carlos E. and Eduardo F. Morales. Distributed Reinforcement Learning for Multiple Objective Optimization Problems. *2000 Congress on Evolutionary Computation*, volume 1, 188–195. Piscataway, New Jersey: IEEE Service Center, July 2000.

131. McClave, James T., P. George Benson, and Terry Sincich. *Statistics for Business and Economics*. Upper Saddle River, NJ: Prentice Hall, 7th edition, 1998.

132. Merkle, Laurence D. *Analysis of Linkage-Friendly Genetic Algorithms*. Phd Dissertation, AFIT/DS/ENG/96-11, Air Force Institute of Technology, Wright-Patterson AFB OH, 1996.

133. Merkle, Laurence D., George H. Gates Jr., and Gary B. Lamont. Scalability of an MPI-Based Fast Messy Genetic Algorithm. *Proceedings of the 1998 ACM Symposium on Applied Computing (SAC'1998)*, 386–393. Atlanta, GA: ACM Press, 1998.

134. Merkle, Laurence D., George H. Gates, Jr., Gary B. Lamont, and Ruth Pachter. Application of the Parallel Fast Messy Genetic Algorithm to the Protein Structure Prediction Problem. *Proceedings of the Intel Supercomputer Users' Group Users Conference*, 189–195, 1994.

135. Merkle, Laurence D. and Gary B. Lamont. A Random Function Based Framework for Evolutionary Algorithms. Bäck, Thomas, editor, *7th International Conference on Genetic Algorithms (ICGA)*, 105–112. Morgan Kaufmann, 1997.

136. Merriam-Webster OnLine: Collegiate Dictionary. *Evolution*, 2001. 8 Aug 01 http://www.webster.com/cgi-bin/dictionary.

137. Meunier, Herve, El-Ghazali Talbi, and Philippe Reininger. A Multiobjective Genetic Algorithm for Radio Network Optimization. *2000 Congress on Evolutionary Computation*, volume 1, 317–324. Piscataway, New Jersey: IEEE Service Center, July 2000.

138. Michalewicz, Zbigniew. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, third edition, 1996.

139. Michalewicz, Zbigniew and J. Arabas. Genetic Algorithms for the 0/1 Knapsack Problem. *Methodologies for Intelligent Systems (ISMIS '94)*, 134–143. Berlin: Springer, 1994.

140. Michalewicz, Zbigniew and David B. Fogel. *How to Solve It: Modern Heuristics*. Berlin: Springer, 2000.

141. Michalewicz, Zbigniew and Garish Nazhiyath. GENOCOP III A Coevolutionary Algorithm for Numerical Optimization Problems With Nonlinear Constraints. *1995 Congress on Evolutionary Computation*, 647–651. Piscataway, New Jersey: IEEE Service Center, 1995.

142. Michaud, Steven R. *Solving the Protein Structure Prediction Problem with Parallel Messy Genetic Algorithms*. Master's thesis, Air Force Institute of Technology, Wright-Patterson AFB, March 2001. AFIT/GCS/ENG/01M-06.

143. Michaud, Steven R., Jesse B. Zydallis, Gary Lamont, Paul K. Harmer, and Ruth Pachter. Protein Structure Prediction with EA Immunological Computation. *Proceedings of the Genetic and Evolutionary Computation Conference(GECCO-2001)*, 1367–1374. San Francisco, CA, July 2001.

144. Michaud, Steven R., Jesse B. Zydallis, Gary Lamont, and Ruth Pachter. Detecting Secondary Peptide Structures by Scaling a Genetic Algorithm. Laudon, Matthew and Bart Romanowicz, editors, *Proceedings of the First International Conference on Computational Nanoscience*, 29–32. Hilton Head, SC, March 2001.

145. Michaud, Steven R., Jesse B. Zydallis, David M. Strong, and Gary Lamont. Load Balancing Search Algorithms on a Heterogeneous Cluster of PCs. *Proceedings of the Tenth SIAM Conference on Parallel Processing for Scientific Computing (PP01)*. Portsmouth, VA, March 2001.

146. Mitchell, Melanie. *An Introduction to Genetic Algorithms*. MA: MIT Press, 1995.

147. Moore, Gordon E. Cramming Mmore Components onto Integrated Circuits. *Electronics*, 38(8):493–530, April 1965.

148. Murty, Katta G. *Operations Research Deterministic Optimization Models*. New Jersey: Prentice Hall, 1995.

149. Neter, John, William Wasserman, and Michael H. Kutner. *Applied Linear Statistical Models: Regression, Analysis of Variance, and Experimental Designs*. Boston, MA: Irwin, 3rd edition, 1990.

150. Okuda, Tamaki, Tomoyuki Hiroyasu, Mitsunori Miki, and Shinya Watanabe. DC-MOGA: Distributed Cooperation Model of Multi-Objective Genetic Algorithm. *PPSN/SAB Workshop on Multiobjective Problem Solving from Nature II (MPSN-II)*. Granada, Spain, September 2002.

151. Ortmann, Matthias and Wolfgang Weber. Multi-Criterion Optimization of Robot Trajectories with Evolutionary Strategies. *Proceedings of the 2001 Genetic and Evolutionary Computation Conference. Late-Breaking Papers*, 310–316. San Francisco, California, July 2001.

152. Osyczka, Andrzej and Sourav Kundu. A New Method To Solve Generalized Multicriteria Optimization Problems Using The Simple Genetic Algorithm. *Structural Optimization*, 10:94–99, 1995.

153. Pacheco, Peter. *Parallel Programming with MPI*. Vintage Books, 1996.

154. Paton, Raymond C. Principles of Genetics. Bäck, Thomas, David B. Fogel, and Zbigniew Michalewicz, editors, *Evolutionary Computation*, volume 1, 27–39. Institute of Physics Publishing, 2000.

155. Périaux, Jacques, Mourad Sefrioui, and Bertrand Mantel. RCS Multi-Objective Optimization of Scattered Waves by Active Control Elements Using GAs. *Proceedings of the Fourth International Conference on Control, Automation, Robotics and Vision (ICARCV'96)*. Singapore, 1996.

156. Petroski, Henry. *Engineers of Dreams: Great Bridge Builders and the Spanning of American*. Morgan Kaufmann Publishers, 1996.

157. Poloni, Carlo. Hybrid GA for Multi-Objective Aerodynamic Shape Optimization. Winter, G., J. Periaux, M. Galan, and P. Cuesta, editors, *Genetic Algorithms in Engineering and Computer Science*, 397–416. Chichester: Wiley & Sons, 1995.

158. Poloni, Carlo, Giovanni Mosetti, and Stefano Contessi. Multiobjective Optimization by GAs: Application to System and Component Design. *Computational Methods in Applied Sciences '96: Invited Lectures and Special Technological Sessions of the Third ECCOMAS Computational Fluid Dynamics Conference and the Second ECCOMAS Conference on Numerical Methods in Engineering*, 258–264. Chichester: Wiley, 1996.

159. Rardin, Ronald R. *Optimization in Operations Research*. New Jersey: Prentice Hall, 1st edition, 1998.

160. Ray, Tapabrata, Tai Kang, and Seow Kian Chye. Multiobjective Design Optimization by an Evolutionary Algorithm. *Engineering Optimization*, 33(3):399–424, 2001.

161. Rowe, Jon, Kevin Vinsen, and Nick Marvin. Parallel GAs for Multiobjective Functions. Alander, Jarmo T., editor, *Proceedings of the Second Nordic Workshop on Genetic Algorithms and Their Applications (2NWGA)*, 61–70. Vaasa, Finland: University of Vaasa, August 1996.

162. Rudolph, Günter. Evolutionary Search for Minimal Elements in Partially Ordered Finite Sets. Porto, V.W., N. Saravanan, D. Waagen, and A.E. Eiben, editors, *Evolutionary Programming VII, Proceedings of the 7th Annual Conference on Evolutionary Programming*, 345–353. Berlin: Springer, 1998.

163. Rudolph, Günter. On a Multi-Objective Evolutionary Algorithm and Its Convergence to the Pareto Set. *Proceedings of the 5th IEEE Conference on Evolutionary Computation*, 511–516. Piscataway, New Jersey: IEEE Press, 1998.

164. Rudolph, Günter. Evolutionary Search under Partially Ordered Fitness Sets. *Proceedings of the International NAISO Congress on Information Science Innovations (ISI 2001)*, 818–822. ICSC Academic Press: Millet/Sliedrecht, 2001.

165. Rudolph, Günter. A Partial Order Approach to Noisy Fitness Functions. *Proceedings of the Congress on Evolutionary Computation 2001 (CEC'2001)*, volume 1, 318–325. Piscataway, New Jersey: IEEE Service Center, May 2001.

166. Rudolph, Günter. Some Theoretical Properties of Evolutionary Algorithms under Partially Ordered Fitness Values. Fabian, Cs. and I. Intorsureanu, editors, *Proceedings of the Evolutionary Algorithms Workshop (EAW-2001)*, 9–22. Bucharest, Romania, January 2001.

167. Rudolph, Günter and Alexandru Agapie. Convergence Properties of Some Multi-Objective Evolutionary Algorithms. *Proceedings of the 2000 Conference on Evolutionary Computation*, volume 2, 1010–1016. Piscataway, New Jersey: IEEE Press, July 2000.

168. Sakawa, Masatoshi, Kosuke Kato, and Toshihiro Shibano. An Interactive Fuzzy Satisficing Method For Multiobjective Multidimensional 0-1 Knapsack Problems Through Genetic Algorithms. *Proceedings of the 1996 International Conference on Evolutionary Computation (ICEC'96)*, 243–246. 1996.

169. Schaffer, J. David. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. Ph.D. thesis, Vanderbilt University, 1984.

170. Schaffer, J. David. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, 93–100. Lawrence Erlbaum, 1985.

171. Schnecke, Volker and Oliver Vornberger. Hybrid Genetic Algorithms for Constrained Placement Problems. *IEEE Transactions on Evolutionary Computation*, 1(4):266–277, November 1997.

172. Schott, Jason R. *Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization*. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, May 1995.

173. Schwefel, Hans-Paul. *Evolution and Optimum Seeking*. New York: Wiley, 1995.

174. Skienna, Steven S. *The Algorithm Design Manual*. NY: Springer, 1998.

175. Spector, Lee, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*. San Francisco, California, USA: Morgan Kaufmann, 7-11 July 2001.

176. Srinivas, N. and Kalyanmoy Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, Fall 1994.

177. Stadler, W. Initiators of Multicriteria Optimization. Jahn, J. and W. Krabs, editors, *Recent Advances and Historical Development of Vector Optimization*, 3–47. Berlin: Springer-Verlag, 1986.

178. Stanley, Timothy J. and Trevor Mudge. A Parallel Genetic Algorithm for Multiobjective Microprocessor Design. Eshelman, Larry J., editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, 597–604. University of Pittsburgh, San Mateo, California: Morgan Kaufmann Publishers, July 1995.

179. Stender, Joachim, editor. *Parallel Genetic Algorithms, Theory and Applications*, volume 14 of *Frontiers in Artificial Intelligence and Applications*. The Netherlands: IOS Press, 1993.

180. Swartz, Stephen. ALP Pilot Problem and Derivation of Mathematical Model, 1999. Wright-Patterson AFB, Dayton, OH.

181. Talbi, El-Ghazali, Malek Rahoual, Mohamed Hakim Mabed, and Clarisse Dhaenens. A Hybrid Evolutionary Approach for Multicriteria Optimization Problems: Application to the Flow Shop. Zitzler, Eckart, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, 416–428. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

182. Tanaka, Masahiro, Hikaru Watanabe, Yasuyuki Furukawa, and Tetsuzo Tanino. GA-Based Decision Support System for Multicriteria Optimization. *Proceedings of the International Conference on Systems, Man, and Cybernetics*, volume 2, 1556–1561. Piscataway, NJ: IEEE, 1995.

183. Teich, Jürgen, Eckart Zitzler, and Shuvra S. Bhattacharyya. 3D Exploration of Software Schedules for DSP Algorithms. *7th International Workshop on Hardware/Software Codesign (CODES'99)*, 168–172. May 1999.

184. Van Veldhuizen, David A. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. Ph.D. thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, May 1999.

185. Van Veldhuizen, David A. and Gary B. Lamont. Evolutionary Computation and Convergence to a Pareto Front. Koza, John R., editor, *Late Breaking Papers at the Genetic Programming 1998 Conference*, 221–228. Stanford University, California: Stanford University Bookstore, July 1998.

186. Van Veldhuizen, David A. and Gary B. Lamont. Multiobjective Evolutionary Algorithm Research: A History and Analysis. Technical Report TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1998.

187. Van Veldhuizen, David A. and Gary B. Lamont. Genetic Algorithms, Building Blocks, and Multiobjective Optimization. Wu, Annie S., editor, *Proceedings of the 1999 Genetic and Evolutionary Computation Conference. Workshop Program*, 125–126. Orlando, Florida, July 1999.

188. Van Veldhuizen, David A. and Gary B. Lamont. MOEA Test Suite Generation, Design & Use. Wu, Annie S., editor, *Proceedings of the 1999 Genetic and Evolutionary Computation Conference. Workshop Program*, 113–114. Orlando, Florida, July 1999.

189. Van Veldhuizen, David A. and Gary B. Lamont. Multiobjective Evolutionary Algorithm Test Suites. Carroll, Janice, Hisham Haddad, Dave Oppenheim, Barrett Bryant, and Gary B. Lamont, editors, *Proceedings of the 1999 ACM Symposium on Applied Computing*, 351–357. San Antonio, Texas: ACM, 1999.

190. Van Veldhuizen, David A. and Gary B. Lamont. Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art. *Evolutionary Computation*, 8(2):125–147, 2000.

191. Van Veldhuizen, David A. and Gary B. Lamont. Multiobjective Optimization with Messy Genetic Algorithms. *Proceedings of the 2000 ACM Symposium on Applied Computing*, 470–476. Villa Olmo, Como, Italy: ACM, 2000.

192. Van Veldhuizen, David A. and Gary B. Lamont. On Measuring Multiobjective Evolutionary Algorithm Performance. *2000 Congress on Evolutionary Computation*, volume 1, 204–211. Piscataway, New Jersey: IEEE Service Center, July 2000.

193. Van Veldhuizen, David A., Brian S. Sandlin, , Robert M. Marmelstein, and Gary B. Lamont. Finding Improved Wire-Antenna Geometries with Genetic Algorithms. Fogel, David B., editor, *Proceedings of the 1998 International Conference on Evolutionary Computation*, 102–107. Piscataway, New Jersey: IEEE, 1998.

194. Van Veldhuizen, David A., Jesse B. Zydallis, and Gary B. Lamont. Issues in Parallelizing Multiobjective Evolutionary Algorithms for Real World Applications. *Proceedings of the 17th ACM Symposium on Applied Computing*, 595–602. Madrid, Spain: ACM Press, 2002.

195. Veldhuizen, David A. Van, Jesse B. Zydallis, and Gary B. Lamont. Considerations in Engineering Parallel Multiobjective Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 2002. In Press.

196. Veldhuizen, David A. Van, Jesse B. Zydallis, and Gary B. Lamont. Issues in Parallelizing Multiobjective Evolutionary Algorithms for Real World Applications. *2002 ACM Symposium on Applied Computing*, 595–602. Madrid, Spain: ACM, March 2002.

197. von Neumann, John and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton, New Jersey: Princeton University Press, 1944.

198. Wackerly, Dennis D., William Mendenhall III, and Richard L. Scheaffer. *Mathematical Statistics with Applications*. New York: Duxbury Press, 5th edition, 1996.

199. Wakefield, David J. *Identification of Preferred Operational Plan Force Mixes Using a Multiobjective Methodology to Optimize Resource Suitability and Lift Cost*. Master's thesis, Department of Operational Sciences. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, May 2001. AFIT/GLM/ENS/01M-24.

200. Wang, Gang, Erik D. Goodman, and William F. Punch III. Simultaneous Multi-Level Evolution. Technical Report GARAGe96-03-01, Michigan State University, 1996.

201. Whitley, Darrell. A Genetic Algorithm Tutorial. *Statistics and Computing*, 4:65–85, 1994.

202. Whitley, Darrell, David Goldberg, Erick Cantu-Paz, Lee Spector, Ian Parmee, and Hans-Georg Beyer, editors. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*. Las Vegas, Nevada, USA: Morgan Kaufmann, 10-12 July 2000.

203. Whitley, Darrell, K. Mathias, S. Rana, and J. Dzubera. Evaluating Evolutionary Algorithms. *Artificial Intelligence*, 85:245–276, 1996.

204. Wolpert, David H. and William G. Macready. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.

205. Wolsey, Laurence A. *Integer Programming*. New York: John Wiley and Sons, Wiley-Interscience, 1998.

206. Zhu, Zhong-Yao and Kwong-Sak Leung. An Enhanced Annealing Genetic Algorithm for Multi-Objective Optimization Problems. Langdon, W.B., E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M.A. Potter, A.C. Schultz, J.F. Miller, E. Burke, and N. Jonoska, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, 658–665. San Francisco, California: Morgan Kaufmann Publishers, July 2002.

207. Zitzler, Eckart. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. Ph.D. thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.

208. Zitzler, Eckart, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Technical Report 70, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, December 1999.

209. Zitzler, Eckart, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms on Test Functions of Different Difficulty. Wu, Annie S., editor, *Proceedings of the 1999 Genetic and Evolutionary Computation Conference. Workshop Program*, 121–122. Orlando, Florida, July 1999.

210. Zitzler, Eckart, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, Summer 2000.

211. Zitzler, Eckart, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, May 2001.

212. Zitzler, Eckart, Marco Laumanns, Lothar Thiele, Carlos M. Fonseca, and Viviane Grunert da Fonseca. Why Quality Assessment of Multiobjective Optimizers Is Difficult. Langdon, W.B., E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M.A. Potter, A.C. Schultz, J.F. Miller, E. Burke, and N. Jonoska, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, 666–673. San Francisco, California: Morgan Kaufmann Publishers, July 2002.

213. Zitzler, Eckart and Lothar Thiele. An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach. Technical Report 43, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, May 1998.

214. Zitzler, Eckart and Lothar Thiele. Multiobjective Optimization Using Evolutionary Algorithms—A Comparative Study. Eiben, A. E., editor, *Parallel Problem Solving from Nature V*, 292–301. Amsterdam: Springer-Verlag, September 1998.

215. Zitzler, Eckart and Lothar Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, November 1999.

216. Zydallis, Jesse B. and Gary B. Lamont. Solving of Discrete Multiobjective Problems Using an Evolutionary Algorithm with a Repair Mechanism. *Proceedings of the IEEE 2001 Midwest Symposium on Circuits and Systems*, volume 1, 470–473. IEEE, 2001.

217. Zydallis, Jesse B., Gary B. Lamont, and David A. Van Veldhuizen. Messy Genetic Algorithm Based Multi-Objective Optimization: A Comparative Statistical Analysis. *PPSN/SAB Workshop on Multiobjective Problem Solving from Nature (MPSN)*. Paris, France, September 2000.

218. Zydallis, Jesse B., Todd A. Sriver, and Gary B. Lamont. Multiobjective Evolutionary Algorithm Approach for Solving Integer Based Optimization Problems. Langdon, W.B., E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M.A. Potter, A.C. Schultz, J.F. Miller, E. Burke, and N. Jonoska, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, 1276. San Francisco, California: Morgan Kaufmann Publishers, July 2002.

219. Zydallis, Jesse B., David A. Van Veldhuizen, and Gary B. Lamont. A Statistical Comparison of Multiobjective Evolutionary Algorithms Including the MOMGA–II. Zitzler, Eckart, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, 226–240. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

220. Zydallis, Jesse B., David A. Van Veldhuizen, and Gary B. Lamont. Using Parallel Concepts In Multiobjective Evolutionary Algorithms. *PPSN/SAB Workshop on Multiobjective Problem Solving from Nature II (MPSN-II)*, 28–29. Granada, Spain, September 2002.

Vita

Captain Jesse B. Zydallis joined the Air Force as a cadet at Detachment 490, Reserve Officer Training Corps (ROTC) at the New Jersey Institute of Technology (NJIT), Newark, NJ. He was awarded a Bachelor of Science Degree in Computer Engineering by NJIT in January 1997. After his ROTC requirements were fulfilled he was commissioned a 2nd Lieutenant, in the US Air Force in January 1997. At this time Jesse was selected for a highly competitive opportunity to enter an AFIT civilian institution slot to receive a Master's Degree in Computer Engineering. He was brought on active duty in January 1997 and completed the Master's of Science degree in Computer Engineering from NJIT, with a specialization in parallel processing and networking, in January 1998. He then served as an Air Force Simulation Systems Engineer in the Air Vehicles Directorate at Wright Patterson AFB, OH. In September 1999 he had the opportunity, and took it, to pursue a Ph.D. whereupon he was assigned to the Air Force Institute of Technology (AFIT) at Wright Patterson AFB in Dayton, OH.

Jesse's military assignments include McGuire AFB, NJ (geographically separated for a M.S. degree at NJIT) and Wright-Patterson AFB, OH. Throughout his active duty career he has had the opportunity to travel to many of the states within the US and many countries within Europe for Test and Evaluation and presentation of research papers. He has published over 20 research articles resultant of the last two years of his dissertation research.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*
03-07-2003

2. REPORT TYPE
PhD Dissertation

3. DATES COVERED *(From - To)*
Oct 1999 - Mar 2003

4. TITLE AND SUBTITLE
Explicit Building-Block Multiobjective Genetic Algorithms: Theory, Analysis, and Development

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
JESSE B. ZYDALLIS, Capt, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Department of Electical and Computer Engineering,
Graduate School of Engineering and Management (AFIT/EN)
Air Force Institute of Technology
2950 Hobson Way
Wright Patterson AFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/DS/ENG/03-01

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Embedded Information Systems Engineering Branch
Attn: Dr Ewing
Air Force Research Laboratory
2241 Avionics Circle
Wright Patterson AFB OH 45422

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/IFTA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This dissertation research emphasizes explicit Building Block (BB) based MOEA performance and detailed symbolic representations. An explicit BB-based MOEA for solving constrained and real-world MOPs is developed, the Multiobjective Messy Genetic Algorithm II (MOMGA-II) to validate symbolic BB concepts. The MOMGA-II provides insight into solving difficult MOPs that is generally not realized through the use of implicit BB-based MOEA approaches. This insight is necessary to increase the effectiveness of all MOEA approaches. Parallel MOEA (pMOEA) concepts are presented to potentially increase MOEA computational efficiency and effectiveness. Communications in a pMOEA implementation is extremely important,
hence innovative migration and replacement schemes are detailed and tested. These parallel concepts support the development of the first explicit BB-based pMOEA, the pMOMGA-II. MOEA theory is also advanced through the derivation of the first MOEA population sizing theory. The sizing theory presented derives a conservative estimate of the MOEA population size necessary to achieve good results with a specified level of confidence. Validated results illustrate insight into building block phenomena, good efficiency, excellent effectiveness, and motivation for future research in the area of explicit BB-based MOEAs.

15. SUBJECT TERMS
Multiobjective Evolutionary Algorithms, Parallel Multiobjective Evolutionary Algortihms, Parallel Processing, Multiobjective Genetic Algorithms, Population Sizing, Multiobjective Evolutionary Algorithm Theory, MOEA, pMOEA, MOMGA-II, Building Blocks

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Gary B. Lamont, AD-24 (ENG)
U	U	U	UL	391	19b. TELEPHONE NUMBER *(Include area code)* 937-255-3636x4718

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18