Niching Ant Colony Optimisation

Daniel John Angus, BEng. BSci.

Complex Intelligent Systems Laboratory
Faculty of Information & Communication Technologies
Swinburne University of Technology
Melbourne, Australia

Submitted in partial fulfilment for the degree of
Doctor of Philosophy

July 2008

Abstract

Optimisation, in the mathematical sense, is the process of finding solutions to a problem such that one or many objectives are minimised or maximised. Optimisation problems are diverse in form, necessitating the need for many different optimisation algorithms. These algorithms can be defined in two categories: deterministic and non-deterministic algorithms. Deterministic algorithms usually have set execution schedules and are fairly exhaustive search methods. Non-deterministic algorithms use randomness and prove useful for problems where it may not be possible to execute a deterministic algorithm due to the size, or nature of the problem search space. In these cases a deterministic algorithm may take days or months to find an optimal solution, whereas a non-deterministic algorithm can usually find an approximate but still near-optimal solution in a matter of minutes or seconds.

Ant Colony Optimisation (ACO), a non-deterministic algorithm class, aims to mimic (and exploit) the behaviours of real ant colonies to solve real-world optimisation problems. ACO algorithms are a class of constructive heuristic algorithms, which build solutions to a given optimisation problem, one solution component at a time, according to a defined set of rules (heuristics), i.e. starting with an 'empty' solution add solution components until a complete solution is built. ACO algorithms are unique in this class by their use of past solutions in manipulating an artificial 'pheromone'. The pheromone being a measure associated to every unique solution component which reflects the estimated utility of this solution component. These pheromone values are used to bias solution construction by influencing the probability of a solution component being added to a growing solution based on the amount of pheromone it contains.

The Population-based Ant Colony Optimisation (PACO) algorithm is a recently developed ant-inspired algorithm which, unlike traditional ACO algorithms, maintains a finite population of solutions as well as pheromone information. It has been demonstrated to be an efficient optimisation algorithm when applied to a range of difficult single-objective, multi-objective and dynamic problem instances. In this thesis a review of existing PACO algorithms is offered and an identification of common features is used in the development of a Population-based ACO framework.

Using the new Population-based ACO framework, several new PACO algorithms imbued with a diversity preservation technique known as niching are defined. Niching has been used extensively in the field of Evolutionary Computation, but to the best knowledge of the author, has never been explicitly applied to an ACO algorithm per se. An empirical analysis of these novel

implementations is presented using a variety of single and multiple objective continuous function and combinatorial optimisation problems. These optimisation problems have been chosen since they demonstrate the advantages and disadvantages of adding niching to a PACO algorithm. To conclude, two of these new PACO algorithms are applied to a real-world optimisation problem.

Acknowledgments

A body of work such as this is never completed without a number of challenges and difficulties along the way. By far my greatest challenge was pursuing my studies without a scholarship for my first year. Although tricky, stressful and at times depressing this first year taught me many lessons on life, love and how fulfilling it can be to feed yourself for a week on $5. My ability to sense the free campus BBQ also became finely tuned! I wish to firstly thank those people who were there for me during this time who ensured that I continued to pursue my studies. Although I can't say it has been clear sailing from there, it has certainly been made easier thanks to the understanding and care of so many family members, friends and colleagues.

To my wife Katherine, your patience, care and understanding have made these last three years some of the most memorable in my life so far. You were with me from day one, and have stood beside me the whole way, for this I thank you.

Anne and Clive Angus, as my parents you have always encouraged me and given me the support I need. You have given me the strength and courage to pursue my dreams, for this I thank you. To Kate and Leigh Thomson, as my sister and brother in law you have always encouraged my studies and showed a genuine interest in my academic pursuits, thank you.

My colleagues at the Complex Intelligent Systems group, you have all helped me at some point and I thank you all for the many meaningful discussions shared over a beer. To my supervisor Prof. Tim Hendtlass, you inspired me from early in my undergraduate research days to pursue a PhD, and here I am now. You have played a most important role as my supervisor and mentor. Thank you for your trust, patience and understanding. To my co-supervisor Assoc. Prof. Marcus Randall your support is also appreciated. To my colleague and old room-mate Jason Brownlee, I have valued your input highly and you deserve an appropriate amount of thanks for your contribution. Your ideas about anything research related always seem to be light-years ahead of mine and I thank you for the positive contribution you have made to my research, and to my life in general.

Family and friends be they university colleagues past or present, high school chums, Rovers and the like, you have all been there at some point to offer encouragement, thank you all.

On a professional note I must thank Adam Deller for his help with Chapter 8 which was derived from his PhD work with the Centre for Astrophysics and Supercomputing at Swinburne University. I also acknowledge the Australian Government for providing me with an Australian Postgraduate Award for a partial component of my PhD program.

Declaration

I declare that this thesis contains no material that has been accepted for the award of any other degree or diploma and to the best of my knowledge contains no material previously published or written by another person except where due reference is made in the text of this thesis.

Daniel John Angus

Publications Arising from this Thesis

Angus, D. *Niching for Population-based Ant Colony Optimization*, Proceedings of the 2nd International IEEE Conference on e-Science and Grid Computing, Workshop on Biologically Inspired Optimisation Methods for Parallel and Distributed Architectures: Algorithms, Systems and Applications, 2006.

Angus, D. *Crowding Population-based Ant Colony Optimisation for the Multi-objective Travelling Salesman Problem*, Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM 2007), 2007, pp. 333–340.

Angus, D. *Population-based Ant Colony Optimisation for Multi-objective Function Optimisation*, Proceedings of the 3rd Australian Conference on Artificial Life (ACAL2007), 2007, pp. 232–244.

Contents

1 Introduction **1**

1.1 Background to the study 1

1.2 Research Motives and Contributions 2

1.3 Thesis Structure 3

2 Optimisation Problems **5**

2.1 Introduction 5

2.2 Combinatorial Optimisation 6

2.2.1 Travelling Salesman Problem 6

2.3 Continuous Function Optimisation 7

2.4 Modality 8

2.5 Multiple Objective Optimisation 8

2.5.1 Objective and Solution space 9

2.5.2 Pareto Optimality 10

2.5.3 Dominance 10

2.5.4 Solving Multiple Objective Optimisation Problems 11

2.5.5 Existing Algorithms 11

2.6 Choice of Problems 11

2.7 Performance Statistics 13

2.8 Chapter Summary 14

3 Ant Colony Optimisation **15**

3.1 Introduction 15

3.2 Early Development 15

3.2.1 Biological Inspiration 15

3.2.2 The Ant Systems Algorithm 17

3.2.3 Ant Colony Optimisation 18
3.3 The Ant Colony Optimisation Metaheuristic Framework 18
3.3.1 Pheromone mapping 20
3.3.2 Ants Generation and Activity 20
3.3.3 Pheromone Trail Evaporation 21
3.3.4 Daemon Actions 21
3.4 ACO Algorithms 21
3.4.1 Ant System for the Travelling Salesman Problem 21
3.4.2 Ant Colony Systems 23
3.4.3 $\mathcal{MAX}$-$\mathcal{MIN}$ Ant Systems 23
3.4.4 Rank Based Ant Systems 24
3.5 Salient Issues in ACO 24
3.5.1 Ant Inspired Algorithms 24
3.5.2 Local search 25
3.6 Chapter Summary 26

4 Population-based Ant Colony Optimisation **27**
4.1 Introduction 27
4.2 FIFO-Queue ACO 27
4.2.1 Algorithm Description 28
4.2.2 A Useful Population 29
4.3 PACO for Dynamic Problems 29
4.3.1 Population Update Strategies 30
4.4 PACO for Multi-objective Problems 30
4.5 ACO for Continuous Domains 31
4.6 Hardware Implementation of ACO 34
4.7 A Population-based ACO Meta-heuristic Framework 34
4.7.1 Process Level Organisation 35
4.7.2 Solution Storage and Maintenance 35
4.8 Similarity to Genetic Algorithms 36
4.8.1 Population Based Incremental Learning 36
4.8.2 Similarities and Differences 37
4.8.3 General Comments 38
4.9 Chapter Summary 38

5 Niching for PACO **41**

5.1 Introduction . . . 41

5.2 Evolutionary Computation . . . 41

5.2.1 Genetic Algorithm . . . 42

5.3 Niching in Evolutionary Computation . . . 43

5.3.1 Crowding . . . 44

5.3.2 Fitness Sharing . . . 44

5.3.3 Difference Measures . . . 46

5.3.4 Pros and Cons of Niching . . . 48

5.3.5 Sequential versus Parallel Niching . . . 48

5.4 Niching Ant Colony Optimisation . . . 49

5.4.1 Constraints . . . 49

5.4.2 Fitness Sharing Population-based Ant Colony Optimisation . . . 49

5.4.3 Crowding Population-based Ant Colony Optimisation . . . 50

5.4.4 Alternative ACO Diversity Preservation Methods . . . 50

5.5 Chapter Summary . . . 52

6 Single Objective Optimisation **53**

6.1 Introduction . . . 53

6.2 Niching PACO for the Travelling Salesman Problem . . . 54

6.2.1 Algorithm Details . . . 54

6.2.2 Results and Discussion: Crown Problem . . . 55

6.2.3 Results and Discussion: Standard Problems . . . 56

6.2.4 Summary . . . 63

6.3 Niching PACO for Continuous Function Optimisation . . . 63

6.3.1 Algorithm Details . . . 63

6.3.2 Test Problems Used . . . 67

6.3.3 Performance Metrics . . . 77

6.3.4 Results and Discussion: Sensitivity Analysis . . . 78

6.3.5 Results and Discussion: Quantitative Analysis . . . 86

6.3.6 Summary . . . 91

6.4 Chapter Summary . . . 93

7 Multiple Objective Optimisation **95**

7.1 Introduction . . . 95
7.2 Existing Algorithms . . . 96
7.3 Performance Metrics . . . 99
7.3.1 Summary Attainment Surface . . . 99
7.3.2 Hypervolume . . . 100
7.3.3 Epsilon Indicator . . . 100
7.3.4 Use of Performance Metrics . . . 101
7.4 Niching PACO for the Multiple Objective Travelling Salesman Problem . . . 101
7.4.1 Algorithm Details . . . 102
7.4.2 Test Problems . . . 103
7.4.3 Results and Discussion . . . 104
7.4.4 Summary . . . 118
7.5 Niching PACO for Multi-objective Function Optimisation . . . 118
7.5.1 Algorithm Details . . . 119
7.5.2 Test Problems . . . 121
7.5.3 Results and Discussion . . . 124
7.5.4 Summary . . . 140
7.6 Chapter Summary . . . 140

8 Case Study: Antenna Correlation 143

8.1 Problem Details . . . 144
8.1.1 Problem Description . . . 144
8.1.2 Objective Function . . . 146
8.1.3 Problem Data . . . 146
8.2 Experiment One: Naive Optimisation . . . 147
8.2.1 Experiment Description . . . 147
8.2.2 Results and Discussion . . . 148
8.3 Experiment Two: Problem Decomposition . . . 153
8.3.1 Experiment Description . . . 153
8.3.2 Results and Discussion . . . 153
8.4 Experiment Three: Application of Heuristics . . . 157
8.4.1 Experiment Description . . . 157
8.4.2 Results and Discussion . . . 157
8.5 Final Solution Analysis . . . 162

8.6 Chapter Summary 164

9 Summary and Final Remarks **165**

9.1 Summary 165

9.2 Contributions 166

9.3 Future Work 167

References **169**

10 Appendix **183**

List of Figures

2.1 Optimal tour of the PCB3038 TSP 7

2.2 Illustrations of different Pareto fronts plotted in the objective space. 10

3.1 Double bridge experimental setup 16

3.2 Graphical reproduction of the double bridge experiment using paths of equal length. 17

3.3 Graphical reproduction of the double bridge experiment using paths of unequal length. 17

3.4 Process organisation of the Ant Colony Optimisation Metaheuristic Framework . 19

3.5 Early development of the field of ACO 19

4.1 An illustration of the new solution creation procedure of ACOCD. 33

4.2 Process organisation of original ACO framework versus the PACO framework . 35

5.1 Generic algorithmic structure of Evolutionary Computation 42

6.1 Average number of similar edges amongst the best 100 and a randomly selected 100 solutions to the Burma14 TSP 55

6.2 Crown TSP problem visual representation 56

6.3 Optimal solutions to Crown problem 56

6.4 Occurrence of the two optimal paths over time for a single experiment run of the Crown TSP. 57

6.5 An illustration of the new solution creation procedure for FSPACO-CFO and CPACO-CFO. 66

6.6 Example of standard deviation used for new solution creation in FSPACO-CFO and CPACO-CFO. 67

6.7 Three Pot Holes function defined in two dimensions. 69

6.8 Ackley's function defined in two dimensions. 70

6.9 Branin's R-Cos function defined in two dimensions. 71

6.10 Himmelblau's function defined in two dimensions. 72
6.11 Rastrigin's function defined in two dimensions. 73
6.12 Schaffer's function defined in two dimensions. 74
6.13 Schwefel's function defined in two dimensions. 75
6.14 Shekel's foxholes function defined in two dimensions. 76
6.15 Plots indicating the effect on the max-peak-like performance metric when the niche radius and history exponent of FSPACO-CFO are varied. 79
6.16 Plots indicating the effect on the max-peak-like performance metric when the crowding window size and history exponent of CPACORank-CFO are varied. . . 80
6.17 Plots indicating the effect on the max-peak-like performance metric when the crowding window size and history exponent of CPACOQuality-CFO are varied. 81
6.18 Plots indicating the effect on the max-peak-like performance metric when the crowding window size of CPACOUnity-CFO is varied. 82
6.19 Example highlighting the effect of replacement error. 83
6.20 Results of CFO quantitative analysis 88
6.20 Results of CFO quantitative analysis 89
6.21 Results of CFO analysis as number of dimensions increases 92
6.22 Results from Three Pot Holes defined in 100 dimensions with a varied number of function evaluations. 93

7.1 Example of the resultant ranks assigned to a set of solutions using the NSGA-II dominance ranking procedure 97
7.2 Example of two hypervolumes created from two different sets, A and B. In this example A would be better than B since A's hypervolume is larger. 100
7.3 1% (best) attainment surface for kroA100 and kroB100 using PACO-MO & CPACO-MOTSP 105
7.4 50% (average) attainment surface for kroA100 and kroB100 using PACO-MO & CPACO-MOTSP 105
7.5 1% (best) attainment surface for kroC100 and kroD100 using PACO-MO & CPACO-MOTSP 106
7.6 50% (average) attainment surface for kroC100 and kroD100 using PACO-MO & CPACO-MOTSP 106
7.7 1% (best) attainment surface for kroA150 and kroB150 using PACO-MO & CPACO-MOTSP 107
7.8 50% (average) attainment surface for kroA150 and kroB150 using PACO-MO & CPACO-MOTSP 107

7.9 1% (best) attainment surface for kroA200 and kroB200 using PACO-MO & CPACO-MOTSP 108

7.10 50% (average) attainment surface for kroA200 and kroB200 using PACO-MO & CPACO-MOTSP 108

7.11 Example of sampling behaviour of PACO-MO 110

7.12 Example of sampling behaviour of CPACO-MOTSP 110

7.13 1% (best) attainment surface for CPACO-MOTSP applied to kroA100, kroB100, kroC100 & kroD100 (isolating KroA100 & KroB100) with 100,000, 250,000 and 500,000 solution evaluations. 111

7.14 50% (average) attainment surface for CPACO-MOTSP applied to kroA100, kroB100, kroC100 & kroD100 (isolating KroA100 & KroB100) with 100,000, 250,000 and 500,000 solution evaluations. 111

7.15 1% (best) attainment surface for CPACO-MOTSP applied to kroA100, kroB100, kroC100 & kroD100 (isolating KroA100 & KroB100) with 100,000, 250,000 and 500,000 solution evaluations. 112

7.16 50% (average) attainment surface for CPACO-MOTSP applied to kroA100, kroB100, kroC100 & kroD100 (isolating KroA100 & KroB100) with 100,000, 250,000 and 500,000 solution evaluations. 112

7.17 1% (best) attainment surface for CPACO-MOTSP applied to kroA100, kroB100, kroC100 & kroD100 (*CPACO_ABCD*), and CPACO-MOTSP applied to only kroA100 & kroB100 (*CPACO_AB*) 113

7.18 50% (average) attainment surface for CPACO-MOTSP applied to kroA100, kroB100, kroC100 & kroD100 (*CPACO_ABCD*), and CPACO-MOTSP applied to only kroA100 & kroB100 (*CPACO_AB*) 113

7.19 1% (best) attainment surface for CPACO-MOTSP applied to kroA100, kroB100, kroC100 & kroD100 (*CPACO_ABCD*), and CPACO-MOTSP applied to only kroC100 & kroD100 (*CPACO_CD*) 114

7.20 50% (average) attainment surface for CPACO-MOTSP applied to kroA100, kroB100, kroC100 & kroD100 (*CPACO_ABCD*), and CPACO-MOTSP applied to only kroC100 & kroD100 (*CPACO_CD*) 114

7.21 Example of selection probability using only pheromone information. 117

7.22 Example of selection probability using pheromone and heuristic information. . . 118

7.23 Attainment surfaces generated from 50 runs of PACO-MOFO and NSGA-II applied to MOP1 125

7.24 Attainment surfaces generated from 50 runs of PACO-MOFO and NSGA-II applied to MOP2 126

7.25 Attainment surfaces generated from 50 runs of PACO-MOFO and NSGA-II applied to MOP3 127

7.26 Attainment surfaces generated from 50 runs of PACO-MOFO and NSGA-II applied to MOP4 . . . 128

7.27 Attainment surfaces generated from 50 runs of PACO-MOFO and NSGA-II applied to MOP6 . . . 129

7.28 Distribution of final population of PACO-MOFO and NSGA-II on MOP2 and MOP3 . . . 131

7.28 Distribution of final population of PACO-MOFO and NSGA-II on MOP4 and MOP6 . . . 132

7.29 Attainment surfaces generated from 50 runs of PACO-MOFO and NSGA-II applied to the benchmark function, MOP2, in 2, 5, 10, 15 and 20 dimensions . . . 133

7.30 (cont'd) Attainment surfaces generated from 50 runs of PACO-MOFO and NSGA-II applied to the benchmark function, MOP4, in 5, 10, 15 and 20 dimensions . . . 134

7.30 (cont'd) Attainment surfaces generated from 50 runs of PACO-MOFO and NSGA-II applied to the benchmark function, MOP4, in 5, 10, 15 and 20 dimensions . . . 135

7.31 Attainment surfaces generated from 50 runs of PACO-MOFO and NSGA-II applied to MOP5 . . . 138

7.32 Attainment surfaces generated from 50 runs of PACO-MOFO and NSGA-II applied to MOP7 . . . 139

8.1 The observed residual delay for one observing band of a baseline (Parkes and ATCA) tracked over a 24 hour period taken from data set STA-131AV. . . . 145

8.2 Arrangement of baselines for the STA-131AU and STA-131AV datasets. . . . 147

8.3 Figures illustrating 100 interesting features of two known problems and one unknown problem. . . . 149

8.4 Results (RMS error value) presented from 100 repeats of the Crowding PACO algorithms and the control algorithm (Crowding Genetic Algorithm) applied to the 35 Dimension problems with each algorithm allowed 100,000 solution evaluations per experimental run. . . . 151

8.5 Figures illustrating possible problem compositions. . . . 154

8.6 Results obtained by plotting the predicted (modelled) residual delays against the observed residual delays. . . . 155

8.6 Results obtained by plotting the predicted (modelled) residual delays against the observed residual delays. . . . 156

8.7 Figure indicating the performance of test and control algorithms on 35D, 28D and 7D variants of the STA-131AU and STA-131AV problems using a heuristic. . . . 159

8.8 Results (RMS error value) presented from 100 repeats of the variable search strategy versus the fixed search strategy applied to the 4×7D problem with each algorithm allowed 1,000,000 solution evaluations per experimental run. . . . 161

List of Tables

2.1 5 city TSP represented using 2D Euclidean coordinates 6

2.2 5 city TSP edge weight matrix 7

3.1 Research Papers published before the formalisation of ACO 18

5.1 Example candidate solution 1 47

5.2 Example candidate solution 1 adjacency matrix 47

5.3 Example candidate solution 2 47

5.4 Example candidate solution 2 adjacency matrix 47

5.5 AND operation applied to solution 1 & 2 adjacency matrices 47

6.1 Statistical comparison of FSPACO-TSP, CPACO-TSP, MMAS and PACO on various benchmark TSP. 58

6.2 Parameter settings used in testing of the TSP 59

6.3 Path lengths of solutions contained in the population of CPACO-TSP (with a large crowding window size) after initialisation and the final population after 1000 iterations for Berlin52. 61

6.4 Path lengths of solutions contained in the population of CPACO-TSP (with a small crowding window size) after initialisation and the final population after 1000 iterations for Berlin52. 62

6.5 Path lengths of solutions contained in the population of CPACO-TSP (with a medium crowding window size) after initialisation and the final population after 10000 iterations for Eil101. 62

6.6 Parameter settings used for sensitivity analysis 78

6.7 Effect of varying crowding size and history exponent on Shekel's foxholes 85

6.8 Parameter settings used for quantitative analysis 87

7.1 Algorithm parameter settings for Bi-objective TSP Problems 104

7.2 Statistical comparison of PACO-MOFO and PACO-MO for various two-objective TSP . . . 109

7.3 Average final population size of PACO-MO versus static population size of CPACO-MOTSP for all test problems . . . 116

7.4 Algorithm parameter settings for MOFO problems . . . 124

7.5 Statistical comparison of PACO-MOFO and NSGA-II for MOP1, MOP2, MOP3, MOP4 and MOP6 defined in two dimensions . . . 124

7.6 Statistical comparison of PACO-MOFO and NSGA-II for MOP2 . . . 130

7.7 Statistical comparison of PACO-MOFO and NSGA-II for MOP4 . . . 136

7.8 Statistical comparison of PACO-MOFO and NSGA-II for MOP5 and MOP7 . . . 137

8.1 The default range of the search space per individual problem dimension for all datasets. . . . 146

8.2 Quantitative comparison of the Random, CGA and CPACO algorithms applied to the STA-131AU and STA-131AV problems. . . . 150

8.3 Parameter settings of test and control algorithms . . . 150

8.4 Statistical comparison of PACO-MOFO and NSGA-II for STA-V131AU and STA-V131AV . . . 152

8.5 Parameter settings of the multiple objective algorithms . . . 152

8.6 The average and standard deviation of the best multiple objective solutions when evaluated using the single objective cost routine. . . . 153

8.7 Quantitative comparison of the 35D, 28D and 7D variants of the STA-131AU and STA-131AV problems. . . . 155

8.8 The magnitude of the search space per individual problem dimension for the STA-131AU dataset after execution of the heuristic routine. . . . 158

8.9 The magnitude of the search space per individual problem dimension for the STA-131AV dataset after execution of the heuristic routine. . . . 158

8.10 Quantitative comparison of the 35D, 28D and 7D variants of the STA-131AU and STA-131AV problems using a heuristic. . . . 158

8.11 Quantitative comparison of the fixed and variable search strategies applied to the STA-131AU and STA-131AV problems. . . . 162

8.12 Positional errors obtained for the STA-131AV dataset using an alternative (but more complex) geodesy model. . . . 163

8.13 Positional errors obtained for the STA-131AV dataset using the 4×7D problem decomposition, heuristic search space reduction and relocation technique from Sec. 8.4.2. . . . 163

8.14 Magnitude of difference between the positional errors reported in Tab. 8.12 and Tab. 8.13. 163

10.1 Statistical significance of results for single objective TSP. 183

10.1 Statistical significance of results for single objective TSP. 184

10.2 Statistical significance of results for Schaffer's Function. 185

10.3 Statistical significance of results for Schwefel's Function. 186

10.4 Statistical significance of results for Himmelblau's Function. 187

10.5 Statistical significance of results for Branin's R-Cos Function. 188

10.6 Statistical significance of results for 35D Antenna Problems. 189

10.7 Statistical significance of results for 35D, 28D and 4×7D Antenna Problems. . . . 189

10.8 Statistical significance of results for 35D, 28D and 4×7D Antenna Problems using heuristics. 189

Commonly Used Acronyms

ACO	Ant Colony Optimisation
ACOCD	Ant Colony Optimisation for Continuous Domains
ACS	Ant Colony Systems
AS	Ant Systems
CFO	Continuous Function Optimisation
CGA	Deterministic Crowding Genetic Algorithm
CPACO	Crowding Population-based Ant Colony Optimisation
DE	Differential Evolution
EC	Evolutionary Computation
EDA	Estimation of Distribution Algorithm
ES	Evolutionary Strategies
FSPACO	Fitness Sharing Population-based Ant Colony Optimisation
GA	Genetic Algorithm
GP	Genetic Programming
MMAS	$\mathcal{MAX} - \mathcal{MIN}$ Ant Systems
MOEA	Multiple Objective Evolutionary Algorithm
MOFO	Multiple Objective Function Optimisation
MOGA	Multiple Objective Genetic Algorithm
MOO	Multiple Objective Optimisation
MOTSP	Multiple Objective Travelling Salesman Problem
NPGA	Niched Pareto Genetic Algorithm
NSGA	Non-dominated Sorting Genetic Algorithm
NSGA-II	Non-dominated Sorting Genetic Algorithm (version 2)
PACO	Population-based Ant Colony Optimisation
PBIL	Population Based Incremental Learning
PCB	Printed Circuit Board
PMBGA	Probabilistic Model-based Genetic Algorithm
PSO	Particle Swarm Optimisation
QAP	Quadratic Assignment Problem
RBAS	Rank-based Ant Systems
RTS	Restricted Tournament Selection
RVGA	Real Value Genetic Algorithm
SA	Simulated Annealing
SNGA	Sequential Niching Genetic Algorithm
TSP	Travelling Salesman Problem
VEGA	Vector Evaluated Genetic Algorithm
VLBI	Very Long Baseline Interferometry

CHAPTER 1

Introduction

1.1 Background to the study

As humans in an increasingly busy world we are confronted with optimisation problems on a daily basis. We are often striving to solve these problems more efficiently and effectively, regardless of whether we are consciously aware of it or not. The simple task of commuting between our homes and workplace can be treated as an optimisation problem since we may seek to minimise the time taken (objective value) to perform this commute. Furthermore there is a finite (albeit large) set of decision variables we can choose from such as roads, bus/train routes and walking paths. Yet while this simple and sometimes mundane task may seem trivial for us as humans to solve, it can prove to be quite difficult for a computer.

Although real world optimisation problems come in a variety of different forms, optimisation algorithm practitioners usually treat these depending on the interplay between problem variables. Consequently, two very popular and well studied problem classes are function optimisation, and combinatorial optimisation. The first class is concerned with selecting values for a finite set of problem variables, defined along either a discrete or continuous range. The later class includes problems where not only the value but also the ordering of variables is important, e.g. Travelling Salesman Problem, Shortest Path Problem, University Timetabling Problem, etc.

Methods used to solve optimisation problems are usually defined into one of two categories: deterministic and non-deterministic algorithms. Deterministic algorithms are usually well defined and understood since their deterministic nature allows for more accurate analysis and estimation of performance. Non-deterministic (stochastic) algorithms are not always understood, and hence performance estimations for these algorithms are usually given as confidence measures.

Non-deterministic algorithms are useful for problems where it may not be possible to execute a deterministic algorithm due to the size, or nature of the problem's search space. Such problems are usually denoted as NP-hard or NP-complete. In these cases a deterministic algorithm may take days or months to find an optimal solution, whereas a non-deterministic algorithm can usually find an approximate but hopefully still near-optimal solution in a matter of minutes or

seconds.

Ant Colony Optimisation (ACO) is a non-deterministic algorithm class that is based on the foraging behaviours of Argentine ants. ACO algorithm aim to mimic (and exploit) the behaviours of real ant colonies in order to solve optimisation problems. ACO algorithms belong to the class of constructive heuristic algorithms which work by building solutions to a given optimisation problem one solution component at a time according to a defined set of rules (heuristics). In other words these algorithms start with an 'empty' solution and add solution components one at a time until a complete solution is built. ACO algorithms are characterised by this solution construction and by their use of past solutions in manipulating an artificial 'pheromone'. This pheromone is a numeric value which is associated to every unique solution component. It reflects the estimated utility of each unique solution component. These pheromone values are used to bias solution construction by influencing the probability of a solution component being added to a growing solution based on the magnitude of the pheromone value.

The ability of ACO algorithms to solve more difficult artificial problem instances is important for researchers, as these difficult artificial problems are often close approximations of industrial (real-world) applications. However, as the complexity of the problem increases, the optimisation performance of many standard ACO algorithms will often decrease. To address this decrease in performance, practitioners often make augmentations to standard ACO algorithms in an attempt to increase their performance on a specific problem, usually through the introduction of complex operations and extra problem specific parameters. These modifications generally adjust the type or amount of problem specific information that the algorithm has access to, or the algorithm behaviour to balance the amount of computation time spent:

- Searching for solutions radically different from those already found (exploration).
- Exploiting information learnt through previously evaluated solutions (exploitation).

Exploration and exploitation somewhat define the difference between the various available algorithms which comprise the field of computational intelligence. It is the (often dynamic) balance between these behaviours that can define a specific algorithms suitability on a given problem.

1.2 Research Motives and Contributions

The Population-based Ant Colony Optimisation (PACO) algorithm is a recently developed ant-inspired algorithm which, unlike traditional ACO algorithms, maintains a population of solutions as well as pheromone information. It has been demonstrated to be an efficient algorithm when applied to a range of difficult single-objective, multi-objective and dynamic problem instances. It is believed that there may still be ways in which to further increase its effectiveness at solving some complex optimisation problems, and that population diversity plays an important role in addressing this issue.

Presently there are a number of disparate algorithms inspired by the first PACO algorithm.

These algorithms are often very different from the inspiration and only loosely base themselves around the idea of maintaining a population rather than strictly reusing all of the procedures defined by the original PACO algorithm. The lack of a common framework for these algorithms makes it difficult for newcomers to the field to identify what the core ideas underpinning the PACO algorithms are, and also makes it difficult for new research to be developed since it is unclear what effect subtle implementation differences have on algorithm performance.

The first contribution of this thesis is a comprehensive review of existing PACO algorithms, identification of common features and use of these features in the development of a PACO framework. The amount of existing work on PACO is minimal but not insignificant, as such the new framework will incorporate elements from the broader fields of ACO and Evolutionary Computation where appropriate. It is intended that this framework should not only encompass existing PACO algorithms but also allow for extension to several new algorithms.

The second contribution is the development of several new PACO algorithms imbued with a diversity preservation technique known as niching. Niching has been used extensively in the field of Evolutionary Computation, but has never been explicitly applied to an ACO algorithm per sé. An empirical analysis of these novel implementations is presented using a variety of single and multiple objective continuous function and combinatorial optimisation problems. These problems have been chosen since they demonstrate the advantages and disadvantages of adding niching to a PACO algorithm.

The final contribution is the application of the new PACO algorithms to an industrial application. This application is included to demonstrate the niching PACO algorithms' applicability to a 'real-world' optimisation problem instance.

1.3 Thesis Structure

To ease the comparison and explanation of key concepts, several standard problem classes are explained in Chapter 2. Performance metrics which will be used to measure algorithm performance are also presented in this chapter. Existing theory from the field of ACO is presented in Chapter 3. Several PACO algorithms and the PACO Meta-heuristic Framework are presented in Chapter 4. Evolutionary Computation and Niching are presented in Chapter 5, as well as several novel Niching PACO algorithms. The new PACO algorithms are tested on several benchmark single objective optimisation problems in Chapter 6 and several benchmark multiple objective problems in Chapter 7. The findings from these chapters are used in the application of the newly developed algorithms to a previously untested industrial application in Chapter 8. Finally, a summary and outlook for future research is offered in Chapter 9.

CHAPTER 2

Optimisation Problems

> When you can measure what you are speaking about and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind.
>
> Lord Kelvin, 1883

2.1 Introduction

Our world has no shortage of optimisation problems, whose difficulties range from trivially simple to infinitely difficult. Many of these problems are able to be modelled in a way which makes them amenable to optimisation by a computer algorithm. These optimisation problems generally comprise a set of finite solution components (variables) and optional constraints governing the range and composition of these solution components required for a feasible solution. Any feasible solution to an optimisation problem is usually able to be assessed for quality (how well it solves the optimisation problem) by a fitness function.

A first challenge in solving a specific problem is representing the problem in a meaningful way. The process of problem representation is a very important step in the overall algorithmic design challenge. A good problem representation not only allows an algorithm to produce feasible solutions, but may also aid the algorithm in finding optimal, or near-optimal solutions to the problem. Conversely a poor problem representation may hinder an algorithms ability to effectively or efficiently find a good solution.

This chapter discusses several classes of optimisation problem, with examples of each problem class included. For each problem class discussed an appropriate problem representation is included. All problems introduced in this chapter are used later in the thesis, either for assisting the explanation of an existing algorithm, explaining a general problem solving issue, or for use as benchmarks for the testing of new algorithms.

2.2 Combinatorial Optimisation

Consider the real world problem of scheduling, be it for designing a students timetable or for sequencing a group of machines in a production line. In these examples the problem is in selecting and ordering a set of discrete components. These are *combinatorial optimisation* problems, where the sequencing of a (usually) finite set of solution components defines the utility (goodness) of a solution [99].

2.2.1 Travelling Salesman Problem

The Travelling Salesman Problem (TSP) [127] can be described as:

> Given a set of n cities (vertices) and weights for each pair of cities, find a round-trip of minimal total weight that visits every city exactly once.

The total number of feasible solutions of a symmetric TSP, that is a TSP for which the weight connecting any two cities is the same regardless of direction of travel is that of (2.2.1). Given the development of many very good heuristics for the TSP (Nearest Neighbour [132], Lin-Kernighan [101]) the size of the search space required to locate the optimal solution and thus the search complexity is much lower than the size of the feasible solution space. Even given these heuristics though, the problem is still classified as NP-hard.

$$\text{Total Solutions} = \frac{(n-1)!}{2} \tag{2.2.1}$$

A suite of standard TSP instances is available from an online reference, TSPLIB [126, 128], and includes a variety of datasets ranging in size and representation. TSP datasets are usually represented in a standard coordinate data system such as 2-dimensional Euclidean, or geographical (using latitude and longitude). This representation is mostly chosen given the nature of the dataset itself, e.g. Burma14 is a dataset which represents 14 cities in Burma using geographical coordinates. Using an appropriate formula the distances connecting each vertex (Tab. 2.1) is calculated and stored in a matrix (Tab. 2.2).

City	X	Y
1	1	49
2	14	26
3	-20	43
4	30	32
5	27	-44

Table 2.1: 5 city TSP represented using 2D Euclidean coordinates

Examples of the TSP can be found in more practical scenarios than just the original context of a travelling salesman. Printed Circuit Board manufacture often requires the drilling of a number of holes and/or placement of a number of components (Fig. 2.1). To minimise the time required to drill/place we can interpret the locations as 'cities' and minimise the problem to minimise

City (to/from)	1	2	3	4	5
1	0.00	26.42	21.84	33.62	96.57
2	26.42	0.00	38.01	17.09	71.20
3	21.84	38.01	0.00	51.20	98.88
4	33.62	17.09	51.20	0.00	76.06
5	96.57	71.20	98.88	76.06	0.00

Table 2.2: 5 city TSP edge weight matrix

the time required to process each PCB. Solving this problem can save a manufacturer much time due to the large number of PCBs processed. Since hundreds of thousands of identical PCBs may be manufactured in a single batch, a small saving in time to manufacture a single PCB may scale to a large saving in the total processing time.

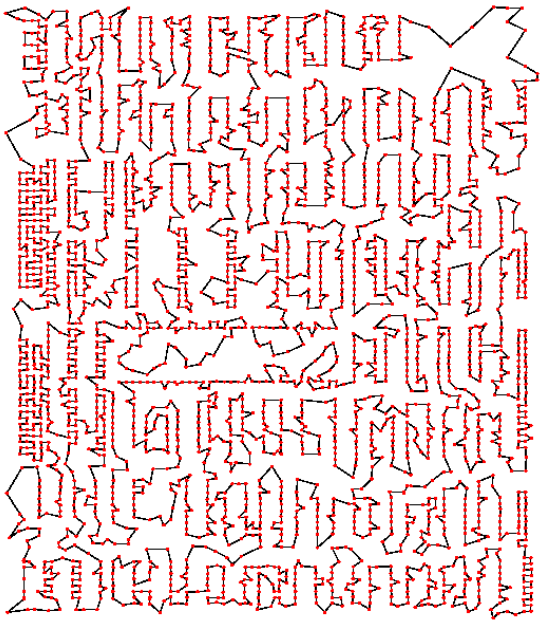


Figure 2.1: Optimal tour of the PCB3038 TSP (Available from http://www.tsp.gatech.edu/gallery/itours/pcb3038.html)

2.3 Continuous Function Optimisation

Many systems, good examples being those arising in economics and engineering, are modelled as Continuous Function Optimisation (CFO) problems. These systems accept a combination of input parameters that are then combined in some way to produce an output value or values. An example may be in a chemical engineering process where measured amounts of input chemicals are combined to produce a single output. Depending on the system, the relationship between the inputs and the output may be well understood, however if the system is complex enough it may be the case that the output can't be reliably back propagated to the inputs.

The principal components of a CFO problem are an objective function (fitness function). This maps the n-dimensional input vector to a single, or multiple output values. As previously illustrated the input vector will represent a set of unique scalar qualities of the system being mod-

elled. Additionally, the input vector is usually defined within boundaries, or constraints. These constraints may be physical, technical, economic, environmental, legal, societal depending on the system. The objective function may still return values outside the constraint boundaries, however the n-dimensional input will be said to be infeasible since it violates the constraints. Depending on the problem the objective function may be defined over any variety of number systems to any order of precision, and some objective functions may accept mixed input vectors e.g. a combination of floating point decimal and binary.

When solving CFO problems the objective is to usually maximise or minimise the output value(s) which can represent physical or non-physical quantities such as profit, product quality, speed of service or job completion, cost, risk, etc. Then though a systematic adjustment of the input values, one or many good or optimal solution(s) are selected.

2.4 Modality

In the context of this thesis, modality refers to the number of optimal (or near-optimal) solutions that exist in the search space of any optimisation problem. Multi-modal problems are interesting to researchers since the modality of a problem is usually associated with its difficulty and thus these problem domains offer challenges in algorithm design. Multi-modal problems are not necessarily the same as deceptive problems. This is because the symmetry of some multi-modal problems may allow optima to be located through the location of other optima thus their modality can actually assist an algorithm rather than hinder it. Deceptive problems tend to trap algorithms at local optima, in effect stopping algorithms from finding better optima by making them converge in the wrong direction.

Closely related to the issue of modality is the requirement for single or multiple final solutions. This may be for robustness e.g., multiple solutions means if one solution proves inadequate another solution may be used. If the problem is dynamic, having multiple solutions may provide the user with more options in the event of a change to the problem. In the case of multi-objective optimisation (Sec. 2.5) we don't always know what shape the Pareto front (Sec.2.5.2) will take and therefore we may not want to pre-define our final solution selection criteria. Due to these circumstances algorithms can sometimes be judged by their ability to locate and maintain multiple (usually spatially separated) optima rather than just one optimum.

2.5 Multiple Objective Optimisation

The simple task of commuting between our home and workplace can be treated as a single objective optimisation problem if we are only concerned with minimising the time taken to travel between the two. This example is fairly simple though, since if time was the only factor in determining our decision, then would it not be best to take an extremely fast (and expensive) form of transport e.g. a helicopter, to work? In fact cost is an important factor in our decision making, therefore is it not best to consider time and cost as separate objectives? If so, then we can

consider this problem as a Multiple Objective (Multi-objective) optimisation (MOO) problem, which is a more general class of optimisation problem of which single-objective optimisation is a subset.

Multi-objective optimisation is, in general, approached in three ways in the literature [88]:

- *A Priori Preference Articulation* – The decision maker combines all objectives into a single objective through the use of a weight vector, turning the multi-objective problem into a single-objective problem to search.
- *Progressive Preference Articulation* – The decision maker may provide partial preference information and adjust this preference information as the search continues by interpreting the results of the search.
- *A Posteriori Preference Articulation* – The decision maker does not provide any preference information, instead the decision maker is presented with a set of candidate solutions (generated by some search process) to choose from.

Neither approach is perfect for all situations and the method must always be matched to the needs of the problem and this will be discussed further in Sec. 2.5.4.

2.5.1 Objective and Solution space

Although multiple definitions exist in Evolutionary Computation literature for *phenotypic* and *genotypic* information [3] in this thesis the following definition will apply. Genotypic information refers to a solution encoding such as a permutation in the case of a TSP or input vector information as in a CFO. Phenotypic information refers to what is sometimes called fitness or objective information, e.g., the tour length in a TSP or the function value in CFO.

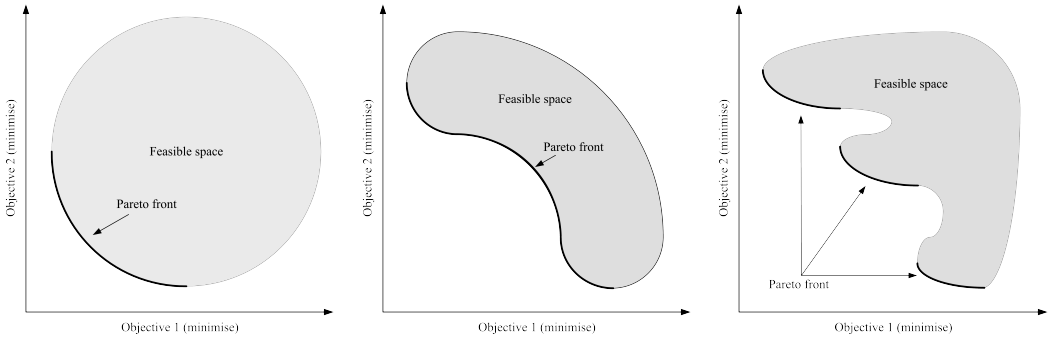
For single-objective optimisation problems, the genotypic information is usually mapped into an n-dimensional space called the solution space, or decision variable space, with the phenotypic information being the value of the points contained in this space. For multi-objective optimisation the relationship between genotypic and phenotypic information is more complex, and as such, the n-dimensional spaces that represent the genotypic and phenotypic information are clearly defined as the solution (decision) and objective (fitness) spaces respectively. It is said to be complex since it is possible that solutions which are close in proximity in the solution space are not necessarily close in the objective space. Due to this non-linearity it must always be clear in which space we are applying operators such as diversity maintenance techniques.

In most multi-objective optimisation problems the genotypic information does not alter from a similar single-objective optimisation problem. The multi-objective TSP is comprised of multiple distance matrices of the same size. A solution in this case is still a permutation, however in this case the permutation is evaluated multiple times according to each individual distance matrix. To interpret this in a real-world context think of each distance matrix corresponding to different costs between cities, e.g. time, distance, monetary value.

2.5.2 Pareto Optimality

Every multi-objective optimisation problem has two or more complimentary or competing objective functions. In the case of complimentary objective functions, a single optimal solution may exist which maximises (or minimises) all objective functions. In the case of competing objectives it will be the case that multiple trade-off solutions exist, and each of these solutions is said to be *Pareto optimal* after the economist Vilfredo Pareto. A Pareto optimal solution is said to be the best trade-off between any objective function such that there exists no change that can be made which will simultaneously benefit multiple objectives.

The number of Pareto optimal solutions making up the Pareto set and the shape of the resultant Pareto front depends on the specific problem instance. Several standard shapes are defined in the literature for specific Pareto fronts, illustrated in Fig. 2.2.



(a) Convex Pareto front **(b)** Concave Pareto front **(c)** Discontinuous Pareto front

Figure 2.2: Illustrations of different Pareto fronts plotted in the objective space.

2.5.3 Dominance

The dominance relationship is a means by which solutions are able to be compared and subsequently ranked. Simply put it compares two solutions (s_1,s_2) and indicates which of the following conditions apply:

- If s_1 is not worse than and is at least better in one objective (but not in all) than s_2, s_1 is said to weakly dominate s_2, denoted as $s_1 \preceq s_2$.
- If s_1 is better in all objectives than s_2, s_1 is said to strongly dominate s_2, denoted as $s_1 \prec s_2$.
- If neither of the above conditions is true s_2 is said to be non-dominated by s_1, denoted as $s_1 \not\preceq s_2$.

The principle of dominance relationships is important for search algorithms that rely on directional (fitness) information to guide their search process since it allows direct comparisons between multiple solutions. The dominance relationship represents one way to rank solutions for this purpose but other methods such as indicator-based selection and hypercube calculations also exist. A possible problem with dominance ranking is that it may not produce enough

grading since simply saying s_1 dominates s_2 does not indicate how much s_1 dominates s_2 by. This is due to the simplicity of the dominance relationship which can also be misleading by introducing a bias due to the projection angle by which solutions are compared [133]. This said, alternative measures are often very complex or cannot be scaled well to more than two or three objectives.

A non-dominated set is a set of solutions which are not weakly dominated by any other solution in the set. A Pareto optimal set will be a non-dominated set however the converse need not be true.

2.5.4 Solving Multiple Objective Optimisation Problems

As discussed in Sec. 2.5 MOO problems are, in general, solved in three ways: *a priori*, *progressive* and *a posteriori*. Knowledge, or lack thereof, of the Pareto front may aid the choice of technique as in some instances it has been proved that an *a priori* weighting can prevent a solver from returning a solution which is Pareto optimal [36]. Furthermore, an even distribution of solutions across the Pareto front is not guaranteed with even spaced weight vectors given a convex Pareto front [32].

In the absence of sufficient knowledge of the Pareto front a *progressive* or *a posteriori* approach may be useful since it may uncover facts about the shape and nature of the objective space. These approaches often strive to meet two conflicting goals:

- Find solutions which are close to the Pareto front.
- Maintain a diverse range of solutions along the Pareto front.

These goals are said to be conflicting since most algorithms adapted to solve MOO work best when concentrating (converging) on specific areas of the search space, however this search behaviour will often lead to a loss of diversity. Thus an ideal approach must be able to balance exploratory (diversity preserving) behaviour with exploitative (quality enhancing) behaviour. In this thesis the focus is on developing and comparing MOO algorithms that address these goals.

2.5.5 Existing Algorithms

A discussion of all existing MOO algorithms is outside the scope of this thesis. Several algorithms are discussed later in the thesis in Sec. 7.2 since these algorithms are used for comparison purposes.

2.6 Choice of Problems

In this thesis several classes of problem have been selected. Each problem chosen has been carefully selected from a much wider family of optimisation problems. Although it would

be desirable to be able to solve every problem to optimality using the same algorithm with a minimal amount of computation, this is simply not possible. It is generally accepted that there is no silver bullet i.e., no algorithm which can solve every problem better than any other, and the no free lunch theorem [162] supports this hypothesis. Alternatively, as algorithm designers the task then becomes to discover the best fits between algorithm and problem i.e., for what problems does a specific algorithm work best.

In the case of the Travelling Salesman Problem (TSP) (Sec.2.2.1) the best performing algorithm in terms of ability to locate an optimal solution at present is a deterministic solver named Concorde [2]. This algorithm has been able to solve some of the most challenging TSP problems proposed in the literature, the largest currently being a TSP of 33,810 cities (The computation took 15.7 CPU years). This Concorde solver is a highly specialised algorithm that relies on parallel computing and large amounts of virtual memory.

Even given the existence of Concorde, algorithm designers still decide to test new algorithms on the TSP for a number of reasons:

- An algorithm designer may believe that they can achieve a better algorithm design than that of Concorde, i.e., one that can find the optimal solution in less computational time with less or similar computational resources.

- Given that Concorde requires massive amounts of computational resources an algorithm designer may wish to develop an algorithm that trades an amount of solution quality for lower computational resource.

- The TSP is a well known problem which is well understood, thus observing an algorithms behaviour on the TSP may give insight into the nature of that algorithm.

It is interesting that Ant Colony Optimisation algorithms tend to be most closely associated with TSP, no doubt because the first ACO algorithms were applied to the TSP. This is akin to Genetic Algorithms (GA) being associated with CFO. De Jong commented that algorithms tend to be defined in very general terms and are later specialised to fit specific applications, hence a case of the cart coming before the horse [89]. Given this, it is often (unfortunately) evident that algorithms are typecast depending on the first problems they are applied to, even if they are by definition more general paradigms.

In this thesis several problems will be used to highlight important properties of proposed algorithms, however, it should not be thought that these problems are by any means the only problems able to be addressed with these algorithms. The ACO paradigm is, by definition, a general algorithmic framework, as such it is possible to apply it to, with an amount of specialisation, a broad range of problems. It is intended that the later problem analyses will highlight general features of the algorithms that will enable the future application of the algorithms to other problems that also exhibit similar features. With this in mind problems have been selected on purpose to identify algorithm strengths and weaknesses so as to allow reasonable postulates as to the expected performance of the algorithms if applied to other problems.

Throughout the thesis many state-of-the-art algorithms for specific problem classes are included. The inclusion of these algorithms is primarily to obtain an expected level of good performance. In the literature there is sometimes a tendency to try to competitively compare two algorithms in order to conclude that one is better [83, 6]. It was maybe put best in [83] that 'most experimental studies of heuristic algorithms resemble track meets more than scientific endeavours'. This thesis, rather than try to competitively compare algorithms, relies on controlled experimentation to obtain insight into a new algorithms behaviour. Existing algorithms can then be used to assist in placing the new algorithm behaviour in the right context, i.e. while the new algorithm might achieve a better quality solution than the control algorithm, and in some instances this is certainly the intent, it is of more interest to answer the question, 'why did it achieve a better quality solution?'

2.7 Performance Statistics

As results from multiple different algorithms are to be compared, it may be necessary to indicate the statistical significance of such a comparison. Some statistical methods assume that data is taken from a known distribution, such as a normal distribution. If this assumption can be made these tests are usually quite reliable and powerful ways of reporting on any difference between two or more samples. As a general rule though these parametric statistical methods are not applied when testing stochastic algorithms. Instead non-parametric statistical methods that rely on rank or permutation tests are commonly used, as these tests make no assumptions as to the nature of the distribution [26].

Rank tests, such as the Mann-Whitney [111], Wilcoxon Rank-sum [161] and Kruskal-Wallis [97] tests pool values from multiple samples and sort them according to some unitary measure, such as path length. In the case of the Mann-Whitney test all samples are assumed to have come from the same population, and the null hypothesis is that the samples are the same. Thus, when the assigned ranks are summed together from individual samples any differences have an effect on whether this null-hypothesis is accepted or rejected. If the samples contain multiple ties then the use of a permutation test such as Fisher's test may be considered, since this test does not discard as much information, however it is far more computationally intensive.

In this thesis many of the aforementioned statistical techniques will be used when comparing algorithms, however, they are reserved for use only when robust conclusions need to be made. For example during a simple qualitative analysis such statistical significance is not required, however when comparing multiple algorithms to determine suitability for a particular application then statistical significance is mandatory.

Statistical significance implies that if a difference between two groups of results is observed then this difference is not the result of chance. The usual method to perform such an analysis is to formulate a null hypothesis and test this hypothesis. An example and commonly used null hypothesis is that results obtained from algorithm A are the same as the results obtained from algorithm B. After defining a null hypothesis, a significance level (α) for the hypothesis

to be rejected must also be selected. Common values for significance are 5% or $\alpha = 0.05$. A statistical significance test (such as those mentioned earlier) will return a p value and if this p value is lower than the significance level (α) then the null hypothesis is rejected. As an example, in determining whether algorithm A is better than algorithm B a pair-wise non-parametric statistical test returns results that find that:

- algorithm A is better than algorithm B with $p = 0.01$
- algorithm B is better than algorithm A with $p = 1$

From these results we can conclude that algorithm A is the better than algorithm B with a significance level of 1%, however algorithm B cannot be said to be better than A since the p value is larger than the generally accepted significance level of 5%. In this thesis the generally accepted 5% significance level is used, unless otherwise explicitly noted.

2.8 Chapter Summary

This chapter introduced the concept of optimisation and gave examples of several optimisation problems. Specifically it concentrated on explaining concepts associated with single and multiple objective optimisation and the Travelling Salesman Problem and Continuous Function Optimisation Problem. The next chapter introduces Ant Colony Optimisation, an optimisation paradigm inspired by the foraging behaviour of ants.

CHAPTER 3

Ant Colony Optimisation

> The neglect of ants in science and natural history is a short-coming that should be remedied, for they represent the culmination of insect evolution, in the same sense that human beings represent the summit of vertebrate evolution.
>
> Hölldobler and Wilson, 1990

3.1 Introduction

Ants, while displaying complex and yet efficient emergent collective behaviours such as nest construction and food collection, are inherently simple on an individual ant level. What is even more amazing is that these emergent properties seem to exist without the requirement for centralised control. This last observation has made this biological system an attractive inspiration for optimisation algorithm research. As such a whole new field of algorithm research has been developed in the last 15 years formally known as *ant inspired algorithms*.

In this chapter Sec. 3.2 presents the first 10 years of the field, from theoretical biological model and associated experiments, to the development of the first ant inspired algorithms through to the eventual definition of the Ant Colony Optimisation Metaheuristic Framework (ACO). Section 3.3 details ACO while Sec. 3.4 reproduces and discusses implementation details of several ACO algorithms. Sec. 3.5 discusses several salient issues arising from the field of ACO.

3.2 Early Development

3.2.1 Biological Inspiration

Tetramorium caespitum ants have evolved to suit an ecological niche where food sources are plentiful and a desired emergent behaviour is one which optimises the distribution of colony resources to maximise the food collection activity [117]. These ants rely on randomness to influence their decision making behaviour, however with the absence of any individual long term

memory, ants rely more on intra-colony communication mechanisms (stigmergy)[1] to influence their foraging decisions.

This species of ant exhibits three distinct behaviours: group-recruitment, mass-recruitment and random-exploration. Group recruitment occurs when an ant finds a new food source, returns to the nest and upon returning to the nest attempts to coerce other ants to follow it back to the food source, laying pheromone (a biochemical marker) along the trail as they move. This recruitment can eventually lead to a mass-recruitment if the food source is large enough, and the pheromone trail is reinforced enough so that ants can begin to follow the pheromone trail independently. Random exploration can occur at any stage, where an ant following a pheromone trail decides to leave the trail to search virgin territory in the hope of finding more food, or a more efficient path to already discovered food. The probability of such an event occurring is inversely proportional to the amount of pheromone and directly proportional to the distance away from the nest.

The importance of random-exploration is to encourage exploration and avoid exploitation of one food source (or collection path) neglecting other possibly more rich food sources or shorter paths [41]. This emergent effect was perhaps most profoundly demonstrated in the double bridge experiments [40, 73]. In these experiments a single food source was placed away from a nest of *Iridomyrmex humilus* ants and two tubes (bridges) of equal length connected the nest to the food (Fig. 3.1). Initially the ants were observed to use both bridges fairly equally to retrieve the food, however the majority of the colony began to favour one bridge over the other (Fig. 3.2) eventually leading to the colony only using this bridge. The experimental set-up was reinitialised several times and it was documented that the colony selected the bridges with an even probability. The experiment was modified with bridges of unequal length [73] to determine which bridge the ants would favour. The result was that in a majority of the experiments the ants favoured the shorter bridge. The researchers explained this emergent (*autocatalytic*) effect by the fact that a shorter distance means that ants can forage on this path more quickly and in a given time this branch will be positively reinforced with more pheromone.

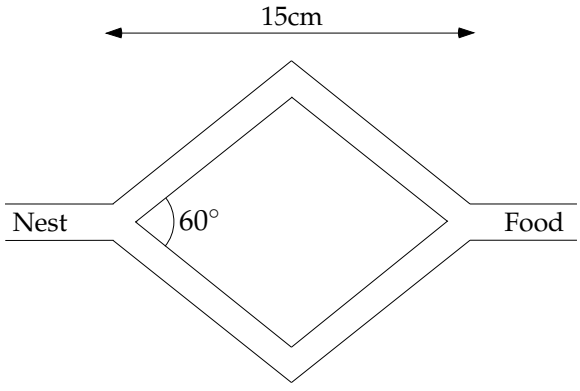
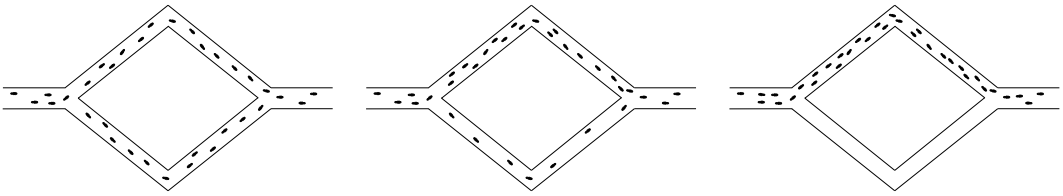


Figure 3.1: Double bridge experimental setup

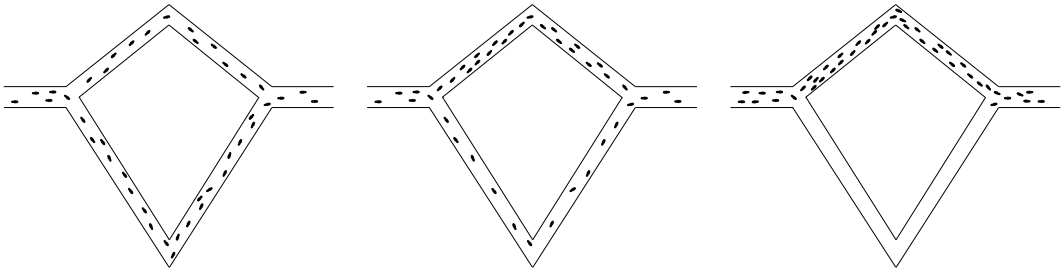
Using the observations from the double-bridge experiment, a model for calculating the selec-

[1] Stigmergy, a term coined by Grassé [74], is interpreted in this context as an indirect communication mechanism which relies on alterations to an environment to relay information.



(a) Time = 1 minute **(b)** Time = 6 minutes **(c)** Time = 10+ minutes

Figure 3.2: Graphical reproduction of the double bridge experiment using paths of equal length.



(a) Time = 1 minute **(b)** Time = 6 minutes **(c)** Time = 10+ minutes

Figure 3.3: Graphical reproduction of the double bridge experiment using paths of unequal length.

tion probability of each bridge was created. The model uses the number of ants that have traversed either bridge as variables (m_1 and m_2) along with two user-specified parameters, h and k. The result of Equ. 3.2.1 is the probability of an ant choosing a particular bridge, p_1 where $p_2 = 1 - p_1$. Experimental results suggest that setting $k \approx 20$ and $h \approx 2$ will lead to the same behaviour observed in the biological experiments.

$$p_1 = \frac{(m_1 + k)^h}{(m_1 + k)^h + (m_2 + k)^h} \tag{3.2.1}$$

3.2.2 The Ant Systems Algorithm

The double bridge experiments described in Sec. 3.2.1 led to the development of three algorithms, Ant-density, Ant-quantity and Ant-cycle [55, 46]. What differentiated these algorithms from the biologist's algorithms was their purpose. The biologists were mainly interested in modelling the behaviour of ants to understand issues such as emergent behaviour and collective intelligence. These new algorithms were the first of their kind to harness the collective behaviour of an ant colony in an attempt to solve an artificial optimisation problem, the TSP.

In these algorithms each (artificial) ant iteratively constructs a solution to a TSP by probabilistically selecting an edge to include to a (initially empty) tour based on a nearest neighbour heuristic, and an artificial pheromone density which is adapted by the artificial ants as the search progresses (much like the mass-recruitment process described in Sec. 3.2.1), the distinguishing feature of each algorithm being the way that this pheromone is adapted. After experimenting with the three simple models, Ant-cycle was shown to be the most effective at optimising the TSP problems addressed [24] and was later refined and reintroduced as Ant Systems (AS) [56][2].

[2]Until this paper Ant System was used to collectively refer to the Ant-density, Ant-quantity and Ant-cycle algo-

The Ant Systems algorithm for the TSP from [56] will be discussed in Sec. 3.4.1.

3.2.3 Ant Colony Optimisation

After the initial successes of AS, several researchers extended the work of Dorigo et. al. by applying AS to other optimisation problems, and by augmenting the algorithm with extra features intended to improve its performance (a listing of some of these algorithms is provided in Tab. 3.1). Around 1996 some researchers began to use the term *Ant Colony Optimisation* in reference to the core ideas underlying the first AS algorithms[3]. By 1998, Ant Colony Optimisation had entered mainstream usage by researchers to describe algorithms inspired by ant behaviour. By this stage a substantial body of work on ant algorithms was appearing and at the turn of the century several researchers published a formalised algorithmic framework for these algorithms: The Ant Colony Optimisation Metaheuristic Framework (ACO) [49, 48, 50, 108]. This early development of ACO is presented visually in Fig.3.5.

Algorithm name	Problem type	Reference
Ant Cycle	TSP	[24, 46, 55]
Ant System (AS)	TSP	[56]
	TSP & QAP	[23]
	QAP	[109, 110]
	Job-Shop Scheduling	[25]
Ant-Q	TSP	[52, 66]
Ant Colony System (ACS)	TSP	[53, 65]
ANTCOL	Graph colouring	[30]
Hybrid Ant System	VRP	[15]
AntNet	Routing	[43, 44, 45]
$\mathcal{MAX} - \mathcal{MIN}$ Ant System (MMAS)	TSP	[147, 148, 149]
	Flow Shop Problem	[146]
Rank-based AS (RBAS)	TSP	[16, 17]
Parallel Ant Systems	Combinatorial	[18]
ANTS	Frequency Assignment	[107]

Table 3.1: Research Papers published before the formalisation of ACO

The first papers published on ACO cite reasons for proposing the framework as: "the desire to provide a unitary view of the ongoing research in this growing field", and that "such a categorization will help to identify the most important aspects of these algorithms". Additionally, several other references discussing ACO exist [27, 47, 57, 11]. A formal description of ACO is included in the proceeding section.

3.3 The Ant Colony Optimisation Metaheuristic Framework

A framework can be defined as *the skeleton upon which various objects are integrated for a given solution*. In other words it is a generic structure which is further specialised for a particular

rithms, much like ACO is used today to refer to many different ant-inspired algorithms.

[3]The first known reference to mention *Ant Colony Optimisation* was [51] which was later published as [53]

application. ACO is a generic algorithmic structure responsible for the scheduling of three processes:

- Ants generation & activity
- Pheromone trail evaporation
- Daemon actions

This section defines these processes as well as other data structures required for the implementation of an ACO algorithm for a specific optimisation problem. A visual representation of the organisation of these processes is provided as Fig. 3.4.

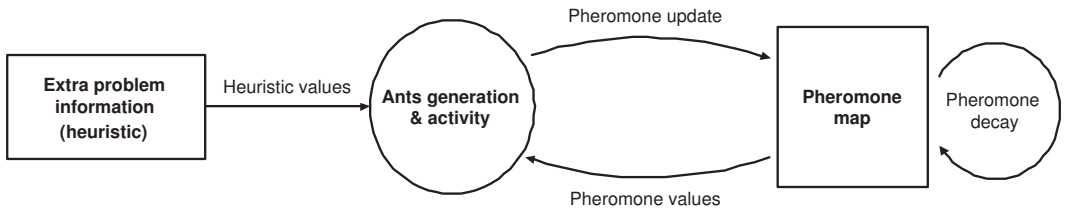


Figure 3.4: Process organisation of the Ant Colony Optimisation Metaheuristic Framework

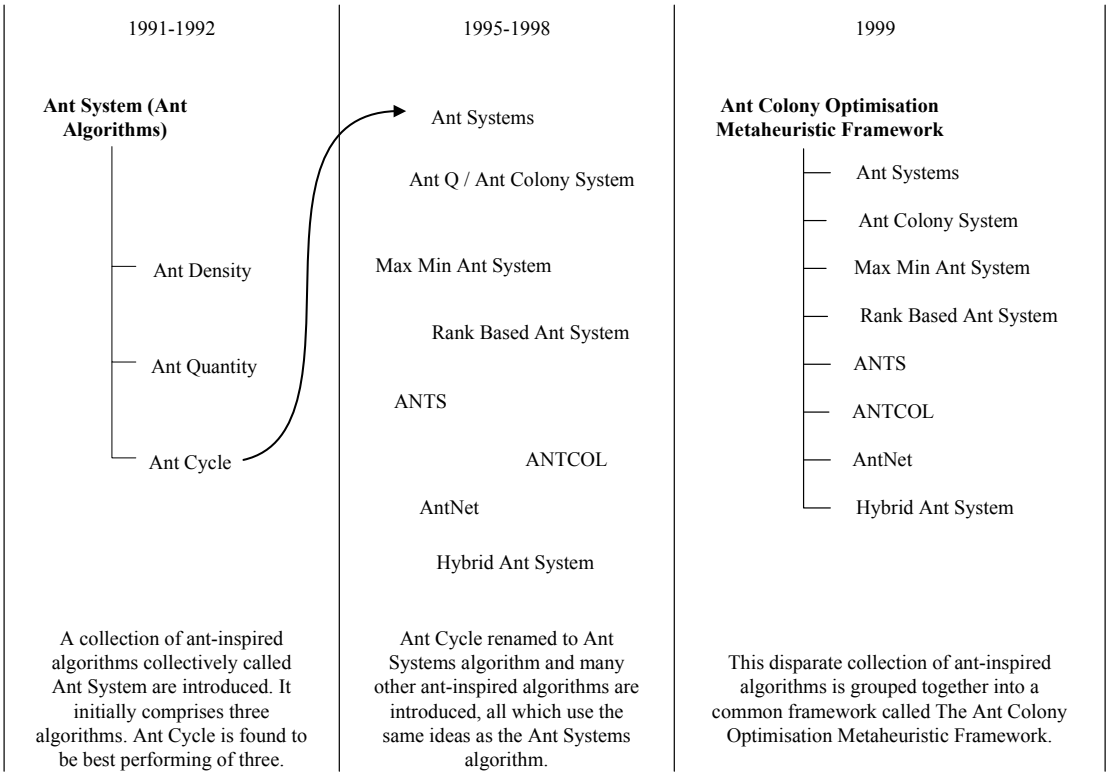


Figure 3.5: Early development of the field of ACO

3.3.1 Pheromone mapping

The pheromone mapping is the means by which solution components are able to be ranked and selected based on past usefulness. The pheromone mapping connects pheromone values from a pheromone map (usually a matrix structure) to specific solution components. The assumption usually being that if a prior solution is good then at least some of its parts (solution components) should also be good, and therefore a remixing of these components with other good components may lead to an optimal or near-optimal solution. A first step in defining an ACO algorithm is to define the pheromone mapping.

The problem domain will dictate how the pheromone mapping should be defined. In applying an ACO algorithm to a combinatorial optimisation problem such as the travelling salesman problem (TSP) it is not of interest which specific components are included, as any feasible solution will include every city once (and only once), it is the order of these components which is important in finding an optimal solution. For the TSP the transition points (edges/arcs) between the specific components can be assigned a specific pheromone value in order to reflect which order of cities works the best. That is, that if a solution included an edge connecting city a to city b and the solution is good then this should be reflected in the pheromone level on this specific edge and the other edges included in the solution.

3.3.2 Ants Generation and Activity

This process is responsible for the creation of new candidate solutions to the optimisation problem being addressed by the algorithm. A temporary population of (artificial) ants is used to construct feasible solutions to the problem being addressed. Each ant is evaluated upon the completion of a feasible solution and the solution information encoded into a global pheromone mapping. Each individual ant is discarded after entering their specific solution information into the pheromone mapping and a new 'empty' ant is created in its place, until some stopping criterion is met.

An ant has the following properties:

- An ant searches for a minimum (or maximum) cost solution to the optimisation problem being addressed.
- Each ant has a memory used to store all solution components used to date, so that the candidate solution can be evaluated at the completion of solution construction; the memory can be used as a tabu list such as in the case of the TSP so that no component is reused.
- An ant can be assigned a starting position, for example an initial city in a TSP.
- An ant can include any feasible solution component (an example of a feasible solution component in a TSP would be a city which has not already been included in the candidate solution) until such time that no feasible components exist or a termination criterion is met (usually correlating to the completion of a candidate solution).

- Ants include solution components according to a combination of a pheromone value and a heuristic value which are associated with every solution component in the problem, the choice of which solution component is usually a probabilistic one.
- When including a new solution component in the growing candidate solution the pheromone value associated with the transition between these components (arc/edge in a TSP), or the solution component itself can be altered (*online step-by-step pheromone update*).
- An ant can retrace a candidate solution at the completion of a solution, updating the pheromone values of all transitions and/or solution components used in the solution (*online delayed pheromone update*).
- Once a candidate solution is created, and after completing online delayed pheromone update (if required) an ant dies, freeing all allocated resources.

3.3.3 Pheromone Trail Evaporation

Like the biological ant colony, the artificial ant colony employs a pheromone evaporation mechanism. This mechanism serves as a useful way of 'forgetting' older search bias [49]. As ACO uses positive reinforcement, if pheromone was allowed to accumulate without decay the system would very quickly converge on a single solution since this solution would continue to be reinforced.

3.3.4 Daemon Actions

Daemon actions can be used to perform specialised functions which often require more knowledge than an individual ant is allowed [49, 47]. For example, a daemon action could inspect all solutions generated in one search cycle, identify the best solution and increment the pheromone values of its solution components more than the regular pheromone update (*offline pheromone update*). An alternative daemon action could be the application of a local search procedure.

3.4 ACO Algorithms

3.4.1 Ant System for the Travelling Salesman Problem

In this instance pheromone values correspond to transitions between cities (edges) and are uniformly initialised to an amount slightly higher than what is expected to be added in one iteration of the algorithm as in Equ. 3.4.1.

After initialisation the AS algorithm runs a *pheromone trail evaporation* procedure which is implemented by applying the rule Equ. 3.4.5 for every pheromone value. This procedure is followed by *ants generation & activity* which is implemented in the following steps:

1. A temporary population of m ants are placed at randomised starting cities.

2. Each ant k applies the *random proportional rule* Equ. 3.4.2 to decide which city to add to its current tour.

3. Step 2 is repeated until every ant k constructs a complete solution.

4. Every individual solution is evaluated and the edges used in this specific solution have their pheromone value adjusted according to Equ. 3.4.4. This equation allocates a higher proportion of new pheromone to better solutions in order to 'reinforce' good decisions and is an implementation of an *online delayed pheromone update* strategy.

5. The temporary population of ants is discarded.

The *pheromone trail evaporation* and *ants generation & activity* procedures are continually repeated until a termination criterion is reached, such as an amount of computation time, or alternatively by implementing a *daemon action* to observe the similarity of the solutions obtained over several iterations of the algorithm to test the convergence of the algorithm.

$$\forall (i,j), \tau_{ij} = \tau_0 = m/C^{nn} \tag{3.4.1}$$

$$p_{ij}^k = \frac{\left[\tau_{ij}\right]^\alpha \left[\eta_{ij}\right]^\beta}{\sum_{l \in N_i^k} \left[\tau_{il}\right]^\alpha \left[\eta_{il}\right]^\beta}, \text{ if } j \in N_i^k \tag{3.4.2}$$

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{3.4.3}$$

$$\tau_{ij} = \tau_{ij} + \frac{Q}{L} \tag{3.4.4}$$

$$\tau_{ij} = (1-\rho)\,\tau_{ij} \tag{3.4.5}$$

Where:

τ_{ij}	:	Pheromone value for edge connecting city i & j
m	:	Number of ants
C^{nn}	:	Length of path found using a nearest neighbour heuristic
p_{ij}^k	:	Probability of ant k selecting the edge connecting city i & j
α	:	Magnitude of pheromone influence on probabilistic decision
η_{ij}	:	Heuristic value for edge connecting city i & j
β	:	Magnitude of heuristic influence on probabilistic decision
N_i^k	:	The set of cities that ant k has not yet visited
d_{ij}	:	The distance between city i & j
ρ	:	Pheromone evaporation rate
Q	:	Amount of pheromone to deposit
L	:	Path length

3.4.2 Ant Colony Systems

Ant Colony Systems [53, 65](initially introduced as Ant Q [52, 66]) differs from AS in three areas:

- Introduction of a *local pheromone update*.
- Modification of the *global pheromone update*.
- Modification of the random proportional rule to become the *pseudo-random proportional rule*.

The local pheromone update is applied by all ants during the solution construction phase. Every ant continually applies the update rule to the last solution component used as in Equ. 3.4.6. The aim of this pheromone update rule is to attempt to diversify the search process as much as possible *during* the solution construction phase. Without it most ants will simply create the same solution which will lead the search into a stagnation behaviour.

$$\tau_{ij} \leftarrow (1-\rho) \cdot \tau_{ij} + \rho \cdot \tau_0 \tag{3.4.6}$$

The global pheromone update is modified so that only the best-so-far or iteration-best solution updates the pheromone map at the completion of solution construction. This means that unless a solution component has been included in the best solution it will not receive any modification from the global pheromone update, as in (3.4.7).

$$\tau_{ij} = \begin{cases} (1-\rho) \cdot \tau_{ij} + \rho \cdot \Delta\tau_{ij} & \text{if } (i,j) \text{ belongs to best tour,} \\ \tau_{ij} & \text{otherwise.} \end{cases} \tag{3.4.7}$$

The value of $\Delta\tau_{ij}$ reflects the utility of the solution and is dependent on the problem e.g. for the TSP as in Sec. 3.4.1 it can simply be the inverse of the path length of the solution.

The final and perhaps most important difference between ACS and AS is the modification of the random proportional rule to become the pseudo-random proportional rule. This rule introduces a new parameter q_0. When a uniformly random value q in the range $[0,1]$ is less than q_0, the largest transition probability value generated by Equ.3.4.2 is used, rather than using a roulette wheel selection of all generated probabilities.

3.4.3 $\mathcal{MAX}$-$\mathcal{MIN}$ Ant Systems

The $\mathcal{MAX}$-$\mathcal{MIN}$ Ant Systems algorithm (MMAS) [147, 148, 149] improved the AS algorithm by introducing pheromone thresholds to counter premature convergence observed in AS. This thresholding is achieved through the introduction of upper and lower pheromone bounds, τ_{min} and τ_{max}. The AS global pheromone update is modified to that of ACS Equ. 3.4.7 and as in ACS only the global best solution is used to apply this update. Pheromone decay is the same as that

of AS. When applying update and decay any individual pheromone value is restricted to the range $[\tau_{min}, \tau_{max}]$. Guidelines for determining the values of τ_{min} and τ_{max} are outlined in [150].

In early works on MMAS the standard AS random proportional rule is used to determine transition probabilities, however in later works some researchers opt to use the ACS pseudo-random proportional rule instead. MMAS employs a pheromone re-initialisation scheme which upon detection of convergence initialises all pheromone values to τ_{max}, while retaining the best solution found so far.

3.4.4 Rank Based Ant Systems

Rank Based Ant Systems (RBAS) is another improvement of AS which works by ranking solutions against each other and only using the μ best for global pheromone updating. It was postulated that since AS reinforces pheromone, albeit to a differing degree, on all found solutions that this could lead to the over-reinforcement of poor solution components.

RBAS modifies the global pheromone update of AS by introducing several steps. All solutions generated in a single iteration of the algorithm are evaluated and ranked against each other. The μ best solutions are then selected and each of these solutions is allowed to update their components pheromone values according to Equ. 3.4.8. Solution construction is by means of the AS random proportional rule.

$$\tau_{ij} = \rho \cdot \tau_{ij} + \Delta\tau_{ij} + \Delta\tau^*_{ij} \tag{3.4.8}$$

$$\Delta\tau_{ij} = \sum_{\mu=1}^{\sigma-1} \Delta\tau^{\mu}_{ij} \tag{3.4.9}$$

$$\Delta\tau^{\mu}_{ij} = \begin{cases} (\sigma - \mu)\,\frac{Q}{L_\mu} & \text{if the } \mu\text{-th best ant travels on edge } (i,j)\,, \\ 0 & \text{otherwise.} \end{cases} \tag{3.4.10}$$

$$\Delta\tau^*_{ij} = \begin{cases} \sigma\,\frac{Q}{L_*} & \text{if edge } (i,j) \text{ is part of the best solution found,} \\ 0 & \text{otherwise.} \end{cases} \tag{3.4.11}$$

3.5 Salient Issues in ACO

3.5.1 Ant Inspired Algorithms

The field of ant inspired algorithms contains any algorithm which is inspired by the mechanics of a biological ant colony. ACO is considered a class of ant inspired algorithms, however while all ACO algorithms are considered ant inspired algorithms the converse is not true. Many ant inspired algorithms use concepts from ACO but don't fall within the strict definition of ACO. Hence common properties of these algorithms are:

- Use of a common history repository (pheromone map) so that every solution component is assigned a singular value (pheromone value) representing the desirability of using this solution component in future solution construction.
- Stepwise solution construction. Starting with an empty solution (or in some cases a partially built solution) incrementally add solution components until a termination criterion is met.
- Use of a population of multiple individual agents (ants) to construct candidate solutions sequentially or in parallel.

Examples of algorithms which are classed as ant inspired algorithm but fall outside of the scope of ACO are:

- An ant inspired reinforcement learning process for robot navigation[100].
- An ant inspired system for making strategic decisions in games[60].
- Ant colony metaphor for searching continuous design spaces [9].
- Hybrid ant system for the quadratic assignment problem [69, 151].
- Population based ACO algorithms that retain a population of solutions in conjunction with the pheromone map (These will be discussed in Chapter 4).

A number of these algorithms were published around the same time as the seminal papers on ACO, however these algorithms have not been the subject of continued research effort like ACO.

3.5.2 Local search

In general ACO algorithms are rarely applied to challenging 'real-world' problem instances without the addition of a domain-specific local search. This local search can be applied sequentially so that the ACO algorithm produces a set of starting positions for the local search to attempt to improve, or in parallel so that candidate solutions are sometimes improved before updating the pheromone map. Local search algorithms are mostly iterative improvement algorithms, and in general are somewhat sensitive to their given starting location. In the absence of a good starting location these techniques may require far more computational resources to achieve an optimal result, or may get stuck in a local optima.

In this work we are most interested in ascertaining algorithms' global search performance characteristics when applied to a variety of different problem domains, under different execution conditions. It is assumed that the amount of improvement that a local search algorithm can provide to two different global search algorithms will be relative to the existing utility of these algorithms. For this reason, when the performance of two different algorithms is to be compared, local search procedures have been omitted. However, all algorithms in this work could quite easily be augmented with local search and it would be expected that this would further increase their search efficacy.

3.6 Chapter Summary

This chapter has introduced the development and definition of the field of ant inspired algorithms and ACO. Biological models and experiments such as the double-bridge experiment and their influence on the creation of the first ant algorithms for optimisation were discussed. The development and introduction of a specific optimisation paradigm, The Ant Colony Optimisation Metaheuristic Framework was reviewed. Several ACO algorithms were reproduced with important features highlighted and discussed. Lastly several salient issues in the field were presented.

The next chapter discusses a class of ant-inspired algorithms known as Population-based ACO. These algorithms use the same probabilistic step-wise construction rules as ACO, however they maintain a storage of a discrete population of solutions. This is unlike ACO algorithms which only use solutions to influence pheromone levels before discarding them, only ever storing a single elite solution in its entirety (in some particular cases).

CHAPTER 4

Population-based Ant Colony Optimisation

> It's not population or enormous resources which open the golden doors to success, but a skillful use of brains and scientific know-how.
>
> John T. Conner

4.1 Introduction

After the construction and consequent evaluation of new solutions, ACO algorithms generally encode new solutions' quality (phenotypic) information into a pheromone map (pheromone update) and discard the solution (genotypic) information. An exception to this is Elite Ant Systems (Elite-AS) [56] and its variants which store a single elite solution to facilitate the positive reinforcement of its solution components in future iterations. An alternative to discarding solutions is to store all solutions in a population structure much like that of a GA. Such an ant-inspired algorithm has been previously proposed by Guntsch[1] and is called Population-based ACO (PACO).

This chapter provides a description of several PACO algorithms. Determining features are identified and a reinterpretation of the seminal work which describes these features and allows for further specialisation is introduced. The chapter concludes with a section describing Probabilistic Model Building Genetic Algorithms, which as will be discussed, have many similarities to PACO algorithms.

4.2 FIFO-Queue ACO

The first published PACO algorithm, FIFO-Queue ACO [78] was applied to static instances of the TSP and QAP and was shown to be competitive against other state-of-the-art ACO algo-

[1] The seminal work on Population-based ACO algorithms is Guntsch's thesis [76].

rithms. This work introduced the idea of using a population of stored solutions alongside a pheromone map and demonstrated that no decrease in performance (in terms of quality of solution found) is incurred through the modification. A motivation for the development of the algorithm was that the population facilitates a fast pheromone update process.

The algorithm uses the pseudorandom proportional rule to construct solutions (like that of ACS), however the pheromone modification is different from any existing ACO algorithm. Each iteration a single solution is added to a finite sized population, and as each solution is added to the population its associated component's pheromone values are positively updated. Once a finite population size (k) is reached, solutions are also removed from the population with their pheromone values negatively updated.

4.2.1 Algorithm Description

Prior to execution of the algorithm the pheromone map's values are initialised uniformly as (τ_{init}) and a maximum pheromone value is set (τ_{max}). Every iteration m solutions are created using the random proportional rule and evaluated. The best (new) solution of these m solutions is selected and compared against the current population. If the new solution is better than the current elite solution it replaces it and updates the associated pheromone values according to Equ. 4.2.1 with the old elite solution decrementing its associated pheromone values by Equ. 4.2.1. For the first iteration the new solution simply becomes the elite solution. The parameter w_e is the weighting applied to the single elite solution.

If the new solution is not better than the elite solution it is added to the population with its associated pheromone values incremented according to (4.2.2). Once this general population is full ($= k - 1$) the oldest solution is removed as a new solution is added and the pheromone associated with the old solution decremented by (4.2.2). An algorithmic representation of FIFO-Queue ACO is provided in Alg. 1.

$$\Delta\tau_e = (w_e)\,(\tau_{max} - \tau_{init})\,/k \tag{4.2.1}$$

$$\Delta\tau = (1 - w_e)\,(\tau_{max} - \tau_{init})\,/k \tag{4.2.2}$$

This update process removes the requirement for decay, and as such reduces the complexity of the standard ACO pheromone update and decay processes from n^2 to that of $2n$, since only those pheromone values associated with the solutions entering and leaving the population require updating versus all pheromone values in the normal case. As such, this process is referred to as a fast pheromone update.

Algorithm 1 FIFO-Queue ACO

```
1: Uniformly initialise pheromone map values to τ_init
2: Construct m solutions
3: Select best solution and insert into population as elite
4: Update pheromone values using elite update (+Δτ_e)
5: while stopping criterion not met do
6:     Construct m solutions
7:     Select best solution
8:     if Best Solution is better than Elite Solution then
9:         Remove old elite solution information from pheromone map (−Δτ_e)
10:        Replace elite solution
11:        Add new elite solution information into pheromone map (+Δτ_e)
12:    else
13:        if currentPopulationSize < k − 1 then
14:            Insert best solution into population
15:            Insert best solution information into pheromone map (+Δτ)
16:        else
17:            Remove oldest solution information from pheromone map (−Δτ)
18:            Remove oldest solution from population
19:            Add best solution to population
20:            Add best solution information into pheromone map (+Δτ)
21:        end if
22:    end if
23: end while
```

4.2.2 A Useful Population

An added advantage of maintaining a small population is that it provides the algorithm with a quick way to adjust the pheromone map if a change occurs to the problem being optimised, since the population can be re-evaluated and the pheromone map adjusted to reflect any change. In dynamic optimisation the speed with which the search algorithm can adapt its history is a strong determinant of its performance. In reference to ACO applied to dynamic optimisation, Guntsch and Middendorf [78]*pg 112 comment: *'it will usually be faster to modify a few solutions directly than to modify the whole pheromone information of a usual ACO algorithm'*. Although the FIFO-Queue ACO algorithm was applied successfully and competitively to static instances of the TSP and QAP problems, its real benefits were postulated to be in dynamic optimisation for the reasons mentioned above.

4.3 PACO for Dynamic Problems

In [77] the FIFO-Queue algorithm was renamed Population-based ACO (PACO) and was applied to instances of the dynamic TSP and dynamic QAP problems. The main contributions of this paper were the application of the PACO algorithm to dynamic problems, and the introduction and comparison of several population update strategies. Quite rightly steady-state genetic algorithms are quoted as being an inspiration for the strategies listed in the proceeding section.

4.3.1 Population Update Strategies

Age

This strategy is the same as that used in the original FIFO-Queue algorithm. The oldest solutions are removed from the population to be replaced by the newer incoming solutions.

Quality

If a new solution is better (in terms of quality) than the worst quality member of the population, the new solution replaces the worst solution, otherwise there is no change to the population. The aim of this strategy is that the population will retain good solutions which may have been found earlier in the search process. A possible weakness of this strategy is that there is no way to ensure that the population does not end up with what are essentially multiple copies of the same solution.

Probability

This strategy addresses the weakness of the *Quality* strategy by probabilistically replacing solutions in the population based on their quality. This way there is a chance that any solution from the population will be replaced with poor quality solutions being more likely to be replaced.

Age & Probability

This is a combination of two of the aforementioned strategies. Firstly the new candidate solution is added to the population, then a population member is selected for removal using the *Probability* scheme. Using this scheme it is possible that the new solution could be added to and removed from the population in a single iteration.

4.4 PACO for Multi-objective Problems

When applied to multi-objective problems PACO maintains a different pheromone matrix for each objective [79]. For each iteration of the algorithm, where iteration refers to every artificial ant creating a complete solution, a random ant is selected from the population (Q) along with its k closest neighbours[2] to form a sub-population P. At any time Q will contain the complete set of non-dominated solutions found to date. The ants in P are then used to update the individual pheromone matrix for each objective. When available a separate heuristic matrix is used for each objective, e.g. in the case of the TSP these heuristic matrices are simply the corresponding edge weights for each individually defined TSP.

[2]Closest neighbour refers to a match in the decision space, not the objective value space. Depending on the problem this distance could be obtained by taking a Euclidean or Hamming distance measure.

PACO uses an *average-rank-weight method* to weight the importance of each objective. These weightings (w) are used to bias the solution construction towards satisfying specific objectives over others. Briefly, the *average-rank-weight method* measures how well each solution in P satisfies each individual objective. Objectives which are better satisfied by the solutions in P relative to the entire population Q are given a higher rank and a subsequently larger weighting.

Once the pheromone matrices have been created and the objective weightings defined the transition probabilities are calculated using (4.4.1), where h is the total number of objectives. The Ant Colony Systems greedy transition rule [54] is then used to create one or more new solutions.

$$p_{ij} = \sum_{d=1}^{h} \left(w_d \cdot \frac{\left[\tau_{ij}^d\right]^\alpha \cdot \left[\eta_{ij}^d\right]^\beta}{\sum_{l \in N_i^k} \left[\tau_{il}^d\right]^\alpha \cdot \left[\eta_{il}^d\right]^\beta} \right) \tag{4.4.1}$$

Once created, each new solution (s) is evaluated for each objective. For s to be inserted into Q it must be checked for dominance against the entire population Q. If s is found to be non-dominated by all members of Q then s is inserted into Q. If s is inserted into Q then Q must be checked for dominance by s. If any existing solutions in Q are dominated by s they are removed from Q.

This approach was shown to be among the state-of-the-art ACO approaches for the multi-objective TSP in [70]. It was conjectured that the good performance of the algorithm can be attributed to the algorithm's ability to target specific areas of the approximate Pareto front for improvement. This is possible since the algorithm is able select few solutions from a larger population to create a temporary pheromone map.

4.5 ACO for Continuous Domains

To date, many ant-inspired approaches for application to CFO problems have been proposed. The Ant Colony Metaphor for Continuous Design Spaces [9] was the first ant-inspired search proposed for CFO. That algorithm starts by placing a 'nest' somewhere in the n-dimensional search space, after which it projects a group of vectors (ants) into the search space around the nest. Over successive iterations it gradually adjusts the direction of these vectors towards promising areas of the search space. Other approaches include ACO for Continuous Domains with Aggregation Pheromones Metaphor (APS) [155, 156], Continuous Interacting Ant Colony (CIAC) [59, 58] and Continuous ACO (CACO) [120]. Strictly speaking most of these approaches are ant-inspired but do not fit the criteria of ACO [142].

ACO for Continuous Domains (ACOCD) [142, 143] is an extension of PACO (Sec. 4.3). It maintains a population where every population member represents a single point in the n-dimensional search space. Each population member is also assigned a weight (w) which is used for selection purposes. Solution construction is achieved by way of sampling each dimension in turn (stepwise) using a combination of the population's Probability Density Functions to resolve each new point (Illustrated in Fig. 4.1). Newly constructed solutions are evaluated

(only if they fall within the bounds of the solution space) and inserted into the population using the PACO *Quality* population management strategy, albeit with a large population size (the minimum allowable population size is equal to the number of independent dimensions).

Algorithm 2 ACO for Continuous Domains (ACOCD)

1: Initialise history with uniform random solutions
2: **while** stopping criterion not met **do**
3: Rank population according to fitness (Fittest member $l = 1$)
4: **for** $i = 1$ to k **do**
5: Calculate selection probabilities for population according to (4.5.2)
6: **end for**
7: **for** $i = 1$ to m **do**
8: Select s using biased random selection
9: **for** $j = 1$ to n **do**
10: $s_i'^j \leftarrow$ Gaussian distributed random value with mean $\mu_l^j = s_l^j$ and standard deviation μ_l^j according to (4.5.3)
11: **end for**
12: Evaluate s_i'
13: **end for**
14: Select best new solution s' from all new solutions $(s_0', \ldots, s_m')$
15: **if** s' is better than worst solution in population **then**
16: Replace worst solution in population with s'
17: **end if**
18: **end while**

$$w_l = \frac{1}{q \times k \times \sqrt{2\pi}} e^{-\frac{(l-1)^2}{2q^2k^2}} \tag{4.5.1}$$

$$p_l = \frac{w_l}{\sum_{r=1}^{k} w_r} \tag{4.5.2}$$

$$\sigma_l^i = \xi \sum_{e=1}^{k} \frac{\left|x_e^i - x_l^i\right|}{k-1} \tag{4.5.3}$$

Where:

s	:	Solution in population
s'	:	New solution
s_l^i	:	i-th coordinate of l-th ranked solution
l	:	Solution rank
w_l	:	Weight of l-th ranked solution (fitness)
q	:	Solution selection greediness factor
k	:	Size of history
n	:	Number of problem dimensions
m	:	Number of ants
p_l	:	Probability of selecting the l-th ranked solution from the population
σ_l^i	:	Standard deviation of i-th dimension of l-th ranked solution (used for creating new solutions)
ξ	:	Convergence rate

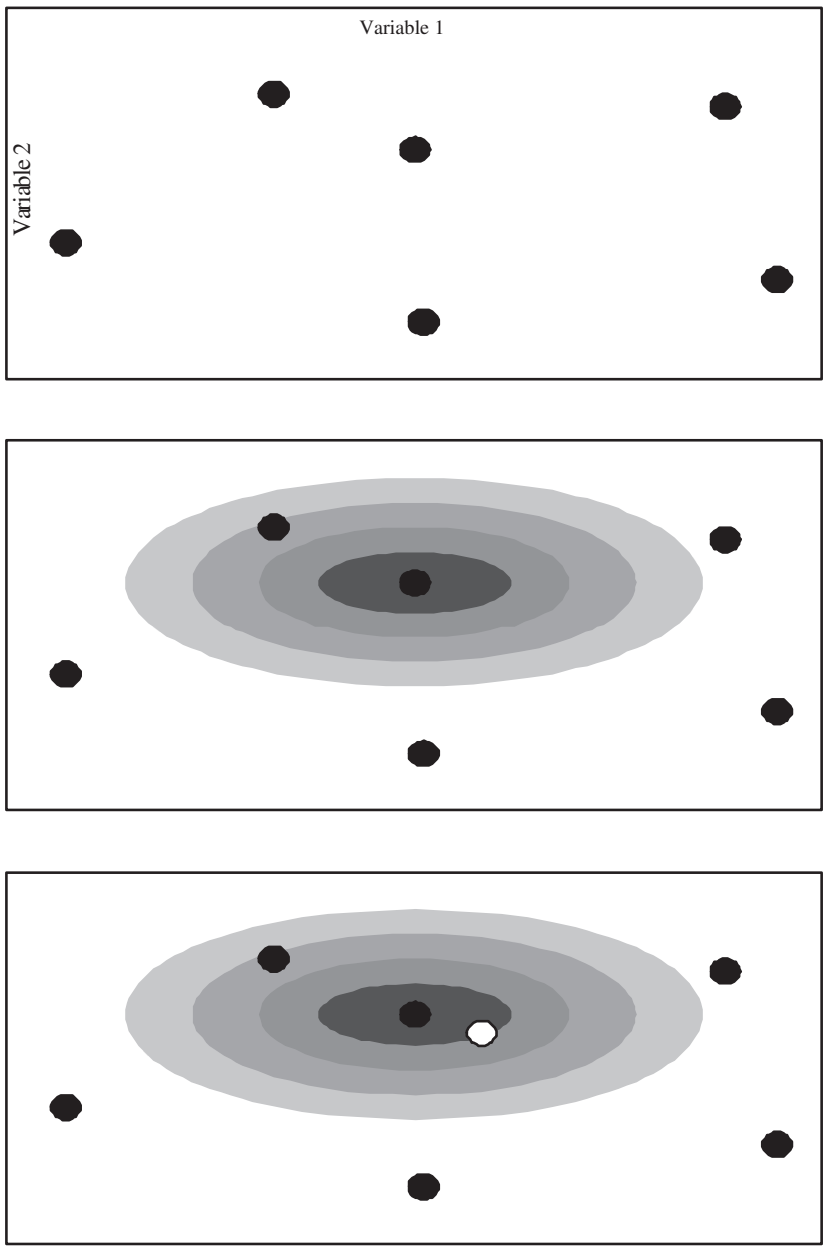


Figure 4.1: An illustration of the new solution creation procedure of ACOCD.

4.6 Hardware Implementation of ACO

Worth briefly mentioning is the successful hardware implementation of PACO demonstrated in [139, 137]. Primarily, applications of ACO are in software, operating on individual sequential machines. Some researchers have attempted to further improve the efficiency of ACO by implementing a multiple colony ACO where several ACO algorithms operate on parallel processors intermittently sharing pheromone information. These approaches are limited by the architecture on which they operate in that there are often large overheads associated in sharing information between colonies as well as the fact that they are often only able to share information about elite solutions. These communication overheads can be considerably reduced if the algorithm is embedded in hardware, as hardware approaches are often naturally parallel. The iterative structure and parallel nature of ACO makes it amenable to a hardware implementation.

ACO is not directly transferable to hardware for a number of practical reasons. Pheromone and heuristic values are usually represented as floating point numbers. Evaporation and update procedures involve a large number of multiplication operations (although it is possible to perform multiplication operations in hardware it is often costly). The use of exponential operators (α, β) to create probability tables is again a costly operation.

These problems are addressed by using an integer based pheromone scheme (Counter-based ACO [138]) coupled with a PACO approach to historic information storage. Pheromone values are calculated from the population as required, so that at no point in time does an entire pheromone map exist, only a partial pheromone map. Thus the *pheromone update & decay* process is performed as required to ensure all relevant information is contained in the pheromone map.

4.7 A Population-based ACO Meta-heuristic Framework

Arguments presented in this chapter for the use of PACO algorithms are:

- To speed-up the pheromone maintenance procedures allowing the algorithm to make fast changes in an unstable search environment increasing the overall search efficacy as in the case of FIFO-Queue ACO (Sec. 4.2).

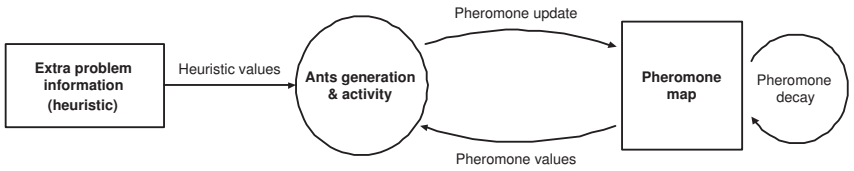
- To increase the applicability of ant-inspired search to continuous domains as in the case of ACO for continuous domains (Sec. 4.5).

- To allow for efficient hardware implementation (Sec. 4.6).

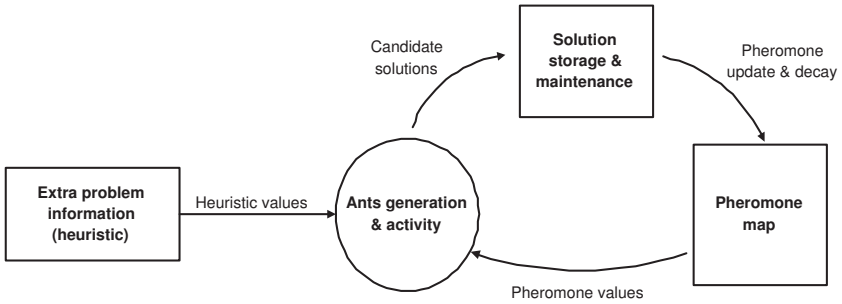
This section outlines a reinterpretation of PACO as a generic algorithmic framework which encompasses the features common to the aforementioned PACO algorithms, as well as allowing for future specialisations.

4.7.1 Process Level Organisation

PACO algorithms reuse many of the processes defined in ACO (discussed previously in Sec. 3.3) and Fig. 4.2 illustrates the process level organisation of ACO as compared to PACO. A subtle difference between ACO and PACO is the introduction of the *solution storage & maintenance* process and removal of the *pheromone decay* process.



(a) ACO process organisation



(b) PACO process organisation

Figure 4.2: Process organisation of original ACO framework versus the PACO framework

4.7.2 Solution Storage and Maintenance

This process is responsible for the storage of candidate solutions and their translation into pheromone information. Solutions generated by the *ants generation & activity* process are added to a finite sized population of solutions. There are many possible replacement schemes which can be implemented to customise the search behaviour of the algorithm. Several of these schemes were already discussed in Sec. 4.3

How the population interacts with the pheromone map is implementation specific. The first PACO algorithms used a concurrent update mechanism where the pheromone map is ever-present and as solutions are added and removed from the population they adjust the map to reflect this change. This tight coupling between pheromone map and population results in fast updates to the pheromone map. An alternative would be to use a temporary pheromone map such as that of PACO for multi-objective problems (Sec. 4.4) where at every algorithmic cycle the current population (or a sub-population) adjusts the pheromone values of an initially uniform pheromone map which is used to create the next batch of candidate solutions. After the next batch of candidate solutions is created this pheromone map is discarded and a new pheromone map created thus it is called a *temporary pheromone map*. In the Hardware-based ACO presented in Sec. 4.6 pheromone values are calculated from the population as required,

so that at no point in time does an entire pheromone mapping exist.

The traditional *pheromone decay* process is replaced by the *solution storage & maintenance* process. The population will at any time contain the most up-to-date solutions obtained by the algorithm and the use of any of the pheromone maintenance schemes outlined previously eliminate the requirement for decay. If a solution is removed from the population its associated pheromone information is either simultaneously removed or not added during the next algorithmic cycle.

As can be seen there are many ways to implement the *solution storage & maintenance* process. To reiterate, this process is responsible for the addition and removal of candidate solutions to a finite population of stored solutions. Furthermore this process is responsible in translating the population of solutions into pheromone information to be used by the *ants generation & activity* process.

4.8 Similarity to Genetic Algorithms

The similarities between ACO algorithms and GA[3] are known and have been previously discussed in numerous studies [10, 112, 27, 167]. This research highlights two particular classes of GA known as Probabilistic Model Based Genetic Algorithms (PMBGA) [118] and Estimation of Distribution Algorithms (EDA) [115] as the most similar. These methods, developed mostly throughout the mid to late 90s, are extensions of the canonical GA and share many similarities with ACO algorithms since they also use probabilistic models to rank solution components. Most comparisons centre on differences between traditional ACO algorithms (such as Ant Systems) and a typical PMBGA, the Population Based Incremental Learning (PBIL) algorithm [4]. Of importance here though is the similarity of PACO to algorithms such as PBIL since PACO algorithms have a more population centric focus which may mean that PACO algorithms are now the closest ant-inspired algorithms to the PMBGA family. In this section the PBIL algorithm is introduced and then a subsequent comparison between PBIL and ACO is offered to determine similarities and points of difference.

4.8.1 Population Based Incremental Learning

The PBIL algorithm (Alg. 3), like many other PMBGA, was motivated by an observed lack of efficacy when traditional GAs were applied to combinatorial optimisation problems such as the TSP that require recombination of many small solution components rather than recombination of few large solution components [118]. A key idea of PBIL is the replacement of the traditional GA Population with a single probability vector (length = l) that indicates for each bit the probability that it should be 1. This vector can then be used to spawn a generation of solutions (size = n) of which a single best solution can be used to update the probability vector to direct the algorithm towards more promising areas of the search space. The magnitude of the update is controlled by a learning rate (LR) that can be used to control algorithm convergence. Mutation

[3]For those readers unfamiliar with them, Genetic Algorithms are introduced in Sec. 5.2.1

is applied per bit with a frequency dependent on a mutation event parameter ($mutate_{event}$) and a magnitude controlled by another parameter ($mutate_{mag}$).

Algorithm 3 Population Based Incremental Learning (PBIL)

for $i = 1$ to l **do**
    $P_l = 0.5$ ▷ Initialise all elements of probability vector (P) as 0.5
end for
while stopping criterion not met **do**
    for $i = 1$ to n **do**
        Create new solution (s_i) by sampling P
        Evaluate s_i
    end for
    Select best new solution (s) from all new solutions ($s_1, \ldots, s_n$)
    for $i = 1$ to l **do**
        $P_l = P_l \times (1.0 - LR) + s_l \times LR$ ▷ Update probability vector
        if random$(0, 1] < mutate_{event}$ **then**
            $P_l = P_l \times (1.0 - mutate_{mag}) + \text{random}\ (0\ or\ 1) \times mutate_{mag}$ ▷ Mutate probability vector
        end if
    end for
end while

4.8.2 Similarities and Differences

There are many similarities between PBIL and PACO, and it is true that PACO pushes the ACO paradigm somewhat conceptually closer to PBIL. Some points of difference still exist though and these are commented on here.

Solution construction in both instances is achieved by sampling a probability distribution in a stepwise manner, however PBIL does not incorporate heuristic information into this process. This said, PACO algorithms like ACOCD do not incorporate heuristic information either and so depending on their application this distinction may or may not exist.

The canonical PBIL only uses the best solution found in a single generation to update the probability vector, whereas it is possible for PACO algorithms to use many solutions of varied quality to influence the pheromone information. As with the incorporation of heuristics though it could be possible for PACO to be run with only an individual elite solution updating the population in a generation. This is the case with the original PACO algorithm, FIFO-Queue ACO, which was parameterised in this manner in some experiments.

One of the more obvious differences between PBIL and ACO is the solution representation. PBIL, like the canonical GA, encodes solutions as strings of bits, whereas the canonical ACO algorithms tend to use higher level encodings incorporating real numbers or integers as solution component identifiers. Exceptions as in the case of the RVGA do exist and it would be fairly trivial to make this change to the PBIL algorithm.

These observations should highlight that these techniques are indeed very similar. Given the right parameterisation it could be quite possible for an existing PACO algorithm to be almost identical to PBIL. However this is exactly the point to be made, these exact parameter settings

do not exist for any one PACO algorithm, as such there was not one PACO algorithm reviewed which was the same as PBIL.

4.8.3 General Comments

The development of biologically inspired techniques are shaped by the point where they first depart from the biological system. Core ideas from these inspirations emerge and it is natural that researchers take these ideas and apply them where they are useful. In short it is entirely possible that researchers while starting from one inspiration find themselves close, or closer, to another inspirational source. These cases are often interesting in their own right since they can benefit both communities by identifying the transition from one field to another, helping to paint a complete picture of biologically inspired computation.

At first inception the GA and ACO communities were quite distinct, however through ongoing research their respective boundaries have become more blurred. This should be quite evident in the work of this thesis, which blurs the boundary even more by augmenting ACO algorithms with niching, a concept developed in the GA community. Of course one must be careful that any newly developed hybridisation does not simply rename an existing work. Thus the recent creation of a sub-field, Evolutionary Algorithms based on Probabilistic Models [102], aims to encompass existing work and provide clear research directions for the GA and ACO communities.

All care has been taken in this thesis to ensure that the algorithms and subsequent analysis are novel in their own right. Even though the work presented here is developed using the ACO paradigm it must be made clear that it could just as easily have been developed from a PMBGA or EDA perspective. This choice is a fairly arbitrary one, biased by the authors prior familiarity with ACO. It is in the ACO field that a majority of inspiration for this thesis is derived, to be clear though this choice does not reflect any advantage of one biological inspiration over the other.

4.9 Chapter Summary

The initial motivation for the introduction of a discrete population to ACO was to address dynamic optimisation problems using the population to achieve fast pheromone updates. From the examples presented in this chapter though it is clear that several other benefits exist. PACO is useful in addressing other problem domains such as CFO, as was demonstrated in Sec. 4.5, and multiple objective optimisation as in Sec. 4.4. It is also a suitable way to address several shortcomings when implementing ACO in hardware (Sec. 4.6). While the algorithms presented differ very minimally from the original PACO algorithm, FIFO-Queue ACO, each is useful in demonstrating how PACO can be applied to a variety of different optimisation problems.

After describing the existing PACO algorithms a common framework encompassing these algorithms was detailed. This framework, like the ACO Meta-heuristic framework, is a useful

way to describe these many different algorithms and from hereon PACO refers to this general description. An important observation made was the similarities between PACO and GA, realised mostly due to the addition of a permanent population structure. Benefits of this include the ability to take ideas from EC algorithms (like the GA) and apply them directly to create novel PACO algorithms, or allow hybridisation of ACO algorithms with other EC algorithms, previously difficult due to differing historical information storage methods i.e., the lack of a permanent population structure in ACO.

The next chapter describes two novel niching PACO algorithms which use niching concepts from Evolutionary Computation. As will be discussed in the next chapter these niching adaptations are made possible through the addition of a population to ACO.

CHAPTER 5

Niching for PACO

> The question that will decide our destiny is not whether we shall expand into space. It is: shall we be one species or a million? A million species will not exhaust the ecological niches that are awaiting the arrival of intelligence.
>
> Freeman Dyson

5.1 Introduction

The concept of diversity has been mentioned as an important factor in algorithm design. For some problems diversity may not be required to obtain optimal solutions while in others it is critical. This chapter discusses a particular diversity preservation technique called *niching*.

This chapter firstly introduces Evolutionary Computation (EC) as an algorithm class. Several EC algorithms which use niching as a diversity preservation technique are presented alongside motivations for their use. Finally, two novel niching PACO algorithms are presented. These algorithms will be the subject of further analysis for the remainder of the thesis.

5.2 Evolutionary Computation

Evolutionary Computation (EC)[1] is a field of research dedicated to the study of algorithms which maintain some form of solution memory which is used to bias future solution creation. Some common examples of algorithms fitting into this broad definition of EC are:

- Genetic Algorithms (GA) [81, 82, 71]
- Genetic Programming (GP) [61, 96]
- Evolutionary Strategies (ES) [125, 140, 8]
- Differential Evolution (DE) [145]

[1] Evolutionary Computation derives its name from its similarities with Darwinian Evolution.

- Simulated Annealing (SA) [91, 20]
- Particle Swarm Optimisation (PSO) [90]
- Ant Colony Optimisation (ACO) (Sec.3)

The descriptor Evolutionary Computation was first used around 1993 to unite the GA, GP and ES fields. However, in recent times EC has grown to include many more algorithms. This inclusion has enabled an increase in the amount of reuse and hybridisation of concepts between fields. As a generic algorithm methodology EC can be visualised as Fig. 5.1

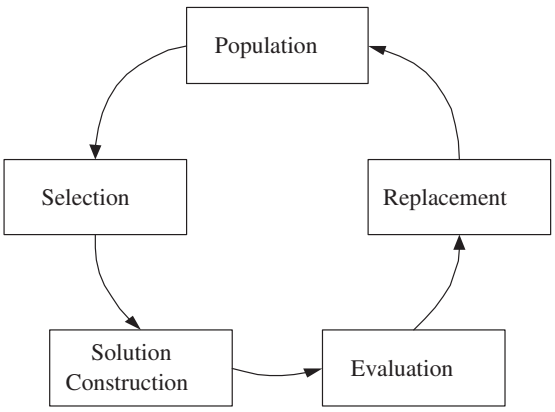


Figure 5.1: Generic algorithmic structure of Evolutionary Computation

5.2.1 Genetic Algorithm

The canonical Genetic Algorithm (GA) [81, 82, 71] differs from other EC algorithms by its solution representation and combination of selection, recombination, mutation and replacement. Solutions were normally binary strings of a fixed length governed by the number of problem variables and precision required for each variable. While much of the early GA used binary strings, later work also includes solutions encoded as arrays of floating point numbers (Real Value Genetic Algorithm – RVGA). The GA is usually initialised with a population of random solutions, although some 'seed' the initial population with solutions which are thought to be good.

After initialisation the GA loops through the following processes until some termination criteria is met:

1. *Selection* – Select 'parent' solutions from the population using a fitness proportionate selection mechanism such as tournament selection or biased roulette wheel selection[3].
2. *Recombination* – Also known as crossover, components of the selected solutions (parents) are mixed to create new candidate solutions (children).
3. *Mutation* – Mutation involves targeting a candidate solution's individual solution components and perturbing them (e.g. flipping a bit value) to introduce local variation.

4. *Evaluation* – The candidate solution(s) are evaluated using a fitness function to determine their utility.

5. *Replacement* – Candidate solutions are inserted into the population usually replacing older or less fit solutions.

5.3 Niching in Evolutionary Computation

The concept of the niche was first introduced in [75], although the definition has altered somewhat since its introduction. In [87] it was suggested that each species occupies its own unique niche within the environment. To define a species' niche we firstly take every facet of the natural environment (e.g. temperature, humidity, abundance of predators, availability of food) and represent them along individual dimensions creating an n-dimensional hyperspace, the environment. The n-dimensional volume defined by mapping a particular species in this space to where we know it could possibly survive is this species' fundamental niche. The same species mapped in this space but restricted to where we know the species does exist will encompass a smaller n-dimensional volume which represents this species' ecological niche. A species' niche can be a useful way to describe the behaviour of this species in a particular environment e.g. removal of competition for resources may result in a species' ecological niche expanding to fill more of its fundamental niche.

Niching as an EC concept was first formally applied to the GA but has also been applied to other EC algorithms such as PSO [13, 14]. Most EC algorithms tend to focus their search in individual areas of the search space. Niching aims to diversify the search focus of an EC algorithm to multiple areas of the search space which is beneficial for multi-modal and multi-objective problems.

In [71] three key features of niching algorithms are presented:

1. Stable maintenance of subpopulations. Once an optimal area of the search space has been located a niching algorithm should maintain several population members in that location so as not to lose (forget) this area of interest.

2. The size of a subpopulation decreases according to the fitness of the area of interest. Considering the limited resources of an algorithm, exploration of any area of the search space should be proportional to the potential quality of solutions returned from a niche.

3. Subpopulations should not compete. Since resources are most likely to be of a fixed size, a niching algorithm should not allow subpopulations to 'fight' for dominance of one area of the search space.

Mahfoud [105] provides a classification scheme for niching algorithms:

1. Spatial: Algorithms that form niches in one algorithm run, sometimes called parallel niching.

(a) Intra-Population (Population): All niches are formed within a single population.

(b) Inter-Population (Geographical): Niches are formed across multiple populations (island populations).

2. Temporal: Algorithms form niches in subsequent sequential runs usually through use of some form of restart strategy (sequential niching).

There have been so many different niching methods presented in recent years that a comprehensive study of all is not possible within the scope of this thesis. Instead two parallel niching methods have been included since they are perhaps the most important historically and will be used later in the thesis. These niching methods are crowding (Sec. 5.3.1) and fitness sharing (Sec. 5.3.2).

Some niching methods require the calculation of similarity measurements between solutions such as a Euclidean distance or a Hamming distance. These measurements are explained in more detail in Sec. 5.3.3.

5.3.1 Crowding

The *crowding factor* model was introduced as a diversity maintenance scheme by De Jong [39]. This scheme was not a niching method according to the criteria outlined in Sec. 5.3, as it was used to increase population diversity, not to locate and maintain multiple optimal areas of the search space [80]. However, the crowding factor model laid the foundation for much of the later work on niching and was most significantly reworked as a niching strategy by Mahfoud [103, 105].

Mahfoud's technique, *Deterministic Crowding*, was designed to slow convergence and maintain diversity by limiting dominant building blocks in a population. The replacement policy in deterministic crowding involves a candidate solution competing (based on solution quality) for a place in the history with the most similar of its parent solutions. If a candidate solution is better than its most similar parent, the parent is replaced, otherwise the candidate solution is discarded.

Another variation of Crowding is *Restricted Tournament Selection*(RTS) [80]. RTS is similar to Deterministic Crowding, but instead of a candidate solution only being compared against its parents, it is compared to a subset of the entire population. The size of this subset in called the window size and the most similar solution from this subset is sought and is replaced if the candidate solution is of higher fitness.

5.3.2 Fitness Sharing

Instead of modifying the replacement mechanism to introduce niching behaviour, as in Crowding, Fitness Sharing [72, 71] modifies the selection mechanism. Since most Genetic Algorithms use a fitness proportionate selection mechanism to select parent solutions, Goldberg and Rich-

ardson found that modifying the selection of parents will avoid convergence to one area of the search space thereby introducing niche formation.

Fitness Sharing derates[2] solutions which occupy the same or a similar position in the search space. For example the adjusted quality (Q') of two solutions which occupy the same position in the search space will be half of their original quality (Q). The result of this adjustment is to remove the selection bias present through a solution being represented multiple times. The removal of this bias allows simultaneous convergence to multiple areas of the search space.

Niche formation, specifically with regard to Fitness Sharing, was explained in [72] using a variation of the k-armed bandit problem [82, 39, 129]. The problem involves a poker machine with k handles, each handle having set pay-out odds. There is also a population of gamblers, who wish to maximise their individual winnings from the poker machine. The variation introduced is that after every gambler has selected a handle they must share their winnings with everyone else who chose their particular handle. For example if there are two handles each having expected payouts of \$25 and \$75 and 100 gamblers all pull the better handle they would each walk away with $\$75/100 = \$0.75$. If however the population divides across both handles proportionate to the expected payout of those handles the expected payout per gambler will be $\$75/75 = \$25/25 = \$1.00$. From this simple experiment it was shown that modifying the payout function in the k-armed bandit problem can introduce a reward for niche formation.

$$Q'_i = \frac{Q_i}{c_i} \tag{5.3.1}$$

$$c_i = \sum_{j=1}^{m} sh\left(d_{ij}\right) \tag{5.3.2}$$

$$sh\left(d\right) = \begin{cases} 1 - \left(\frac{d}{\sigma_{share}}\right)^{\alpha} & \text{if } d < \sigma_{share}, \\ 0 & \text{otherwise.} \end{cases} \tag{5.3.3}$$

Where:

Q_i	:	Quality of individual solution i
c_i	:	Niche count for solution i
d_{ij}	:	Distance (difference) between solution i and solution j
$sh\left(x\right)$	:	Sharing function
σ_i	:	Niche radius
α	:	Power used to modify the shape of the sharing function

With Equ. 5.3.1, 5.3.2 & 5.3.3 if a solution is alone in a neighbourhood (i.e. its nearest neighbour is of a distance $> \sigma_{share}$), then its quality is unaltered since: $sh\left(d\right) = 1$ for itself, and this will be the only contribution to the niche count (c_i), ($Q'_i = Q_i$ in this case). As presented previously, if two identical solutions are acted upon with the fitness sharing equations then their respective

[2]Derate is a commonly used term in niching literature that means 'to decrease the fitness of'.

quality will be halved since the niche count $(c_i) = 2$, and this effectively halves the quality of each solution $(Q_i' = Q_i/2)$.

5.3.3 Difference Measures

Difference can be calculated (or estimated) in the phenotypic or genotypic space. In the genotypic space the problem encoding will define the use of difference (similarity) measurement. A common method used to calculate the genotypic similarity of solutions encoded as permutations (e.g. TSP) is to use adjacency matrices for each solution and calculate the Hamming Distance [114, 131].

Firstly the adjacency matrix of each permutation (Tab. 5.1 & 5.3) is constructed (Tab. 5.2 & 5.4). The adjacency matrix is constructed according to the rule: If two solution elements(cities) are adjacent to each other in a permutation this element is set to 1 in the adjacency matrix, otherwise it is set to 0. For example, in the case of a TSP the adjacency matrix highlights the edge connections of a candidate TSP solution.

The adjacency matrices (Tab. 5.2 & 5.4) are insensitive to the starting city of the solution, i.e. solution 3-4-5-6-1-2 would produce the same matrix as that shown in Tab. 5.2. This insensitivity to starting position is important since the starting position is irrelevant when comparing solutions to a TSP.

The Hamming distance can be calculated by using an AND operation on each solutions adjacency matrix and summing all of the values in the matrix (Tab. 5.5). To calculate the shared edges between permutations of a symmetric TSP it is necessary to divide the Hamming distance by 2; for this example the shared edges would equal 4. In order to obtain a difference measurement it is necessary to alter the edge count using Equ. 5.3.4. Using Equ. 5.3.4, one would represent the solutions being completely different i.e. sharing no common edges, whereas zero represents the solutions being exactly the same.

$$\text{Difference} = 1 - \frac{\text{shared edges}}{\text{number of cities}} \tag{5.3.4}$$

In CFO problems where solutions genotypic information are points in some n-dimensional hyperspace a Euclidean distance can be used to calculate the genotypic distance between two solutions (x, y) as in Equ. 5.3.5.

$$\text{EuclideanDistance} = \sum_{i=1}^{n} \sqrt{(x_i - y_i)^2} \tag{5.3.5}$$

The use of phenotypic distance is usually avoided in niching since it does not provide an accurate reflection of the spatial separation of solutions, i.e., two solutions with similar fitness may be located on separate optima a large genotypic distance away from each other.

Position	1	2	3	4	5	6
City	1	2	3	4	5	6

Table 5.1: Example candidate solution 1

Cities	1	2	3	4	5	6
1	0	1	0	0	0	1
2	1	0	1	0	0	0
3	0	1	0	1	0	0
4	0	0	1	0	1	0
5	0	0	0	1	0	1
6	1	0	0	0	1	0

Table 5.2: Example candidate solution 1 adjacency matrix

Position	1	2	3	4	5	6
City	1	2	4	3	5	6

Table 5.3: Example candidate solution 2

Cities	1	2	3	4	5	6
1	0	1	0	0	0	1
2	1	0	0	1	0	0
3	0	0	0	1	1	0
4	0	1	1	0	0	0
5	0	0	1	0	0	1
6	1	0	0	0	1	0

Table 5.4: Example candidate solution 2 adjacency matrix

Cities	1	2	3	4	5	6
1	0	1	0	0	0	1
2	1	0	0	0	0	0
3	0	0	0	1	0	0
4	0	0	1	0	0	0
5	0	0	0	0	0	1
6	1	0	0	0	1	0

Table 5.5: AND operation applied to solution 1 & 2 adjacency matrices

5.3.4 Pros and Cons of Niching

Most search algorithms are designed (or were initially designed) to find a single optimal solution to a difficult problem. This design trait is borne of the problems that exist in the literature where the goal tends to be optimisation of a single objective, some of which where discussed in Sec. 2. Niching algorithms tend to be best applied in situations where multiple optimal solutions are required rather than a single optimum, such as in multi-modal and multiple objective problems.

Alternatively, niching algorithms can also be applied in situations where a single optimum solution is desired. This may be because some search advantage is gained over non-niching algorithms due to the specific search landscape of the problem. Niching algorithms in this sense may permit a more effective use of available resources by a search algorithm by either implicitly or explicitly dividing and searching different areas of the search space in parallel [106]. Such automatic resource re-allocation is useful in preserving useful population diversity for an extended search time.

Since niching techniques spread the population across a wider search area, they can often waste computation by continually searching areas of the search space where no interesting optima exist. Also recombination of solution components from different niches can lead to the introduction of 'lethals' [38] which are solutions created between two optima but not located on an optima themselves. Some niching techniques introduce extra parameters in addition to the base algorithm. These parameters often come without good heuristics to set them and thus require extra sensitivity analysis to ensure best performance [160].

5.3.5 Sequential versus Parallel Niching

The Sequential Niching Genetic Algorithm (SNGA) [7] which is typical of many sequential niching techniques works as follows:

1. A randomly initialised GA is used to search the search space until convergence or some alternative stopping criteria is met.

2. The location of the best optima found and saved.

3. The fitness of this optima as well as the fitness of neighbouring points for a specified radius (much like the sharing radius of fitness sharing) is decreased.

4. Repeat steps 1-3 until a finishing criteria is met.

In [104], Mahfoud compared four niching methods on a variety of multi-modal CFO problems: fitness sharing, deterministic crowding, parallel hill-climbing and the SNGA. The study showed that parallel hill-climbing performed best on simple problems, however it was fitness sharing and deterministic crowding that performed best across all problems, particularly those with complex search spaces. SNGA did not perform well on any of the test problems and per-

formed particularly poorly on the complex problems. This poor performance was attributed to a number of factors:

- The deration of optimal solutions restricts their good solution components from being reused to find other optima which also use these good solution components.
- The deration of areas of the search space may create false optima or eliminate some optima entirely.
- While it may be easy to locate optima at the start of the search, some optima may be left located in areas of uniform fitness where they will be harder to find at the end of the search due to an absence of directional information.
- The algorithm repeatedly converges to the same optima, despite the deration of these optima.

Other detractors of the algorithm included a slower runtime than the parallel niching algorithms and the number of evaluations of SNGA being between 5-8 times that of fitness sharing. The large number of evaluations was attributed to the same optima being re-discovered over subsequent restarts. Also, since the SNGA had to start from random positions each restart, it meant that the algorithm wasted a lot of time re-exploring areas of the search space previously explored.

In a practical context, Horn's doctoral thesis [84] outlined a pipeline design problem that was shown to contain five optima of varied cost. Of these optima it was noted that several were found more frequently than others using a restart strategy, however the parallel niching model presented in the thesis was able to identify all five optima in a single run.

5.4 Niching Ant Colony Optimisation

5.4.1 Constraints

To be able to easily implement standard niching methods such as crowding and fitness sharing with ACO, the ability to readily measure the distance between solutions is required. Since most standard ACO algorithms store solution quality (phenotypic) information in the pheromone map but don't store the actual solutions (genotypic) we cannot access genotypic distance information between multiple generations of solutions. While it may be possible to achieve a niche-like behaviour in ACO, it is much more straightforward to use PACO (Sec. 4) to implement standard niching techniques such as crowding and fitness sharing, since in PACO, access to a multi-generational population in the traditional EC sense is guaranteed.

5.4.2 Fitness Sharing Population-based Ant Colony Optimisation

Fitness Sharing PACO (FSPACO) uses a generational replacement where each generation all newly created solutions are inserted into the population, supplanting the oldest solutions (much

like FIFO-queue ACO in Sec. 4.2). After the population is updated, the fitness sharing equations (Sec. 5.3.2) are applied to the entire population to determine how much each solution's fitness is derated by (if any). Once the adjusted fitness values are calculated, a temporary pheromone map is constructed using the entire population. This temporary pheromone map is used for the construction of the next generation of solutions then discarded. A general pseudo code representation is provided as Alg. 4. In this pseudo code example p represents the population, and p_i the ith member of that population.

Algorithm 4 Fitness Sharing Population-based Ant Colony Optimisation: FS-PACO

```
1: while stopping criterion not met do
2:     Construct temporary pheromone matrix
3:     Construct Solutions
4:     Update history (Replace oldest solutions)
5:     De-rate quality
6: end while
7: procedure DE-RATE QUALITY
8:     for j = 1 to p_size do
9:         nicheCount = 0
10:        for k = 1 to p_size do
11:            d = distance(p_j, p_k)
12:            if d < σ then
13:                shareValue = (1 − (d/σ)^α)
14:            else
15:                shareValue = 0
16:            end if
17:            nicheCount = nicheCount + shareValue
18:        end for
19:        h_j.quality = h_j.quality / nicheCount
20:    end for
21: end procedure
```

5.4.3 Crowding Population-based Ant Colony Optimisation

Unlike the GA, PACO algorithms do not select parent solutions for new solution construction and the pheromone map tends to reflect the experience of the entire population. Given this, Restricted Tournament Selection is a suitable crowding model since it compares new solutions against a subset of the entire population. The size of the subset selected is denoted as the crowding size (window size), thus Crowding PACO (CPACO) introduces one extra parameter to the basic PACO algorithm. The crowding size can be set anywhere between one and the size of the population. A general pseudo code representation of Crowding PACO (CPACO) is outlined in Alg 5.

5.4.4 Alternative ACO Diversity Preservation Methods

As indicated in Horn's doctoral dissertation[84] it is prudent to discuss alternative diversity preservation mechanisms alongside niching, as niching is itself a form of diversity preservation. Focusing on ACO algorithms, the issue of diversification versus intensification has been

Algorithm 5 Crowding Population-based Ant Colony Optimisation: CPACO

```
1: Uniformly initialise pheromone map values to τ_init
2: for j = 1 to h do
3:     Create and evaluate random solution
4:     Insert random solution into history
5:     Add random solution information into pheromone map (+Δτ)
6: end for
7: while stopping criterion not met do
8:     Construct m new solutions (s^new)
9:     Evaluate solutions
10:    Crowding history update
11: end while
12: procedure CROWDING HISTORY UPDATE
13:    for j = 1 to m do
14:        Select random subset (size=c) of solutions (s) from population (p)
15:        for k = 1 to c do
16:            d = distance(s_j^new, s_k)
17:            if d < leastDistance then
18:                leastDistance = d
19:                s_closest = s_k
20:            end if
21:        end for
22:        if s_j^new.quality > s_closest.quality then
23:            Remove s_closest information from pheromone map (−Δτ)
24:            Remove s_closest from population
25:            Add s_j^new to population
26:            Add s_j^new information into pheromone map (+Δτ)
27:        end if
28:    end for
29: end procedure
```

a driving force behind the development of ACO and the balance between these factors is often cited as one of the distinguishing features of different ACO algorithms. This is particularly evident in algorithms such as ACS (Sec. 3.4.2) that use elite solutions to promote intensification and localised decay to ensure diversification.

Some research specifically targets the issue of diversification by adding features to basic ACO algorithms which introduce randomness [116, 68, 123, 124] dependent on different criteria such as the measured diversity of the population. For the problems addressed those techniques allow the modified algorithms to overcome some of the issues associated with premature convergence to suboptimal solutions. While these approaches do improve the basic ACO algorithms they are very different to niching. These modifications usually delay convergence to a single area of the search space. While niching aims at increasing diversity, it does not hold off convergence or prevent it, nor does it purposely introduce extra randomness. Niching aims at creating and maintaining stable subpopulations. This is quite different to slowing convergence or introducing randomness since niching algorithms can be tuned to be highly convergent, but convergent to multiple areas of the search space.

5.5 Chapter Summary

This chapter discussed two popular niching techniques, fitness sharing and crowding. Two new PACO algorithms which incorporate these forms of niching were presented. In the next chapter these algorithms are applied to a group of single-objective optimisation problems to characterise their performance and provide insight into their behaviour.

CHAPTER 6

Single Objective Optimisation

> Normal people don't understand this concept; they believe that if it ain't broke, don't fix it. Engineers believe that if it ain't broke, it doesn't have enough features yet.
>
> Scott Adams, *The Dilbert Principle*, 1996

6.1 Introduction

Historically the TSP is the first application for most new ACO algorithm variations, and this has been used in the analysis of Niching PACO. While it is not expected that the niching adaptation will be useful for TSP an analysis is still included. The intention of such an analysis is that it should allow the identification of interesting features, since the Niching PACO can be directly compared to existing non-niching ACO algorithms. After this analysis the Niching PACO algorithms are applied to several CFO problems, since this is a problem domain that most niching algorithms tend to predominantly be applied to. The choice of this domain is mostly due to the ability to readily design challenging benchmark problems with features such as multiple diverse optima. The problems are also easily scaled to multiple dimensions which can rapidly increase the computational complexity, but such scaling does not necessarily come at a cost to our ability to analyse and comment on algorithms applied to them. This is because these CFO problems allow for easy visualisation of their search space, while still preserving their neighbourhood relationships. This is not so true of combinatorial domains whose search spaces tend to be quite easily disrupted if visualised in a lower number of dimensions[1]. In this later analysis two criterion for success are used, the algorithms' ability to locate and maintain multiple optimal or near-optimal solutions, and the best quality optimum found overall. A set of metrics are defined which are useful in benchmarking the Niching ACO algorithms against standard niching Genetic Algorithms and ACO approaches to the CFO problem.

[1] Techniques exist to map all solutions to an n-dimensional TSP into a 2-dimensional space [154], however these techniques fail to preserve the neighbourhood relationship of the TSP.

6.2 Niching PACO for the Travelling Salesman Problem

The TSP is perhaps the most common problem used by the ACO research community. As was previously mentioned it is useful to include an analysis of the Niching PACO algorithms applied to the TSP for comparison purposes with other well established ACO algorithms. In this section implementation details of the Niching PACO algorithms are outlined. The algorithms are tested on several standard benchmark and one original TSP instance. Results of testing are included and reasons for the algorithms' performance given.

6.2.1 Algorithm Details

FSPACO-TSP and CPACO-TSP are similar to the PACO algorithm presented in Sec. 4.3 but for a few differences which are discussed in this section.

Solution Construction (Ants Generation & Activity)

Solution construction is performed in the same way as that of ACS, MMAS and PACO for TSP by using the greedy transition rule. At each transition a random number between 0 and 1 is selected and if this number is above the threshold Q the absolute best solution component is added to the current solution, otherwise a solution component is probabilistically selected. Like any other ACO-TSP algorithm a tabu list that restricts possible solution components to those cities not yet visited is used.

Pheromone Update

FSPACO-TSP uses an inverse tour length to determine the original solution quality and this quality is then adjusted using the niche count. The adjusted quality is then used to update the pheromone values of a temporary pheromone matrix which is reinitialised before update at every iteration.

CPACO-TSP uses the inverse of the tour length to update the pheromone values, which means that the PACO fast pheromone update procedure can be used. A rank based approach was considered but rejected on the grounds that if used it would mean that a temporary pheromone matrix would be required. This is because re-ranking of solutions may mean that solutions that have already updated the pheromone values when entering the population would need to adjust their pheromone values if they change rank.

Solution Storage and Maintenance

For FSPACO-TSP a generational replacement method is implemented where at each iteration the entire population is replaced with the newly created solutions. For CPACO-TSP the crowding comparison operation explained in Sec.5.4.3 is used to manage the insertion of new solutions into the population. For both algorithms an initial population is created using a uniform

random sampling of the the search space.

6.2.2 Results and Discussion: Crown Problem

At the outset applying a niching algorithm to the TSP does not seem to be a good idea since it has been shown that the TSP seems to benefit from greedy search behaviour, focused on one area of the search space at a time [150]. In their work on $\mathcal{MAX} - \mathcal{MIN}$ Ant System (MMAS) [147, 148, 149], Stützle and Hoos comment that Ant Systems (AS) performs poorly on the TSP in comparison to later ACO algorithms because AS does not exploit good solutions strongly enough. This is true since all of the ACO algorithms mentioned in Sec. 3 that have improved on the results of AS for problems such as the TSP include variations such as greedy transition rules or pheromone update rules that strongly bias the reinforcement of elite solutions. Since elitism tends to correlate strongly with an increase in search efficacy this may indicate something about the problem: that the n-best solutions to a TSP all contain similar elements and thus are located in a similar area of the search space. To test this the best 100 solutions to a small TSP instance[2], the Burma14 have been compared to each other to determine their similarity. Figure 6.1 details the average similarity (number of common edges) of each of the best 100 solutions compared to the other 99 best solutions. In this figure the most similar any two solutions can be without being identical is 12 edges. Error bars are included to indicate the standard deviation of each average taken. For comparison 100 random solutions have also been selected and subjected to the same test.

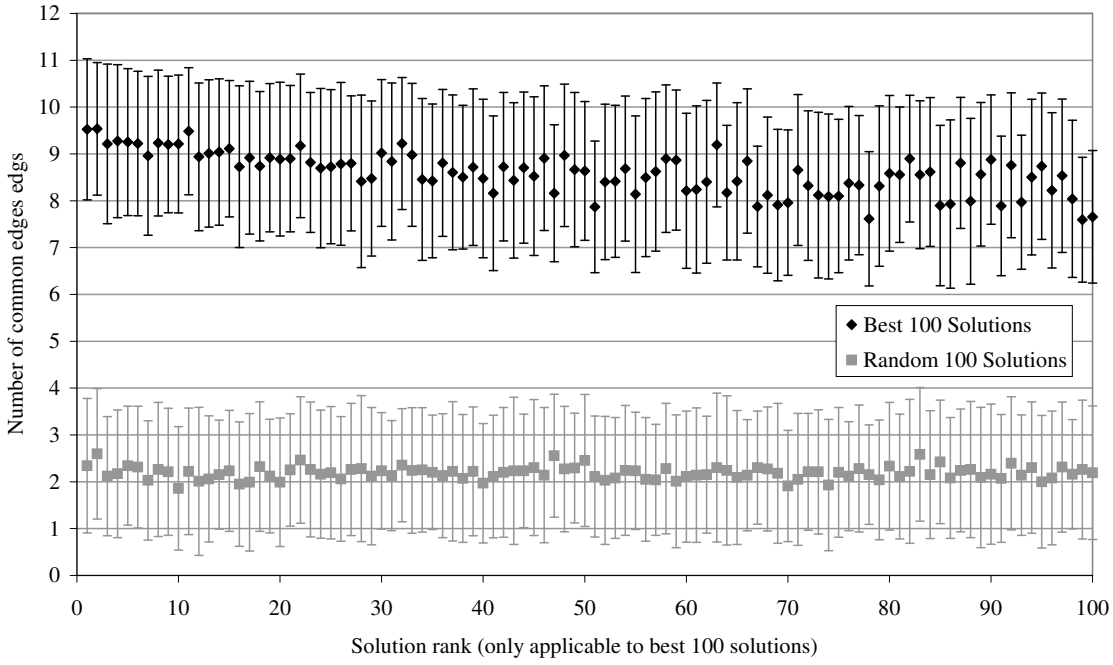


Figure 6.1: Average number of similar edges amongst the best 100 and a randomly selected 100 solutions to the Burma14 TSP

As is indicated in Fig.6.1 the best 100 solutions all occupy a similar space in the overall search space having on average 9 similar edges to all of the other best solutions. The baseline random

[2]Since an exhaustive search was required to find the best 100 solutions only a small instance could be used.

sample indicates that from a random sample of 100 solutions only 2 edges are the same on average when selecting from across the entire search space.

Since niching algorithms strive to maintain diversity, it is intuitive that niching algorithms are not suited to solving problems such as the single-objective TSP that benefit from strong convergent behaviour. However, cases may exist where a niching algorithm would be suited to solving the TSP, to examine let's consider a fabricated TSP: the Crown problem.

The Crown problem is a symmetric, 2-Dimensional Euclidean TSP containing 6 vertices (Fig. 6.2)[3], and has the interesting property of containing two distinct yet equal global optima (Fig. 6.3). These optima are also a reasonable distance apart only sharing 3 out of 6 edges (difference = 0.5).

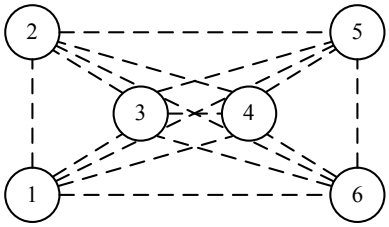
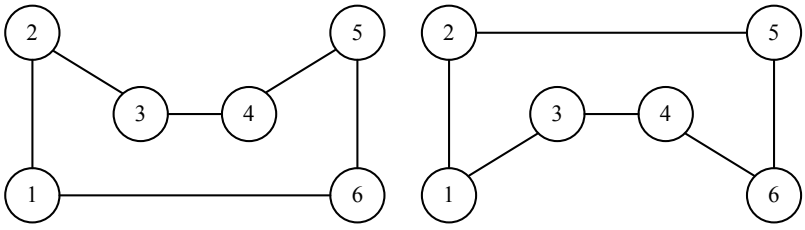


Figure 6.2: Crown TSP problem visual representation



(a) A - 1,2,3,4,5,6 length = 541 **(b)** B - 1,2,5,6,4,3 length = 541

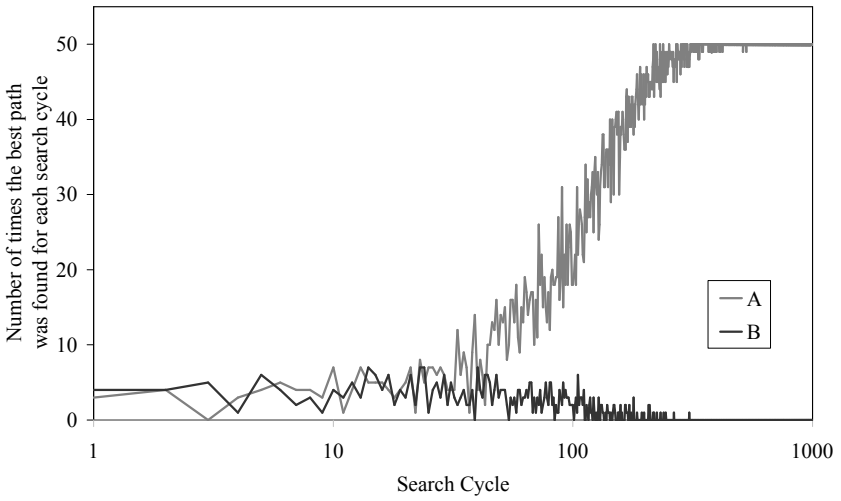
Figure 6.3: Optimal solutions to Crown problem

For comparison purposes the MMAS algorithm was tested[4] using 50 ants per iteration, and the number of times either of the two optima were found per iteration was recorded. This experiment was repeated 100 times for consistency of reported results. The graphs presented in Fig. 6.4a, Fig. 6.4b and Fig. 6.4c, although illustrating single experimental runs, are indicative of each algorithms behaviour. These graphs indicate that on any individual experimental run the MMAS algorithm is able to locate both optima (not surprising given the small size of the search space) however, over time the algorithm will converge to only one of these optima. Conversely both FSPACO-TSP and CPACO-TSP are able to locate and maintain both optima for an extended period of time.

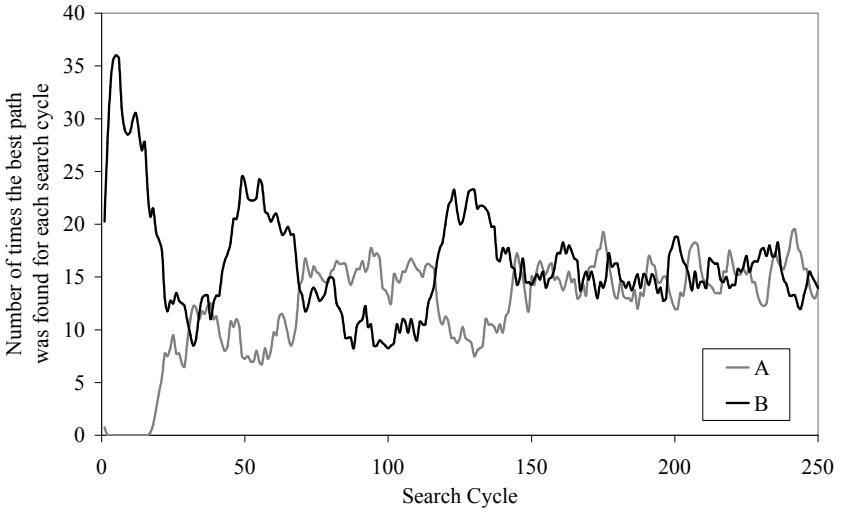
6.2.3 Results and Discussion: Standard Problems

The Crown problem is trivially small and quite unique, with regard to the presence of two distinct optima. For completeness the niching PACO algorithms are tested on several small-

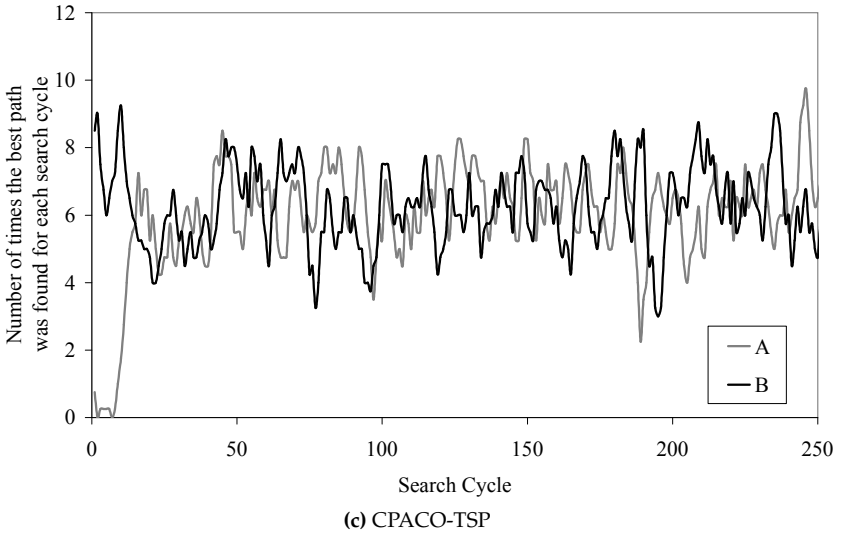
[3]The coordinates of this dataset are: {{0,0},{0,100},{50,50},{100,50},{150,100},{150,0}}
[4]Using the standard parameters as in [149]



(a) MMAS



(b) FSPACO-TSP



(c) CPACO-TSP

Figure 6.4: Crown problem: Occurrence of the two optimal paths over time for a single (although indicative) experiment run (A & B are the two distinct optimal paths). Each algorithm tested constructs 50 solutions per iteration. Different running times are reported to better illustrate specific algorithm behaviour, importantly though the convergence characteristics do not change past the maximum number of iterations reported on the graphs.

medium sized TSP from TSPLIB [128] to observe the effect of niching on the algorithm performance (with regard to locating the optimal solution). The results of the testing are presented in Tab. 6.1, the parameters used for this experimentation are included in Tab. 6.2. For comparison purposes the MMAS algorithm and PACO algorithm using the Quality replacement strategy from Sec. 4.3 are also included in the testing.

Problem \Algorithm	FSPACO-TSP	CPACO-TSP	MMAS	PACO
Berlin52	0.077	0.085	**0.027**	0.058
KroA100	0.120	0.209	**0.092**	**0.105**
eil101	0.101	0.132	0.119	**0.035**
ch130	0.115	0.243	0.091	**0.063**
ch150	0.110	0.300	**0.060**	**0.057**
gr202	0.122	0.196	**0.064**	0.084

Table 6.1: Results of testing FSPACO-TSP, CPACO-TSP, MMAS and PACO on standard TSP instances. Each algorithm was allowed $n \times 1000$ solution evaluations (where n is the number of cities) and were repeated 100 times with different random seeds. The figures reported are the percentage deviation from the optimum with the best performing algorithm(s) for each problem highlighted in bold (in some cases two algorithms tied). The statistical significance of the results are included in the Appendix as Tab. 10.1

FSPACO for the TSP

FSPACO manages convergence by de-rating similar solutions. This process ensures that optima become populated in a fitness-proportionate way, as was described in Sec. 5.3.2. In the case of the TSP most local optima will contain many similar components to the global optima. Thus if these tightly clustered solutions are all present in a population they will be subsequently de-rated, effectively deterring the algorithm from recombining their parts. Thus as is indicated in Tab. 6.1, the performance of FSPACO-TSP as compared to the PACO and MMAS algorithms is quite poor, and for these problems fitness sharing does not seem to offer any advantage. Worse still, the computational complexity of the FSPACO-TSP algorithm is higher than the PACO algorithm considered here which makes the extension even more inefficacious.

To elaborate, FSPACO-TSP adds a fair amount of computational complexity to the standard PACO algorithm. The most significant area of computational complexity increase is in the comparison of population members to each other to determine the niche count. To determine the niche count every population member must be measured against every other population member resulting in a non-optimised n^2 comparisons. While it is possible to reduce this complexity to $n\left(n-1\right)/2$, the resultant number of comparisons can still be seen as an extra computational burden.

FSPACO-TSP uses a generational replacement and thus it is possible that the entire population may change from one iteration to the next, thus a temporary pheromone matrix is used. Every iteration the pheromone matrix must be initialised and the elements in the matrix corresponding to solutions in the population must be adjusted. This will result in an increase in the number of pheromone value adjustments per iteration. Since the low number of adjustments required to the pheromone matrix was one of the original selling points of PACO this change is undesirable.

CPACO-TSP	
Parameter	**Value**
Number of ants / Population size	50
Initial Pheromone	1/Nearest Neighbour Solution Length
Pheromone Exponent (α)	1
Heuristic Exponent (β)	3
Solution selection greediness (q)	0.5
Crowding window size	0.1
FSPACO-TSP	
Parameter	**Value**
Number of ants / Population size	50
Initial Pheromone	1/Nearest Neighbour Solution Length
Pheromone Exponent (α)	1
Heuristic Exponent (β)	3
Solution selection greediness (q)	0.5
Niche radius	0.10
Sharing Power	1
MMAS	
Parameter	**Value**
Number of Ants	25
Initial Pheromone	1/Nearest Neighbour Solution Length
Pheromone Exponent (α)	1
Heuristic Exponent (β)	3
Maximum Pheromone (τ_{max})	1.0
Decay factor (ρ)	0.1
PACO	
Parameter	**Value**
Number of ants	10
Population size	2
Initial Pheromone	1/Nearest Neighbour Solution Length
Pheromone Exponent (α)	1
Heuristic Exponent (β)	5
Maximum Pheromone (τ_{max})	1.0
Solution selection greediness (q)	0.5
Elite factor	0.25

Table 6.2: Parameter settings used in testing of the TSP

In some cases extra computation may be worthwhile if it results in an increase in solution quality or algorithm robustness, however in the case of FSPACO the modifications did not increase either of these qualities and thus the increase in computational complexity cannot be justified.

CPACO for the TSP

Like FSPACO-TSP, CPACO-TSP did not perform well on the benchmark TSP instances reported in Tab. 6.1. This poor performance is most likely due to the structure of the search space not being compatible with the crowding replacement operation.

Any ACO algorithm applied to the TSP usually uses a strong heuristic bias ($\beta \approx 3$). While this is beneficial to standard ACO algorithms applied to this problem for reasons discussed earlier in Sec. 6.2.2, this heuristic does disrupt the operation of the crowding replacement operation of CPACO. It was observed that if CPACO-TSP is initialised with a random population that many of the population members present at this initialisation are also present at the end of run time. This is due to a subset of the population always 'winning' the closeness comparison operation and thus being the only members of the population to ever be given the chance to be updated. This effect is magnified by the presence of the heuristic; since the heuristic restricts the search space and thus any initial population members that have been initialised outside this restricted search space will have a very low chance of being updated. To highlight this, a table indicating the initial and final population after a single run of CPACO-TSP applied to the Berlin52 TSP is included in Tab. 6.3.

A possible way to address this negative effect is to reduce the crowding window size. This allows for the possibility of replacement of solutions from a larger distance away even if closer solutions exist in the population. This is because these close solutions are not always included in the subset of solutions that can be replaced by a new solution. The side-effect of this is that it allows for duplicate solutions to be entered into the population as is indicated in Tab. 6.4.

Even when tested with a small population size the solution duplication problem is still evident. The Eil101 TSP was tested using the same parameters as for Berlin52 however this time the population size was reduced to 5. Table 6.5 contains the path lengths of the initial population and the final population (after 10000 iterations).

CPACO-TSP adds a small amount of computational complexity to the standard PACO algorithm. Like FSPACO-TSP the most significant area of computational complexity increase is in the comparison of new solutions to existing population members to determine the closest match. This replacement routine is dependent on the crowding window size and has a worst case complexity if the crowding window is the same size as the population. At all times the total complexity of the crowding replacement operation per iteration is $c \cdot m$, where c is the crowding window and m is the number of new solutions (ants). A reduction in the crowding window size will reduce the overall replacement complexity, however the trade-off is an increase in the probability of duplicate solutions as was seen in Sec.6.2.3.

The number of modifications to the pheromone matrix may be higher than that of the standard

Initialisation	After 1000 Iterations	Initialisation	After 1000 Iterations
26538	**7615**	29937	29937
27095	27095	30019	30019
27711	27711	30066	30066
27828	27828	30386	30386
27955	27955	30507	30507
27973	27973	30512	30512
28065	28065	30551	30551
28232	28232	30645	30645
28325	28325	30708	30708
28494	28494	30745	30745
28685	28685	30745	30745
28745	28745	30780	30780
28951	28951	30801	30801
29047	29047	30845	30845
29068	29068	30890	30890
29211	29211	31002	31002
29419	29419	31073	31073
29446	29446	31131	31131
29475	29475	31501	31501
29638	29638	31708	31708
29703	29703	31777	31777
29717	29717	31936	31936
29755	29755	32177	32177
29855	29855	32235	32235
29930	29930	33288	33288

Table 6.3: Path lengths of solutions contained in the population of CPACO-TSP after initialisation (ordered by length) and the final population after 1000 iterations (constructing 50 solutions per iteration) for Berlin52. In this example CPACO-TSP has a large crowding window size. While it is possible for different solutions to have the same path length, those solutions reported with the same path length are in fact the same solution. Note that only one solution in the final population is different from the initial population.

Initialisation	After 1000 Iterations	Initialisation	After 1000 Iterations
26538	**8066**	29937	**8066**
27095	**8066**	30019	**8066**
27711	**8066**	30066	30066
27828	**8066**	30386	30386
27955	**8066**	30507	30507
27973	**8066**	30512	30551
28065	**8066**	30551	**8066**
28232	28232	30645	**8066**
28325	**8066**	30708	30708
28494	**8066**	30745	30745
28685	**8066**	30745	30745
28745	**8066**	30780	30780
28951	**8066**	30801	30801
29047	29047	30845	30845
29068	**8066**	30890	**8066**
29211	**8066**	31002	31002
29419	29419	31073	31073
29446	29446	31131	31131
29475	29475	31501	31501
29638	29638	31708	31708
29703	29703	31777	31777
29717	29717	31936	31936
29755	**8066**	32177	32177
29855	29855	32235	32235
29930	29930	33288	33288

Table 6.4: Path length of solutions contained in the population of CPACO-TSP after initialisation (ordered by length) and the final population after 1000 iterations (constructing 50 solutions per iteration) for Berlin52. In this example CPACO-TSP has a small crowding window size. While it is possible for different solutions to have the same path length, those solutions reported with the same path length are in fact the same solution. Note the duplication of one particular solution throughout the final population. While many solutions from the initial population have been replaced they have all been replaced with the same solution.

Initialisation	After 10000 Iterations
3153.0	**723.0**
3399.0	**723.0**
3559.0	3559.0
3586.0	**723.0**
3696.0	3696.0

Table 6.5: Path length of solutions contained in the population of CPACO-TSP after initialisation and the final population after 10000 iterations (constructing 5 solutions per iteration) for Eil101. While it is possible for different solutions to have the same path length, those solutions reported with the same path length are in fact the same solution. Note the duplication of one particular solution throughout the final population.

PACO[5] since it is likely for there to be higher solution turnover in the population, however this was not specifically tested given the poor results obtained.

6.2.4 Summary

It was mentioned at the beginning of this section that the performance of the niching PACO algorithms on the single objective TSP was unlikely to be good. For the benchmark problems tested from TSPLIB the results tend to agree with the initial postulate that the use of niching for this problem does not seem to be justified. Not only is the quality of the result worse, but the algorithms add additional computational complexity.

Given these findings though, it must be said that while the standard TSP instances of TSPLIB did not benefit from niching, the special case Crown problem did. While this problem was entirely contrived it did demonstrate the niching algorithms' strength in the simultaneous location and maintenance of multiple optima on a combinatorial problem domain. For the overwhelming majority of single objective TSP in TSPLIB[128] this property is probably not required, however there do exist other variations to the basic TSP that require the location and maintenance of diverse solutions, one such variation being the multiple objective TSP investigated in Sec. 7.4.

6.3 Niching PACO for Continuous Function Optimisation

CFO is perhaps the most highly tested problem by niching evolutionary algorithms. Reasons for this were provided in Sec. 6.1. An analysis of the Niching PACO algorithms applied to the CFO will firstly allow a determination as to whether niching has been implemented successfully. Secondly it will allow a determination as to how useful this adaptation is as compared to other existing niching and non-niching techniques. Implementation details of the Niching PACO algorithms for CFO are firstly outlined. A brief sensitivity analysis on several benchmark CFO problems of small dimensionality is performed to determine the effect of certain introduced parameters. Following this sensitivity analysis the algorithms are run in set configurations on more benchmark CFO problems with varied qualities and difficulty to determine their effectiveness at locating multiple diverse optima and locating a single global optimum. The algorithms are analysed using a combination of qualitative and quantitative analysis techniques.

6.3.1 Algorithm Details

FSPACO-CFO and CPACO-CFO are similar to the PACO algorithm for Continuous Domains (ACOCD) presented in Sec. 4.5 but for a few differences which are discussed in this section.

[5] Pheromone modification only takes place when a solution is replaced which means that if there are no replacements the pheromone matrix is left untouched.

Solution Representation

In FSPACO-CFO and CPACO-CFO each individual solution is represented as a vector (length=n, where n is the number of dimensions) of real values restricted to be within the bounds of the search space (feasible search region). If s is a solution then s_i is the i^{th} component of that solution. Each solution (once evaluated) also contains a solution quality value (s_{score}) which can be used to determine the fitness of the solution, if required.

Solution Construction (Ants Generation and Activity)

Like ACOCD a Gaussian sampling technique is used to generate a new position in the n-dimensional search space. However, unlike ACOCD, different solutions from the population (population size = k) can be used to determine the mean for each dimension of the problem. The solution construction procedure is described in Alg. 6 and visually in Fig. 6.5.

Algorithm 6 Solution construction procedure for CFO

1: Calculate selection probabilities for population according to Alg.7, Alg.8 or Alg.9.
2: Create new empty solution s^{new}
3: **for** $i = 1$ to n **do**
4: Probabilistically select a solution (s) from the population using a biased roulette wheel selection strategy with replacement.
5: $\mu = s_i$
6: $r = |n_{max} - n_{min}|$ ▷ Calculate dimension range
7: $c = (\sin(\pi/2 \times \text{remaining evaluations}/\text{maximum evaluations}))^2$ ▷ Calculate convergence factor
8: $\sigma = r \times c/6$ ▷ Calculate standard deviation
9: **repeat**
10: $s_i^{new} =$ Gaussian weighted random value according to Equ. 6.3.1
11: **until** $n_{min} <= s_i^{new} <= n_{max}$ ▷ New coordinate must be within bounds of problem otherwise resample
12: **end for**

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{6.3.1}$$

There are three variants of the solution selection scheme: Rank, Quality and Unity. These schemes relate to the method of calculation of the selection probabilities of the solutions in the population. With rank-based (Alg.7) each solution is ordered based on its fitness value and assigned a rank from 1 to k, where k is the size of the population. The inverse of this rank is then used as the basis for selection, thus a larger rank (worse solution) has a lower probability of selection. Quality-based (Alg.8) firstly determines the minimum and maximum fitness values, then it normalises the fitness of every solution between these bounds so that the best solution will have a fitness value of 1 and the worst solution a fitness value of 0. The reason for this normalisation is that it makes the quality-based selection strategy impervious to the scale of the fitness function. Unity-based (Alg.9) assigns an even probability of selection to every solution. In essence this rank procedure is unnecessary as the entire solution selection procedure could simply be replaced with a uniform random selection strategy with replacement, however

this way it can be applied in the same manner as the other solution selection schemes.

Algorithm 7 Rank based solution ranking procedure for CFO. Determines probability (p) of selecting Solution i based on the rank of this solution relative to the rest of the population.

```
Sort population from best to worst.
p_total = 0
for i = 1 to k do                          ▷ Calculate probabilities (unscaled & non-normalised)
    p(i) = 1/i
    p_total = p_total + p(i)
    for i = 1 to k do                      ▷ Normalise and power scale probabilities
        p(i) = (p(i) / p_total)^α
    end for
end for
```

Algorithm 8 Quality based solution ranking procedure for CFO. Determines probability (p) of selecting Solution i based on the normalised quality of this solution.

```
for i = 1 to k do                          ▷ Find bounds of solution scores
    if s^i_score > max then
        max = s^i_score
    end if
    if s^i_score < min then
        min = s^i_score
    end if
end for
for i = 1 to k do                          ▷ Calculate probabilities (unscaled & non-normalised)
    if Minimisation Problem then
        p(i) = 1 − (s^i_score − min) / (max − min)
    end if
    if Maximisation Problem then
        p(i) = (s^i_score − min) / (max − min)
    end if
    p_total = p_total + p(i)
end for
for i = 1 to k do                          ▷ Normalise and power scale probabilities
    p(i) = (p(i) / p_total)^α
end for
```

Algorithm 9 Unity based solution ranking procedure for CFO. Assigns every Solution i an even probability (p) of selection.

```
for i = 1 to k do                          ▷ Calculate probabilities (normalised)
    p(i) = 1/k
end for
```

Solution Storage and Maintenance

For FSPACO-CFO a generational replacement method is implemented where at each iteration the population is replaced with the newly created solutions. For CPACO-CFO the crowding comparison operation explained in Sec.5.4.3 is used to manage the insertion of new solutions into the population. For both algorithms an initial population is created using a uniform random sampling of the search space.

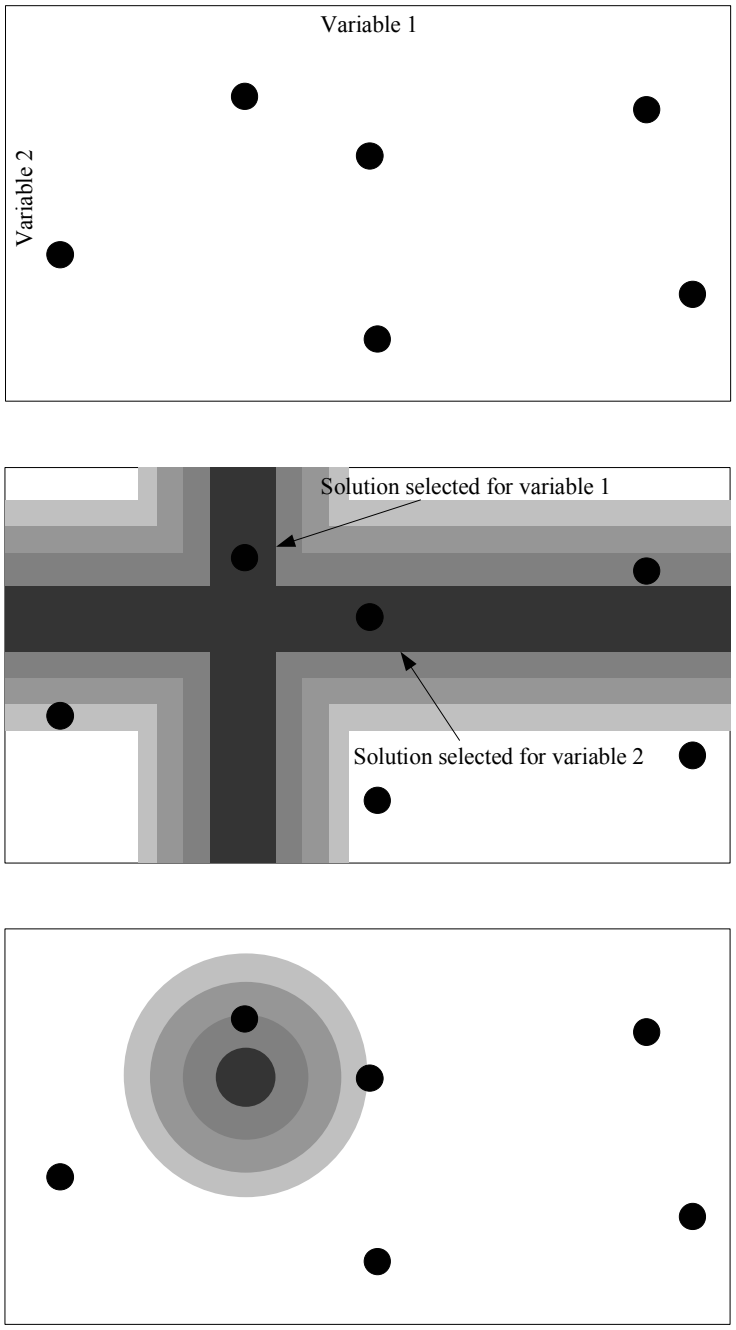


Figure 6.5: An illustration of the new solution creation procedure for FSPACO-CFO and CPACO-CFO.

Reasons for differences with ACOCD

A fundamental characteristic of ACO algorithms is probabilistic stepwise construction based on prior utility of individual solutions. In ACOCD each ant performs stepwise construction, however this stepwise construction is based mostly around a single solution from the population. ACOCD performs a Gaussian sampling around a single existing solution with a unique standard deviation for each dimension. This means that the new solution is most likely to be in the neighbourhood of the original solution. This choice was made since the selection of a different mean for each dimension did not provide enough convergence pressure (K. Socha, personal communication, 3rd May, 2007).

In the case of FSPACO-CFO and CPACO-CFO it is not desirable to converge to a single optima so the ACOCD construction method was replaced by the afore-mentioned solution construction method (Alg. 6). The reduction in convergence behaviour is addressed by reducing the standard deviation of samples over time which is demonstrated visually in Fig.6.6. More-so since FSPACO-CFO and CPACO-CFO are treated as global-search techniques their aim is to produce a population of distributed solutions in good neighbourhoods of the search space which can subsequently be improved by a local search technique.

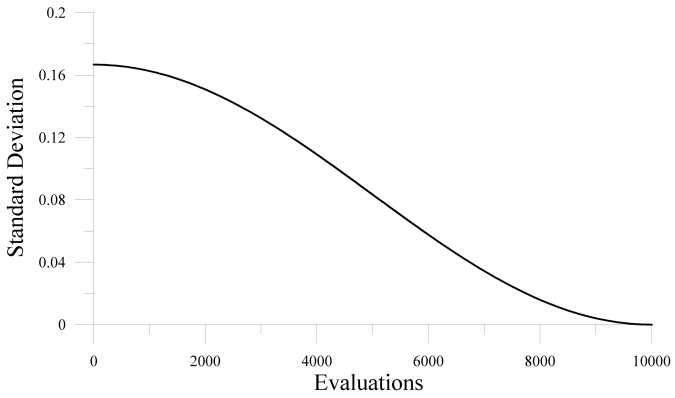


Figure 6.6: Plot demonstrating the variation of the standard deviation over the number of function evaluations made. Although in this figure the standard deviation is plotted for 10,000 evaluations the slope of the graph is always the same relative to any arbitrary number of evaluations.

6.3.2 Test Problems Used

A suite of commonly used CFO problems have been selected as benchmark test problems. These problems have been used in past studies to analyse the behaviour of niching algorithms [72, 105] and are useful problems for preliminary analysis since they can be easily visualised in two dimensions while still being moderately difficult to solve. Each problem used has been selected due to properties such as modality, uniformity, quantity of noise and deceptiveness. Some of the problems are strictly defined in 2 dimensions, while others are able to be scaled to an arbitrary number of dimensions. Although the majority of the functions contain numerous optima, the target optima (those optima we wish the algorithms to locate) varies depending on the problem definition. In this section each problem is described in terms of these qualities.

Of the eight problems chosen, half require the identification of multiple optima and the other half require the identification of only a single global optimum. Of the eight problems, four unique problems were chosen for each of the sensitivity and quantitative analyses. The choice of problem was fairly arbitrary, however it was ensured that the balance was kept at two single optimum and two multiple optimum problems per analysis.

3-pot holes

The 3-pot holes function[122] is a minimisation problem containing three distinct optima (regardless of the number of dimensions) with large basins of attraction leading towards these optima. This problem can be defined in any number of dimensions and in each dimension it is defined over the bounds $[-10, 10]$. As the optima are not spaced in-line in any individual dimension this problem may prove difficult for algorithms which exhibit axis searching behaviour (this will be discussed in later sections). When defined in any number of dimensions the objective is to locate all three optima.

$$f(x) = \sqrt{\sqrt{\sum_{i=1}^{n}\left((x(i)+8)^2+0.1\right)}+\sqrt{\sum_{i=1}^{n}\left((x(i)+2)^2+0.2\right)}+\sqrt{\sum_{i=1}^{n}(x(i)-3)^2}} \quad (6.3.2)$$

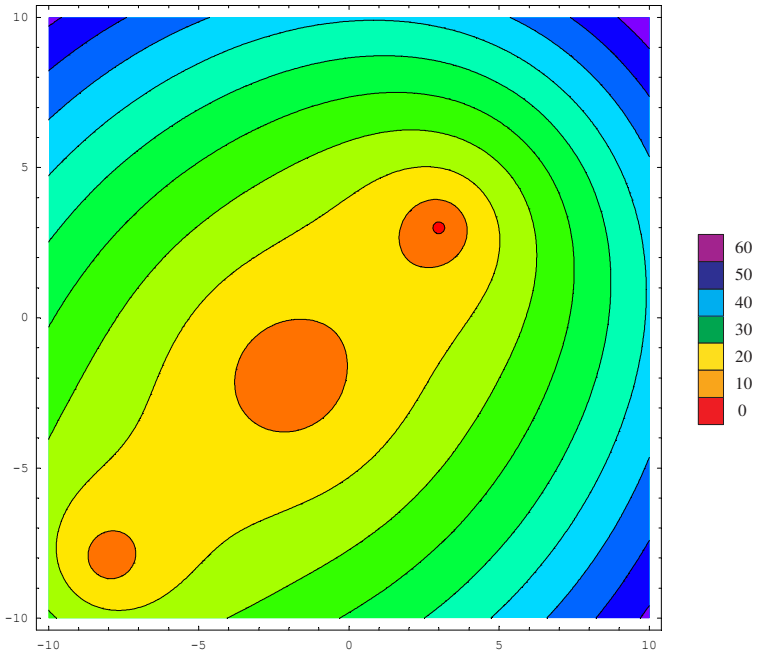


Figure 6.7: Three Pot Holes function defined in two dimensions.

Ackley's

Ackley's function[1], also known as Ackley's Path Function, is a minimisation problem that contains a global optimum at the origin. The function also contains other false optima ($3^n - 1$ in total, where n is the number of dimensions) located at the function bounds. The function contains a minor amount of uniform noise. It can be defined over any number of dimensions and in each dimension is defined over the bounds $[-32.768, 32.768]$. The objective is to locate the single global optima at the origin.

$$\begin{aligned} f(x) &= -a \times \exp\left(-b \times \sqrt{1/n \times \sum_{i=1}^{n}\left(x(i)^2\right)}\right) \\ &\quad - \exp\left(1/n \times \sum_{i=1}^{n}\left(\cos\left(c \times x(i)\right)\right)\right) + a + \exp 1 \\ &\quad \text{Where: } a = 20, b = 0.2, c = 2\pi; \end{aligned} \tag{6.3.3}$$

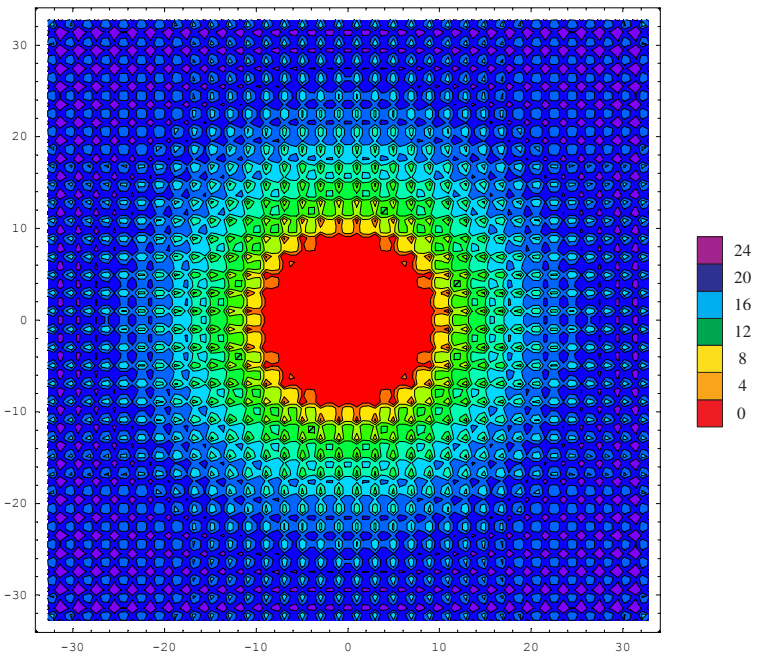


Figure 6.8: Ackley's function defined in two dimensions.

Branin's R-Cos

Branin's R-Cos function[12] is defined in two dimensions within the bounds: $-5 \leq x \leq 10$, $0 \leq y \leq 15$. It is a minimisation problem which could be best described as a valley with three optima located on the valley floor. Its difficulty lies in that while the valley walls are steep the valley floor is flat. These conditions combined may lead an algorithm to quickly locate and converge on one (but not necessarily the best) optima on the valley floor. This problem is defined in two dimensions and like the 3-pot holes problem contains optima that are not in-line in individual dimensions. When searching this problem the objective is to locate all three optima.

$$f(x,y) = a\left(y - b \times x^2 + c \times x - d\right)^2 + e\left(1 - f\right) \times cos\left(x\right) + e \tag{6.3.4}$$
$$\text{Where: } a = 1, b = 5.1/\left(4\pi^2\right), c = 5/\pi, d = 6, e = 10, f = 1/\left(8\pi\right)$$

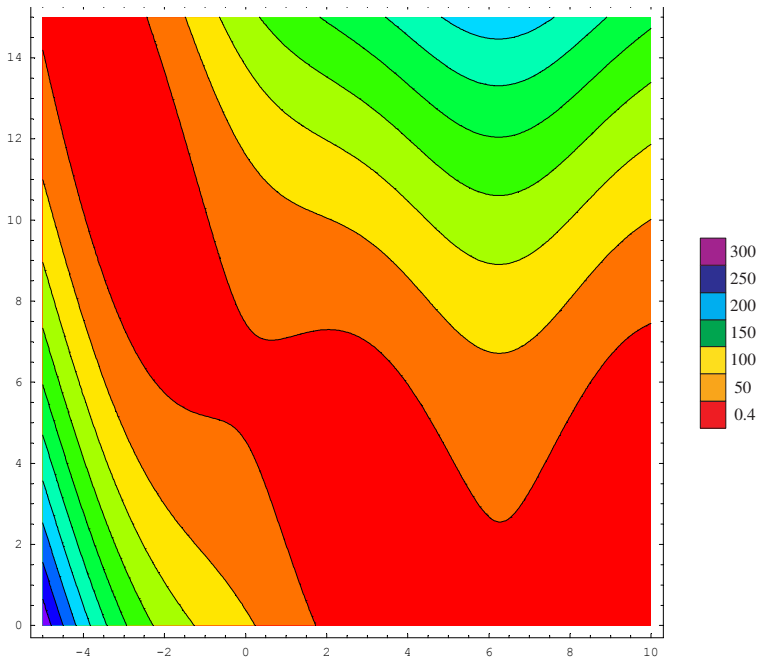


Figure 6.9: Branin's R-Cos function defined in two dimensions.

Himmelblau's

Himmelblau's function[34] contains 4 distinct optima that, like Branin's R-Cos function, are located at the bottom of a valley. These optima are not regularly spaced (like 3 Pot-Holes and Branin's R-Cos) and the lack of gradient information may prove difficult for an algorithm to successfully locate all optima. It is defined over the bounds: $-6 \leq x \leq 6, -6 \leq y \leq 6$. This problem is defined in two dimensions and when searching this problem the objective is to locate all four optima.

$$f(x,y) = \frac{\left(\left(x^2 + y - 11.0\right)^2 + \left(x + y^2 - 7.0\right)^2\right)}{2186} \tag{6.3.5}$$

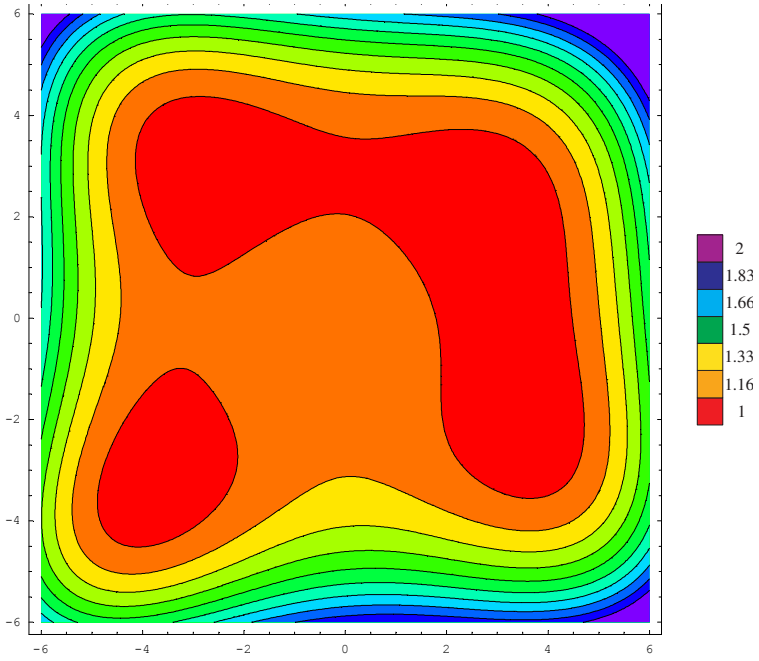


Figure 6.10: Himmelblau's function defined in two dimensions.

Rastrigin's

Rastrigin's function was first proposed in [153] as a difficult, highly multi-modal test function. It is a minimisation problem and the function has a general bowl shape sloping toward the global optimum but contains numerous, though evenly spaced, optima overlaid onto this bowl. The function is defined in the bounds $[-5.12, 5.12]$ per dimension and can be defined in any number of dimensions. The objective is to locate the single global optimum.

$$f(x) = 10 \times n + \sum_{i=1}^{n} \left(x(i)^2 - 10 \times \cos(2\pi \times x(i)) \right) \tag{6.3.6}$$

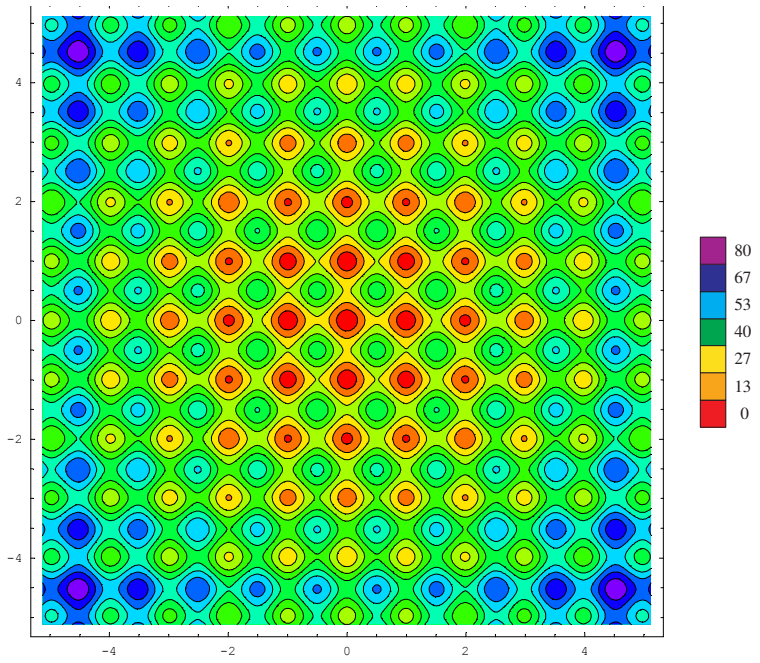


Figure 6.11: Rastrigin's function defined in two dimensions.

Schaffer's

Schaffer's function[136] is a two dimensional 'trapping' function. The function contains a single global optimum at the origin that is surrounded by a series of rings. The problem is a maximisation problem, with the value of the global optimum being one. The rings that surround the optimum alternate between large values close to one and small values close to zero and tend to a value of 0.5 at the limit. The effect is that while it may be easy to traverse through one of these rings at the edges of the function the difference between the peak and trough becomes larger as we approach the optimum. This gives rise to the trapping effect since an algorithm can easily settle at a peak of a ring and simply navigate around the ring rather than moving towards the origin[33]. The objective of this function is to locate the single global optimum.

$$f(x,y) = 0.5 + \frac{\sin^2\left(\sqrt{x^2+y^2}\right) - 0.5}{\left(1 + 0.001\left(x^2+y^2\right)\right)} \tag{6.3.7}$$

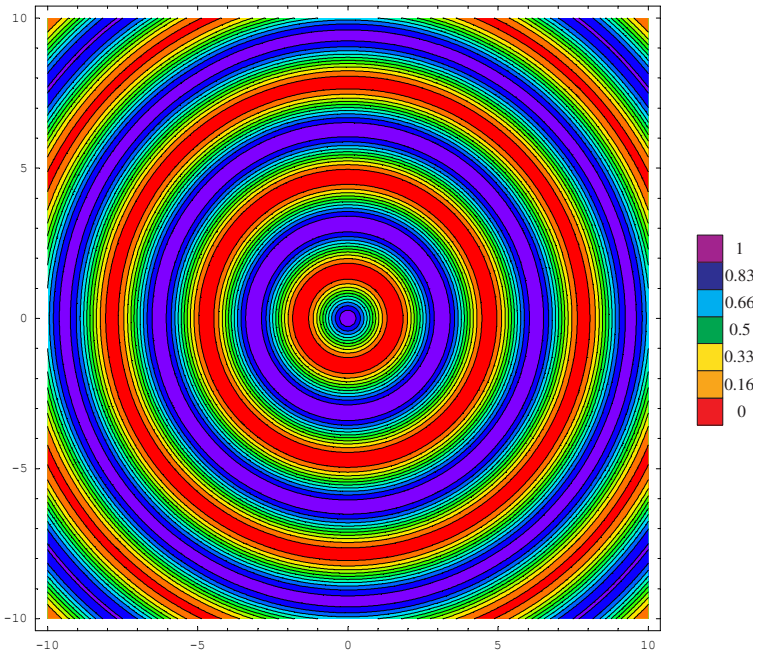


Figure 6.12: Schaffer's function defined in two dimensions.

Schwefel's

Schwefel's function [141] is a deceptive, multi-modal function. Given that the global optimum is a significant distance from the next best optimum there is a chance that a search algorithm will converge in the wrong direction thereby missing the global optimum. It can be defined in any number of dimensions and is bounded between $[-500, 500]$. As the number of dimensions increases so does the number of optima, however when searching this problem the goal is to locate just the global optimum.

$$f(x) = \sum_{i=1}^{n} \left(-x(i) \cdot \sin\left(\sqrt{|x(i)|} \right) \right) \tag{6.3.8}$$

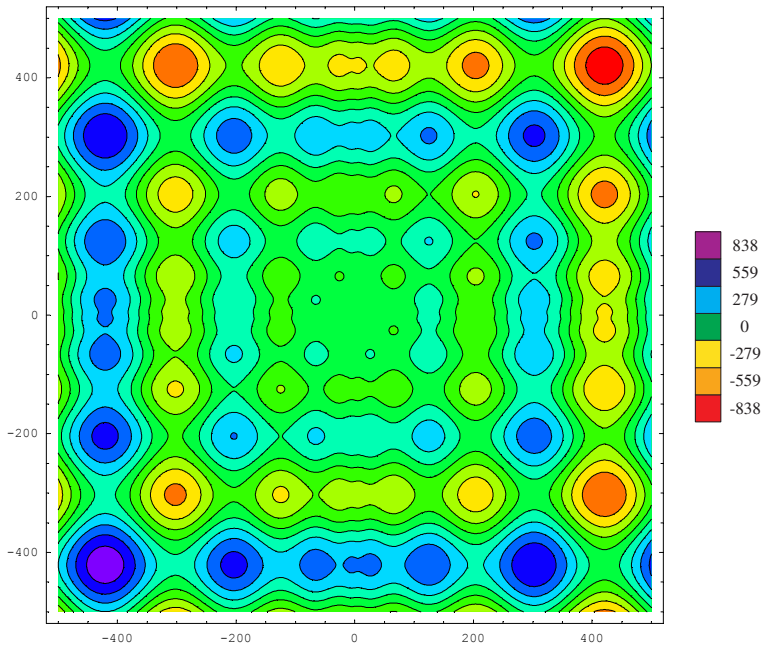


Figure 6.13: Schwefel's function defined in two dimensions.

Shekel's foxholes

Like the name suggests, Shekel's foxholes[39] are a series of evenly spaced minima (holes) of varying depth, totalling 25 in two dimensions. This function is difficult due to the small radius, number and closeness of optima. It is defined over the bounds: $-65.54 \leq x \leq 65.54$, $-65.54 \leq y \leq 65.54$. The function is defined in two dimensions, and the goal is to locate all 25 optima.

$$\frac{1}{f(x,y)} = \frac{1}{500} + \sum_{j=1}^{25} \left(\frac{1}{j + \sum_{i=1}^{2} \left((x - a_{ij})^6 + (y - a_{ij})^6 \right)} \right) \tag{6.3.9}$$

$$\text{Where: } a_{ij} = \begin{bmatrix} -32 & -16 & 0 & 16 & 32 & \cdots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & \cdots & 32 & 32 & 32 \end{bmatrix}$$

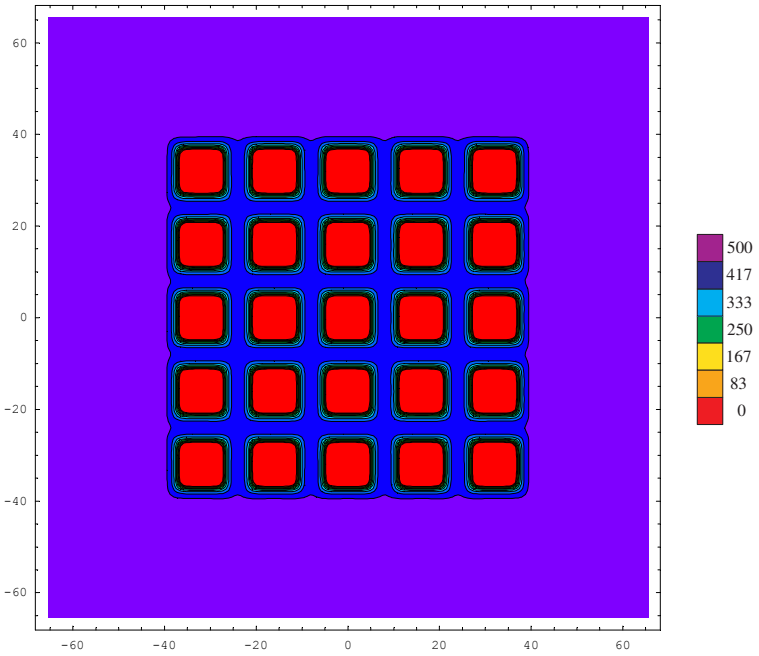


Figure 6.14: Shekel's foxholes function defined in two dimensions.

6.3.3 Performance Metrics

Modified Max-Peak Ratio

The max-peak ratio is a ratio of the number of optima present in the final population (within a small radius of the actual optima) versus the number of actual optima. This measure is an indicator of an algorithms ability to obtain solutions across multiple optima. The max-peak ratio is a binary measurement and optima are either said to be located or not located. The reasoning behind this method is that if a hill-climbing algorithm was seeded with a solution within the radius of the optima it would be able to navigate easily to the actual optima.

Rather than assign a radius and measure whether a solution is contained within the radius; a similar measurement would be to locate the closest solutions to each known optima and measure the average Euclidean distance between the solutions and the optima. This modification to the max-peak ratio means that:

- It removes the requirement for an arbitrary radius.
- An algorithm which does not come anywhere close to the optima is no longer equivalent to one that finds solutions quite close but not within the pre-defined radius.
- Two algorithms are no longer said to be equivalent when one algorithm does a much better job at finding solutions at the very extremes of the optima.

The modified max-peak ratio is included as Alg.10.

Algorithm 10 Calculation of modified max-peak ratio

1: totalDistance $= 0$
2: **for** $i = 1$ to numberOfOptima **do**
3: $\quad s \leftarrow$ Closest solution to optima$_i$
4: $\quad$ distance $= 0$
5: $\quad$ **for** $j = 1$ to numberOfDimensions **do**
6: $\quad\quad$ distance $=$ distance $+ \left(\frac{|s_j - optima_{ij}|}{|max_j - min_j|} \right)^2$
7: $\quad$ **end for**
8: $\quad$ totalDistance $=$ totalDistance $+ \sqrt{\text{distance}} / \sqrt{\text{numberOfDimensions}}$
9: **end for**
10: ModifiedMaxPeak $=$ totalDistance$/$numberOfOptima

Ideally an algorithm would obtain a modified max-peak ratio of zero. The ratio is independent of the number of optima and dimensions so it should be able to provide a good measure to compare algorithms across a single problem defined over a different number of dimensions. The ratio has been simplified in that it does not introduce optima quality proportionate information. This means that all optima are treated evenly regardless of their quality, the reasoning behind this choice is that for the benchmarks chosen the optima are for the most part fairly comparable with regards to quality anyway.

6.3.4 Results and Discussion: Sensitivity Analysis

This section outlines a sensitivity analysis of the three CPACO-CFO algorithm variants and the FSPACO-CFO algorithm using different parameter configurations with the aim of determining the effect of these parameters on algorithm performance, in terms of the modified max-peak ratio. Four of the test problems described in Sec.6.3.2 are used for this analysis. The results of this analysis are used to apply the Niching PACO algorithms with set parameter configuration to four more, previously untested, problems and then to two of these problems over a larger number of dimensions.

A qualitative analysis of the sampling behaviour of the Niching PACO algorithms is also provided to give an insight into their niche formation and maintenance characteristics. To perform this analysis several test runs of two of the functions are made with all solutions created in those runs reported and a qualitative assessment of the niching quality made.

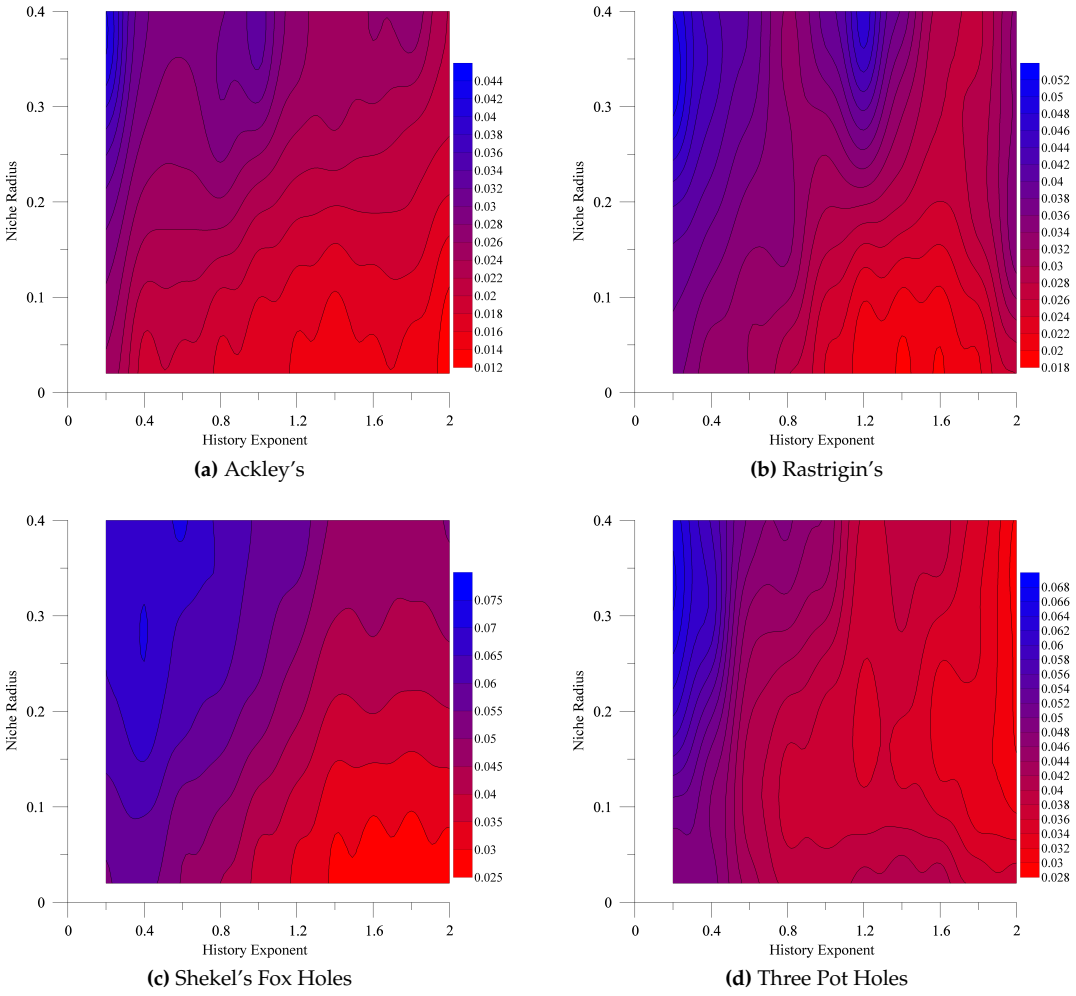
The problems chosen for the sensitivity analysis are: Ackley's, Rastrigin's, Shekel's Fox Holes and Three Pot Holes. All of these functions are defined in two dimensions, with Ackley's and Rastrigin's functions containing a single desired optima (but still containing other optima) while Shekel's Fox Holes and Three Pot Holes each contain multiple desired optima. The results reported are the average modified max-peak-ratio from 50 repeats of each specific algorithm configuration. The parameters that are set constant for this analysis are reported in Tab.6.6. No statistical significance tests are used in this section as this is a simple introductory evaluation of the effects of various parameters. Robust conclusions about the algorithms performance relative to each other are left for the following sections.

CPACO-CFO	
Parameter	**Value**
Number of ants / Population size	50
History Exponent (only used with CPACO-Quality & CPACORank)	$0.1 \rightarrow 2.0$
Crowding window size	$0.1 \rightarrow 1.0$
FSPACO-CFO	
Parameter	**Value**
Number of ants / Population size	50
History Exponent	$0.2 \rightarrow 2.0$
Niche radius	$0.02 \rightarrow 0.4$

Table 6.6: Parameter settings used for sensitivity analysis

To produce the surfaces in Fig. 6.15 for FSPACO-CFO, the history exponent (α) was varied in the range $[0.2, 2.0]$ in increments of 0.2 and the niche radius varied in the range $[0.02, 0.4]$ in increments of 0.02. In Fig. 6.16 and Fig. 6.17, CPACORank-CFO and CPACOQuality-CFO respectively, the history exponent (α) was varied in the range $[0.1, 2.0]$ in increments of 0.1 and the crowding window size varied in the range $[0.1, 1.0]$ in increments of 0.1. For Fig. 6.18 the history exponent is not required thus only the crowding window size was varied in the range $[0.1, 1.0]$ in increments of 0.1. Each of these surfaces have been smoothed using a Kriging interpolation method[42] the parameters of which are included in Appendix 10.

For FSPACO-CFO the algorithm tends to achieve a better result as the niche radius decreases



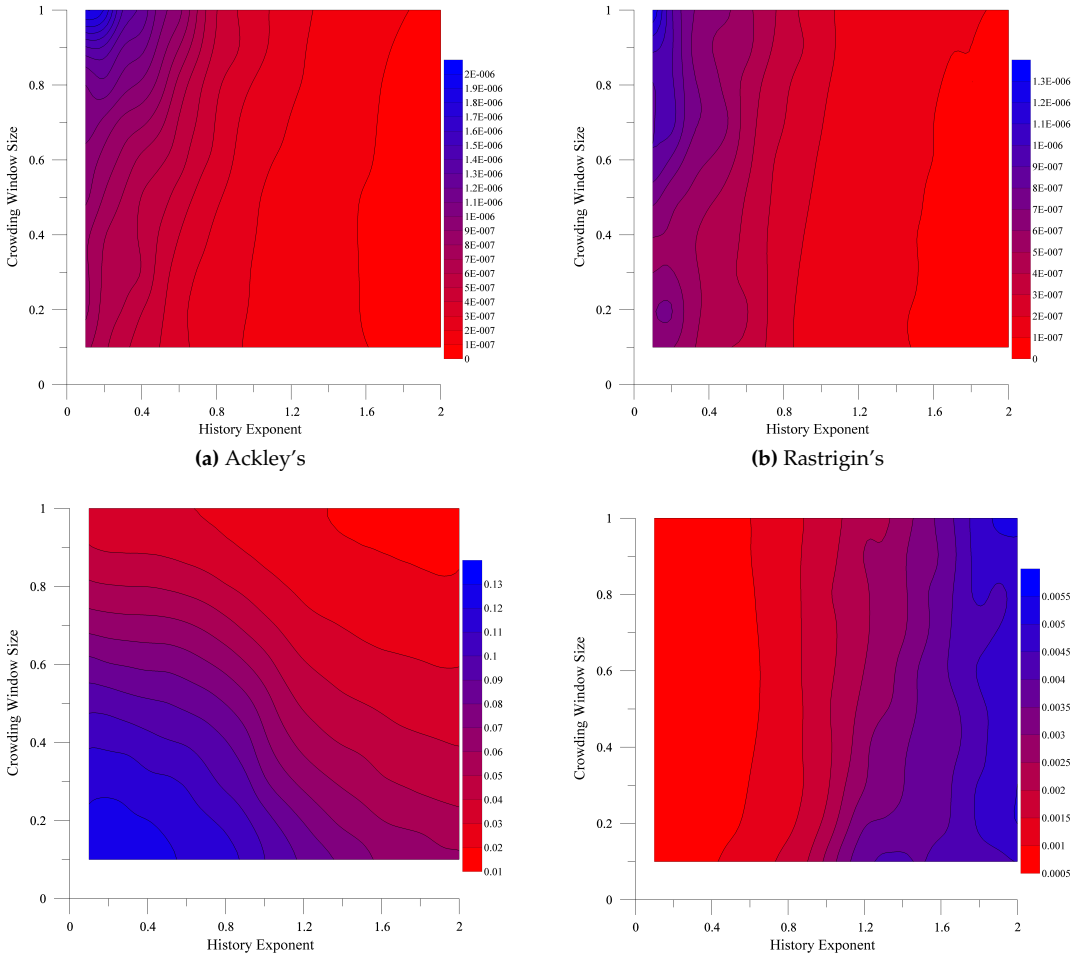
(a) Ackley's

(b) Rastrigin's

(c) Shekel's Fox Holes

(d) Three Pot Holes

Figure 6.15: Plots indicating the effect on the max-peak-like performance metric when the niche radius and history exponent of FSPACO-CFO are varied.



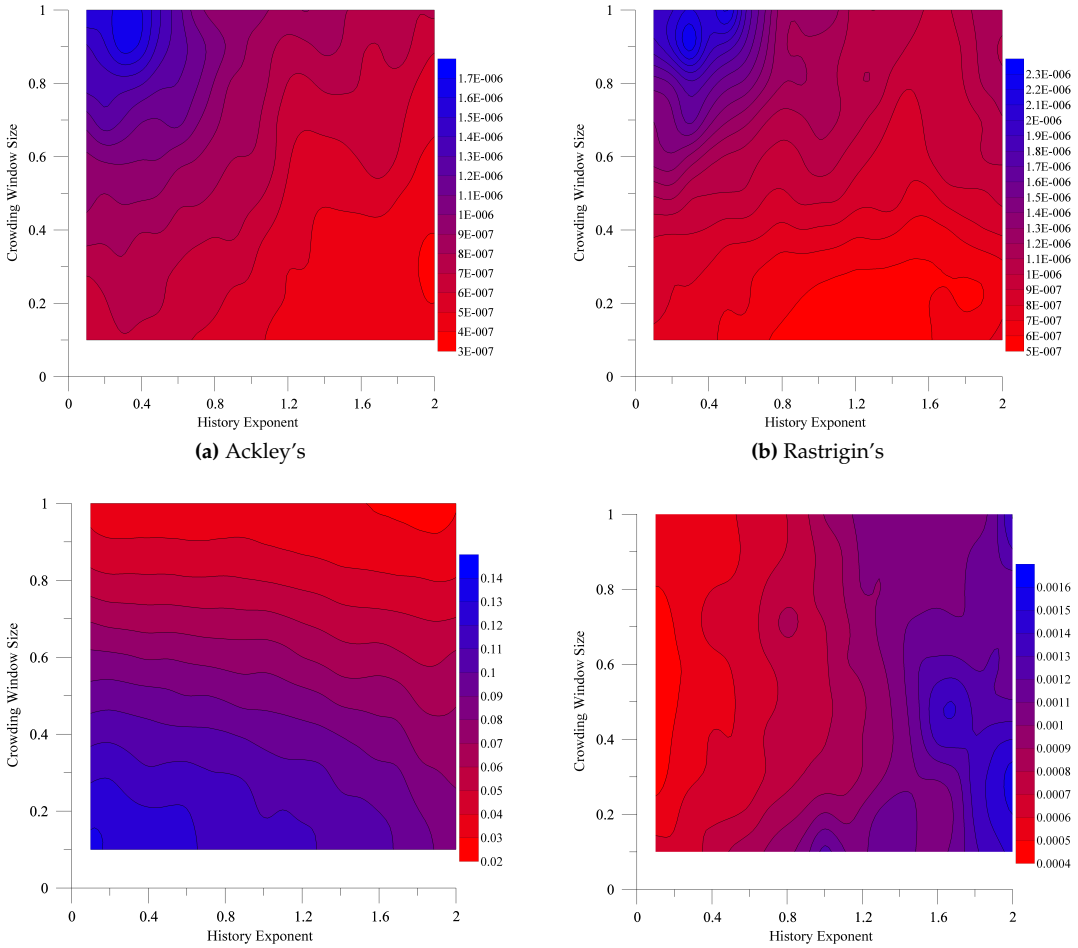
(a) Ackley's

(b) Rastrigin's

(c) Shekel's Fox Holes

(d) Three Pot Holes

Figure 6.16: Plots indicating the effect on the max-peak-like performance metric when the crowding window size and history exponent of CPACORank-CFO are varied.



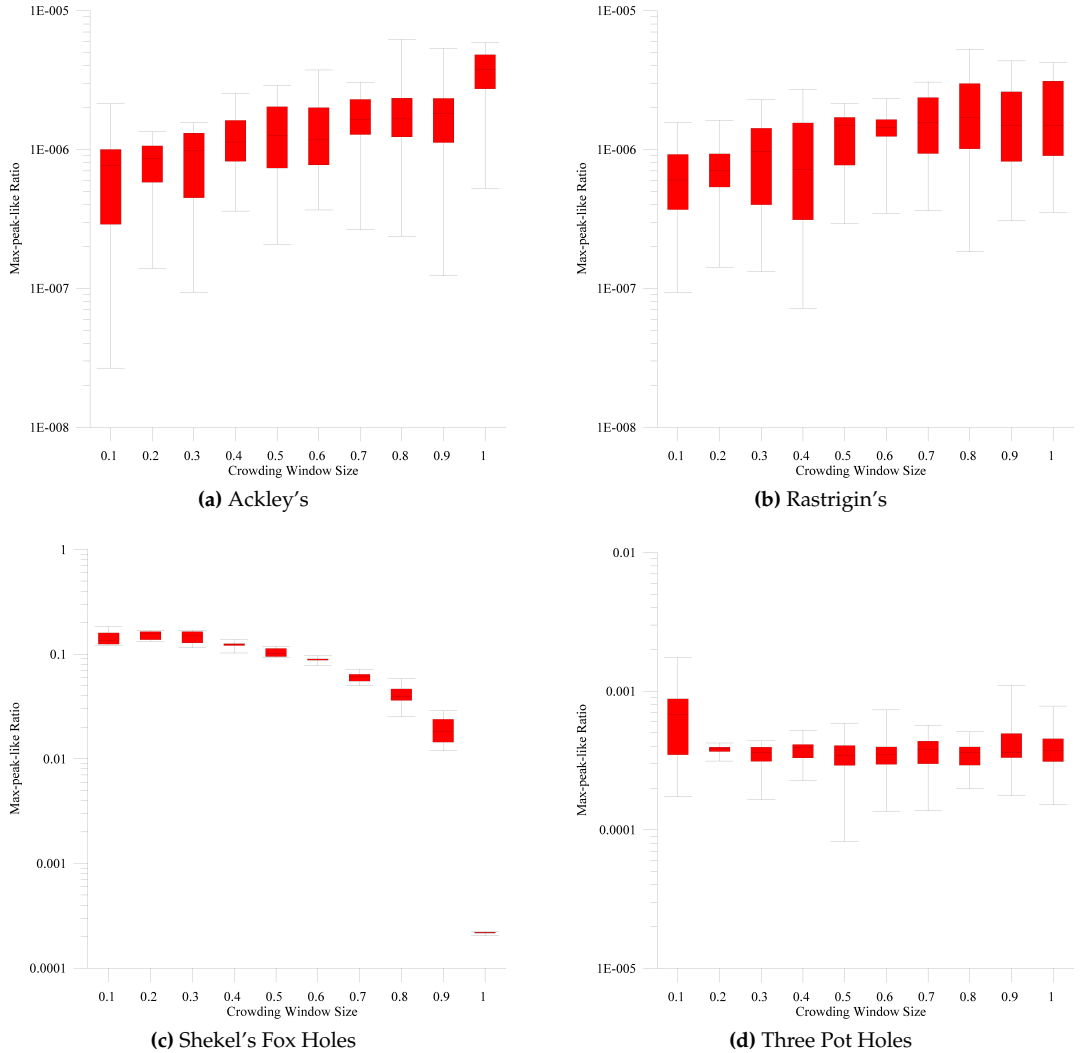
(a) Ackley's

(b) Rastrigin's

(c) Shekel's Fox Holes

(d) Three Pot Holes

Figure 6.17: Plots indicating the effect on the max-peak-like performance metric when the crowding window size and history exponent of CPACOQuality-CFO are varied.



(a) Ackley's

(b) Rastrigin's

(c) Shekel's Fox Holes

(d) Three Pot Holes

Figure 6.18: Plots indicating the effect on the max-peak-like performance metric when the crowding window size of CPACOUnity-CFO is varied.

and the history exponent is increased. This said the actual level of performance was not spectacular compared to the CPACO-CFO variants however this will become clearer in the next section of experimentation. As far as selecting parameters for this algorithm is concerned a niche radius equal to 0.02 and a history exponent equal to 1.4 seem to be reasonable parameter settings. From a theoretical standpoint these parameters seem to make sense since a larger niche radius increases the likelihood of overlap between niches which is undesirable according to [34].

For the CPACO-CFO variants the results of the sensitivity analysis point to the development of very distinct algorithm behaviours that are dependent on the nature of the problem being addressed. For Shekel's Fox Holes decreasing the crowding window size for all algorithms decreased the modified max-peak ratio performance. This is an intuitive result as this problem is characterised by several tightly spaced optima, and therefore reducing the crowding window size increases the possibility of replacement error quite dramatically. To explain consider an example.

Figure 6.19 shows minimisation problem with a single dimension multi-modal landscape and three coloured dots representing two existing population members (red and blue) and a newly created solution (green). Let us say that a new solution has been created close to an optimum where there is already another solution in the population that is close to this particular optimum (red dot). If a crowding window size of 1.0 is used then the red solution will be used to compare against the new (green) solution, and the end result will be that the red solution is replaced by the green solution. However if the crowding window size is decreased it will become more likely that the red solution may not be selected to be compared against. Thus another solution that is located close to another optimum may be selected for comparison instead e.g., the blue solution. Since the green solution is of better fitness than the blue solution, the blue solution will be replaced and the algorithm will have 'stolen' a solution from another optimum and in this case it means that the knowledge of this other optimum is lost.

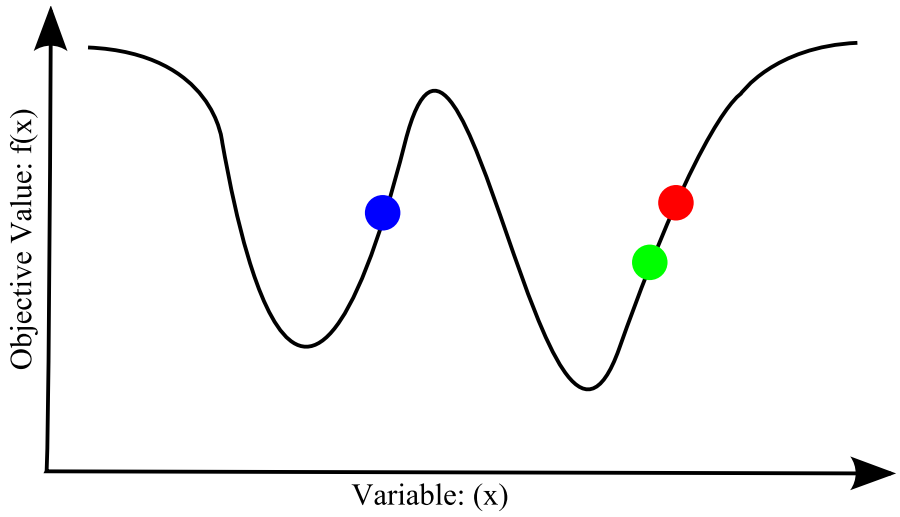


Figure 6.19: Replacement error example. The red and blue dots represent existing population members, the green dot is a newly created solution (see text for explanation).

This replacement error can be visualised by displaying all solutions created in a single exper-

iment run using different crowding window sizes as in Tab. 6.7. This figure clearly highlights that as the crowding window size decreases the number of optima that the algorithm identifies and maintains decreases. Interestingly as well the history exponent can accelerate this effect as it provides some positive feedback to the selection process. If a replacement error occurs it may also mean that some optima begin to contain many solutions and thus also have a greater chance of being used in future solution construction. This may increase the number of replacement errors as the number of solutions being created around a particular optima will increase. It may not be clear in this figure that in all cases where the crowding window size was set to 1.0 all optima were located and maintained. However, the distribution of the searching was influenced by the history exponent, this meant that when the history exponent was larger the better optima were selected more frequently thus directing the search pattern to be located mostly around these better optima[6]. A smaller history exponent saw an increase in uniform sampling of all optima, regardless of their relative quality.

When applied to the 3-pot holes problem the replacement error problem was not as evident, probably due to the fact that there were fewer optima to consider than Shekel's fox holes. This is to some extent validated by the results for CPACOUnity-CFO whose results were fairly uniform for the range of values tested for the crowding window size. However, while the crowding window size didn't have a marked effect, the performance of CPACOQuality-CFO and CPACORank-CFO seemed to depend mostly on the history exponent with better performance observed when this parameter was smaller. This result may be due to the positive reinforcement process described in the previous example. Even though the population is seeded with a uniform random sample, if the history exponent is large then a few solutions may dominate the selection process. Thus a large proportion of the population may reside close to one optimum. This may mean that even though there may be population members located at other optima, they will not be improved towards their respective optima since they are unlikely to be selected, thus increasing the value of the modified max-peak ratio.

To this point the problems that require multiple solutions seem to be advantaged by large crowding window size and small history exponent values. For problems requiring the identification of a single global optima, i.e., Rastrigin's and Ackley's, the reverse was observed to be true. These problems seem to favour smaller crowding window sizes and larger history exponents. This result is not unexpected since for the two dimensional case these problems are not overly complex and so it may not be so beneficial to spread the population around the search space. An effect of the application of niching is a slowing down of convergence to some degree. This means that when the CPACO-CFO variants are tuned for multiple solution cases (large crowding window, and small history exponent), they have no doubt located the same global optima as their counterparts that are tuned for a single solution case (small crowding window, and large history exponent). The difference is that by tuning for single solutions the algorithms spend more time exploiting the best solutions and thus return a better result for this individual global optimum. In this case the replacement error problem that was identified earlier can actually be beneficial to the algorithms search development since it allows more resources (in the

[6]Remember that the 25 optima in Shekel's Fox Holes are of different quality

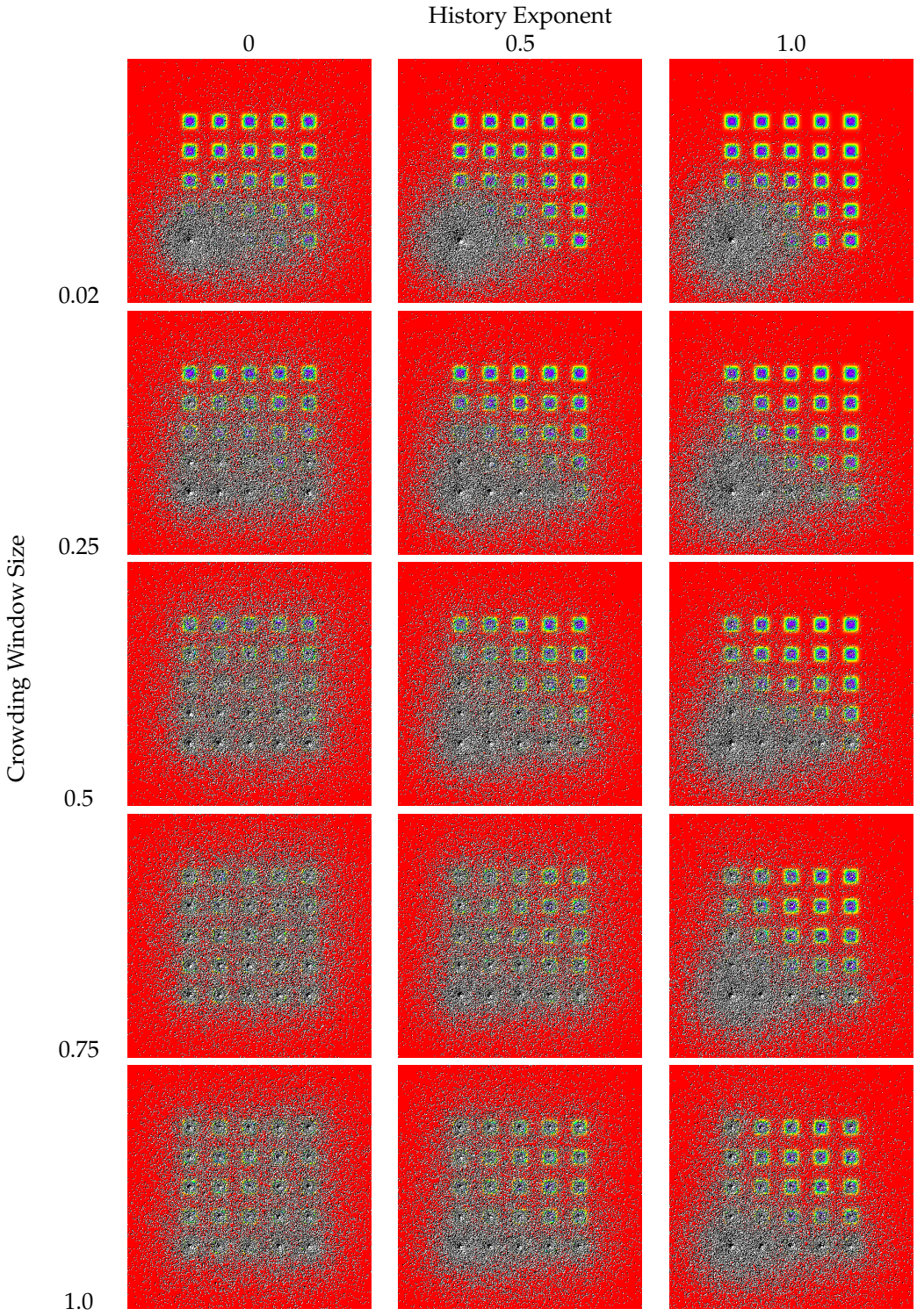


Table 6.7: Shekel's foxholes: Effect of varying crowding size and history exponent, white dots represent all solutions evaluated during 50,000 evaluations.

form of population members) to be allocated to the best areas of the search space.

6.3.5 Results and Discussion: Quantitative Analysis

Using the insights gained from the sensitivity analysis we can now embark on an analysis of the performance of the Niching PACO algorithms as compared to existing algorithms. To judge the new algorithms' performance a PACO algorithm without niching (ACOCD) and the Deterministic Crowding Genetic Algorithm (CGA), are also applied to the chosen benchmarks. It is expected that for problems requiring the location of multiple optima the CGA will be a good benchmark of expected performance, and for problems requiring the location of a single global optimum ACOCD will be a good basis for comparison. Initially the modified max-peak ratio will be used to report the performance of the algorithms, later the best solution found is used as a basis of comparison for one of the problems scaled to larger numbers of dimensions. The existing algorithms are used as benchmarks of an approximate level of good performance that the niching PACO algorithms should be able to achieve. The significance of results found are discussed where appropriate using the Mann-Whitney rank-sum test.

Four benchmark problems have been chosen for the first experiment: Schaffer's, Schwefel's, Himmelblau's and Branin's R-Cos. All of these problems are defined in 2 dimensions, and like the previous sensitivity analysis there are two problems with a single desired optimum and two problems with multiple desired optima.

The second experiment involves two problems: Schwefel's and Three Pot Holes. These problems are chosen since they can be scaled to any number of dimensions. This experiment is to test the scalability of the algorithms, namely how well the algorithms perform as the size of the search space is increased.

It was highlighted in the sensitivity analysis that the performance of all CPACO-CFO variants is strongly dependent on the number of optima required and that by altering the crowding window size and history exponent these algorithms can be tuned for multiple or single optimum cases. As such the CPACO-CFO variants are tested using three configurations that benefit the location and maintenance of a single optimum, multiple optima and a balance of both. In the sensitivity analysis it was found that the FSPACO-CFO algorithm performed similarly on single and multiple optima instances and thus a single parameter configuration is used to test this algorithm. The CGA and ACOCD algorithms are configured to enhance their best performance as multiple optima and single optimum algorithms respectively. The parameter settings chosen for all algorithms are presented in Tab. 6.8. The CPACO-CFO variants use the following combinations of crowding window size and history exponent: Greedy (0.1,2.0), Balanced (0.5,1.0), Fair (1.0,0.2).

CPACO-CFO	
Parameter	**Value**
Number of ants / Population size	50
History Exponent (only used with CPACO-Quality & CPACORank)	2.0, 1.0, 0.2
Crowding window size	0.1, 0.5, 1.0
FSPACO-CFO	
Parameter	**Value**
Number of ants / Population size	50
History Exponent	1.4
Niche radius	0.02
ACOCD	
Parameter	**Value**
Number of ants	2
History Size	50
Solution selection greediness (q)	0.85
Convergence rate (ξ)	0.0001
CGA	
Parameter	**Value**
Population size	200
Crossover Probability	0.95
Mutation Probability	0.85
Standard Deviation of Gaussian used for Mutation	0.01% of dimension range

Table 6.8: Parameter settings used for quantitative analysis

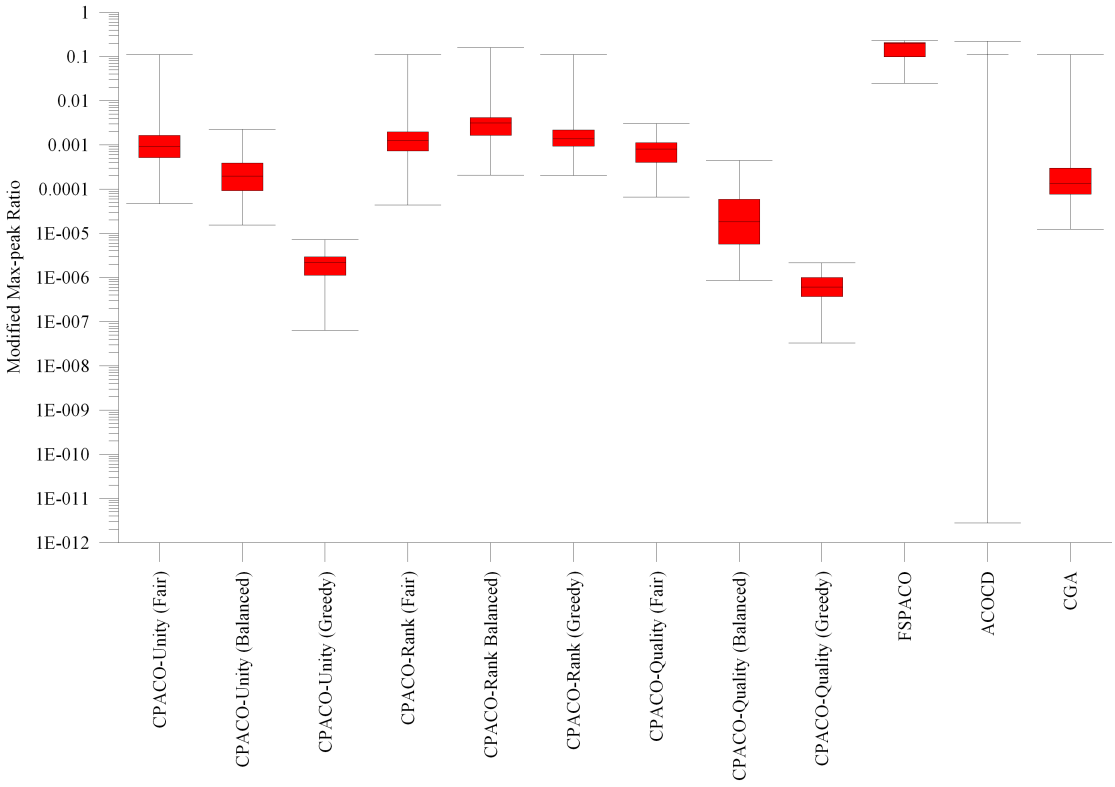
Experiment 1: 2D Problems

For this experiment each algorithm was allowed 50,000 function evaluations[7]. Each algorithm was run 100 times using different random seeds and the results presented as box-and-whisker plots (Fig. 6.20) to indicate not only the mean level of performance but also the reliability of each algorithm. Tables indicating the statistical significance of the results are included in the Appendix as Tab. 10.2, 10.3, 10.4 & 10.5.

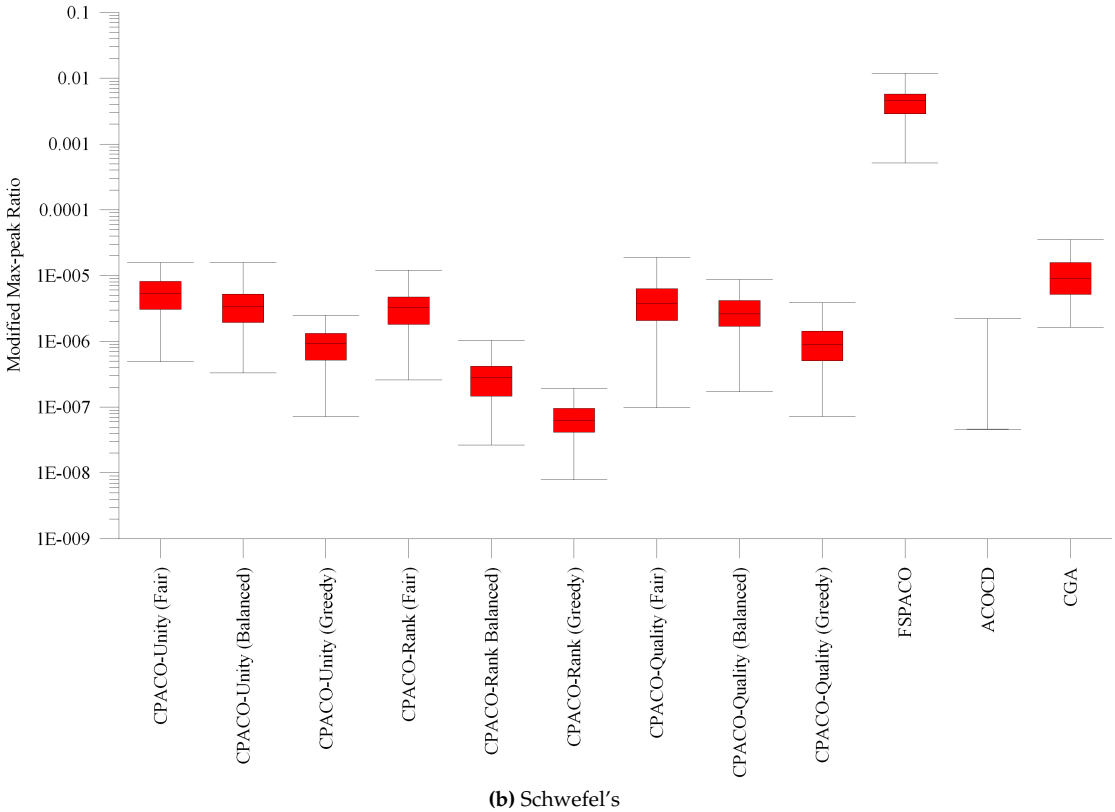
For Schaffer's function the ACOCD algorithm produced the best minimum performance, however the median performance was the second worst just ahead of FSPACO-CFO. This erratic performance is most likely due to the presence of other local optima in the search space. The ACOCD suffers from premature convergence to a suboptimal area of the search space and thus while it does converge quite well to the extreme of the optimum it selects, it is making the wrong choice of optimum to converge towards. The CPACOUnity-CFO and CPACOQuality-CFO algorithms performed better with the balanced and greedy configurations proving the best.

For Schwefel's function the ACOCD algorithm was not overcome by the premature convergence problem it had with Schaffer's function and managed to converge very close to the global optimum in all cases. The greedy configuration of CPACORank-CFO and ACOCD performed the best but all of the algorithms tested (except for FSPACO-CFO) managed to get extremely close to the global optimum (within 0.00001%). It was no surprise that for this problem the

[7]Past this many evaluations the increase in quality of the result was no longer worthwhile; since the algorithm is intended to be a global search algorithm, past this point a local search routine would be far more effective.

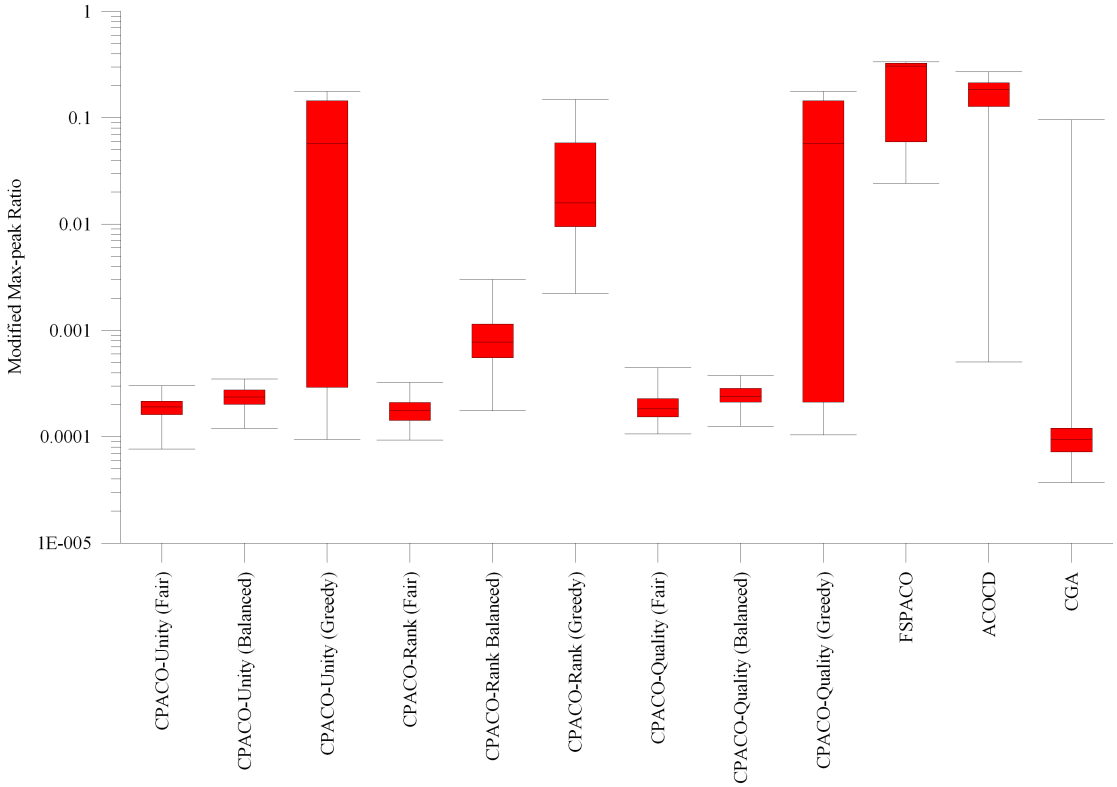


(a) Schaffer's

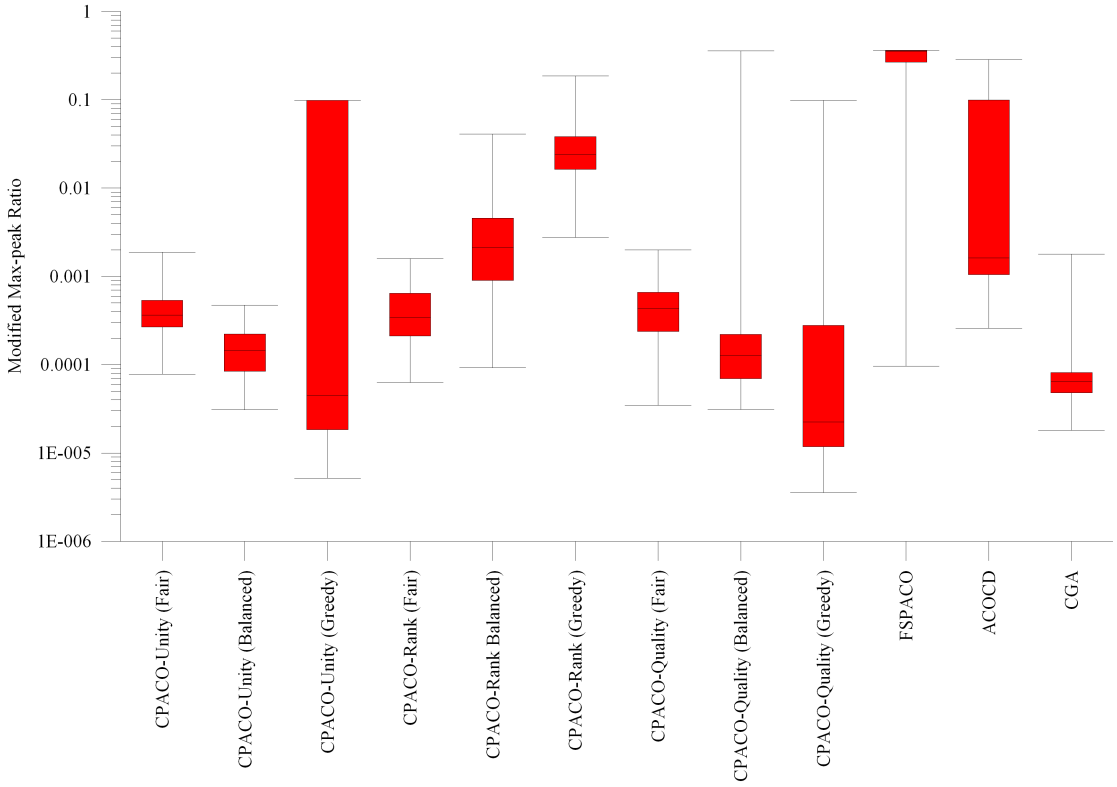


(b) Schwefel's

Figure 6.20: Results (modified max-peak performance metric) presented from 100 repeats of the Niching PACO algorithms and the ACOCD and CGA each allowed 50,000 solution evaluations per experimental run and configured according to the parameters presented in Tab. 6.8.



(c) Himmelblau's



(d) Branin's R-Cos

Figure 6.20: (cont'd) Results (modified max-peak performance metric) presented from 100 repeats of the Niching PACO algorithms and the ACOCD and CGA each allowed 50,000 solution evaluations per experimental run and configured according to the parameters presented in Tab. 6.8.

CPACO-CFO variants performed somewhat better when configured to be more greedy.

The algorithms behaved consistently on the remaining two functions, Himmelblau's and Branin's R-Cos. For these problems the ACOCD and FSPACO performed poorly, this being no surprise in the case of ACOCD since it has no inbuilt mechanism to simultaneously converge to multiple optima. CPACOUnity-CFO and CPACOQuality-CFO produced consistently good results when configured to be fair or balanced, however when configured as greedy they both produced a wide range of average results. CPACORank-CFO produced consistently good results when configured to be fair.

To summarise the findings for this section, the balanced configuration settings for CPACO-CFO produced consistently good, reliable results. It could be concluded then, that if nothing is known of a problem domain that these settings would be a good way to configure the CPACO-CFO algorithms, and as such these settings are used for the next experiment. When compared to the control algorithms the CPACO-CFO variants produced good results whereas the FSPACO-CFO algorithm performed poorly on all problems tested. The poor results obtained by FSPACO-CFO are in-line with findings obtained from earlier work with fitness sharing which comment that the high sensitivity of the niche radius and other parameters hinder the ability of fitness sharing to obtain good results [31, 157].

The poor results for FSPACO may also be attributed to the combination of solution construction (stepwise construction) and population replacement techniques. FSPACO uses a generational replacement operation where the current population is replaced completely by the new generation. This means that if a good solution component is not used (or only used in poor quality solutions) it may be lost in successive generations. This is confirmed somewhat by the good results of the CPACO-CFO variants which utilise a form of elitism by only allowing existing solutions to be replaced with better solutions. This said FSPACO-CFO is still included for comparison in the next experiment, although its performance is not expected to be good for the reasons just provided.

The variation of the crowding window size and history exponent of the CPACO-CFO variants produced the expected behaviours for multiple and single optimum problems. For the next experiment the CPACO-CFO variants are parameterised using the balanced setting, with a crowding window of 0.5 and a history exponent of 1.0. These settings are sufficient to obtain a mean level of performance to determine the scalability of the CPACO-CFO variants.

Experiment 2: Higher Dimension Problems

This experiment will determine how well the niching PACO algorithms perform as the number of decision variables (dimensions) is increased. Two test problems, Schwefel's and Three Pot Holes, have been selected since they allow arbitrary scaling of their dimensions and are single optimum (Schwefel's) and multiple optima (Three Pot Holes) problems.

Schwefel's function is defined in 2, 5, 10, 30 and 100 dimensions and Three Pot Holes is defined in 2-10 and 100 dimensions. Each algorithm was allowed 50,000 function evaluations for every instance, except for the Three Pot Holes function in 100 dimensions which was allowed up

to 300,000 function evaluations. In the case of the random search 100 uniformly distributed random samples were taken. Each algorithm was run 100 times using different random seeds and the results, in the form of the average modified max-peak ratio and average percentage deviation from optimum[8], are presented in (Fig. 6.21).

The results for the Three Pot Holes problem speak well for the niching PACO algorithms ability to locate and maintain multiple optima. As far as the niching PACO algorithms were concerned the CPACO-CFO variants performed best on the lower dimensions (2-10). The performance of the CPACO-CFO variants followed that of the CGA algorithm closely although the performance of CPACORank-CFO was slightly worse than CPACOQuality-CFO and CPACOUnity-CFO. Figure 6.22 presents the results of testing all algorithms on the Three Pot Holes problem in 100 dimensions with increasing numbers of function evaluations (up to 300,000 function evaluations).

An analysis of Fig. 6.22 reveals a steep increase in performance when the number of function evaluations are doubled from 50,000 to 100,000, however past this point there is minimal increase in performance. The better performing algorithms are the CGA, FSPACO-CFO and CPACOQuality-CFO.

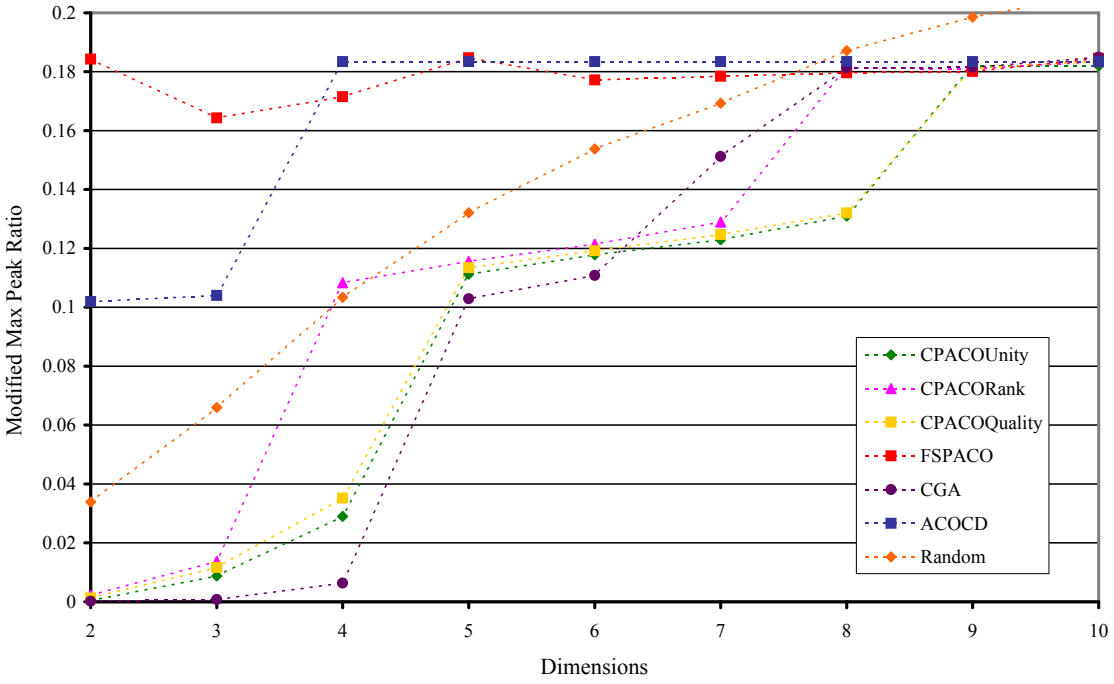
For Schwefel's problem the average best solution found by CGA, CPACO-CFO and FSPACO-CFO algorithms surpassed that found by the non-niching algorithm, ACOCD. This result suggests that the useful diversity gained through the use of crowding and fitness sharing means an increase in solution quality for larger numbers of function dimensions. As an additional side benefit this result is good since niching allows multiple optima to be located while, in this case, simultaneously assisting the identification of a global optimum. *A priori*, it was expected that the identification of multiple optima would decrease the algorithms ability to exploit good areas of the search space, however, as has been demonstrated, this is not the case. Of the niching algorithms tested the CPACO-CFO variants produced very good results. These algorithms were the best performing algorithms for the parameters and range of dimensions tested.

6.3.6 Summary

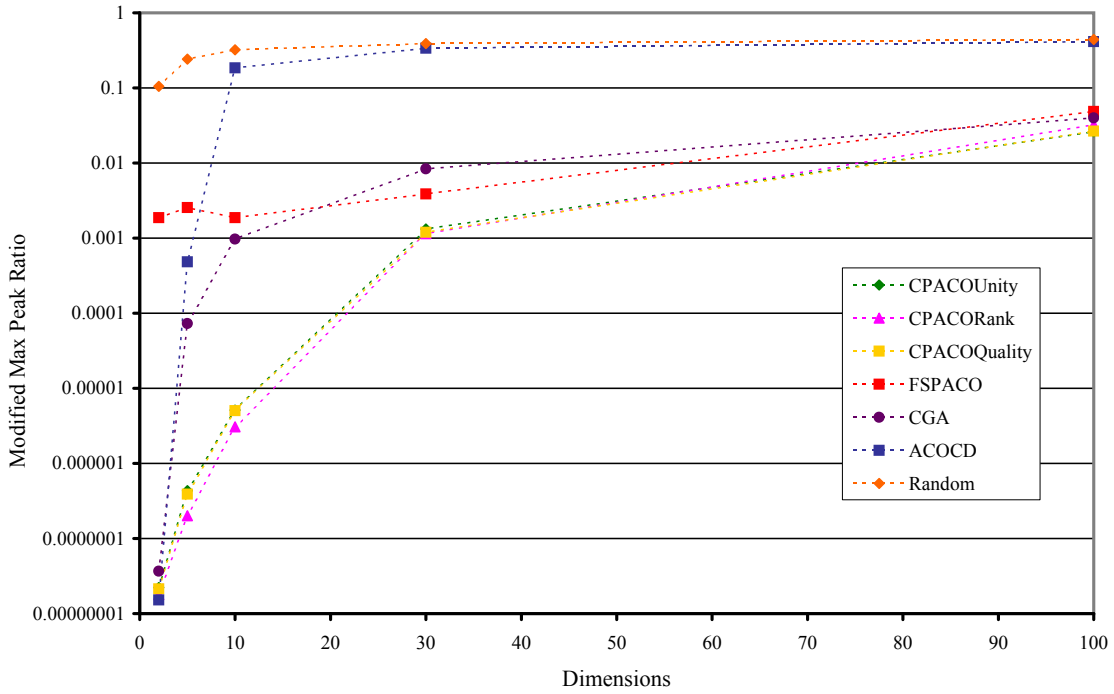
In this section the niching PACO algorithms CPACO-CFO and FSPACO-CFO were tested on a variety of single objective function optimisation problems. Niche formation and maintenance by the niching PACO algorithms was demonstrated on a variety of test cases. A range of parameter settings were tested to determine the effect of newly introduced algorithm specific parameters. For the CPACO-CFO variants the following determinations were made:

- For problems requiring the location of multiple closely located optima, a large crowding window size (1.0) and small history exponent (0.2) are recommended.
- The converse is true for problems containing few or individual optima that are located a distance from each other, as such better performance is obtained by a small crowding

[8]The average percentage derivation from optimum is a ratio derived by normalising the returned value across the functions range, so that all points will be in the range [0,1].



(a) Average modified max-peak ratio found for 3 Pot Holes Function



(b) Average best solution found (% deviation from optimal) for Schwefel's Function

Figure 6.21: Results of testing on benchmark functions in various numbers of dimensions (decision variables).

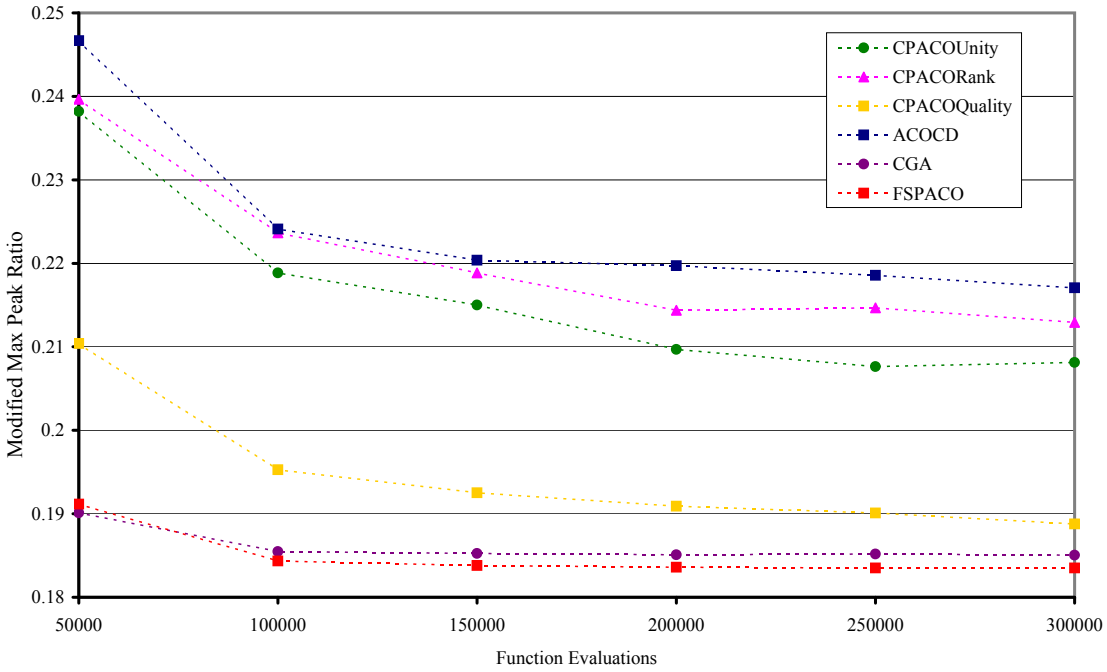


Figure 6.22: Results of testing on Three Pot Holes defined in 100 dimensions with a varied number of function evaluations. Results reported are the average modified max-peak ratio.

window (0.1) and large history exponent (2.0).

- If in doubt about the nature of the optima for a function, moderate settings of the crowding window (0.5) and history exponent (1.0) are recommended.

Of the three variants tested CPACOUnity-CFO is assessed as being the most promising CPACO-CFO variant. This is due to it demonstrating good performance for all problems tested (in terms of both multiple optima location and maintenance and the location of single global optimum) and that it has the lowest computational complexity of the CPACO-CFO variants tested since it does not require any ranking of solutions in the population to assign individual quality measures.

The performance of FSPACO-CFO was poor compared to the CGA and CPACO-CFO variants and it is postulated that this poor performance may be due to two factors: disruptive solution construction and generational replacement. These two factors mean that over time the algorithm may lose good solution information and as such 'forget' specific optima.

6.4 Chapter Summary

In this chapter the niching PACO algorithms were applied to a variety of single objective TSP and CFO problems. The algorithms do indeed exhibit niche formation and maintenance characteristics which meant that results for the CFO problems were good with regard to the ability to locate and maintain multiple optima and to assist in the location of single global optimum. For the benchmark TSP problems tested it was evident that niching does not assist in the location of global optimum. It was demonstrated that the most likely cause for this result was the

nature of the search space of the TSPs tested. A single test case, the Crown TSP, was fabricated to demonstrate how niching may be useful for a very specific kind of TSP.

In the next chapter the multiple objective TSP and multiple objective function optimisation problems are introduced which extend the complexity of the single objective cases presented here. While niching has been demonstrated to be of no benefit to solving standard single objective TSPs, the multiple objective TSP is a different case since it requires the location of multiple solutions versus a single global optimum. The multiple objective function optimisation problem is also different since in some cases it contains an infinite number of optima, rather than a finite number like the single objective case.

CHAPTER 7

Multiple Objective Optimisation

> The project will come in on time, on budget and will be of good quality... now pick any two
>
> Engineering adage

7.1 Introduction

In this chapter two PACO algorithms incorporating niching concepts are developed to attempt to solve two challenging multiple objective problem classes. The problem classes chosen are extensions of the previously tested single objective problem classes; The Travelling Salesman Problem and Continuous Function Optimisation. A PACO algorithm has already been previously defined for the Multiple Objective Travelling Salesman Problem (MOTSP), thus the proposed niching PACO algorithm is an extension to this algorithm. For the Multiple Objective Function Optimisation problem (MOFO) no true ACO approaches exist and thus a new algorithm is proposed by reusing and extending upon algorithmic components already discussed in this thesis.

An analysis of existing approaches for Multiple Objective Optimisation (MOO) is offered first which includes discussion of some ant-inspired approaches. Several quantitative analysis techniques are described which are used for algorithm performance analysis in the later sections. The remainder of the chapter is dedicated to the MOTSP and MOFO problems with one niching PACO algorithm developed for each problem class. The new algorithms are tested in each of these sections on several benchmark problems with results compared against current state-of-the-art MOO algorithms.

Current research trends [22, 36] include the application of multiple objective evolutionary algorithms (MOEA) to problems with more than two objectives, and the reduction of computational complexity, thus the two MOO problems presented in this section include variants with more than two objectives. Of interest as well is the identification in the literature of the importance of niching/diversity preservation in MOO algorithms. This will be commented on later as it is a major motivation for the application of Niching PACO algorithms to these problems.

7.2 Existing Algorithms

Research into MOEA has been ongoing for the last two decades, however it has received somewhat of a surge of interest in the past decade [21]. In this section several important MOEA are introduced and discussed. The focus of the discussion is not to reiterate the exact algorithm implementation details, rather it is on identifying the key features of these algorithms and as such what are deemed as essential components of a MOEA.

Historically the first MOEA proposed was the vector evaluated GA or VEGA, which was first proposed by Schaffer in 1984 [134, 135]. The VEGA was a simple idea which extended on the canonical GA. The difference came in the evaluation of solutions for selection purposes for which VEGA partitioned the population evenly according to the number of objectives (e.g. a population of 90 for a three objective problem would be partitioned into three groups of 30). After partitioning, each partition is evaluated according to a single objective. The intention being that solutions that are good for one objective will mix with good solutions from other objectives in the reproduction phase and produce better solutions closer to the Pareto front. Realistically though, what actually happens is that the population ends up comprised of mostly solutions that are good in one objective only. As a first approach to MOO though, VEGA was a computationally lean and somewhat effective algorithm.

Little work on MOEA was published for ten years after Schaffer's initial VEGA, although the VEGA was mentioned by Goldberg in [71] and some suggested improvements made which incorporated early ideas on the use of dominance for solution ranking and niching as a way to enhance population diversity. These ideas, while not imbued into a specific algorithm at the time, form the basis of most state-of-the-art MOEA today.

MOGA which stands for Multi-objective Genetic Algorithm, was introduced in [62]. Like VEGA it is a very simple extension of the canonical GA which alters the fitness assignment routine. It is mentioned here because the fitness assignment is a rank-based procedure which tests solutions in the population for dominance. If a solution is non-dominated it receives a rank of one, otherwise it receives a rank equal to one plus the number of solutions dominating it. These ranks are then used as the basis for a second fitness assignment routine which also incorporates fitness sharing to enhance population diversity.

The aforementioned algorithms are among what are known as first generation MOEA [22], to which other similar algorithms such as the Non-dominated Sorting Genetic Algorithm (NSGA) [144] and Niched Pareto Genetic Algorithm (NPGA) [85, 86] also belong. These first generation MOEA are typified by simplistic dominance-based solution ranking and the use of standard crossover and mutation routines. The second generation MOEA began to emerge in the mid to late 90s and represent a body of algorithms designed from the ground up to be applied to multi-objective problems rather than modifications of single objective algorithms applied to multi-objective problems. Many second generation MOEA include multiple populations and the use of explicit elitism.

While there are many algorithms that belong to the second generation MOEA, such as SPEA

[165], SPEA2 [164], PAES [93], PESA [29] and PESA2 [28], this chapter focuses on one second generation MOEA, the NSGA-II, since this algorithm remains a state-of-the-art MOEA today and contains many important characteristics of a well designed MOEA.

NSGA-II [37] despite its name is not simply an extension to NSGA. From a population of solutions (size = N) the algorithm creates N new solutions using a crowding tournament selection routine and standard GA crossover and mutation operations. The new solutions are added to the population such that the population becomes of size $2N$. To control the size of the population the solutions are sorted according to successive dominance, Fig. 7.1 gives an example of the resultant solution ranks assigned using this ranking procedure. The worst ranked solutions are successively discarded from the population in groups until such time as to discard a particular ranked group would allow the population to become less than N. At this point this group of solutions are ranked according to a second criterion, a crowding distance that, simply put, measures the distance from this solution to its closest neighbours in the entire population. The worst solutions are those that are closest to another solution in the population and using this criterion the population is reduced to size = N.

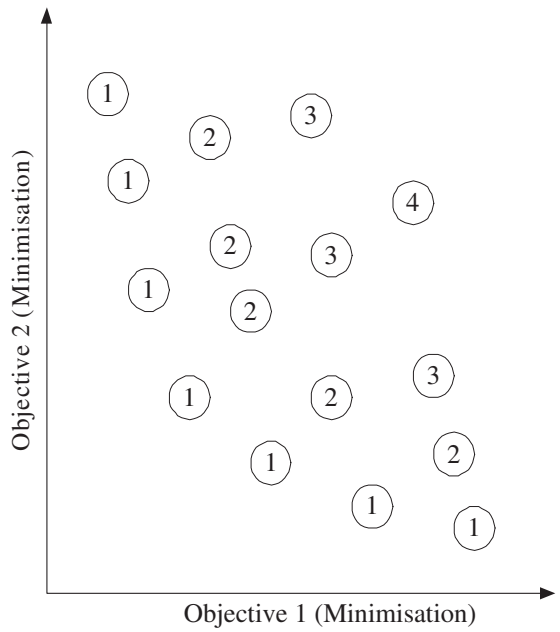


Figure 7.1: Example of the resultant ranks assigned to a set of solutions using the NSGA-II dominance ranking procedure

MOEA are not limited to only Genetic Algorithms or Evolutionary Systems approaches. Since the early 90's researchers in the ACO field have also attempted to solve MOO problems through the modification of existing ACO-based approaches. The focus in this area though has been on solving mostly multi-objective combinatorial problems rather than multi-objective function optimisation problems as in the GA community.

The earliest references found to a multi-objective ACO algorithm were with [68, 67] and [130]. The first of these suggested the use of a multi-colony ACS variant for a multiple objective vehicle routing problem (MACS-VRP). For this problem the total number of vehicles and total travel time were to be minimised (these are purported to be conflicting objectives). The algorithm is

not a Pareto-based approach since it optimises the objectives independently in the two separate colonies, although it shares solutions between these processes. The reduction in travel time seems to be the dominant objective whereas the reduction of vehicles is mostly a secondary consideration. Given this it seems unlikely that the algorithm would be able to be applied to a MOO problem with evenly weighted objectives, though it does seem to produce very good results for its chosen problem. The second example approaches a water distribution problem in much the same way with an extension of the Ant-Q algorithm (MOAQ). MOAQ distributes a population of ants amongst the objectives evenly and independently optimises each objective while assigning reward to those solutions that produce solutions that satisfy a primary objective well. Like the previous example MOAQ is not Pareto-based approach and thus it would be unlikely that the MOAQ algorithm would perform well on more general MOO problems.

Another two-phase approach is used in solving a bi-criteria flow shop problem in [152]. This example may almost be considered an example of multi-objectivization [95], which is the process of constructing a multi-objective problem from a single-objective problem instance with the view that the problem can be solved better if phrased in a multi-objective way.

MACS-VRP was extended in [5] with a Pareto-based approach that eliminated the requirement for multiple pheromone matrices[1]. In the new algorithm the amount that solutions update the pheromone matrix depends on how close they are to the Pareto front as compared to other solutions of the same generation. The extension proved to be successful on a multicast routing problem that was defined using multiple conflicting objectives.

Shelkoar et. al. [121] proposed an ant-inspired approach for the MOFO problem, however, since this approach was based on the Ant Colony Metaphor for Continuous Design Spaces it is strictly not an ACO algorithm per se [142]. While the algorithm was demonstrated as a good approach for the problems tested, it resembles something closer to a Genetic Algorithm since it uses crossover and mutation to generate new solutions rather than stepwise construction.

Cardoso et. al. implemented a simple multi-objective ACO variant to a bi-objective TSP in [19]. This variation involved multiple pheromone and heuristic matrices that were combined to form a single probability value for each solution component (much like that of PACO-MO in Sec. 4.4). The algorithm performed well against a Genetic Algorithm, and was the subject of a later study when many other ACO approaches were also compared on a series of multi-objective TSP [70]. This was mentioned previously in Sec. 4.4 since the PACO-MO algorithm was tested and performed amongst the best of all tested algorithms. This algorithm is included later in this chapter as a benchmark algorithm since it is a demonstrated good approach for solving the Multi-objective TSP and because it is a PACO algorithm.

[1] It should become apparent that most multi-objective ACO algorithms tend to split the objectives amongst multiple colonies or pheromone matrices.

7.3 Performance Metrics

Performance metrics that reflect the different facets of algorithm performance are important in stochastic algorithm design and validation. For single objective problems these metrics are usually fairly straightforward to obtain. This is because algorithms applied to these problems usually return a single best solution leading to the development of a univariate distribution when run multiple times. This means that a wealth of parametric and non-parametric statistical techniques are available to allow comparison between multiple experimental runs, some of which have been used in prior chapters of this thesis.

MOO presents a further challenge to this performance analysis process since MOO algorithms can return multiple Pareto and sub-Pareto optimal solutions rather than a single best solution. A non-trivial exercise is the development of unary quality metrics that, when compared, can allow the algorithm designer to make a reliable comparison between different algorithms. In [166] and [94] the authors offer an excellent analysis of many older unary metrics that measure the distribution of solutions and their closeness to the actual Pareto front. The metrics include Generational Distance [158], Spacing Metric [37], Maximum Pareto Front Error [158] and Extent of Approximation Set [36]. In their analysis the authors challenge many assumptions about the reliability of these metrics by providing case studies where many of these indicators give false or misleading assessments as to the quality of different distributions. One important conclusion made is that the combination of these older metrics was often thought to provide more reliable information, in a 'strength in numbers' way. However, this is not the case as the combination of multiple inaccurate metrics does not make them any more accurate. In fact it is argued that just one good quality unary metric would be better than multiple bad metrics.

In this section many carefully chosen performance metrics are introduced. The metrics introduced are for the most part reliable enough to be used on their own, however, as will be discussed some metrics are complimentary and when used together allow more robust conclusions to be made.

7.3.1 Summary Attainment Surface

The summary attainment surface is in some ways inspired by the traditional univariate distribution. In traditional univariate distributions single values represent the mean, minimum and maximum among other important features. With summary attainment surfaces the same is true, however instead of a single value being used, a multi-dimensional surface (with the same number of dimensions as the number of objectives), represents these features [64].

Once obtained, summary attainment surfaces that describe different algorithms' mean performance on a single problem can be compared using non-parametric statistical methods like Mann-Whitney and Kruskal-Wallis rank sum tests. An advantage of this approach is that it eliminates the requirement for specialised knowledge of the problem domain (such as *a-priori* knowledge of the location of Pareto optimal solutions). Other performance metrics tend to require knowledge of the Pareto front to obtain difference measurements between it and an

experimental result. A concern is that this limits the kind of problems that can be measured. Mainly because a deterministic solver must be used to ensure Pareto optimal solutions from across the entire Pareto front are obtained, and can also mean that different Pareto fronts are obtained by different researchers, thus skewing results.

In this study summary attainment surfaces are used for a combination of qualitative and quantitative analysis. The summary attainment surface plotting method [92] is used to visualise and thus qualitatively compare multiple summary attainment surfaces. All summary attainment surfaces produced in this study have been created with the aid of Knowles' software package available from `http://dbkgroup.org/knowles/plot_attainments/`.

7.3.2 Hypervolume

The hypervolume metric [163] is a measure of the volume of the enclosed space between a reference point z and the weakly dominated portion of the objective space (fig. 7.2). The setting of the reference point can be accomplished by taking the two sets to be compared, taking the extreme worst objective values of these sets and using the combination of those values as the reference point. The reference point must be strongly dominated by every member in all sets for any comparison to be meaningful.

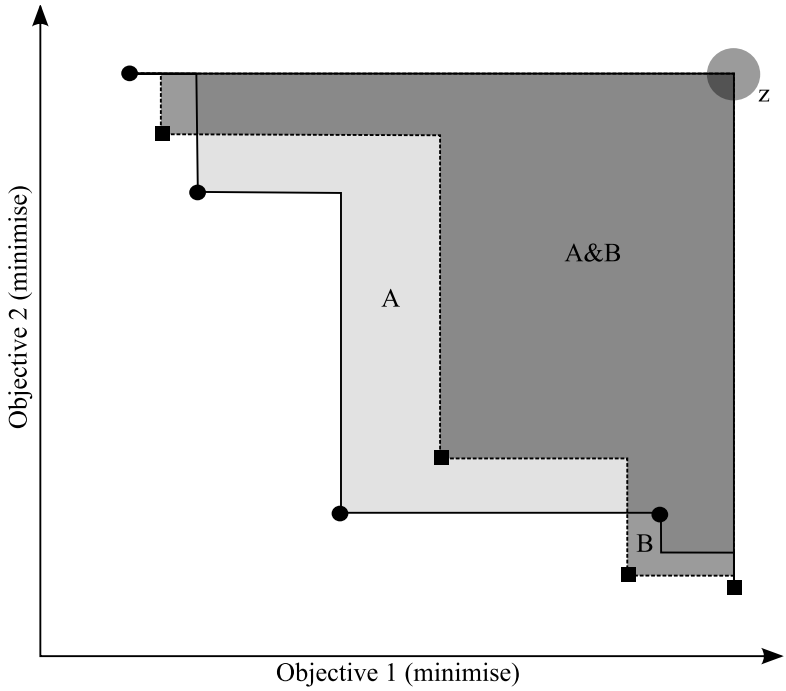


Figure 7.2: Example of two hypervolumes created from two different sets, A and B. In this example A would be better than B since A's hypervolume is larger.

7.3.3 Epsilon Indicator

Introduced in [166] the epsilon indicator is a value ϵ that represents the minimum factor that every point in one set needs to be multiplied by, so that the resultant transformed set is weakly

dominated by a second reference set. The principle of ϵ-dominance is defined as (7.3.1).

$$s^1 \preceq_\epsilon s^2 \Leftrightarrow \forall i \in 1..n : s_i^1 \leq \epsilon \times s_i^2 \tag{7.3.1}$$

In short this metric indicates the amount by which the solutions in one set would have to be moved to be equivalent to a reference set. The less movement that is required, the better. This is a useful metric as, like the hypervolume, it provides an indication as to the extent by which one set is better or worse than another. However, being slightly different to the hypervolume it may be possible to get a reversed outcome when comparing two sets which would indicate that the two sets are incomparable. In this way it is very useful to treat the hypervolume and epsilon indicator as a joint measurement pair that can reliably indicate whether one set of solutions is better, no different or worse than another.

7.3.4 Use of Performance Metrics

The summary attainment surfaces provide us with a good visual qualitative assessment tool and can also be compared using a statistical testing procedure to provide a more formal assessment as to whether two sets of results are comparable. In this chapter summary attainment surfaces will be used to provide visual inspection and also used to compare algorithms performance. Results will be reported visually in the form of two and three dimensional plots and tests will be reported in tables with the statistical significance of any comparison reported.

Hypervolume and Epsilon Indicators are both useful as they allow another form of comparison between two or more algorithms. The actual values for these metrics are not reported though since the use of a reference point (z) means that the actual metric value is relative to the setting of this reference point. Even though the reference point can be reported, from a future comparison point of view reporting the final population of solutions is more beneficial. By reporting the final population it allows any other future comparison to be made, not just those reported here. Furthermore, in this case the metrics are supplementary to the summary attainment surface comparison, therefore the actual results are less meaningful than the comparison which will be used to validate conclusions made using the summary attainment surface method.

7.4 Niching PACO for the Multiple Objective Travelling Salesman Problem

Instead of just considering one cost matrix as in the single objective TSP we can consider several (maybe conflicting) cost matrices which may refer to length, monetary cost, time, etc. This problem is a multiple objective TSP (MO-TSP) where we are interested in obtaining solutions that optimise multiple, usually conflicting, objectives. A practical example of a multiple objective TSP can be found in transportation where we can simultaneously consider the length of a route, monetary cost (the route may include toll roads) and the time taken (the shorter route

in terms of length may include more congested or lower speed limited roads). The MO-TSP is similar to the single objective TSP in terms of the solution representation, however, the resultant search spaces are quite different.

As was stated in Sec.7.2 there have been many ant-inspired algorithms proposed for multi-objective optimisation problems. In [70] eight major ant-inspired algorithms along with two state-of-the-art Genetic Algorithms (NSGA-II & SPEA2) were benchmarked using the MO-TSP. For the particular test cases used (instances of a bi-criteria TSP) the ant-inspired algorithms performed well. One algorithm in particular, Population-based ACO for multiple objectives (PACO-MO) (discussed in Sec. 4.4), performed consistently within the top three ant-inspired algorithms. Since PACO-MO is amongst the best ACO approaches for the MO-TSP, it is used as a benchmark here. It was highlighted by Guntsch [76] that an area of improvement for the PACO-MO algorithm may be the introduction of an intelligent population control mechanism, since the PACO-MO algorithm does not place an upper bound on the population size. The use of a niching mechanism to control which solutions are inserted into the population may provide an answer to this previously posed research question.

In this section an extension of the PACO-MO algorithm is introduced, Crowding Population-based Ant Colony Optimisation for the Multi-objective TSP (CPACO-MOTSP). This algorithm is then tested using several benchmark MO-TSP, specifically moderate sized bi-objective cases. The results of this initial testing are used as the basis of a qualitative and quantitative assessment of the performance of the new algorithm as compared to PACO-MO. The new algorithm is then tested with an increased number of objectives to determine its scalability, with regard to the number of objectives.

7.4.1 Algorithm Details

Rather than use the super/sub-population scheme as in PACO-MO, a single population (S) of pre-set size is maintained, which is initialised with randomly generated solutions. Every generation a population of new solutions is created (Y) and a crowding replacement operation is used where each new solution is compared against a randomly selected subset S' of S to find its closest match and only replaces the existing solution if the new solution is better. In the case of multi-objective problems better is taken to mean strongly-dominating. This procedure is outlined in Alg. 11.

Algorithm 11 CPACO-MOTSP Crowding Replacement Scheme

```
for i = 1 to Y_size do
    S' = c randomly chosen solutions from S
    S'_j = closest match from S' to Y_i
    if Y_i ≻≻ S'_j then
        Replace S'_j with Y_i
    end if
end for
Discard Y
```

Whereas PACO-MO uses individual pheromone and heuristic matrices for each objective, CP-

ACO-MOTSP uses a single pheromone matrix with individual heuristic matrices. Each iteration a new pheromone matrix is calculated as follows:

1. All solutions (s) in the population (S) are assigned an integer rank according to the dominance ranking procedure used by the NSGA-II algorithm.

2. All elements in the pheromone matrix are initialised to a pre-determined value (τ_{init}).

3. All solutions in the population increment their corresponding elements in the pheromone matrix according to the inverse of their rank, i.e. $\Delta\tau_{ij}^{s} = 1/\left(s_{rank}\right)$.

CPACO-MOTSP does not use the PACO-MO *average-rank-weight method* to determine weightings for each objective as this is seen as computationally expensive and unnecessary, instead it assigns each ant a set of random heuristic exponent weighting factors (λ), similar to the procedure outlined in [5]. This allows each unique ant to exploit heuristic matrices by different amounts while still using a common pheromone matrix. The procedure used to assign these values is outlined in Alg. 12 and the transition probabilities are calculated using (7.4.1), where h defines the number of objectives.

Algorithm 12 CPACO-MOTSP Heuristic Scaling Value Assignment Procedure

```
for i = 1 to h do
    R_i = random [0, 1]
end for
Sort R in ascending order
λ_1 = R_1
for i = 2 to h − 1 do
    λ_i = R_i − R_{i−1}
end for
λ_h = 1 − R_h
```

$$p_{ij} = \frac{\left[\tau_{ij}\right]^{\alpha} \cdot \prod_{d=1}^{h} \left[\eta_{ij}^{d}\right]^{\lambda_d \beta}}{\sum_{l \in N_i^k} \left[\tau_{il}\right]^{\alpha} \cdot \prod_{d=1}^{h} \left[\eta_{il}^{d}\right]^{\lambda_d \beta}} \tag{7.4.1}$$

7.4.2 Test Problems

The '*Kro*' set of Travelling Salesman Problems have been selected to test the CPACO-MOTSP algorithm. This set consists of a series of four 100, two 150 and two 200 city TSP which are combined to create four bi-objective TSPs of varying complexity and a single quad-objective TSP. The individual instances are labelled KroA100, KroB100, KroC100, KroD100, KroA150, KroB150, KroA200 and KroB200 and are available from [128]. Each of the multi-objective TSPs contains a discontinuous, convex Pareto front.

7.4.3 Results and Discussion

Experiment 1: Bi-objective Problems

Both PACO-MO and CPACO-MOTSP were run 50 times using different random seeds and allowed 50,000 solution evaluations on each bi-objective problem using the parameter settings given in Tab. 7.1, where n is the number of cities. This is clearly quite a small amount of solution evaluations when one considers the size of the problems. However, it is deemed enough to determine the general global optimisation performance of these algorithms since after this point the amount of improvement made to the Pareto set begins to fall off quite rapidly. There is no reason why we could not run a local search process at the completion of the global search phase, however to compare the performance of these two global optimisers this local search phase is omitted lest it skew the results. The 1% (best) and 50% (average) attainment surfaces for each of these problems are presented as Figs. 7.3, 7.5, 7.7 & 7.9 and Figs. 7.4, 7.6, 7.8 & 7.10 respectively. These figures indicate that for all bi-objective problems tested, CPACO-MOTSP was able to obtain better mean and absolute attainment surfaces than PACO-MO. This result is validated by the comparison of the attainment surfaces in Tab. 7.2. Also reported with these results are the results of comparing the hypercube and epsilon indicator metrics which both indicate that CPACO achieves a better quality result than PACO.

PACO-MO	
Parameter	**Value**
Number of Ants	1
Sub-population size (k)	5
Population size	$n/2$
Initial Pheromone (τ_{max})	$1/(n-1)$
Pheromone Exponent (α)	1
Heuristic Exponent (β)	3
Solution selection greediness (q)	0.9
Maximum Pheromone	1
CPACO-MOTSP	
Parameter	**Value**
Population size	$n/2$
Number of Ants	$n/2$
Initial Pheromone	$1/(n-1)$
Pheromone Exponent (α)	1
Heuristic Exponent (β)	3
Solution selection greediness (q)	0.9
Crowding window size	0.1

Table 7.1: Algorithm parameter settings for Bi-objective TSP Problems

It is speculated that the PACO-MO algorithm's lack of diversity preservation, combined with a greedy selection strategy and a strongly biased pheromone matrix (PACO-MO only uses a minimal subset of its population to construct the pheromone matrix) most likely leads to premature convergence in suboptimal areas of the search space.

While the strongly biased pheromone matrix allows the algorithm to improve existing solutions towards the Pareto front through small perturbations, it may be making it too difficult for the algorithm to construct solutions which are vastly different from those already in the population

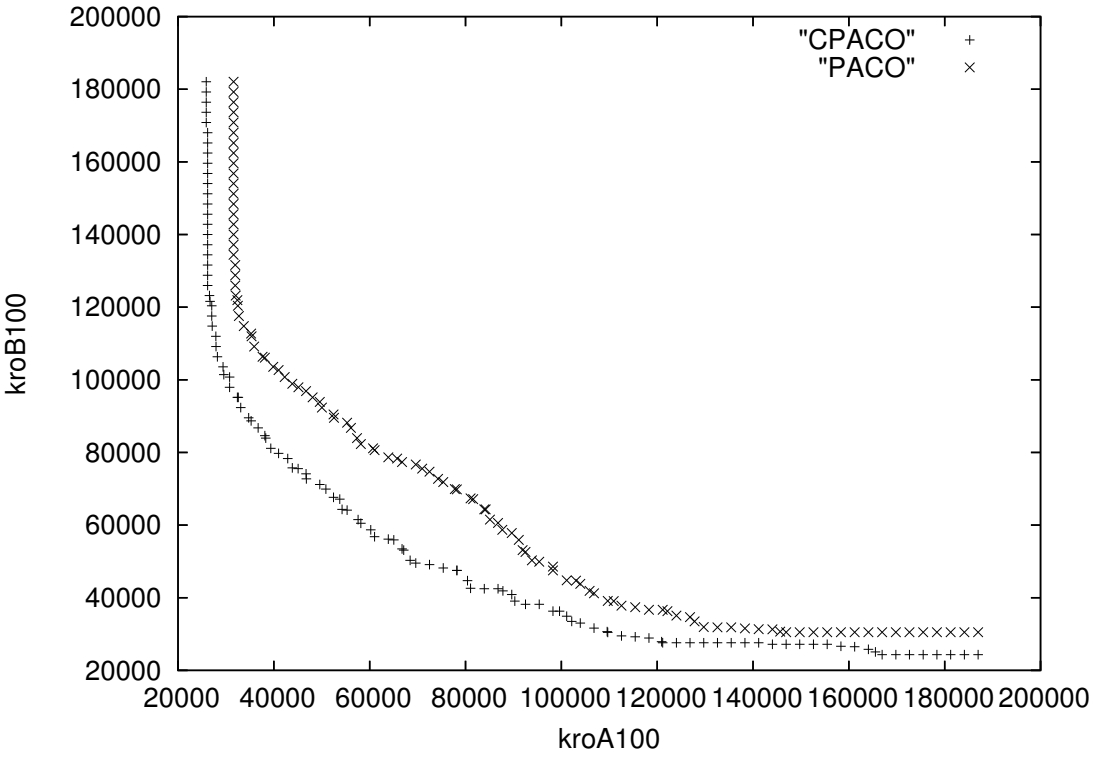


Figure 7.3: 1% (best) attainment surface for kroA100 and kroB100 using PACO-MO & CPACO-MOTSP

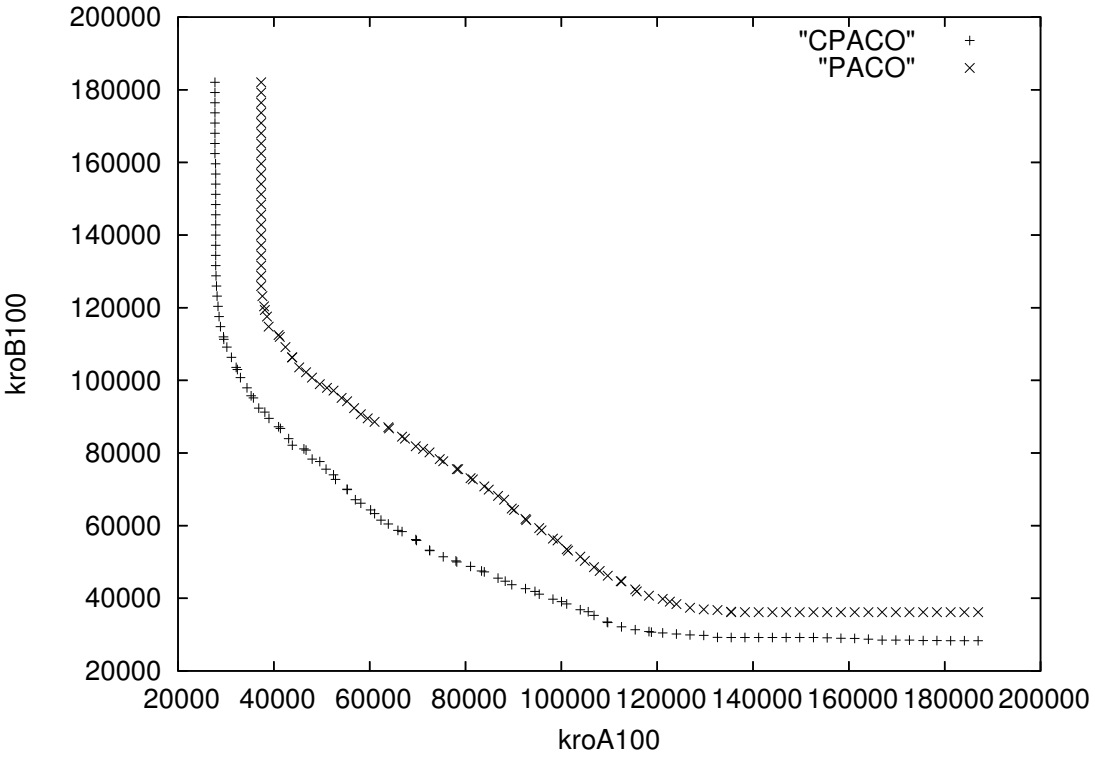


Figure 7.4: 50% (average) attainment surface for kroA100 and kroB100 using PACO-MO & CPACO-MOTSP

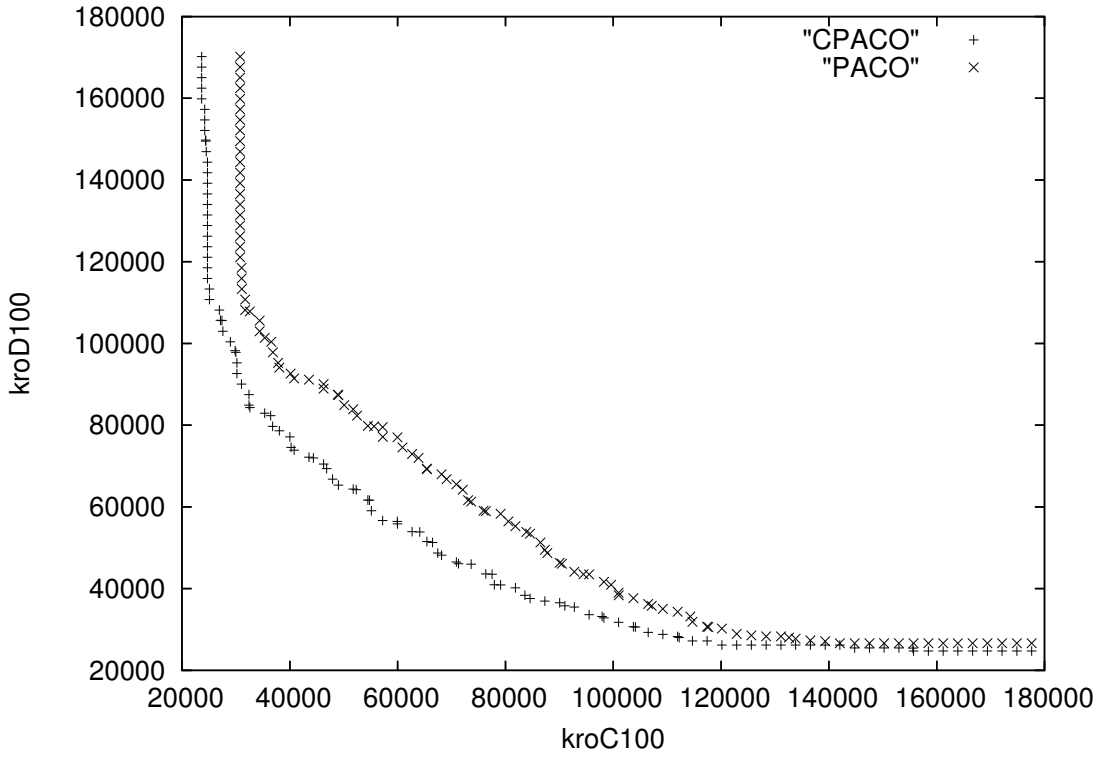


Figure 7.5: 1% (best) attainment surface for kroC100 and kroD100 using PACO-MO & CPACO-MOTSP

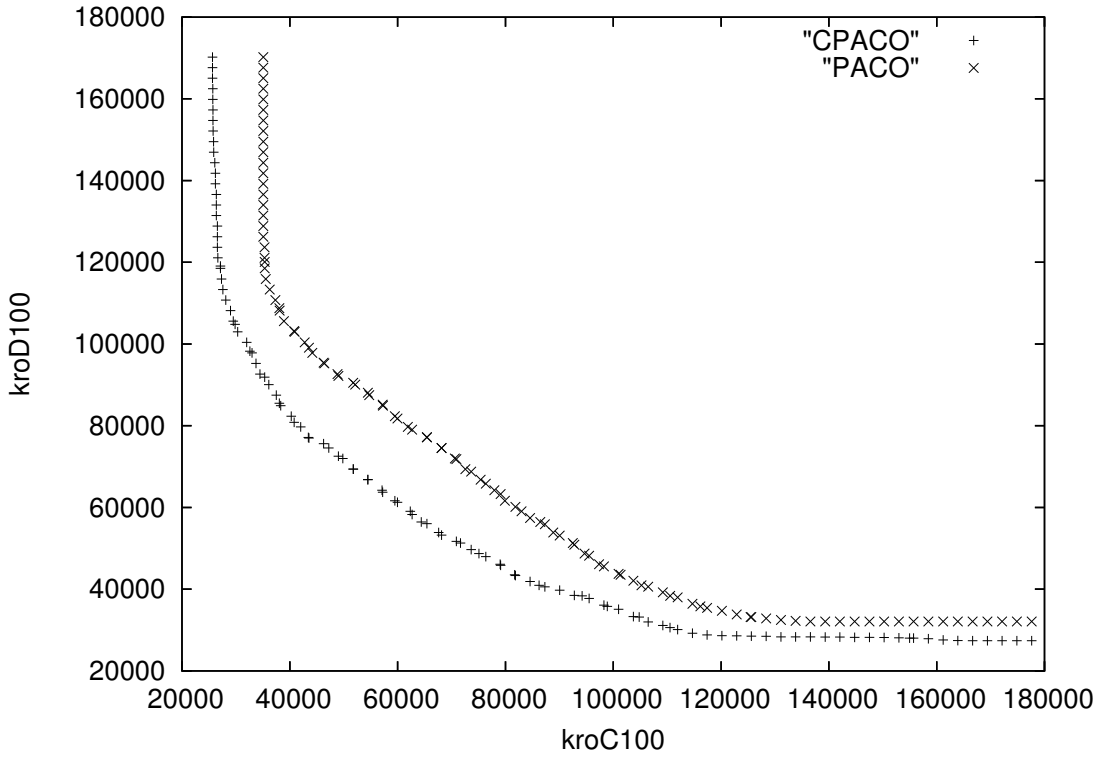


Figure 7.6: 50% (average) attainment surface for kroC100 and kroD100 using PACO-MO & CPACO-MOTSP

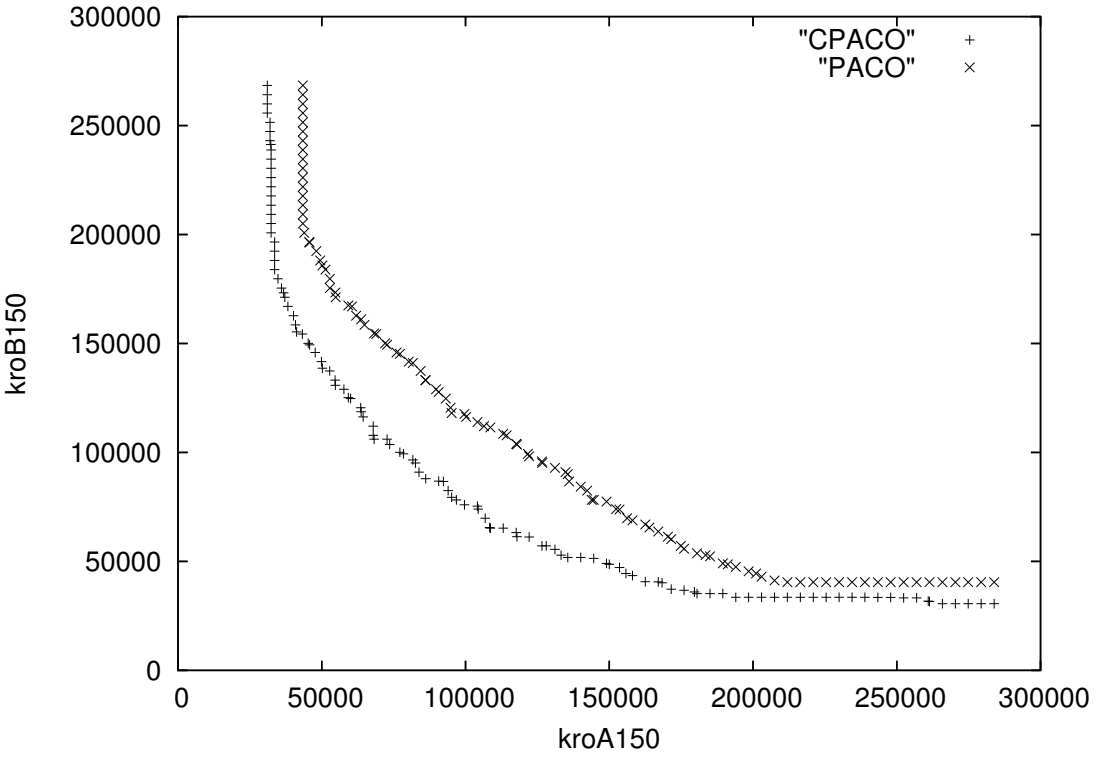


Figure 7.7: 1% (best) attainment surface for kroA150 and kroB150 using PACO-MO & CPACO-MOTSP

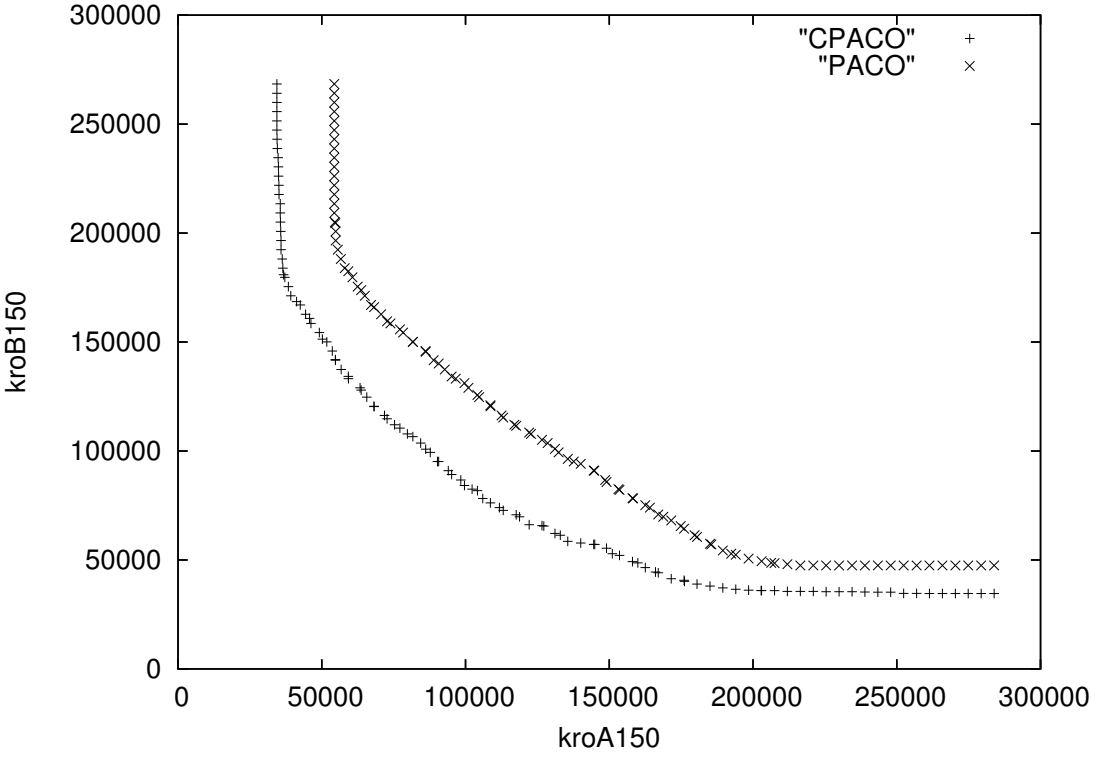


Figure 7.8: 50% (average) attainment surface for kroA150 and kroB150 using PACO-MO & CPACO-MOTSP

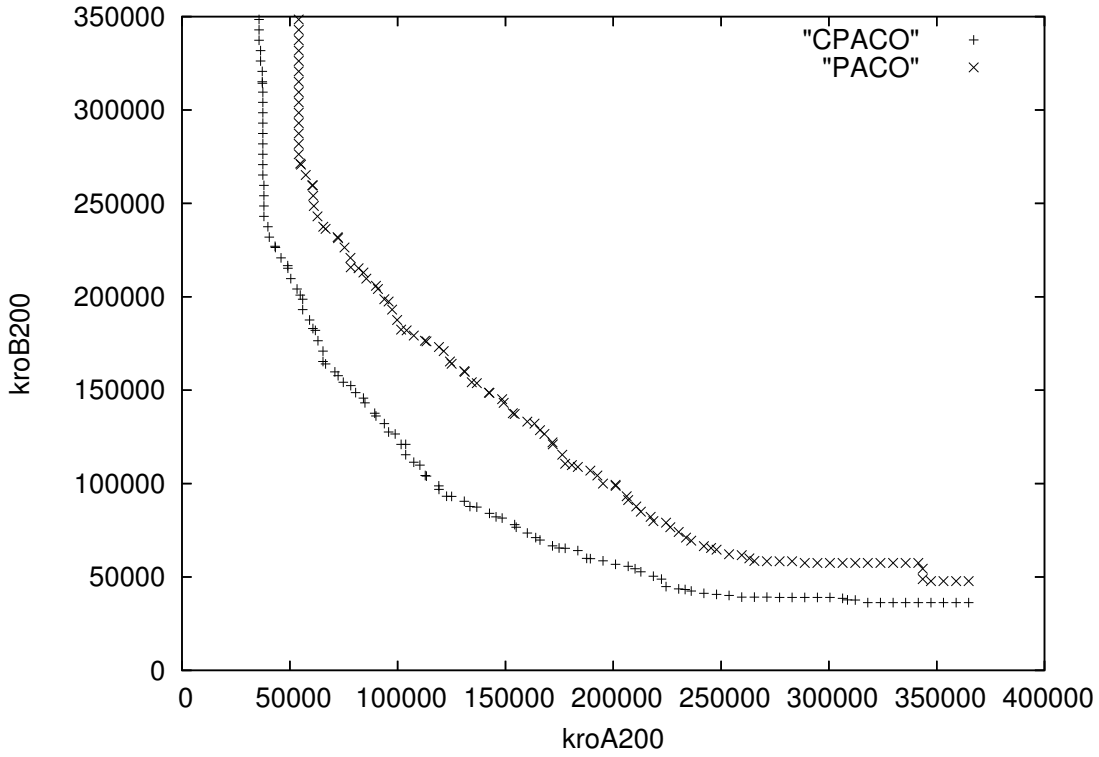


Figure 7.9: 1% (best) attainment surface for kroA200 and kroB200 using PACO-MO & CPACO-MOTSP

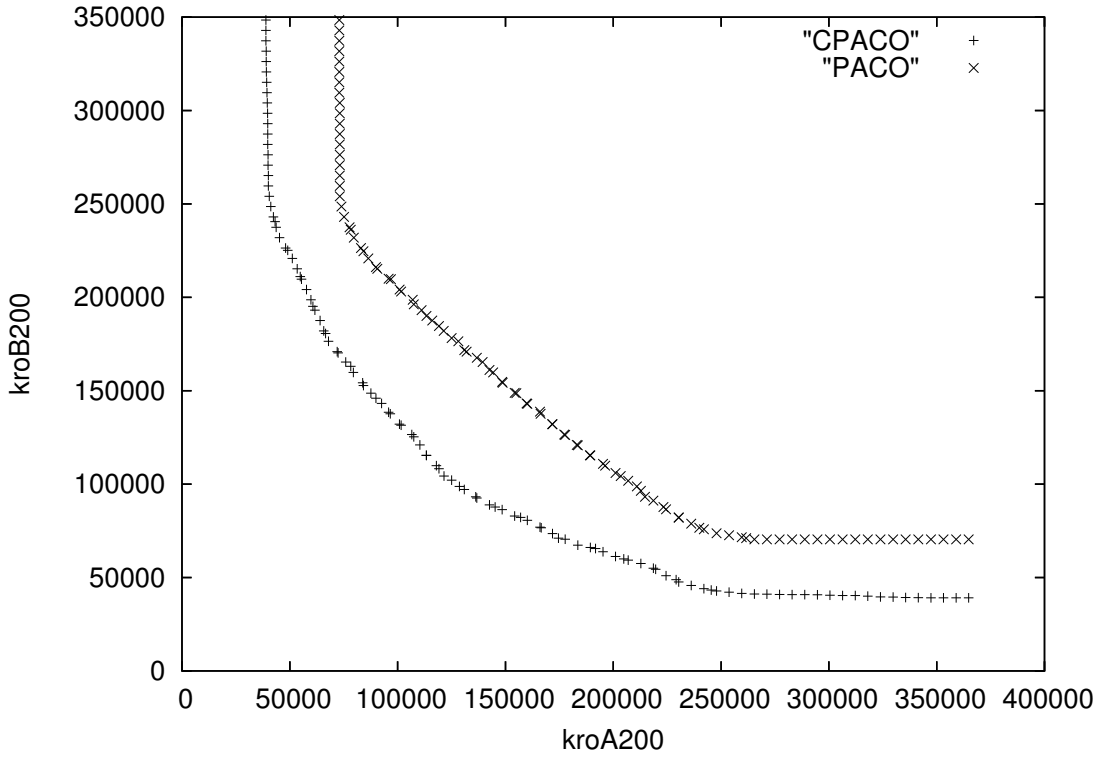


Figure 7.10: 50% (average) attainment surface for kroA200 and kroB200 using PACO-MO & CPACO-MOTSP

Problem	Metric		
	Attainment surface	Hypervolume	Epsilon indicator
KroA100KroB100	> (0)	> (0)	> (0)
KroC100KroD100	> (0)	> (0)	> (0)
KroA150KroB150	> (0)	> (0)	> (0)
KroA200KroB200	> (0)	> (0)	> (0)

Table 7.2: Results of comparison between PACO-MO and CPACO using the summary attainment surface, hypervolume and epsilon indicator metrics. Indicators used indicate if CPACO is significantly better than PACO-MO (>), if there is no significant difference (=), or if PACO-MO is significantly better than CPACO (<). The statistical confidence of the result (p value) is also indicated. The Kruskal-Wallis test was used for summary attainment surface comparisons and the Mann-Whitney Rank-Sum test for all other comparisons.

(Fig. 7.11).

The CPACO-MOTSP algorithm is able to locate and maintain a diverse set of solutions across the Pareto front which may also contribute to the algorithm's ability to find better solutions along all areas of this front (Fig. 7.12). There may be some wasted effort in CPACO-MOTSP since it constructs solutions based on a pheromone matrix which reflects the performance of the entire population, regardless of a solution's position on the approximate Pareto front; however this, combined with the greedy transition rule, makes for a good balance between exploration and exploitation. It is worth noting that the ranking of solutions using the NGSA-II ranking procedure is a vast improvement over early attempts which did not use any ranking and allowed solutions to update the pheromone matrix uniformly.

Experiment 2: Quad-objective Optimisation

CPACO-MOTSP was tested on the KroABCD100 quad-objective problem to examine its performance on a larger problem than the bi-objective problem. The results of the trials were analysed using attainment surfaces by isolating two objectives at a time (kroA100/kroB100 and kroC100/kroD100), even though the problem was attempted simultaneously on all four unique objectives. The attainment surfaces generated from the original bi-objective cases (kroA100/kroB100 and kroC100/kroD100) were used as comparisons against the quad-objective case isolated to two objectives. An ideal outcome of the analysis would be if little to no difference is observed between the previously obtained attainment surfaces and the quad-objective case analysed in two objectives.

For this experiment the population size and number of ants were both increased to 200, the crowding replacement size was kept at 0.1, and the number of iterations was increased from 50,000 to 100,000, 250,000 and 500,000. All other parameters remain the same as stated in Tab. 7.1, results are reported for 50 repeats from random starting positions. Results of just the quad-objective cases isolated to objectives kroA100 & kroB100 and kroC100 & kroD100 with different numbers of iterations are included in Figs. 7.13, 7.14, 7.15 & 7.16. The 100,000 iteration quad-objective case isolated to objectives kroA100 & kroB100 and kroC100 & kroD100 is compared against the original bi-objective cases in Figs. 7.17, 7.18, 7.19 & 7.20.

From the attainment surfaces it is evident that the extra iterations provided to the CPACO-

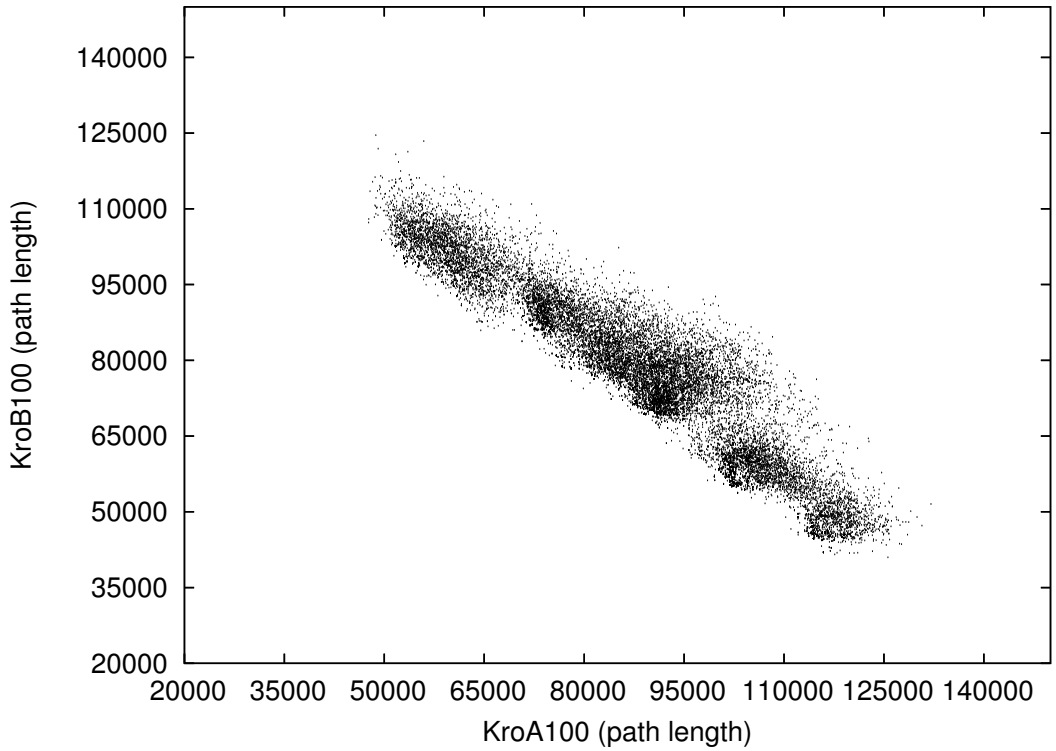


Figure 7.11: PACO-MO sampling behaviour for one complete experimental run (20,000 evaluations) of the algorithm on the KroA100KroB100 problem. Each point indicates an evaluated solution.

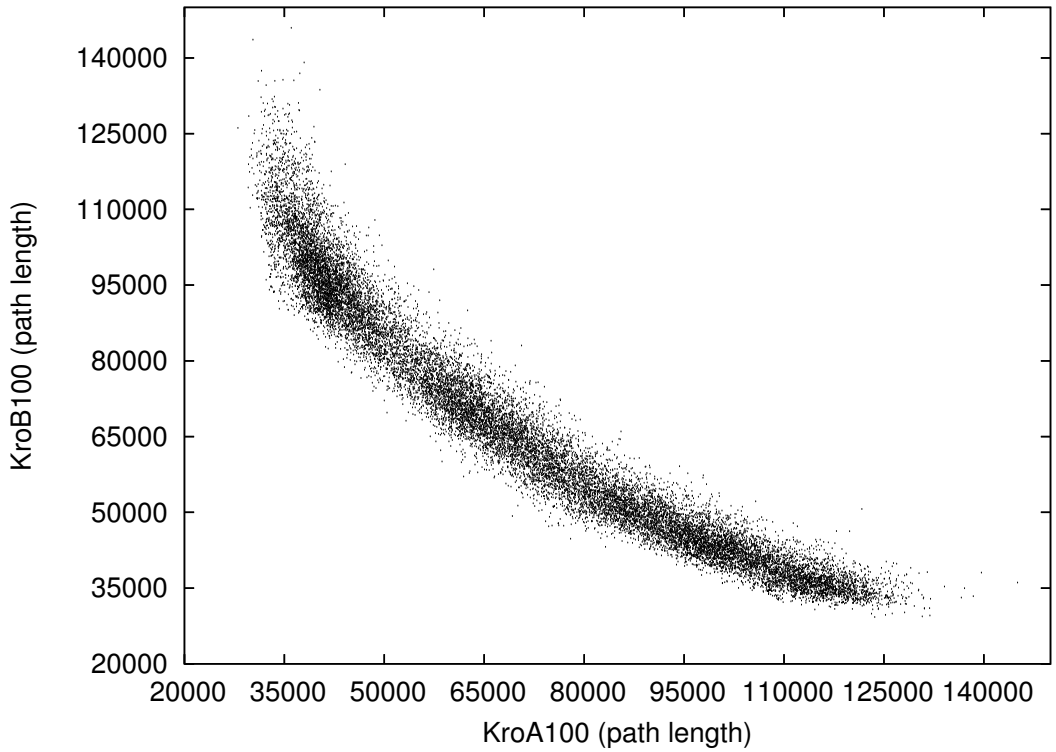


Figure 7.12: CPACO-MOTSP sampling behaviour for one complete experimental run (20,000 evaluations) of the algorithm on the KroA100KroB100 problem. Each point indicates an evaluated solution.

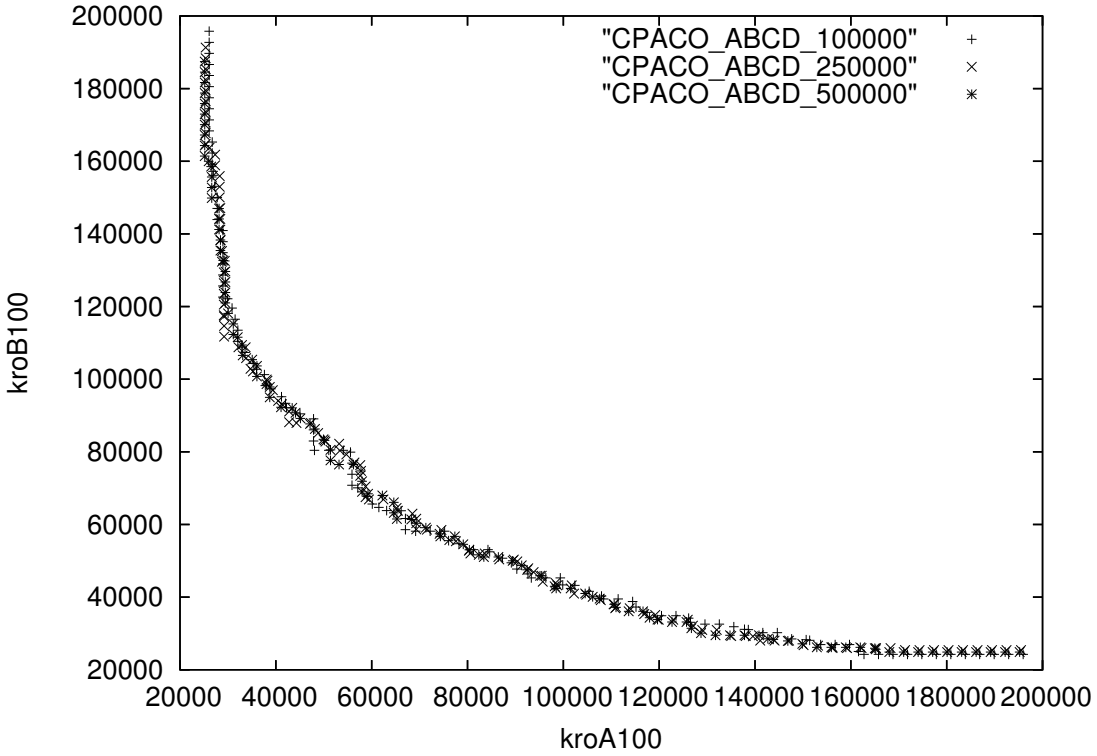


Figure 7.13: 1% (best) attainment surface for CPACO-MOTSP applied to kroA100, kroB100, kroC100 & kroD100 (isolating KroA100 & KroB100) with 100,000, 250,000 and 500,000 solution evaluations.

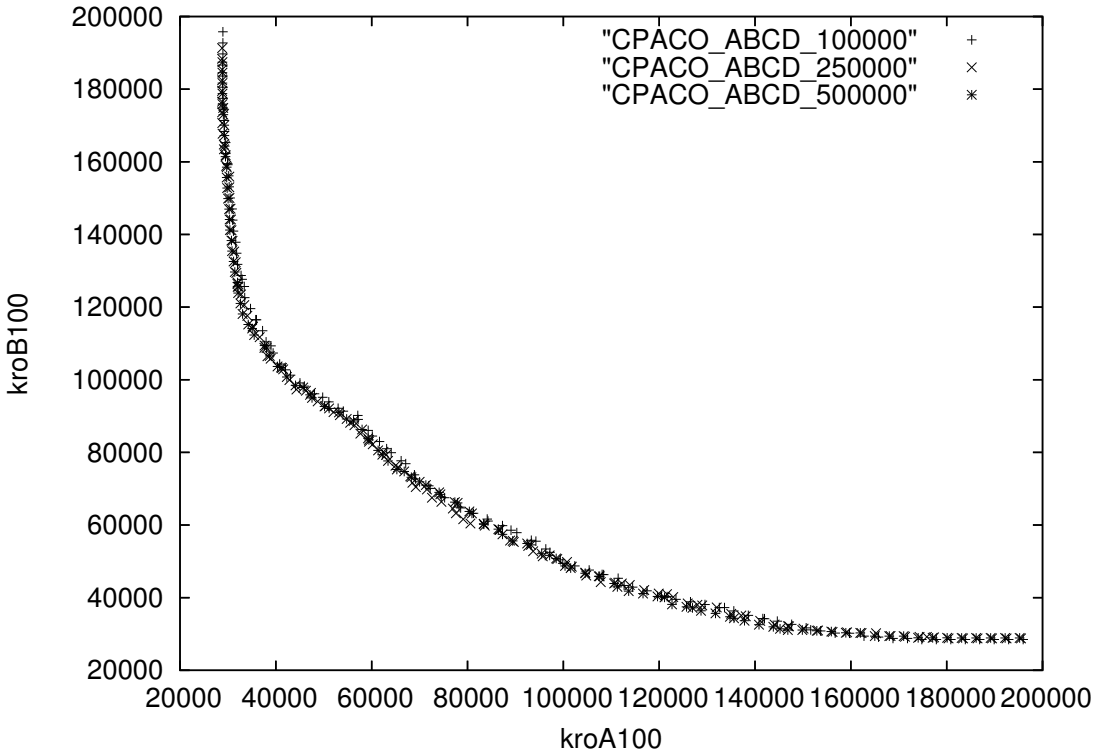


Figure 7.14: 50% (average) attainment surface for CPACO-MOTSP applied to kroA100, kroB100, kroC100 & kroD100 (isolating KroA100 & KroB100) with 100,000, 250,000 and 500,000 solution evaluations.

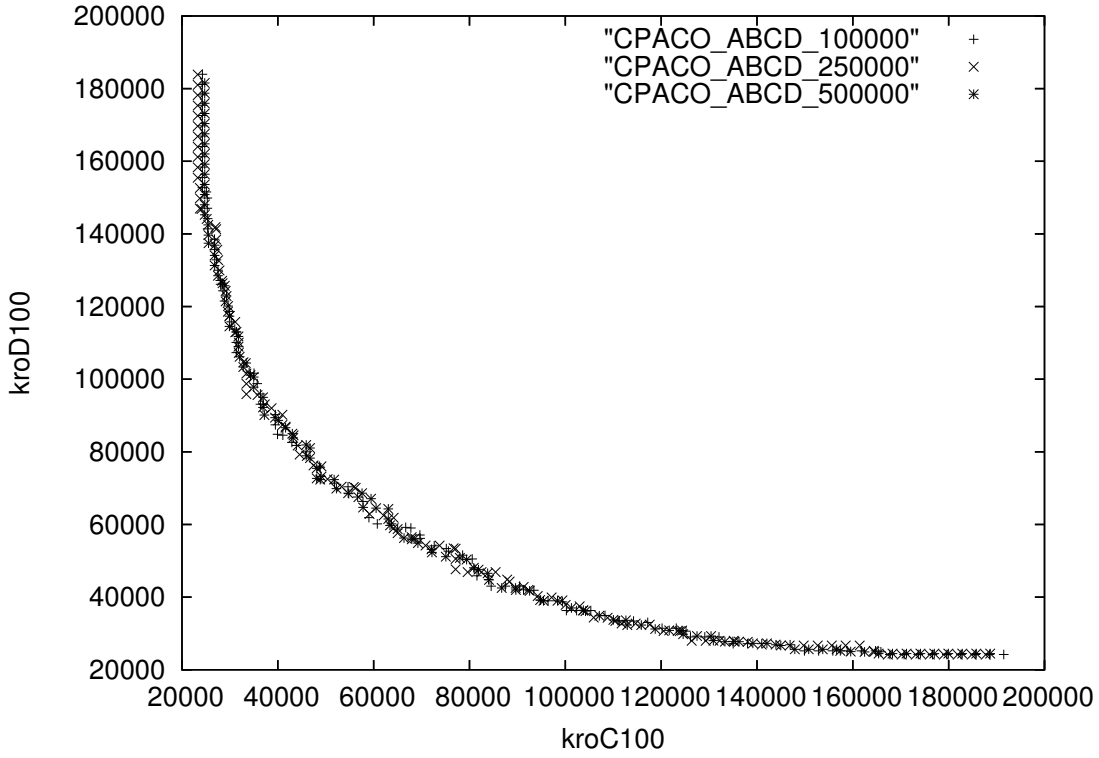


Figure 7.15: 1% (best) attainment surface for CPACO-MOTSP applied to kroA100, kroB100, kroC100 & kroD100 (isolating KroA100 & KroB100) with 100,000, 250,000 and 500,000 solution evaluations.

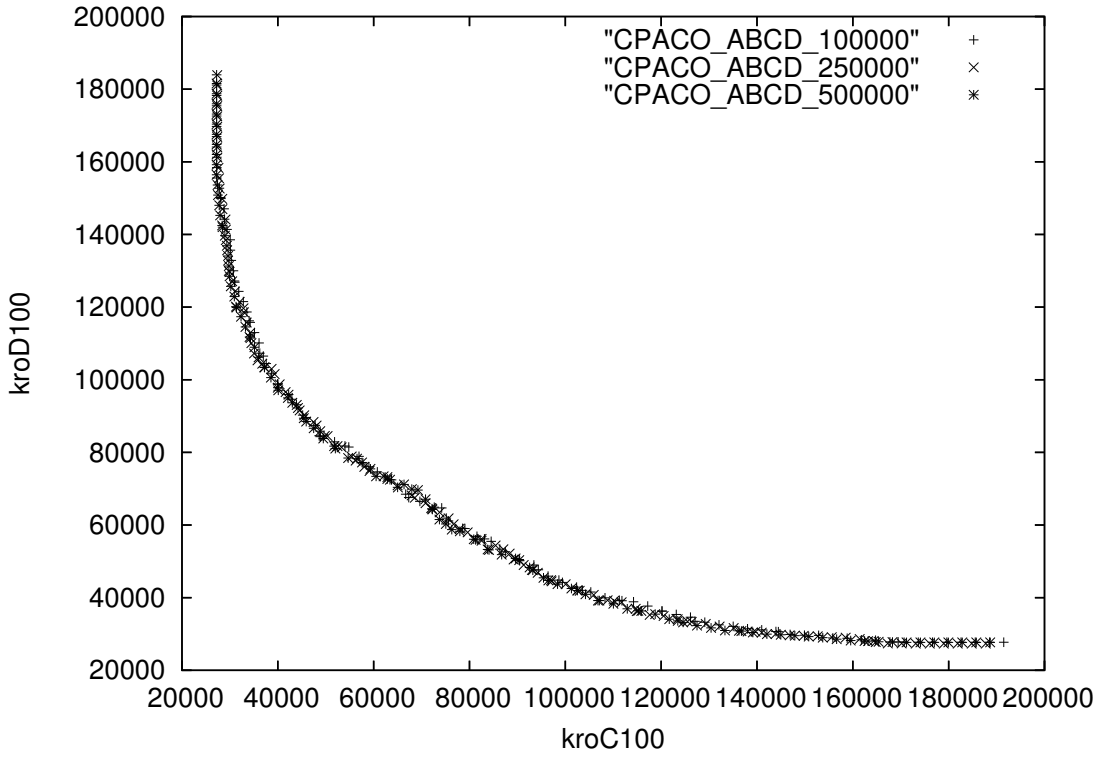


Figure 7.16: 50% (average) attainment surface for CPACO-MOTSP applied to kroA100, kroB100, kroC100 & kroD100 (isolating KroA100 & KroB100) with 100,000, 250,000 and 500,000 solution evaluations.

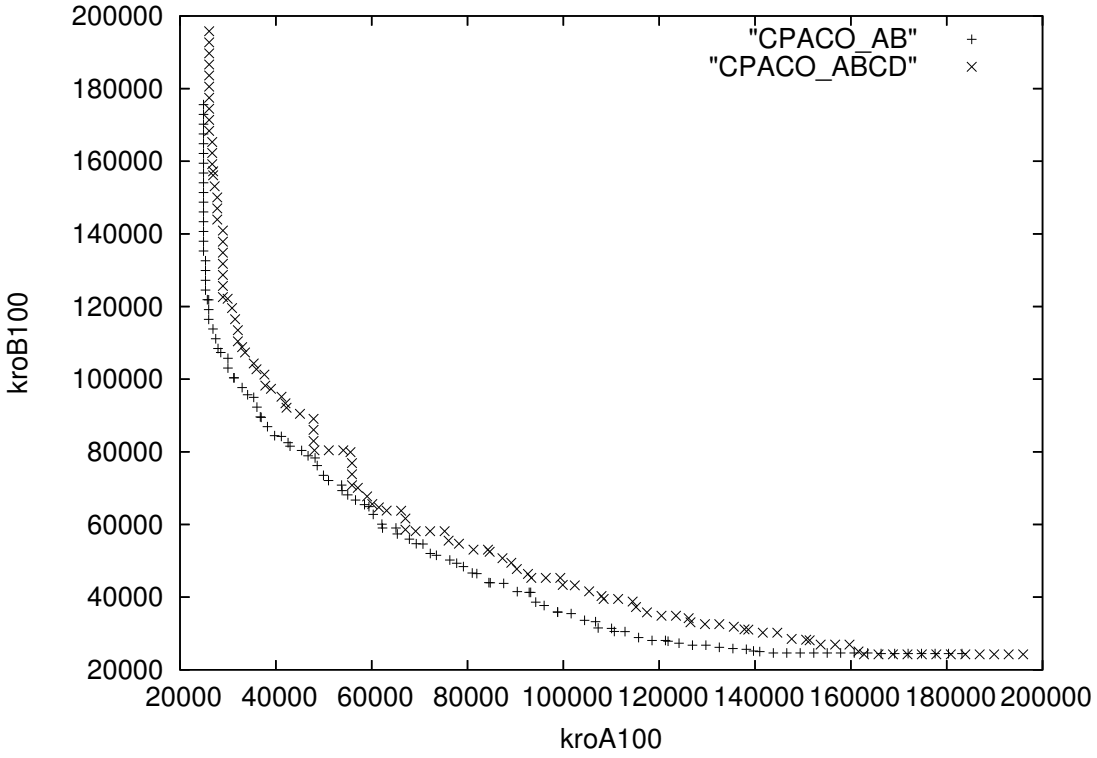


Figure 7.17: 1% (best) attainment surface for CPACO-MOTSP applied to kroA100, kroB100, kroC100 & kroD100 (*CPACO_ABCD*), and CPACO-MOTSP applied to only kroA100 & kroB100 (*CPACO_AB*)

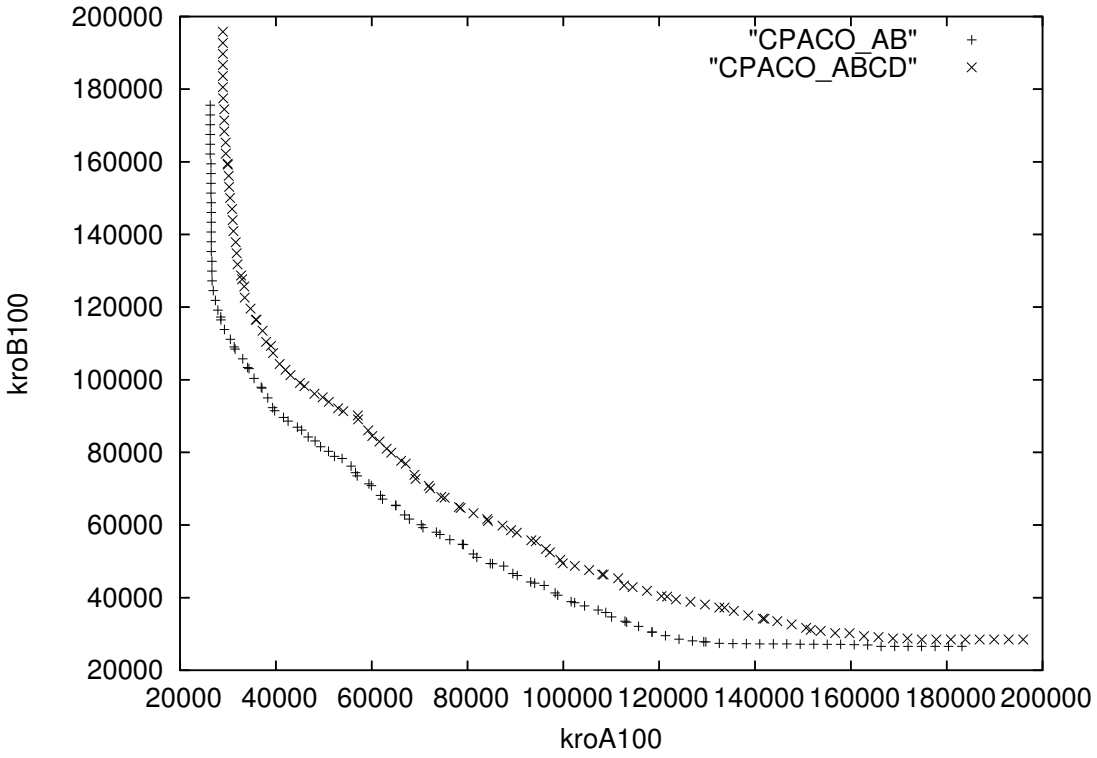


Figure 7.18: 50% (average) attainment surface for CPACO-MOTSP applied to kroA100, kroB100, kroC100 & kroD100 (*CPACO_ABCD*), and CPACO-MOTSP applied to only kroA100 & kroB100 (*CPACO_AB*)

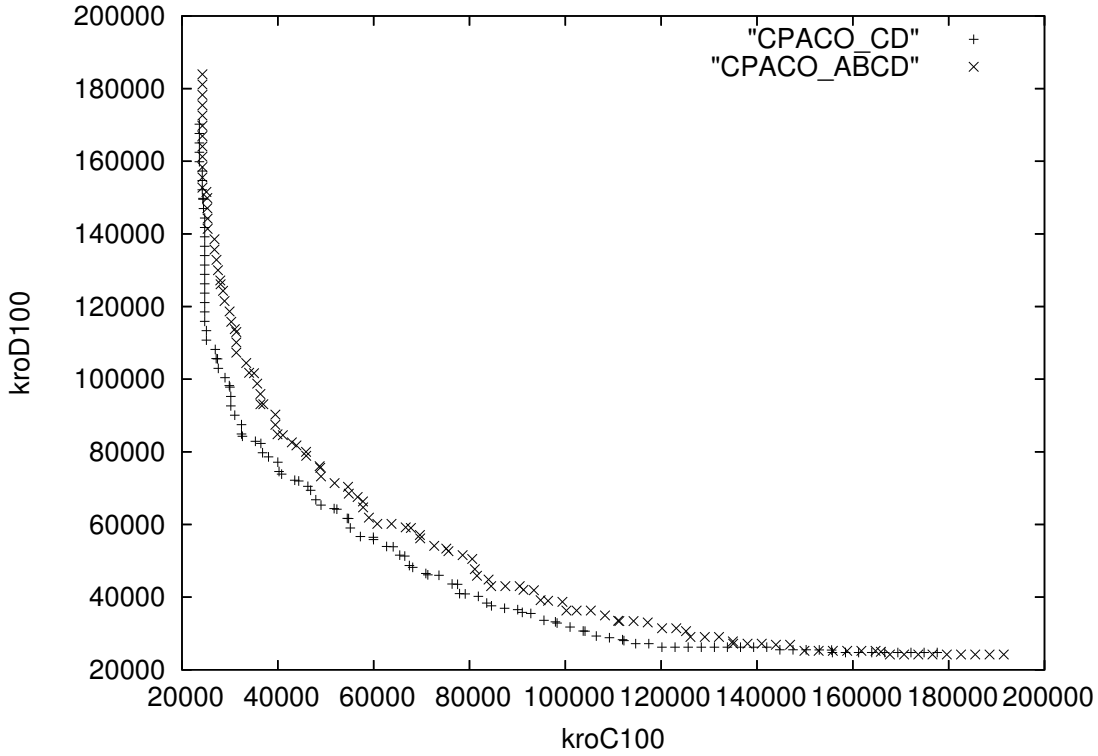


Figure 7.19: 1% (best) attainment surface for CPACO-MOTSP applied to kroA100, kroB100, kroC100 & kroD100 (*CPACO_ABCD*), and CPACO-MOTSP applied to only kroC100 & kroD100 (*CPACO_CD*)

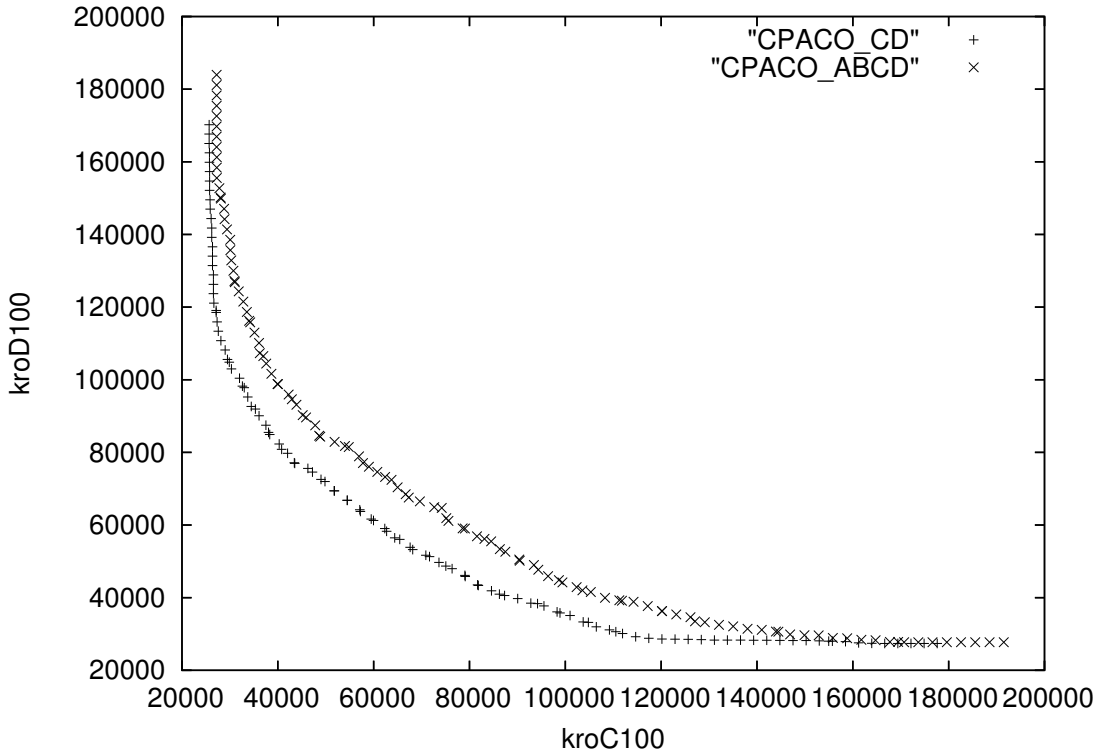


Figure 7.20: 50% (average) attainment surface for CPACO-MOTSP applied to kroA100, kroB100, kroC100 & kroD100 (*CPACO_ABCD*), and CPACO-MOTSP applied to only kroC100 & kroD100 (*CPACO_CD*)

MOTSP algorithm do little to improve the quality of the solutions obtained. It could be assumed then that the algorithm has converged near 100,000 iterations and it would be better to allow a local search to take over rather than continuing to run the CPACO-MOTSP algorithm. As far as the quality of the results is concerned, when isolating the quad-objective results in both the kroA100 & kroB100 and kroC100 & kroD100 objectives the original bi-objective attainment surfaces are better. This is not unexpected given that the quad-objective case is a much more difficult problem due to the added complexity of two extra objectives. It is interesting that the major difference occurs in the mid-point of the approximate Pareto front which suggests that CPACO-MOTSP may not be making best use of its historic (pheromone) information. This is likely a trade-off of using the entire population in the creation of new solutions.

Computational Efficiency

The CPACO-MOTSP algorithm is comparably, if not less, computationally complex than the PACO-MO algorithm. CPACO-MOTSP reconstructs the pheromone matrix every iteration like PACO-MO reconstructs its pheromone matrices (one per objective) and, although CPACO-MOTSP includes more information in the matrix since it uses the entire population rather than a subset, this is offset by the fact that CPACO-MOTSP constructs many more solutions from the pheromone matrix per iteration than the PACO-MO algorithm. PACO-MO also requires the identification of k closest neighbours, an operation which has a worst case complexity of $O(N^2)$ where N is the population size. Using the parameters from this study for the KroA100KroB100 test problem:

- CPACO-MOTSP constructs 50 solutions per matrix which is composed of 50 solutions
- PACO-MO constructs 1 solution per set of matrices which are composed of 6 solutions

Using these parameters, PACO-MO performs pheromone matrix modifications on twice the number of pheromone matrices, six times more frequently than CPACO-MOTSP.

PACO-MO does not require dominance ranking every iteration like CPACO-MOTSP since all solutions are assigned a uniform pheromone update value. However, since the NSGA-II ranking procedure is used, the worst case complexity of this ranking procedure is $O(hN^2)$ where h is the number of objectives. PACO-MO does use the average-rank-weight method to assign ranks to solutions which is of approximate worst-case complexity $O(hN^2)$, which is comparable.

When checking whether to insert a new solution into the population, PACO-MO performs a non-dominance check of worst-case complexity $O(hN)$. If the new solution is non-dominated by the population then it is inserted and the population is checked for non-dominance by the new solution with a worst case complexity of $O(hN^2)$. CPACO-MOTSP selects a subset (in this case 0.1 of N) of the population and uses a crowding comparison to find the closest subset member to the new solution and then performs a single non-dominance check, of total complexity $O((N/10)^2 + h)$.

The observed average final population size during experimentation was larger for PACO-MO than the static population size used for CPACO-MOTSP (Tab. 7.3). Even though these figures

indicate the average final population size, it was observed that the population size grew rapidly at the start of algorithm execution before fluctuating about the final recorded level. As noted earlier the uncontrolled population size was highlighted as an issue for future development for PACO-MO.

Table 7.3: Average final population size of PACO-MO versus static population size of CPACO-MOTSP for all test problems

Problem	PACO-MO final pop size (avg)	CPACO-MOTSP pop size
KroA100/KroB100	105	50
KroA150/KroB150	135	75
KroA200/KroB200	148	100

To summarise, both PACO-MO and CPACO-MOTSP require the use of a distance sorting routine (albeit for different purposes), although CPACO-MOTSP does so on a subset of the population reducing the computation considerably. PACO-MO maintains a larger population than CPACO-MOTSP, and performs approximately six times as many pheromone updates as CPACO-MOTSP (across multiple pheromone matrices). CPACO-MOTSP uses the non-dominance sorting routine to assign ranks to solutions, while PACO-MO uses the average-rank-weight method to assign objective rankings (both of similar complexity). PACO-MO requires non-dominance checking of the entire population per insertion of a new solution while CPACO-MOTSP only performs one non-dominance check per insertion. PACO-MO maintains a larger population which is on average twice the size of CPACO-MOTSP. PACO-MO thus appears to be a more computationally expensive algorithm than CPACO-MOTSP.

Reasons for Success

The influence of the heuristic is a strong determinant of the performance of the CPACO-MOTSP algorithm. The heuristic filters the historical (pheromone) information so that the probability of selecting edges which are better for specific areas of the Pareto front are boosted.

Without access to heuristic or pheromone information the CPACO-MOTSP algorithm would behave as a random search. Upon introducing heuristic information the probable search space (the search space most likely to be explored) becomes much smaller than the original search space. As heuristics go, the nearest neighbour heuristic for the TSP is a good approximate heuristic which on its own can restrict the size of the search space but can't guarantee an optimal solution. This heuristic in combination with historic search information (i.e., a meta-heuristic) has been proven to be a good technique at locating near-optimal solutions.

In the case of the multiple-objective TSP many ant-inspired techniques use multiple histories (usually one for each objective) as was mentioned in Sec. 7.2. When determining the next component to be selected they combine all heuristic and historic information using exponential weightings much like in CPACO-MOTSP. It is suggested that in the case of the MO-TSP the use of one history versus multiple histories (for each objective) does not reduce the performance of the algorithm. To test this consider a test case, the KroA100KroB100 problem.

Test case: multiple versus single pheromone matrices The reasons for wanting to maintain multiple pheromone matrices is to ensure that when constructing a solution at a specific area of the approximate Pareto Front only local solutions are used in biasing solution construction. The basic premise is that a stored solution which trades-off against objective A to advantage objective B would probably not be a good solution to use in constructing a solution which tries to optimise objective A over objective B. This multiple pheromone matrix strategy assumes that an even mixture of solutions from objective A and objective B will allow good solution construction at the midpoint (that is the 50/50 trade-off point) of these objectives, which may not be true. Storing solutions' quality information in a single pheromone matrix which are good solutions for the entire extent of the approximate Pareto Front means that when targeting the midpoint of two objectives there should be a good representative solution for this area of the approximate Pareto Front. This means that the assumption that a mixture of historic solutions from objective A and objective B will provide good solutions at the midpoint does not have to be true for good performance.

To investigate this further, consider two Pareto optimal solutions that are located at opposite ends of the Pareto front of the KroA100KroB100 MO-TSP, i.e., the nadir points. The solution satisfying objective A is denoted solA, and the other solution solB.

An ant is heuristically weighted 10%/90% to objective A and objective B respectively and is initialised at city 5. While all cities connecting to city 5 receive an initial amount of pheromone (≈ 0.01) four specific components also receive an extra unit (1) of pheromone since these components are included in SolA and SolB, being cities 43 and 86 and cities 53 and 62 respectively. Without a heuristic (using only the pheromone) the probability weightings from city 5 would resemble something like Fig. 7.21.

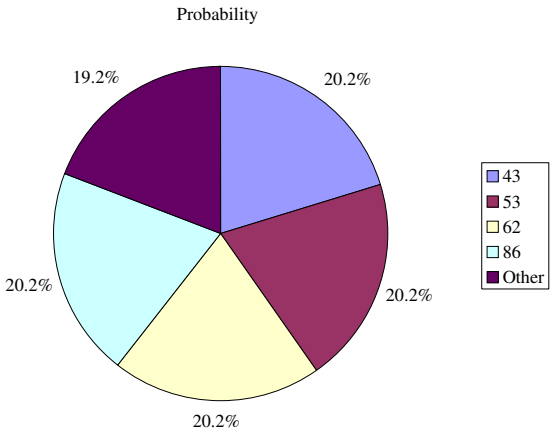


Figure 7.21: Selection probabilities from city 5 using only pheromone information.

However as this ant constructs a solution it is desirable that it pays more attention to the solutions components that are biased by solB, since it has been weighted 10%/90%. After considering heuristics the probability weightings are closer to that of Fig. 7.22, and as can be seen the heuristic has ensured that cities 62 and 53 have a strong selection bias.

While this is a very simple test case, it does illustrate the strong influence that the heuristic has on the selection probabilities. In the case of the MO-TSP the heuristic could be considered

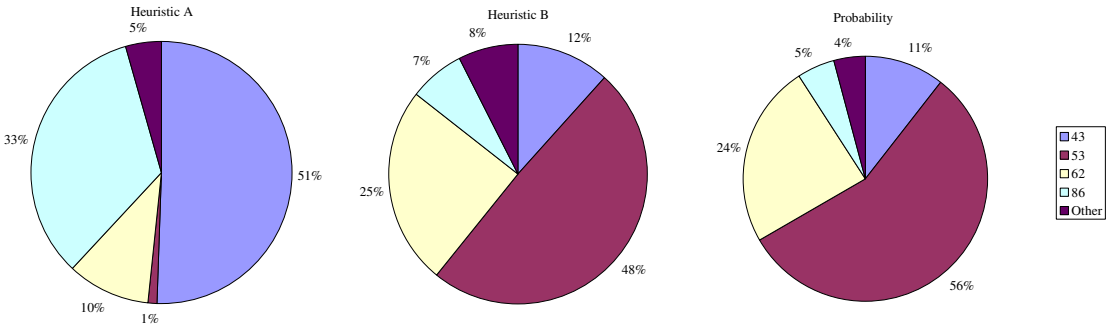


Figure 7.22: Selection probabilities from city 5 using a combination of pheromone information and biased heuristic information (10%/90%).

as a 'filter' that boosts pheromone information that is relevant while eliminating pheromone information which isn't. To split the pheromone information up amongst the objectives seems unnecessary since the heuristic simply filters out unwanted pheromone information anyway.

7.4.4 Summary

In this section the CPACO-MOTSP algorithm was introduced and tested on several benchmark multiple objective TSP. The results for the bi-objective TSP speak well for the CPACO-MOTSP which was able to achieve a better result, in terms of the attainment surface, with less computational complexity than its inspiration, PACO-MO. The CPACO-MOTSP was also tested on a quad-objective TSP which when analysed by isolating two objectives at a time saw a small decrease in the quality of the solutions found. By increasing the number of solution evaluations it was demonstrated that the CPACO-MOTSP algorithm had converged at around 100,000 solution evaluations, meaning that an increase in quality would be best obtained using a local search process on the final population.

The use of a crowding replacement operation was demonstrated to be a good way to control the population size and make-up. The use of a single pheromone matrix was justified by the presence of multiple heuristic matrices which, depending on the specific heuristic weighting of an ant, was able to filter the necessary information from the pheromone matrix.

In the next section a niching PACO algorithm for the Multiple Objective Function Optimisation Problem is introduced. Elements of the CPACO-MOTSP algorithm are used in the design of this algorithm, however, considering the problem lacks heuristic information a different algorithmic approach is required.

7.5 Niching PACO for Multi-objective Function Optimisation

The multiple objective function optimisation (MOFO) problem is often treated as an extension of the single objective continuous function optimisation (CFO) problem. However Deb [36] offers an interesting insight, that the single objective case is actually a specialisation of the multiple objective case. Like the CFO, MOFO involves the optimisation of multiple decision

variables that map, via multiple objective functions, to multiple objective values. The problems may require the minimisation or maximisation of these objectives and the exact mapping between the decision variables and the objective space is usually non-linear.

As the CFO seems to be most closely associated with Evolutionary Algorithms such as the Genetic Algorithm, most early attempts at solving complex MOFO problems tended to be with Multi-objective Evolutionary Algorithms such as those introduced in Sec. 7.2. These algorithms offer a benchmark of good performance and as such one of these algorithms, NSGA-II has been selected as a control algorithm in this analysis.

Three experiments are performed and observations that are relevant to those specific experiments are included. More general comments about generic algorithm behaviour and areas of likely improvement are reserved for the final summary. The intention in this section is not to derive an algorithm that outperforms the control, since on the benchmark problems included in this section the control algorithm is near optimal anyway. The intention rather is to extend some of the ideas from CPACO-MOTSP (Sec. 7.4) to observe how they apply to a multiple objective problem domain that does not include reliable heuristics.

The experiments included are in no way exhaustive, however care has been taken to include problems with features that do test the algorithms' ability to obtain multiple diverse Pareto optimal solutions. The experiments are organised into three sections based on the combination of decision variables and objective functions, these experiments are:

1. Two decision variables, two objectives.
2. Multiple decision variables, two objectives.
3. Two decision variables, multiple objectives.

7.5.1 Algorithm Details

PACO-MOFO is an extension of CPACO-CFO that incorporates similar design elements as those in CPACO-MOTSP. Given the problem domain, a pheromone matrix comprised of discrete values is not required, instead Probability Density Functions are used in a similar fashion to the algorithms discussed in Sec. 6.3. The population is modified using a crowding replacement operation. While this crowding replacement operation ensures that the population retains diverse solutions from the objective space, a fitness sharing selection strategy encourages an even sampling of the objective space, details of which can be found in Sec. 7.5.1. Like CPACO-CFO, PACO-MOFO is an *a posteriori* preference articulation method as defined in Sec. 2.5. Algorithm 13 outlines the PACO-MOFO algorithm details.

Niching Methods

As was previously mentioned, PACO-MOFO uses two forms of niching, crowding and fitness sharing, for the population replacement and selection mechanisms. The crowding technique

Algorithm 13 Population-based ACO for Multi-objective Function Optimisation (PACO-MOFO)

```
Initialise population with uniform random solutions (size = number of ants)
while stopping criterion not met do
    Rank population according to NSGA-II fitness ranking procedure
    Set fitness of each population member as inverse of rank
    Adjust fitness of population by applying fitness sharing
    for i = 1 to number of ants do
        Create new empty solution s_i^new
        for j = 1 to number of dimensions do
            Probabilistically select a solution (s) from the population based on the adjusted fitness raised to a history exponent power (fitness^α), using a biased roulette wheel selection strategy with replacement.
            μ = s_j                                                    ▷ Calculate mean
            r = Dimension j's range
            c = (sin(π/2 × remaining evaluations/maximum evaluations))^2
            σ = r × c/6                                                ▷ Calculate standard deviation
            repeat
                s_{i,j}^new = Gaussian weighted (Equ. 6.3.1) random value using calculated σ and μ.
            until s_{i,j}^new is within bounds of dimension j
        end for
        Evaluate new solution s_i^new for all objectives
    end for
    for i = 1 to number of ants do
        Select random subset of solutions S from population
        if s_i^new is better in all objectives (strongly dominates) than closest matching solution from S (closest match in terms of objective space) then
            Replace closest matching solution with s_i^new
        else
            Discard s_i^new
        end if
    end for
end while
```

is the same as that used in CPACO-MOTSP (Sec. 7.4). Used in this way crowding is responsible for managing the amount of stored diversity in the population. For the MO-TSP the use of crowding alone was enough to ensure an even sampling of the entire Pareto front, due to the nature of the Pareto front and also since the selection mechanism incorporated heuristic information which balanced the focus of the search across the entire Pareto front.

For the MOFO some thought must be given as to how to ensure an even sampling of the Pareto front since for this problem there is no heuristic information, nor can we guarantee an evenly distributed, convex Pareto front. To explain further, when a problem has a decision space that maps fairly linearly to the objective space, the crowding replacement encourages the initial population to spread out to encompass the entire Pareto front fairly uniformly, as was demonstrated with the MO-TSP. However, when the mapping is skewed to favour a particular region, it means that a large proportion of initial solutions are located in a specific area of the objective space which can lead to an over-sampling of this area. This behaviour is self-reinforcing since the more sampling that occurs in one area the better those solutions will become in terms of their NSGA-II ranking and thus the more likely they are to continue to be re-used. In other

words if only a subset of solutions in the population are ever selected then this negates the usefulness of having diversity in the population in the first place. In PACO-MOFO fitness sharing has been employed to ensure a balanced sampling of the population.

The fitness sharing distance calculation is based on the proximity of solutions in the objective space and it derates the quality[2] of similar solutions. In this way the best solutions are those that are located close to the Pareto front and which also belong to sparsely populated regions of the objective space. Before applying fitness sharing, all of the extreme maximum and minimum objective values in the current population are used in the determination of normalised objective values for all population members. This is a temporary value which is only used to assist the fitness sharing derating process and is described in Alg. 14, where m is the population size, s_i^j the j^{th} objective value of the i^{th} population member and h the number of objectives. These values are important since they allow for a generalised fitness sharing radius, rather than a problem specific one.

Algorithm 14 Objective Value Normalisation Procedure

1: **for** $i = 1$ to m **do**
2: **for** $j = 1$ to h **do**
3: $s_i^{j'} = \left(s_i^j - j_{min}\right) / \left(j_{max} - j_{min}\right)$
4: **end for**
5: **end for**

The general rule to determine the fitness sharing radius has been derived to ensure that with the use of fitness sharing the introduction of extra algorithm parameters is kept to a minimum. The rule is derived by approximating the Pareto surface to a linear plane and dividing this plane into sections depending on the size of the population, the ideal case being that the population is distributed evenly across the entire plane. Of course, this is an approximation and cannot account for discontinuities and irregularities in the real Pareto front. The rule is presented in (7.5.1), where m is the population size and h the number of objectives.

$$\text{sharing radius} = \frac{\sqrt{h}}{m^{1/h} - 1} \tag{7.5.1}$$

7.5.2 Test Problems

The problems selected for testing the proposed algorithm are well documented benchmark test functions each chosen for specific properties such as the shape of the Pareto front, and the mapping between the decision space and the objective space. They are the same as those used in Van Veldhuizen's PhD Dissertation [158], and the naming convention from that source is reused, although the primary sources are included.

[2]Quality here is the rank assigned by the NSGA-II ranking procedure.

MOP1

Schaffer's two objective function [135] is a historically significant test function. It has a one dimensional decision space and two objectives. It is usually unbounded or defined over large bounds, in this work the bounds were set to: $-10^5 \leq x \leq 10^5$.

$$\begin{aligned} f_1(x) &= x^2 \\ f_2(x) &= (x-2)^2 \end{aligned} \tag{7.5.2}$$

MOP2

Fonseca's two objective function [63] is a useful test problem since it allows arbitrary scaling of the number of decision variables without changing the shape of the Pareto front which is continuous and concave. The decision variable space is defined for each dimension as: $-4 \leq x_i \leq 4; i = 1, 2, \ldots, n$

$$\begin{aligned} f_1(\vec{x}) &= 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i - \frac{1}{\sqrt{n}}\right)^2\right) \\ f_2(\vec{x}) &= 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i + \frac{1}{\sqrt{n}}\right)^2\right) \end{aligned} \tag{7.5.3}$$

MOP3

Poloni's two objective function [119] contains two decision variables and is a maximisation problem that has two discontinuous Pareto fronts. Its decision space is bounded as: $-\pi \leq x, y \leq \pi$

$$\begin{aligned} f_1(x,y) &= -\left[1 + (A_1 - B_1)^2 + (A_2 - B_2)^2\right] \\ f_2(x,y) &= -\left[(x+3)^2 + (y+1)^2\right] \\ A_1 &= 0.5 \times \sin(1) - 2.0 \times \cos(1) + \sin(2) - 1.5 \times \cos(2) \\ A_2 &= 1.5 \times \sin(1) - \cos(1) + 2 \times \sin(2) - 0.5 \times \cos(2) \\ B_1 &= 0.5 \times \sin(x) - 2.0 \times \cos(x) + \sin(y) - 1.5 \times \cos(y) \\ B_2 &= 1.5 \times \sin(x) - \cos(x) + 2\sin(y) - 0.5 \times \cos(y) \end{aligned} \tag{7.5.4}$$

MOP4

Kursawe's two objective function [98] is comprised of three disconnected Pareto fronts which map to multiple discontinuous and non-symmetric areas of the decision space. Like MOP2 the number of decision variables is unspecified, however, changing the number of decision

variables does affect the shape and position of the Pareto front. The decision variable space is defined for each dimension as: $-5 \leq x_i \leq 5; i = 1, 2, \ldots, n$

$$\begin{aligned} f_1(\vec{x}) &= \sum_{i=1}^{n-1} \left(-10 \exp\left(-0.2 \times \sqrt{x_i^2 + x_{i+1}^2} \right) \right) \\ f_2(\vec{x}) &= \sum_{i=1}^{n} \left(|x_i|^{0.8} + 5 \sin(x_i)^3 \right) \end{aligned} \tag{7.5.5}$$

MOP5

Viennet's third MOFO problem [159] is a three objective problem that contains a single complex Pareto front in three dimensions. The decision space that corresponds with this Pareto front is disconnected. Its decision space is bounded as: $-3 \leq x, y \leq 3$

$$\begin{aligned} f_1(x,y) &= 0.5 \times \left(x^2 + y^2 \right) + \sin\left(x^2 + y^2 \right) \\ f_2(x,y) &= \frac{(3x - 2y + 4)^2}{8} + \frac{(x - y + 1)^2}{27} + 15 \\ f_3(x,y) &= \frac{1}{(x^2 + y^2 + 1)} - 1.1 \exp\left(-x^2 - y^2 \right) \end{aligned} \tag{7.5.6}$$

MOP6

This problem was proposed by Deb [35] and is a two objective, two decision variable problem with four discontinuous Pareto fronts. The decision variable space is defined as: $0 \leq x, y \leq 1$

$$\begin{aligned} f_1(x,y) &= x \\ f_2(x,y) &= (1 + 10y) \times \left[1 - \left(\frac{x}{1 + 10y} \right)^2 - \frac{x}{1 + 10y} \sin(8\pi x) \right] \end{aligned} \tag{7.5.7}$$

MOP7

Viennet's second MOFO problem [159] is a three objective problem that contains a single Pareto front in three dimensions. The decision space that corresponds with this Pareto front is connected and the mapping is simple. Its decision space is bounded as: $-4 \leq x, y \leq 4$

$$\begin{aligned} f_1(x,y) &= \frac{(x - 2)^2}{2} + \frac{(y + 1)^2}{13} + 3 \\ f_2(x,y) &= \frac{(x + y - 3)^2}{36} + \frac{(-x + y + 2)^2}{8} - 17 \\ f_3(x,y) &= \frac{(x + 2y - 1)^2}{175} + \frac{(-x + 2y)^2}{17} - 13 \end{aligned} \tag{7.5.8}$$

7.5.3 Results and Discussion

Experiment 1: Testing With Two Decision Variables and Two Objectives

This section outlines an analysis of the PACO-MOFO algorithm applied to several two objective, two decision variable MOFO problems. Attainment surface comparison is used to determine the performance of the PACO-MOFO algorithm as compared to the NSGA-II.

The MOP1, MOP2, MOP3, MOP4 and MOP6 problems described in Sec. 7.5.2 are used in this analysis and both PACO-MOFO and NSGA-II were run 50 times on each of the test problems defined with two decision variables and two objectives (as some of the problems can be configured otherwise). Each algorithm was allowed 100,000 solution evaluations (as this was enough evaluations to observe algorithm convergence) and the final population was recorded. The algorithm configuration parameters were the same for all trials and are reproduced in Tab. 7.4. These parameters were chosen based on previous experimentation with similar algorithms such as CPACO-MOTSP, CPACO-CFO and FSPACO-CFO. The best and average attainment surfaces obtained for the five test problems are shown in Fig. 7.23, 7.24, 7.25, 7.26 & 7.27.

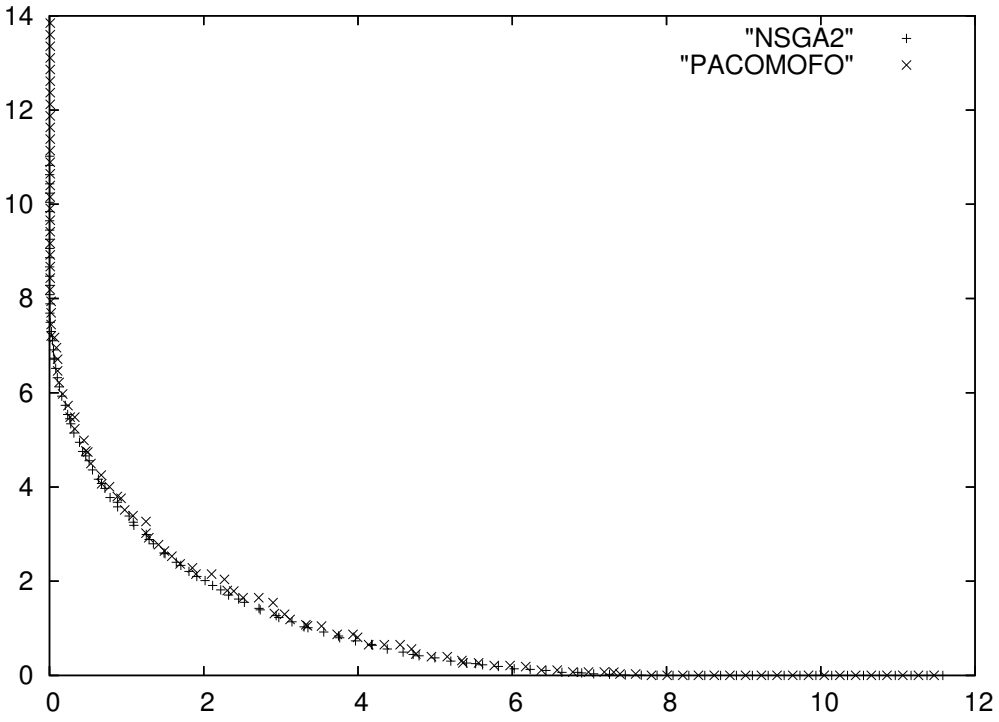
NSGA-II	
Parameter	**Value**
Population size	50
Crossover Probability	0.97
Mutation Probability	0.50
Std. Dev. of Gaussian Mutation	1% of dimension range
PACO-MOFO	
Parameter	**Value**
Number of ants (m) / Population size	50
History Exponent	1.0
Crowding Window Size	0.5
Fitness Sharing Radius (h = objectives)	$1/\left((m)^{1/h}-1\right)$
Fitness Sharing Power	1.0

Table 7.4: Algorithm parameter settings for MOFO problems

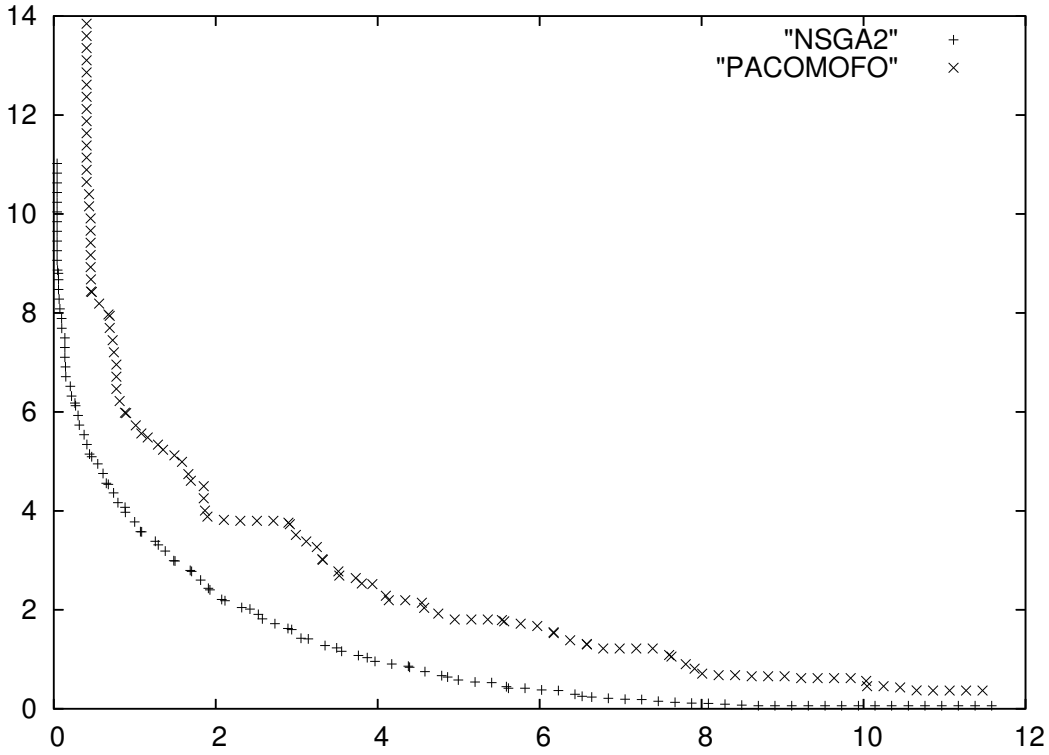
Problem	Metric		
	Attainment surface	Hypervolume	Epsilon indicator
MOP1	< (0)	< (0)	< (0)
MOP2	=	< (0)	< (0)
MOP3	=	> (0)	=
MOP4	=	< (0)	< (0)
MOP6	=	> (0.01)	> (0.02)

Table 7.5: Results of comparison between PACO-MOFO and NSGA-II using the summary attainment surface, hypervolume and epsilon indicator metrics. Indicators used indicate if PACO-MOFO is significantly better than NSGA-II (>), if there is no significant difference (=), or if NSGA-II is significantly better than PACO-MOFO (<). The statistical confidence of the result (p value) is also indicated. The Kruskal-Wallis test was used for summary attainment surface comparisons and the Mann-Whitney Rank-Sum test for all other comparisons.

As is indicated in Tab. 7.5, with the exception of MOP1 (where NSGA-II is better) there is no statistical difference between the attainment surfaces generated for the control algorithm (NSGA-II) and the PACO-MOFO algorithm. While the other metrics indicate differences, any difference

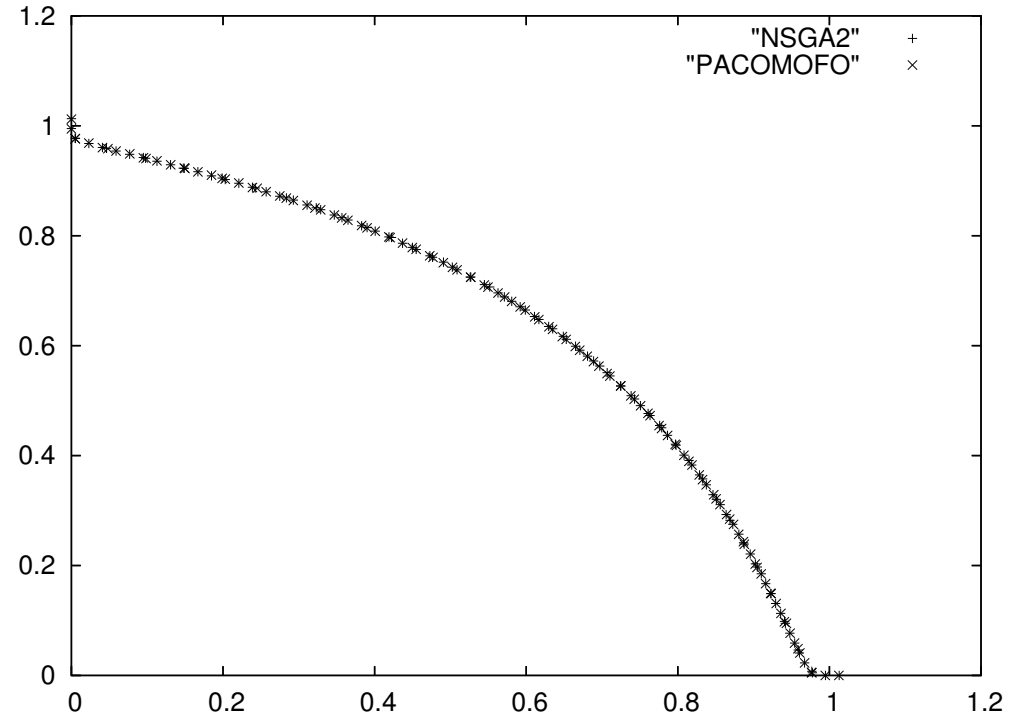


(a) 1% (best) attainment surfaces

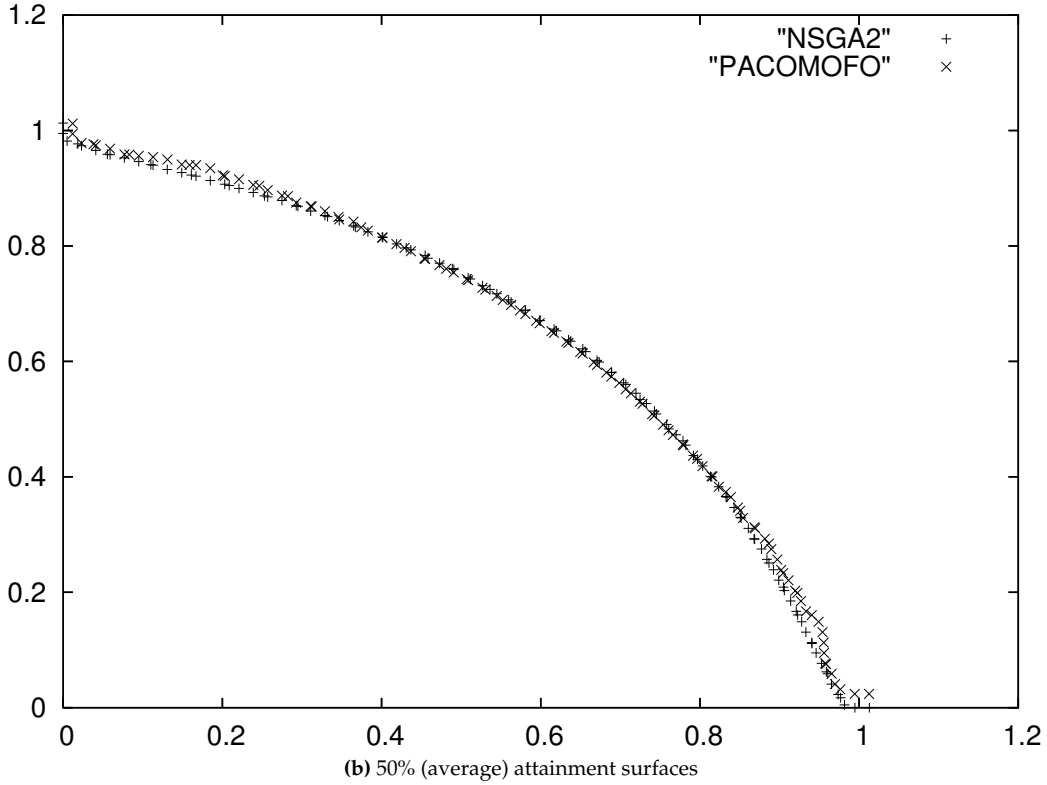


(b) 50% (average) attainment surfaces

Figure 7.23: Attainment surfaces generated from 50 runs of PACO-MOFO and NSGA-II applied to MOP1

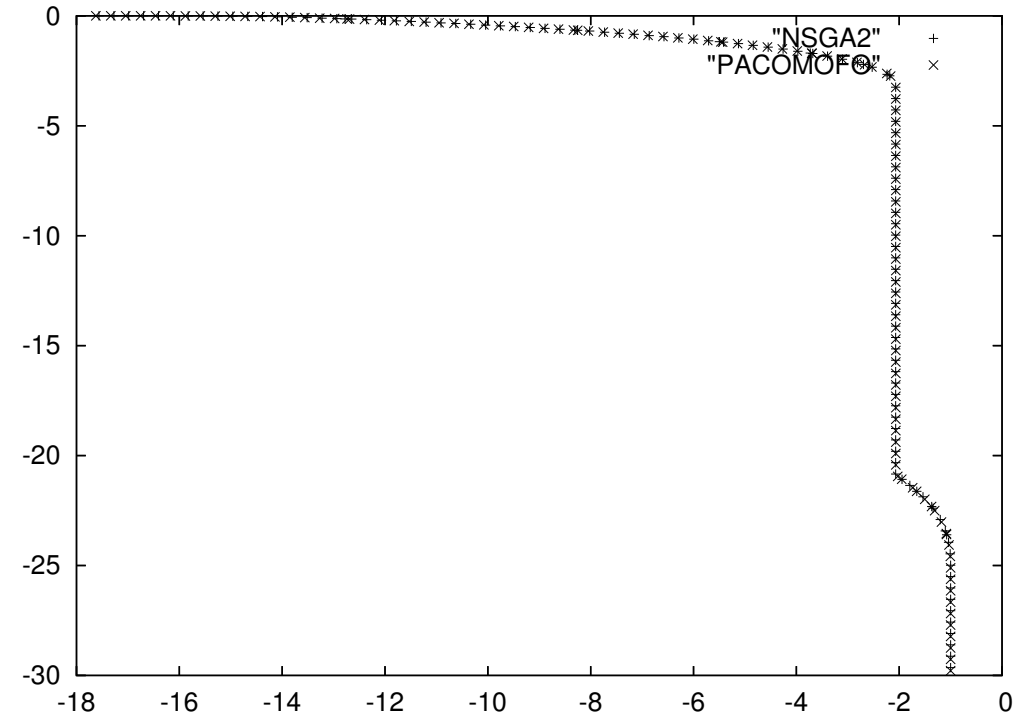


(a) 1% (best) attainment surfaces

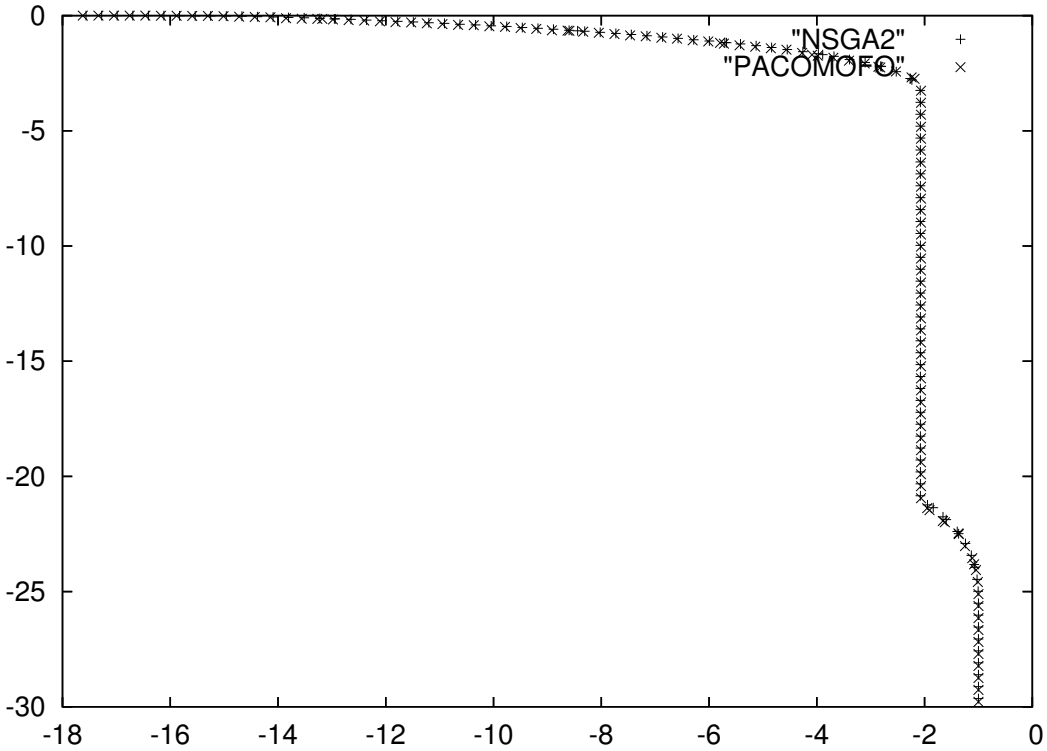


(b) 50% (average) attainment surfaces

Figure 7.24: Attainment surfaces generated from 50 runs of PACO-MOFO and NSGA-II applied to MOP2

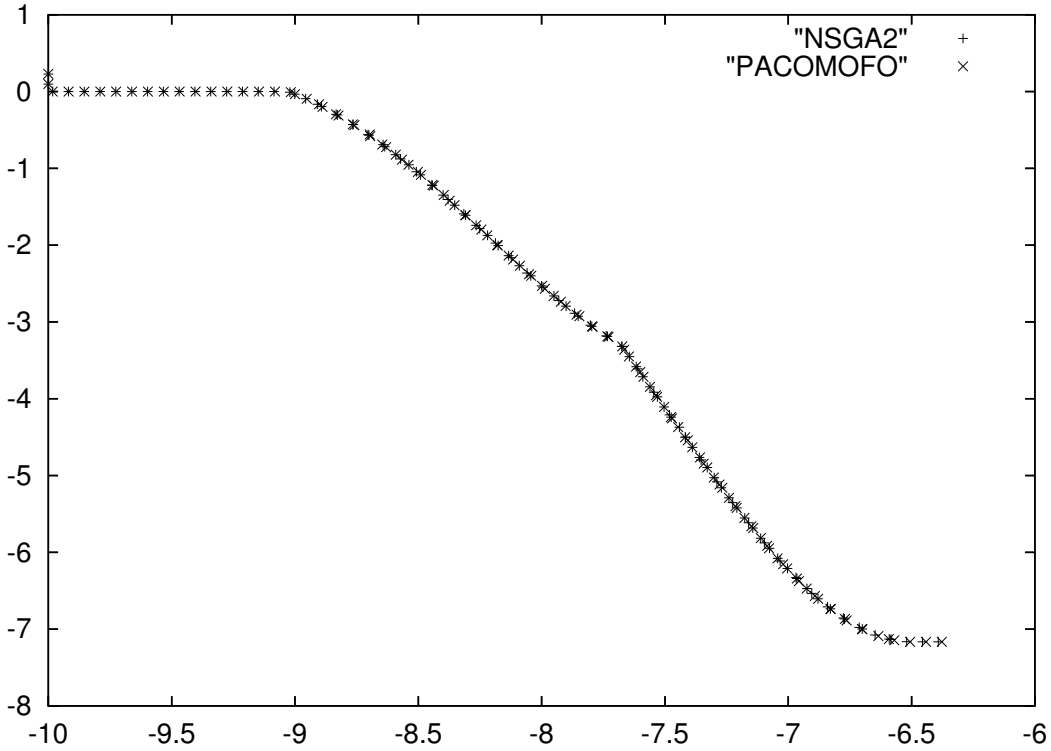


(a) 1% (best) attainment surfaces

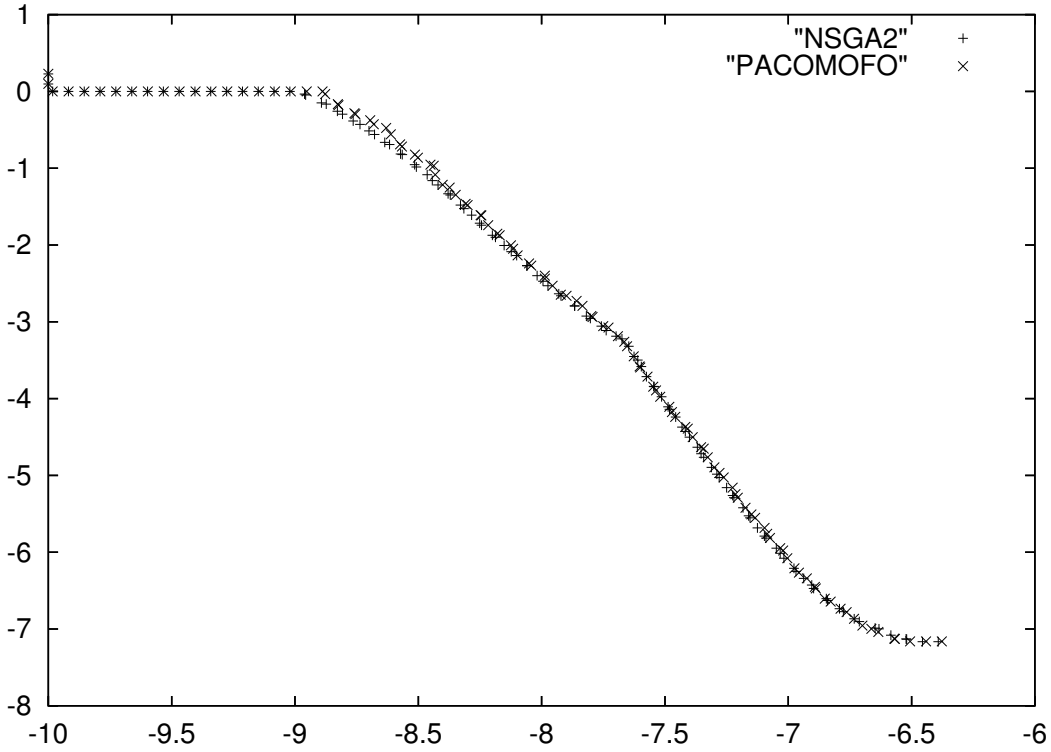


(b) 50% (average) attainment surfaces

Figure 7.25: Attainment surfaces generated from 50 runs of PACO-MOFO and NSGA-II applied to MOP3

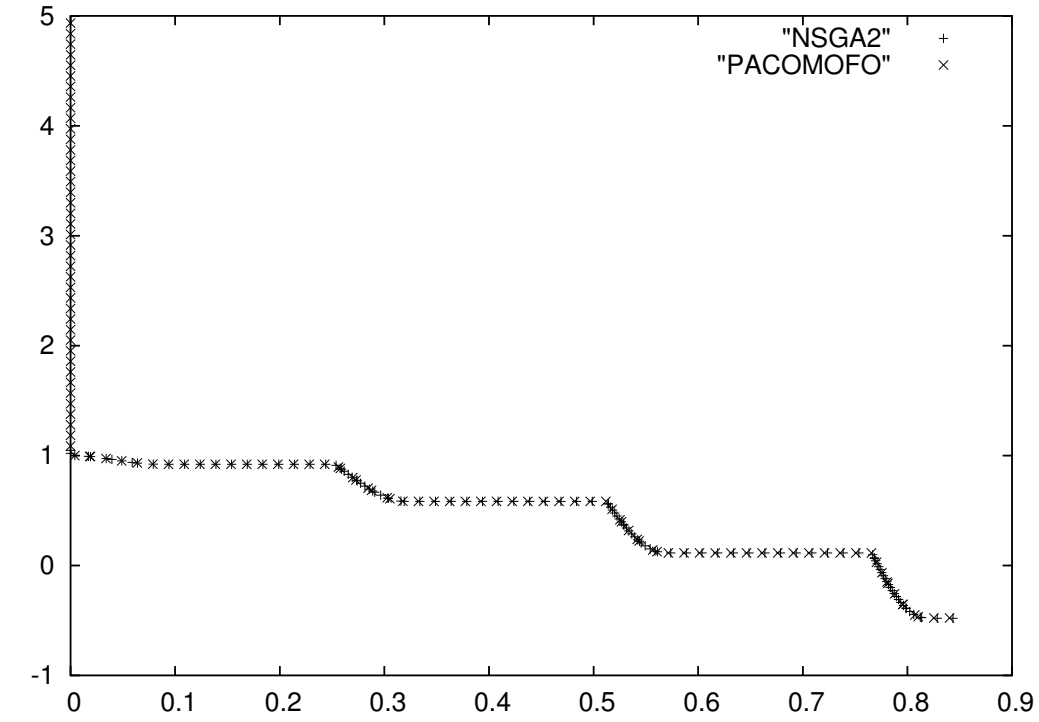


(a) 1% (best) attainment surfaces

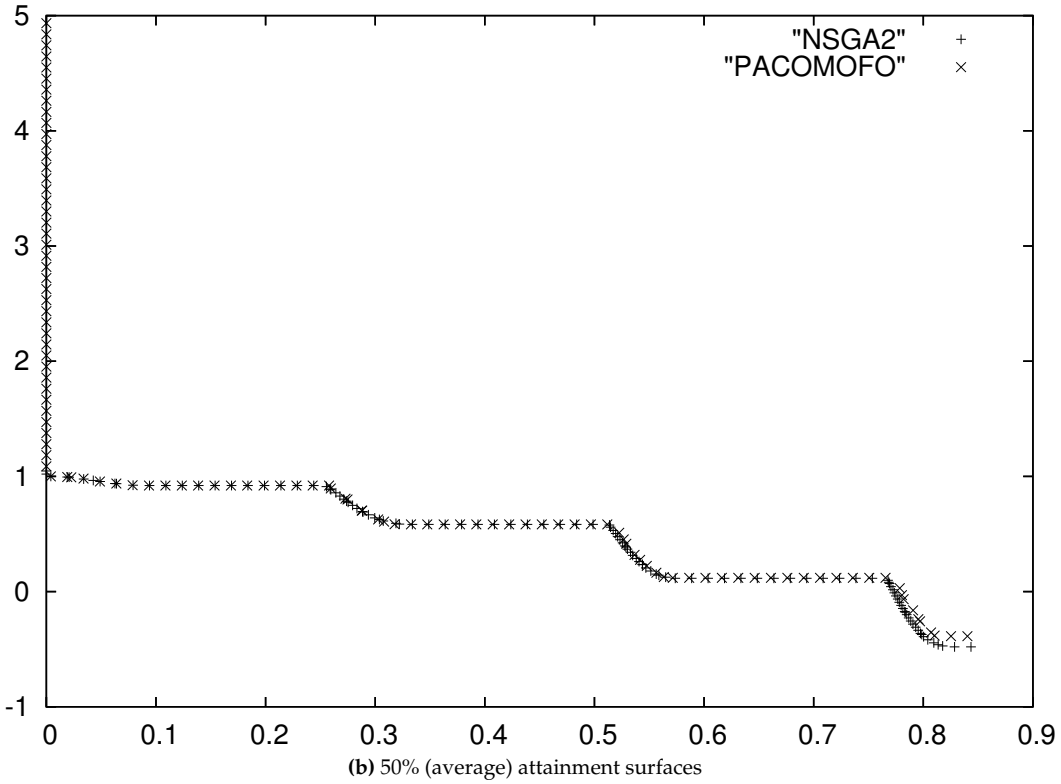


(b) 50% (average) attainment surfaces

Figure 7.26: Attainment surfaces generated from 50 runs of PACO-MOFO and NSGA-II applied to MOP4



(a) 1% (best) attainment surfaces



(b) 50% (average) attainment surfaces

Figure 7.27: Attainment surfaces generated from 50 runs of PACO-MOFO and NSGA-II applied to MOP6

even though it is statistically significant is deemed to be minor given the result of the summary attainment surface comparison. As such these other results are not used in drawing conclusions about the performance of the algorithms. This result is passable for PACO-MOFO as it demonstrates that on these few benchmark instances the PACO-MOFO algorithm performs just below an accepted state-of-the-art MOO algorithm.

Important qualitative assessments of the PACO-MOFO algorithm were also observed. Figure 7.28 indicates typical final populations from individual runs of PACO-MOFO and NSGA-II. As can be seen PACO-MOFO distributes its population fairly evenly across the entire Pareto front, particularly in difficult cases where the Pareto front is concave, or discontinuous. More-so it displays strong convergence to the Pareto front. This behaviour is different to that of NSGA-II whose final population tends to contain more solutions from other non-Pareto optimal areas. This is expected since the crowding replacement routine of PACO-MOFO aims to ensure that the entire population approaches the Pareto front and only allows replacement if a new solution dominates an existing solution. In essence PACO-MOFO is more elitist than NSGA-II.

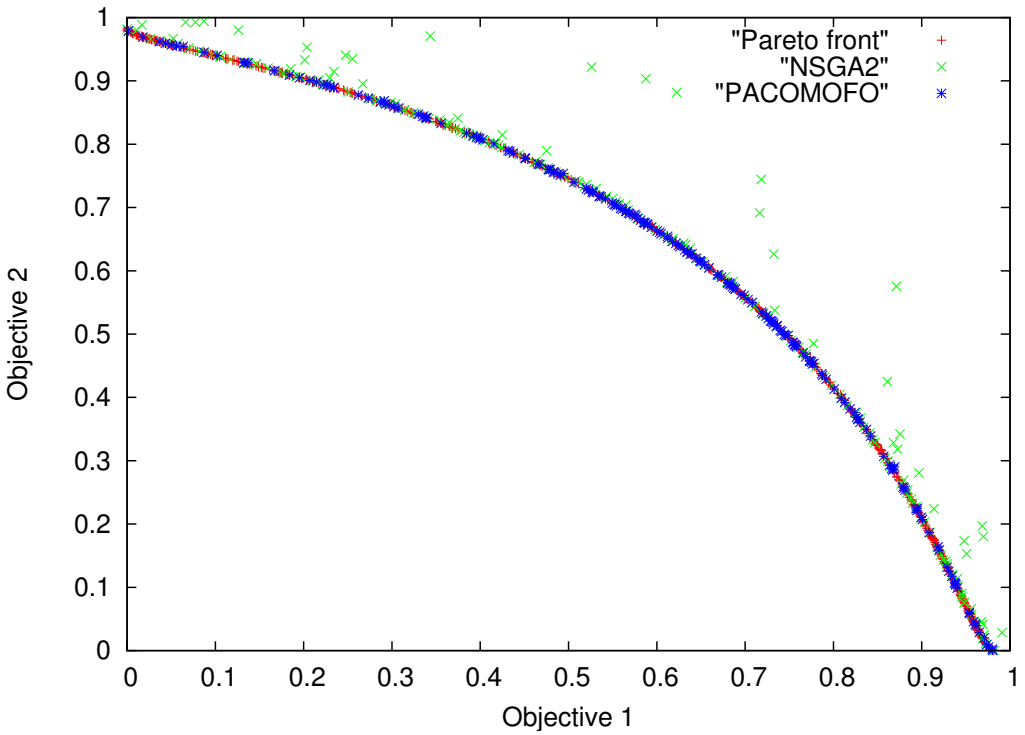
Experiment 2: Testing With More Decision Variables and Two Objectives

The MOP2 and MOP4 problems were included in the test problem suite since they allow arbitrary scaling of the decision space. Scaling MOP2 does not affect the shape or position of the Pareto front, however, scaling MOP4 does. To test the PACO-MOFO algorithms ability to scale (with regard to the number of decision variables) the MOP2 and MOP4 problems were defined in various decision variable dimensions (5,10,15,20) and compared against the NSGA-II algorithm. Each algorithm was allowed 100,000 function evaluations, and was repeated 50 times. The average attainment surfaces generated are included in Fig. 7.29 and Fig 7.30.

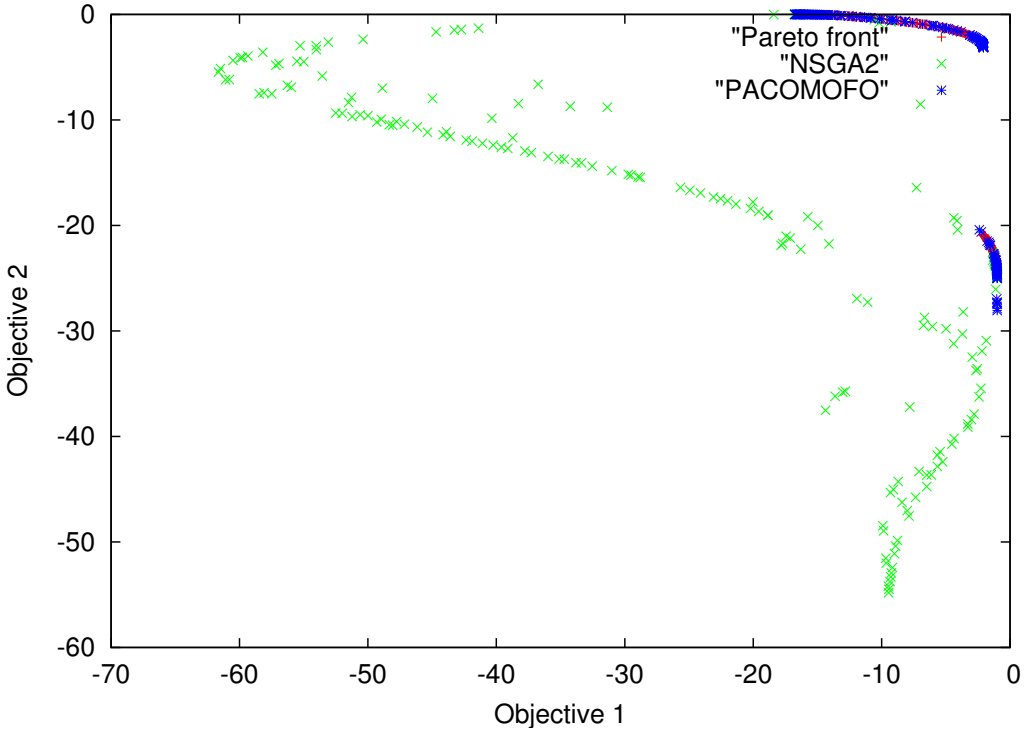
Dimensions	Metric		
	Attainment surface	Hypervolume	Epsilon indicator
2	=	< (0)	< (0)
5	=	< (0)	< (0)
10	=	< (0)	< (0)
15	> (0.04)	< (0)	< (0)
20	> (0)	> (0)	> (0)

Table 7.6: Results of comparison between PACO-MOFO and NSGA-II on MOP2 using the summary attainment surface, hypervolume and epsilon indicator metrics. Indicators used indicate if PACO-MOFO is significantly better than NSGA-II (>), if there is no significant difference (=), or if NSGA-II is significantly better than PACO-MOFO (<). The statistical confidence of the result (p value) is also indicated. The Kruskal-Wallis test was used for summary attainment surface comparisons and the Mann-Whitney Rank-Sum test for all other comparisons.

Perhaps the most interesting, and obvious, observations between the two algorithms is the difference in the solutions found in the centre of the Pareto front where the solutions found by NSGA-II are worse than those found by PACO-MOFO, while at the extremes the converse is true. The MOP2 problem is extremely difficult in higher dimensions due to the lack of directional information (i.e. the objective functions return a value of 1 for most points in the decision variable space). Given that a random initial population will most likely contain many solutions with objective values of (1.0, 1.0), the PACO-MOFO algorithm will approach the Pareto front

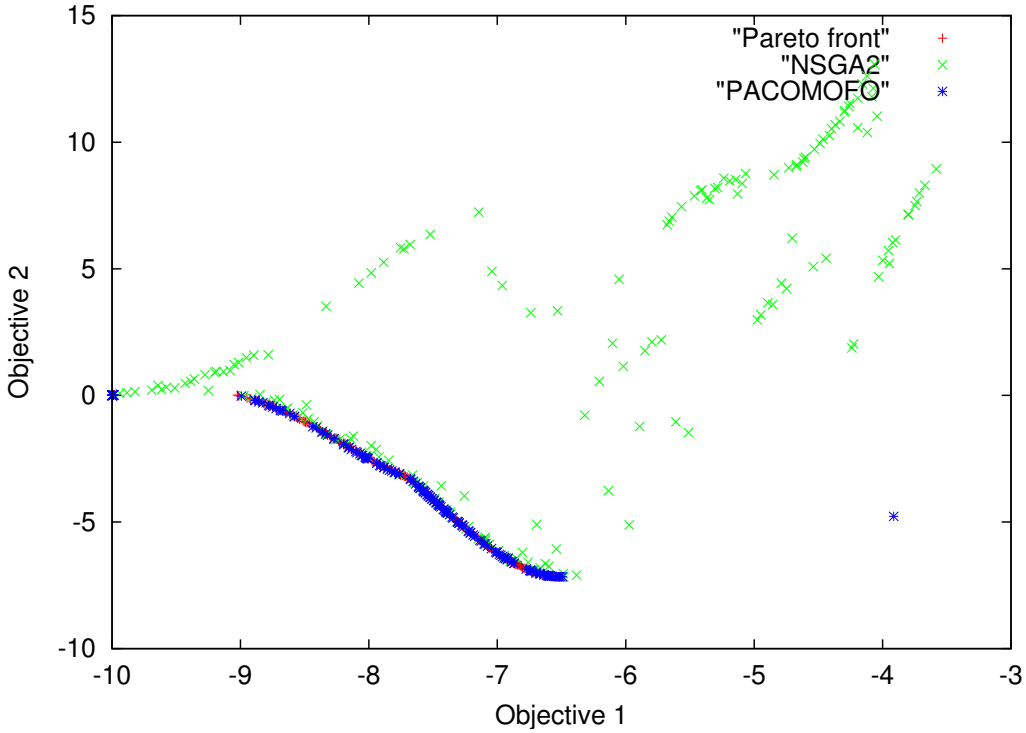


(a) MOP2

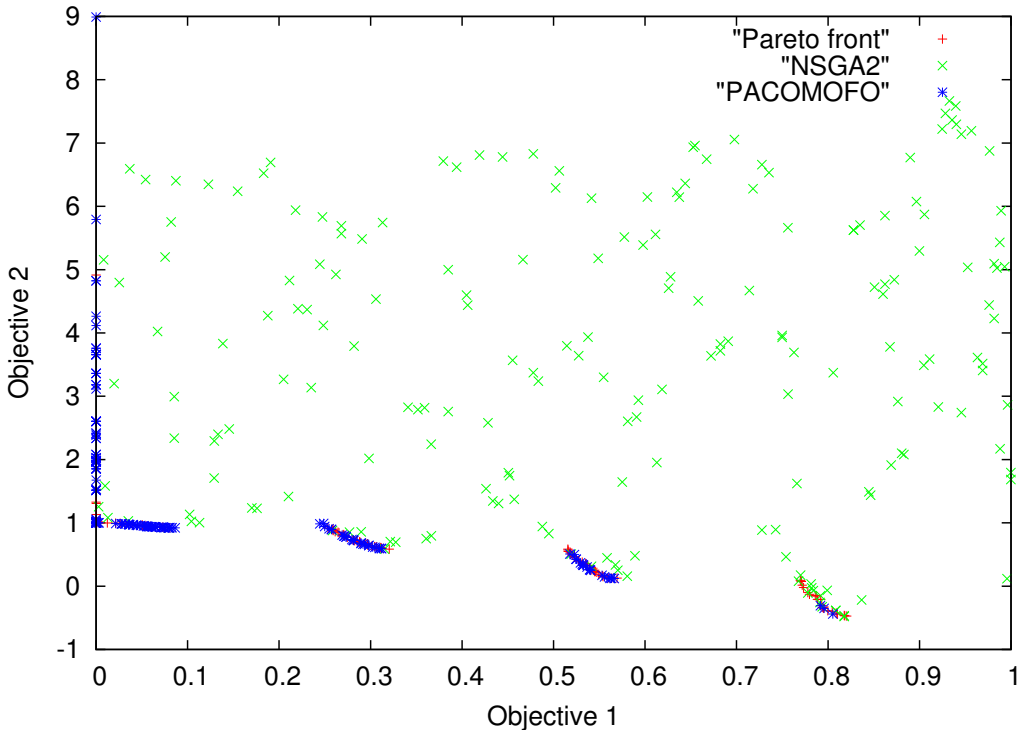


(b) MOP3

Figure 7.28: Example of a typical final population of PACO-MOFO and NSGA-II after each algorithm was allowed 100,000 solution evaluations. The Pareto front was approximated by taking 10,000,000 random evaluations and removing all dominated solutions. Note the strong convergence of the PACO-MOFO algorithm to and along the Pareto front.

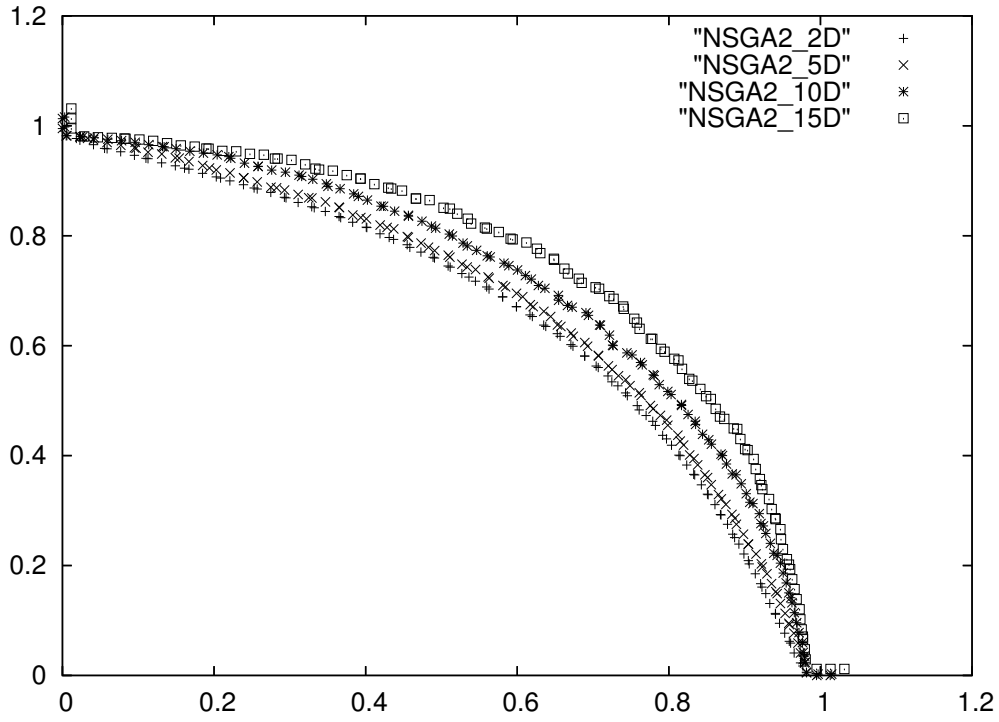


(c) MOP4

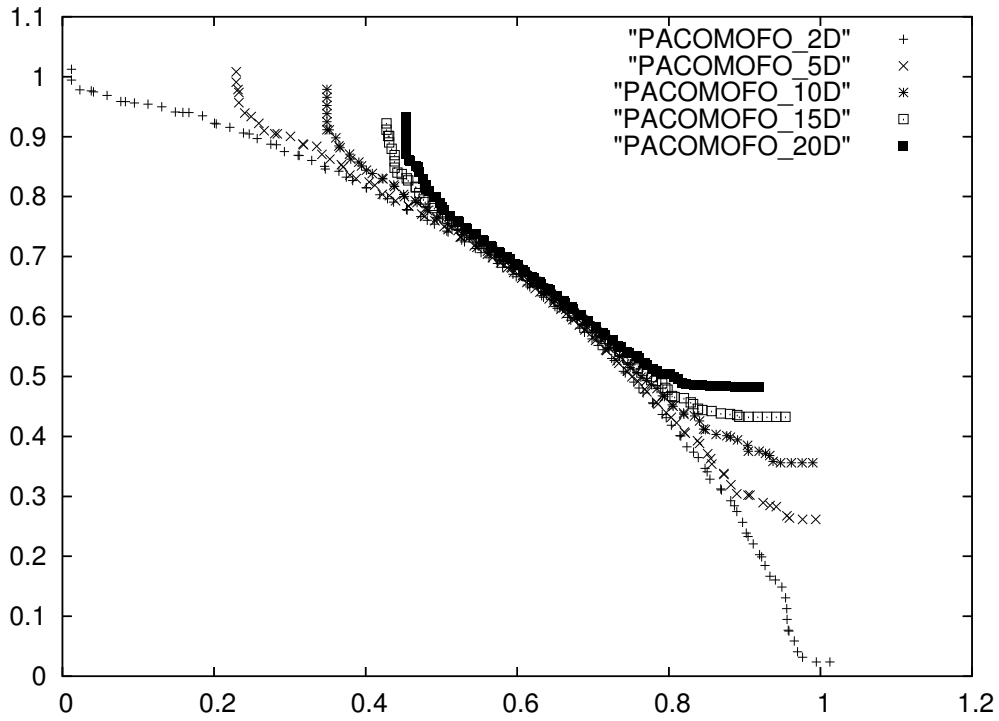


(d) MOP6

Figure 7.28: (cont'd) Example of a typical final population of PACO-MOFO and NSGA-II after each algorithm was allowed 100,000 solution evaluations. The Pareto front was approximated by taking 10,000,000 random evaluations and removing all dominated solutions. Note the strong convergence of the PACO-MOFO algorithm to and along the Pareto front.

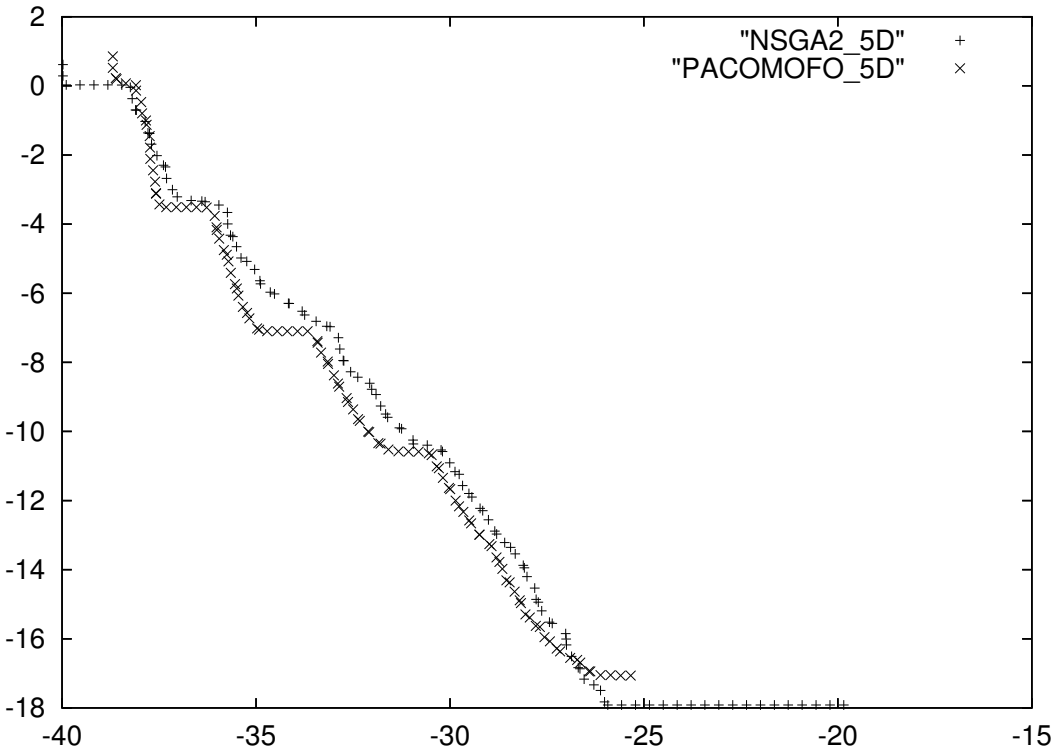


(a) Average attainment surfaces for MOP2 in 2, 5, 10, and 15 dimensions using NSGA-II

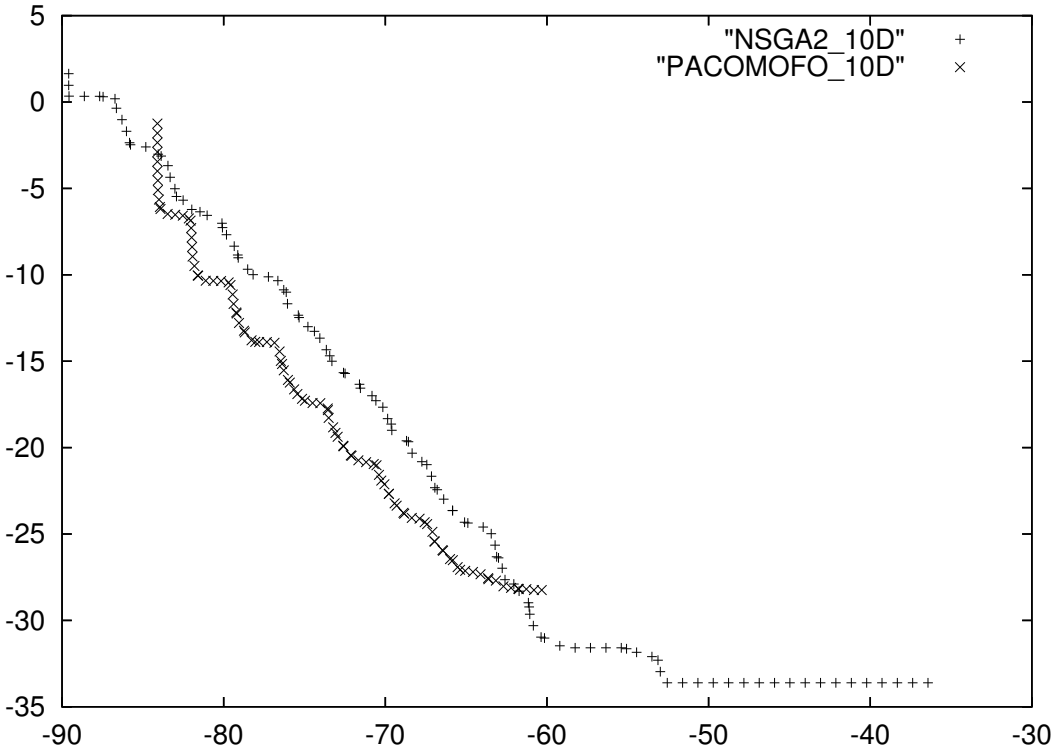


(b) Average attainment surfaces for MOP2 in 2, 5, 10, 15 and 20 dimensions using PACO-MOFO

Figure 7.29: Attainment surfaces generated from 50 runs of PACO-MOFO and NSGA-II applied to the benchmark function, MOP2, in 2, 5, 10, 15 and 20 dimensions

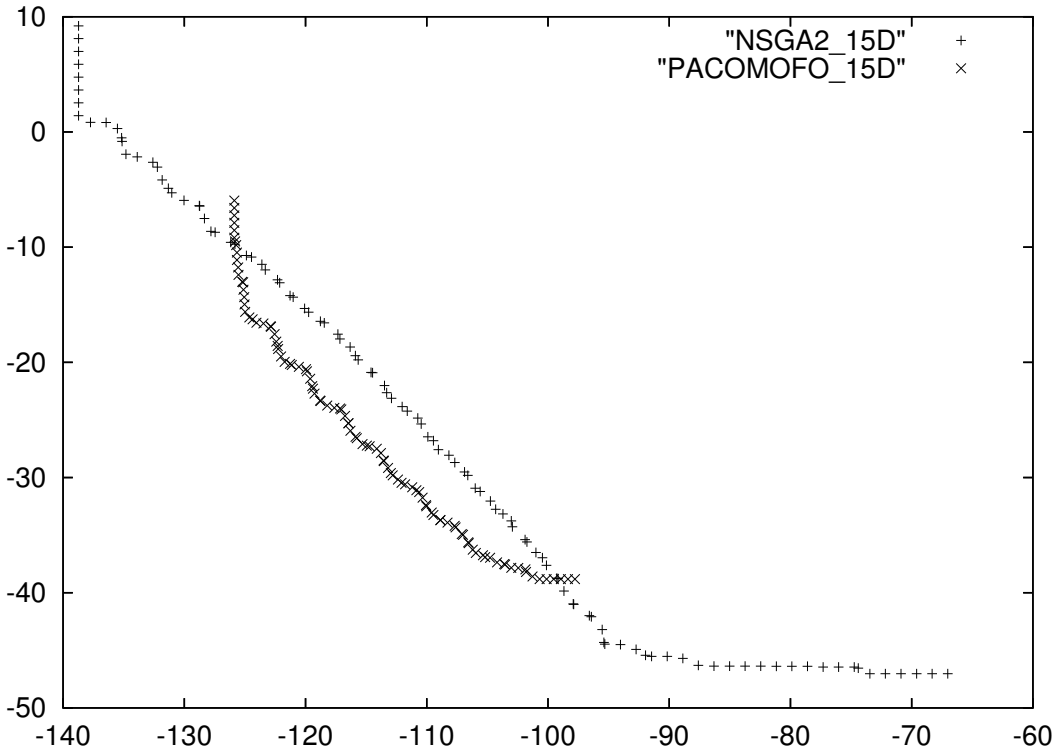


(a) Average attainment surfaces for MOP4 in 5 dimensions using NSGA-II and PACO-MOFO

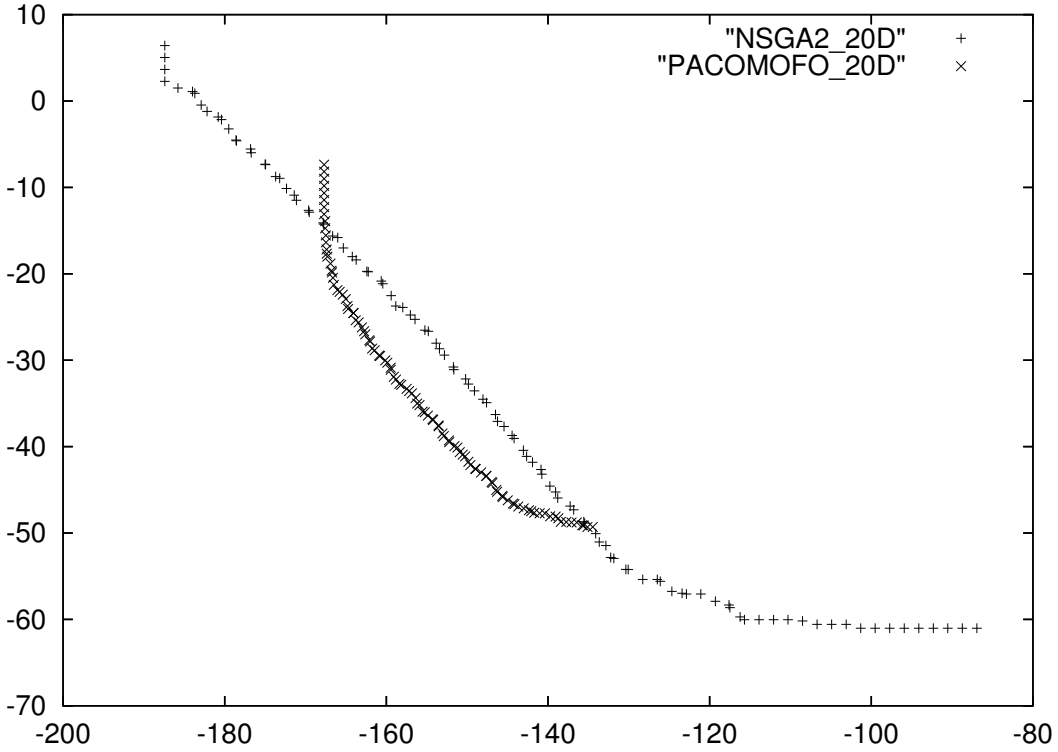


(b) Average attainment surfaces for MOP4 in 10 dimensions using NSGA-II and PACO-MOFO

Figure 7.30: (cont'd) Attainment surfaces generated from 50 runs of PACO-MOFO and NSGA-II applied to the benchmark function, MOP4, in 5, 10, 15 and 20 dimensions



(c) Average attainment surfaces for MOP4 in 15 dimensions using NSGA-II and PACO-MOFO



(d) Average attainment surfaces for MOP4 in 20 dimensions using NSGA-II and PACO-MOFO

Figure 7.30: (cont'd) Attainment surfaces generated from 50 runs of PACO-MOFO and NSGA-II applied to the benchmark function, MOP4, in 5, 10, 15 and 20 dimensions

Dimensions	Metric		
	Attainment surface	Hypervolume	Epsilon indicator
2	=	< (0)	< (0)
5	> (0.006)	=	< (0.04)
10	> (0)	< (0.01)	< (0)
15	> (0)	< (0)	< (0)
20	> (0)	< (0)	< (0)

Table 7.7: Results of comparison between PACO-MOFO and NSGA-II on MOP4 using the summary attainment surface, hypervolume and epsilon indicator metrics. Indicators used indicate if PACO-MOFO is significantly better than NSGA-II (>), if there is no significant difference (=), or if NSGA-II is significantly better than PACO-MOFO (<). The statistical confidence of the result (p value) is also indicated. The Kruskal-Wallis test was used for summary attainment surface comparisons and the Mann-Whitney Rank-Sum test for all other comparisons.

from the top-right corner of Fig. 7.29b. Since a crowding replacement scheme is being used, the PACO-MOFO algorithm will be slow in exploring outward across the extent of the entire Pareto front in a similar way to the problem faced when solving the single objective TSP (Sec. 6.2.3). With the single objective TSP it was demonstrated how many of the initial population members aren't replaced since they are never selected for replacement and thus continue to persist in the population for a long time. This effect explains, to some degree, the convex Pareto fronts generated by PACO-MOFO. Adding to this, the only way a solution can be replaced is if it is strongly dominated, meaning that the population make-up will change quite slowly and may stagnate to some degree. Similar results were obtained when the algorithms were tested on MOP4, however, with this problem it is more difficult to observe the progressive worsening of the results as the Pareto fronts can't be directly compared.

The NSGA-II algorithm does not suffer from the replacement problem observed with PACO-MOFO since it replaces all solutions in the population every iteration and then removes those which are located furthest away from the Pareto front. Since the NSGA-II population is changing far more frequently it means a quicker uptake of solutions from across the entire Pareto front, since old solutions are more likely to be left out of the population over successive generations.

These differences are also highlighted in the statistical comparison which for most problem variants ranks the NSGA-II better in terms of the hypervolume and epsilon indicator metrics, while the PACO-MOFO is deemed better in terms of the summary attainment surface comparison. As such it cannot be said which algorithm is better, since each seems to achieve different goals, the NSGA-II covers the Pareto front more evenly, while the PACO-MOFO algorithm finds better solutions in the centre of the Pareto front to the detriment of diversity. As an aside it was of interest that NSGA-II failed to find any solutions other than those with objective values (1.0, 1.0) when tested on MOP2 above 20 dimensions, while it took PACO-MOFO until 30 dimensions to do the same.

Experiment 3: Testing With Two Decision Variables and Three Objectives

In this section the PACO-MOFO and NSGA-II algorithms are tested on two three-objective MOFO problems, MOP5 and MOP7. This test should give some indication as to how well the PACO-MOFO algorithm performs as the number of objectives is increased. To reiterate each of the MOP5 and MOP7 problems contain two decision variables and a single connected Pareto front. The NSGA-II and PACO-MOFO algorithms were allowed 100,000 function evaluations per experiment run, and were repeated 50 times for statistical significance. The summary attainment surfaces have been included (Fig. 7.31 and Fig. 7.32) however they can't be used for anything but qualitative assessment purposes due to the presence of three objectives. The hypervolume and epsilon indicators are used for quantitative assessment purposes and are included in Tab. 7.8.

Problem	Metric	
	Hypervolume	Epsilon indicator
MOP5	> (0)	> (0)
MOP7	> (0)	> (0)

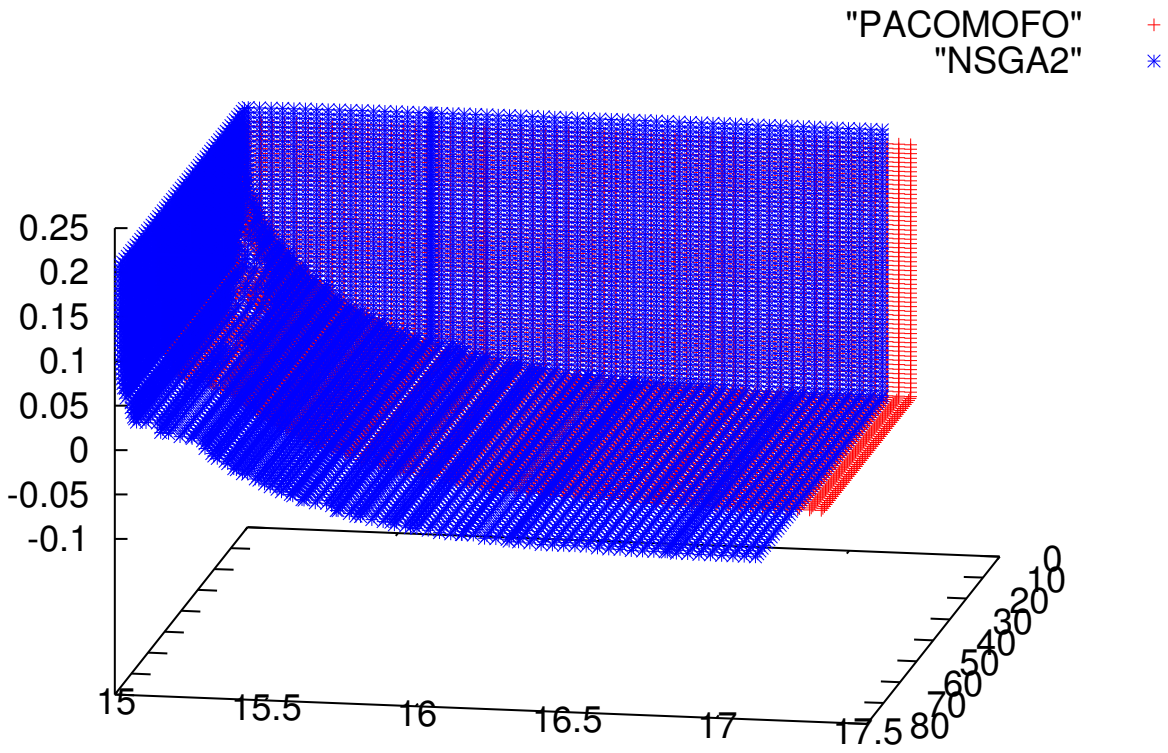
Table 7.8: Results of comparison between PACO-MOFO and NSGA-II using the hypervolume and epsilon indicator metrics. Indicators used indicate if PACO-MOFO is significantly better than NSGA-II (>), if there is no significant difference (=), or if NSGA-II is significantly better than PACO-MOFO (<). The statistical confidence of the result (p value) is also indicated. The Kruskal-Wallis test was used for summary attainment surface comparisons and the Mann-Whitney Rank-Sum test for all other comparisons.

The hypervolume and epsilon indicators indicate that PACO-MOFO achieves a statistically significant better result than NSGA-II on the MOP5 and MOP7 problems. A visual inspection of the attainment surfaces for MOP5 and MOP7 confirms that for the majority of the attainment surfaces plotted PACO-MOFO does achieve a better result than NSGA-II. To interpret these graphs correctly the perspective shown looks towards the origin, since the attainment surface of PACO-MOFO is hidden behind that of NSGA-II it should be clear that the attainment surface of PACO-MOFO is closer to the origin and the true Pareto front. This result, while limited to two simple MOFO problems does speak well for the PACO-MOFO algorithms' ability to attempt the optimisation of more than two objective MOFO problems.

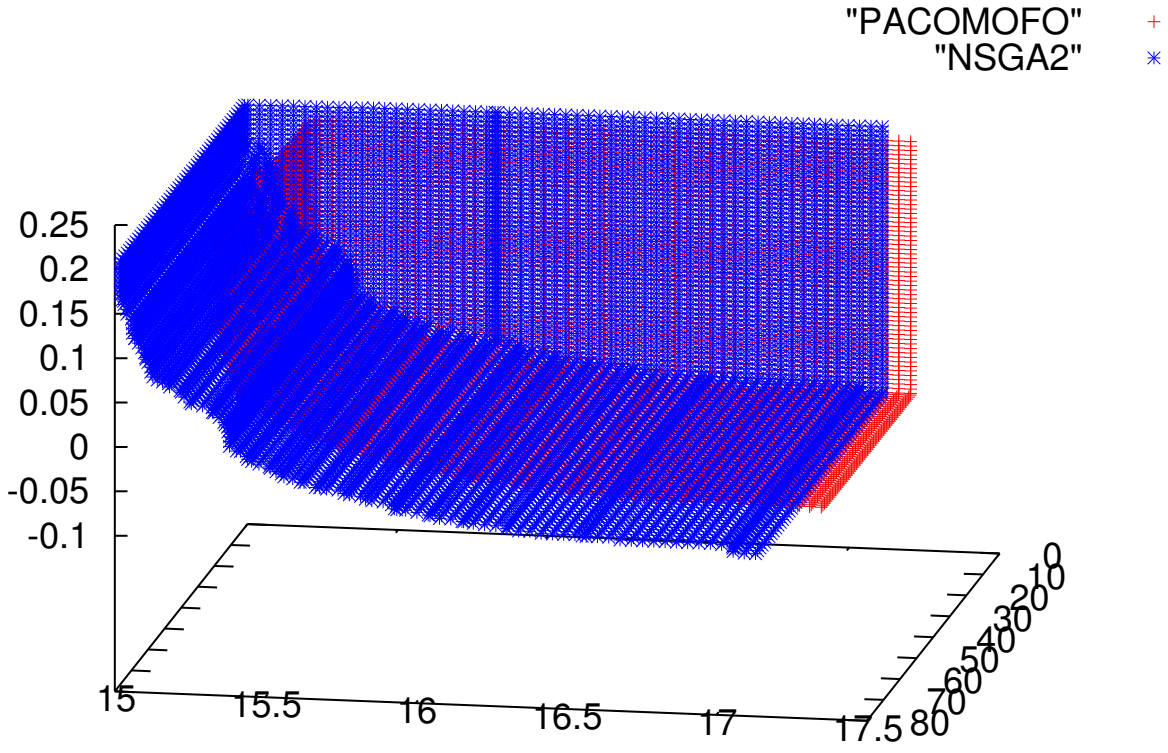
Computational Efficiency

Neither algorithm's implementation was optimised for execution speed, as a general guide though the run times of the NSGA-II and PACO-MOFO were approximately equal. More precisely though, the computational complexity of the PACO-MOFO algorithm is derived mostly from two sources, solution ranking (including fitness sharing) and population maintenance (replacement).

The solution ranking procedure comprises two steps, the first step is the NSGA-II ranking procedure which has a worst case complexity of $O(hN^2)$ where h is the number of objectives, and N the population size. The second step is the fitness sharing quality adjustment which has a complexity of $O(N^2)$ since every population member has to determine its closest members. It

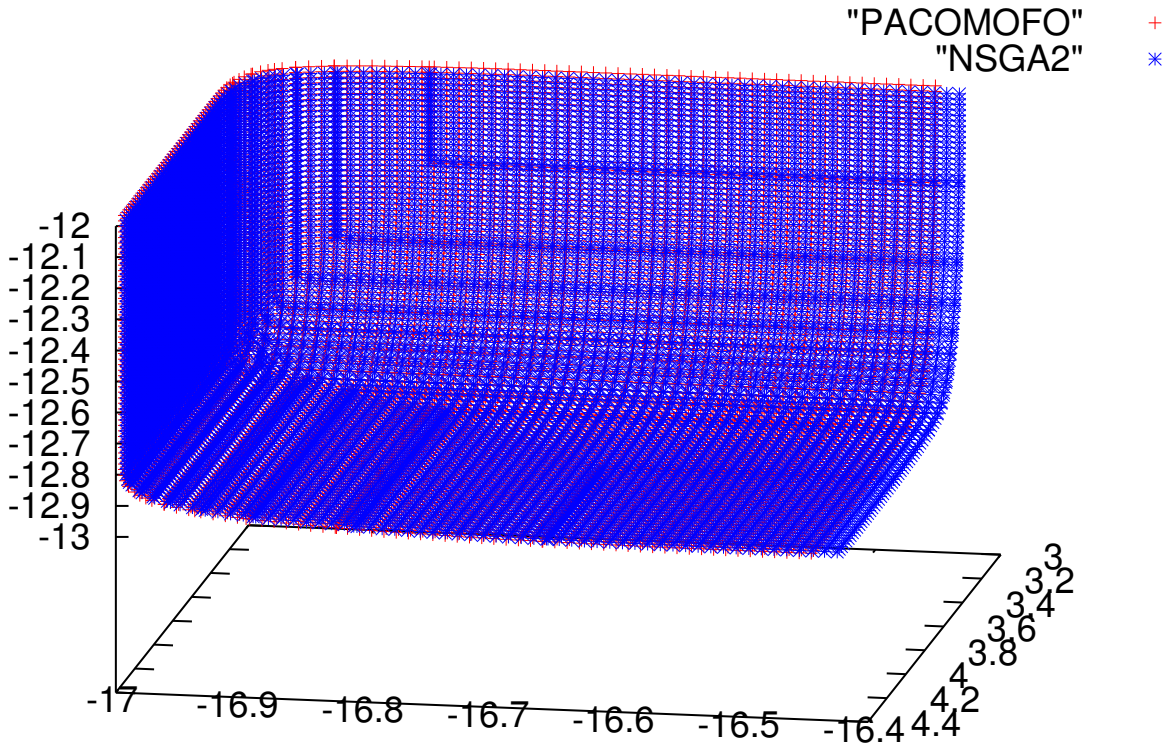


(a) 1% (best) attainment surfaces

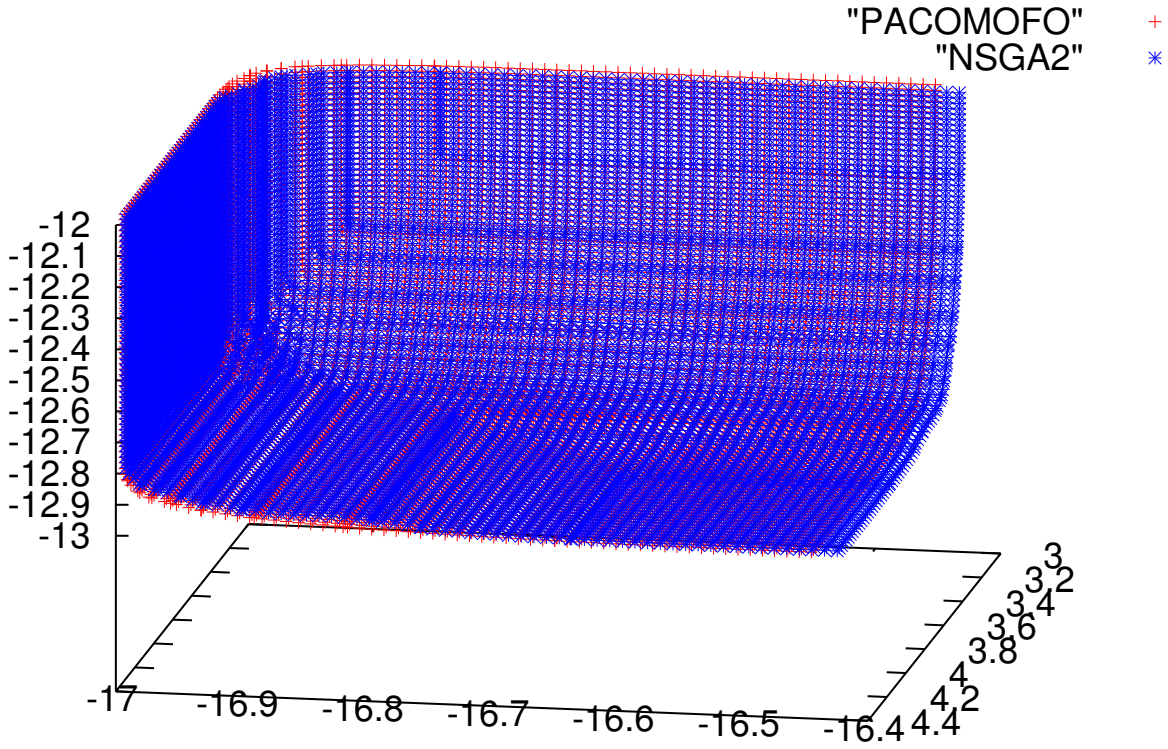


(b) 50% (average) attainment surfaces

Figure 7.31: Attainment surfaces generated from 50 runs of PACO-MOFO and NSGA-II applied to MOP5



(a) 1% (best) attainment surfaces



(b) 50% (average) attainment surfaces

Figure 7.32: Attainment surfaces generated from 50 runs of PACO-MOFO and NSGA-II applied to MOP7

may be possible to mitigate some of this computational effort by combining these two steps. The complexity of the crowding replacement operation is dependent on the size of the crowding window (w) since it selects a subset (in this case 1/2 of N) of the population and uses an objective space comparison to find the closest subset member to the new solution and then performs a single non-dominance check, of total complexity $O((N/w)^2 + h)$. The solution creation procedure is relatively lean compared to these other processes and as such has not been included. Compared to NSGA-II the PACO-MOFO algorithm is comparable since both algorithms perform similar sort and crowding distance calculations.

7.5.4 Summary

In this section the PACO-MOFO algorithm was introduced and tested on several benchmark multiple objective function optimisation problems. The benchmark functions were broken into three distinct categories, two variable two objective, multiple variable two objective and two variable multiple objective. For the simple two variable two objective problems the PACO-MOFO algorithm was able for the most part to achieve similar results to that of an accepted state-of-the art MOFO algorithm. When scaled to include more decision variables, the PACO-MOFO algorithm was observed to concentrate on improving the middle of the Pareto front which was attributed to the nature of the decision space becoming increasingly flat as the number of decision variables was increased. Perhaps the best result was reserved for last though where on two simple two variable three objective MOFO problems the PACO-MOFO algorithm was able to achieve a better result than the control algorithm. This result must not be overstated since it is in no way conclusive of a trend given that the experiment was limited to two problem instances and static parameter configurations. However, overall the results do provide some confidence in the ability of PACO-MOFO to obtain good solutions to a variety of MOFO problems.

7.6 Chapter Summary

In this chapter two niching PACO algorithms CPACO-MOTSP and PACO-MOFO were introduced and tested on the multiple objective TSP and CFO problems respectively. CPACO-MOTSP, an extension of an existing multi-objective PACO algorithm, was demonstrated to improve not only the solution quality in terms of diversity and closeness to the Pareto front but also the computational complexity when applied to the MO-TSP. The use of a crowding replacement operation was identified as a good way to control the population size and composition for this problem. The presence of a strong heuristic was identified as a possible reason for the strong performance of the algorithm and also as a way to successfully do without a pheromone matrix per objective, as is the case with many previously proposed ACO algorithms. By removing the requirement for multiple pheromone matrices the CPACO-MOTSP seems better equipped to handle combinatorial optimisation problems with larger numbers of objectives. This is due to the complexity of the algorithm not scaling as poorly as alternative algorithms as

the number of objectives is increased.

The second algorithm, PACO-MOFO was proposed as an extension of CPACO-MOTSP to determine how a similar algorithm would perform on a problem lacking heuristic information. To be fair, the results on some of the simple two decision variable two objective, while good, were not as good as NSGA-II which was to be expected given that the NSGA-II is a second generation MOEA and was designed specifically for the MOFO problem. However, the experimentation did point out some weaknesses in the algorithm design, such as the slow uptake of solutions when using a crowding replacement operation. This said, the results of the last experiment (two decision variables, three objectives) were promising with the CPACO-MOFO algorithm achieving better results (with regards to the summary attainment surface, hypervolume and epsilon indicators) on both benchmark instances than NSGA-II.

The next chapter is the final experimentation chapter where a real-world optimisation problem has been selected for analysis. The problem is represented in two ways, as a single objective continuous function optimisation and multiple objective continuous function optimisation problem. Up to this point the algorithms presented have only been applied to problems with known optima and well understood search spaces. By applying the Niching PACO algorithms to a problem with unknown search space characteristics and unknown optima, a greater understanding of the algorithms' wider applicability is possible.

CHAPTER 8

Case Study: Antenna Correlation

> Astronomy is perhaps the science whose discoveries owe least to chance, in which human understanding appears in its whole magnitude, and through which man can best learn how small he is.
>
> Georg C. Lichtenberg (1742-1799)

The testing of Niching PACO algorithms on several simple multi-modal CFO problems in Sec. 6.3 indicated that the addition of a specific form of niching (crowding) to existing PACO algorithms generally increases the algorithm performance in terms of the overall search efficacy and diversity. Another Niching PACO algorithm, PACO-MOFO, which uses both crowding and fitness sharing was applied in Sec. 7.5 to several MOFO problems with fairly acceptable results when compared to an existing state-of-the-art MOEA algorithm. While these previously presented problems are all challenging NP-hard problem instances, they still contain known global optima and much is known about their particular search space characteristics such as their modality. To be useful in a 'real-world' context, the Niching PACO algorithms should be able to be applied to other unknown problem domains and return a performance similar to those obtained on these benchmark problem instances.

In this chapter the Crowding PACO and PACO-MOFO algorithms are applied to a challenging 'real-world' problem for the purpose of determining their performance on a complex, unknown problem domain. The problem instances presented in this chapter are previously untested using global optimisation techniques and as such little information is available about their search space characteristics. Also, there is no specific knowledge of exact global optimal locations. The results gained in this experimentation give a glance at the usefulness of these search techniques when applied to a previously untested problem domain.

The problem chosen is an antenna correlation problem from the field of Very Long Baseline Interferometry (VLBI) and is described in Sec. 8.1. The problem is concerned with reducing the error in an astronomical observation through the use of a parameterised model. The solution to this problem is significant to radio astronomers, since an accurate geometric model is essential to correctly align sampled data sequences from different antennae in the time domain for any interferometry experiment. The problem can be closely modelled as a multiple objective

or single objective CFO problem which is useful as the previously presented Niching PACO algorithms can be applied directly to the problem without modification.

As the search space, and consequently the global optimum, is unknown, the results obtained by the Niching PACO algorithms are validated by comparing them against a baseline random sample and two control algorithms: NSGA-II and the Deterministic Crowding Genetic Algorithm (CGA). These algorithms have been used previously in this thesis (Sec.6.3 and Sec.7.5) as they are state-of-the-art algorithms for multiple and single objective problems respectively. Given that little is known about the nature of the search space, experiments are presented which control various aspects of the algorithm execution, mostly centred around the setting of the problems' parameter bounds. Each experiment contains a brief description followed by results and relevant discussion. General conclusions from these experiments are made by relating the results back to the 'real-world' problem in Sec. 8.5.

8.1 Problem Details

8.1.1 Problem Description

Radio interferometry involves the temporal alignment of signals from different radio telescopes (by means of electronic delays) and the correlation of the signals for the purpose of determining information about the spatial frequencies of the radio sky within the telescope's field of view. The collected data is later Fourier transformed to create an image of the radio sky. This data stream alignment requires each telescope's geocentric (geographical) position to be precisely determined, and is made more difficult due to the addition of propagation delays through the Earth's atmosphere, time-stamping errors, the source position and structure, and Earth orientation.

The residual (unmodelled) delay between a pair of antennas (a baseline) can be estimated by fitting the correlated signal phase as a function of frequency, a process known as *fringe-fitting*. For typical astronomical observations, the residual delay (usually resulting from unmodelled atmospheric contributions and time-stamp errors) is simply an error that is removed, and its source is immaterial. However, it is possible to use the residual delays to determine errors in antenna positions, a procedure known as *geodesy*. Geodesy is a challenging problem, since attaining millimetre accuracy requires measurement of relative delays on the order of picoseconds, and the removal of all other contaminating effects on the residual delay. The value of residual delay δ from antenna B to antenna A obtained from the fringe-fit solutions can be written as Equ. 8.1.1.

$$\delta_{B-A} = ((\mathbf{B}' \cdot \hat{\mathbf{S}}' - \mathbf{B} \cdot \hat{\mathbf{S}}) - (\mathbf{A}' \cdot \hat{\mathbf{S}}' - \mathbf{A} \cdot \hat{\mathbf{S}}))/c + (C_B - C_A) + (A_A - A_B) + N_{BA} \qquad (8.1.1)$$

Where:

$\mathbf{B}$ and $\mathbf{A}$	:	Geocentric antenna vectors (user defined variables)
$\hat{\mathbf{S}}$	:	Unit vector toward the source (user defined)
$\mathbf{B}'$ and $\mathbf{A}'$	:	Geocentric antenna vectors (observed values)
$\hat{\mathbf{S}}'$	:	Unit vector toward the source (observed value)
c	:	Speed of light
C	:	Time-stamp error at an antenna
A	:	Unmodelled delay due to atmospheric propagation
N_{BA}	:	Error introduced by finite signal/noise.

Of these values, N_{BA} and C have no dependence on antenna or source position. N_{BA} is randomly distributed and C is a smoothly varying function of time with modern frequency standards, leaving the geometric ($\mathbf{A}$,$\mathbf{B}$,$\hat{\mathbf{S}}$) and propagation terms (A). In a dedicated geodesic experiment, weather information is monitored at all antennas and used to correct *a priori* atmospheric propagation models, leaving antenna position errors as the dominant contribution to residual delay. However, even if atmospheric and source position errors cannot be completely subtracted from the residual delays, an estimate of antenna position errors can still be made by assuming the atmospheric and source error delay contributions to be uncorrelated with baseline error delay contributions. An example of an observed residual delay for a single baseline tracked over a 24 hour period is presented in Fig. 8.1.

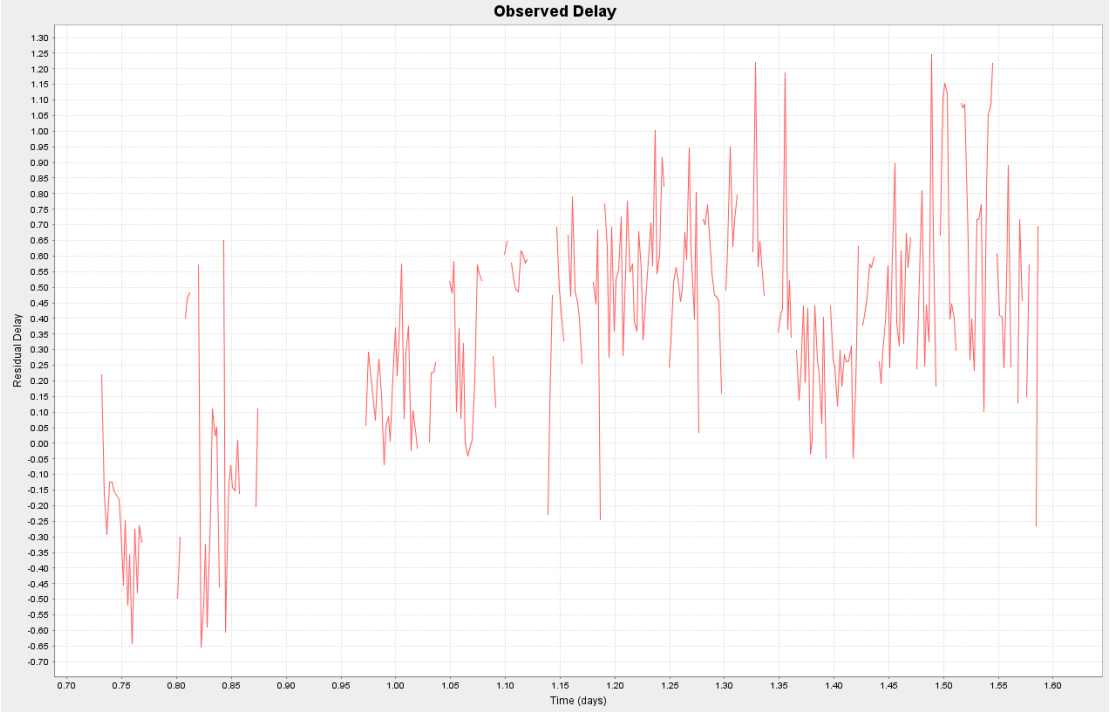


Figure 8.1: The observed residual delay for one observing band of a baseline (Parkes and ATCA) tracked over a 24 hour period taken from data set STA-131AV.

8.1.2 Objective Function

From an optimisation perspective the interest is in minimising the magnitude of the residual delay per baseline (antenna pair). This is achieved in this case by selecting appropriate values for the vectors **A**,**B** and $\hat{\mathbf{S}}$. It is not possible to back-propagate the observed data to obtain the vector values, thus like most black box optimisation problems these values are obtained through trial and error.

The vectors translate into 14 configurable (scalar) parameters per baseline. However, since one antenna is used as a common reference antenna for all antennae in the system, the actual number of configurable parameters per antenna is 7. These values represent the x, y, and z positions, two clock offsets (one per observed frequency band), a clock rate (electronic delay) and clock acceleration (drift in timestamp error). The numbers are real numbers limited within the range given in Tab. 8.1 and are valid within this entire range. Units for these values are indicated in Tab. 8.1.

Parameter (units)	Lower bound	Upper bound	Range
1,2,3 (m)	-1.0	1.0	2.0
4,5 (sec)	-5.0E-9	5.0E-9	1.0E-8
6 (sec/day),7 (sec/day^2)	-5.0E-8	5.0E-8	1.0E-7

Table 8.1: The default range of the search space per individual problem dimension for all datasets.

The objective value for each baseline is calculated by measuring the difference between each predicted data point (derived from the 14 parameters selected per baseline) and its corresponding observed (real) data point. These differences are combined in an RMS fashion into a single composite objective value for each baseline. An optimal parameter selection would allow the predicted and observed residual delays to overlap each other perfectly such that if one was subtracted from the other they would essentially cancel each other out. Due to the random noise term N_{BA} though, such a result is impossible, and a solution with minimal difference is deemed optimal.

8.1.3 Problem Data

The problem data used was gathered from five observation points, Parkes, ATCA, Mopra, Hobart and Ceduna. For all data sets Parkes is used as the reference antenna for all baselines as it is the most sensitive, thus minimising the contribution of the noise term N_{BA}. The data sets STA-131AU and STA-131AV were based on observations of two different sets of radio sources and are available upon request from the Australia Telescope National Facility[1]. Use of these datasets gives rise to four baselines using Parkes as a common reference, this configuration is visualised in Fig. 8.2.

[1]Contact: `vlbi@atnf.csiro.au`

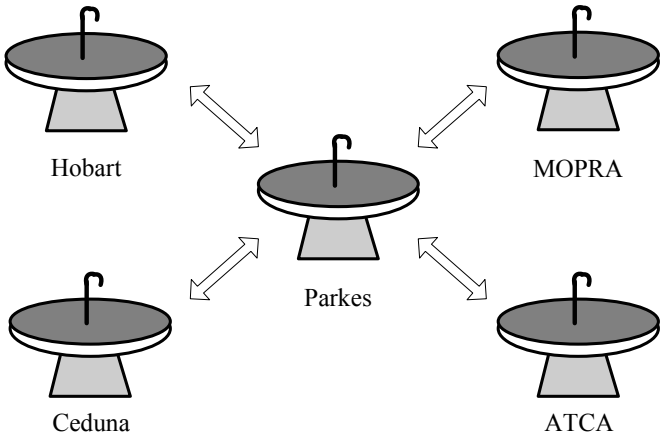


Figure 8.2: Arrangement of baselines for the STA-131AU and STA-131AV datasets. As the common reference, Parkes is included in the calculation of the residual delay for all four baselines.

8.2 Experiment One: Naive Optimisation

8.2.1 Experiment Description

As described in Sec. 8.1.2 each antenna in the system has 7 configurable continuous (real number) parameters. As such, the problem can be treated as a $7n$ dimension CFO problem, where n is the number of antennae inclusive of the reference antenna[2]. The bounds of the search space (described in Tab. 8.1) are set to a reasonably large size, using domain-specific expert knowledge, such that the global optimal solution is in all likelihood contained within this search space. It is worth mentioning that possible problems with such an approach are that the search space is too large to be effectively searched, even for a niching algorithm.

The calculation of objective values for each baseline was described in Sec. 8.1.2. This objective function scoring routine results in an individual error for each baseline, which in turn lends itself to multiple objective optimisation with $n - 1$ objectives (where n is the number of antennae inclusive of the reference antenna). Alternatively all objective values for all baselines can be summed together to form a single composite error for all baselines which lends itself to single objective optimisation.

In this experiment several multiple and single objective algorithms are applied to the problems, treating each problem instance as a 35 dimension problem and using multiple or single objective cost functions as appropriate. In addition a random search is applied to provide a baseline for comparison.

[2]This is one of many possible ways of formulating the problem. Other alternative problem formulations are described in later experiments.

8.2.2 Results and Discussion

Initial Analysis

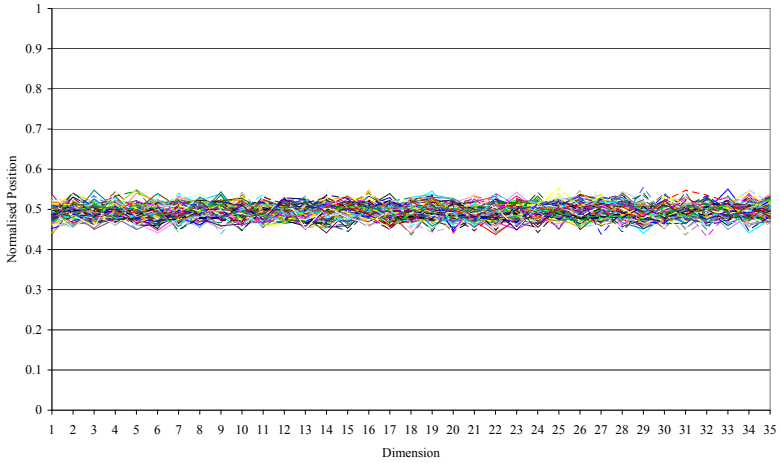
To gain an initial understanding of the search space composition, a random search with crowding replacement was used to obtain multiple interesting search space features. This search involved a total of 200 million independent random samples which were created in a sequential fashion. 100 samples were generated at a time with the first 100 comprising the initial population. At each iteration the new samples were compared against the existing population and a crowding replacement operation was used to update the population. This crowding replacement operation is the same as has been used with CPACO with a crowding window size of 1.0. The intent is that after 200 million samples the remaining population of 100 solutions should provide insight into some of the major features of the search space.

This process was applied to two known uni-modal and multi-modal functions, a simple linear function with a global optima at the origin and Schwefels' function (Sec. 6.3.2), in multiple dimensions. The application of this technique to known functions (Fig. 8.3) is used to illustrate the difference between these cases in an effort to assist in the correct interpretation of the results of the random sampling of the problem data from Sec. 8.1.3. All solutions are normalised according to the possible dimensional boundaries and are plotted for comparison in Fig. 8.3. In Fig. 8.3 all dimensions are presented in order along the x-axis while the normalised position of a solution in this dimension is presented on the y-axis, with each solution's values connected with a line.

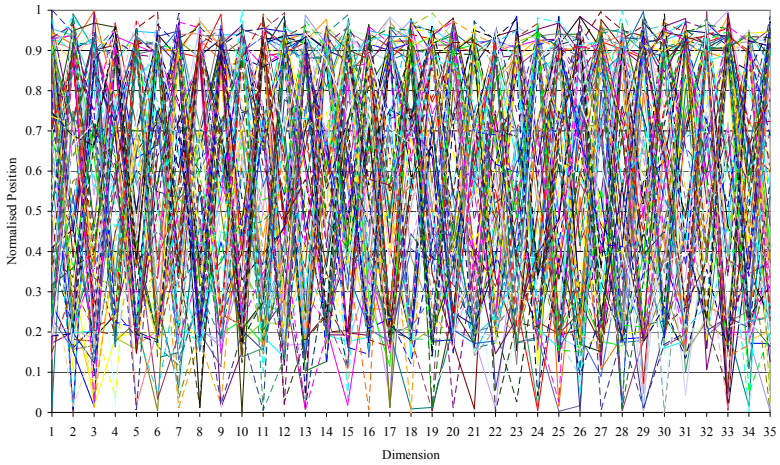
Observation of the graphs in Fig. 8.3 highlights several interesting features. It is observed that the STA-131AV dataset contains the same erratic jumps that are also evident in the multi-modal example. It is also observed that there is some regularity to the samples, with regard to the extent of coverage of the domain in all graphs. For the unimodal example this is an indication that all interesting points are located in a single area of the search space, which is known to be the case. For the multimodal example many values are at the extremes of the range which is also to be expected since the optima are located at the boundaries. It is to be noted that there is not the same clustering evident in this example since the presence of these many optima cause the crowding replacement to return many suboptimal solutions. There is a periodicity in the STA-131AV example dataset which is due to the fact that there are four antennae (each with seven variables) displayed along the x-axis and for each baseline the last two variables (acceleration and rate of acceleration) seem to always be values that are close to zero[3]. This indicates that for this problem these values should most likely be kept close to zero since this is likely to be where optimal solutions exist.

It should be clear that for each baseline, five of the seven variables tend to behave erratically across the entire variable boundaries and thus overall the STA-V131 dataset more closely resembles that of the multi-modal control dataset. The conclusion therefore is that the STA-131AV dataset, and by extension other similar datasets, are most likely multi-modal in nature.

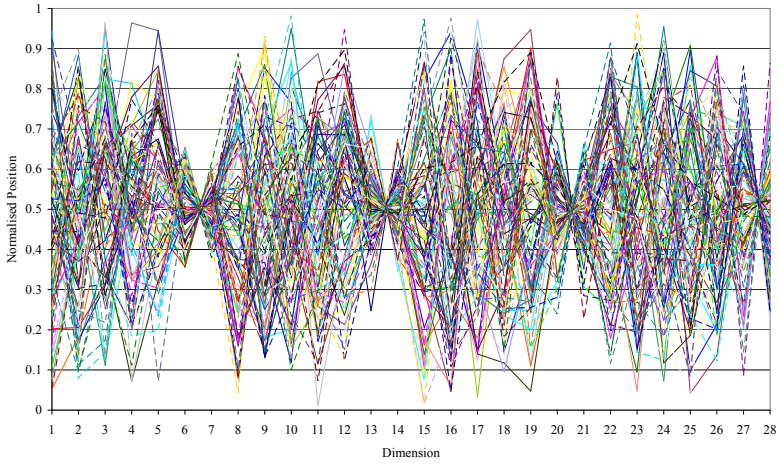
[3]Note that zero in the search space domain is normalised to 0.5 on this graph



(a) Linear function with optima at origin (Unimodal)



(b) Schwefels function (Multi-modal)



(c) STA-131AV dataset (Unknown)

Figure 8.3: Figures illustrating 100 interesting features of two known problems and one unknown problem generated using sequential random solution generation (200,000,000 samples) with a crowding reduction strategy.

Single Objective Analysis

As was indicated in the experiment description, one version of this particular experiment combines the error from all antennae into a composite RMS error value, and it is desirable to minimise this single error value. Figure 8.4 illustrates the overall distribution of results from 100 repeats of several algorithms using different random seeds with the mean and standard deviations reported in Tab. 8.2. Each algorithm was allowed 100,000 function evaluations with the best value found in that particular run recorded. The parameter settings of each algorithm are described fully in Tab. 8.3.

Algorithm / Problem	STA-131AU	STA-131AV
Random*	12.60 (1.44)	12.66 (1.62)
CGA	2.44 (0.51)	2.22 (0.39)
CPACOQuality	**1.19 (0.16)**	**1.18 (0.21)**
CPACORank	1.23 (0.24)	1.21 (0.25)
CPACOUnity	1.29 (0.19)	1.38 (0.23)

Table 8.2: Quantitative comparison of the Random, CGA and CPACO algorithms applied to the STA-131AU and STA-131AV problems. The values indicated are the mean (with standard deviation) and those highlighted in bold are the best values found. The statistical significance of the results are included in the Appendix as Tab. 10.6. *Note that the random search generated 200,000,000 samples and this measure indicates statistics of 100 of the best samples found.

CPACO	
Parameter	**Value**
Number of ants / Population size	100
History Exponent (only used with CPACOQuality & CPACORank)	1.0
Crowding window size	0.1

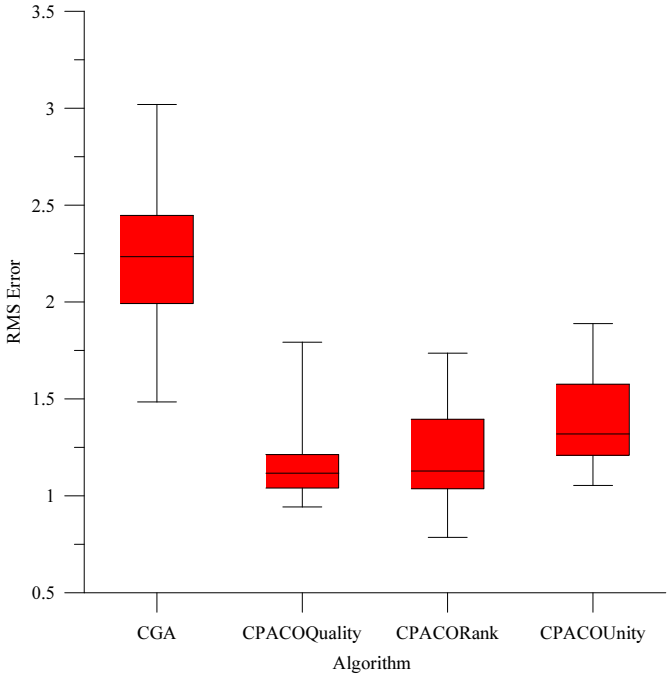
CGA	
Parameter	**Value**
Population size	100
Crossover Probability	0.95
Mutation Probability	0.85
Standard Deviation of Gaussian used for Mutation	0.01% of dimension range

Table 8.3: Parameter settings of test and control algorithms

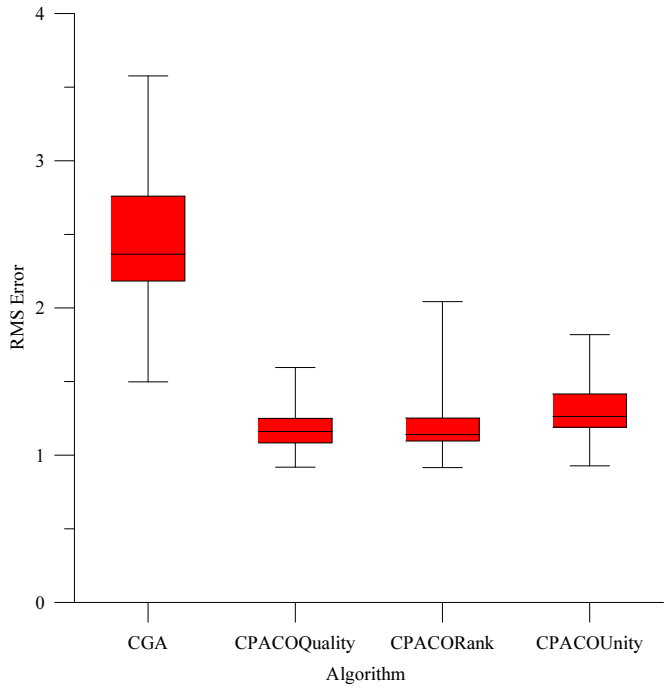
As can be seen from Fig. 8.4 the CPACO algorithms all produce good quality solutions as compared to the control algorithms. Of these CPACO variants the CPACOQuality algorithm produced the best quality results, although the difference is marginal when compared to previous results presented in Sec. 6.3.

Multiple Objective Analysis

By keeping the antennae errors separate the problem is able to be treated as a multiple objective problem with $7n$ dimensions and $n-1$ objectives (where n is the number of antennae inclusive of the reference antenna). For the STA-V131AU and STA-V131AV problems $n = 5$, as such this makes it difficult to analyse the resulting objective space given that the space is four



(a) STA-131AV



(b) STA-131AU

Figure 8.4: Results (RMS error value) presented from 100 repeats of the Crowding PACO algorithms and the control algorithm (Crowding Genetic Algorithm) applied to the 35 Dimension problems with each algorithm allowed 100,000 solution evaluations per experimental run.

dimensional. Instead the results are compared by using the hypervolume and epsilon quality metrics.

The NSGA-II and PACO-MOFO algorithms were both applied to the STA-V131AU and STA-V131AV problems and allowed 100,000 solution evaluations. The final populations were recorded for each experiment run, of which there were 100. The results of the quantitative analysis are indicated in Tab. 8.4. The parameters for both the NSGA-II and PACOMOFO algorithms are included in Tab. 8.5.

Problem	Metric	
	Hypervolume	Epsilon indicator
STA-V131AU	> (0)	> (0)
STA-V131AV	> (0)	> (0)

Table 8.4: Results of comparison between PACO-MOFO and NSGA-II on STA-V131AU and STA-V131AV datasets using the hypervolume and epsilon indicator metrics. Indicators used indicate if PACO-MOFO is significantly better than NSGA-II (>), if there is no significant difference (=), or if NSGA-II is significantly better than PACO-MOFO (<). The statistical confidence of the result (p value) is also indicated. The Mann-Whitney Rank-Sum test was used for all comparisons.

NSGA-II	
Parameter	**Value**
Population size	50
Crossover Probability	0.97
Mutation Probability	0.50
Std. Dev. of Gaussian Mutation	1% of dimension range
PACO-MOFO	
Parameter	**Value**
Number of ants (m) / Population size	50
History Exponent	1.0
Crowding Window Size	0.5
Fitness Sharing Radius (h = objectives)	$1/\left((m)^{1/h}-1\right)$
Fitness Sharing Power	1.0

Table 8.5: Parameter settings of the multiple objective algorithms

As indicated in Tab. 8.4, PACO-MOFO outperformed NSGA-II using both the hypervolume and epsilon metrics. For interest, the solutions were also evaluated using the single objective cost routine, meaning that all objectives were summed together to form a single objective value. The best (in terms of the single objective score) Pareto solution was extracted for each repeat and the average and standard deviation of these values are presented in Tab. 8.6. As can be seen in Tab. 8.6 the multiple objective algorithms did not achieve the same solution quality using the single objective cost function. This result is to be expected given that these algorithms must concentrate their search across a much larger area of the search space when compared to the single objective algorithms.

A simple dominance check was performed between the results of the CPACORank algorithm and the PACOMOFO algorithm. This dominance check was performed such that a single best solution from a single run of the CPACORank algorithm was compared against the entire Pareto set from a single run of the PACOMOFO algorithm, with this pair-wise comparison repeated

for all 100 runs. It was found that the single CPACORank solutions strongly dominated the PACOMOFO Pareto solutions in all 100 runs. This suggests that for this particular problem a multiple objective approach such as this is not justified. It is postulated that this may be because the problems' objectives are not strictly orthogonal, thus the remainder of experimentation is performed using only the single objective cost function.

Algorithm / Problem	STA-131AU	STA-131AV
PACO-MOFO	8.36 (1.81)	8.89 (2.25)
NSGA-II	39.84 (14.02)	43.09 (18.73)

Table 8.6: The average and standard deviation of the best multiple objective solutions when evaluated using the single objective cost routine.

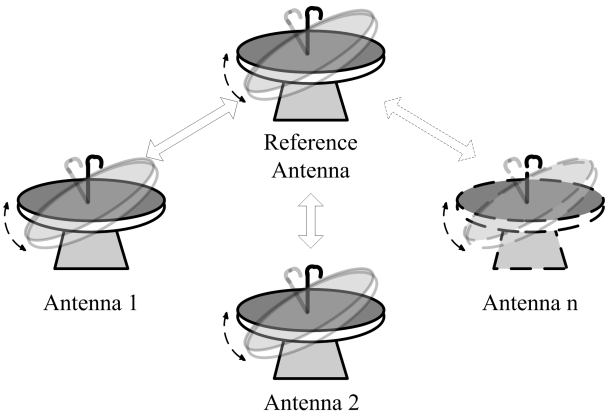
8.3 Experiment Two: Problem Decomposition

8.3.1 Experiment Description

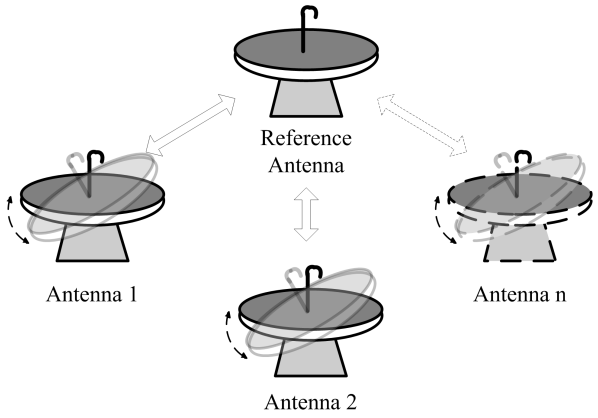
In the previous experiment the problem was treated as a $7n$ dimension problem, where n is the number of antennae, inclusive of the reference antenna. Alternatively, if the values for the reference antenna are fixed at zero this allows the problem to be decomposed into several smaller (less dimensions) problems since the effect of the reference is removed. Removal of the reference antenna's variability is a valid approach since baseline measurements are always relative to this reference antenna, and as such it will mean that all baseline variation is occurring at the individual antennae rather than at both the reference and individual antennae. By fixing the reference antenna the problem can be treated as a single 28 dimension problem or as four independent 7 dimension problems (four independent baselines). These choices are represented visually in Fig. 8.5 . For this experiment these two configurations are each tested against the original 35 dimension configuration to determine what effect they have on the overall solution quality. All experiments are completed using the CPACORank algorithm with the same algorithm parameters as used in experiment one, and the algorithm is run for the same number of evaluations (being 100,000). In this experiment the problem is only treated as a single objective problem.

8.3.2 Results and Discussion

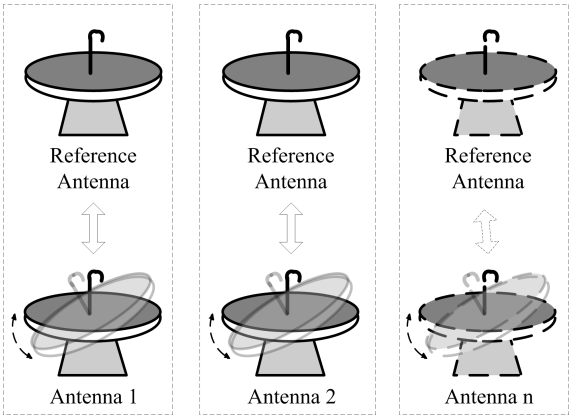
The results reported in Tab. 8.7 indicate that the alternative problem decompositions increase the quality of the result. This result is not unexpected given the findings from Sec. 6.3.5 which indicated that all of the algorithms used here decrease in efficacy as the number of dimensions is increased. Also of interest is the decrease in computational complexity of the algorithms that comes as a result of the reduction in number of dimensions. Given that the complexity of the crowding comparison operation is proportional to the number of dimensions, the less dimensions required for comparison, the smaller the search complexity becomes. This result speaks well for the 4×7D decomposition which is computationally less complex and achieves



(a) All antennae are allowed to vary, as such a change in the reference antenna will affect all antennae in the system.



(b) All antennae are allowed to vary except for the reference antenna which is fixed at zero.



(c) All antennae are allowed to vary except for the reference antenna which is fixed at zero and the problem is decomposed into several sub-problems.

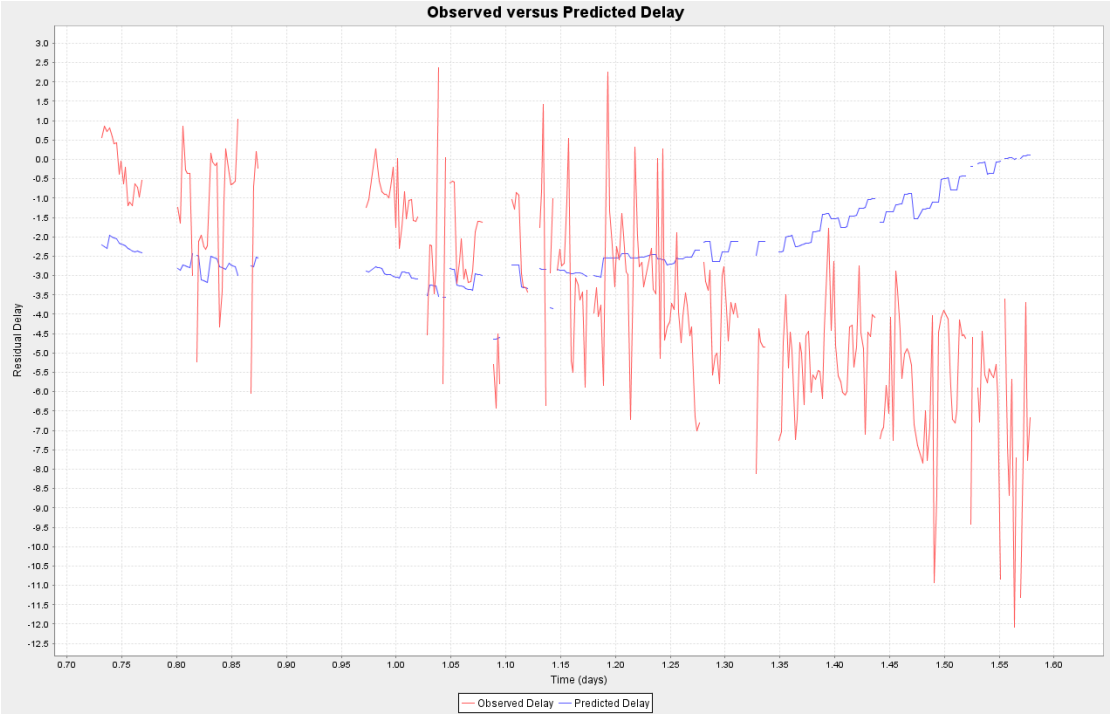
Figure 8.5: Figures illustrating possible problem compositions.

a better result than the 28D and 35D variations.

A qualitative comparison of these results is offered by plotting the predicted (modelled) residual delays against the observed residual delays (Fig. 8.6). The results displayed in Fig. 8.6 are presented for one specific baseline, Ceduna and Parkes, where Parkes is the reference antenna. It is clear from Fig. 8.6 that the quality of the fit between the observed and the predicted data improves on a qualitative level as the number of dimensions in the problem is decreased. Specifically the area of most improvement between the 35D and 28D variants is at the start of the graph, however both seem to do a poor job of fitting the latter half of the observed data. It is postulated that there may be some local minima that correspond to solutions that fit the first half of the data well, but not the second, and that both the 35D and 28D variants get trapped there[4]. It is clear from comparing quantitative results in Tab. 8.7 and qualitative results in Fig. 8.6, that a halving of the RMS error between the 35D and 4×7D variants results in a dramatic improvement in the solution obtained.

Problem	STA-131AU	STA-131AV
35D	1.20 (0.16)	1.18 (0.21)
28D	1.01 (0.14)	0.96 (0.14)
4×7D	0.60 (0.04)	0.53 (0.02)

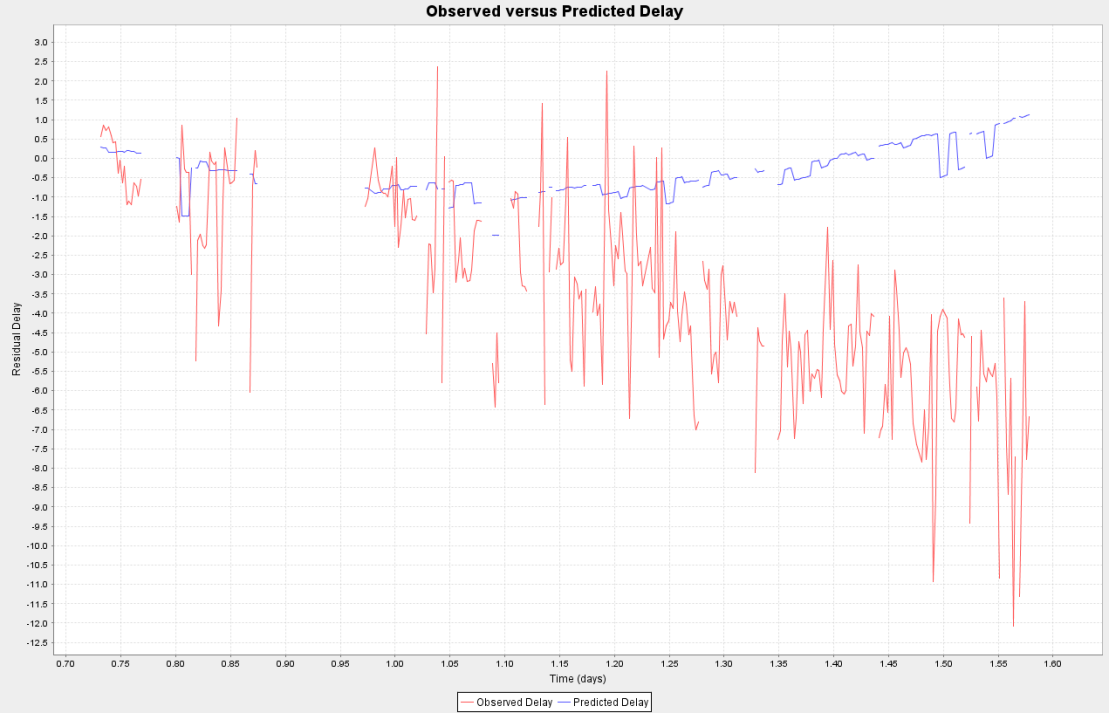
Table 8.7: Quantitative comparison of the 35D, 28D and 7D variants of the STA-131AU and STA-131AV problems. The values indicated are the mean (with standard deviation) when using the CPACORank algorithm for 100,000 solution evaluations. The statistical significance of the results are included in the Appendix as Tab. 10.7



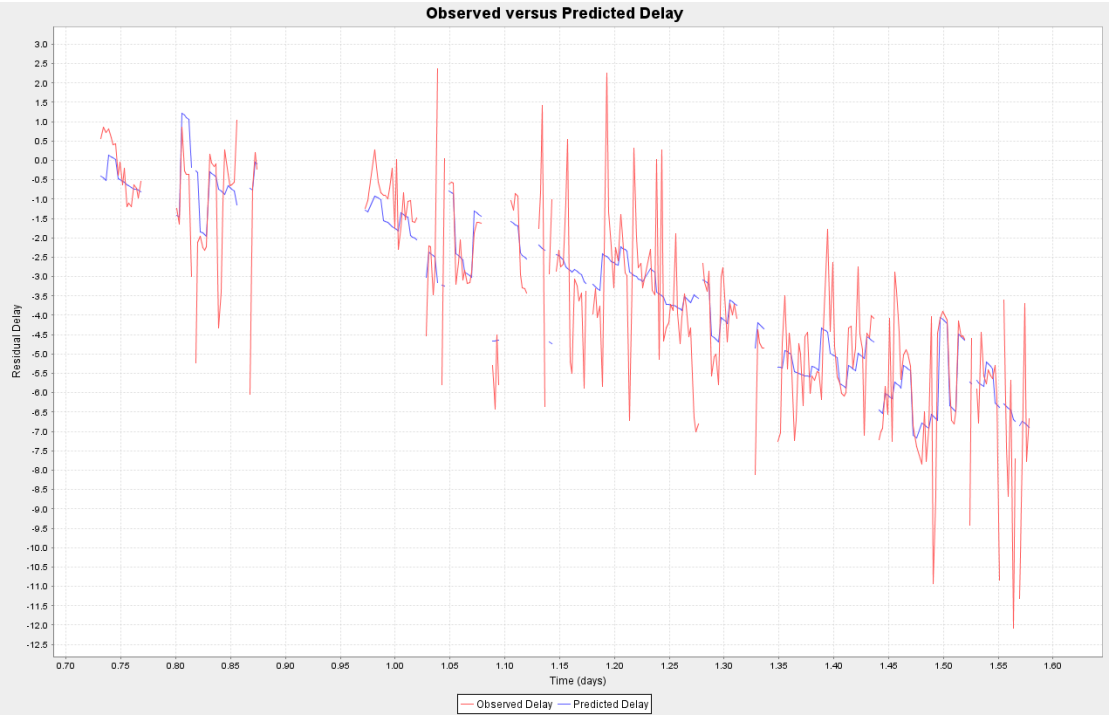
(a) 35D

Figure 8.6: (Continued on next page)

[4]There is nothing to suggest though that local minima do not exist that fit the latter half of the data well while not fitting the first half of the data.



(b) 28D



(c) 4×7D

Figure 8.6: Results obtained by plotting the predicted (modelled) residual delays against the observed residual delays. These plots are indicative of the mean level of performance of the CPACORank algorithm when run for 100,000 solution evaluations on the various problem decompositions. The plots indicate one specific baseline for one observing frequency band, Ceduna and Parkes, where Parkes is the reference antenna, even though the optimisation process was performed across multiple (4) baselines.

To be clear the same number of solution evaluations were allowed for all problems, such that each of the four 7D problems contained in the single 4×7D problem was allowed 100,000 solution evaluations. Thus the results reported for the 4×7D problem have been allowed 400,000 solution evaluations. However, given that the solution evaluation procedures' complexity scales linearly, the equivalent number of solution evaluations is approximately the same as for the 28D problem being allowed 100,000 solution evaluations. Thus the result is as close to a direct comparison as possible, meaning that the original observation of the 4×7D problem being superior in terms of computational complexity and solution efficacy holds true.

8.4 Experiment Three: Application of Heuristics

8.4.1 Experiment Description

It is intuitive that there is a finite amount of the search space that can be examined in any optimisation run, given that the number of function evaluations is finite. For the stochastic algorithms used in this analysis, a reduction in the overall search space size equates to a reduction in the problem complexity, which can hopefully lead to an increase in efficiency (speed to solution), efficacy (quality of solution), or both. A possible downside to such a reduction in search space would be that the search space no longer includes the global optimum. The purpose of this experiment is to test whether a search space reduction improves the overall result.

The search space size is reduced using a solution estimation technique which performs a rough fit of the data. This first pass rough fit of the data is performed, after which the search space is initialised around the location of the estimate solution. The estimation is performed using a first order approximation that minimises the error per individual baseline.

This experiment is performed in a similar manner to the previous experiments. All algorithms are allowed 100,000 solution evaluations of the heuristically adjusted problems with results reported for 100 repeats of each algorithm.

8.4.2 Results and Discussion

Fixed Position

After executing the solution estimation technique, the search space was initialised within the ranges given in Tab. 8.8, and Tab. 8.9 for the STA-131AU and STA-131AV datasets respectively. For comparison, refer back to the default window sizes used for the previous experiments provided in Tab. 8.1. By comparing these tables it can be seen that the search space size has been dramatically reduced by the heuristic routine.

Figure 8.7 indicates the performance of all single objective algorithms on the STA-131AV and STA-131AU datasets. The average and standard deviation of the best solutions found for all 100 repeats is reported in Tab. 8.10.

The results reported in Fig. 8.7 and Tab. 10.8 are better (with regard to the average solution

Parameter (units) / Baseline	ATCA	Mopra	Hobart	Ceduna
1,2,3 (m)	0.274	0.355	0.401	1.324
4 (sec)	4.69E-10	6.16E-10	6.72E-10	2.21E-9
5 (sec)	4.45E-10	5.69E-10	6.66E-10	2.20E-9
6 (sec/day),7 (sec/day^2)	9.14E-10	1.19E-9	1.34E-9	4.42E-9

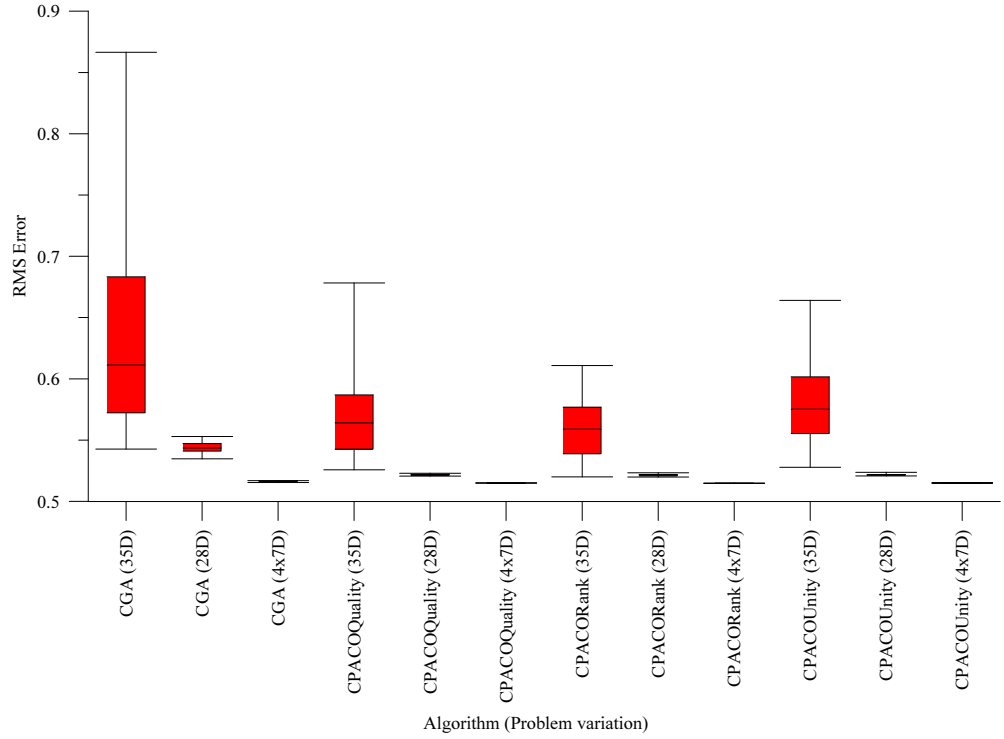
Table 8.8: The magnitude of the search space per individual problem dimension for the STA-131AU dataset after execution of the heuristic routine.

Parameter (units) / Baseline	ATCA	Mopra	Hobart	Ceduna
1,2,3 (m)	0.236	0.272	0.530	1.099
4 (sec)	3.52E-10	4.68E-10	8.34E-10	1.95E-9
5 (sec)	4.36E-10	4.4E-10	9.33E-10	1.72E-9
6 (sec/day),7 (sec/day^2)	7.88E-10	9.08E-10	1.77E-9	3.67E-9

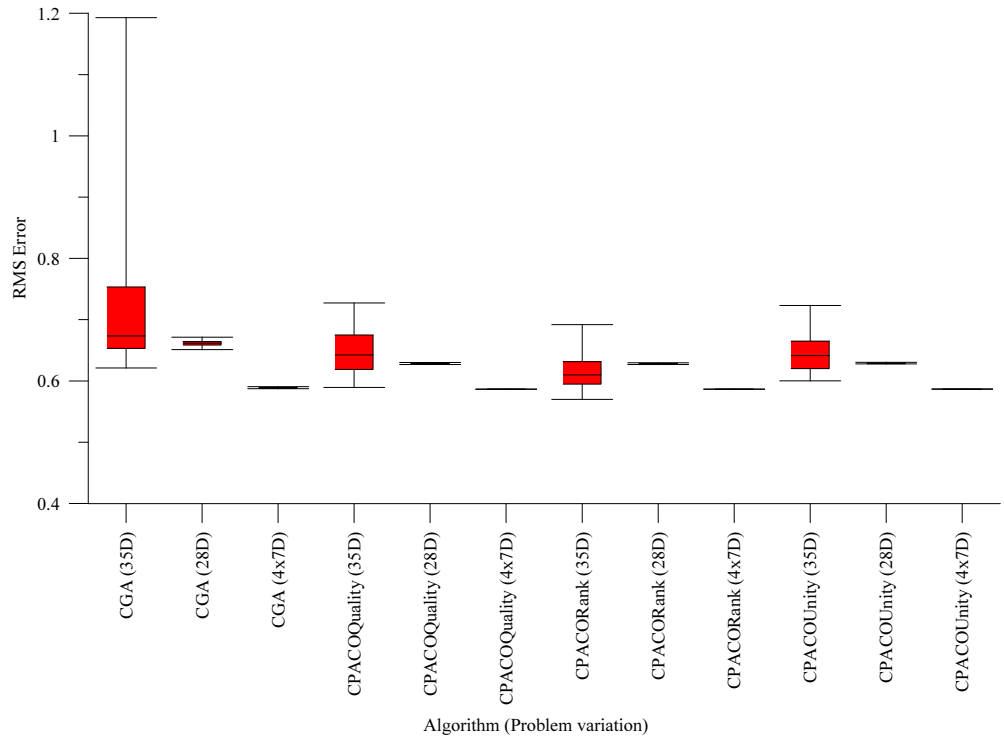
Table 8.9: The magnitude of the search space per individual problem dimension for the STA-131AV dataset after execution of the heuristic routine.

Problem	STA-131AU		
	35D	28D	4×7D
CGA	0.71 (0.11)	0.66 (0.01)	0.59 (0.00)
CPACOQuality	0.65 (0.04)	0.63 (0.00)	0.58 (0.00)
CPACORank	0.61 (0.03)	0.63 (0.00)	0.58 (0.00)
CPACOUnity	0.65 (0.03)	0.63 (0.00)	0.58 (0.00)
Problem	STA-131AV		
	35D	28D	4×7D
CGA	0.64 (0.09)	0.54 (0.01)	0.52 (0.00)
CPACOQuality	0.57 (0.04)	0.52 (0.00)	0.51 (0.00)
CPACORank	0.56 (0.02)	0.52 (0.00)	0.51 (0.00)
CPACOUnity	0.58 (0.04)	0.52 (0.00)	0.51 (0.00)

Table 8.10: Quantitative comparison of the 35D, 28D and 7D variants of the STA-131AU and STA-131AV problems using a heuristic to reduce the search space size. The values indicated are the mean (with standard deviation) when using the CPACO algorithms and the CGA algorithm each for 100,000 solution evaluations and 100 repeats. The statistical significance of the results are included in the Appendix as Tab. 10.8



(a) STA-131AV



(b) STA-131AU

Figure 8.7: Figure indicating the performance of test and control algorithms on 35D, 28D and 7D variants of the STA-131AU and STA-131AV problems using a heuristic to reduce the search space size. The values indicated are the RMS error values obtained per run when using the CPACO algorithms and the CGA algorithm each for 100,000 solution evaluations and 100 repeats

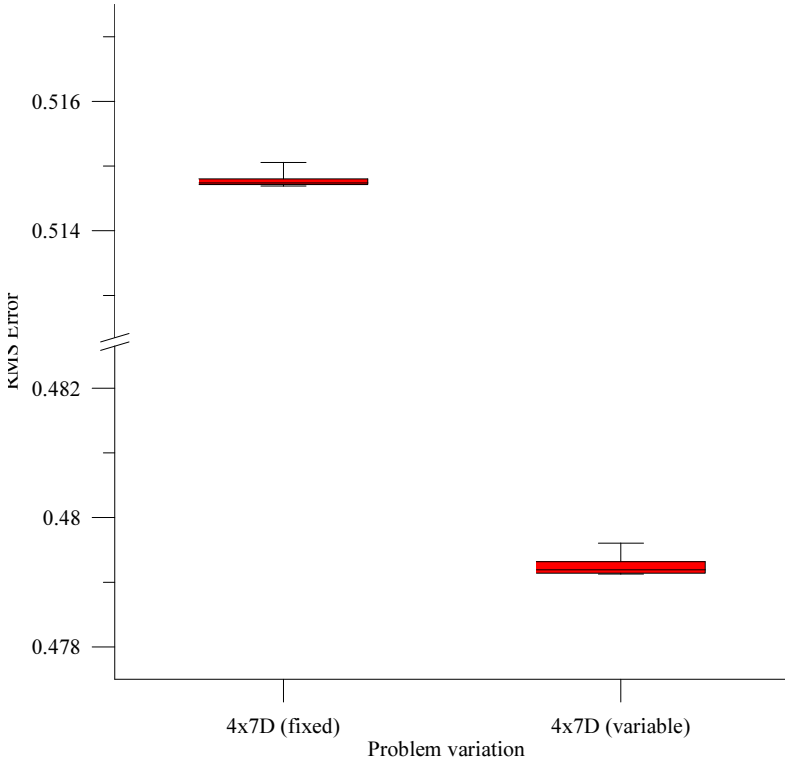
found) than those reported in the previous experiment which did not use a heuristic to reduce the search space size. It is interesting that in the STA-131AU example, the 35D problem variation found a better absolute minimum solution, although on average it was worse than the 28D and 4×7D problems. This highlights a concern that the reduction of the window size combined with a reduction in the number of searchable dimensions may exclude the optimal solution. When one considers the likelihood of actually finding this optimal solution, it may be likely that the heuristic improves the average solution quality, even though it has eliminated the global optimal solution, as has been found in this particular case. A similar result was reported in [113], where through the use of heuristics for a bin packing problem the optimal solution was excluded, however the average solution quality returned by an ACO algorithm was increased. In these cases since the location of global optimum is unlikely anyway, it is deemed better to achieve a better average solution quality than to execute a futile search for the global optimum. In other words it is preferred that the solution quality robustness be increased.

Overall the variation in performance (in terms of the quality of solutions found) between different algorithms has been reduced substantially, and one may conclude that the algorithms are approaching a minimum boundary. This result confirms that even though the ideal solution to the problem would be zero, due to random noise in the model, such a result is impossible to obtain.

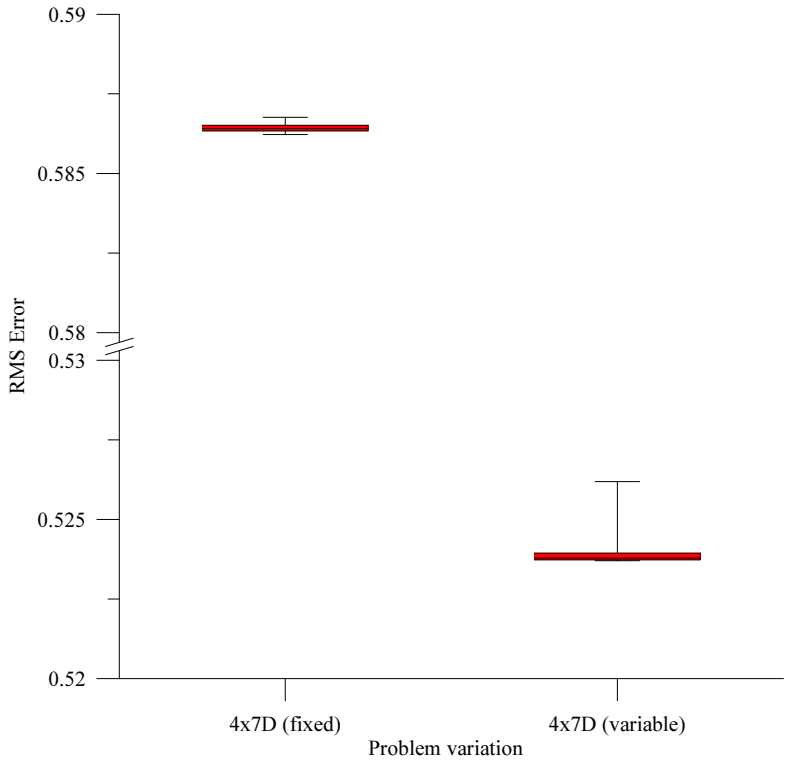
Variable Position

While problem variable ranges are chosen carefully, by expanding the range beyond where optima should be located there is no certainty that this does not exclude good areas of the search space. A simple approach to solve this problem is to expand the search space for the few variables that require a larger range, however the variables that require a larger range are often different for each problem tested, thus it is never known a-priori which variable ranges will require adjustment. This means that if the search space was to be expanded, it would have to be expanded for all variables, which can result in the search space becoming orders of magnitude larger in all dimensions, thus removing the benefit of the initial reduction in the search space size.

Another solution to the problem is to allow the algorithm to search the space for a given number of solution evaluations, and to then reposition the search space boundaries evenly around the best solution found. After repositioning the search space boundaries the algorithm can be re-initialised and run again from this new position. This process of running the algorithm and moving the search space boundaries can be repeated as many times as is required by the user, and means that if an algorithm converges at a boundary, or boundaries, in a particular run the next iteration will see this boundary lie in the centre of the new range. This method is not entirely foolproof, as a possible problem is that the algorithm may converge to the wrong end of a particular dimension of the searchable search space, and thus the next run will see the window move in the wrong direction. By comparing the results from the fixed window position experiments it should become quickly apparent whether this is the case though.



(a) STA-131AV



(b) STA-131AU

Figure 8.8: Results (RMS error value) presented from 100 repeats of the variable search strategy versus the fixed search strategy applied to the 4×7D problem with each algorithm allowed 1,000,000 solution evaluations per experimental run.

To test this variation the CPACORank algorithm was run for 1,000,000 solution evaluations with 10 shifts. This equates to 100,000 solution evaluations per unique window position. The window was initialised as in the previous experiment using the heuristic search space reduction technique. Results comparing this technique to the static search space set using the heuristic are reported in Fig. 8.8 and Tab. 8.11.

Problem Variation / Problem	STA-131AU	STA-131AV
Variable	**0.52 (0.00)**	**0.48 (0.00)**
Fixed	0.58 (0.00)	0.51 (0.00)

Table 8.11: Quantitative comparison of the fixed and variable search strategies applied to the STA-131AU and STA-131AV problems. The values indicated are the mean (with standard deviation) and those highlighted in bold are the best values found. For both strategies the CPACORank algorithm was used as the base algorithm.

The results reported in Tab. 8.11 and Fig. 8.8 indicate that the initial assumptions about the convergence of the algorithm to boundaries of the search space was correct. By allowing the algorithm to adjust the search space boundaries dynamically, it is shown that the algorithm efficacy can be improved. It is important to note that this result is not simply due to more solution evaluations being available to the algorithm, since the fixed heuristic strategy was also allowed 1,000,000 solution evaluations, and returned solutions similar to those obtained when only 100,000 solution evaluations were allowed. When combined, a heuristic to reduce the search space size and dynamic search space relocation are shown here to be effective problem solving strategies.

8.5 Final Solution Analysis

Given that the results obtained in this chapter are for an actual 'real world' problem, it is useful to make a brief comment as to whether the results meet the needs of a radio astronomy domain expert. A more accurate geodesy model than the model used in this analysis was used in an external experiment (this experimental analysis is currently unpublished) to compute the positional errors for the STA-131AV dataset. While the values reported using this complex model are more accurate than those obtained in this chapter, this complex model requires magnitudes more computational effort and incorporates more domain specific (heuristic) information to obtain a result. The results for the positional errors are reported for all four antennas (using Parkes as the reference) using the complex model (Tab. 8.12) and to compare the best result obtained using the 4×7D model with a heuristic and shifting search space boundaries from Sec. 8.4.2. The difference between these results are reported in Tab. 8.14 which indicates the difference between both approximations.

The results reported in Tab. 8.14 are acceptable considering the geodesy model used[5]. Given that the estimated error in the alternative model's measurements are approximately ±10cm per dimension, the results for both ATCA and Mopra are deemed comparable to the alternative

[5]According to a personal communication with a domain expert (Adam Deller, Centre for Astrophysics and Supercomputing, Swinburne University of Technology).

Baseline	x (m)	y (m)	z (m)
ATCA	-0.08	0.13	0.31
Mopra	-0.08	0.08	0.31
Hobart	0.00	0.18	-0.07
Ceduna	-1.72	1.65	-0.89

Table 8.12: Positional errors obtained for the STA-131AV dataset using an alternative (but more complex) geodesy model.

Baseline	x (m)	y (m)	z (m)
ATCA	-0.16	0.14	0.22
Mopra	-0.15	0.16	0.13
Hobart	-0.15	0.36	-0.31
Ceduna	-1.54	1.47	-0.58

Table 8.13: Positional errors obtained for the STA-131AV dataset using the 4×7D problem decomposition, heuristic search space reduction and relocation technique from Sec. 8.4.2.

complex model. For the Hobart and Ceduna baselines the positional errors are larger than the allowed ±10cm, which may be due to unaccounted systemic errors or the optimisation algorithm not obtaining a good solution.

Systemic errors could manifest themselves in a variety of ways since:

- The complex model uses cleaner data where all the radio sources are good, as opposed to the data used here where the sources are less reliable.
- The complex model models the atmosphere better thus reducing introduced noise.
- The complex model uses multiple widely spaced frequency bands which results in an increase in sensitivity.

While further improvements could possibly be made to the optimisation algorithm used, the results obtained suggest that systemic errors account for the discrepancy in results for the Hobart and Ceduna baselines. The consistently good results obtained for the ATCA and Mopra baselines suggest that the optimisation algorithm is able to optimise the model effectively, since in these cases the model may be more accurate. In the case of Hobart and Ceduna the model may be less accurate, thus no amount of optimisation would be able to produce positional errors that agree with the complex model's reported findings. In future experimentation the previously mentioned systemic errors could be addressed and if the results are improved this would strengthen such an assertion. For this experiment though it may be suggested that the results obtained are good, given the model used.

Baseline	x (m)	y (m)	z (m)
ATCA	0.08	0.01	0.09
Mopra	0.07	0.08	0.18
Hobart	0.15	0.18	0.24
Ceduna	0.18	0.18	0.31

Table 8.14: Magnitude of difference between the positional errors reported in Tab. 8.12 and Tab. 8.13.

8.6 Chapter Summary

In this chapter the Crowding PACO and PACO-MOFO algorithms were applied to an antenna correlation problem from the field of VLBI. The problem was concerned with the reduction of errors in an astronomical observation through the use of a parameterised model. The problem was modelled as single and multiple objective function optimisation problems, with the best results obtained for a 4×7D decomposition of the problem using a single objective error measurement. The use of a heuristic to reduce the search space size improved the obtained result which is not an unintuitive finding. The combination of heuristic and dynamic search space boundaries provided the best results for all experiments tried. This result was shown to be good in a real-world context when systematic errors are taken into consideration. Overall the results obtained in this chapter speak well for the applicability of Niching PACO algorithms to complex 'real-world' problems, however further successful testing on different optimisation problems would strengthen such an assertion.

CHAPTER 9

Summary and Final Remarks

9.1 Summary

Through a review of existing PACO algorithms a general descriptive framework for PACO was developed. Using the framework, several new PACO algorithms were developed all incorporating diversity preservation techniques from the Evolutionary Computation field. These niching techniques had been previously demonstrated to be effective at maintaining population diversity with a limited population size. The niching techniques had also been demonstrated to be effective at increasing the algorithm efficacy in many different problem domains.

An empirical analysis of these novel implementations was presented using a variety of benchmark single and multiple objective continuous function and combinatorial optimisation problems. These problems were chosen since they demonstrate the advantages and disadvantages of adding niching to a PACO algorithm. The empirical results indicated that the use of niching benefited some problem domains while offering no substantial advantage to others. The most meaningful results from empirical testing are:

- **Single objective TSP:** For the problems tested there seemed to be a marked decrease in obtained solution quality. A study of the neighbourhood relationship of an indicative TSP problem instance highlighted that the TSP may offer a natural advantage to algorithms that tend to direct search effort in one area of the search space. A simple TSP problem was constructed to demonstrate that the Niching PACO algorithms tested were exhibiting niche formation behaviour, but ultimately for this problem domain there seems to be no advantage in the application of these niching techniques.

- **Single objective CFO:** Unlike the single objective TSP some of these problems did contain multiple spatially separated optima. The performance metrics used were such that algorithms were rewarded for identification and continued maintenance of many of these spatially separated optima, thus algorithms without niching were placed at a distinct disadvantage. Of the Niching PACO algorithms tested the Crowding PACO variants exhibited the best performance, while the Fitness Sharing PACO algorithm did not seem to perform as well. Reasons for this lack of performance were most probably due to the

sensitivity of parameter selection.

Other problems without multiple optima were also tested and while this lifted the performance of the non-niching algorithms tested, the niching PACO algorithms were still able to locate the global optimal solution within a small number of function evaluations. These results illustrated that niching can be of benefit not just in the location and maintenance of multiple optima but also in assisting the location of a single global optimum.

- **Multiple objective TSP:** Unlike the single objective TSP which saw poor results for the Niching PACO algorithms, the application of the Crowding PACO algorithm to the multiple objective TSP saw a marked increase in algorithm efficacy and computational efficiency. For the problems tested, the Crowding PACO algorithm was able to maintain a good coverage of the approximate Pareto front which was closer to the actual Pareto front than that of the control algorithm. The algorithm was able to achieve this increase in efficacy while simultaneously decreasing the computational complexity of the control algorithm. This decrease was achieved through the reduction of the number of pheromone maps, a reduction in population size, and by constructing more solutions per iteration.

- **Multiple objective CFO:** The Crowding PACO algorithm that was applied to the MOTSP was modified to include fitness sharing for application to the MOFO problem domain. The resultant algorithm, PACO-MOFO, was demonstrated to be able to achieve results mostly on par with an existing state-of-the-art MOEA algorithm, NSGA-II. For some of the problem instances the PACO-MOFO algorithm tended to concentrate its search on the middle of the approximate Pareto front, this behaviour being attributed to the crowding replacement operation.

The final chapter discussed the application of the Niching PACO algorithms to an industrial application. This application was included to investigate the algorithms' applicability to a 'real-world' optimisation problem. The results demonstrated that the Niching PACO algorithms were able to consistently optimise the given problems better than the control algorithms and most importantly better than had been previously achieved using other problem solving techniques. This result confirms to some degree that the algorithms presented are able to be applied with confidence to problems where the optimal solution is unknown.

9.2 Contributions

The first contribution of this thesis was the review of all known PACO algorithms which focused on the identification of features common to these algorithms. The purpose of the review was to use commonalities between existing PACO algorithms for the development of a PACO framework. The framework presented was able to make use of existing elements from previous PACO algorithms and shows a strong similarity to the existing ACO Metaheuristic Framework. The framework provides a new perspective on existing work, while allowing for further algorithm developments and refinement.

The second contribution was the development of several new PACO algorithms imbued with a diversity preservation technique known as niching. Niching has been studied extensively in the field of Evolutionary Computation, but had never been explicitly applied to an ACO algorithm. Two of the more popular niching techniques, crowding and fitness sharing were implemented with algorithms extended from the PACO framework.

Of the problem domains used to test the new algorithms, the continuous function optimisation problem domain is one which has not received as much attention as other discrete combinatorial problem domains by the ACO literature. As such, modifications to the basic solution construction were made which extend existing ACO solution construction techniques for this problem domain. Prior to this study most ACO algorithms tended to use a single solution for new solution construction, whereas an alternative solution construction technique which samples the domain more coarse way was presented here.

While many Multiple Objective ACO algorithms have been presented in the past, this study provided the first Multiple Objective ACO algorithm that combines a single pheromone matrix, with Pareto based evaluation and an online population. This algorithm was shown to dramatically improve on an existing state-of-the-art Multiple Objective ACO algorithm. A general discussion of the utility of using multiple pheromone maps in Multiple Objective ACO algorithms was also provided. While many Multiple Objective ACO have been proposed, this work is the first to present an ant-inspired algorithm for the Multiple Objective Function Optimisation problem.

The final contribution was the application of the new algorithms to an industrial application. This like many problems residing in the astrophysics domain had never previously been attempted to be optimised by a nature-inspired metaheuristic technique. This last contribution is perhaps then the most significant since it demonstrates to the Computational Intelligence community the effectiveness of some of these Niching PACO algorithms, while also demonstrating to the Astrophysics community the possible advantages gained through the use of nature-inspired problem solving techniques.

9.3 Future Work

PACO is naturally suited to be run as a multiple population (island population) algorithm. The presence of a permanent population makes for easy sharing of solutions between populations (migration) and each population can be used to influence its own pheromone map. To-date there have not been any multiple population algorithms using the PACO metaphor, and this is an area of likely future work.

The niching algorithms developed in this thesis are able to maintain diverse (yet stable) populations of solutions over an extended period of time, while still making incremental improvements to the problem being solved (when it has not been solved to optimality). This observed behaviour could prove useful on dynamic problem instances where a non-diverse population may have difficulty adapting to changing problem constraints. Since the niched population is

already dispersed across the solution space it may prove quicker to adapt or react to changes in the solution space.

The PACO algorithm was chosen for the application of niching since niching requires access to a population of solutions for comparison against new solutions (crowding) or to influence new solution construction through modification of the fitness landscape (fitness sharing). While this is a very straightforward way to achieve niche formation, it may be possible to achieve niche formation through other means without resorting to explicit niching techniques such as those presented here. Alternative niching techniques would no doubt prove as useful as those presented in this study in similar problem domains where convergence to multiple areas of the search space is advantageous to the overall search efficacy.

Another area of possible future work is the application of the Niching PACO algorithms to other industry problems. One such problem identified but not tested in this thesis is from the field of control theory. The problem is concerned with the optimisation of a closed-loop control system through the insertion of a Proportional Integral Derivative (PID) controller. Such PID controllers require the setting of parameters to control the overall system behaviour and have been previously tuned by deterministic and non-deterministic algorithms, including EC techniques. The purpose of such a controller is to optimise several conflicting and complimentary objectives such as stability, overshoot, steady-state error, rise time and settling time. However, much of this research has focused on single objective variations which combine these parameters *a priori*. It may be worthwhile to attempt the optimisation of such a system using an *a posteriori* multiple objective optimisation approach.

On a theoretical level an analysis of PACO in the context of Evolutionary Algorithms based on Probabilistic Models (EAPM) would be rewarding. Some discussion was offered on the similarities between EAPM and PACO in Sec. 4.8. In this analysis it was concluded that PACO does fit into this category, as such it may be useful to more fully demonstrate exactly what features are common between PACO and other EAPM. It may also be interesting to investigate the effects of extra constraints on the effectiveness of PACO. Such extra constraints are often found in MOO problems, thus an understanding of the addition of extra constraints on algorithm performance could hopefully lead to a wider applicability of Multiple Objective PACO algorithms.

References

[1] D. H. Ackley. *A connectionist machine for genetic hillclimbing*, volume 28 of *Kluwer International Series In Engineering And Computer Science*. Kluwer Academic Publishers, Norwell, MA, USA, 1987.

[2] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. *The Traveling Salesman Problem*. Princeton University Press, 2006.

[3] T. Bäck, D. B. Fogel, and Z. Michalewicz, editors. *Evolutionary Computation 1, Basic Algorithms and Operators*. Institute of Physics Publishing, 2000.

[4] S. Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning,. Technical Report CMU-CS-94-163, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1994.

[5] B. Barán and M. Schaerer. A multiobjective ant colony system for vehicle routing problem with time windows. In *Proceedings of the 21st IASTED International Conference on Applied Informatics*, pages 97–102, Innsbruck, Austria, February 2003.

[6] R. S. Barr, B. L. Golden, J. P. Kelly, M. G. Resende, and W. R. Stewart. Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics*, 1(1):9–32, 1995.

[7] D. Beasley, D. R. Bull, and R. R. Martin. A sequential niche technique for multimodal function optimization. *Evolutionary Computation*, 1(2):101–125, 1993.

[8] P. Bienert. *Aufbau einer Optimierungsautomatik für drei Parameter*. Dipl.-Ing. Thesis, Technical University of Berlin, Institute of Measurement and Control Technology, 1967.

[9] G. Bilchev and I. C. Parmee. The ant colony metaphor for searching continuous design spaces. In T. C. Fogarty, editor, *Proceedings of the AISB Workshop on Evolutionary Computation*, Evolutionary Computing, pages 25–39. Springer-Verlag, April 1995.

[10] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, New York, 1999.

[11] E. Bonabeau, M. Dorigo, and G. Theraulaz. Inspiration for optimization from social insect behaviour. *Nature*, 406(6791):39–42, 2000.

[12] F. K. Branin. A widely convergent method for finding multiple solutions of simultaneous nonlinear equations. *IBM Journal of Research and Development*, pages 504–522, September 1972.

[13] R. Brits. Niching strategies for particle swarm optimization. Master's thesis, Department of Computer Science, University of Pretoria, South Africa, 2002.

[14] R. Brits, A. P. Engelbrecht, and F. van den Bergh. Scalability of niche PSO. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium (SIS '03)*, pages 228–234, 2003.

[15] B. Bullnheimer, R. Hartl, and C. Strauss. Applying the ant system to the vehicle routing problem. In *Proceedings of the second Metaheuristics International Conference (MIC'97)*, Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, Sophia-Antipolis, France, 1997. Kluwer.

[16] B. Bullnheimer, R. Hartl, and C. Strauss. A new rank based version of the Ant System - a computational study. Working Paper 3/97, Institute of Management Science, University of Vienna, Austria, April 1997.

[17] B. Bullnheimer, R. Hartl, and C. Strauss. A new rank based version of the ant system: A computational study. *Central European Journal for Operations Research and Economics*, 7(1):25–38, 1999.

[18] B. Bullnheimer, G. Kotsis, and C. Strauss. Parallelization strategies for the Ant System. In R. De Leone, A. Murli, P. Pardalos, and G. Toraldo, editors, *High Performance Algorithms and Software in Nonlinear Optimization*, volume 24 of *Applied Optimization*, pages 87–100. Kluwer, Dordrecht, 1998.

[19] P. Cardoso, M. Jesus, and A. Márquez. Multiple Objective TSP based on ACO. In *III Encuentro Andaluz de Matemáticas Discretas*, Almeria, Spain, 2003.

[20] V. Cerny. A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45:41–51, 1985.

[21] C. Coello Coello. List of references on evolutionary multiobjective optimization, 2007. Available at: `http://www.lania.mx/ ccoello/EMOO/`.

[22] C. Coello Coello. Metaheuristics for multiobjective optimization. Tutorial given at 2007 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM 2007), March 2007.

[23] A. Colorni, M. Dorigo, F. Maffioli, V. Maniezzo, G. Righini, and M. Trubian. Heuristics from nature for hard combinatorial problems. *International Transactions in Operational Research*, 3(1):1–21, 1996.

[24] A. Colorni, M. Dorigo, and V. Maniezzo. An investigation of some properties of an "Ant Algorithm". In R. Männer and B. Manderick, editors, *Proceedings of the Second International Conference on Parallel Problem Solving from Nature (PPSN II)*, pages 509–520. Elsevier Science, 1992.

[25] A. Colorni, M. Dorigo, V. Maniezzo, and M. Trubian. Ant System for job-shop scheduling. *JORBEL - Belgian Journal of Operations Research, Statistics and Computer Science*, 34(1):39–53, 1994.

[26] W. J. Conover. *Practical Nonparametric Statistics*. John Wiley and Sons, 3 edition, 1999.

[27] O. Cordón, F. Herrera, and T. Stützle. A review of the ant colony optimization metaheuristic: Basis, models and new trends. *Mathware & Soft Computing*, 9(2,3), 2002.

[28] D. W. Corne, N. R. Jerram, J. D. Knowles, and M. J. Oates. PESA-II: Region-based Selection in Evolutionary Multiobjective Optimization. In L. Spector, E. D. Goodman, A. Wu, W. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 283–290, San Francisco, California, 2001. Morgan Kaufmann Publishers.

[29] D. W. Corne, J. D. Knowles, and M. J. Oates. The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, editors, *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 839–848, Paris, France, 2000. Springer. Lecture Notes in Computer Science No. 1917.

[30] D. Costa and A. Hertz. Ants can colour graphs. *Journal of the Operational Research Society*, 48:295–305, 1997.

[31] P. J. Darwen and X. Yao. A Dilemma for Fitness Sharing with a Scaling Function. In *Proceedings of the Second IEEE International Conference on Evolutionary Computation*, Piscataway, New Jersey, 1995. IEEE Press.

[32] I. Das and J. Dennis. A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. Technical Report 96–36, Dept. Of Computational and Applied Mathematics, Rice University, Houston, Texas, USA, 1996.

[33] L. N. de Castro and J. Timmis. An artificial immune network for multimodal function optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC '02)*, volume 1, pages 699–704, 2002.

[34] K. Deb. Genetic algorithms in multimodal function optimization. Master's thesis, Department of Engineering Mechanics, University of Alabama, Tuscaloosa, AL, 1989.

[35] K. Deb. 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.

[36] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley-Interscience Series in Systems and Optimization. John Wiley & Sons, 2nd edition, 2002.

[37] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature – PPSN VI*, pages 849–858, Berlin, 2000. Springer.

[38] K. Deb and W. M. Spears. C6.2: Speciation methods. In T. Bäck, D. B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*. Institute of Physics Publishing, 1997.

[39] K. A. DeJong. *An analysis of the behaviour of a class of genetic adaptive systems*. PhD thesis, University of Michigan, 1975.

[40] J. L. Deneubourg, S. Aron, S. Goss, and J. M. Pasteels. The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, 3(2):159–168, March 1990.

[41] C. Detrain, J.-L. Deneubourg, and J. Pasteels. Decision-making in foraging by social insects. In C. Detrain, J.-L. Deneubourg, and J. Pasteels, editors, *Information Processing in Social Insects*, chapter 4, pages 331–354. Birkhäuser Verlag, 1999.

[42] C. V. Deutsch and A. G. Journel. *GSLIB : geostatistical software library and user's guide*. Applied geostatistics series. Oxford University Press, 2nd edition, 1998.

[43] G. Di Caro and M. Dorigo. AntNet: A mobile agents approach to adaptive routing. Technical Report 97-12, IRIDIA, Université Libre de Bruxelles, Belgium, 1997.

[44] G. Di Caro and M. Dorigo. AntNet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9:317–365, 1998.

[45] G. Di Caro and M. Dorigo. Mobile agents for adaptive routing. In *31st Hawaii International Conference on System*, January 1998.

[46] M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politechico di Milano, Italy, 1992.

[47] M. Dorigo, E. Bonabeau, and G. Theraulaz. Ant algorithms and stigmergy. *Future Generation Computer Systems*, 16:851–871, 2000.

[48] M. Dorigo and G. Di Caro. Ant colony optimization: A new meta-heuristic. In P. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, editors, *Proceedings of the 1999 Congress on Evolutionary Computation (CEC99)*, pages 1470–1477, Washington DC, July 1999. IEEE, IEEE Press.

[49] M. Dorigo and G. Di Caro. The Ant Colony Optimization Meta-Heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimisation*, pages 11–32. McGraw-Hill, London, UK, 1999. Also available as Tech.Rep.IRIDIA/99-1, Université Libre de Bruxelles, Belgium.

[50] M. Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5:137–172, Spring 1999.

[51] M. Dorigo and L. Gambardella. Ant colonies for the traveling salesman problem. Technical Report 1996-3, IRIDIA, Université Libre de Bruxelles, Belgium, 1996.

[52] M. Dorigo and L. Gambardella. A study of some properties of Ant-Q. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Proceedings of the Fourth International Conference on Parallel Problem Solving from Nature (PPSN IV)*, pages 656–665. Springer-Verlag, Berlin, September 1996.

[53] M. Dorigo and L. Gambardella. Ant colonies for the traveling salesman problem. *Biosystems*, 43:73–81, 1997.

[54] M. Dorigo and L. M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computing*, 1(1):53–66, 1997.

[55] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: An autocatalytic optimizing process. Technical report, Dipartmento di Electronica e Informazione, 1991.

[56] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 26(1):29–41, Feb 1996.

[57] M. Dorigo and T. Stützle. The ant colony optimization metaheuristic: Algorithms, applications and advances. Technical report, IRIDIA, 2000.

[58] J. Dréo and P. Siarry. Continuous interacting ant colony algorithm based on dense heterarchy. *Future Generation Computer Systems*, 20(5):841–856, June 2004.

[59] J. Dréo and P. Siarry. Dynamic Optimization through Continuous Interacting Ant Colony. In M. Dorigo, M. Birattari, C. Blum, L. Gambardella, F. Mondada, and T. Stützle, editors, *Proceedings of ANTS 2004 – Fourth International Workshop on Ant Colony Optimization and Swarm Intelligence*, volume 3172 of *Lecture Notes in Computer Science*, pages 422–423, Brussels, Belgium, September 2004. Springer Verlag.

[60] A. Drogoul. When ants play chess (or can strategies emerge from tactical behaviors ?). In C. Castelfranchi and J. Müller, editors, *From reaction to cognition*, volume 957 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin-Heidelberg, 1995.

[61] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. Wiley, 1966.

[62] C. M. Fonseca and P. J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Genetic Algorithms: Proceedings of the Fifth International Conference*, pages 416–423. Morgan Kaufmann, 1993.

[63] C. M. Fonseca and P. J. Fleming. Multiobjective genetic algorithms made easy: selection sharing and mating restriction. In *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, pages 45–52. IEE, September 1995.

[64] C. M. Fonseca and P. J. Fleming. On the performance assessment and comparison of stochastic multiobjective optimizers. In *PPSN IV: Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*, pages 584–593, London, UK, 1996. Springer-Verlag.

[65] L. Gambardella and M. Dorigo. Solving symmetric and asymmetric TSPs by ant colonies. In IEEE, editor, *Proceedings of the third IEEE International Conference on Evolutionary Computation (ICEC)*, pages 622–627, Nagoya, Japan, 1996. IEEE Press.

[66] L. M. Gambardella and M. Dorigo. Ant-Q: A reinforcement learning approach to the travelling salesman problem. In A. Prieditis and S. Russell, editors, *12th International Conference on Machine Learning (ML95)*, pages 252–260, Tahoe City, CA, 1995. Morgan Kaufmann.

[67] L. M. Gambardella, Éric Taillard, and G. Agazzi. Macs-vrptw: a multiple ant colony system for vehicle routing problems with time windows. *New ideas in optimization*, pages 63–76, 1999.

[68] L. M. Gambardella, E. Taillard, and G. Agazzi. MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. Technical report, IDSIA, 1999.

[69] L. M. Gambardella, E. D. Taillard, and M. Dorigo. Ant colonies for the quadratic assignment problem. *Journal of the Operational Research Society*, 50(2):167–176, February 1999.

[70] C. Garcìa-Martínez, O. Cordón, and F. Herrera. A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for bi-criteria tsp. *European Journal of Operational Research*, 180(1):116–148, June 2007.

[71] D. E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley, 1989.

[72] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49, 1987.

[73] S. Goss, S. Aron, J. L. Deneubourg, and J. M. Pasteels. Self-organized shortcuts in the argentine ant. *Naturwissenschaften*, 76:579–581, December 1989.

[74] P.-P. Grasseé. La reconstruction du nid et les coordinations inter-individuelles chez bellicositermes natalensis et cubitermes sp. la théorie de la stigmergie: Essai d'interpretation du comportement des termites constructeurs. *Insectes Sociaux*, 6:41–81, 1959.

[75] J. Grinnell. The niche relationship of the california thrasher. *Auk*, 34:427–433, 1917.

[76] M. Guntsch. *Ant Algorithms in Stochastic and Multi-Criteria Environments*. PhD thesis, Universität Fridericiana zu Karlsruhe, 2004.

[77] M. Guntsch and M. Middendorf. Applying population based ACO to dynamic optimization problems. In *ANTS '02: Proceedings of the Third International Workshop on Ant Algorithms*, pages 111–122, London, UK, 2002. Springer-Verlag.

[78] M. Guntsch and M. Middendorf. A population based approach for ACO. In S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, and G. R. Raidl, editors, *EvoWorkshops*, pages 72–81. Springer-Verlag, 2002.

[79] M. Guntsch and M. Middendorf. Solving Multi-criteria Optimization Problems with Population-Based ACO. In G. Goos, J. Hartmanis, and J. van Leeuwen, editors, *Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003)*, volume 2632 of *LNCS*, pages 464–478. Springer, April 2003.

[80] G. R. Harik. Finding multimodal solutions using restricted tournament selection. In L. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 24–31, San Francisco, CA, 1995. Morgan Kaufmann.

[81] J. H. Holland. Nonlinear environments permitting efficient adaptation. *Computer and Information Sciences II*, 1967.

[82] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introduction With Applications to Biology, Control, and Artificial Intelligence*. MIT Press, 1975.

[83] J. Hooker. Testing heuristics: We have it all wrong. *Journal of Heuristics*, pages 33–42, 1996.

[84] J. Horn. *The Nature of Niching: Genetic Algorithms and the Evolution of Optimal, Cooperative Populations*. PhD thesis, University of Illinois, 1997.

[85] J. Horn and N. Nafpliotis. Multiobjective Optimization using the Niched Pareto Genetic Algorithm. Technical Report IlliGAl Report 93005, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 1993.

[86] J. Horn, N. Nafpliotis, and D. E. Goldberg. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, volume 1, pages 82–87, Piscataway, New Jersey, June 1994. IEEE Service Center.

[87] G. E. Hutchinson. Concluding remarks, cold spring harbor symposium. *Quantitative Biology*, 22:415–427, 1957.

[88] C.-L. Hwang and A. S. M. Masud. *Multiple objective decision making, methods and applications: a state-of-the-art survey*. Number 164 in Lecture notes in economics and mathematical systems. Springer Verlag, 1979.

[89] K. A. D. Jong. Genetic algorithms are not function optimizers. In L. D. Whitley, editor, *Proceedings of the Second Workshop on Foundations of Genetic Algorithms*, pages 5–17. Morgan Kaufmann, 1992.

[90] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

[91] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[92] J. Knowles. A summary-attainment-surface plotting method for visualizing the performance of stochastic multiobjective optimizers. In *ISDA '05: Proceedings of the 5th International Conference on Intelligent Systems Design and Applications*, pages 552–557, Washington, DC, USA, 2005. IEEE Computer Society.

[93] J. D. Knowles and D. W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[94] J. D. Knowles, L. Thiele, and E. Zitzler. A tutorial on the performance assessment of stochastic multiobjective optimizers. Technical Report TIK Report No. 214, Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland, 2006.

[95] J. D. Knowles, R. A. Watson, and D. W. Corne. Reducing Local Optima in Single-Objective Problems by Multi-objectivization. In E. Zitzler, K. Deb, L. Thiele, C. A. C. Coello, and D. Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 268–282. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[96] J. R. Koza. *Genetic Programming*. MIT Press, 1992.

[97] W. H. Kruskal and W. A. Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952.

[98] F. Kursawe. A variant of evolution strategies for vector optimization. In H. P. Schwefel and R. Männer, editors, *1st Workshop on Parallel Problem Solving from Nature (PPSN)*, volume 496 of *Lecture Notes in Computer Science*, pages 193–197, Berlin, 1991. Springer.

[99] E. Lawler. *Combinatorial Optimization*. Holt, Rinehart & Winston, 1976.

[100] L. Leerink, S. Schultz, and M. Jabri. A reinforcement learning exploration strategy based on ant foraging mechanisms. In *Sixth Australian Conference on Neural Networks*, Sydney, Australia, 1995.

[101] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21:498–516, 1973.

[102] P. J. A. Lozano, D. Q. Zhang, and P. P. L. naga, editors. *Special Issue on Probabilistic Model Based EA*, IEEE Transactions on Evolutionary Computation, 2007.

[103] S. W. Mahfoud. Crowding and preselection revisited. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature 2 (PPSN2)*, pages 27–36, Amsterdam, 1992. North-Holland.

[104] S. W. Mahfoud. A comparison of parallel and sequential niching methods. In L. Eshelman, editor, *Sixth International Conference on Genetic Algorithms*, pages 136–143, San Francisco, CA, 1995. Morgan Kaufmann.

[105] S. W. Mahfoud. *Niching methods for genetic algorithms*. PhD thesis, University of Illinois, 1995.

[106] S. W. Mahfoud. Niching methods. In T. Back, D. B. Fogel, and Z. Michalewicz, editors, *Evolutionary Computation 2: Advanced Algorithms and Operators*, pages 87–92. Institute of Physics Publishing, UK, 2000.

[107] V. Maniezzo and A. Carbonaro. An ANTS heuristic for the Frequency Assignment Problem. Technical Report CSR 98-4, C. L. in Scienze dell'Informazione, Università di Bologna, Sede di Cesena, Italy, 1998.

[108] V. Maniezzo and A. Carbonaro. Ant colony optimization: an overview. In P. Hansen and C. Ribeiro, editors, *Proceedings of the third Metaheuristics International Conference (MIC'99)*, Angra dos Reis, Brazil, 1999.

[109] V. Maniezzo and A. Colorni. The ant system applied to the quadratic assignment problem. *Knowledge and Data Engineering*, 11(5):769–778, 1999.

[110] V. Maniezzo, A. Colorni, and M. Dorigo. The Ant System applied to the Quadratic Assignment Problem. Technical Report 94-28, IRIDIA, Université Libre de Bruxelles, Belgium, 1994.

[111] H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, 18:50–60, 1947.

[112] N. Monmarché, E. Ramat, G. Dromel, M. Slimane, and G. Venturini. On the similarities between as, bsc and pbil: toward the birth of a new meta-heuristic. Technical Report 215, Ecole d'Ingénieurs en Informatique pour l'Industrie (E3i), Université de Tours, 1999.

[113] J. Montgomery. Alternative representations for the job shop scheduling problem in ant colony optimsation. In M. Randall, H. A. Abbass, and J. Wiles, editors, *Progress in Artifical Life, Proceedings of the 3rd Australian Conference on Artificial Life (ACAL07)*, number 4828 in LNAI, pages 1–12. Springer, 2007.

[114] A. Moraglio and R. Poli. Topological crossover for the permutation representation. In *GECCO'05*, Washington, DC, USA, 2005.

[115] P. L. naga and J. A. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Kluwer, 2001.

[116] Y. Nakamichi and T. Arita. Diversity control in ant colony optimization. *Artificial Life and Robotics*, 7(4):198–204, 2004.

[117] J. M. Pasteels, J.-L. Deneubourg, and S. Goss. Self-organization mechanisms in ant societies (1) : Trail recruitment to newly discovered food sources. In J. M. Pasteels and J.-L. Deneubourg, editors, *From Individual to Collective Behaviour in Social Insects*. Birkhäuser Verlag, 1987.

[118] M. Pelikan, D. E. Goldberg, and F. G. Lobo. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21(1):5–20, 2002.

[119] C. Poloni, G. Mosetti, and S. Contessi. Multiobjective Optimization by GAs: Application to System and Component Design. In *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*, pages 258–264, Chichester, 1996. Wiley.

[120] S. H. Pourtakdoust and H. Nobahari. An Extension of Ant Colony System to Continuous Optimization Problems. In M. Dorigo, M. Birattari, C. Blum, L. Gambardella, F. Mondada, and T. Stützle, editors, *Proceedings of ANTS 2004 – Fourth International Workshop on Ant Colony Optimization and Swarm Intelligence*, volume 3172 of *Lecture Notes in Computer Science*, pages 294–301, Brussels, Belgium, September 2004. Springer Verlag.

[121] B. D. K. Prakash S. Shelokar, V. K. Jayaraman. Ant algorithm for single and multiobjective reliability optimization problems. *Quality and Reliability Engineering International*, 18(6):497–514, 2002.

[122] B. Prime and T. Hendtlass. Evolutionary computation using island populations in time. In B. Orchard, C. Yang, and M. Ali, editors, *Innovations in Applied Artificial Intelligence: IEA/AIE 2004*, volume 3029 of *LNCS*, pages 573–582. Springer, May 2004.

[123] M. Randall. Maintaining explicit diversity within individual ant colonies. In *Recent Advances in Artificial Life*, chapter 17. World Scientific, 2005.

[124] M. Randall and E. Tonkes. Intensification and diversification strategies in ant colony system. *Complexity International*, 9, 2002.

[125] I. Rechenberg. Cybernetic solution path of an experimental problem. Technical report, Ministry of Aviation, Royal Aircraft Establishment Translation 1122, 1965.

[126] G. Reinelt. TSPLIB - a traveling salesman problem library. *ORSA Journal on Computing*, 3:376–384, 1991.

[127] G. Reinelt. *The Traveling Salesman: Computational Solutions for TSP Applications*. Number 840 in Lecture Notes in Computer Science. Springer, 1994.

[128] G. Reinelt. Tsplib95, 1995. Available at: `http://www.iwr.uni-heidelberg.de/groups/comopt/software/tsplib95`.

[129] H. Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 55:527–535, 1952.

[130] C. E. M. Romero and E. M. Manzanares. MOAQ an Ant-Q Algorithm for Multiple Objective Optimization Problems. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Genetic and Evolutionary Computing Conference (GECCO 99)*, volume 1, pages 894–901, San Francisco, California, July 1999. Morgan Kaufmann.

[131] S. Ronald. Distance functions for order-based encodings. In *Proceedings of the IEEE Conference on Evolutionary Computation, ICEC*, pages 49–54. IEEE, Piscataway, NJ, United States, 1997.

[132] D. J. Rosenkrantz, R. E. Stearns, and P. M. L. II. An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, 6(3):563–581, 1977.

[133] H. Sato, H. E. Aguirre, and K. Tanaka. Local dominance including control of dominance area of solutions in MOEAs. In *2007 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM 2007)*, pages 310–317, 2007.

[134] J. D. Schaffer. *Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, Nashville, TN, 1984.

[135] J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 93–100, Mahwah, NJ, USA, 1985. Lawrence Erlbaum Associates, Inc.

[136] J. D. Schaffer, R. Caruana, L. J. Eshelman, and R. Das. A study of control parameters affecting online performance of genetic algorithms for function optimization. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 51–60, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[137] B. Scheuermann. *Ant Colony Optimization on Runtime Reconfigurable Architectures*. PhD thesis, Universität Fredericiana zu Karlsruhe, 2005.

[138] B. Scheuermann and M. Middendorf. Counter-based ant colony optimization as a hardware-oriented meta-heuristic. In *EvoWorkshops*, pages 235–244, 2005.

[139] B. Scheuermann, K. So, M. Guntsch, M. Middendorf, O. Diessel, H. ElGindy, and H. Schmeck. FPGA implementation of population-based ant colony optimization. *Applied Soft Computing*, 4(3):303–322, August 2004.

[140] H.-P. Schwefel. *Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik*. Dipl.-Ing. Thesis, Technical University of Berlin, Hermann Föttinger Institute for Hydrodynamics, 1965.

[141] H.-P. Schwefel. *Numerical optimization of computer models*. Wiley & Sons, 1981.

[142] K. Socha. ACO for Continuous and Mixed-Variable Optimization. In M. Dorigo, M. Birattari, and C. Blum, editors, *Proceedings of ANTS 2004 - Fourth International Workshop on Ant Colony Optimization and Swarm Intelligence*, volume 3172 of *LNCS*, pages 25–36. Springer-Verlag, Berlin, Germany, September 2004.

[143] K. Socha and M. Dorigo. Ant colony optimization for continuous domains. Technical Report 2005-037, IRIDIA, December 2005.

[144] N. Srinivas and K. Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[145] R. Storn and K. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, ICSI, 1995.

[146] T. Stützle. An ant approach to the Flow Shop Problem. In *Proceedings of the 6th European Congress on Intelligent Techniques and Soft Computing*, pages 1560–1564, Aachen, 1998.

[147] T. Stützle and H. Hoos. The $\mathcal{MAX} - \mathcal{MIN}$ Ant System and local search for combinatorial optimization problems: Towards adaptive tools for global optimization. In *Proceedings of the second Metaheuristics International Conference (MIC'97)*, Sophia-Antipolis, France, 1997.

[148] T. Stützle and H. Hoos. $\mathcal{MAX} - \mathcal{MIN}$ Ant System and local search for the Traveling Salesman Problem. In IEEE, editor, *Proceedings of the fourth International Conference on Evolutionary Computation (ICEC)*, pages 308–313. IEEE Press, 1997.

[149] T. Stützle and H. Hoos. Improvements on the Ant System: Introducing the $\mathcal{MAX} - \mathcal{MIN}$ Ant System. In *Third International Conference on Artificial Neural Networks and Genetic Algorithms*, University of East Anglia, Norwich, UK, 1997. Springer Verlag.

[150] T. Stützle and H. Hoos. $\mathcal{MAX} - \mathcal{MIN}$ Ant System. *Future Generation Computer Systems*, 16(8):889–914, 2000.

[151] E. D. Taillard and L. Gambardella. Adaptive memories for the quadratic assignment problems. Technical report, IDSIA, 1997.

[152] V. T'kindt, N. Monmarché, F. Tercinet, and D. Laügt. An Ant Colony Optimization algorithm to solve a 2-machine bicriteria flowshop scheduling problem. *European Journal of Operational Research*, 142(2):250–257, October 2002.

[153] A. Törn and A. Zilinskas. *Global Optimization*, volume 350 of *Lecture Notes In Computer Science*. Springer-Verlag New York, 1989.

[154] S. Tsubakitani. A two-dimensional mapping for the traveling salesman problem. *Computers & Mathematics*, 26:65–73, 1993.

[155] S. Tsutsui. Ant colony optimisation for continuous domains with aggregation pheromones metaphor. In *Proceedings of the 5th International Conference on Recent Advances in Soft Computing (RASC-04)*, pages 207–212, 2004.

[156] S. Tsutsui, M. Pelikan, and A. Ghosh. Performance of aggregation pheromone system on unimodal and multimodal problems. In *The IEEE Congress on Evolutionary Computation, 2005 (CEC2005)*, volume 1, pages 880–887. IEEE, September 2005.

[157] R. K. Ursem. When sharing fails. In *Proceedings of the Third Congress on Evolutionary Computation (CEC-2001)*, pages 873–879, 2001.

[158] D. A. V. Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, May 1999.

[159] R. Viennet, C. Fontiex, and I. Marc. Multicriteria optimization using a genetic algorithm for determining a pareto set. *International Journal of Systems Science*, 27(2):255–260, 1996.

[160] J.-P. Watson. A performance assessment of modern niching methods for parameter optimization problems. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Genetic and Evolutionary Computation Conference*, volume 1, pages 702–709, Orlando, Florida, USA, 1999. Morgan Kaufmann.

[161] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics*, 1:80–83, 1945.

[162] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.

[163] E. Zitzler, D. Brockhoff, and L. Thiele. *Evolutionary Multi-Criterion Optimization*, volume 4403 of *Lecture Notes in Computer Science*, chapter The Hypervolume Indicator Revisited: On the Design of Pareto-compliant Indicators Via Weighted Integration, pages 862–876. Springer, 2007.

[164] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou, and T. Fogarty, editors, *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pages 95–100, Athens, Greece, 2002.

[165] E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[166] E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, and V. Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.

[167] M. Zlochin and M. Dorigo. Model-based search for combinatorial optimization: A comparative study. In *Proceedings of 7th International Conference on Parallel Problem Solving from Nature (PPSN VII)*, 2002.

CHAPTER 10

Appendix

	FSPACO	CPACO	MMAS	PACO
FSPACO		= (0.57)	< (0)	< (0.039)
CPACO	= (0.57)		< (0)	< (0.017)
MMAS	> (0)	> (0)		> (0)
PACO	> (0.039)	> (0.017)	< (0)	

(a) Berlin52

	FSPACO	CPACO	MMAS	PACO
FSPACO		> (0)	< (0.003)	= (0.168)
CPACO	< (0)		< (0)	< (0)
MMAS	> (0.003)	> (0)		= (0.099)
PACO	= (0.168)	> (0)	= (0.099)	

(b) KroA100

	FSPACO	CPACO	MMAS	PACO
FSPACO		> (0.009)	= (0.055)	< (0)
CPACO	< (0.009)		= (0.224)	< (0)
MMAS	= (0.055)	= (0.224)		< (0)
PACO	> (0)	> (0)	> (0)	

(c) eil101

Table 10.1: Statistical significance of results for single objective TSP. Indicators used indicate if the algorithm in the left column is significantly better than the algorithm in the top row (>), if there is no significant difference (=), or if the algorithm in the top row is significantly better than the algorithm in the left column (<). The statistical confidence of the result (p value) is also indicated. The Mann-Whitney Rank-Sum test was used for all comparisons.

	FSPACO	CPACO	MMAS	PACO
FSPACO		> (0)	< (0.011)	< (0)
CPACO	< (0)		< (0)	< (0)
MMAS	> (0.011)	> (0)		< (0.002)
PACO	> (0)	> (0)	> (0.002)	

(d) ch130

	FSPACO	CPACO	MMAS	PACO
FSPACO		> (0)	< (0)	< (0)
CPACO	< (0)		< (0)	< (0)
MMAS	> (0)	> (0)		= (0.617)
PACO	> (0)	> (0)	= (0.617)	

(e) ch150

	FSPACO	CPACO	MMAS	PACO
FSPACO		> (0)	< (0)	< (0)
CPACO	< (0)		< (0)	< (0)
MMAS	> (0)	> (0)		> (0.001)
PACO	> (0)	> (0)	< (0.001)	

(f) gr202

Table 10.1: (cont.) Statistical significance of results for single objective TSP. Indicators used indicate if the algorithm in the left column is significantly better than the algorithm in the top row (>), if there is no significant difference (=), or if the algorithm in the top row is significantly better than the algorithm in the left column (<). The statistical confidence of the result (p value) is also indicated. The Mann-Whitney Rank-Sum test was used for all comparisons.

	CPACO-Unity (Fair)	CPACO-Unity (Balanced)	CPACO-Unity (Greedy)	CPACO-Rank (Fair)	CPACO-Rank (Balanced)	CPACO-Rank (Greedy)	CPACO-Quality (Fair)	CPACO-Quality (Balanced)	CPACO-Quality (Greedy)	FSPACO	ACOCD	CGA
CPACO-Unity(Fair)		< (0)	< (0)	> (0.004)	> (0)	< (0)	= (0.07)	< (0)	< (0)	> (0)	> (0)	> (0)
CPACO-Unity(Balanced)	> (0)		< (0)	> (0)	> (0)	> (0)	> (0)	< (0)	< (0)	> (0)	> (0)	= (0.053)
CPACO-Unity(Greedy)	> (0)	> (0)		> (0)	> (0)	> (0)	> (0)	> (0)	< (0)	> (0)	> (0)	> (0)
CPACO-Rank(Fair)	< (0.004)	< (0)	< (0)		> (0)	= (0.129)	< (0)	< (0)	< (0)	> (0)	> (0)	> (0)
CPACO-Rank(Balanced)	< (0)	< (0)	< (0)	< (0)		< (0)	< (0)	< (0)	< (0)	> (0)	> (0)	< (0)
CPACO-Rank(Greedy)	> (0)	< (0)	< (0)	= (0.129)	> (0)		< (0)	< (0)	< (0)	> (0)	> (0)	> (0)
CPACO-Quality(Fair)	= (0.07)	< (0)	< (0)	> (0)	> (0)	> (0)		< (0)	< (0)	> (0)	> (0)	> (0)
CPACO-Quality(Balanced)	> (0)	> (0)	< (0)	> (0)	> (0)	> (0)	> (0)		< (0)	> (0)	> (0)	> (0)
CPACO-Quality(Greedy)	> (0)	> (0)	> (0)	> (0)	> (0)	> (0)	> (0)	> (0)		> (0)	> (0)	> (0)
FSPACO	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)		< (0)	< (0)
ACOCD	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)	> (0)		< (0)
CGA	< (0)	= (0.053)	< (0)	< (0)	> (0)	< (0)	< (0)	< (0)	< (0)	> (0)	> (0)	

Table 10.2: Statistical significance of results for Schaffer's Function. Indicators used indicate if the algorithm in the left column is significantly better than the algorithm in the top row (>), if there is no significant difference (=), or if the algorithm in the top row is significantly better than the algorithm in the left column (<). The statistical confidence of the result (p value) is also indicated. The Mann-Whitney Rank-Sum test was used for all comparisons.

	CPACO-Unity (Fair)	CPACO-Unity (Balanced)	CPACO-Unity (Greedy)	CPACO-Rank (Fair)	CPACO-Rank (Balanced)	CPACO-Rank (Greedy)	CPACO-Quality (Fair)	CPACO-Quality (Balanced)	CPACO-Quality (Greedy)	FSPACO	ACOCD	CGA
CPACO-Unity (Fair)		< (0)	< (0)	< (0)	< (0)	< (0)	< (0.008)	< (0)	< (0)	> (0)	< (0)	> (0)
CPACO-Unity (Balanced)	> (0)		< (0)	= (0.667)	< (0)	< (0)	= (0.079)	= (0.086)	< (0)	> (0)	< (0)	> (0)
CPACO-Unity (Greedy)	> (0)	> (0)		> (0)	< (0)	< (0)	> (0)	> (0)	= (0.817)	> (0)	< (0)	> (0)
CPACO-Rank (Fair)	> (0)	= (0.667)	< (0)		< (0)	< (0)	> (0.037)	= (0.178)	< (0)	> (0)	< (0)	> (0)
CPACO-Rank (Balanced)	> (0)	> (0)	> (0)	> (0)		< (0)	> (0)	> (0)	> (0)	> (0)	< (0)	> (0)
CPACO-Rank (Greedy)	> (0)	> (0)	> (0)	> (0)	> (0)		> (0)	> (0)	> (0)	> (0)	< (0)	> (0)
CPACO-Quality (Fair)	> (0.008)	= (0.079)	< (0)	< (0.037)	< (0)	< (0)		< (0.001)	< (0)	> (0)	< (0)	> (0)
CPACO-Quality (Balanced)	> (0)	= (0.086)	< (0)	= (0.178)	< (0)	< (0)	> (0.001)		< (0)	> (0)	< (0)	> (0)
CPACO-Quality (Greedy)	> (0)	> (0)	= (0.817)	> (0)	< (0)	< (0)	> (0)	> (0)		> (0)	< (0)	> (0)
FSPACO	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)		< (0)	< (0)
ACOCD	> (0)	> (0)	> (0)	> (0)	> (0)	> (0)	> (0)	> (0)	> (0)	> (0)		> (0)
CGA	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)	> (0)	< (0)	

Table 10.3: Statistical significance of results for Schwefel's Function. Indicators used indicate if the algorithm in the left column is significantly better than the algorithm in the top row (>), if there is no significant difference (=), or if the algorithm in the top row is significantly better than the algorithm in the left column (<). The statistical confidence of the result (p value) is also indicated. The Mann-Whitney Rank-Sum test was used for all comparisons.

	CPACO-Unity (Fair)	CPACO-Unity (Balanced)	CPACO-Unity (Greedy)	CPACO-Rank (Fair)	CPACO-Rank (Balanced)	CPACO-Rank (Greedy)	CPACO-Quality (Fair)	CPACO-Quality (Balanced)	CPACO-Quality (Greedy)	FSPACO	ACOCD	CGA
CPACO-Unity (Fair)		> (0)	> (0)	< (0.028)	> (0)	> (0)	= (0.759)	> (0)	> (0)	> (0)	> (0)	> (0)
CPACO-Unity (Balanced)	< (0)		> (0)	< (0)	> (0)	> (0)	< (0)	= (0.486)	> (0)	> (0)	> (0)	> (0)
CPACO-Unity (Greedy)	< (0)	< (0)		< (0)	< (0)	< (0.001)	< (0)	< (0)	= (0.381)	> (0)	> (0)	< (0)
CPACO-Rank (Fair)	> (0.028)	> (0)	> (0)		> (0)	> (0)	= (0.078)	> (0)	> (0)	> (0)	> (0)	> (0)
CPACO-Rank (Balanced)	< (0)	< (0)	> (0)	< (0)		> (0)	< (0)	< (0)	> (0.001)	> (0)	> (0)	> (0)
CPACO-Rank (Greedy)	< (0)	< (0)	> (0.001)	< (0)	< (0)		< (0)	< (0)	= (0.374)	> (0)	> (0)	< (0)
CPACO-Quality (Fair)	= (0.759)	> (0)	> (0)	= (0.078)	> (0)	> (0)		> (0)	> (0)	> (0)	> (0)	> (0)
CPACO-Quality (Balanced)	< (0)	= (0.486)	> (0)	< (0)	> (0)	> (0)	< (0)		> (0)	> (0)	> (0)	> (0)
CPACO-Quality (Greedy)	< (0)	< (0)	= (0.381)	< (0)	< (0.001)	= (0.374)	< (0)	< (0)		> (0)	> (0)	< (0)
FSPACO	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)		< (0)	< (0)
ACOCD	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)	> (0)		< (0)
CGA	< (0)	< (0)	> (0)	< (0)	< (0)	> (0)	< (0)	< (0)	> (0)	> (0)	> (0)	

Table 10.4: Statistical significance of results for Himmelblau's Function. Indicators used indicate if the algorithm in the left column is significantly better than the algorithm in the top row (>), if there is no significant difference (=), or if the algorithm in the top row is significantly better than the algorithm in the left column (<). The statistical confidence of the result (p value) is also indicated. The Mann-Whitney Rank-Sum test was used for all comparisons.

	CPACO-Unity (Fair)	CPACO-Unity (Balanced)	CPACO-Unity (Greedy)	CPACO-Rank (Fair)	CPACO-Rank (Balanced)	CPACO-Rank (Greedy)	CPACO-Quality (Fair)	CPACO-Quality (Balanced)	CPACO-Quality (Greedy)	FSPACO	ACOCD	CGA
CPACO-Unity (Fair)		< (0)	> (0.002)	= (0.757)	> (0)	> (0)	= (0.336)	> (0)	> (0)	> (0)	> (0)	< (0)
CPACO-Unity (Balanced)	> (0)		> (0.029)	> (0)	> (0)	> (0)	> (0)	= (0.322)	> (0)	> (0)	> (0)	< (0)
CPACO-Unity (Greedy)	< (0.002)	< (0.029)		< (0.002)	< (0.001)	< (0)	< (0.002)	< (0.035)	< (0.002)	> (0)	> (0)	= (0.467)
CPACO-Rank (Fair)	= (0.757)	< (0)	> (0.002)		> (0)	> (0)	= (0.328)	> (0)	> (0)	> (0)	> (0)	< (0)
CPACO-Rank (Balanced)	< (0)	< (0)	> (0.001)	< (0)		> (0)	< (0)	< (0)	> (0)	> (0)	= (0.479)	< (0)
CPACO-Rank (Greedy)	< (0)	< (0)	> (0)	< (0)	< (0)		< (0)	< (0)	< (0)	> (0)	> (0)	< (0)
CPACO-Quality (Fair)	= (0.336)	< (0)	> (0.002)	= (0.328)	> (0)	> (0)		> (0)	> (0)	> (0)	> (0)	< (0)
CPACO-Quality (Balanced)	< (0)	= (0.322)	> (0.035)	< (0)	> (0)	> (0)	< (0)		> (0)	> (0)	> (0)	< (0)
CPACO-Quality (Greedy)	< (0)	< (0)	> (0.002)	< (0)	< (0)	> (0)	< (0)	< (0)		> (0)	> (0)	< (0)
FSPACO	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)	< (0)		< (0)	< (0)
ACOCD	< (0)	< (0)	< (0)	< (0)	= (0.479)	< (0)	< (0)	< (0)	< (0)	> (0)		< (0)
CGA	> (0)	> (0)	= (0.467)	> (0)	> (0)	> (0)	> (0)	> (0)	> (0)	> (0)	> (0)	

Table 10.5: Statistical significance of results for Branin's R-Cos Function. Indicators used indicate if the algorithm in the left column is significantly better than the algorithm in the top row (>), if there is no significant difference (=), or if the algorithm in the top row is significantly better than the algorithm in the left column (<). The statistical confidence of the result (p value) is also indicated. The Mann-Whitney Rank-Sum test was used for all comparisons.

	CPACOQuality	CPACORank	CPACOUnity	CGA
CPACOQuality		> (0.013)	> (0)	> (0)
CPACORank	< (0.013)		> (0)	> (0)
CPACOUnity	< (0)	< (0)		> (0)
CGA	< (0)	< (0)	< (0)	

(a) STA-131AV

	CPACOQuality	CPACORank	CPACOUnity	CGA
CPACOQuality		> (0.01)	> (0)	> (0)
CPACORank	< (0.01)		> (0)	> (0)
CPACOUnity	< (0)	< (0)		> (0)
CGA	< (0)	< (0)	< (0)	

(b) STA-131AU

Table 10.6: Statistical significance of results for for 35D Antenna Problems. Indicators used indicate if the algorithm in the left column is significantly better than the algorithm in the top row (>), if there is no significant difference (=), or if the algorithm in the top row is significantly better than the algorithm in the left column (<). The statistical confidence of the result (p value) is also indicated. The Mann-Whitney Rank-Sum test was used for all comparisons.

	35D	28D	4×7D
35D		< (0)	< (0)
28D	> (0)		< (0)
4×7D	> (0)	> (0)	

Table 10.7: Statistical significance of results for for 35D, 28D and 4×7D antenna problems. Indicators used indicate if the result in the left column is significantly better than the result in the top row (>), if there is no significant difference (=), or if the result in the top row is significantly better than the result in the left column (<). The statistical confidence of the result (p value) is also indicated. The Mann-Whitney Rank-Sum test was used for all comparisons.

	35D	28D	4×7D
35D		< (0)	< (0)
28D	> (0)		< (0)
4×7D	> (0)	> (0)	

Table 10.8: Statistical significance of results for for 35D, 28D and 4×7D antenna problems using heuristics. Indicators used indicate if the result in the left column is significantly better than the result in the top row (>), if there is no significant difference (=), or if the result in the top row is significantly better than the result in the left column (<). The statistical confidence of the result (p value) is also indicated. The Mann-Whitney Rank-Sum test was used for all comparisons.