Abstract

MEDAGLIA, ANDRÉS L. Simulation Optimization Using Soft Computing. (Under the direction of Dr. Shu-Cherng Fang and Dr. Henry L. W. Nuttle.)

To date, most of the research in simulation optimization has been focused on single response optimization on the continuous space of input parameters. However, the optimization of more complex systems does not fit this framework. Decision makers often face the problem of optimizing multiple performance measures of systems with both continuous and discrete input parameters. Previously acquired knowledge of the system by experts is seldom incorporated into the simulation optimization engine. Furthermore, when the goals of the system design are stated in natural language or vague terms, current techniques are unable to deal with this situation. For these reasons, we define and study the *fuzzy single response simulation optimization* (FSO) and *fuzzy multiple response simulation optimization* (FMSO) problems.

The primary objective of this research is to develop an efficient and robust method for simulation optimization of complex systems with multiple vague goals. This method uses a fuzzy controller to incorporate existing knowledge to generate high quality approximate Pareto optimal solutions in a minimum number of simulation runs.

For comparison purposes, we also propose an evolutionary method for solving the FMSO problem. Extensive computational experiments on the design of a flow line manufacturing system (in terms of tandem queues with blocking) have been conducted. Both methods are able to generate high quality solutions in terms of Zitzler and Thiele's "dominated space" metric. Both methods are also able to generate an even sample of the Pareto front. However, the fuzzy controlled method is remarkably more efficient, requiring far fewer simulation runs than the evolutionary method to achieve the same solution quality.

To accommodate the complexity of natural language, this research also provides

a new Bezier curve–based mechanism to elicit knowledge and express complex vague concepts. To date, this is perhaps the most flexible and efficient mechanism for both automatic and interactive generation of membership functions for convex fuzzy sets.

SIMULATION OPTIMIZATION USING SOFT COMPUTING

by

Andrés L. Medaglia

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

OPERATIONS RESEARCH

Raleigh

2000

APPROVED BY:

Dr. Shu-Cherng Fang
Chair of Advisory Committee

Dr. Henry L. W. Nuttle
Co-Chair of Advisory Committee

Dr. Stephen D. Roberts
Advisory Committee

Dr. James. R. Wilson
Advisory Committee

A Olga Lucía,
por confiar en mi y su apoyo incondicional.

A mis padres,
por la formación que me dieron y su constante sacrificio.

Biography

Andrés Leonardo Medaglia was born on April 14, 1969 in Bogotá (Colombia), son of Gloria González and Bonifacio Medaglia. He graduated from Rochester School in 1987 and earned a B.S. (1992) degree in Industrial Engineering from the Pontificia Universidad Javeriana in Bogotá. Upon graduation, he was employed as a Technical Support Engineer for NASCO S.A., the former representative of SAS Institute Inc. in Colombia and Ecuador. In 1993, he entered Universidad de los Andes in Bogotá to pursue graduate studies in Industrial Engineering. After completing his M.Sc. (1995) degree, he was appointed as an instructor in the Mathematical Modeling Area of the Industrial Engineering Department. During that time, he also worked as a consultant for Flota Mercante Grancolombiana (now Transportadora Marítima) on several projects developing strategic decision support systems for logistics and planning. In the Summer of 1996, he married Olga Lucía Sarmiento in Bogotá and moved with her to Raleigh, NC, to pursue his Ph.D. in the Operations Research Program of North Carolina State University. In the Fall of 1996, he was appointed as a Research Assistant. During that time, he was awarded a membership in the Omega Rho Honor Society and a Director's Partnership Award by the National Textile Center. Since the Summer of 1999, he has worked as an Optimization Specialist for the Operations Research Department at SAS Institute Inc., located in Cary, NC.

Acknowledgments

I thank Dr. Shu-Cherng Fang, for his guidance, responsiveness, and support as an advisor. He always went beyond his duties and responsibilities as an advisor and taught me valuable lessons that I will take with me for the rest of my life. Especially, I thank him for his constant encouragement to look deeper into the essence of matters. His dedication to doing the job well is really admirable. I thank Dr. Henry L. W. Nuttle for his support and valuable contributions as co-advisor. Especially, I thank him for his pragmatic view, which on many occasions shed light on my research. He also had the patience and incredible ability to make sense out of my not–always–readable manuscripts. I really appreciate all the time, energy, and resources he invested in me. I thank Dr. Stephen D. Roberts for teaching me the principles of object–oriented simulation, and above all, some principles of life. I thank Dr. James R. Wilson for his thoroughness and for always being encouraging and supportive. His sense of serving the community and passion for teaching are truly inspiring and will guide me through the rest of my life.

I thank the students at the Fuzzy and Neural Group at North Carolina State University. I always found their support on many facets of my life as a graduate student. Special thanks to my "big brothers" Nanchieh Chiu and Ta-Wei Hung for making my life easier. I thank Shyh-Huei Chen, who has been a good friend since the first day I joined the group.

I thank Dr. Marc-david Cohen for his incredible support throughout the time that I have been working at SAS. I especially appreciate his seasoned view of the Operations Research world and I thank him for sharing it with me. I am grateful to my colleagues at SAS, who provided me the perfect escape from school work and helped me relieve the pressure.

Several organizations have contributed to my Ph.D. degree. First of all, I thank Colciencias and Colfuturo for their significant contribution. I thank the National Textile Center for the support I received through my research assistantship. I also

thank the Universidad de los Andes and the Fulbright Commission for their initial trust and support.

I have met people that have supported me along my life and they well deserve my gratitude. I thank Carlos López-Duarte, the outstanding instructor who opened my eyes to the exciting world of Operations Research. I thank Carmen Lucila Osorno, who encouraged me to quit my job and pursue a Ph.D., when it seemed like a crazy idea. I thank Luis Pinzón, María Elsa Correal, Dr. Fernando Palacios, Mario Castillo, and Gonzalo Torres, at the Universidad de los Andes, for their continuing support. They all are among the most encouraging individuals I have ever met. I also thank my very good old friends, Gabriel E. Pardo, Edgar E. Blanco, Juan Diego Londoño, and Alejandro Sierra. All of them have provided me the inspiration that true friendship always provides. I also thank my in-laws, Olguita and Rafael Sarmiento, for their continuos support; and my brother-in-law, Ricardo Parra, who has helped me out since the first time I met him. Last, but not least, I thank Ligia Restrepo, my godmother, who has always been there to support me and my family through very rough times. Without her support, I would not have made it through college.

I thank my mother, who has supported me unceasingly since the day she brought me to this world. She has worked extremely hard and sacrificed herself to give me the best education. *¡Gracias mamá!* I thank my father, who despite his short life, taught me many of the most important things in life. I just wish he could be here with me. I thank Alexandra, my sister, who was my very first teacher and always knew how to keep me grounded to reality.

Above all, I thank my loving wife Olga Lucía. She was the one who believed in my dreams and followed them faithfully. She encouraged me to pursue them when they were just a blurry idea, not even visible far in the horizon. She believed in me, even more so than I believed in myself. Olga Lucía endured my bad moods and absences with grace and understanding.

Contents

List of Tables **ix**

List of Figures **x**

1 Introduction **1**
1.1 Simulation Optimization 1
1.2 Simulation Optimization with Vague Goals 3
1.3 Scope and Objectives of Research 4
1.4 Organization of the Dissertation 5

2 Literature Review **6**
2.1 Simulation Optimization 6
2.1.1 Optimization Over a Finite Set 7
2.1.1.1 Multiple Comparison Procedures 7
2.1.1.2 Ranking and Selection 7
2.1.2 Response Surface Methodology 7
2.1.3 Gradient Based Algorithms 8
2.1.3.1 Stochastic Approximation 8
2.1.3.2 Gradient Estimation Techniques 9
2.1.3.2.1 Finite Differences 9
2.1.3.2.2 Perturbation Analysis 10
2.1.3.2.3 Likelihood Ratio Method 10
2.1.3.2.4 Frequency Domain Experimentation 10
2.1.4 Derivative–free Methods 11
2.1.4.1 Nelder–Mead Based Methods 11
2.1.4.2 Simulated Annealing 11
2.1.5 Other Methods 12
2.1.6 Multicriteria Simulation Optimization 12
2.2 Soft Computing 12
2.2.1 Fuzzy Logic 13
2.2.2 Neurocomputing 13
2.2.3 Evolutionary Computing 14
2.2.3.1 Multicriteria Evolutionary Optimization 15

2.2.3.1.1 Aggregation Approaches 15
2.2.3.1.2 Non Pareto–based Approaches 16
2.2.3.1.3 Pareto–based Approaches 17
2.3 Simulation Optimization and Soft Computing 19

3 A Fuzzy Controlled Approach 20
3.1 Introduction 20
3.2 Proposed Approach 21
3.2.1 Simulator 22
3.2.2 Fuzzy Controller 23
3.2.3 Handling Multiple Criteria 26
3.2.4 Algorithms 28
3.2.4.1 Target Threshold for the FSO Problem 28
3.2.4.2 Pareto Optimization for the FMSO Problem 30
3.3 Knowledge Acquisition 31
3.4 Application to Flow Line Design 35
3.4.1 Introduction 35
3.4.2 Framework 36
3.4.3 Simulator 37
3.4.4 Knowledge Base Design 38
3.4.4.1 State and Control Variables 38
3.4.4.2 Linguistic Data Base Design 39
3.4.4.3 Rule Base Design 40
3.4.4.3.1 Single Antecedent Rules 40
3.4.4.3.2 Multiple Antecedent Rules 41
3.4.5 Computational Experiments 43
3.4.5.1 Single Objective 43
3.4.5.2 Multiple Objectives: Generating the Efficient Frontier 46
3.5 Concluding Remarks 47

4 A Multicriteria Evolutionary Approach 49
4.1 Evolutionary Algorithm 49
4.1.1 Representation 51
4.1.2 Evaluation 52
4.1.3 Fuzzification 52
4.1.4 Fitness Assignment 53
4.1.4.1 Example 57
4.1.5 Selection 59
4.1.6 Genetic Operators 59

4.1.7 Approximate Pareto Optimal Set 61
4.1.8 Elitism 61
4.2 Application to the Flow Line Design Problem and Comparison with the Fuzzy Controlled Approach 63
4.3 Conclusions 65

5 An Efficient and Flexible Mechanism for Constructing Membership Functions 68
5.1 Introduction 68
5.2 Membership Function Generation 70
5.2.1 Overview 70
5.2.2 Current Methods 71
5.3 Proposed Mechanism 72
5.3.1 Bézier Curves 72
5.3.2 Mathematical Framework 73
5.3.3 Methodology 77
5.3.3.1 Basic Operations 77
5.3.3.2 Data–driven Estimation 77
5.4 Performance 82
5.4.1 Flexibility 82
5.4.2 Numerical Examples 83
5.4.3 Computational Efficiency 88
5.5 Conclusion 88

6 Summary and Recommendations 90
6.1 Summary 90
6.2 Recommendations for Future Research 92

A Membership Functions 110
A.1 State Variables 110
A.2 Control Variables 116

B Rule Base for the Multiple Objective Scenario 121

C Results for the Fuzzy Controlled Approach 124

D Results for OSEA 127

List of Tables

3.1 Flow line simulation inputs ranges 38
3.2 Initial conditions for the single–goal scenario with improved controllability 45

4.1 Evaluated and fuzzified performance measures for $E(t)$. 58
4.2 Comparison of fitness assignment methods. 59
4.3 Bounds on weights for OSEA experiments. 64

5.1 Control points (before change). 83
5.2 Sum of square errors (SSE) 86
5.3 SSE progress for the test benchmark cases ($\epsilon = 0.0010$; †: final solution). 86

A.1 Flow line fuzzy controller inputs. 111
A.2 Flow line fuzzy controller outputs 116

List of Figures

1.1 Simulation optimization framework 2

3.1 Fuzzy controlled simulation optimization 22
3.2 Target Threshold Algorithm 29
3.3 Pareto Optimization Algorithm 32
3.4 Fuzzy controlled simulation optimization and knowledge extraction . 34
3.5 Tandem of queues with blocking (flow line) 36
3.6 Flow line optimization framework 37
3.7 Overall work–in–process (ϖ) 39
3.8 Utilization at station 1 (φ_1) 39
3.9 Change in server rate at station 1 ($\Delta\gamma_1$) 40
3.10 Correlation coefficients extracted from the full correlation matrix . . . 41
3.11 Response surface for the work–in–process ($s_3 = 4$, $s_4 = 6$, $b_2 = 4$, $b_4 = 5$, and $s_2 = 7$) 42
3.12 Response surface for the utilization at station 1 ($s_3 = 4$, $s_4 = 6$, $b_2 = 4$, $b_4 = 5$, and $s_2 = 7$) 43
3.13 Control path for the overall work–in–process membership function for the single–goal scenario. 45
3.14 Solution for different initial conditions for the single–goal scenario with improved controllability 46
3.15 Approximate Pareto front for the two-goal scenario 47
3.16 Zitzler and Thiele's dominated space metric [150] for the two-goal scenario 48

4.1 Optimal Scoring Evolutionary Algorithm (OSEA) 50
4.2 Representation in OSEA. 52
4.3 Optimal fitness in performance space. The symbols ▲, •, and ■ represent the first, second, and third tier of optimal scores, respectively. . 60
4.4 Stochastic Universal Sampling 60
4.5 One–point crossover in OSEA. 61
4.6 Updating $\mathcal{P}^*_{\text{approximate}}(t)$ 62
4.7 Comparison of the dominated space. 65
4.8 Comparison of the number of simulation runs. 66

4.9 Comparison of the Pareto front. 66

5.1 Types of membership functions. 76
5.2 Finding $\mu_{\tilde{A}}(x)$ given x 78
5.3 Finding ${}^{\alpha}\tilde{A}$ given α 79
5.4 Data–driven estimation of the right membership function 82
5.5 Effect on the change of a single control point. 84
5.6 Data–driven estimation. Data: subject 35 (*Old man*). 85
5.7 Data-driven estimation ($\epsilon = 0.0010$). 87

A.1 Overall work-in-process (w) 112
A.2 Work-in-process at stage 1 (w_1) 112
A.3 Work-in-process at stage 2 (w_2) 112
A.4 Work-in-process at stage 3 (w_3) 113
A.5 Work-in-process at stage 4 (w_4) 113
A.6 Time in system (T) 113
A.7 Utilization at station 1 (ρ_1) 114
A.8 Utilization at station 2 (ρ_2) 114
A.9 Utilization at station 3 (ρ_3) 114
A.10 Utilization at station 4 (ρ_4) 115
A.11 Change in server rate at station 1 ($\Delta\mu_1$) 118
A.12 Change in server rate at station 3 ($\Delta\mu_3$) 118
A.13 Change in servers at station 2 (Δs_2) 118
A.14 Change in servers at station 3 (Δs_3) 119
A.15 Change in servers at station 4 (Δs_4) 119
A.16 Change in buffer space at station 2 (Δb_2) 119
A.17 Change in buffer space at station 4 (Δb_4) 120

C.1 Experiment # 1 of the two-goal scenario with the fuzzy controlled approach. 124
C.2 Experiment # 2 of the two-goal scenario with the fuzzy controlled approach. 125
C.3 Experiment # 3 of the two-goal scenario with the fuzzy controlled approach. 125
C.4 Experiment # 4 of the two-goal scenario with the fuzzy controlled approach. 126
C.5 Experiment # 5 of the two-goal scenario with the fuzzy controlled approach. 126

D.1 Experiment # 1 of the flow line design problem with two goals solved by OSEA. 127
D.2 Experiment # 2 of the flow line design problem with two goals solved by OSEA. 128

D.3 Experiment # 3 of the flow line design problem with two goals solved by OSEA. 128
D.4 Experiment # 4 of the flow line design problem with two goals solved by OSEA. 129
D.5 Experiment # 5 of the flow line design problem with two goals solved by OSEA. 129
D.6 Experiment # 6 of the flow line design problem with two goals solved by OSEA. 130

Chapter 1

Introduction

1.1 Simulation Optimization

A system is a collection of entities that act and interact toward the accomplishment of a logical end [116]. In order to study a system rigorously, the system is modeled in the form of logical and mathematical relationships. If the model is simple enough, the performance of the underlying system can be evaluated analytically. Nevertheless, in real–world scenarios, the presence of stochastic elements and complex interactions between the system entities often preclude the possibility of obtaining an analytical solution. In these cases, the model can be studied using *simulation*. In this dissertation, *simulation* refers to *discrete event simulation*.

In *discrete event simulation*, the state of the system may change with the occurrence of instantaneous events at separate and countable points in time [81]. Real world computer, communication, and manufacturing networks are examples of highly complex systems that are commonly evaluated using discrete event simulation [56, 81, 110, 113].

The simulation approach to problem solving, typically involves a series of trials in which changes are made to the input variables so that resulting changes in the output variables (responses or system performances) can be observed and identified [19]. Even though simulation can provide a very detailed and accurate model, it is in itself more of a descriptive than a prescriptive modeling tool [124].

The problem known as *simulation optimization* is that of finding the values for the input parameters such that an expected system performance is optimized. Figure

1.1 depicts a simulation optimization framework. In this framework the output of a complex model is introduced into an optimization strategy which adjusts and feeds the inputs back to the model.

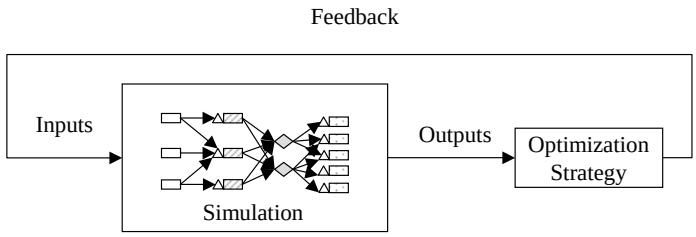


Figure 1.1: Simulation optimization framework

Formally, the *single response simulation optimization* (SRSO) problem can be stated as:

$$\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \tag{1.1}$$

where, $f(\mathbf{x}) = E[L(\mathbf{x},\omega)]$ is the expected value of the system performance measure of interest, $L(\mathbf{x},\omega)$ is the sample performance, ω represents the stochastic effects of the system, $\mathbf{x}$ is a vector of N controllable parameters, and $\mathcal{X}$ is a closed set of constraints on $\mathbf{x}$.

The SRSO problem has received much attention by the simulation community. Several reviews on the field of simulation optimization deal almost exclusively with this problem [3, 5, 19, 45, 70, 90, 111]. Brief descriptions of various methods of simulation optimization are given in Chapter 2.

In practice, most of the time an analyst has to consider several criteria simultaneously. In the presence of conflicting objectives, a simulation optimization method must take into account the tradeoff between these criteria. The *multiple response simulation optimization* (MRSO) problem is:

$$\min_{\mathbf{x}\in\mathcal{X}}\{(f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_K(\mathbf{x}))\} \tag{1.2}$$

where for $k = 1, \ldots, K$, $f_k(\mathbf{x}) = E[L_k(\mathbf{x},\omega)]$ is the expected value of the k-th performance measure of interest; $L_k(\mathbf{x},\omega)$ is the sample of the k-th performance; ω represents the stochastic effects of the system; $\mathbf{x}$ is the vector of controllable parameters; and $\mathcal{X}$ is the closed set of constraints on $\mathbf{x}$.

Despite of its real world applicability, relatively little research has been conducted on the MRSO problem. Montgomery and Bettencourt [92] use the Geoffrion, Dyer, Feinberg method (GDF) to optimize a simulation model with four criteria and two input parameters. Clayton et al. [25] and Rees et al. [106] use a goal programming approach. Biles and Swain [11, 12] propose and compare a first and second order response surface approach with a direct search algorithm. Evans et al. [36] suggest a set of guidelines for the multicriteria optimization of simulation models. A brief description of these methods is given in Chapter 2.

1.2 Simulation Optimization with Vague Goals

Most of the time the goals for a system are stated in vague natural language by the decision maker. For instance, in a manufacturing setting a decision maker may want to design a system with *low* work–in–process and *high* utilization of a certain expensive machine. In this case the vague terms *low* and *high* introduced by the decision maker have to be incorporated into the analysis, deduced, and interpreted according to the context. In this setting, the problem is to find the value of the controllable parameters such that all the objectives of the system are satisfied to a high degree.

Fuzzy technology provides a proven and efficient way to compute with words and incorporate natural language into the area of simulation optimization. Fuzzy set theory was first introduced in the 1960s by Lotfi A. Zadeh as a way to capture uncertainty and vagueness often overlooked in the analysis of complex systems [144]. A fuzzy set $\tilde{A}$ is characterized by its *membership function* $\mu_{\tilde{A}}$, which maps each element of the universe X to the interval $[0, 1]$. This function indicates the degree to which each element belongs to the set.

Rewriting SRSO to incorporate vagueness, we have the *fuzzy single response simulation optimization* or *fuzzy simulation optimization* (FSO) problem:

$$\max_{\mathbf{x} \in \mathcal{X}} \mu_{\tilde{G}}(f(\mathbf{x})) \tag{1.3}$$

where, $\mu_{\tilde{G}}(f(\mathbf{x}))$ is a measure of the degree of satisfaction of the vague target represented by the fuzzy set $\tilde{G}$ and, as above, $f(\mathbf{x})$ is the expected value of the performance measure associated with the target, $\mathbf{x}$ is the vector of N controllable parameters, and $\mathcal{X}$ is the closed set of constraints on $\mathbf{x}$.

MRSO has its corresponding problem that incorporates vagueness, namely the *fuzzy multiple response simulation optimization* or *fuzzy multicriteria simulation optimization* (FMSO) problem:

$$\max_{\mathbf{x}\in\mathcal{X}} \left\{ \left(\mu_{\tilde{G}_1}(f_1(\mathbf{x})), \mu_{\tilde{G}_2}(f_2(\mathbf{x})), \ldots, \mu_{\tilde{G}_K}(f_K(\mathbf{x})) \right) \right\} \tag{1.4}$$

where for $k = 1, \ldots, K$, $\mu_{\tilde{G}_k}(f_k(\mathbf{x}))$ is the degree of satisfaction of the k-th vague target represented by the fuzzy set $\tilde{G}_k$; $f_k(\mathbf{x})$ is the expected value of the performance measure associated with the k-th target.

1.3 Scope and Objectives of Research

The primary objective of this research is to develop an efficient and robust method for the multicriteria optimization of simulated complex systems with vague goals. This method, which uses a fuzzy controller, incorporates existing knowledge, satisfies vaguely stated goals, generates a high quality approximate Pareto optimal set, and is efficient in terms of simulation runs.

To date, most of the research in simulation optimization has been focused on single response optimization on the continuous space of input parameters $\mathcal{X}$. However, the optimization of more complex systems does not fit this framework. For instance, decision makers often face the problem of optimizing multiple performance measures of systems with both continuous and discrete input parameters. Additionally, previously acquired knowledge of the system by experts has been largely ignored by simulation optimization techniques proposed in the literature. These techniques do not provide any means of incorporating this valuable knowledge into the optimization engine. Furthermore, if the goals of the system design are stated in natural language or vague terms, current techniques are simply unable to deal with this imprecision. Our approach for simulation optimization will take into account the issues of continuous and discrete input parameters, multiple criteria, preexistent knowledge, and vagueness in the system goals.

The objectives of this research may be summarized as follows.

1. Develop methods to solve problems FSO in (1.3) and FMSO in (1.4).

2. Develop a procedure for the solution of simulation optimization problems by taking into consideration both continuous and discrete input parameters, pre-

viously acquired knowledge on the system, vagueness in the system goals, and considering multiple criteria.

3. Develop an alternative and competitive approach to unveil the strengths and weaknesses of the proposed approach in 2.

4. Conduct an extensive experimental performance evaluation of the proposed approaches on a Flow Line Design problem.

To validate the mechanisms proposed to achieve objectives 2 and 3, we use the popular trapezoidal fuzzy sets. To further improve the expressive power of the vague concepts used therein, we also

5. Investigate, formulate, and develop an efficient and flexible mechanism to represent virtually any vague concept expressed in natural language.

1.4 Organization of the Dissertation

In Chapter 2 we include a survey of previously proposed methods for simulation optimization with single and multiple responses. We briefly review the elements of soft computing and emphasize the existing multicriteria evolutionary optimization techniques found in the literature. Chapter 3 presents a method, based on a fuzzy controller, for solving problems FSO in (1.3) and FMSO in (1.4). We illustrate the method on a Flow Line Design problem. In Chapter 4 we propose an alternative evolutionary approach to the fuzzy controlled mechanism presented in Chapter 3. This evolutionary approach, which we call Optimal Scoring Evolutionary Algorithm (OSEA), provides a means to assess the quality of the approximate Pareto front generated by the fuzzy controlled approach to FMSO. An extensive comparison with the fuzzy controlled mechanism on the Flow Line Design problem is presented. Chapter 5 presents an efficient and flexible Bézier curve–based mechanism for constructing membership functions of convex normal fuzzy sets. This mechanism is useful for representing virtually any kind of vague concept. Chapter 6 contains the conclusions from our research and recommendations for future study.

Chapter 2

Literature Review

In the first part of this chapter, we summarize the existing alternatives to simulation optimization. The second part, deals with an overview of soft computing with emphasis on the existing literature on evolutionary multicriteria optimization.

2.1 Simulation Optimization

The dramatic improvement in computer technology, its relatively low cost, and broad availability, have led researchers in industry and academia to increased efforts in the area of simulation optimization of complex systems. Even though a lot of research has been conducted in the last twenty–five years, many problems remain open and unsolved. Due to these loose ends and the high impact on industry, the area keeps growing and attracting researchers and practitioners year after year.

Several reviews on the field of simulation optimization are available [3–5, 19, 39, 45, 52, 70, 90, 111, 119]. The one written by Fu [45] presents an excellent overview of the field. The main forum for simulation researchers and practitioners is the Winter Simulation Conference (WSC). Its proceedings are a valuable source to keep up to date on progress in the area.

In the remaining part of this section, we present the different methods that are used to solve simulation optimization problems. Sections 2.1.1 to 2.1.5 refer to single–response systems while Section 2.1.6 refers to multiple–response systems.

2.1.1 Optimization Over a Finite Set

The methods discussed here are useful when the number of choices is not too large, say 2 to 20 [55]. They are basically statistical procedures that fall in one of two categories: multiple comparison or ranking–and–selection.

2.1.1.1 Multiple Comparison Procedures

The purpose of these procedures is the construction of confidence intervals based on pairwise comparisons. Some of the methods in this category are the *all–pairwise* comparisons based on paired–t confidence intervals and the Bonferroni inequality; *multiple comparisons with the best* (MCB); and *all–pairwise multiple comparisons* (MCA) approach. The typical assumptions for these methods are independence and normality of the performance measure of interest. For further details on this subject the reader is referred to Goldsman and Nelson [55] and Law and Kelton [81].

2.1.1.2 Ranking and Selection

Ranking and selection methods are statistical procedures designed to select the best system or a subset of systems that include the best one, from a finite set of choices. Provided some assumptions are met, these methods guarantee that the correct selection is made with at least a (user specified) probability. These methods can be classified in two major categories: indifference zone and subset selection. The method proposed by Dudewicz and Dalal [34] falls into the first category, while that proposed by Sullivan and Wilson [123] falls into the subset selection methods category. Both procedures are particularly useful for simulation optimization because they do not require variances to be equal or to be known. Again, for further details on this subject the reader is referred to Goldsman and Nelson [55] and Law and Kelton [81].

2.1.2 Response Surface Methodology

Perhaps the most used technique in simulation optimization is response surface methodology (RSM). RSM techniques can be categorized into metamodels and sequential procedures. In the metamodel methodology, a regression model is fitted to the response of interest after evaluating it through the use of simulation at several values

of input parameters. Once a regression model is estimated, it is treated as a deterministic function and is optimized [19].

In the literature, simulation optimization using RSM usually refers to the second category, i.e., sequential procedures. The basic idea is to approach the vicinity of the optimum through a sequence of first order regression models. Once an optimum neighborhood is reached, higher order regression models are used. The optimum is derived analytically.

The major disadvantage of RSM is that a large number of simulation runs may be needed [70]. On the other hand, RSM has a strong theoretical basis and can be applied to a broad variety of simulation problems [140].

For further details on this subject the reader is refer to the overview written by Kleijnen [77], the review written by Jacobson and Schruben [70] and the classical books by Myers and Montgomery [93] and Box and Draper [16].

2.1.3 Gradient Based Algorithms

These algorithms are based on improving the input parameters by moving iteratively in the direction of the estimated gradient of the response of interest. One of the major concerns with this type of algorithm is the estimation of the gradient and its statistical properties.

2.1.3.1 Stochastic Approximation

The first stochastic approximation algorithms were introduced by Robbins and Monro [109], and Kiefer and Wolfowitz [75]. The basic idea is that the single response simulation optimization problem presented in (1.1) can be solved by finding a vector $\mathbf{x}$ of input parameters such that

$$\nabla f(\mathbf{x}) = 0. \tag{2.1}$$

The general form of the stochastic approximation algorithm is:

$$\mathbf{x}_{n+1} = \Pi_{\Theta}(\mathbf{x}_n - \alpha_n \hat{\nabla} f(\mathbf{x}_n)) \tag{2.2}$$

where $\mathbf{x}_n$ is the vector of input parameters at the n-th iteration, $\hat{\nabla} f(\mathbf{x}_n)$ is an estimate of the gradient $\nabla f(\mathbf{x}_n)$ from iteration n, α_n is a positive step size, and Π_{Θ} is a projection onto the set of continuous input parameters Θ.

If $\hat{\nabla} f(\mathbf{x}_n)$ is an unbiased estimator of the gradient $\nabla f(\mathbf{x}_n)$, the algorithm is of the Robbins–Monro [109] type. On the other hand, if the gradient is estimated by finite differences (as described in Section 2.1.3.2.1 below), the algorithm is of the Kiefer–Wolfowitz [75] type. Under fairly general conditions they converge almost surely to the optimum. However, they could converge to a local optimum and may not always work well [3]. Recently new algorithms have been proposed to improve some of the weaknesses of these classical algorithms. A thorough review of new methods on stochastic approximation can be found in [3] and [4].

2.1.3.2 Gradient Estimation Techniques

Naturally, the heart of gradient–based algorithms is the technique used to estimate the gradient. Here we present the most common methods used in the simulation optimization literature. For further details the reader is referred to [82].

2.1.3.2.1 Finite Differences This method is the simplest and the most commonly used. The gradient at $\mathbf{x}$ at the n-th iteration is estimated as follows:

$$\hat{\nabla} f(\mathbf{x}_n) = [\hat{\nabla}_1 f(\mathbf{x}_n), \ldots, \hat{\nabla}_p f(\mathbf{x}_n)]^T, \tag{2.3}$$

where,

$$\hat{\nabla} f_i(\mathbf{x}_n) = \frac{\hat{f}(\mathbf{x}_n - c_n \mathbf{e}_i) - \hat{f}(\mathbf{x}_n + c_n \mathbf{e}_i)}{2c_n} \tag{2.4}$$

is used for the finite–difference gradient estimator using *central differences* while

$$\hat{\nabla} f_i(\mathbf{x}_n) = \frac{\hat{f}(\mathbf{x}_n + c_n \mathbf{e}_i) - \hat{f}(\mathbf{x}_n)}{c_n} \tag{2.5}$$

is used for the finite–difference gradient estimator using *forward differences*. In (2.4) and (2.5) $\mathbf{e}_i$ represents the i-th unit vector. The total number of simulations needed to estimate the gradient is $2p$ for the central differences estimator and $p+1$ for the forward differences estimator. The method of finite–differences has some known problems related to slow convergence, and bias and large variance of the gradient estimate [3].

2.1.3.2.2 Perturbation Analysis Perturbation analysis (PA) is an approach of using *sample path* analysis for gradient estimation. Perhaps the most common technique in this class is *infinitesimal perturbation analysis* (IPA). The main principle behind IPA is that if an input parameter is perturbed by an infinitesimal amount, the sensitivity of the output to that parameter can be estimated by tracking its effect through the system. A basic requirement of IPA is that these small perturbations should not cause changes in the sequence of events. Unfortunately, for complex systems this requirement is very difficult to guarantee. One strength of this technique is that the gradient can be estimated by making just one simulation run. For more information the reader is referred to [52].

2.1.3.2.3 Likelihood Ratio Method The Likelihood ratio (LR) method is also known as the *score function* (SF). With this method the gradient is estimated by expressing the derivative of the expected value of the response with respect to an input parameter as the expected value of a function of input and simulation parameters. This value is recorded in the simulation run for a future estimation of the gradient. This method requires only one simulation run to estimate the gradient and is more generally applicable than IPA. A weakness of LR is that it may produce gradient estimates with larger variance than those obtained through IPA [3, 45].

2.1.3.2.4 Frequency Domain Experimentation The basic idea behind *frequency domain experimentation* (FDE) is to explore the sensitivity of the responses by sinusoidal oscillations of the value of the input parameters during the simulation. Initially, this method introduced by Schruben and Cogliano [117], was intended for use in factor screening (i.e., identifying the relevant input parameters in a simulation study). The input parameters are modulated as follows:

$$\mathbf{x}(t) = \mathbf{x}_0 + \gamma \sin(\tilde{w}t) \tag{2.6}$$

where $\mathbf{x}_0$ is the vector of input parameters, γ is the vector of oscillation amplitudes, $\tilde{w}$ is the vector of distinct oscillation frequencies, and t is the oscillation time index (different from the simulation time).

While FDE has the desired property of being able to estimate the gradient in just one simulation run, it has the problem of having to determine the oscillation index, frequencies, and amplitudes (see (2.6)). A possible way to overcome this problem by making changes in the event–generation code has been suggested in [45].

2.1.4 Derivative–free Methods

As the name of the section suggests, these methods do not move in the direction of the estimated gradient. With fewer requirements and assumptions than the methods described above, these techniques and their variants can be used for simulation optimization with discrete input parameters.

2.1.4.1 Nelder–Mead Based Methods

These methods are based on the classical algorithm for unconstrained nonlinear programming proposed by Nelder and Mead [95]. Basically $p + 1$ vertices forming a simplex in the p-dimensional space are maintained throughout the algorithm. The algorithm proceeds by continuously replacing the worst vertex. The replacement is found by moving in the *reflection direction*; i.e., in the negative of the direction defined by the vector formed by the difference between the simplex centroid and the worst point in the simplex (point which is being dropped). Several authors have proposed different implementations for simulation optimization based on this classical algorithm [8, 9, 66]. Box [17] proposed a constrained version of the Nelder–Mead algorithm called *complex search*. A modified version of Box's algorithm for simulation optimization can be found at [6].

2.1.4.2 Simulated Annealing

Simulated annealing is an iterative stochastic search method, analogous to the physical annealing process in which material is cooled down until a minimum level of energy is achieved. This method generates a sequence of solutions with a decreasing trend but not always decreasing response (in the minimization case). By allowing *hill–climbing* behavior, the possibility of being trapped in a local minima is reduced. This method was proposed for deterministic optimization by Kirkpatrick et al. [76]. Several algorithms for simulation optimization have been based on this approach [2, 46, 57]. Under suitable conditions, converges almost surely to the optimal solution.

2.1.5 Other Methods

Norkin et al. [96] proposed a method for discrete simulation optimization based on the classical *branch and bound* integer programming technique. Glover et al. have used scatter search and tabu search as the primary engine of their commercial package OptQuest [51]. Nozari and Morris [98] have proposed a modification of the classical algorithm of Hooke and Jeeves [62]. Healy and Schruben [60] proposed what is called retrospective simulation response optimization that can be seen as the dual of metamodeling [45]. Finally, genetic algorithms and evolutionary strategies have also been proposed. These are discussed in Section 2.3 under the larger topic of soft computing.

2.1.6 Multicriteria Simulation Optimization

In practice, when a simulation model is used, most of the time the analyst has to consider more than one criterion simultaneously. Despite of this fact, most of the research in the field has been done in the area of single response simulation optimization.

Montgomery and Bettencourt [92] used the Geoffrion, Dyer, Feinberg method (GDF) to optimize a simulation model with four criteria and two input parameters. A goal programming approach was used by Clayton et al. [25] and Rees et al. [106]. Biles and Swain [11, 12] proposed and compared a first and second order RSM approach with a direct search algorithm. Evans et al. [36] survey the area of multicriteria simulation optimization.

2.2 Soft Computing

To get a better understanding of what this new term of *soft computing* really means, let us quote Lotfi A. Zadeh, father of fuzzy logic and one of the leaders in the soft computing community:

> "In traditional –hard– computing, the prime desiderata are precision, certainty and rigor. By contrast, the point of departure in soft computing is the thesis that precision and certainty carry a cost and that computation, reasoning, and decision making should exploit –wherever possible– the tolerance for imprecision and uncertainty" [146].

The following definition of soft computing is also given by Zadeh [73]:

> "Soft computing is an emerging approach to computing which parallels the remarkable ability of the human mind to reason and learn in an environment of uncertainty and imprecision."

Soft computing is an association of methodologies that mainly brings together fuzzy logic, evolutionary computing, neurocomputing and probabilistic computing. An essential aspect of soft computing is that these methodologies are complementary rather than competitive or exclusive [147].

The remaining part of this section describes briefly the function of those soft computing methodologies that will be used in the scope of our research on simulation optimization via soft computing.

2.2.1 Fuzzy Logic

Fuzzy logic was invented in the sixties by Lotfi A. Zadeh [144], who being an expert in control engineering, realized that control theory was unable to solve many complex real system problems. In a narrow sense, fuzzy logic can be viewed as a logical system that aims at a formalization of approximate reasoning. In a broad sense, fuzzy logic is used as a synonym for fuzzy set theory. Fuzzy set theory has several branches such as fuzzy arithmetic, fuzzy mathematical programming, fuzzy topology, fuzzy graph theory, fuzzy data analysis, and fuzzy logic, among others [146].

The contribution of fuzzy logic[1] to the area of soft computing is to introduce flexibility in classification, querying and problem solving, and to capture imprecision when there is lack of information [33].

In our context, fuzzy logic brings an effective way of compressing and representing knowledge through the use of *linguistic variables*, *linguistic values*, and *fuzzy if–then–rules*.

2.2.2 Neurocomputing

Fuzzy logic does not have adaptation or learning features, since it lacks the mechanism to extract knowledge from existing data. On the other hand, this is the nature of

[1] From this point the term *fuzzy logic* is used in its broad sense unless otherwise expressed

neurocomputing and is what it brings to the soft computing arena. Neural networks provide an efficient technique able to learn from examples of input–output pairs.

In our context, *learning* will refer to tuning a fuzzy controller's linguistic terms and the construction of the fuzzy if–then–rules. An example of a hybrid system that uses neural networks to tune a fuzzy logic system is the work on Adaptive Neural Fuzzy Inference Systems (ANFIS) [72, 73].

Bishop [13] presents a comprehensive treatment of neural networks. Jang et al. [73] and Lin et al. [83] cover neurofuzzy systems.

2.2.3 Evolutionary Computing

Evolutionary computing provides to soft computing an efficient mechanism for solving difficult problems through a systematic stochastic search based on the principles of natural selection. There have been several schools of thought that have contributed to and enriched the field, but share the same underlying principles, i.e., evolutionary strategies [105, 118], evolutionary programming [40], and genetic algorithms [61].

Evolutionary–based algorithms have been applied to a variety of problems, many of which conventional methods have failed to solve. For instance, in soft computing, the process of extracting knowledge for the fuzzy logic inference system requires the solution of optimization problems which are often nonlinear and combinatorial. Evolutionary–based algorithms can effectively solve these and other hard problems.

The basic idea of an evolutionary algorithm is to simulate the natural selection process and obtain better individuals as the algorithm progresses. The evolutionary algorithm maintains a population of *individuals* (or *chromosomes*), in which each individual *represents* a potential solution to the problem. The *representation* is the mapping of solutions to individuals. The form and complexity of the representation is problem dependent. As the algorithm progresses, the population of individuals evolves through successive iterations, called *generations*. In every generation, each individual is *evaluated* and assigned a measure of its *fitness* for survival. New individuals for the next generation are generated by combining and altering members of the population through *genetic operators* or transformations. A common unary transformation is *mutation*, in which new individuals are created by applying modifications to a single individual. Higher order transformations, such as *crossover*, are also a source of new individuals. In crossover, several parents are combined to generate one or more

children. The population for the new generation is formed by *selecting* the more fit among all individuals. After several generations, the algorithm converges to a good population (i.e., good solutions), and possibly, to the best individual representing the "optimum". Good introductory material can be found in the books authored by Michalewicz [91] and Gen and Cheng [48].

2.2.3.1 Multicriteria Evolutionary Optimization

Evolutionary algorithms are well suited for exploring a vast set of alternatives, partially because they are based on evolving (a *population* of) solutions in parallel [150]. Contrary to classical mathematical programming techniques, evolutionary algorithms can be designed to search for the entire set of Pareto optimal solutions in a single run and do not make assumptions about the shape and mathematical properties (e.g., continuity) of the Pareto front [29]. Moreover, there are few, if any, competitive alternatives to multicriteria optimization, and even fewer methods available that tolerate noisy and uncertain objective functions [63].

Since the pioneering work of Schaffer [114] on the Vector Evaluated Genetic Algorithm (VEGA), a substantial amount of research has been conducted in the area of evolutionary multicriteria optimization [2]. Two recent reviews have surveyed the area of evolutionary algorithms for multicriteria optimization [29, 134]. Other surveys are [42, 63, 129]. An annotated bibliography by Ehrgott and Gandibleux [35] concentrates on multicriteria combinatorial optimization.

In the next sections, we classify and review various evolutionary algorithms applied to multicriteria optimization.

2.2.3.1.1 Aggregation Approaches

This is perhaps the most natural and common approach for fitness assignment [29, 63]. For a given individual, the values of the multiple criteria are combined into a single scalar using a linear or nonlinear combination. The main strength of this approach is its computational efficiency and simple implementation. Its main weakness is the difficulty to determine the value of the weights that reflect the relative importance of each criterion. Daas and Dennis [31] have commented why a weighted

[2] A list of references on evolutionary multicriteria optimization is available at: http://www.lania.mx/ ccoello/EMOO/EMOObib.html

sum approach does not work properly when the shape of the Pareto front is not convex, regardless of the weights used. However, Daas and Dennis' problem setting is somewhat restrictive, with continuity and differentiability requirements.

Several applications of evolutionary algorithms using aggregation approaches have been reported. A number of authors have provided examples of the use of the common method known as *weighted–sum approach* [10, 71, 74, 85, 127, 135, 142]. Gen et al. [47, 48] have extended this approach to handle uncertainty using fuzzy logic. Medaglia and Fang [88] have proposed the use of adaptive weights instead of predetermined fixed weights. Hajela and Lin [58] have used an evolutionary approach (HLGA) in which the weights are discretized and encoded in the chromosome. Some researchers have proposed a nonlinear aggregative method, closely related to goal programming [21], called *distance–to–target* approach [112, 139]. Goal attainment is a related technique that seeks to minimize the weighted difference between criteria values and the corresponding goals [141]. Treating criteria threshold constraints by means of penalty functions can be seen as another aggregation approach used by several researchers [54, 86, 104, 107, 121]. Wallace [138] proposes the use of a decision maker's *probability of acceptance* function for each criterion, with the probability of simultaneous acceptance being obtained by multiplication.

2.2.3.1.2 Non Pareto–based Approaches

In his pioneering work on evolutionary multicriteria optimization, Schaffer [114, 115] proposed the *Vector Evaluated Genetic Algorithm* (VEGA). For a problem with K criteria, the population size is equally divided in K subpopulations. The selection mechanism is applied to each subpopulation using the corresponding criterion. Then, the subpopulations are shuffled together to obtain the main population, where crossover and mutation are applied in the usual way. This method was the first evolutionary approach developed to generate and search for the Pareto optimal set in a single run. Because this technique selects individuals who excel in one criterion, without considering the other criteria, a problem known as *speciation* may occur. Individuals with *middling* performance (i.e., acceptable performance in all dimensions) which are desirable from a decision maker's point of view, are simply not selected due to their failure to excel in at least one criterion. Several researchers have applied and proposed modifications of VEGA to different domains [108, 125, 126, 129, 130].

Fourman [44] suggested a selection scheme known as *lexicographic ordering*. In this

selection mechanism, criteria are assigned different priorities. Selection is performed by comparing pairs of individuals according to the criterion with the highest priority. If this results in a tie, then the criterion that follows in the priority list is used, and so on. Fourman also proposed, as a variation of this scheme, to randomly select the criterion to be used for comparison. Kursawe [79] proposed a multicriteria version of evolution strategies [118] based on lexicographic ordering. As with VEGA, all of these approaches experience speciation. Using an aggregation technique with random weights, Ishibushi and Murata [69] claim to generalize Kursawe's method and avoid speciation.

In the spirit of VEGA, the use of genders has been proposed as yet another way of defining subpopulations for each criterion. In a bicriteria optimization problem, Allenson [1] proposed a VEGA–like algorithm that associates each criterion with a gender. Lis and Eiben [84] extended this concept to multiple genders (i.e., multiple criteria) and used *panmictic* reproduction (i.e., several parents generate a single child). These gender–based methods impose mating restrictions at crossover.

Other non Pareto–based approaches have been proposed. Motivated by game theory, Périaux et al. [100] proposed an evolutionary algorithm based on the concept of Nash equilibrium [94]. Some researchers have used the concept of *min–max optimum*, which compares relative deviations from separately attainable minima [26–28, 58, 99]. Valenzuela and Uresti [133] proposed a method based on *learning classifier systems*.

2.2.3.1.3 Pareto–based Approaches

Pareto–based fitness assignment was first proposed by Goldberg [54]. The idea is to rank the population according to Pareto optimality. First, the nondominated individuals are given rank one and then removed from the population. The newly nondominated individuals are given rank two and then removed, and so on. Goldberg also suggested niching and speciation methods to promote and maintain subpopulations along the Pareto front.

In Fonseca and Fleming's [41, 43] Multi–objective Genetic Algorithm (MOGA) the individual's rank corresponds to the number of individuals in the current population by which it is dominated. After sorting the population according to the ranks, fitness is assigned by interpolating from the best to the worst individuals in the population. MOGA also uses *fitness sharing* [53] within a rank, such that the individuals are further ranked according to their fitness sharing *niche counts*. The niche count is a

measure of the individual's neighborhood crowding. In MOGA selection is performed with *stochastic universal sampling* [7]. The main strengths of MOGA are its efficiency and relatively easy implementation. Its main weakness is that its performance is highly dependent on the *sharing factor*.

In Horn and Nafpliotis' [64, 65] Niched Pareto Genetic Algorithm (NPGA) a selection scheme based on *Pareto domination tournaments* is used. To determine the dominance status of two competing individuals, a sample of (typically about 10) other individuals from the current population is drawn. If one of the two individuals is dominated by a member of the sample, while the other is not dominated, then the nondominated individual wins the tournament. If both or neither are dominated, then the result of the tournament is resolved by selecting the individual with the lower niche count. The main strength of this method is that it is very fast because does not apply Pareto selection to the entire population. Its main weakness is that it requires tuning of the sharing parameter and tournament sample size.

Srinivas and Deb [120] proposed the Non–dominated Sorting Genetic Algorithm (NSGA). NSGA follows Goldberg's original idea on Pareto–based ranking very closely [54]. In NSGA fitness sharing is done in the parameter value space, calculating distances between vectors in the solution space rather than in criteria space. The main strength of NSGA is that sharing is performed in the solution space, allowing the algorithm to discover multiple solutions and potentially generating an even distribution of the Pareto front. Some researchers [29] have reported that NSGA is highly sensitive to the sharing parameter and could be computationally expensive.

Zitzler and Thiele [150] have proposed the Strength Pareto Evolutionary Algorithm (SPEA). SPEA uses a secondary population of nondominated solutions in the fitness assignment procedure. A solution in the population is assigned a fitness value according to the number of vectors in the secondary population that dominates its corresponding criteria vector. For computational efficiency, SPEA uses a clustering procedure to reduce the size of the nondominated set while preserving its distribution.

Zitzler et al. [149] have compared several evolutionary algorithms using a comprehensive set of complex test functions. For their chosen test problems and parameter settings, they found a clear hierarchy of algorithms in terms of the distance to the theoretical Pareto optimal front. Sorting the algorithms from the best to the worst, they found three tiers: SPEA is in the first tier; NSGA is in the second tier and; VEGA, HLGA, NPGA, and MOGA, are in the third tier. Furthermore, note that

the only aggregation method considered is HLGA [58] and the only non–Pareto based approach is VEGA [114].

2.3 Simulation Optimization and Soft Computing

We have seen that simulation optimization is a very complex problem that has been treated by different approaches (see Section 2.1). We strongly believe that the idea of bringing soft computing methodologies into the area of simulation optimization will lead to the solution of real world system problem in an efficient manner.

To our knowledge, there is no study in the field of simulation optimization that combines the search capabilities of genetic algorithms with the learning ability of neural networks and the knowledge compression ability of fuzzy logic. The absence of such a study combined with the synergistic view of soft computing is one of the motivations of this research.

The use of genetic algorithms in simulation optimization has been reported in the literature [14, 15, 59, 131, 143] merely as a random search technique working in isolation. Of special interest is the work conducted by Boesel and Nelson [14, 15] at Northwestern University who have tried to provide statistical guarantees on the quality of the solution obtained when applying the genetic algorithm.

The use of neural networks in the field of simulation optimization has been reported by Glover et al. [51]. Basically they give the user the option of engaging a neural network accelerator to help their search engine in the screening of some values of the input parameter vector.

Chapter 3

A Fuzzy Controlled Approach

Simulation optimization deals with finding the values of input parameters of a complex simulated system which result in desired output. Traditional techniques may require an enormous amount of simulation runs to evaluate the system. To alleviate this problem, the approach proposed in this chapter provides the means of incorporating knowledge, expressed in natural language, that is often available among analysts and decision makers. Using convenient linguistic representations, the proposed mechanism can satisfy vaguely stated goals to a high degree (e.g. "high utilization" or "low inventory"). This mechanism is also able to generate an approximate Pareto optimal set in the presence of multiple goals. The optimization strategy used here depends on a fuzzy controller guided by a set of rules derived from statistical concepts, response surface models, and experts' knowledge. To illustrate this approach we present computational experiments on the design of a flow line manufacturing system (in terms of a tandem of queues with blocking) with one and two goals.

3.1 Introduction

Classical simulation optimization research has been primarily concerned with adapting classical mathematical programming techniques to primarily solve problem (1.1) and, to a lesser extent, problem (1.2). Chapter 2 provides an overview of several of these methods.

Unfortunately, valuable knowledge is put aside and not incorporated into the optimization process simply because those techniques cannot handle words as a com-

puting element. In particular, the following two key issues were missed by classical simulation optimization research:

1. *Vague targets.* Decision makers typically state their aspiration levels associated with the system performance measures in vague manner. For instance, in a manufacturing facility a *high* service level can become the driving goal, while having a *low* cell loss can become the driving direction for a simulation study of an ATM network. The vague terms *high* and *low* used by the decision maker should be directly incorporated into the analysis and the optimization strategy directed to satisfy such targets. Furthermore, these targets may involve multiple criteria (e.g. *high* service level and *low* work in process).

2. *Knowledge.* Despite of the fact that knowledge is often expressed by rules using natural language, the classical approaches to simulation optimization do not provide any mechanism to incorporate this type of information. For instance, in the manufacturing setting it is possible to come up with rules such as "if the service level is low then the factory production rate should be increased by a large amount". Unfortunately knowledge expressed in terms of rules is not included in any classical optimization strategy.

In this chapter, we propose a new mechanism for simulation optimization based on fuzzy control that enhances existing optimization strategies by incorporating vague targets and knowledge expressed in rules in an efficient and natural way. We are interested in finding the values of the input parameters of a simulation model such that all the objectives are satisfied to a high degree.

The chapter is organized as follows. In Section 3.2, we propose a fuzzy controlled simulation optimization framework. Section 3.3 describes how the rules in the fuzzy controller are designed. In Section 3.4, a flow line design problem in a manufacturing setting is presented and used to illustrate our approach. Computational experiments with one and two goals are presented. Finally, conclusions are given in Section 3.5.

3.2 Proposed Approach

Figure 3.1 shows the basic idea of our proposed approach. The system to be optimized is modeled using a configurable *simulator*. The M *performance measures* of the

simulator are represented by *linguistic variables* and compressed using fuzzy sets (*linguistic values*) by means of a *fuzzification* module. The fuzzified performance measures become the inputs or state variables for a *fuzzy controller*, the core of our optimization strategy. The fuzzy controller has a *knowledge base* composed of a *linguistic data base* and a *rule base* (S rules). Based on the simulation performance measures (state variables), the system consults the knowledge base and the fuzzy controller determines the adjustments to be made to the simulation input parameters. These adjustments are expressed in terms of fuzzy sets and need to be converted into numbers through the *defuzzification* module. The defuzzified adjustments are used to update the N *simulation input parameters* and then a new simulation model is obtained. This cycle (iteration) continues until specified performance *targets* for the simulated system are satisfied to a high degree. In this section we describe these elements in more detail.

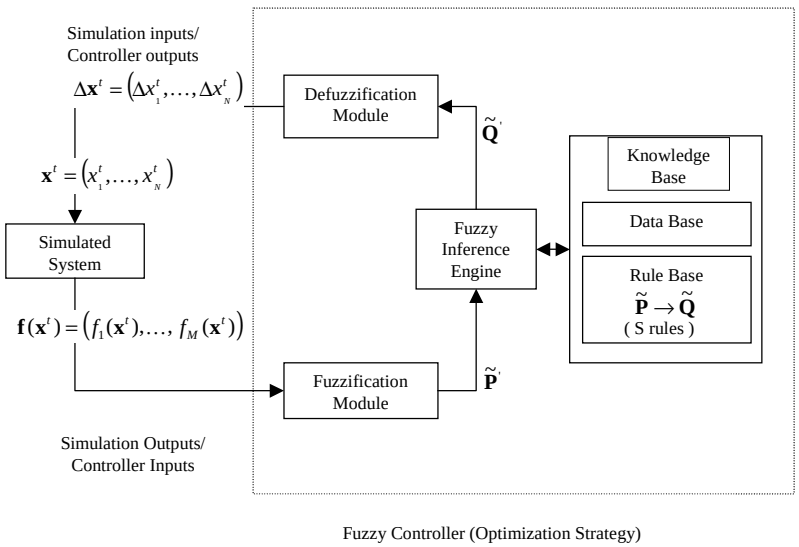


Figure 3.1: Fuzzy controlled simulation optimization

3.2.1 Simulator

The simulator is the component that contains the discrete event simulation model. Depending on the system that is being modeled, different inputs can be controlled and several responses can be measured.

Let $\mathbf{x} = (x_1, \ldots, x_N)$ be the N-dimensional vector of input parameters, with $\mathbf{x} \in \mathcal{X}$ and $x_j \in X_j$, for $j = 1, \ldots, N$. This vector should be chosen in such a way that only inputs relevant to the performance measures of interest are considered. For specific applications, it may be helpful to have some factor screening procedure to identify the relevant inputs before the controller is designed [77, 81, 117].

Let $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_M(\mathbf{x}))$ be the M-dimensional vector of average system performances measures, with $\mathbf{f}(\mathbf{x}) \in \mathcal{Y}$ and $f_i(\mathbf{x}) = y_i \in Y_i$, for $i = 1, \ldots, M$. These measures should be easily collectable and retrievable after the simulator completes a batch of replications.

The simulator also has to be easily configurable to allow the adjustment of the structural parameters, such as the number of replications, run length, and random seeds.

3.2.2 Fuzzy Controller

When a simulation model is defined, the M inputs (or state variables) and N outputs (or control variables) of the controller are identified. Figure 3.1 shows the correspondence between the outputs of the simulation model and the inputs of the controller, and the inputs of the simulation model with the outputs of the controller.

Our controller uses the concept of a *linguistic variable* in order to express natural language or imprecise information. The approximate values of the variable are known as *linguistic terms*. Fuzzy sets provide a convenient way to represent the linguistic terms that refer to a base variable whose values range over a universe of discourse. When the linguistic terms are expressed by fuzzy sets, the membership functions capture the meaning of each term. Once the inputs and outputs of the controller are identified, we have to select meaningful linguistic values for each linguistic variable. For instance, the simulation optimization of a manufacturing system may have a performance measure called *utilization*. As a linguistic variable, utilization could be compressed into the terms low, medium and high, with each membership function defined over the universe of discourse $X = [0\%, 100\%]$.

In a fuzzy controller, knowledge is stored in the form of *fuzzy inference rules*. Our approach uses rules of the following form [87]:

$$\textbf{\textit{if}}\ p_1 \text{ is } \tilde{P}_1 \text{ and } \ldots \text{ and } p_M \text{ is } \tilde{P}_M\ \textbf{\textit{then}}\ q_j \text{ is } \tilde{Q}_j \tag{3.1}$$

where p_i is a state linguistic variable with its corresponding linguistic value $\tilde{P}_i$ defined over the universal set Y_i (for $i = 1, \ldots, M$), and q_j is a linguistic control variable with its corresponding linguistic value $\tilde{Q}_j$ defined over the universal set ΔX_j, $j = \{1, \ldots, N\}$.

The fuzzification interface establishes a mapping between observed average values of performance measures coming from the simulator and fuzzy sets defined in the universe of the corresponding variables. Once these state variables are fuzzified they become inputs for the fuzzy controller.

For $i = 1, \ldots, M$ and $r = 1, \ldots, S$ we define the following mapping:

$$\tilde{P}'_{ri} = F_r(f_i(\mathbf{x}^t)) \tag{3.2}$$

where, $\mathbf{x}^t$ is the vector of simulation inputs for the t-th iteration (a full cycle in Figure 3.1), $\tilde{P}'_{ri}$ is a fuzzy set associated with the r-th rule and the observed i-th average performance measure $f_i(\mathbf{x}^t)$ defined over the universal set Y_i, and $F_r(\cdot)$ is a fuzzification function.

We use a special case of the fuzzification function called *singleton* fuzzification [78]. This function constructs a fuzzy set $\tilde{P}'_{ri}$ as follows:

$$\mu_{\tilde{P}'_{ri}}(f_i(\mathbf{x})) = \begin{cases} 1, & \text{if } f_i(\mathbf{x}) = f_i(\mathbf{x}^t) \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

Probably the most fundamental rule in logic is the rule of *Modus Ponendo Ponens*, more familiarly known as *Modus Ponens*. Modus Ponens (MP) states that if we have a conditional (rule) and a known *antecedent* (fact), then we can infer the *consequent* (conclusion). To allow similar inference with linguistic variables, Zadeh [145] proposed an extension of the classical Modus Ponens called the Generalized Modus Ponens (GMP).

For a single rule with one antecedent and one consequent. GMP can be written as

Rule	if p is $\tilde{P}$ then q is $\tilde{Q}$
Fact	p is $\tilde{P}'$
Conclusion	q is $\tilde{Q}'$

where $\tilde{P}$, $\tilde{Q}$, $\tilde{P}'$, and $\tilde{Q}'$ are fuzzy sets and $\tilde{P}'$ is close to $\tilde{P}$ and $\tilde{Q}'$ is close to $\tilde{Q}$.

The fuzzy rule "*if p is* $\tilde{P}$ *then q is* $\tilde{Q}$" represents a *fuzzy relation* between $\tilde{P}$ and $\tilde{Q}$. The fuzzy relation $\tilde{P} \longrightarrow \tilde{Q}$ is expressed by a fuzzy set $\tilde{R}$ defined on the space $Y \times \Delta X$.

Different definitions of union, intersection, and complement, lead to different ways to express the *fuzzy implication* $\tilde{P} \longrightarrow \tilde{Q}$. The standard intersection and union of fuzzy sets (*t-norm* and *t-conorm*, respectively) are:

$$\mu_{\tilde{A}\cap\tilde{B}}(u) = \min(\mu_{\tilde{A}}(u), \mu_{\tilde{B}}(u)) \tag{3.4}$$

$$\mu_{\tilde{A}\cup\tilde{B}}(u) = \max(\mu_{\tilde{A}}(u), \mu_{\tilde{B}}(u)) \tag{3.5}$$

In particular, using (3.4), the *Mamdani implication* [87] is:

$$\mu_{\tilde{R}_m}(y, \Delta x) = \min(\mu_{\tilde{P}}(y), \mu_{\tilde{Q}}(\Delta x)), \text{ for } y \in Y, \Delta x \in \Delta X \tag{3.6}$$

To complete the fuzzy inference engine we need a mechanism to derive the membership of the consequent (i.e., $\mu_{\tilde{Q}'}(\Delta x)$ for each $x \in \Delta X$), once the fact is known (i.e., $\mu_{\tilde{P}'}(y)$ for each $y \in Y$). The most commonly used mechanism is the *compositional rule of inference* (CRI) proposed by Zadeh [145]:

$$\mu_{\tilde{Q}'}(\Delta x) = \max_{y\in Y}\Big(T(\mu_{\tilde{P}'}(y), \mu_{\tilde{R}}(y, \Delta x))\Big), \text{ for } \Delta x \in \Delta X \tag{3.7}$$

where T is a t-norm.

Choosing Mamdani's implication operator $\tilde{R}_m$ defined in (3.6), the standard fuzzy intersection in (3.4) as the t-norm for T, and the fuzzy singleton fuzzification in (3.3), the CRI reduces to the following expression for the single rule with one antecedent and one consequent:

$$\mu_{\tilde{Q}'}(\Delta x) = \min(\mu_{\tilde{P}}(f(\mathbf{x}^t)), \mu_{\tilde{Q}}(\Delta x)), \text{ for } \Delta x \in \Delta X, \tag{3.8}$$

where $\mu_{\tilde{P}}(f(\mathbf{x}^t))$ is called the *firing strength* of the rule when the performance level $f(\mathbf{x}^t)$ is obtained via the simulator.

For rules with multiple antecedents such as the one in (3.1), the consequent is obtained by generalizing the idea of the CRI. For $r = 1, \ldots, S$,

$$\mu_{\tilde{Q}'_{rj}}(\Delta x_j) = \min\Big(\min\left(\mu_{\tilde{P}_{r1}}(f_1(\mathbf{x}^t)), \ldots, \mu_{\tilde{P}_{rM}}(f_M(\mathbf{x}^t))\right), \mu_{\tilde{Q}_{rj}}(\Delta x_j)\Big) \text{ for } \Delta x_j \in \Delta X_j \tag{3.9}$$

where $\min(\mu_{\tilde{P}_{r1}}(f_1(\mathbf{x}^t)), \ldots, \mu_{\tilde{P}_{rM}}(f_M(\mathbf{x}^t)))$ is the "firing strength" of the r-th rule.

To aggregate the information obtained from these rules, a connective operator is needed. Normally, the fuzzy union is used as this connective. Using the standard fuzzy union presented in (3.5), the aggregate fuzzy set for input j, $j = 1, \ldots, N$, is given by

$$\mu_{\tilde{Q}'_j}(\Delta x_j) = \max\left(\mu_{\tilde{Q}'_{1j}}(\Delta x_j), \ldots, \mu_{\tilde{Q}'_{Sj}}(\Delta x_j)\right), \text{ for } \Delta x_j \in \Delta X_j \tag{3.10}$$

This aggregation procedure produces one fuzzy (adjustment) set for each simulation input.

The defuzzification mechanism maps the fuzzy sets obtained from the inference procedure into crisp adjustments in the values of the inputs for the simulator.

There are a number of different defuzzification methods used in practice [78]. Among these, we have chosen to use the *centroid of area* as our defuzzification mechanism. With this mechanism, for $j = 1, \ldots, N$,

$$\Delta x_j^t = \frac{\int_{\Delta x_j} \Delta x_j \mu_{\tilde{Q}'_j}(\Delta x_j) d\Delta x_j}{\int_{\Delta x_j} \mu_{\tilde{Q}'_j}(\Delta x_j) d\Delta x_j} \tag{3.11}$$

3.2.3 Handling Multiple Criteria

The simulator collects information from M performance measures, namely $f_1(\mathbf{x}), \ldots, f_M(\mathbf{x})$. The level of performance in the system is measured against a set of vague targets for $K \leq M$ of these, i.e.,

$$p_{i_1} \text{ should be } \tilde{P}_{i_1} \text{ and } \ldots \text{ and } p_{i_K} \text{ should be } \tilde{P}_{i_K}. \tag{3.12}$$

where, $i_k \in \{1, \ldots, M\}$ for $k = 1, \ldots, K$ are the indices of state variables used as vague targets and $\tilde{P}_{i_k}$ is the desired linguistic value for the k-th vague target p_{i_k}.

The degree of satisfaction of the k-th target is given by $\mu_{\tilde{P}_{i_k}}(\mathbf{x})$ for $k = 1, \ldots, K$ and its range is $[0, 1]$. A fully satisfied goal has value of 1.

Following Section 3.1, when $K = 1$ (i.e., a target is specified for only one performance measure) then we have an FSO problem, while when $K > 1$ we have a FMSO problem.

When dealing with different and conflicting goals simultaneously, and in the absence of a mathematical specification of the decision maker's utility function, our approach provides the decision maker with an approximate set of *Pareto* optimal solutions. We introduce the following multicriteria terminology.

Definition 1. For $\mathbf{x} \in \mathcal{X}$, $i_k \in \{1, \ldots, M\}$, and $k = 1, \ldots, K$, $\vec{\mu}(\mathbf{x}) \triangleq \left(\mu_{\tilde{P}_{i_1}}(f_{i_1}(\mathbf{x})), \ldots, \mu_{\tilde{P}_{i_K}}(f_{i_K}(\mathbf{x}))\right)$ is called a criterion vector, and $\mathcal{M} = \{\vec{\mu}(\mathbf{x}) | \mathbf{x} \in \mathcal{X}\}$ is called the criterion space.

Definition 2. A solution $\mathbf{x}^* \in \mathcal{X}$ is efficient or Pareto optimal if and only if there does not exist any $\mathbf{x} \in \mathcal{X}$ such that $\mu_{\tilde{P}_{i_k}}(f_{i_k}(\mathbf{x})) \geq \mu_{\tilde{P}_{i_k}}(f_{i_k}(\mathbf{x}^*))$, for $i_k \in \{1, \ldots, M\}$ and $k = 1, \ldots, K$, and $\mu_{\tilde{P}_{i_k}}(f_{i_k}(\mathbf{x})) > \mu_{\tilde{P}_{i_k}}(f_{i_k}(\mathbf{x}^*))$ for at least one i_k. The set of all Pareto optimal solutions is denoted by $\mathcal{P}^*$.

Definition 3. Let $\vec{\mu}(\mathbf{x}), \vec{\mu}(\mathbf{z}) \in \mathcal{M}$ be two criterion vectors. Then, $\vec{\mu}(\mathbf{x})$ dominates $\vec{\mu}(\mathbf{z})$ if and only if $\vec{\mu}_k(\mathbf{x}) \geq \vec{\mu}_k(\mathbf{z})$, for $k = 1, \ldots, K$, and $\vec{\mu}_k(\mathbf{x}) > \vec{\mu}_k(\mathbf{z})$ for at least one k. The notation is $\vec{\mu}(\mathbf{x}) \succ \vec{\mu}(\mathbf{z})$.

Definition 4. Let $\vec{\mu}^* \in \mathcal{M}$. Then, $\vec{\mu}^*$ is nondominated if and only if there does not exist any $\vec{\mu} \in \mathcal{M}$ that dominates $\vec{\mu}^*$. Otherwise, $\vec{\mu}^*$ is a dominated criterion vector.

Definition 5. The Pareto front (efficient frontier) $\mathcal{PF}^*$ is defined as

$$\mathcal{PF}^* \triangleq \{\vec{\mu}(\mathbf{x}) \in \mathcal{M} | \mathbf{x} \in \mathcal{P}^*\}$$

For further detail on multicriteria optimization the reader is referred to Steuer [122].

Recall that the knowledge base of the fuzzy controller is composed of a set of S rules. For $r = 1, \ldots, S$ let these rules be

$$\textbf{\textit{if}}\ p_{r1} \text{ is } \tilde{P}_{r1} \text{ and } \ldots \text{ and } p_{rM} \text{ is } \tilde{P}_{rM}\ \textbf{\textit{then}}\ q_{rj} \text{ is } \tilde{Q}_{rj} \tag{3.13}$$

Based on the fact that each of these rules is designed so that it is able to drive the simulated system towards the achievement of at least one target, our method activates (fires) to a greater extent those rules that drive the simulation toward the achievement of the currently less fulfilled goals.

Let w_r be the "weight" associated with the r-th rule (for $r = 1, \ldots, S$) and defined by

$$w_r = \max_k \{\theta_{rk}(f_{i_k}(\mathbf{x}))\} \tag{3.14}$$

where $i_k \in \{1, \ldots, M\}$, $k = 1, \ldots, K$, and

$$\theta_{rk}(f_{i_k}(\mathbf{x})) = 1 - \mu_{\tilde{P}_{i_k}}(f_{i_k}(\mathbf{x})), \tag{3.15}$$

if the k-th target is addressed through the r-th rule, 0 otherwise; and $f_{i_k}(\mathbf{x})$ is the $k-th$ performance for the simulation experiment with input parameter vector $\mathbf{x} = (x_1, \ldots, x_N)$.

The normalized weights can be defined by

$$w'_r = \frac{w_r}{\max_{r \in \{1,\ldots,S\}}\{w_r\}}, \tag{3.16}$$

if $\max_{r \in \{1,\ldots,S\}}\{w_r\} \neq 0$; $w'_r = 0$, otherwise.

We use the normalized weights to modify the aggregated fuzzy sets obtained in the fuzzification step used to adjust the simulation inputs. When using multiple criteria, Equation (3.10) is replaced by:

$$\mu_{\tilde{Q}'_j}(\Delta x_j) = \max\left(w'_1 \mu_{\tilde{Q}'_{1j}}(\Delta x_j), \ldots, w'_S \mu_{\tilde{Q}'_{Sj}}(\Delta x_j)\right), \text{ for } \Delta x_j \in \Delta X_j \tag{3.17}$$

The normalized weights can be viewed as an adaptive pressure mechanism for obtaining a significative portion of this *Pareto optimal set*. The weights are updated every time the controller is invoked, so that diversity on the Pareto front is obtained by affecting the firing strength (i.e., larger weights) of those rules that are able to drive the simulation to the achievement of the currently less fulfilled goals.

3.2.4 Algorithms

3.2.4.1 Target Threshold for the FSO Problem

To solve the fuzzy single response simulation optimization (FSO) problem (1.3) we developed a heuristic algorithm based on our fuzzy control mechanism. The algorithm tries to meet a user specified threshold on the degree of satisfaction of the single vague target represented by a fuzzy set.

Let $\mathbf{x}^t = (x_1^t, \ldots, x_N^t) \in \mathcal{X}$, $\Delta\mathbf{x}^t = (\Delta x_1^t, \ldots, \Delta x_N^t) \in \Delta X$, and $\mathbf{f}(\mathbf{x}^t) = (f_1(\mathbf{x}^t), \ldots, f_M(\mathbf{x}^t)) \in \mathcal{Y}$ be the simulation inputs, input adjustments, and outputs at iteration t, respectively. Let $\tilde{\mathbf{P}} \longrightarrow \tilde{\mathbf{Q}}$ be the knowledge base with rules in the form of (3.1). Let $i_1 \in \{1, \ldots, M\}$ be the index of the state variable used as the vague target and $\tilde{P}_{i_1}$ be its desired linguistic value. Thus in (1.3), $\tilde{G} = \tilde{P}_{i_1}$. Let $g \in [0, 1]$ be the minimum desired degree of satisfaction (threshold) for the vague target, i.e., if $\mu_{\tilde{P}_{i_1}}(f_{i_1}(\mathbf{x}^t)) \geq g$ then the target is deemed to be satisfied. Let $t_{\max}$ be a maximum allowable number of iterations. Figure 3.2 displays the algorithm that solves FSO.

Step 0: Initialization

$t \leftarrow 1$.

Generate an initial feasible solution $\mathbf{x}^1 \in \mathcal{X}$.

$\mathbf{x}^* \leftarrow \mathbf{x}^1$.

$g \leftarrow 1 - \epsilon$, where $0 \leq \epsilon \ll 1$.

Step 1: Simulation

Evaluate $\mathbf{f}(\mathbf{x}^t)$

Step 2: Termination

`if` $\mu_{\tilde{P}_{i_1}}(f_{i_1}(\mathbf{x}^t)) \geq g$ `then`

$\mathbf{x}^* \leftarrow \mathbf{x}^t$

$t \leftarrow t_{\max}$

`if` $t \geq t_{\max}$ `then`

`return` $\mathbf{x}^*$, $f_{i_1}(\mathbf{x}^*)$, and $\mu_{\tilde{P}_{i_1}}(f_{i_1}(\mathbf{x}^*))$

`stop`

Step 3: Fuzzification ($\tilde{\mathbf{P}}'$)

For each $r = 1, \ldots, S$ and $i = 1, \ldots, M$

$\mu_{\tilde{P}'_{ri}}(f_i(\mathbf{x})) \leftarrow 1$, if $f_i(\mathbf{x}) = f_i(\mathbf{x}^t)$;

$\mu_{\tilde{P}'_{ri}}(f_i(\mathbf{x})) \leftarrow 0$, otherwise (see (3.3)).

Step 4: Inference ($\tilde{\mathbf{Q}}'$)

For each $r = 1, \ldots, S$ and $j = 1, \ldots, N$

calculate $\mu_{\tilde{Q}'_{rj}}(\Delta x_j)$, for all $\Delta x_j \in \Delta X_j$ according to (3.9).

For $j = 1, \ldots, N$, aggregate the fuzzy sets $\tilde{Q}'_{rj}$ (for $r = 1, \ldots, S$) into $\tilde{Q}'_j$ using (3.10).

Step 5: Defuzzification

For each $j = 1, \ldots, N$, calculate Δx_j^t according to (3.11).

Step 6: Update

$\mathbf{x}^{t+1} \leftarrow \Pi_{\mathcal{X}}(\mathbf{x}^t + \Delta \mathbf{x}^t)$

$t \leftarrow t + 1$

Go to Step 1. ■

Figure 3.2: Target Threshold Algorithm

Note that when $\mathcal{X}$ is taken to be $l_j \leq x_j^t \leq u_j$, for all $j \in \{1, \ldots, N\}$, the projection onto the set of feasible input parameters $\mathcal{X}$, $\Pi_{\mathcal{X}}(\mathbf{x}^t + \Delta\mathbf{x}^t) = (\Pi_{X_1}(x_1^t + \Delta x_1^t), \ldots, \Pi_{X_N}(x_N^t + \Delta x_N^t))$, is defined by

$$\Pi_{X_j}(x_j^t + \Delta x_j^t) = \begin{cases} \inf(X_j), & \text{if } x_j^t + \Delta x_j^t < \inf(X_j) \\ h(x_j^t + \Delta x_j^t), & \text{if } \inf(X_j) \leq x_j^t + \Delta x_j^t \leq \sup(X_j) \\ \sup(X_j), & \text{if } x_j^t + \Delta x_j^t > \sup(X_j) \end{cases} \tag{3.18}$$

where $\inf(X_j)$ and $\sup(X_j)$ are the *infimum* and *supremum* of the set X_j, respectively; and

$$h(x_j^t + \Delta x_j^t) = \begin{cases} x_j^t + \Delta x_j^t & \text{if } j \notin I \\ \lceil x_j^t + \Delta x_j^t - \frac{1}{2} \rceil & \text{if } x_j^t + \Delta x_j^t < 0 \text{ and } j \in I \\ \lfloor x_j^t + \Delta x_j^t + \frac{1}{2} \rfloor & \text{if } x_j^t + \Delta x_j^t \geq 0 \text{ and } j \in I \end{cases} \tag{3.19}$$

where $I \subseteq \{1, \ldots, N\}$ is an index set such that $x_j^t \in \mathbb{Z}$, for all $j \in I$; $\lceil \cdot \rceil$ is the *ceiling function* that denotes the least integer greater than or equal to the argument; and $\lfloor \cdot \rfloor$ is the *floor function* that denotes the greatest integer less than or equal to the argument.

3.2.4.2 Pareto Optimization for the FMSO Problem

For problems with multiple and conflicting criteria we developed a heuristic algorithm to solve the fuzzy multicriteria simulation optimization (FMSO) problem (1.4). The proposed algorithm discovers a significant portion of the Pareto optimal set by using adaptive weights that affect the firing strength of the rules (see Section 3.2.3).

Let $\mathbf{x}^t = (x_1^t, \ldots, x_N^t) \in \mathcal{X}$, $\Delta\mathbf{x}^t = (\Delta x_1^t, \ldots, \Delta x_N^t) \in \Delta X$, and $\mathbf{f}(\mathbf{x}^t) = (f_1(\mathbf{x}^t), \ldots, f_M(\mathbf{x}^t)) \in \mathcal{Y}$ be the simulation inputs, input adjustments, and outputs at iteration t, respectively. Let $\tilde{\mathbf{P}} \longrightarrow \tilde{\mathbf{Q}}$ be the knowledge base with rules in the form of (3.1). Let $i_k \in \{1, \ldots, M\}$ (for $k = 1, \ldots, K$) be the index of the state variable used as the k-th vague target and $\tilde{P}_{i_k}$ be its desired linguistic value ($K \leq M$). Let $\mathcal{P}^*_{\text{approximate}} \subseteq \mathcal{X}$ be the approximate Pareto optimal set generated by the algorithm. To stop the algorithm, we let $g_{i_k} \in [0, 1]$ be the minimum desired degree of satisfaction (threshold) for the k-th vague target and $t_{\max}$ be the maximum number of iterations. Due to the conflicting nature of the multiple criteria, g_{i_k} can rarely be

achieved simultaneously for $k = 1, \ldots, K$. In other words, these collection of values as a whole represents a multicriteria ideal target. Figure 3.3 shows the Pareto Optimization Algorithm that solves problem FMSO.

Another important element of the proposed approach is the concept of *knowledge extraction*; that is, the capability of the system to learn, evolve and adapt based on the experience that is gained throughout the simulation runs. Initially, it is possible to start the controller with limited knowledge. As the cycle described in Figure 3.1 is repeated, more and more simulations are performed. The idea is to obtain better information on the relations between inputs and outputs as the iteration continues. This is illustrated in Figure 3.4. The process of shaping and tuning the fuzzy controller data and rule base is known as *fuzzy system identification*. Fuzzy system identification is, itself, a complex optimization problem that has drawn the attention of many researchers [73, 83, 128]. The objective of this work is to validate the fuzzy controlled simulation optimization framework. The next section describes the essentially manual approach to knowledge extraction we have used in the framework validation. An automated soft computing based fuzzy system identification mechanism is out of the scope of this dissertation.

3.3 Knowledge Acquisition

The rule structure of the approach presented in Section 3.2 allows knowledge from experts to be incorporated into the optimization strategy. However, if the simulated system is complex, important rules may be overlooked or it may be beneficial to include rules that are difficult for an expert to recognize. This section briefly describes some techniques that can be applied to generate rules in a systematic manner.

In order to build the rule base, it is necessary to explore the relations between inputs and outputs of the simulated system. *Experimental design* provides a way of planning which configurations to simulate so that the desired information can be obtained with the least simulation effort [81]. In experimental design terminology, the input parameters are called *factors* and the performance measures (outputs) are called *responses*. The "experiment" is the execution of a simulation model with the factors fixed at certain levels. A carefully planned design of experiments can provide us with valuable data from which to extract rules based on the relations between factors and responses. Depending on the complexity of the model, available time,

Step 0: Initialization

$t \leftarrow 1$

Generate an initial feasible solution $\mathbf{x}^1 \in \mathcal{X}$.

$\mathbf{x}^* \leftarrow \mathbf{x}^1$

$\mathcal{P}^*_{\text{approximate}} = \{\mathbf{x}^1\}$;

$g_{i_k} \leftarrow 1 - \epsilon_k$ (for $k = 1, \ldots, K$), where $0 \leq \epsilon_k \ll 1$; and

Step 1: Simulation

Evaluate $\mathbf{f}(\mathbf{x}^t)$

Step 2: Termination

`if` $\mu_{\tilde{P}_{i_k}}(f_{i_k}(\mathbf{x}^t)) \geq g_{i_k}$ (for all k) `or` $t \geq t_{\max}$ `then`

`return` $\mathcal{P}^*_{\text{approximate}}$

`stop`

Step 3: Dominance

3.1. $flag \leftarrow 1$

3.2. For each $\mathbf{d} \in \mathcal{P}^*_{\text{approximate}}$

`do`

`if` $\vec{\mu}(\mathbf{x}^t) \succ \vec{\mu}(\mathbf{d})$ `then` $\mathcal{P}^*_{\text{approximate}} \leftarrow \mathcal{P}^*_{\text{approximate}} \setminus \{\mathbf{d}\}$

`if` $\vec{\mu}(\mathbf{d}) \succ \vec{\mu}(\mathbf{x}^t)$ `then`

$flag \leftarrow 0$

Go to Step 3.3.

`end`

3.3. `if` $flag = 1$ `then` $\mathcal{P}^*_{\text{approximate}} \leftarrow \mathcal{P}^*_{\text{approximate}} \cup \{\mathbf{x}^t\}$

Step 4. Weights

For $r = 1, \ldots, S$ use (3.14),(3.15), and (3.16) to calculate the adaptive firing strength weights w'_r.

Step 5: Fuzzification ($\tilde{\mathbf{P}}'$)

For each $r = 1, \ldots, S$ and $i = 1, \ldots, M$

$\mu_{\tilde{P}'_{ri}}(f_i(\mathbf{x})) \leftarrow 1$, if $f_i(\mathbf{x}) = f_i(\mathbf{x}^t)$;

$\mu_{\tilde{P}'_{ri}}(f_i(\mathbf{x})) \leftarrow 0$, otherwise (see (3.3)).

Figure 3.3: Pareto Optimization Algorithm

Step 6: Inference ($\tilde{\mathbf{Q}}'$)

For each $r = 1, \ldots, S$ and $j = 1, \ldots, N$

calculate $\mu_{\tilde{Q}'_{rj}}(\Delta x_j)$, for all $\Delta x_j \in X_j$ according to (3.9).

For $j = 1, \ldots, N$, aggregate the fuzzy sets $\tilde{Q}'_{rj}$ (for $r = 1, \ldots, S$) into $\tilde{Q}'_j$ using (3.17).

Step 7: Defuzzification

For each $j = 1, \ldots, N$, calculate Δx_j^t according to (3.11).

Step 8: Update

$\mathbf{x}^{t+1} \leftarrow \Pi_{\mathcal{X}}(\mathbf{x}^t + \Delta \mathbf{x}^t)$, where $\Pi_{\mathcal{X}}(\cdot)$ is a projection onto $\mathcal{X}$ according to (3.18) and (3.19).

$t \leftarrow t + 1$

Go to Step 1. ■

Figure 3.3: Pareto Optimization Algorithm (continued).

and computational resources, acceptable choices for our framework may be 2^N or 3^N factorial designs (for $N < 10$). For further information on experimental design the reader is referred to [16, 81, 93].

By executing the simulation model according to the experimental design, a set with L observations of inputs (factors) and outputs (responses) is obtained. By examining the associated matrix of correlation coefficients between inputs and outputs, single antecedent rules of the type "if p is $\tilde{P}$ then q is $\tilde{Q}$" can be generated. The sample correlation coefficient, for the input $j \in \{1, \ldots, N\}$ and output $i \in \{1, \ldots, M\}$, is defined as:

$$\rho_{ji} = \frac{\sum_{l=1}^{L}(x_{jl} - \bar{x}_j)(f_{il}(\mathbf{x}_l) - \bar{f}_i)}{\sqrt{\sum_{l=1}^{L}(x_{jl} - \bar{x}_j)^2}\sqrt{\sum_{l=1}^{L}(f_{il}(\mathbf{x}) - \bar{f}_i)^2}} \tag{3.20}$$

where L is the number of observations (experiments); x_{jl} is the value of the j-th input (factor) for the l-th observation; $f_{il}(\mathbf{x}_l)$ is the i-th output (response) obtained with input parameters $\mathbf{x}_l$; $\bar{x}_j = \sum_l^L \frac{x_{jl}}{L}$ is the average of the j-th input across all observations; and $\bar{f}_i = \sum_l^L \frac{f_{il}(\mathbf{x}_l)}{L}$ is the average of the i-th output across all observations.

The correlation coefficient ρ_{ji} lies between -1 and +1. It measures the strength of association between the input j and output i. Roughly speaking, a value of $\rho_{ji} > 0$ implies that increasing the value of input j has the tendency to increase the value

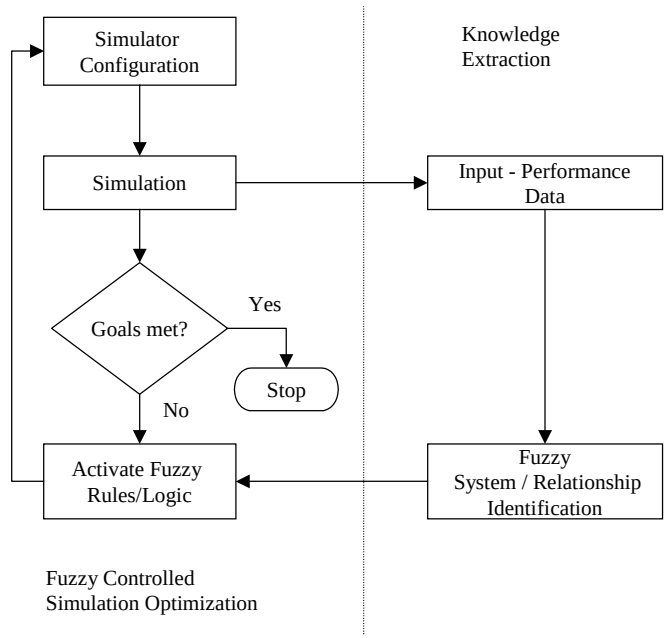


Figure 3.4: Fuzzy controlled simulation optimization and knowledge extraction

of the output i, while decreasing the value of input j has the tendency to decrease the value of the output i. On the other hand, a value of $\rho_{ji} < 0$ implies that an increase in the input j has a decremental effect on output i, while a decrease has an incrementing effect. Based on this relationship and its strength (closeness to -1 and +1), simple rules "if p is $\tilde{P}$ then q is $\tilde{Q}$" can be generated.

To enrich the expressive power of the rule base, rules of the type "if p_{i_1} is $\tilde{P}_{i_1}$ and p_{i_2} is $\tilde{P}_{i_2}$ then q is $\tilde{Q}$" $(i_1, i_2 \in \{1, \ldots, M\})$ may be considered. The correlation coefficient does not provide any means measuring the effect of interactions among multiple inputs with reference to a single output.

To get a feeling for these relationships we can use the method of least squares to fit quadratic response surface regression models to the observations obtained through the design of experiments [16, 93]. These response surface models or *metamodels* can also be used to test for the significance of individual factors and interactions and to predict new values of the response. The quadratic model can be expressed as

$$f_i(\mathbf{x}) = \mathbf{x}'\mathbf{A}\mathbf{x} + \mathbf{b}\mathbf{x} + e \tag{3.21}$$

where $\mathbf{x} = (x_1, \ldots, x_N)'$ are the simulation inputs, $f_i(\mathbf{x})$ is the i-th performance

measure (for $i = \{1, \ldots, M\}$), $\mathbf{A}$ is a $N \times N$ symmetric parameter matrix, $\mathbf{b}$ is a linear parameter vector, and e is the error term.

By fitting these models to our responses, we can extrapolate the scattered measurements obtained with the design of experiments and get a sense of the effect on the outputs, obtained by changing multiple inputs simultaneously. In classical response surface methodology [16, 93], the interest is on finding the best combination of inputs to accurately predict the response. In our case, we just want to discover basic relationships between inputs and outputs that can be easily translated into fuzzy rules.

3.4 Application to Flow Line Design

3.4.1 Introduction

We illustrate the proposed approach using a common problem that arises in manufacturing, namely the Flow Line Design Problem (FLDP). The flow line is a widely used way to organize production, especially for products made in sufficient volume to justify the investment in dedicated machines, operators, and material handling systems.

The flow line that is modeled in this section is a single product line with human unpaced workers [18]. The interarrival times are stochastic (not necessarily exponential). The line has V stations (or stages), with a buffer in front of every station. There is no transit time between one station and the next one. Every station has one or more identical servers. The service time is stochastic (not necessarily exponential). This flow line, which falls into the category of asynchronous lines, does not have any coordination of job movement between stations. An available operator starts a job as soon as it is available and, upon completion, the job leaves the station provided there is room for it in the next station. This mode of operation, may cause *starvation* and *blocking* of servers. It is appropriate to model this flow line as a tandem queueing network with blocking such as the one depicted in Figure 3.5, where λ is the arrival rate to the flow line, and γ_v, s_v, and b_v, are the service rate, number of servers, and buffer capacity for station $v \in \{1, \ldots, V\}$, respectively.

For very small queueing networks with blocking, it is possible to solve numerically for the stationary distribution of the underlying stochastic process. Unfortunately,

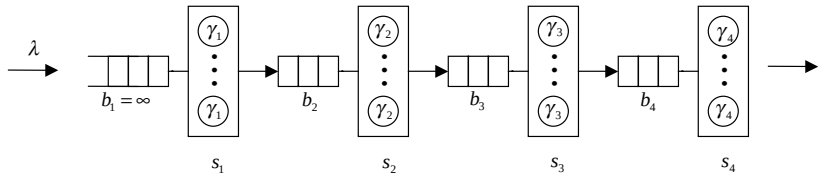


Figure 3.5: Tandem of queues with blocking (flow line)

this approach can be computationally intensive and impractical for most realistic scenarios. One approach to analyze these configurations is through the use of an approximation algorithm based on the idea of decomposition. The idea is to break the tandem configuration into subsystems and to analyze each subsystem individually. To perform this analysis in isolation, it is necessary to have input from the other subsystems. Therefore, the approximate algorithm is designed as an iterative scheme that tries to satisfy some convergence criterion in the system. An overview of these algorithms can be found in [101].

Discrete event simulation provides a more flexible approach for analyzing the system shown in Figure 3.5. This is the approach that we have used.

There is no unique Flow Line Design Problem (FLDP) definition. It can be formulated as a problem of minimizing the labor cost of meeting a given throughput, subject to constraints on meeting quality targets and keeping work–in–process or space less than given maximum levels. An alternative formulation can be stated as that of minimizing the sum of labor costs, work–in–process and space costs subject to quality and throughput constraints [18]. Our approach is flexible enough to let the user specify the goal to be targeted. Moreover, we allow the user to find a design evaluated by multiple criteria.

3.4.2 Framework

Figure 3.6 shows the actual implementation of the simulation optimization framework, presented in Figure 3.4, used to tackle the flow line design problem. Again, the basic components are a discrete event simulator of the flow line and a fuzzy controller that are tightly coupled. These components are described in the following sections. Figure 3.6 also indicates that the fuzzy controller was built off–line. In other words, first the controller was built and then it was used to control the simulation. Once the

controller was built, new simulation runs did not affect the controller design.

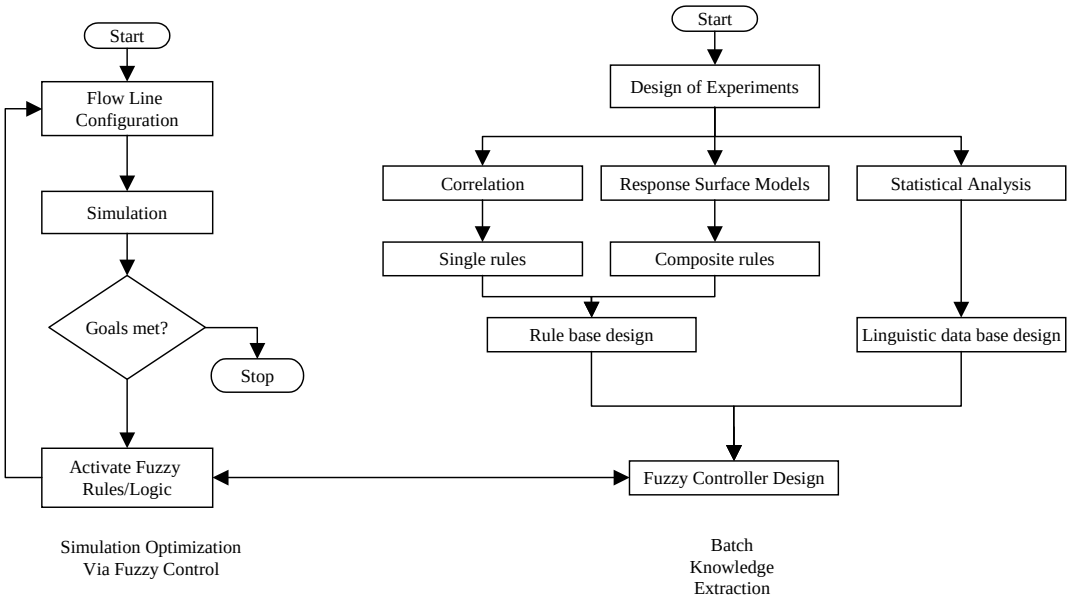


Figure 3.6: Flow line optimization framework

3.4.3 Simulator

Arrivals of jobs to the system were generated according to a homogeneous Poisson process with mean arrival rate λ. Each station $v \in \{1, \ldots, V\}$ had s_v identical servers with a controllable service rate γ_v. The service times were exponentially distributed. Each station had a limited buffer size b_v.

Table 3.1 lists all the input parameters and their possible values for the flow line model shown in Figure 3.5. Note that those inputs for which the lower and upper bound are identical, were considered fixed. In total, there were seven controllable inputs (i.e., $N = 7$).

The performance measures that were collected by the simulator and define the state variables of the fuzzy controller are the overall time in system (τ), overall work–in–process or WIP (ϖ), work–in–process at station v (ϖ_v), and server utilization at station v (φ_v), for $v = 1, \ldots, 4$. In total, there were ten system performance measures (i.e., $M = 10$).

Throughout this work we used *transient* simulation analysis based on the *method of independent replications* [80]. Note that in this particular case, it is not hard to

Parameter	Station 1		Station 2		Station 3		Station 4	
	Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.
Arrival rate (λ)	2	2	-	-	-	-	-	-
Server rate (γ)	2.1	4.9	0.5	0.5	0.3	2.1	0.7	0.7
Servers (s)	1	1	5	9	1	7	3	9
Buffer size (b)	∞	∞	1	7	3	3	1	9

Table 3.1: Flow line simulation inputs ranges

find the set of conditions on the service rates that makes the flow line simulation reach steady state (i.e., $\frac{\lambda}{s_v \cdot \gamma_v} < 1$ for every station v). However, in the more general case of supply chain or telecommunication systems these conditions are frequently unknown.

To explore the relations between inputs and outputs of the flow line model, a set of simulation experiments was executed. For each of the seven inputs or factors, three different levels were specified. The levels were set to the minimum, maximum and the mid point of the ranges in Table 3.1. The total number of combinations in this 3^N design is $2,187$ (i.e., 3^7), but only $1,458$ simulations were actually needed due to the easily verifiable steady state condition cited above. For each experiment, 20 replications of 1,440 time units were conducted. The simulator was implemented using AweSim [103].

3.4.4 Knowledge Base Design

3.4.4.1 State and Control Variables

Recall that, the state and control variables of the fuzzy controller are coupled to the outputs and inputs of the simulator, respectively.

The state variables for the fuzzy controller and their possible values were specified as follows: time in system (τ) can be short ($\tilde{S}_\tau$), medium ($\tilde{M}_\tau$), or long ($\tilde{L}_\tau$); overall work–in–process (ϖ) can be low ($\tilde{L}_\varpi$), medium ($\tilde{M}_\varpi$), or high ($\tilde{H}_\varpi$); work–in–process at station v (ϖ_v) can be low ($\tilde{L}_{\varpi_v}$), medium ($\tilde{M}_{\varpi_v}$), or high ($\tilde{H}_{\varpi_v}$); and utilization at station v (φ_v) can be low ($\tilde{L}_{\varphi_v}$), medium ($\tilde{M}_{\varphi_v}$), or high ($\tilde{H}_{\varphi_v}$).

Figure 3.7 and Figure 3.8, depict the fuzzy sets representing the possible values of work–in–process and utilization at station 1, respectively. The rest of the fuzzy sets are given in Appendix A.

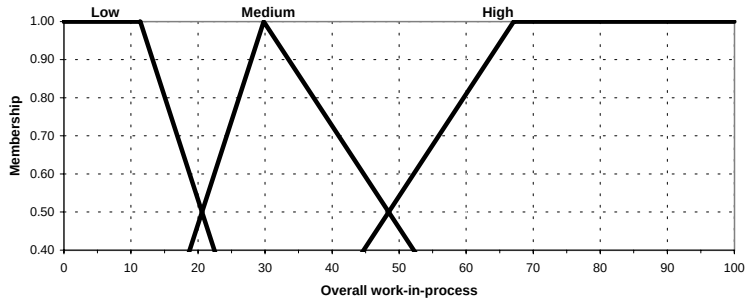


Figure 3.7: Overall work–in–process (ϖ)

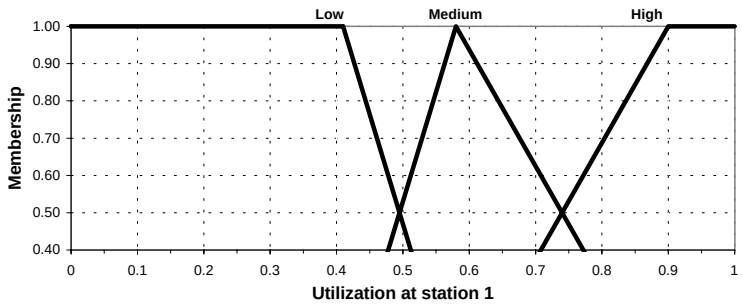


Figure 3.8: Utilization at station 1 (φ_1)

The control variables for the fuzzy controller and their possible values were specified as follows: change in server rate at station v ($\Delta\gamma_v$) can be negatively large ($\tilde{NL}_{\Delta\gamma_v}$), negatively small ($\tilde{NS}_{\Delta\gamma_v}$), zero ($\tilde{Z}_{\Delta\gamma_v}$), positively small ($\tilde{PS}_{\Delta\gamma_v}$), or positively large ($\tilde{PL}_{\Delta\gamma_v}$); change in number of servers at station v (Δs_v) can be negatively large ($\tilde{NL}_{\Delta s_v}$), negatively small ($\tilde{NS}_{\Delta s_v}$), zero ($\tilde{Z}_{\Delta s_v}$), positively small ($\tilde{PS}_{\Delta s_v}$), or positively large ($\tilde{PL}_{\Delta s_v}$); and change in buffer size at station v (Δb_v) can be negatively large ($\tilde{NL}_{\Delta b_v}$), negatively small ($\tilde{NS}_{\Delta b_v}$), zero ($\tilde{Z}_{\Delta b_v}$), positively small ($\tilde{PS}_{\Delta b_v}$), or positively large ($\tilde{PL}_{\Delta b_v}$).

Figure 3.9 depicts the fuzzy sets representing the possible values of the change in server rate at station 1. The rest of fuzzy sets are presented in Appendix A.

3.4.4.2 Linguistic Data Base Design

When a fuzzy controller is designed, the state and control variables and their values are known qualitatively. The membership functions that encapsulate this knowledge, have to be parameterized based on observed data or the opinion of experts.

For the flow line state variables, an off–line method based on univariate statistical

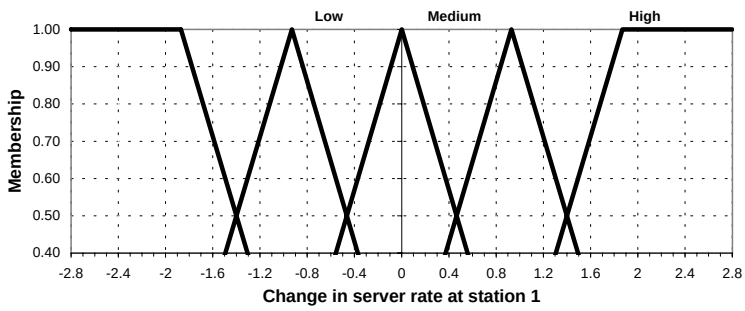


Figure 3.9: Change in server rate at station 1 ($\Delta\gamma_1$)

analysis was performed to tune these functions. Using data collected in the design of experiments (see Section 3.4.3), descriptive statistics were calculated to obtain an idea of the spread of the performance measures from the simulation (i.e., fuzzy controller state variables). For instance, for the work–in–process (ϖ), using *The SAS System* [67] we calculated the 25th (11.4), 50th (29.8), and 75th (67.1) percentiles. These values are reflected in the shapes of the values $\tilde{L}_\varpi$, $\tilde{M}_\varpi$, and $\tilde{H}_\varpi$, shown in Figure 3.7. The rest of state variables were tuned in a similar fashion.

The membership functions of the control variables were determined using the information in Table 3.1. The range for each output was divided evenly into 5 fuzzy sets: negatively large ($\tilde{NL}.$), negatively small ($\tilde{NS}.$), zero ($\tilde{Z}.$), positively small ($\tilde{PS}.$), and positively large ($\tilde{PL}.$). For instance, let us consider the membership function for the change in server rate at station 1 shown in Figure 3.9. From Table 3.1 we see that the associated simulation input varies from 2.1 to 4.9. If the parameter is at its lowest value, the maximum change that can be made in the server rate is +2.8. Similarly, if the server rate is in its maximum (4.9), a maximum decrease of -2.8 is possible. Therefore, the range of the control variable is -2.8 to 2.8. The rest of control variables were tuned in a similar way.

3.4.4.3 Rule Base Design

3.4.4.3.1 Single Antecedent Rules After executing the simulation experiments described in Section 3.4.3, the correlation matrix shown in Figure 3.10 was calculated.

The correlations in Figure 3.10 were used to estimate the magnitude and direction of association between the simulation inputs and outputs. Based on this information single antecedent rules were built. For instance, the strong and negative association

System performance measures

Inputs	φ_1	φ_2	φ_3	φ_4	ϖ	ϖ_1	ϖ_2	ϖ_3	ϖ_4	τ
γ_1	-0.9713	0.3152	0.0139	0.3085	-0.3406	-0.3653	0.221	0.0702	0.0534	-0.3403
γ_3	0.0018	0.0122	-0.8609	0.0294	-0.0488	-0.0136	-0.1106	-0.6366	0.0122	-0.0493
s_2	0.0295	0.2187	0.0093	0.2079	-0.2056	-0.2184	0.0517	0.0694	0.038	-0.2055
s_3	0.0314	0.2748	0.4773	0.2686	-0.2627	-0.2755	-0.2595	0.2882	0.0718	-0.2625
s_4	0.0332	0.3076	0.0152	0.3265	-0.3474	-0.3013	-0.3488	-0.3884	-0.6377	-0.3472
b_2	0.0188	0.1126	0.008	0.1101	-0.1036	-0.1245	0.3654	0.0256	0.0175	-0.1036
b_4	0.0142	0.1307	0.009	0.1247	-0.1158	-0.1253	-0.0559	-0.0602	0.3583	-0.1156

Figure 3.10: Correlation coefficients extracted from the full correlation matrix

between the server rate and the utilization at station 1 (i.e., -0.9713), suggests that if we want to increase the level of utilization at that station we should consider decreasing the server rate. On the other hand, if we increase the server rate, the utilization will decrease. The following three rules were constructed based on this particular correlation:

- If utilization at station 1 (φ_1) is high ($\tilde{H}_{\varphi_1}$) then the change in the server rate at station 1 ($\Delta\gamma_1$) should be zero ($\tilde{Z}_{\Delta\gamma_1}$).
- If utilization at station 1 (φ_1) is medium ($\tilde{M}_{\varphi_1}$) then the change in the server rate at station 1 ($\Delta\gamma_1$) should be negatively small ($\tilde{NS}_{\Delta\gamma_1}$).
- If utilization at station 1 (φ_1) is low ($\tilde{L}_{\varphi_1}$) then the change in the server rate at station 1 ($\Delta\gamma_1$) should be negatively large ($\tilde{NL}_{\Delta\gamma_1}$).

Other single antecedent rules, which will be presented later, were generated in a similar fashion.

3.4.4.3.2 Multiple Antecedent Rules To get a more detailed idea of the relations between inputs and outputs, response surfaces were fitted to the data obtained from the simulation experiments.

For instance, the following quadratic model with interactions was obtained for the overall work–in–process ($R^2 = 0.8143$) using *The SAS System* [68]:

$$\varpi = 1118.934697 - 135.130079\gamma_1 - 199.081179\gamma_3 - 68.950721s_2 - 74.839592s_3 - 53.882586s_4 - 22.531804b_2 - 15.677057b_4 + 11.144542\gamma_1^2 + 1.870049\gamma_3 \cdot \gamma_1 + 51.294286\gamma_3^2 + 3.841387s_2 \cdot \gamma_1 + 1.331735s_2 \cdot \gamma_3 + 2.737977s_2^2 + 0.909195 + s_3 \cdot \gamma_1 + 0.449057s_3 \cdot \gamma_3 +$$

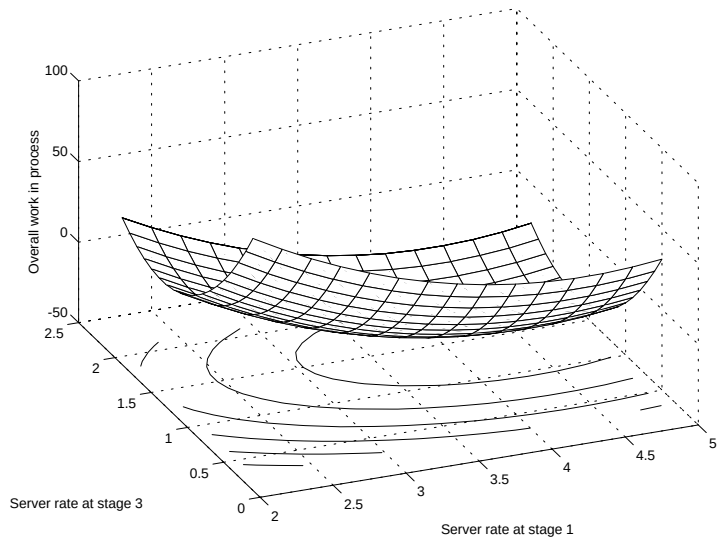


Figure 3.11: Response surface for the work–in–process ($s_3 = 4$, $s_4 = 6$, $b_2 = 4$, $b_4 = 5$, and $s_2 = 7$)

$$1.085798 s_3 \cdot s_2 + 4.23128 s_3^2 - 0.43481 s_4 \cdot \gamma_1 + 2.986399 s_4 \cdot \gamma_3 - 0.257443 s_4 \cdot s_2 + 1.923021 s_4 \cdot s_3 + 2.588835 s_4^2 + 2.364254 b_2 \cdot \gamma_1 + 0.243739 b_2 \cdot \gamma_3 + 1.136898 b_2 \cdot s_2 + 0.164275 b_2 \cdot s_3 - 0.091285 b_2 \cdot s_4 + 0.41866 b_2^2 - 0.063077 b_4 \cdot \gamma_1 + 1.176102 b_4 \cdot \gamma_3 - 0.021688 b_4 \cdot s_2 + 0.829086 b_4 \cdot s_3 + 0.968249 b_4 \cdot s_4 - 0.002358 b_4 \cdot b_2 + 0.249043 b_4^2$$

An associated surface for the work–in–process (ϖ) versus server rate at station 1 (γ_1) and server rate at station 3 (γ_3), with $s_3 = 4$, $s_4 = 6$, $b_2 = 4$, $b_4 = 5$, and $s_2 = 7$ is shown in Figure 3.11.

Similar models were obtained for other responses. Figure 3.12 shows a response surface for the utilization at the first station (φ_1). The R^2 for this full quadratic model is 0.9960.

Composite rules can be derived from inspection of these surfaces. For instance, looking at Figures 3.11 and 3.12 the following rule was created:

- If the overall work–in–process (ϖ) is high ($\tilde{H}_{\varpi}$) and the utilization at station 1 (φ_1) is high ($\tilde{H}_{\varphi_1}$) then the change in the server rate at station 1 ($\Delta\gamma_1$) should be positively large ($\tilde{PL}_{\Delta\gamma_1}$).

Other composite rules, which will be presented later, were generated in a similar

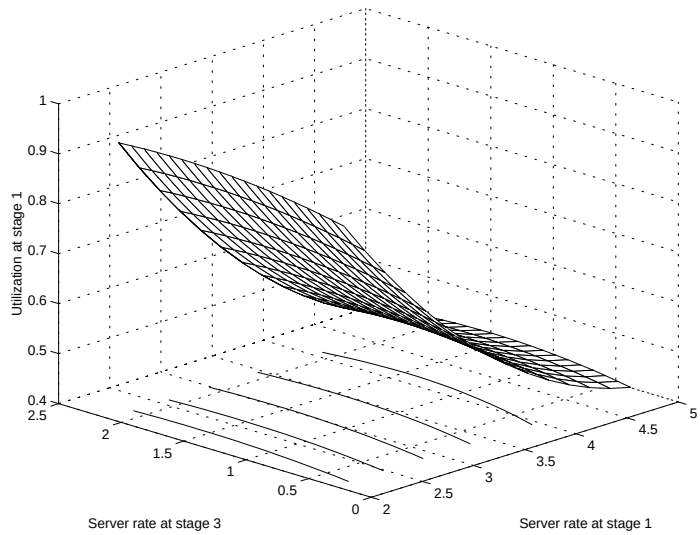


Figure 3.12: Response surface for the utilization at station 1 ($s_3 = 4$, $s_4 = 6$, $b_2 = 4$, $b_4 = 5$, and $s_2 = 7$)

fashion.

3.4.5 Computational Experiments

We present computational experience with single–goal and two–goal flow line design scenarios to illustrate the application of the algorithms for the FSO and FMSO presented in Section 3.2.4.

3.4.5.1 Single Objective

We first consider the case of a single–goal scenario in which a flow line is to be designed trying to achieve a low overall work–in–process.

The target threshold algorithm presented in Section 3.2.4.1 was used to design the flow line. The initial feasible solution was set to $\mathbf{x}^0 = (\gamma_1^0, \gamma_3^0, s_2^0, s_3^0, s_4^0, b_2^0, b_4^0)' = (2.5, 0.7, 5, 4, 4, 2, 2)'$. The state variable overall work–in–process ϖ was used as the target variable and $\tilde{L}_\varpi$ as its desired linguistic value shown in Figure 3.7. The algorithm's threshold on the degree of satisfaction of the *low overall work–in–process* target was set to $g = 0.999$ and the maximum number of iterations was set to $t_{\max} = 10$.

The rule base was built using the techniques described in Section 3.4.4.3. The rule base composed by 6 single and 2 multiple antecedent rules follows:

Rule 1. If the overall work–in–process (ϖ) is high ($\tilde{H}_{\varpi}$) then the change in server rate at station 1 ($\Delta\gamma_1$) should be positively small ($\tilde{PS}_{\Delta\gamma_1}$).

Rule 2. If the overall work–in–process (ϖ) is medium ($\tilde{M}_{\varpi}$) then the change in server rate at station 1 ($\Delta\gamma_1$) should be positively small ($\tilde{PS}_{\Delta\gamma_1}$).

Rule 3. If the overall work–in–process (ϖ) is low ($\tilde{L}_{\varpi}$) then the change in server rate at station 1 ($\Delta\gamma_1$) should be zero ($\tilde{Z}_{\Delta\gamma_1}$).

Rule 4. If the overall work–in–process (ϖ) is high ($\tilde{H}_{\varpi}$) then the change in the number of servers at station 4 (Δs_4) should be positively small ($\tilde{PS}_{\Delta s_4}$).

Rule 5. If the overall work–in–process (ϖ) is medium ($\tilde{M}_{\varpi}$) then the change in the number of servers at station 4 (Δs_4) should be positively small ($\tilde{PS}_{\Delta s_4}$).

Rule 6. If the overall work–in–process (ϖ) is low ($\tilde{L}_{\varpi}$) then the change in the number of servers at station 4 (Δs_4) should be zero ($\tilde{Z}_{\Delta s_4}$).

Rule 7. If the overall work–in–process (ϖ) is high ($\tilde{H}_{\varpi}$) and the utilization at station 3 (φ_3) is high ($\tilde{H}_{\varphi_3}$) then the change in the server rate at station 3 ($\Delta\gamma_3$) should be positively large ($\tilde{PL}_{\Delta\gamma_3}$).

Rule 8. If the overall work–in–process (ϖ) is medium ($\tilde{M}_{\varpi}$) and the utilization at station 3 (φ_3) is high ($\tilde{H}_{\varphi_3}$) then the change in the server rate at station 3 ($\Delta\gamma_3$) should be positively small ($\tilde{PS}_{\Delta\gamma_3}$).

The algorithm stopped with the solution $\mathbf{x}^* = (\gamma_1^*, \gamma_3^*, s_2^*, s_3^*, s_4^*, b_2^*, b_4^*)' = (3.6162, 1.4027, 5, 4, 6, 2, 2)'$. The degree of satisfaction of the goal "low overall work–in–process" ($\tilde{L}_{\varpi}$) as the algorithm progresses can be seen in Figure 3.13. This graph shows that the objective becomes highly satisfied (i.e., 0.9920 on a scale from 0 to 1), with dramatic improvement made in very few iterations using very few rules.

To illustrate the system's controllability, ten runs of the algorithm presented in Figure 3.2 were executed choosing random initial conditions. The initial conditions are shown in Table 3.2. To improve the effectiveness of the algorithm, seven additional multiple antecedent rules were added to the rule base. We set $t_{\max} \leftarrow 10$ and $g \leftarrow 1$.

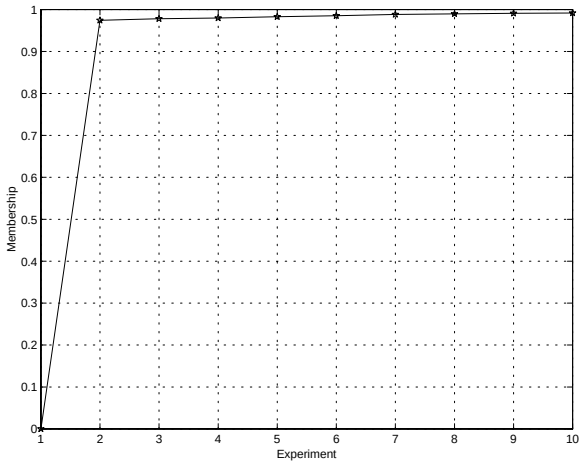


Figure 3.13: Control path for the overall work–in–process membership function for the single–goal scenario.

Run	γ_1	γ_3	s_2	s_3	s_4	b_2	b_4
1	2.1	2.1	5	1	3	1	1
2	4.4	1.4	7	6	7	5	3
3	3.2	1.1	6	6	6	1	9
4	2.9	1.9	8	3	5	3	4
5	4.6	0.9	6	6	7	6	4
6	3.3	1.9	8	2	8	5	9
7	4.2	1.8	5	4	4	2	4
8	3.4	0.9	9	3	4	3	7
9	3.4	1.4	5	6	3	1	6
10	4	0.9	8	5	8	6	2

Table 3.2: Initial conditions for the single–goal scenario with improved controllability

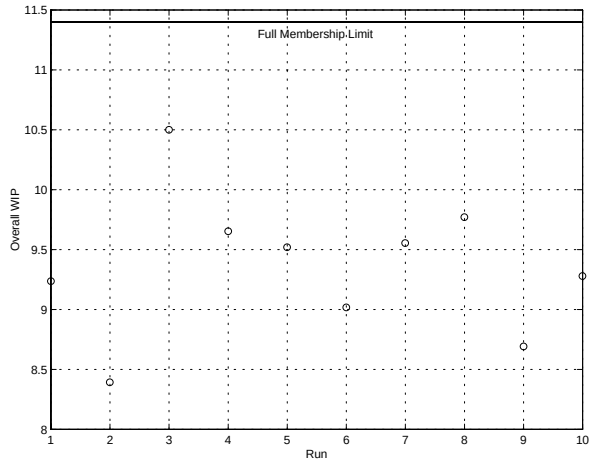


Figure 3.14: Solution for different initial conditions for the single–goal scenario with improved controllability

Regardless of the initial conditions (run number), the controller was able to find a configuration of the simulated flow line such that the goal of low work–in–process level is satisfied to the highest possible degree. Recall from Figure 3.7 that $\mu_{\tilde{L}_\varpi}(x) = 1$ for $x \leq 11.4$. The final solution $(f_\varpi(\mathbf{x}^*))$ for each of the ten runs is illustrated in Figure 3.14.

3.4.5.2 Multiple Objectives: Generating the Efficient Frontier

In this section we present a scenario in which a flow line is designed based on two conflicting goals which we wish to satisfy simultaneously. The purpose of this example is to illustrate how we can obtain an approximate Pareto front by guiding the simulation of the flow line with a fuzzy controller using the algorithm presented in Figure 3.3. Specifically, we wish to design the flow line to achieve "low work–in–process" and "high utilization at station 1", simultaneously.

For the fuzzy controlled approach we developed a rule base composed of 18 rules (Appendix B). The state variables used as vague targets were overall work–in–process (ϖ) and utilization at station 1 (φ_1) with values $\tilde{L}_\varpi$ (Figure 3.7) and $\tilde{H}_{\varphi_1}$ (Figure 3.8), respectively. We conducted ten runs of the algorithm shown in Figure 3.3, with randomly selected initial conditions. For each run, we set $g_\varpi = 1$ and $g_{\varphi_1} = 1$ and $t_{\max} = 10$.

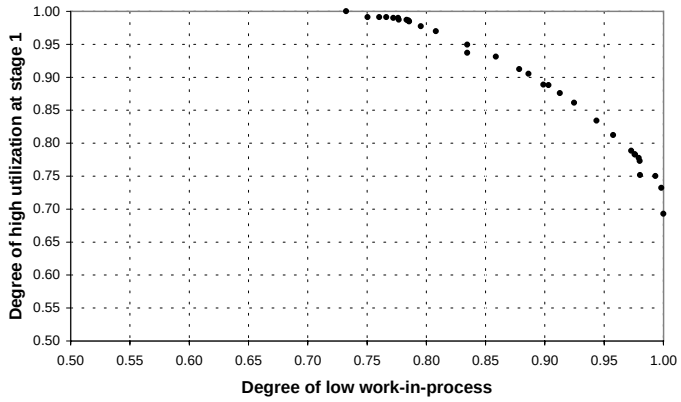


Figure 3.15: Approximate Pareto front for the two-goal scenario

Figure 3.15 shows the approximate Pareto front obtained by the proposed approach. With only 100 simulation runs the Pareto front is evenly generated and a good sense of its shape is obtained.

Even though it is customary to assess the quality of the Pareto front by visual inspection, Zitzler and Thiele [150] have recently proposed a metric based on the dominated space defined by the nondominated vectors of the Pareto front. Because each axis of the Pareto front is associated to the degree of satisfaction of a criterion, the dominated space is bounded by 0 and 1, being 1 the best value possible. This metric is further discussed in Section 4.2. In Figure 3.16 our visual assessment is validated by Zitzler and Thiele's metric, which after 100 simulation runs is 0.973. Perhaps, the most revealing result is the fast convergence to a high quality solution. Note that only 25 simulation runs are needed to generate a Pareto front with Zitzler and Thiele's metric of 0.962.

3.5 Concluding Remarks

We have proposed a new mechanism for the optimization of complex systems modeled by discrete event simulation. Contrary to classical methods, our approach works with imprecise concepts and natural language to aid the decision maker in the system design process. A distinctive feature of our simulation optimization strategy is the use of approximate reasoning through a fuzzy controller to drive the optimization process using a small set of rules that encapsulates the relevant knowledge of the system.

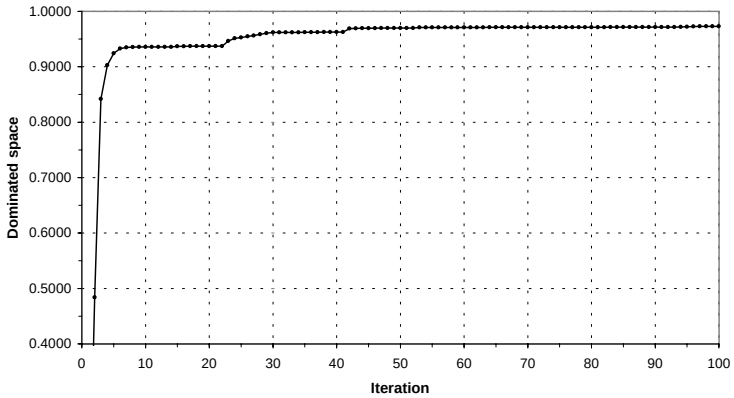


Figure 3.16: Zitzler and Thiele's dominated space metric [150] for the two-goal scenario

Using these rules, which are easily generated from statistical correlation measures and quadratic response surface models, the controller drives the system towards a high degree of satisfaction of one or more vaguely stated targets. In the presence of multiple and conflicting objectives, the proposed approach is able to construct an approximate Pareto optimal set. The computational results confirm that the proposed approach delivers a high quality solution (in terms of the size of the dominated space [150]) in a fast and efficient manner.

Chapter 4

A Multicriteria Evolutionary Approach

This chapter provides an alternative evolutionary approach to solving the fuzzy multicriteria simulation optimization (FMSO) problem. To the best of our knowledge, there has been no study in the field of simulation optimization that combines the search power of evolutionary algorithms and the natural multicriteria characteristics of simulation. The absence of such study, combined with the synergistic view of these techniques, has motivated us to develop the evolutionary algorithm presented in this chapter. In addition, using this alternative approach we are able to assess the quality of the solutions obtained by the method presented in the previous chapter. In the first part of this chapter, we describe the evolutionary approach, while the second part deals with the comparison of the performance of these two approaches in solving the Flow Line Design Problem.

4.1 Evolutionary Algorithm

We propose a new approach to multicriteria optimization, which we call the Optimal Scoring Evolutionary Algorithm (OSEA). This algorithm is designed to discover Pareto optimal solutions. Although it is possible to apply OSEA to solve general multicriteria optimization problems, it has been designed specifically to solve the fuzzy multicriteria simulation optimization problem (FMSO). The algorithm presented in Figure 4.1 gives the pseudocode for OSEA.

$t \leftarrow 0$

Randomly generate the members of $P(0)$

Evaluate members of $P(0)$ via simulation

Fuzzify observed performance levels for members of $P(0)$

Update $\mathcal{P}^*_{\text{approximate}}(0)$ from $P(0)$

$E(0) \leftarrow P(0)$

Assign fitness to members of $E(0)$

$t \leftarrow 1$

`While` $t \leq t_{max}$ `then do`

  Select $P(t)$ from $E(t-1)$

  Generate $C_m(t)$ from $P(t)$ (mutation)

  Generate $C_c(t)$ from $P(t)$ (crossover)

  $C(t) = C_m(t) \cup C_c(t)$

  Evaluate members of $C(t)$ via simulation

  Fuzzify observed performance levels for members of $C(t)$

  Update $\mathcal{P}^*_{\text{approximate}}(t)$ from $P(t)$

  Generate $C_e(t)$ from $\mathcal{P}^*_{\text{approximate}}(t)$ (elitism)

  $E(t) = P(t) \cup C(t) \cup C_e(t)$

  Assign fitness to members of $E(t)$

  $t \leftarrow t + 1$ ■

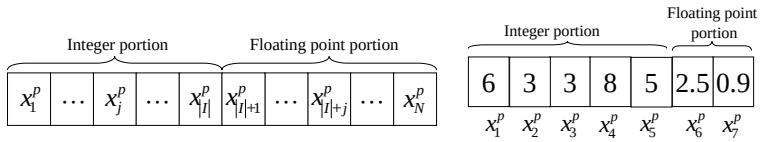
Figure 4.1: Optimal Scoring Evolutionary Algorithm (OSEA)

where t is the generation number and t_{max} the maximum allowed number of generations. For the t-th generation, $P(t)$ is the set (of size P_{max}) of parents and $C(t)$ is the set of children, with $C_m(t)$ being the children generated through mutation and $C_c(t)$ being the children generated through crossover. $\mathcal{P}^*_{\text{approximate}}(t)$ is the current approximate Pareto optimal set generated by the algorithm; $C_e(t)$ is a subset of $\mathcal{P}^*_{\text{approximate}}(t)$ selected for reinsertion into the population; and the set $E(t)$ is the resulting expanded population at iteration t.

In the remaining parts of this chapter we will describe the basic elements of the algorithm shown in Figure 4.1.

4.1.1 Representation

Let $\mathbf{x}^p = (x_1^p, \ldots, x_N^p) \in \mathcal{X}$ be a vector of N controllable parameters of the simulation model, where $\mathcal{X}$ is a closed set of constraints on $\mathbf{x}^p$, from the FMSO problem. In OSEA we use a *mixed* representation, in which genes are encoded by integers and real numbers. The j-th gene of the p-th chromosome (i.e., individual) is represented by $x_j^p \in X_j$, where X_j its the allowable range. Let $I \subseteq \{1, \ldots, N\}$ be an index set for the integer genes. For each gene x_j^p with $j \in I$, we use an integer encoding; while for x_j^p with $j \notin I$, we use a floating point encoding [91]. Figure 4.2 (a) illustrates the mixed representation for the p-th individual. As an example, let us consider an individual in the flow line design problem (represented by a tandem of queues), discussed later in Section 4.2. This p-th individual is represented by a six–dimensional vector, where $x_1^p \in [5, 9]$ is the number of servers allocated to the second station; $x_2^p \in [1, 7]$ is the size of the buffer space allocated to the second station; $x_3^p \in [1, 7]$ is the number of servers allocated to the third station; $x_4^p \in [3, 9]$ is the number of servers allocated to the fourth station; $x_5^p \in [1, 9]$ is the size of the buffer space allocated to the fourth station; $x_6^p \in [2.1, 4.9]$ is the server rate in the first station; and $x_7^p \in [0.3, 2.1]$ is the server rate in the third station. Figure 4.2 (b) shows the p-th individual $\mathbf{x}^p = (6, 3, 3, 8, 5, 2.5, 0.9)$ as represented in OSEA. Furthermore, $I = \{1, 2, 3, 4, 5\}$, $X_1 = [5, 9]$, $X_2 = [1, 7]$, $X_3 = [1, 7]$, $X_4 = [3, 9]$, $X_5 = [1, 9]$, $X_6 = [2.1, 4.9]$, and $X_7 = [0.3, 2.1]$.



(a) Mixed representation.

(b) Example of an individual in the Flow Line Design Problem.

Figure 4.2: Representation in OSEA.

4.1.2 Evaluation

Let $\mathbf{f}(\mathbf{x}^p) = (f_1(\mathbf{x}^p), \ldots, f_M(\mathbf{x}^p))$ be the M-dimensional vector of performance measures obtained by executing the simulation model with controllable parameters set to $\mathbf{x}^p$. These measures should be easily collectable and retrievable after the simulator completes a batch of replications. The mapping $\mathbf{x}^p \rightarrow \mathbf{f}(\mathbf{x}^p)$ represents the evaluation of the p-th individual of the population via the simulator.

Simulation optimization is computationally expensive due to the multiple number of simulation scenarios evaluated. To streamline the evaluation process, we implement a *simulator accelerator*. Before calling the simulator, a database with the history of previously executed simulation runs is consulted. If the model defined by $\mathbf{x}^p$ has already been executed, the average performance measures are retrieved directly from the database without invoking the simulator. If the model has not been run, the simulator is invoked setting the controllable parameters to $\mathbf{x}^p$.

4.1.3 Fuzzification

The simulator collects information on M performance measures, namely $f_1(\mathbf{x}^p), \ldots, f_M(\mathbf{x}^p)$. The level of performance of the system is measured against a subset of vague targets for K, $K \leq M$, of these, i.e.,

$$g_{i_1} \text{ should be } \tilde{G}_{i_1} \text{ and } \ldots \text{ and } g_{i_k} \text{ should be } \tilde{G}_{i_k} \ldots \text{ and } g_{i_K} \text{ should be } \tilde{G}_{i_K} \quad (4.1)$$

where, $i_k \in \{1, \ldots, M\}$ (for $k = 1, \ldots, K$) are the indices of the system performance measures used as vague targets, g_{i_k} is the linguistic variable associated with the i_k-th

system performance measure, and $\tilde{G}_{i_k}$ is the desired linguistic value for g_{i_k}. The degree of satisfaction of the k-th target for the p-th individual is given by $\mu_{\tilde{G}_{i_k}}(\mathbf{f}_{i_k}(\mathbf{x}^p)) \in [0,1]$ for $k = 1, \ldots, K$. To simplify the notation, we denote $\mu_{\tilde{G}_{i_k}}(\mathbf{f}_{i_k}(\mathbf{x}^p))$ by $\mu_{\tilde{G}_{i_k}}(\mathbf{x}^p)$. A fully satisfied goal has value of 1. Fuzzification in OSEA can also be seen as a function that maps the M-dimensional vector of observed performance levels obtained via simulation to a K-dimensional vector of degree of satisfaction of the vague targets, i.e., $\mathbf{f}(\mathbf{x}^p) \rightarrow \vec{\mu}(\mathbf{x}^p)$.

To illustrate the use of fuzzy goals, in a simulation optimization model of a telecommunications network, the desired level of performance of the system might be stated as "*low cell–loss* and a *high throughput*", where the terms *low* and *high* are the desired vague values for the system performance measures *cell–loss* and *throughput*, respectively.

4.1.4 Fitness Assignment

Evolutionary algorithms for multicriteria optimization differ mainly in the way of assigning a fitness value to individuals in the population of solutions [150]. Several different aggregation methods have been reported [63], with the most commonly used involving the reduction of the multicriteria evaluation into a single scalar by means of a linear combination. This linear combination is formed using previously specified weights, proportional to the importance of each criterion. This aggregation approach's main strengths are its computational efficiency and ease of implementation; whereas its main weakness is the difficulty in determining appropriate weights [29].

As the name suggests, OSEA (Optimal Scoring Evolutionary Algorithm) uses a methodology that we call *optimal scoring* for fitness assignment. The basic idea of *optimal scoring* is to evaluate each element of the expanded population $E(t)$ using a flexible aggregation approach. Optimal scoring addresses the main weakness of aggregation approaches, namely the difficulty to agree upon a set of fixed weights, by assigning "optimal" weights to each individual of the population under evaluation. The idea is to let each individual choose weights which give it as high a score as possible, subject to assigning weights and scores within the allowable ranges for all individuals. After all the individuals in $E(t)$ are optimally evaluated, their optimal scores become their fitness assignment in the evolutionary algorithm.

Optimal scoring was developed following the principles of *data envelopment anal-*

ysis [22, 30]. Data Envelopment Analysis (DEA) is a linear programming based technique for measuring the relative performance of units, where the presence of multiple inputs and outputs makes comparisons difficult. Contrary to classical DEA models, optimal scoring considers outputs (i.e., system performance measures) exclusively. However, as in DEA, optimal scoring requires the solution of a linear program [37] for each individual (i.e., unit) under evaluation. This seems to be a very high price to pay in the context of an evolutionary algorithm, but as it will be seen later, it is possible to streamline the computation significantly.

For $k_r \in \{1, \ldots, K\}$ ($r \in \{1, \ldots, K\}$), let $\mu_{\tilde{G}_{i_{k_r}}}(\mathbf{x}^p)$ be the evaluated degree of satisfaction of the k_r-th target for the p-th individual in the expanded population $E(t)$ (see Section 4.1.3). Let $\overline{\mu_{\tilde{G}_{i_{k_1}}}(\mathbf{x}^p)}$, $\overline{\mu_{\tilde{G}_{i_{k_1}}}(\mathbf{x}^p) \cdot \mu_{\tilde{G}_{i_{k_2}}}(\mathbf{x}^p)}$, ..., and $\overline{\mu_{\tilde{G}_{i_{k_1}}}(\mathbf{x}^p) \cdot \mu_{\tilde{G}_{i_{k_2}}}(\mathbf{x}^p) \cdot \ldots \cdot \mu_{\tilde{G}_{i_{k_K}}}(\mathbf{x}^p)}$, be the *normalized* counterparts for $\mu_{\tilde{G}_{i_{k_1}}}(\mathbf{x}^p)$, $\mu_{\tilde{G}_{i_{k_1}}}(\mathbf{x}^p) \cdot \mu_{\tilde{G}_{i_{k_2}}}(\mathbf{x}^p)$, ..., and $\mu_{\tilde{G}_{i_{k_1}}}(\mathbf{x}^p) \cdot \mu_{\tilde{G}_{i_{k_2}}}(\mathbf{x}^p) \cdot \ldots \cdot \mu_{\tilde{G}_{i_{k_K}}}(\mathbf{x}^p)$, respectively, where the normalized fuzzified levels of performance are defined by $\overline{\mu_{\tilde{G}_{i_{k_1}}}(\mathbf{x}^p)} = \frac{\mu_{\tilde{G}_{i_{k_1}}}(\mathbf{x}^p)}{\max_{p \in E(t)} \mu_{\tilde{G}_{i_{k_1}}}(\mathbf{x}^p)}$ (for $k_1 \in \{1, \ldots, K\}$), $\overline{\mu_{\tilde{G}_{i_{k_1}}}(\mathbf{x}^p) \cdot \mu_{\tilde{G}_{i_{k_2}}}(\mathbf{x}^p)} = \frac{\mu_{\tilde{G}_{i_{k_1}}}(\mathbf{x}^p) \cdot \mu_{\tilde{G}_{i_{k_2}}}(\mathbf{x}^p)}{\max_{p \in E(t)} \left(\mu_{\tilde{G}_{i_{k_1}}}(\mathbf{x}^p) \cdot \mu_{\tilde{G}_{i_{k_2}}}(\mathbf{x}^p) \right)}$ (for $k_1, k_2 \in \{1, \ldots, K\}, k_1 < k_2$)), ... , $\overline{\mu_{\tilde{G}_{i_{k_1}}}(\mathbf{x}^p) \cdot \mu_{\tilde{G}_{i_{k_2}}}(\mathbf{x}^p) \cdot \ldots \cdot \mu_{\tilde{G}_{i_{k_K}}}(\mathbf{x}^p)} = \frac{\mu_{\tilde{G}_{i_{k_1}}}(\mathbf{x}^p) \cdot \mu_{\tilde{G}_{i_{k_2}}}(\mathbf{x}^p) \cdot \ldots \cdot \mu_{\tilde{G}_{i_{k_K}}}(\mathbf{x}^p)}{\max_{p \in E(t)} \left(\mu_{\tilde{G}_{i_{k_1}}}(\mathbf{x}^p) \cdot \mu_{\tilde{G}_{i_{k_2}}}(\mathbf{x}^p) \cdot \ldots \cdot \mu_{\tilde{G}_{i_{k_K}}}(\mathbf{x}^p) \right)}$ (for $k_1, k_2, \ldots, k_K \in \{1, \ldots, K\}$, $k_1 < k_2 < \ldots < k_K$).

Let β_{k_1} be the weight for the k_1-th criterion, $\beta_{k_1 k_2}$ (for $k_1 < k_2$) be the weight for the interaction between the k_1-th and k_2-th criteria, ..., and $\beta_{k_1 k_2 \ldots k_K}$ (for $k_1 < k_2 < \ldots < k_K$) be the weight for the interaction among all criteria. The interactions among criteria are considered to avoid *speciation*, i.e., the selection of individuals who excel in just one dimension [29]. Let l_{k_1} and u_{k_1} be the lower and upper bounds for β_{k_1}, respectively; let $l_{k_1 k_2}$ and $u_{k_1 k_2}$ (for $k_1 < k_2$) be the lower and upper bounds for $\beta_{k_1 k_2}$, respectively; ...; and $l_{k_1 k_2 \ldots k_K}$ and $u_{k_1 k_2 \ldots k_K}$ (for $k_1 < k_2 < \ldots < k_K$) the lower and upper bounds for $\beta_{k_1 k_2 \ldots k_K}$, respectively. By letting the β's be the decision variables, the *optimal score* (fitness assignment) for the p'-th individual in the expanded population $E(t)$, denoted $z^*(\mathbf{x}^{p'})$, is determined by solving the following linear program:

$$\begin{aligned}\max\ z(\mathbf{x}^{p'}) \quad = \quad & \textstyle\sum_{k_1=1}^{K} \beta_{k_1} \cdot \overline{\mu_{\tilde{G}_{i_{k_1}}}(\mathbf{x}^{p'})} + \underbrace{\sum_{k_1=1}^{K}\sum_{k_2=1}^{K}}_{k_1<k_2} \beta_{k_1k_2} \cdot \overline{\mu_{\tilde{G}_{i_{k_1}}}(\mathbf{x}^{p'}) \cdot \mu_{\tilde{G}_{i_{k_2}}}(\mathbf{x}^{p'})} + \ldots \\ & + \underbrace{\sum_{k_1=1}^{K}\sum_{k_2=1}^{K}\cdots\sum_{k_K=1}^{K}}_{k_1<k_2<\ldots<k_K} \beta_{k_1k_2\ldots k_K} \cdot \overline{\mu_{\tilde{G}_{i_{k_1}}}(\mathbf{x}^{p'}) \cdot \mu_{\tilde{G}_{i_{k_2}}}(\mathbf{x}^{p'}) \cdot \ldots \cdot \mu_{\tilde{G}_{i_{k_K}}}(\mathbf{x}^{p'})}\end{aligned} \tag{4.2}$$

subject to:

$$\begin{aligned}& \textstyle\sum_{k_1=1}^{K} \beta_{k_1} \cdot \overline{\mu_{\tilde{G}_{i_{k_1}}}(\mathbf{x}^{p})} + \underbrace{\sum_{k_1=1}^{K}\sum_{k_2=1}^{K}}_{k_1<k_2} \beta_{k_1k_2} \cdot \overline{\mu_{\tilde{G}_{i_{k_1}}}(\mathbf{x}^{p}) \cdot \mu_{\tilde{G}_{i_{k_2}}}(\mathbf{x}^{p})} + \ldots \\ & + \underbrace{\sum_{k_1=1}^{K}\sum_{k_2=1}^{K}\cdots\sum_{k_K=1}^{K}}_{k_1<k_2<\ldots<k_K} \beta_{k_1k_2\ldots k_K} \cdot \overline{\mu_{\tilde{G}_{i_{k_1}}}(\mathbf{x}^{p}) \cdot \mu_{\tilde{G}_{i_{k_2}}}(\mathbf{x}^{p}) \cdot \ldots \cdot \mu_{\tilde{G}_{i_{k_K}}}(\mathbf{x}^{p})} \quad \leq \quad 1 \quad \text{for all } x^p \in E(t)\end{aligned} \tag{4.3}$$

$$\sum_{k_1=1}^{K} \beta_{k_1} + \underbrace{\sum_{k_1=1}^{K}\sum_{k_2=1}^{K}}_{k_1<k_2} \beta_{k_1k_2} + \ldots + \underbrace{\sum_{k_1=1}^{K}\sum_{k_2=1}^{K}\cdots\sum_{k_K=1}^{K}}_{k_1<k_2<\ldots<k_K} \beta_{k_1k_2\ldots k_K} = 1 \tag{4.4}$$

$$\left.\begin{array}{lclcll} l_{k_1} & \leq & \beta_{k_1} & \leq & u_{k_1} & \text{for all } k_1 \in \{1,\ldots,K\} \\ l_{k_1k_2} & \leq & \beta_{k_1k_2} & \leq & u_{k_1k_2} & \text{for all } k_1,k_2 \in \{1,\ldots,K\},\ k_1<k_2 \\ & & \vdots & & & \\ l_{k_1k_2\ldots k_K} & \leq & \beta_{k_1k_2\ldots k_K} & \leq & u_{k_1k_2\ldots k_K} & \text{for all } k_1,k_2,\ldots,k_K \in \{1,\ldots,K\}, \\ & & & & & k_1<k_2<\ldots<k_K \end{array}\right\} \tag{4.5}$$

$$\left.\begin{array}{lcll} \beta_{k_1} & \geq & 0 & \text{for all } k_1 \in \{1,\ldots,K\} \\ \beta_{k_1k_2} & \geq & 0 & \text{for all } k_1,k_2 \in \{1,\ldots,K\},\ k_1<k_2 \\ & & \vdots & \\ \beta_{k_1k_2\ldots k_K} & \geq & 0 & \text{for all } k_1,k_2,\ldots,k_K \in \{1,\ldots,K\},\ k_1<k_2<\ldots<k_K \end{array}\right\} \tag{4.6}$$

Note that (4.2) represents the maximization of the optimal score of the p'-th individual in $E(t)$. This optimal score is a nonlinear aggregation function of the degree of satisfaction of the observed performance levels, but a linear function of the weights (β's). The constraints in (4.3) force the assigned scores of all the individuals to be within the allowable range between 0 to 1, while maximizing the score for the p'-th individual. Furthermore, (4.4), (4.5), and (4.6), force the weights to be greater than 0, to fall within user–defined ranges, and to add up to 100%, respectively. Note that additional constraints can be added to the model to include other problem specific issues. It is worth mentioning that when the lower bounds equal the upper bounds in (4.5), optimal scoring reduces to aggregation with fixed weights.

One linear program must be solved for each $x^{p'} \in E(t)$. Solving $|E(t)|$ linear programs just for fitness assignment could be a very expensive process. However, we can streamline the solution of the linear program described in (4.2)-(4.6) by using the following Proposition:

Proposition 1. The constraints in (4.3) are redundant.

Proof. By definition, $0 \leq \mu_{\tilde{G}_{i_{k_r}}}(\mathbf{x}^p) \leq 1$. By (4.4) and (4.6), $0 \leq \beta_{k_1} \leq 1$, $0 \leq \beta_{k_1 k_2} \leq 1$, ..., $0 \leq \beta_{k_1 k_2 \ldots k_K} \leq 1$. Thus, the constraints in (4.3) are always satisfied. $\diamond$

By dropping the constraints in (4.3), the resulting linear program can be solved efficiently using the following result.

Proposition 2. Let $c_1, c_2, \ldots, c_{2^K-1}$ be the objective function coefficients in (4.2), where $c_1 = \overline{\mu_{\tilde{G}_{i_1}}(\mathbf{x}^p)}$, $c_2 = \overline{\mu_{\tilde{G}_{i_2}}(\mathbf{x}^p)}$, $\ldots$, $c_K = \overline{\mu_{\tilde{G}_{i_K}}(\mathbf{x}^p)}$, $\ldots$, $c_{K+1} = \overline{\mu_{\tilde{G}_{i_1}}(\mathbf{x}^p) \cdot \mu_{\tilde{G}_{i_2}}(\mathbf{x}^p)}$, $c_{K+2} = \overline{\mu_{\tilde{G}_{i_1}}(\mathbf{x}^p) \cdot \mu_{\tilde{G}_{i_3}}(\mathbf{x}^p)}$, $\ldots$, $c_{K+\frac{K(K-1)}{2!}} = \overline{\mu_{\tilde{G}_{i_{K-1}}}(\mathbf{x}^p) \cdot \mu_{\tilde{G}_{i_K}}(\mathbf{x}^p)}$, $\ldots$, $c_{2^K-1} = \overline{\mu_{\tilde{G}_{i_1}}(\mathbf{x}^p) \cdot \mu_{\tilde{G}_{i_2}}(\mathbf{x}^p) \cdot \ldots \cdot \mu_{\tilde{G}_{i_K}}(\mathbf{x}^p)}$. Let the decision variables of the linear program defined by (4.2), (4.4), (4.5), and (4.6), be $w_1, w_2, \ldots, w_{2^K-1}$, where $w_1 = \beta_1$, $w_2 = \beta_2$, $\ldots$, $w_K = \beta_K$, $w_{K+1} = \beta_{1\,2}$, $w_{K+2} = \beta_{1\,3}$, $\ldots$, $w_{K+\frac{K(K-1)}{2!}} = \beta_{K-1\,K}$, $\ldots$, $w_{2^K-1} = \beta_{1\,2\,\ldots\,K}$. Let us rank the variables by a "primed indexing" so that $c_{1'} \geq c_{2'} \geq \ldots \geq c_{(2^K-1)'}$, where $1', 2', \ldots, (2^K-1)'$ constitute a permutation of the numbers $1, 2, \ldots, (2^K-1)$. Then the linear program defined by (4.2), (4.4), (4.5), and (4.6), is solved optimally by taking $w_{1'}$ as large as possible without violating (4.4) and the bounds in (4.5) and (4.6), then taking $w_{2'}$ as large as possible subject to the value already assigned $w_{1'}$ and so on.

Proof. The linear program defined by (4.2), (4.4), (4.5), and (4.6) is a "bounded variable" knapsack problem that can be solved using the "Greedy Algorithm" [50].$\diamond$

In summary, optimal scoring in OSEA defines a mapping from the K-dimensional vector of degree of satisfaction of vague targets to a scalar representing the fitness level of the p-th individual in $E(t)$. Mathematically, it is a mapping $\vec{\mu}(\mathbf{x}^p) \rightarrow z^*(\mathbf{x}^p)$.

4.1.4.1 Example

To better understand how the fitness assignment in OSEA works, let us consider an example with two system performance measures ($M = 2$), two criteria ($K = 2$), and membership functions given by

$$\mu_{\tilde{G}_1}(\mathbf{x}^p) = \begin{cases} 0 & \text{if } f_1(\mathbf{x}^p) < 5.9 \\ \frac{f_1(\mathbf{x}^p)-5.9}{19.6} & \text{if } 5.9 \leq f_1(\mathbf{x}^p) \leq 25.5 \\ 1 & \text{if } f_1(\mathbf{x}^p) > 25.5 \end{cases} \tag{4.7}$$

and

$$\mu_{\tilde{G}_2}(\mathbf{x}^p) = \begin{cases} 0 & \text{if } f_2(\mathbf{x}^p) < 3.9 \\ \frac{f_2(\mathbf{x}^p)-3.9}{21.6} & \text{if } 3.9 \leq f_2(\mathbf{x}^p) \leq 25.5 \\ 1 & \text{if } f_2(\mathbf{x}^p) > 25.5 \end{cases} \tag{4.8}$$

The observed evaluations (see Section 4.1.2) and corresponding fuzzifications (see Section 4.1.3) for each of the individuals in $E(t)$ are given in Table 4.1. The performance measure evaluations $f_1(\mathbf{x}^p)$ and $f_2(\mathbf{x}^p)$ are based on the first example presented by Zitzler and Thiele [150] to illustrate the fitness assignment mechanism in their Strength Pareto Evolutionary Algorithm (SPEA).

In this example we let each criterion account for 20% to 40% of the total score, while their interaction accounts for 30% to 50%.

For the 7-th individual, the linear program defined by (4.2), (4.4), (4.5), and (4.6), is given by:

$$\begin{aligned} \max z(\mathbf{x}^7) &= 0.362\beta_1 + 0.560\beta_2 + 0.378\beta_{1\,2} \\ \text{subject to:} & \\ & \beta_1 + \beta_2 + \beta_{1\,2} = 1 \\ & 0.20 \leq \beta_1 \leq 0.40 \\ & 0.20 \leq \beta_2 \leq 0.40 \\ & 0.30 \leq \beta_{1\,2} \leq 0.50 \\ & \beta_1, \beta_2, \beta_{1\,2} \geq 0 \end{aligned} \tag{4.9}$$

Table 4.1: Evaluated and fuzzified performance measures for $E(t)$.

p	Evaluation		Fuzzification			
	$f_1(\mathbf{x}^p)$	$f_2(\mathbf{x}^p)$	$\mu_{\tilde{G}_1}(\mathbf{x}^p)$ $\left(= \overline{\mu_{\tilde{G}_1}(\mathbf{x}^p)}\right)$	$\mu_{\tilde{G}_2}(\mathbf{x}^p)$ $\left(= \overline{\mu_{\tilde{G}_2}(\mathbf{x}^p)}\right)$	$\mu_{\tilde{G}_1}(\mathbf{x}^p) \cdot \mu_{\tilde{G}_2}(\mathbf{x}^p)$	$\overline{\mu_{\tilde{G}_1}(\mathbf{x}^p) \cdot \mu_{\tilde{G}_2}(\mathbf{x}^p)}$
1	5	5	0.000	0.051	0.000	0.000
2	14	3	0.413	0.000	0.000	0.000
3	25	4	0.974	0.005	0.005	0.008
4	30	10	1.000	0.282	0.282	0.527
5	18	12	0.617	0.375	0.232	0.432
6	7	15	0.056	0.514	0.029	0.054
7	13	16	0.362	0.560	0.203	0.378
8	20	20	0.719	0.745	0.536	1.000
9	6	25	0.005	0.977	0.005	0.009
10	10	30	0.209	1.000	0.209	0.390

Using Proposition 2, $\beta_1^* = 0.2$, $\beta_2^* = 0.4$, and $\beta_{1\,2}^* = 0.4$. Thus, the optimal score $z^*(\mathbf{x}^7) = 0.378$.

Table 4.2 compares the fitness assignments obtained through OSEA and SPEA [150]. The second column shows the optimal fitness assignments obtained in OSEA by solving a linear program for each individual in $E(t)$. The third column, shows the results for the fitness assignment in SPEA presented by Zitzler and Thiele [150]. The column labeled *Dominated by Nondominated* shows the number of nondominated vectors that dominate the criteria vector associated with the p-th individual. The last two columns assign a rank based on the sorted fitness assignment for each algorithm. Note that the two rankings are very similar and lead to the exact same tiers in the *Dominated by Nondominated* column.

Figure 4.3 presents graphically the optimal fitness assignments in criteria space. Note that the fitness values obtained by optimal scoring are consistent in the Pareto sense. For instance, the individuals in the first tier (i.e., ▲), which are nondominated, are fitter than those in the third tier (i.e., ■), which are dominated by at least two nondominated vectors in $E(t)$.

In conclusion, the fitness assignment mechanism in OSEA is consistent with the one used in SPEA. Points closer to the Pareto front are more fit than those which are

Table 4.2: Comparison of fitness assignment methods.

p	OSEA Fitness $z^*(\mathbf{x}^p)$	Zitzler and Thiele's SPEA	Dominated by Nondominated	Ranking	
				OSEA	SPEA
8	0.636	0.625	0	1	3
4	0.569	0.375	0	2	2
10	0.526	0.375	0	3	1
5	0.415	1.625	1	4	6
9	0.394	1.375	1	5	5
3	0.393	1.375	1	6	4
7	0.378	1.625	1	7	7
6	0.231	2	2	8	8
2	0.165	2	2	9	9
1	0.02	2.375	3	10	10

farther from it [150].

4.1.5 Selection

OSEA selects P_{max} individuals for generation t (i.e., $P(t)$), from the expanded population $E(t-1)$. For efficiency and to eliminate bias in the selection mechanism, we use stochastic universal sampling [7], as shown in the pseudocode shown in Figure 4.4.

4.1.6 Genetic Operators

For the mutation operator, we use the *uniform mutation* [91]. In the uniform mutation, every gene x_j^p (for $j = \{1, \ldots, N\}$) from each individual $\mathbf{x}^p \in P(t)$ has a chance p_m of undergoing the mutation process. The result of the application of this operator on the j'-th gene, implies that the new child will have a new gene $x_{j'}^p$ randomly drawn from the allowable range $X_{j'}$. Thus, for any single parent, more than one gene can be selected to undergo the mutation process, thereby generating a single child with several mutated genes.

For the crossover operator, we use the *one–point crossover* [91]. Each individual $\mathbf{x}^p \in P(t)$ has a chance of p_c of becoming a parent in the crossover process. If

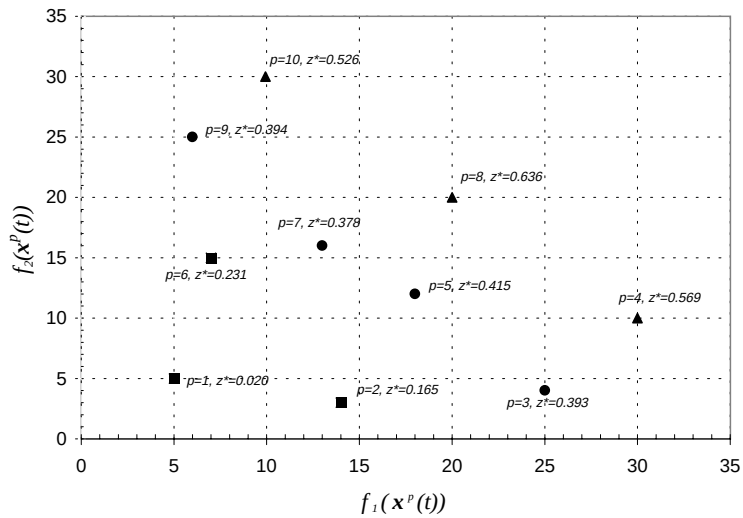


Figure 4.3: Optimal fitness in performance space. The symbols ▲, ●, and ■ represent the first, second, and third tier of optimal scores, respectively.

$P(t) \leftarrow \emptyset$

$\delta \leftarrow \frac{1}{P_{max}}$

$\Delta \leftarrow \delta \cdot u$, where $u \sim U[0,1]$

$\sigma \leftarrow 0$

`for each` $\mathbf{x}^p \in E(t-1)$ `do`

  $P\{\text{choosing } \mathbf{x}^p\} = \frac{z^*(\mathbf{x}^p)}{\sum_{\mathbf{x}^q \in E(t-1)} z^*(\mathbf{x}^q)}$

  $\sigma \leftarrow \sigma + P\{\text{choosing } \mathbf{x}^p\}$

  `while` $\sigma > \Delta$ `do`

    $P(t) \leftarrow P(t) \cup \{\mathbf{x}^p\}$

    $\Delta \leftarrow \Delta + \delta$

`return` $P(t)$ ■

Figure 4.4: Stochastic Universal Sampling

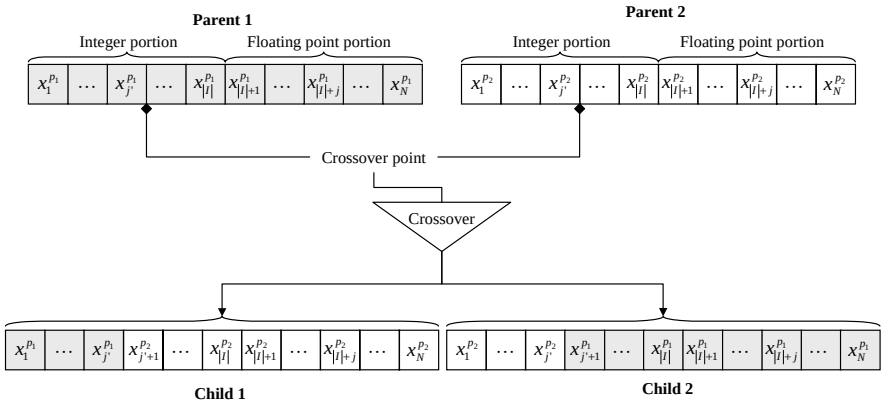


Figure 4.5: One–point crossover in OSEA.

the individual is selected for crossover, we say that it joins the *crossover pool* of parents. From the crossover pool, pairs of parents are randomly selected to undergo the crossover operation, until the crossover pool is empty (or there are not enough parents to form a pair). Figure 4.5 illustrates the case where the crossover point has been randomly chosen to split the parents after the j'-th integer gene.

4.1.7 Approximate Pareto Optimal Set

As recommended by Van Veldhuizen and Lamont [134], OSEA was designed to maintain a secondary population consisting of the Pareto optimal solutions obtained so far, namely $\mathcal{P}^*_{\text{approximate}}(t)$. OSEA not only maintains $\mathcal{P}^*_{\text{approximate}}(t)$, but uses this information in the elitism mechanism, discussed in Section 4.1.8, to improve the quality of the population.

Let $\vec{\mu}(\mathbf{x}^p) = \left(\mu_{\tilde{G}_{i_1}}(\mathbf{x}^p), \ldots, \mu_{\tilde{G}_{i_K}}(\mathbf{x}^p)\right)$ be the criterion vector of fuzzified observed performance levels for $\mathbf{x}^p$ (Section 4.1.3). Let $\hat{\mathbf{x}}^p$ be the p-th individual in $\mathcal{P}^*_{\text{approximate}}(t)$. Also let $flag = 1$, if the p-th individual in $P(t)$ is efficient; $flag = 0$, otherwise. The following pseudocode describes a mechanism to update $\mathcal{P}^*_{\text{approximate}}(t)$ based on $P(t)$.

4.1.8 Elitism

Traditionally, *elitism* has been used in single objective evolutionary optimization as a selection mechanism that preserves the best individual for the next generation,

Step 0: Initialization.

$B \leftarrow P(t)$.

Step 1: Termination.

`if` $B \leftarrow \emptyset$, `then`

`return` $\mathcal{P}^*_{\text{approximate}}(t)$.

`stop`.

Step 2: Checking Pareto optimality.

Pick any $\mathbf{x}^p \in B$ and `do`:

$flag \leftarrow 1$

`if` $\mathcal{P}^*_{\text{approximate}}(t) = \emptyset$, `then` go to Step 3.

`for each` $\hat{\mathbf{x}}^p \in \mathcal{P}^*_{\text{approximate}}(t)$ `do`:

`if` $\vec{\mu}(\mathbf{x}^p) \succ \vec{\mu}(\hat{\mathbf{x}}^p)$, `then` $\mathcal{P}^*_{\text{approximate}}(t) \leftarrow \mathcal{P}^*_{\text{approximate}}(t) \setminus \{\hat{\mathbf{x}}^p\}$.

`if` $\vec{\mu}(\hat{\mathbf{x}}^p) \succ \vec{\mu}(x^p)$, `then` $flag \leftarrow 0$ and go to Step 3.

Step 3: Updating the approximate Pareto optimal set.

`if` $flag = 1$, `then` $\mathcal{P}^*_{\text{approximate}}(t) \leftarrow \mathcal{P}^*_{\text{approximate}}(t) \cup \{\mathbf{x}^p\}$.

$B \leftarrow B \setminus \{\mathbf{x}^p\}$ and go to Step 1. ■

Figure 4.6: Updating $\mathcal{P}^*_{\text{approximate}}(t)$

so that errors of sampling are corrected [48, 49]. Zitzler et al. [149] have found that elitism is an important factor in evolutionary mulicriteria optimization. To implement elitism in our algorithm, we use the approximate Pareto optimal set $\mathcal{P}^*_{\text{approximate}}(t)$ as source of individuals to be preserved. Elitism is implemented in OSEA as a random technique that reinserts elements of $\mathcal{P}^*_{\text{approximate}}(t)$ into the expanded population $E(t)$. The probability of selecting an individual from $\mathcal{P}^*_{\text{approximate}}(t)$ through the elitism mechanism is p_e.

4.2 Application to the Flow Line Design Problem and Comparison with the Fuzzy Controlled Approach

In this section we present a scenario in which the flow line, depicted in Figure 3.5, is designed based on two conflicting goals which we wish to satisfy simultaneously. Specifically, we want to design the flow line to achieve "low work–in–process" and "high utilization at station 1", simultaneously.

The main purpose of this example is to compare the fuzzy controlled approach presented in Chapter 3 and OSEA, in terms of both the quality of the approximate Pareto front produced and algorithmic efficiency.

As described in Section 3.4.5.2, for the fuzzy controlled approach we developed a rule base composed of 18 rules (Appendix B). The state variables used as vague targets were overall work–in–process (ϖ) and utilization at station 1 (φ_1) with target values $\tilde{L}_\varpi$ (Figure 3.7) and $\tilde{H}_{\varphi_1}$ (Figure 3.8), respectively. Five independent experiments were conducted. Each experiment consisted of ten runs of the algorithm shown in Figure 3.3, with randomly selected initial conditions. For each run, we set $g_\varpi = 1$, $g_{\varphi_1} = 1$, and $t_{\max} = 10$.

For OSEA, six independent experiments were conducted. In each experiment, we set $P_{max} = 20$, $t_{max} = 100$, $p_c = 0.40$, $p_m = 0.05$, and p_e=0.10. These parameters represent in our experience the best settings for performance. Table 4.3 gives the bounds on the weights used in the optimal scoring mechanism for fitness assignment.

One of the most challenging problems in multicriteria optimization is to measure the quality of the solutions obtained [29]. Here we use a performance metric recently proposed by Zitzler and Thiele [150] to compare the approximate Pareto fronts ob-

Table 4.3: Bounds on weights for OSEA experiments.

Experiment	Bounds					
	β_1 ($\tilde{H}_{\varphi_1}$)		β_2 ($\tilde{L}_{\varpi}$)		$\beta_{1\,2}$ ($\tilde{H}_{\varphi_1}$ and $\tilde{L}_{\varpi}$)	
	l_1	u_1	l_2	u_2	$l_{1\,2}$	$u_{1\,2}$
1	0.25	0.45	0.25	0.45	0.15	0.30
2	0	1	0	1	0	1
3	0.25	0.45	0.25	0.45	0.25	0.45
4	0.15	0.30	0.15	0.30	0.30	0.60
5	0.30	0.60	0.15	0.30	0.15	0.30
6	0.15	0.30	0.30	0.60	0.15	0.30

tained by the fuzzy controlled approach with those obtained by OSEA. Zitzler and Thiele's metric is a measure of the size of the dominated space defined by the approximate Pareto front $\mathcal{PF}^*_{\text{approximate}}$ generated by each approach. In our case, where two criteria are considered, each vector in $\mathcal{PF}^*_{\text{approximate}}$, dominates a rectangular area. The dominated space metric is estimated by calculating the area defined by the union of the collection of rectangles defined by $\mathcal{PF}^*_{\text{approximate}}$. After fuzzification, the minimum and maximum degree of satisfaction are 0 and 1, respectively. If we were to obtain an *ideal solution*, one that obtains full satisfaction of both criteria simultaneously, the value of Zitzler and Thiele's metric would be 1. On the other hand, if we were to obtain only solutions for which both criteria are not satisfied at all, the metric would be 0. Therefore, in our case, Zitzler and Thiele's metric is bounded by 0 and 1, with higher values preferred over lower values. Moreover, 1 is not actually attainable due to the conflicting nature of the criteria.

The performance comparison is shown by Figures 4.7 to 4.9. From Figure 4.7 we can see that both approaches have the ability to generate high quality solutions (with Zitzler and Thiele's metric equal to 0.965), but the fuzzy controlled approach's metric achieves its peak much faster than that of OSEA. More specifically, the fuzzy controlled approach takes about 25 iterations to reach a high quality solution while the evolutionary approach takes about 60 iterations to reach the same level.

Figure 4.8 shows that for every iteration, the fuzzy controlled approach requires far fewer simulation runs than OSEA. Specifically, the fuzzy controlled approach

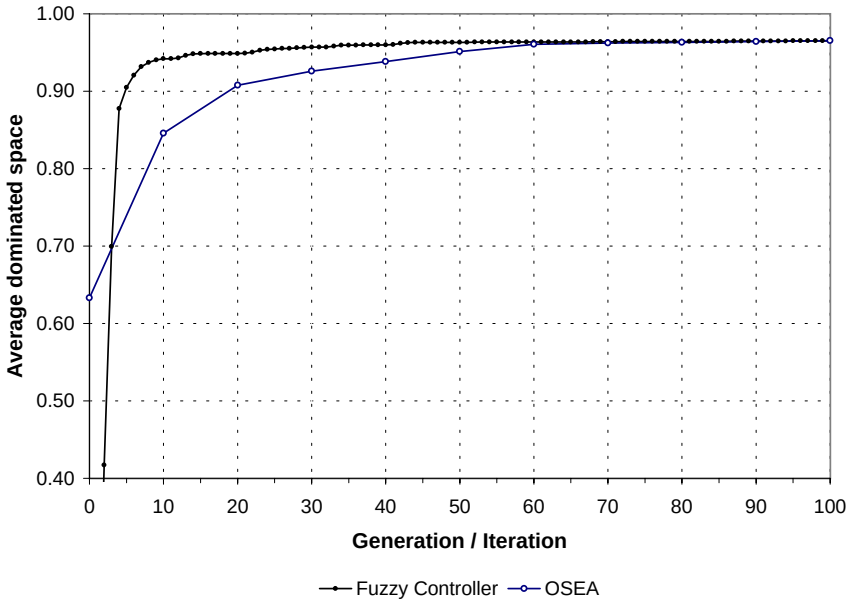


Figure 4.7: Comparison of the dominated space.

requires 25 simulation runs (in 25 iterations) to generate a high quality solution while the evolutionary approach requires about 798 simulation runs (in 60 generations) to generate a solution of the equivalent quality.

Figure 4.9 further confirms that both approaches have the capability to generate a well distributed and solid Pareto front.

For further detail, the results for each experiment using the fuzzy controlled approach are given in Appendix C and for OSEA in Appendix D.

4.3 Conclusions

We have presented OSEA, an evolutionary approach for solving the FMSO problem. OSEA is able to generate the approximate Pareto optimal set, using a methodology that we called *optimal scoring* for fitness assignment. By using a flexible aggregating approach, OSEA uses optimal scoring to assign a fitness value to each individual of the population. Optimal scoring overcomes the main weakness of an aggregation approach, the difficulty in selecting a set of predefined fixed weights. The idea behind optimal scoring is to let each individual freely choose its weights so that it can score as high as possible, while satisfying simultaneously basic ground rules applicable to

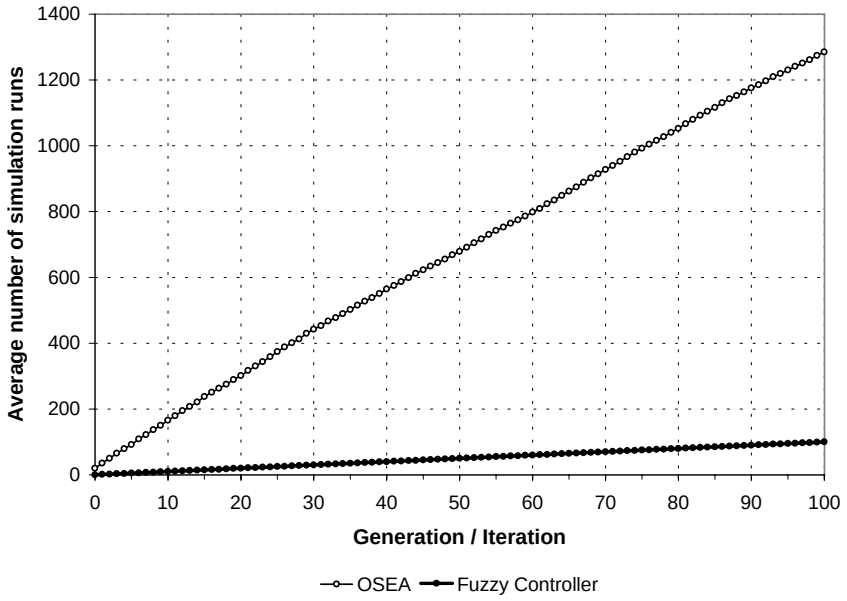


Figure 4.8: Comparison of the number of simulation runs.

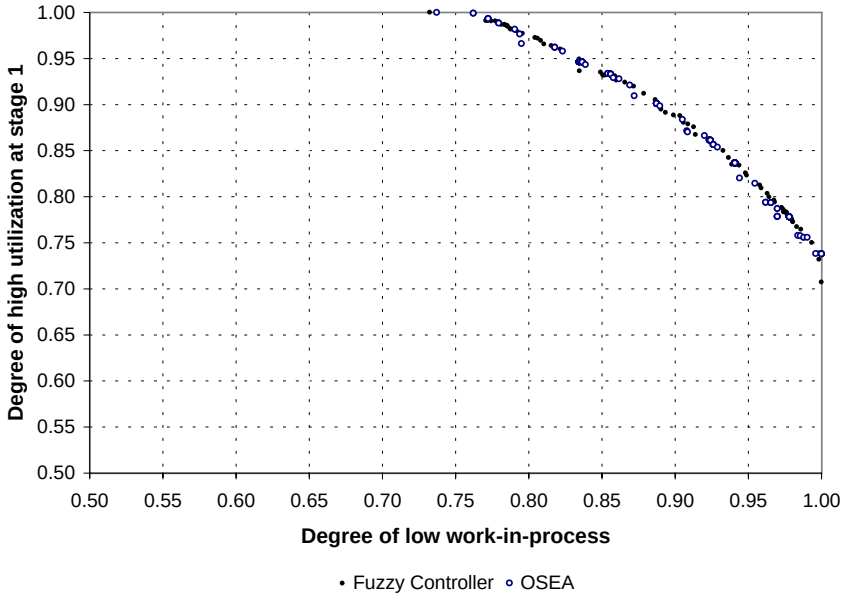


Figure 4.9: Comparison of the Pareto front.

all individuals. In order to find the optimal weights, optimal scoring requires the solution of a linear program for each individual. We have shown that by exploiting the structure of this linear program, the optimal score (i.e., fitness) can be found efficiently. Our fitness assignment mechanism is validated by comparison with that in SPEA [150].

We have also compared the results obtained by OSEA with those obtained with the fuzzy controlled approach presented in Chapter 3 on the Flow Line Design problem. Both algorithms are able to generate a high quality solution in terms of the dominated space metric proposed by Zitzler and Thiele [150]. Also, both algorithms are able to generate an even sample of the Pareto front. The fuzzy controlled approach is remarkably efficient, requiring far fewer simulation runs than the evolutionary approach to obtain the same solution quality. Moreover, it requires very few iterations to obtain an acceptable solution. This directly translates into big savings in computer time. Nevertheless, we have to say in behalf of OSEA, that it starts the search for the Pareto front from scratch, without assuming any knowledge of the system being optimized. In contrast, the fuzzy controlled approach uses the knowledge of the system embedded in the rule base. In cases where there is a complete lack of knowledge of the relationships between inputs and performance of the system, OSEA is the only known alternative for solving the FMSO problem.

Chapter 5

An Efficient and Flexible Mechanism for Constructing Membership Functions

In the two previous chapters we have developed and compared methods for the solution of problems FSO and FMSO. In illustrating our methods, we used the popular, but limited special case of convex normal fuzzy sets defined by trapezoidal membership functions. To further enhance the expressive power of vague concepts in our methods, it is necessary to provide the user with an efficient mechanism to express the complexities and subtleties of natural language. This chapter introduces a Bézier curve–based mechanism for constructing membership functions of convex normal fuzzy sets that can represent virtually any vague concept. The mechanism can fit any given data set with a minimum level of discrepancy. In the absence of data, the mechanism can be intuitively manipulated by the user to construct membership functions with the desired shape. Some numerical experiments are included to compare the performance of the proposed mechanism with conventional methods.

5.1 Introduction

A fuzzy set $\tilde{A}$ is characterized by its *membership function* $\mu_{\tilde{A}}$, which maps each element of the universe X to the interval $[0, 1]$. This function indicates the degree of belonging to $\tilde{A}$ for each element of X. One of the most important concepts of fuzzy

sets is the concept of an α-cut. Given a fuzzy set $\tilde{A}$ defined on X and $\alpha \in (0, 1]$, the α-cut is defined as ${}^{\alpha}\tilde{A} = \{x \in X : \mu_{\tilde{A}}(x) \geq \alpha\}$. For continuity purposes, we take ${}^{0}\tilde{A} = \lim_{\alpha \to 0} {}^{\alpha}\tilde{A}$. A fuzzy set $\tilde{A}$ is *convex* if and only if each of its α-cuts is a convex set. A fuzzy set $\tilde{A}$ is *normal* if ${}^{1}\tilde{A} \neq \emptyset$.

Even though there is no universal agreement on the proper characterization of membership functions, Dombi [32] reported that there are some characteristics shared by the majority of continuous membership functions found in the literature. Among others, there is an apparent demand for membership functions with the following properties: they should be piecewise monotone nonincreasing or nondecreasing; they should achieve null and full membership for at least two different elements in the universal set; and they should be able to represent fuzzy convex sets. Commonly seen examples are the simple triangular, trapezoidal, and bell–shaped membership functions.

Problem formulations based on fuzzy sets can have greater expressive power than their counterparts based on crisp sets, but the applicability of fuzzy technology depends on the ability to construct membership functions that appropriately represent various concepts in different contexts [78]. To fully exploit the benefits provided by fuzzy technology, we need an efficient membership function generating mechanism with the following desirable characteristics:

1. *Accurate.* In the presence of data, the resulting membership functions should reflect the knowledge contained in the data in the most accurate way possible. Data in the form of membership values for points in the universe is usually obtained from experts.

2. *Flexible.* The methodology should provide a broad family of membership functions.

3. *Computationally affordable.* The method should be computationally tractable in order to be of any practical use. Medasani [89] has highlighted the importance of having membership functions that can be easily tuned and adjusted. Other authors have expressed the need for methods in which computer graphics can facilitate the process of constructing membership functions by allowing the user an easy and direct manipulation of different shapes [20].

4. *Easy to use.* Once a membership function has been generated, it should be easy

to find $\mu_{\tilde{A}}(x)$ for a given x; and it should be easy to find ${}^{\alpha}\tilde{A}$ for a given α.

In this chapter we propose a mechanism that exploits the properties of Bézier curves to address these issues and to provide the user with a flexible and efficient way of generating membership functions.

The chapter is organized as follows. In Section 5.2, we review the basic techniques used for generating membership functions. Section 5.3 describes the proposed mechanism and some fundamental definitions and properties of Bézier curves. In Section 5.4, test problems found in the literature are used to illustrate the proposed mechanism and compare its performance with that of two methods which appear in the literature. Finally, conclusions and current research directions are given in Section 5.5.

5.2 Membership Function Generation

5.2.1 Overview

Membership functions can be constructed from data when it is available. This data can be elicited by interacting with experts using a *direct approach* (or *direct rating*) [78, 97, 132]. The direct approach requires the degree of membership of a collection of points in the universal set. A membership function that describes the underlying concept is fitted to the collected set. This is known as *data–driven membership function estimation*. Sometimes this approach can be overly precise in capturing subjective judgment. By formulating easier and simpler questions, knowledge can also be acquired through an indirect approach. We will not deal with the indirect approach in this chapter, but the reader is referred to the paper by Chameau and Santamarina [20] and the book by Klir and Yuan [78].

When data is not available in the form of value–membership pairs, a membership function has to be constructed subjectively. In this case, the conventional approach is to first pick the shape of the membership function from a list of families, and then to fine–tune the values of the parameters of that function. It is always desirable to have a parsimonious, meaningful parameterization of membership functions [32].

5.2.2 Current Methods

In the literature fuzzy sets are most commonly modeled by triangular, trapezoidal, and bell–shaped membership functions. However, other parameterized functional shapes are useful in particular situations. More details can be found in Dombi [32] and Medasani et al. [89].

An effort to create a broad class of functions was made by Zysno [151] and Zimmermann and Zysno [148]. In their model, the membership function for a fuzzy set $\tilde{A}$ is given by:

$$\mu_{\tilde{A}}(x) = \text{mid}\left(0, \left(\frac{1}{1+e^{-a(x+b)}} - c\right)\frac{1}{d} + \frac{1}{2}, 1\right), \quad \forall x \in X \subseteq R \tag{5.1}$$

where $a, b \in R$, $0 \leq c \leq 1$, and $0 \leq d \leq 2\min(1-c, c)$. The function $\text{mid}(0, f(x), 1)$ is defined such that $\text{mid}(0, f(x), 1) = f(x)$, if $0 \leq f(x) \leq 1$; $\text{mid}(0, f(x), 1) = 0$, if $f(x) < 0$; and $\text{mid}(0, f(x), 1) = 1$, if $f(x) > 1$.

Even though the model provides the user with a commonly used family of S-shapes, the determination of the parameters from empirical data poses some problems and there is no direct numerical method for optimal parameter estimation [148, 151]. The model may be used for estimating membership functions subjectively, with the parameters a, b, c, and d, being fixed by the expert.

Dombi [32] proposed a model with properties similar to the one presented by Zysno and Zimmermann. In his model a membership function for fuzzy set $\tilde{A}$ is constructed using the S-shaped monotonically increasing function

$$\mu_{\tilde{A}}(x) = \frac{(1-\nu)^{\lambda-1}(x-a)^{\lambda}}{(1-\nu)^{\lambda-1}(x-a)^{\lambda} + \nu^{\lambda-1}(b-x)^{\lambda}} \tag{5.2}$$

and/or the S-shaped monotonically decreasing function

$$\mu_{\tilde{A}}(x) = \frac{(1-\nu)^{\lambda-1}(b-x)^{\lambda}}{(1-\nu)^{\lambda-1}(b-x)^{\lambda} + \nu^{\lambda-1}(x-a)^{\lambda}} \tag{5.3}$$

where $x \in [a, b]$; $a, b \in R$; the steepness is given by $\lambda \geq 1$; and the inflection point is determined by $0 < \nu < 1$. When data is available, Dombi proposed a method for estimating the parameters based on linearized forms of (5.2) and (5.3).

Both of these models provide similar membership functions because they use the same underlying form, i.e., $\mu_{\tilde{A}}(x) = \frac{1}{1+d(x)}$, where $d(x)$ is a measure of distance. Even

though these models provide flexibility for estimating S-shaped functions, they fail to provide more general monotonic curves.

Chen and Otto [23] present a novel method for constructing membership functions using interpolation and measurement theory. Following a systematic approach, their method is able to construct general monotonic functions from data. However, their methodology does not provide a mechanism for adjusting or building a membership function in the absence of data.

In the area of fuzzy system identification, sophisticated methods based on neural networks and evolutionary algorithms have been proposed to generate and tune both fuzzy rules and membership functions. However, they are basically case by case approaches [73, 83].

In the next section we shall introduce an interactive and efficient approach for both data–driven and subjective estimation of membership functions. Based on Bézier curves, the method is able to generate a broad family of functions.

5.3 Proposed Mechanism

5.3.1 Bézier Curves

One of the major breakthroughs in computer aided design (CAD) is the theory of Bézier curves and surfaces, independently developed by P. de Casteljau and P. Bézier while working for the French automakers Citröen and Renault, respectively [38].

The theory of Bézier curves provides a mathematical foundation for representing a smooth curve that passes through the vicinity of a set of *control points*. Definition 1 gives a formal expression of a Bézier curve in terms of Bernstein polynomials.

Definition 1. A Bézier curve with $n+1$ control points $\mathbf{p} \triangleq (\mathbf{p}_0, \ldots, \mathbf{p}_n)$ is given by

$$f(t, n, \mathbf{p}) \triangleq \sum_{k=0}^{n} B_{n,k}(t)\mathbf{p}_k$$

where $t \in [0, 1]$, $\mathbf{p}_k \triangleq (x_k, y_k)^T$, and $B_{n,k}(t) = \begin{pmatrix} n \\ k \end{pmatrix} t^k(1-t)^{n-k}$ are the Bernstein polynomials. Since $f(t, n, \mathbf{p}) \in R^2$, we usually denote $f(t, n, \mathbf{p}) = [f_x(t, n, \mathbf{x}), f_y(t, n, \mathbf{y})]^T$, where $\mathbf{x} \triangleq (x_0, \ldots, x_n)^T$, $\mathbf{y} \triangleq (y_0, \ldots, y_n)^T$.

Bézier curves have several properties that are particularly useful in the context of this chapter [38].

Property 1. The Bézier curve $f(t, n, \mathbf{p})$ defined over $t \in [0, 1]$, lies in the convex hull of the polygon defined by the control points $\mathbf{p} \triangleq (\mathbf{p}_0, \ldots, \mathbf{p}_n)$.

Property 2. The Bernstein polynomial $B_{n,k}(t)$ achieves its unique maximum at $t = k/n$. If the control point $\mathbf{p}_k$ is moved, then the curve is mostly affected in the region around the parameter $t = k/n$.

Property 3. The Bézier curve interpolates its first ($\mathbf{p}_0$) and last ($\mathbf{p}_n$) control points. In other words, $f(0, n, \mathbf{p}) = \mathbf{p}_0$ and $f(1, n, \mathbf{p}) = \mathbf{p}_n$.

These properties have practical effects in the curve design process. Property 1 guarantees that the curve will not fall outside the "*control polygon*". By using this property along with Property 2, a Bézier curve can be designed by exaggerating the target shape using the control polygon. Even though a single control point displacement will change the whole curve, this "*pseudo–local control*" property gives us the sense that the control points work locally as magnets on the curve. Property 3 is very useful for breaking the construction of a complex curve into simpler parts.

A complete discussion on Bézier curves and its properties can be found in the book by Farin [38].

5.3.2 Mathematical Framework

In this section we give the mathematical framework of a broad family of membership shapes based on Bézier curves.

Let $\tilde{A}$ be a fuzzy set on the universal set X. The following conditions are commonly required for its membership function, $\mu_{\tilde{A}}(\cdot)$.

Condition 1. The membership function $\mu_{\tilde{A}}$ is a mapping from the universal set X to $[0, 1]$, i.e., $\mu_{\tilde{A}} : X \to [0, 1]$.

Condition 2. There exist $x_1, x_2 \in X$ such that $\mu_{\tilde{A}}(x_1) = 1$ and $\mu_{\tilde{A}}(x_2) = 0$. In other words, we say that $x_1 \in X$ fully belongs to the set $\tilde{A}$, while $x_2 \in X$ does not belong to $\tilde{A}$.

Condition 3. For $x_1, x_2 \in X$ and $\lambda \in [0, 1]$, we have $\mu_{\tilde{A}}(\lambda x_1 + (1 - \lambda)x_2) \geq \min\{\mu_{\tilde{A}}(x_1), \mu_{\tilde{A}}(x_2)\}$.

Condition 1 is conventional in the fuzzy literature. The *normality* requirement implicit in Condition 2 (i.e., existence of $x \in X$ such that $\mu_{\tilde{A}}(x) = 1$) can be easily relaxed, but we preserve it for the sake of clarity in our presentation. Condition 3 guarantees that the fuzzy set $\tilde{A}$ is *convex*.

A convenient, parametric form for expressing our membership function model is:

$$\mu_{\tilde{A}}(x(t)) = \begin{cases} 0 & \text{if } x(t) < m_L - \gamma \\ \mu_{\tilde{A}_L}(x(t)) & \text{if } m_L - \gamma \leq x(t) \leq m_L \\ 1 & \text{if } m_L < x(t) < m_R \\ \mu_{\tilde{A}_R}(x(t)) & \text{if } m_R \leq x(t) \leq m_R + \beta \\ 0 & \text{if } x(t) > m_R + \beta \end{cases} \tag{5.4}$$

where γ and β are the left and right spreads, respectively; $m_L, m_R \in X$ are the lowest and highest values with full membership, respectively; and $\mu_{\tilde{A}_L}(x(t))$ and $\mu_{\tilde{A}_R}(x(t))$ are the left and right membership values. Assume that $\mathbf{p}_L = (\mathbf{p}_{L,0}, \ldots, \mathbf{p}_{L,n_L})^T$ and $\mathbf{p}_R = (\mathbf{p}_{R,0}, \ldots, \mathbf{p}_{R,n_R})^T$ are n_L+1 and n_R+1 control points for generating the left and right membership functions, respectively. The left and right membership functions are part of the following parametric expressions:

$$\begin{aligned} [x(t), \mu_{\tilde{A}_L}(x(t))]^T &= \vec{\mu}_{\tilde{A}_L}(t, n_L, \mathbf{p}_L) \\ &\triangleq \sum_{k=0}^{n_L} B_{n_L,k}(t)\mathbf{p}_{L,k} \end{aligned} \tag{5.5}$$

$$\begin{aligned} [x(t), \mu_{\tilde{A}_R}(x(t))]^T &= \vec{\mu}_{\tilde{A}_R}(t, n_R, \mathbf{p}_R) \\ &\triangleq \sum_{k=0}^{n_R} B_{n_R,k}(t)\mathbf{p}_{R,k} \end{aligned} \tag{5.6}$$

where $\vec{\mu}_{\tilde{A}_L}(\cdot)$ and $\vec{\mu}_{\tilde{A}_R}(\cdot)$ are the Bézier curves for the left and right membership functions, respectively; $t \in [0, 1]$; $\mathbf{p}_{L,k} \triangleq (x_{L,k}, y_{L,k})^T$ is the k-th Bézier control point for the left membership function (for $k = 0, \ldots, n_L$); $\mathbf{p}_{R,k} \triangleq (x_{R,k}, y_{R,k})^T$ is the k-th Bézier control point for the right membership function (for $k = 0, \ldots, n_R$); and $B_{n_L,k}(t)$ and $B_{n_R,k}(t)$ are Bernstein polynomials. As before, in two–dimensional space, we denote $\vec{\mu}_{\tilde{A}_L}(t, n_L, \mathbf{p}_L) = [f_x(t, n_L, \mathbf{x}_L), f_y(t, n_L, \mathbf{y}_L)]^T$ and $\vec{\mu}_{\tilde{A}_R}(t, n_R, \mathbf{p}_R) = [f_x(t, n_R, \mathbf{x}_R), f_y(t, n_R, \mathbf{y}_R)]^T$, where $\mathbf{x}_L \triangleq (x_{L,0}, \ldots, x_{L,n_L})^T$, $\mathbf{y}_L \triangleq (y_{L,0}, \ldots, y_{L,n_L})^T$, $\mathbf{x}_R \triangleq (x_{R,0}, \ldots, x_{R,n_R})^T$, and $\mathbf{y}_R \triangleq (y_{R,0}, \ldots, y_{R,n_R})^T$.

The type of shapes that can be obtained using the family of membership functions described by (5.4) are presented in Figure 5.1.

In order to satisfy Conditions 1 to 3 we need to impose some restrictions on the parametric form expressed by (5.5) and (5.6).

For Conditions 1 and 2,

Proposition 3. The first and last control points of $\vec{\mu}_{\tilde{A}_L}(\cdot)$ are $\mathbf{p}_{L,0} = (m_L - \gamma, 0)^T$ and $\mathbf{p}_{L,n_L} = (m_L, 1)^T$.

Proof. It follows from Property 3 of the Bézier curves. $\diamond$

Proposition 4. The first and last control points of $\vec{\mu}_{\tilde{A}_R}(\cdot)$ are $\mathbf{p}_{R,0} = (m_R, 1)^T$ and $\mathbf{p}_{R,n_R} = (m_R + \beta, 0)^T$.

Proof. It follows from Property 3 of the Bézier curves. $\diamond$

For Condition 3,

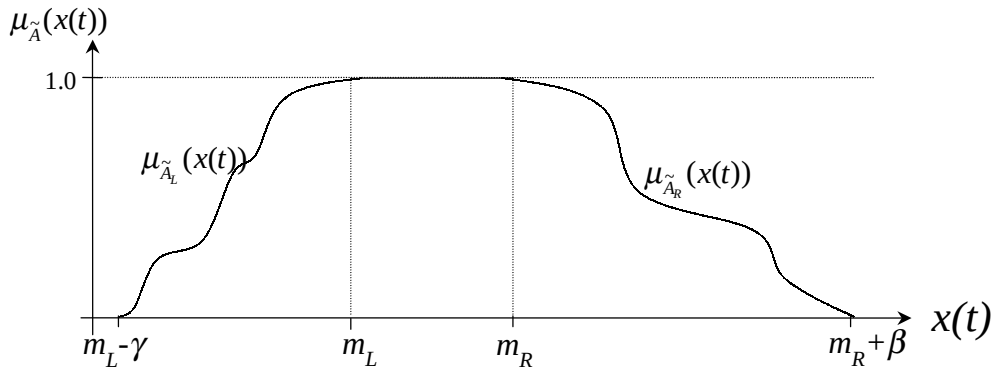
Proposition 5. If the control points $\mathbf{p}_L$ of $\vec{\mu}_{\tilde{A}_L}(\cdot)$ are chosen such that $x_{L,0} \leq \ldots \leq x_{L,n_L}$ and $y_{L,0} \leq \ldots \leq y_{L,n_L}$, then $\mu_{\tilde{A}_L}(x(t))$ is monotonically nondecreasing for $m_L - \gamma \leq x(t) \leq m_L$ and $x(t)$ is monotonically nondecreasing for $0 \leq t \leq 1$.

Proof.

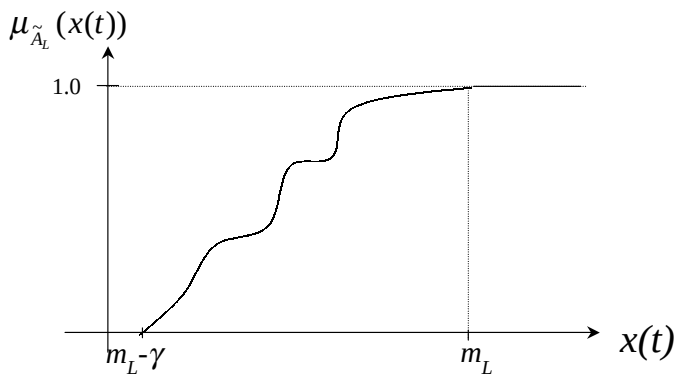
$$\begin{aligned} f_y^{'}(t, n_L, \mathbf{p}_L) &= \mu_{\tilde{A}_L}^{'}(x(t)) \\ &= \lim_{\delta \to 0} \frac{\frac{\mu_{\tilde{A}_L}(x(t+\delta)) - \mu_{\tilde{A}_L}(x(t))}{\delta}}{\frac{x(t+\delta)-x(t)}{\delta}} \\ &= \frac{f_y^{'}(t, n_L, \mathbf{y}_L)}{f_x^{'}(t, n_L, \mathbf{x}_L)} \\ &= \frac{\sum_{k=0}^{n_L} n_L [B_{n_L-1,k-1}(t) - B_{n_L-1,k}(t)] y_k}{\sum_{k=0}^{n_L} n_L [B_{n_L-1,k-1}(t) - B_{n_L-1,k}(t)] x_k} \\ &= \frac{\sum_{k=0}^{n_L-1} B_{n_L-1,k}(t) \Delta y_k}{\sum_{k=0}^{n_L-1} B_{n_L-1,k}(t) \Delta x_k} \end{aligned} \tag{5.7}$$

where $t \in [0, 1]$, $\Delta y_k \triangleq y_{k+1} - y_k$, and $\Delta x_k \triangleq x_{k+1} - x_k$, for $k = 0, \ldots, n_L - 1$. From the result in (5.7), if $\Delta y_k \geq 0$ and $\Delta x_k \geq 0$, then $\mu_{\tilde{A}_L}^{'}(x(t)) \geq 0$. Thus we conclude that $\mu_{\tilde{A}_L}(x(t))$ is monotonically nondecreasing. $\diamond$

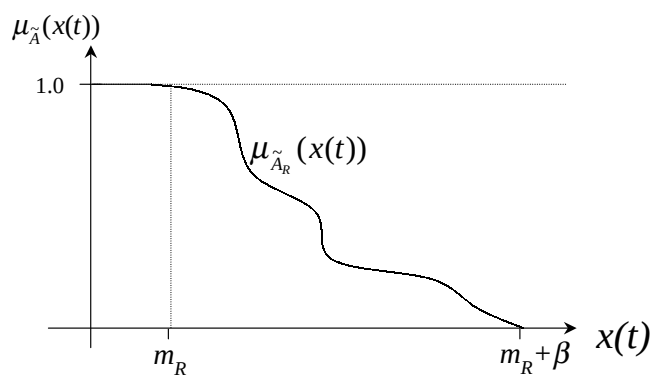
The basic results used in the proof of Proposition 5 can be found in Farin [38] and Wagner and Wilson [137].



(a) Monotonic nondecreasing and nonincreasing.



(b) Monotonic nondecreasing (left).



(c) Monotonic nonincreasing (right).

Figure 5.1: Types of membership functions.

Similar to Proposition 5, the following result applies for the monotonically nonincreasing membership function, $\vec{\mu}_{\tilde{A}_R}(\cdot)$.

Proposition 6. If the control points $\mathbf{p}_R$ of $\vec{\mu}_{\tilde{A}_R}(\cdot)$ are chosen such that $x_{R,0} \leq \ldots \leq x_{R,n_R}$ and $y_{R,0} \geq \ldots \geq y_{R,n_R}$, then $\mu_{\tilde{A}_R}(x(t))$ is monotonically nonincreasing for $m_R \leq x(t) \leq m_R + \beta$ and $x(t)$ is monotonically nondecreasing for $0 \leq t \leq 1$.

The next result follows from Propositions 5 and 6.

Proposition 7. If the control points $\mathbf{p}_L$ of $\vec{\mu}_{\tilde{A}_L}(\cdot)$ are chosen such that $x_{L,0} \leq \ldots \leq x_{L,n_L}$ and $y_{L,0} \leq \ldots \leq y_{L,n_L}$; and the control points $\mathbf{p}_R$ of $\vec{\mu}_{\tilde{A}_R}(\cdot)$ are chosen such that $x_{R,0} \leq \ldots \leq x_{R,n_R}$ and $y_{R,0} \geq \ldots \geq y_{R,n_R}$, then the fuzzy set $\tilde{A}$ is convex and satisfies Condition 3.

5.3.3 Methodology

5.3.3.1 Basic Operations

In the previous section we imposed conditions on the placement of the control points to guarantee the generation of membership functions that satisfy Conditions 1 to 3. It remains to discuss how to calculate $\mu_{\tilde{A}}(x)$ given x and ${}^{\alpha}\tilde{A}$ given α. Assuming the location of the control points $\mathbf{p}_L$ and $\mathbf{p}_R$ are known, the algorithms presented in Figures 5.2 and 5.3 can be used.

The computational burden of the algorithm presented in Figure 5.2 is the solution a root finding problem on a polynomial of degree n_L or n_R. This problem can be solved efficiently using the bisection method [24] or the methods proposed by Müller or Laguerre [102].

Again the computational bottleneck of the algorithm presented in Figure 5.3 is a root finding problem on a polynomial of degree n_L or n_R.

5.3.3.2 Data–driven Estimation

In a direct approach to knowledge acquisition, experts are required to provide the degree of membership for each of a collection of points in the universal set [78]. The resulting set of value–membership pairs is used to construct the membership function of the underlying concept. This section provides a mechanism for constructing membership functions from data by determining the number of control points and their locations in the $(x, \mu(x))$ space.

`if` $x \leq m_L - \gamma$ `or` $x \geq m_R + \beta$

  `then` $\mu_{\tilde{A}}(x) = 0.$

`if` $m_L \leq x \leq m_R$

  `then` $\mu_{\tilde{A}}(x) = 1.$

`if` $m_L - \gamma < x < m_L$

  `then`

    Find $t \in [0, 1]$ such that

    $\sum_{k=0}^{n_L} \begin{pmatrix} n_L \\ k \end{pmatrix} t^k (1-t)^{n_L - k} x_{L,k} = x$

    and compute

    $\mu_{\tilde{A}}(x) = \sum_{k=0}^{n_L} \begin{pmatrix} n_L \\ k \end{pmatrix} t^k (1-t)^{n_L - k} y_{L,k}.$

`if` $m_R < x < m_R + \beta$

  `then`

    Find t such that

    $\sum_{k=0}^{n_R} \begin{pmatrix} n_R \\ k \end{pmatrix} t^k (1-t)^{n_R - k} x_{R,k} = x$

    and compute

    $\mu_{\tilde{A}}(x) = \sum_{k=0}^{n_R} \begin{pmatrix} n_R \\ k \end{pmatrix} t^k (1-t)^{n_R - k} y_{R,k}.$

`return` $\mu_{\tilde{A}}(x)$. ■

Figure 5.2: Finding $\mu_{\tilde{A}}(x)$ given x

```
if α = 0
  then l ← m_L − γ, u ← m_R + β.
else
  if α = 1
    then l ← m_L, u ← m_R.
  else
    if γ ≠ 0
      then
        Find t ∈ [0,1] such that
        Σ_{k=0}^{n_L} (n_L choose k) t^k (1−t)^{n_L−k} y_{L,k} = α
        and compute
        x = Σ_{k=0}^{n_L} (n_L choose k) t^k (1−t)^{n_L−k} x_{L,k}.
        set l ← x.
    else l ← m_L
    if β ≠ 0
      then
        Find t such that
        Σ_{k=0}^{n_R} (n_R choose k) t^k (1−t)^{n_R−k} y_{R,k} = α
        and compute
        x = Σ_{k=0}^{n_R} (n_R choose k) t^k (1−t)^{n_R−k} x_{R,k}.
        set u ← x.
    else u ← m_R
set ^α Ã ← [l, u].
return ^α Ã. ■
```

Figure 5.3: Finding ${}^{\alpha}\tilde{A}$ given α

The left side of the membership function can be estimated independently from the right side. We formulate a mathematical model and propose an algorithm for estimating the monotonically nonincreasing portion (right side) of a membership function. A similar approach can be used for estimating the nondecreasing (left side) portion.

Let the given data points be $\mathbf{d}_{R,i} = (\check{x}_{R,i}, \check{y}_{R,i})^T$ for $i = 1, \ldots, M_R$, where M_R is the total number of data points and $\check{y}_{R,i}$ is the membership given by the expert through the direct approach to the i-th value $\check{x}_{R,i} \in X$. Without loss of generality, assume there are at least three data points (i.e., $M_R \geq 3$) which are sorted in ascending order by their first component. Also let the $n_R + 1$ control points be $\mathbf{p}_R = ((x_{R,0}, y_{R,0}) \ldots (x_{R,n_R}, y_{R,n_R}))^T$.

Let the decision variables be $x_{R,k}$ and $y_{R,k}$, the first and second coordinates of the k-th control point ($k = 0, \ldots, n_R$); t_i, the parameter value of the Bézier curve for the i-th data point ($i = 1, \ldots, M_R$); and n_R, the maximum value of the index associated with the control points to be placed. By Proposition 4, the first and last control points are fixed in $\mathbf{p}_{R,0} = (\check{x}_1, 1)^T$ and $\mathbf{p}_{R,n_R} = (\check{x}_{M_R}, 0)^T$. Thus the final value of some variables is known before performing any optimization, namely, $x_{R,0} = \check{x}_{R,1}$, $y_{R,0} = 1$, $x_{R,n_R} = \check{x}_{R,M_R}$, $y_{R,n_R} = 0$, $t_1 = 0$, and $t_{M_R} = 1$.

The following mathematical program minimizes the sum of the squared errors (SSE) between the fitted membership function and the empirical data.

$$\min \sum_{i=2}^{M_R-1} \left(\check{y}_{R,i} - \sum_{k=0}^{n_R} \binom{n_R}{k} t_i^k (1-t_i)^{n_R-k} y_{R,k} \right)^2 \tag{5.8}$$

subject to:

$$
\left.
\begin{array}{lcll}
\sum_{k=0}^{n_R} \begin{pmatrix} n_R \\ k \end{pmatrix} t_i^k (1-t_i)^{n_R-k} x_{R,k} = \check{x}_{R,i}, & & & \text{for } i = 2, \ldots, M_R - 1 \\
t_i & \leq & t_{i+1}, & \text{for } i = 1, \ldots, M_R - 1 \\
x_{R,k} & \leq & x_{R,k+1}, & \text{for } k = 0, \ldots, n_R - 1 \\
y_{R,k} & \geq & y_{R,k+1}, & \text{for } k = 0, \ldots, n_R - 1 \\
x_{R,k} & \geq & \check{x}_{R,1}, & \text{for } k = 1, \ldots, n_R - 1 \\
x_{R,k} & \leq & \check{x}_{R,M_R}, & \text{for } k = 1, \ldots, n_R - 1 \\
y_{R,k} & \geq & 0, & \text{for } k = 1, \ldots, n_R - 1 \\
y_{R,k} & \leq & 1, & \text{for } k = 1, \ldots, n_R - 1 \\
t_i & \geq & 0, & \text{for } i = 2, \ldots, M_R - 1 \\
t_i & \leq & 1, & \text{for } i = 2, \ldots, M_R - 1
\end{array}
\right\} \tag{5.9}
$$

$$
n_R \in \{2, 3, \ldots\} \tag{5.10}
$$

The fact that the number of control points is unknown and integer increases dramatically the complexity of the problem described by (5.8), (5.9), and (5.10). Fortunately, in most practical applications the number of control points required is small. By treating this number as a parameter, we can solve a series of nonlinear programs, instead of dealing directly with a more difficult mixed integer nonlinear program. For a given n_R, the nonlinear program has $2n_R + M_R - 4$ continuous variables, $M_R - 2$ nonlinear constraints, $M_R + 2n_R - 1$ linear constraints, and $2(M_R + 2n_R - 4)$ lower and upper bounds.

Given n_R, let $e(n_R)$ be the sum of the square errors between the fitted membership function and the empirical data when $n_R + 1$ control points are used. Let $NLP(\mathbf{d}_R, n_R)$ be a function that solves the nonlinear program described by (5.8) and (5.9). $NLP(\mathbf{d}_R, n_R)$ takes the empirical data $\mathbf{d}_R$ and a specified value of n_R as its arguments and returns the optimal value of the objective function described in (5.8), $e(n_R)$, and the optimal locations of the control points, $\mathbf{p}_R(n_R)$. Then the algorithm shown in Figure 5.4 can be used to solve the data–driven estimation for the right membership functions. The algorithm stops when the improvement in SSE is less than a given small quantity ϵ_0 (say, $\epsilon_0 = 0.0010$) or when the maximum number of control points to be placed is reached.

```
set ε ← ε_0, n_R ← 1, e(1) ← +∞.
do
  n_R ← n_R + 1
  (e(n_R), p_R(n_R)) ← NLP(d_R, n_R)
  if e(n_R − 1) − e(n_R) ≤ ε or n_R = M_R − 1
    if e(n_R − 1) − e(n_R) < 0
      then return p_R(n_R − 1), e(n_R − 1), n_R − 1.
    else
      return p_R(n_R), e(n_R), n_R.
end
■
```

Figure 5.4: Data–driven estimation of the right membership function

5.4 Performance

5.4.1 Flexibility

In current practice, users choose the shape of the membership functions from a pool of commonly used parameterized families. After the shape is selected, the parameters are manipulated to tune the shape. As discussed in Section 5.2.1 the pool of parameterized families of membership functions include triangular, trapezoidal, Gaussian, generalized bell curve, sigmoid, and S-shaped. In contrast, our approach can be used to produce the membership function of almost any imprecise concept. Basically, our approach can be viewed as a generalized free form generator of membership functions that satisfy the basic requirements presented in Section 5.3.2.

The example in Table 5.1 and Figure 5.5 illustrates the ease with which a membership function can be constructed and tuned interactively using our approach. By placing the control points in the locations shown in Table 5.1, the membership function depicted in Figure 5.5(a), with the control points being represented by black dots, can be obtained. By changing the location of the second control point on the left side ($k = 1$) from $(25, 0.1)$ to $(15, 0.5)$, the curve bends toward the new point as if

Table 5.1: Control points (before change).

k	$\mathbf{p}_{L,k}$		$\mathbf{p}_{R,k}$	
	$x_{L,k}$	$y_{L,k}$	$x_{R,k}$	$y_{R,k}$
0	10	0.0	50	1.0
1	25	0.1	60	0.3
2	30	0.8	70	0.2
3	50	1.0	75	0.0

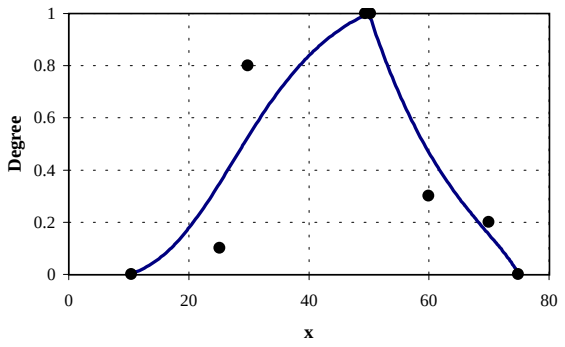
there were some magnetic attraction between the control point and the membership function (left portion). This is shown in Figure 5.5(b). Moreover, due to the Property 2 of the Bézier curves presented in Section 5.3.1, we observe that, even though this change affects the whole left membership function, the change is more noticeable in the vicinity of the control point.

This new flexible and interactive way of building and tuning a membership function can be leveraged by using a graphical user interface(GUI). Currently, we are developing a GUI that helps the user add, move, and delete control points to obtain the desired free–form membership function.

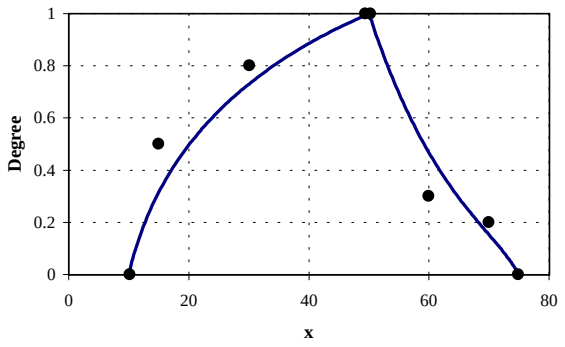
5.4.2 Numerical Examples

For data–driven estimation, we tested our approach using data originally published by Zysno [151] and compared its performance to that of the methods reported in Zysno [151] and Dombi [32]. Sixty–four persons from 21 to 25 years of age were asked to rate 52 different statements related to age concepts. The group was divided into four subgroups of 16. The individuals within a subgroup were asked to rate one of the 4 concepts: *very young man*, *young man*, *old man*, and *very old man*. The subjects were asked to give the degree of membership in the designated fuzzy set of a man of x years of age on a 0% to 100% scale.

Figure 5.6 shows the progress of our algorithm when applied to automatically estimate the membership functions for the fuzzy set *old man* based on the data collected by interviewing subject 35 in Zysno [151]. In the figure, a black square represents a control point. A number beside a control point is used when more than one control point shares the same location. The number represents the total number of control points in the given location. Empty circles represent data points. Lines are

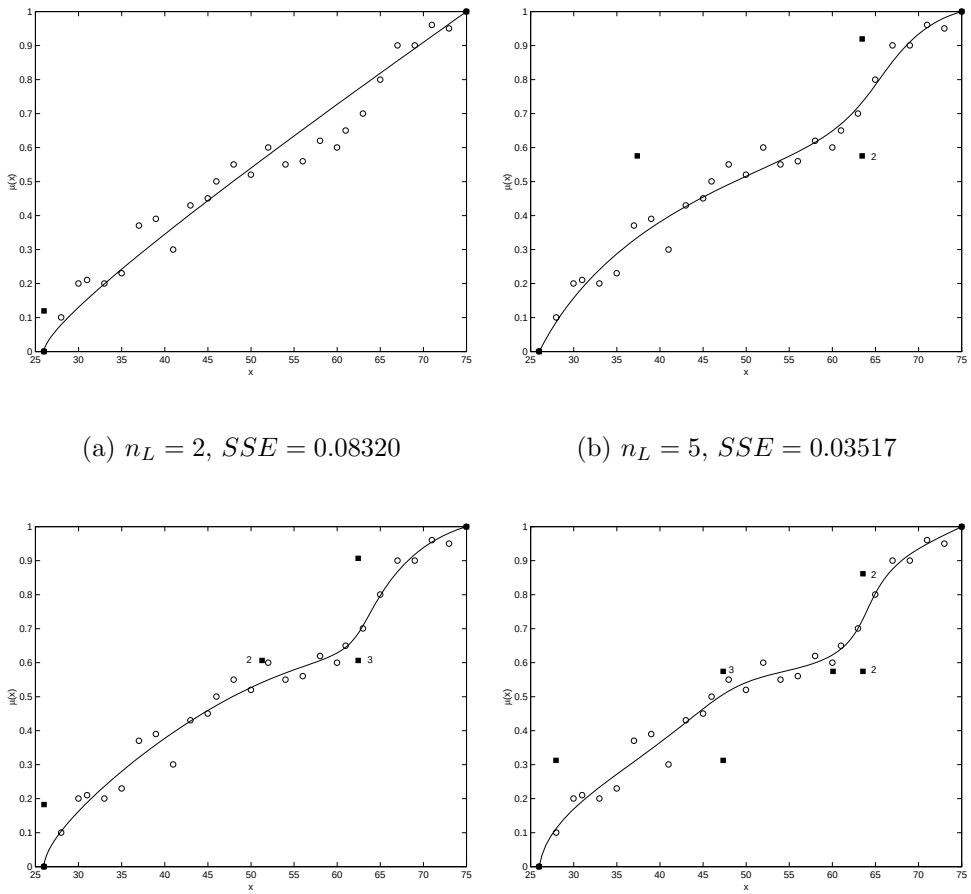


(a) Before.



(b) After.

Figure 5.5: Effect on the change of a single control point.



(a) $n_L = 2$, $SSE = 0.08320$

(b) $n_L = 5$, $SSE = 0.03517$

(c) $n_L = 8$, $SSE = 0.02846$

(d) $n_L = 11$, $SSE = 0.02469$

Figure 5.6: Data–driven estimation. Data: subject 35 (*Old man*).

used to display the estimated membership functions.

As is customary in the literature, to compare our method with those of Zysno and Dombi we used the sum of the squared errors (SSE) as the measure of goodness of fit between a membership function model and the empirical data. Dombi used data for subjects 9, 18, 35, 44 and 58 in Zysno [151] as his benchmark test cases and measured the corresponding SSEs. Zysno estimated the parameters of his model for all the data sets (64 subjects), but did not provide SSE as the measure of goodness of fit. In order to make valid comparisons, we calculated the SSE for Zysno's model for the benchmark test cases chosen by Dombi. Table 5.2 gives the SSE for the benchmark test cases for the three models, namely, Dombi, Zysno, and ours. The

Table 5.2: Sum of square errors (SSE)

Model	Data set (subject)				
	9	18	35	44	58
Zysno	0.10074	0.05054	0.17808	0.07572	0.02641
Dombi	0.13204	0.05103	0.14841	0.05284	0.03027
Proposed Approach ($\epsilon = 0.0010$)	0.07149	0.02390	0.02469	0.03610	0.01941

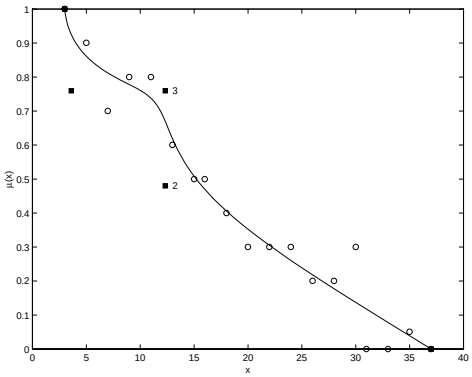
Table 5.3: SSE progress for the test benchmark cases ($\epsilon = 0.0010$; †: final solution).

Control points	Data set (subject)				
$(n_R + 1)$	9	18	35	44	58
3	0.09231	0.07353	0.08320	0.12333	0.06127
4	0.09044	0.04557	0.05242	0.04838	0.02100
5	0.08822	0.03420	0.04498	0.04071	0.01981
6	0.08167	0.02406	0.03517	0.03800	$0.01941^{\dagger}$
7	0.07538	$0.02390^{\dagger}$	0.03133	0.03665	
8	$0.07149^{\dagger}$		0.03011	$0.03610^{\dagger}$	
9	0.07428		0.02846		
10			0.02715		
11			0.02549		
12			$0.02469^{\dagger}$		

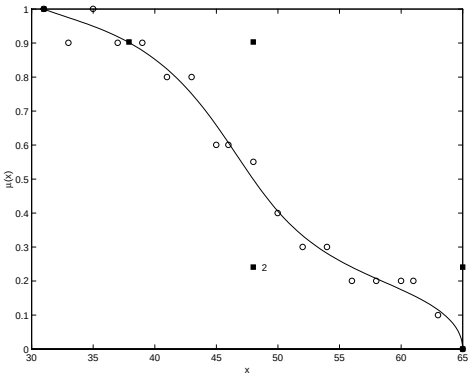
superior performance of our approach is clearly seen.

Table 5.3 shows the evolution of the SSE for each of the test benchmark cases when our data–driven estimation mechanism is used. The resulting estimated membership functions are shown in Figures 5.6 and 5.7. Note that most of the intermediate solutions shown in Table 5.3 are better than the final solutions provided by Zysno and Dombi. By monitoring the progress of the SSE, the algorithm may be interrupted as soon as the user is satisfied with the current SSE. We have a very small ϵ that may cause overfitting. However, our method for membership function generation can get arbitrarily close to the empirical data.

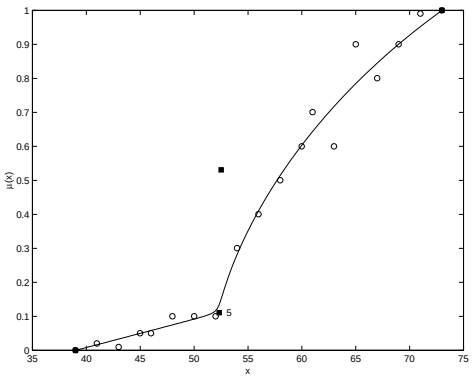
A final remark should be made. After fitting a membership function to data, the user can still go back and tune the membership by moving the control points as described in Section 5.4.1. This high level of interaction and flexibility between the



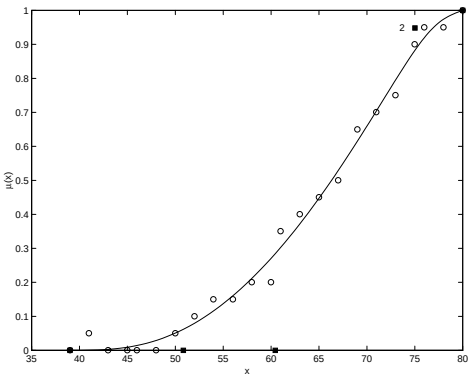
(a) Subject 9. $n_R = 7$, $SSE = 0.07149$



(b) Subject 18. $n_R = 6$, $SSE = 0.02390$



(c) Subject 44. $n_R = 7$, $SSE = 0.03610$



(d) Subject 58. $n_R = 5$, $SSE = 0.01941$

Figure 5.7: Data-driven estimation ($\epsilon = 0.0010$).

model and the user is a desirable feature when designing imprecise concepts.

5.4.3 Computational Efficiency

In the absence of data, our approach requires from the user the number and location of the control points. We have shown in Section 5.4.1 how easy it is to change the shape of the membership function by displacing the control points. It is important to note that the computational effort needed to redraw a membership function, whenever a control point is moved, is just the simple evaluation of (5.5) and (5.6).

Once a membership function has been generated (either with or without data), as shown in Section 5.3.3.1, the calculations required to find $\mu_{\tilde{A}}(x)$ for a given x and find ${}^{\alpha}\tilde{A}$ for a given α reduce to solving a computationally inexpensive root finding problem in a closed interval ($t \in [0, 1]$).

When our approach is used to fit membership functions to data, it was seen in Section 5.3.3.2 that the computational bottleneck is finding a solution to a nonlinear program with $2n_R+M_R-4$ variables, M_R-2 nonlinear constraints, M_R+2n_R-1 linear constraints, and $2(M_R+2n_R-4)$ lower and upper bounds. For the numerical examples presented in Section 5.4.2 we used AMPL as the algebraic modeling language and MINOS 5.4 as the nonlinear optimizer. Using a computer with a 266 MHz Pentium II processor, all the nonlinear programs took less than 4 seconds to run.

5.5 Conclusion

We have proposed a new mechanism based on Bézier curves for generating membership functions well suited for a broad spectrum of fuzzy modeling. By placing control points in different locations, the shape of the membership functions can be altered in a very natural and intuitive way. Mechanisms for dealing with subjective and data–driven estimation of membership functions were discussed. Some advantages of this approach are its flexibility, ease of use, computational efficiency, and suitability for a graphical interactive implementation. The major advantage is its immense power of fitting data as close as possible without a priori assumption of the shape of the function.

Several aspects of this work are in progress. First, a tailored interior point algorithm that can exploit the structure of the nonlinear program presented in (5.8) and

(5.9) is currently under investigation [37]. We are currently exploring the application of this methodology to the methods proposed in Chapters 3 and 4.

Chapter 6

Summary and Recommendations

6.1 Summary

In Chapter 1 we introduced simulation optimization, a fundamental problem which has attracted the attention of many researchers in the simulation community, in terms of the *single response simulation optimization* (SRSO) and *multiple response simulation optimization* (MRSO) problems. Since the goals for a system are often stated in vague natural language by the decision maker, we discussed the need for the incorporation of vague goals into simulation optimization. This led to the study of the *fuzzy single response simulation optimization* (FSO) and *fuzzy multiple response simulation optimization* (FMSO) problems.

In Chapter 2 we surveyed the literature on simulation optimization and gave an overview of soft computing. Within the framework of soft computing, we provided a comprehensive review of evolutionary algorithms for multicriteria optimization. We highlighted the absence of research on the synergistic merger of simulation optimization with soft computing techniques.

In Chapter 3 we proposed a fuzzy controlled method for solving FSO and FMSO problems. A distinctive feature of the proposed simulation optimization strategy is the use of approximate reasoning through a fuzzy controller to drive the optimization process, using a small set of rules that encapsulates the relevant knowledge of the system. Using these rules, which can be generated from statistical correlation measures and quadratic response surface models, we showed how the controller is able to drive a simulation model of a flow line, represented by a tandem of queues with

blocking, towards a high degree of satisfaction of one or more vaguely stated targets. Moreover, in the presence of multiple and conflicting goals, the proposed approach was able to construct a high quality approximate Pareto optimal solution set. The quality of the solution set was measured in terms of Zitzler and Thiele's metric based on the size of dominated space [150].

In Chapter 4 we provided an alternative evolutionary method for solving the FMSO problem. This evolutionary method (OSEA) provided a means to assess the quality of the approximate Pareto front generated by the fuzzy controlled approach proposed in Chapter 3. We showed how OSEA generates an approximate Pareto optimal set, using a DEA[1]–based methodology called *optimal scoring* for fitness assignment. We discussed how optimal scoring overcomes the main difficulty in selecting a set of predefined fixed weights for an aggregation approach. We also discussed how optimal scoring requires the solution of a linear program for each individual. We compared the results obtained by OSEA with those obtained by using the fuzzy controlled approach presented in Chapter 3 on the Flow Line Design problem. Both methods are capable of generating high quality solutions in terms of the dominated space metric proposed by Zitzler and Thiele [150], and both are capable of generating an even sample of the Pareto front. However, the fuzzy controlled method is remarkably more efficient, requiring far fewer simulation runs than the evolutionary method to obtain the same level of solution quality.

In Chapter 5 we introduced a new Bézier curve–based mechanism for constructing membership functions of normal convex fuzzy sets that can represent virtually any vague concept. In particular, we showed how the Bezier curve–based mechanism can fit any given data set with a minimum level of discrepancy. We tested our approach using data originally published by Zysno [151] and compared its performance with the methods reported in Zysno [151] and Dombi [32]. The advantages of this approach include its flexibility, ease of use, computational efficiency, and suitability for a graphical interactive implementation. To date, this is perhaps the most flexible and efficient mechanism for both automatic and interactive generation of membership functions for convex fuzzy sets.

Traditional simulation optimization techniques require an enormous number of simulation runs to evaluate the system. This research shows that by incorporating knowledge, expressed in natural language, that is often available among analysts and

[1]Data Envelopment Analysis.

decision makers, the FSO and FMSO problems can be solved efficiently using the proposed fuzzy controlled method. In case there is a complete lack of knowledge of the relationships between inputs and performance of the system, OSEA might be the only alternative for solving the FMSO problem.

6.2 Recommendations for Future Research

This research has succeeded in bringing new elements into the area of simulation optimization. While the results of our experiments on the flow line design problem are generally good, there are still open avenues for future research.

1. The fuzzy–controlled approach implemented in this research used a controller that was built off–line. Research may be conducted on the development of an efficient and automatic mechanism for knowledge extraction. We envision a system that is continuously learning and adapting along with the execution of new simulation experiments. This problem is usually referred to as the *fuzzy system identification problem*. It is conceivable that by combining neural networks, fuzzy logic, and evolutionary algorithms, a system that will be continuously learning and adapting while working to fulfill the system's goals, could be developed.

2. OSEA has been specifically designed for solving the FMSO problem. However, it is possible to generalize it to solve traditional formulations of multicriteria optimization problems. Future research should be conducted in this generalization. A thorough comparison with existing evolutionary algorithms for multicriteria optimization (e.g., VEGA, NPGA, NSGA, SPEA, MOGA, and HLGA) could be another interesting research topic.

3. The empirical results obtained so far with OSEA suggest that there is an underlying relation between the optimal fitness scores and the distance to the Pareto front. The development of theoretical results along this line of research could have significant computational effects.

4. In OSEA, it is possible to add additional constraints to the linear program behind the optimal scoring for fitness assignment. These constraints may be used

to model complex relations among the criteria. Future research may consider exploring the meaning and computational effects of these additional constraints.

5. The methods proposed in Chapters 3 and 4 evaluated a system by using average performance measures. A possible research topic is to modify the proposed methods based on the variance of the performance measures.

6. To alleviate the problem of having to make an enormous number of simulation runs to evaluate a system, it is worthwhile to investigate the possibility of implementing *neuro-accelerators* for fast evaluation of simulation models. If successful, the results of this investigation could be integrated into the methods proposed in Chapters 3 and 4.

7. This research used the flow line design problem to illustrate the proposed methods. It is important to expand the experiments to larger and more complex systems, like those arising in the context of supply chain optimization.

8. Another avenue of research lies in the Bézier curve–based mechanism for constructing membership functions as proposed in Chapter 5. A thorough analysis of the structure of the nonlinear program which must be solved within the automated curve–fitting process may result in a more efficient algorithm. A tailored interior point algorithm [37] that exploits the structure of this nonlinear program is currently under investigation. Advances made in this algorithm may be extended to the research on using Bézier distributions to model simulation input processes [136, 137].

Bibliography

[1] Robin Allenson. Genetic Algorithms with Gender for Multi-function Optimisation. Technical Report EPCC-SS92-01, Edinburgh Parallel Computing Centre, Edinburgh, Scotland, 1992.

[2] M. H. Alrefaei and S. Andradóttir. A simulated annealing algorithm with constant temperature for discrete stochastic optimization. *Management Science*, 45(5):748–764, 1999.

[3] S. Andradóttir. *Handbook of Simulation*, chapter Simulation optimization, pages 307–333. John Wiley & Sons, 1998. Edited by Jerry Banks.

[4] S. Andradóttir. A review of simulation optimization techniques. In *Proceedings of the 1998 Winter Simulation Conference*, pages 151–158, 1998.

[5] F. Azadivar. A tutorial on simulation optimization. In *Proceedings of the 1992 Winter Simulation Conference*, pages 198–204, 1992.

[6] F. Azadivar and Y. H. Lee. Optimization of discrete variable stochastic systems by computer simulation. *Mathematics and Computers in Simulation*, 30:331–345, 1988.

[7] J. E. Baker. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms*, pages 14–21, 1987.

[8] R. R. Barton and J. S. Ivey. Modifications of the Nelder-Mead simplex method for stochastic simulation response optimization. In *Proceedings of the 1991 Winter Simulation Conference*, pages 945–953, 1991.

[9] R. R. Barton and J. S. Ivey Jr. Nelder-Mead simplex modifications for simlation optimization. *Management Science*, 42(7):954–973, 1996.

[10] B. Bhanu and S. Lee. *Genetic Learning for Adaptive Image Segmentation*. Kluwer, Boston, MA, 1994.

[11] W. E. Biles and J. J. Swain. Strategies for optimization of multiple response simulation models. In *Proceedings of the 1977 Winter Simulation Conference*, pages 134–142, 1977.

[12] W. E. Biles and J. J. Swain. Mathematical programming and the optimization of computer simulations. *Mathematical Programming Study*, 11:189–207, 1979.

[13] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, New York, 1995.

[14] J. Boesel and B. L. Nelson. Accounting for randomness in heuristic simulation optimization. Technical report, Northwestern University, Department of Industrial Engineering and Management Sciences, Evanston, Illinois, 1998.

[15] J. Boesel, B. L. Nelson, and Nobuaki Ishii. A framework for simulation-optimization software. Technical report, Northwestern University, Department of Industrial Engineering and Management Sciences, Evanston, Illinois, 1999.

[16] G. E. P Box and Norman R. Draper. *Empirical Model-Building and Response Surfaces*. Wiley, New York, 1987.

[17] M. J. Box. A new method for constrained optimization and comparison with other methods. *Computer Journal*, 8:42–52, 1965.

[18] John A. Buzacott and George Shanthikumar. *Stochastic Models of Manufacturing Systems*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1993.

[19] Yolanda Carson and Anu Maria. Simulation optimization: Methods and applications. In *Proceedings of the 1997 Winter Simulation Conference*, pages 118–126, 1997. Edited by S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson.

[20] J. L. Chameau and J. C. Santamarina. Membership functions I: Comparing methods of measurement. *International Journal of Approximate Reasoning*, 1:287–301, 1987.

[21] A. Charnes and W. W. Cooper. *Management Models and Industrial Applications of Linear Programming*, volume 1. John Wiley, New York, 1961.

[22] A. Charnes, Cooper W. W., and E. Rhodes. Measuring the efficiency of decision making units. *European Journal of Operations Research*, 2:429–444, 1978.

[23] Joseph E. Chen and Kevin N. Otto. Constructing membership functions using interpolation and measurement theory. *Fuzzy Sets and Systems*, 73:313–327, 1995.

[24] W. Cheney and D. Kincaid. *Numerical Mathematics and Computing*. Brooks-Cole Publishing Co., Monterey, CA, 1980.

[25] E. R. Clayton, W. E. Weber, and B. W. Taylor III. A goal-programming approach to the optimization of computer simulation models. *IIE Transactions*, 14(4):282–287, 1982.

[26] Carlos A. Coello and Alan D. Christiansen. Two New GA-based methods for multiobjective optimization. *Civil Engineering Systems*, 15(3):207–243, 1998.

[27] Carlos A. Coello and Alan D. Christiansen. MOSES : A Multiobjective Optimization Tool for Engineering Design. *Engineering Optimization*, 31(3):337–368, 1999.

[28] Carlos A. Coello, Alan D. Christiansen, and Arturo Hernández. Using a New GA-Based Multiobjective Optimization Technique for the Design of Robot Arms. *Robotica*, 16(4):401–414, July–August 1998.

[29] Carlos A. Coello Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems*, 1(3):269–308, 1999.

[30] William W. Cooper, Lawrence M. Seiford, and Kaoru Tone. *Data Envelopment Analysis : A Comprehensive Text with Models, Applications, References and DEA-Solver Software*. Kluwer Academic Publishers, 1999.

[31] Indraneel Daas and John Dennis. A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems. *Structural Optimization*, 14(1):63–69, 1997.

[32] J. Dombi. Membership function as an evaluation. *Fuzzy Sets and Systems*, 35:1–21, 1990.

[33] D. Dubois and H. Prade. Soft computing, fuzzy logic, and artificial intelligence. *Soft Computing*, 2(1):7–11, 1998.

[34] E. J. Dudewicz and S. R. Dalal. Allocation of measurements in ranking and selection with unequal variances. *Sankhya*, B37:28–78, 1975.

[35] Matthias Ehrgott and Xavier Gandibleux. An annotated bibliography of multiobjective combinatorial optimization. Technical Report 62/2000, Universitat Kaiserlautern. Fachbereich Mathematik., 2000.

[36] G. W. Evans, B. Stuckman, and M. Mollaghasemi. Multicriteria optimization of simulation models. In *Proceedings of the 1991 Winter Simulation Conference*, pages 894–900, 1991.

[37] S.-C. Fang and Sarat Puthenpura. *Linear Programming and Extensions*. Prentice-Hall, Englewood Cliffs, NJ, 1993.

[38] Gerald E. Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*. Academic Press, 4th edition, 1997.

[39] W. Farrell. Literature review and bibliography of simulation optimization. In *Proceedings of the 1977 Winter Simulation Conference*, pages 117–124, 1977.

[40] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. John Wiley, New York, 1966.

[41] Carlos M. Fonseca and Peter J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California, 1993. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.

[42] Carlos M. Fonseca and Peter J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, 1995.

[43] Carlos M. Fonseca and Peter J. Fleming. Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms—Part I: A Unified Formulation. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 28(1):26–37, 1998.

[44] M. P. Fourman. Compaction of Symbolic Layout using Genetic Algorithms. In *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 141–153. Lawrence Erlbaum, 1985.

[45] Michael C. Fu. Optimization via simulation: a review. *Annals of Operations Research*, 53:199–247, 1994. Simulation and modeling.

[46] S. B. Gelfand and S. K. Mitter. Simulated annealing with noisy or imprecise energy measurements. *Journal on Optimization Theory and Application*, 62:49–62, 1989.

[47] M. Gen, K. Ida, and Y. Li. Solving bicriteria solid transportation problem with fuzzy numbers by genetic algorithm. *International Journal of Computers and Industrial Engineering*, 29:537–543, 1995.

[48] Mitsuo Gen and Runwei Cheng. *Genetic Algorithms & Engineering Design*. John Wiley & Sons, 1997.

[49] Mitsuo Gen and Runwei Cheng. *Genetic Algorithms & Engineering Optimization*. John Wiley & Sons, 2000.

[50] F. Glover. *Handbook of Operations Research: Foundations and Fundamentals*, chapter Integer Programming and Combinatorics, pages 120–146. Van Nostrand Reinhold Company, 1978. Edited by J. J. Moder and S. E. Elmaghraby.

[51] F. Glover, J. P. Kelly, and M. Laguna. New advances and applications of combining simulation and optimization. In *Proceedings of the 1996 Winter Simulation Conference*, pages 144–152, 1996.

[52] P. W. Glynn. Optimization of stochastic systems via simulation. In *Proceedings of the 1989 Winter Simulation Conference*, pages 90–105, 1989.

[53] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49, 1987.

[54] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

[55] David Goldsman and Barry L. Nelson. *Handbook on simulation*, chapter Comparing systems via Simulation, pages 273–306. Wiley, 1998. Edited by Jerry Banks.

[56] MacDougall M. H. *Simulating Computer Systems - Techniques and Tools*. M.I.T. Press, 1989.

[57] J. Haddock and J. Mittenthal. Simulation optimization using simulated annealing. *Computers and Industrial Engineering*, 22:387–395, 1992.

[58] P. Hajela and C. Y. Lin. Genetic search strategies in multicriterion optimal design. *Structural Optimization*, 4:99–107, 1992.

[59] J. D. Hall, R. O. Bowden, and J. M. Usher. Using evolution strategies and simulation to optimize a pull production system. *Journal of Materials Processing Technology*, 61:47–52, 1996.

[60] K. Healy and L. W. Schruben. Retrospective simulation response optimization. In B. L. Nelson, W. D. Kelton, and G. M. Clark, editors, *Proceedings of the 1991 Winter Simulation Conference*, pages 901–906, Piscataway, New Jersey, 1991. IEEE.

[61] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

[62] R. Hooke and T. A. Jeeves. A direct search solution of numerical and statistical problems. *Journal of the ACM*, 8:212–229, 1961.

[63] Jeffrey Horn. F1.9 multicriterion decision making. In *Handbook of Evolutionary Computation*, Bristol, U.K., 1997. IOP Publishing Ltd and Oxford University Press. T. Bäck, D. B. Fogel, and Z. Michalewicz.

[64] Jeffrey Horn and Nicholas Nafpliotis. Multiobjective Optimization using the Niched Pareto Genetic Algorithm. Technical Report IlliGAl Report 93005, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 1993.

[65] Jeffrey Horn, Nicholas Nafpliotis, and David 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.

[66] D. G. Humphrey and J. R. Wilson. A revised simplex search procedure for stochastic simulation response surface optimization. *INFORMS Journal on Computing*, 12(4), 2000.

[67] SAS Institute Inc. *SAS Procedures Guide, Version 6*. SAS Institute Inc., Cary, NC, 3rd edition, 1990.

[68] SAS Institute Inc. *SAS/STAT User's Guide, Version 6*. SAS Institute Inc., Cary, NC, 4th edition, 2000.

[69] Hisao Ishibuchi and Tadahiko Murata. Multi-Objective Genetic Local Search Algorithm. In Toshio Fukuda and Takeshi Furuhashi, editors, *Proceedings of the 1996 International Conference on Evolutionary Computation*, pages 119–124, Nagoya, Japan, 1996. IEEE.

[70] Sheldon H. Jacobson and Lee W. Schruben. Techniques for simulation response optimization. *Operations Research Letters*, 8(1):1–9, 1989.

[71] W. Jakob, M. Gorges-Schleuter, and C. Blume. Application of Genetic Algorithms to task planning and learning. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2nd Workshop*, Lecture Notes in Computer Science, pages 291–300, Amsterdam, 1992. North-Holland Publishing Company.

[72] J.R. Jang. ANFIS: Adaptive-network-based-fuzzy-inference system. *IEEE Transactions on Systems, Man and Cybernetics*, 23(3):665–685, 1993.

[73] Jyh-Shing Roger Jang, Chuen-Tsai Sun, and Elji Mizutani. *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Prentice-Hall, Upper Saddle River, New Jersey, 1997.

[74] Gareth Jones, Robert D. Brown, David E. Clark, Peter Willett, and Robert C. Glen. Searching Databases of Two-Dimensional and Three-Dimensional Chemical Structures using Genetic Algorithms. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 597–602, San Mateo, California, 1993. Morgan Kaufmann.

[75] J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *Annals of Mathematical Statistics*, 23:462–466, 1952.

[76] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vechi. Optimization by simulated annealing. *Science*, 221:671–680, 1983.

[77] J. P. C. Kleijnen. *Handbook on simulation*, chapter Experimental design for sensitivity analysis, optimization, and validation of simulation models, pages 173–223. Wiley, 1998. Edited by Jerry Banks.

[78] George J. Klir and Bo Yuan. *Fuzzy sets and fuzzy logic: theory and applications*. Prentice-Hall Inc, Upper Saddle River, NJ, 1995.

[79] Frank Kursawe. A variant of evolution strategies for vector optimization. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop, PPSN I*, volume 496 of *Lecture Notes in Computer Science*, pages 193–197, Berlin, Germany, oct 1991. Springer-Verlag.

[80] Stephen S. Lavenberg, editor. *Computer Performance Modeling Handbook*, chapter The Statistical Analysis of Simulation Results, pages 267–329. Academic Press, New York, New York, 1983.

[81] Averill M. Law and W. David Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, Singapore, 2nd edition, 1991.

[82] P. L'Ecuyer. An overview of derivative estimation. In B. L. Nelson, W. D. Kelton, and G. M. Clark, editors, *Proceedings of the 1991 Winter Simulation Conference*, pages 207–217, Piscataway, NJ, 1991. IEEE.

[83] C. T. Lin and C. S. George Lee. *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*. Prentice-Hall, Upper Saddle River, New Jersey, 1996.

[84] Joanna Lis and A. E. Eiben. A Multi-Sexual Genetic Algorithm for Multiobjective Optimization. In Toshio Fukuda and Takeshi Furuhashi, editors, *Proceedings of the 1996 International Conference on Evolutionary Computation*, pages 59–64, Nagoya, Japan, 1996. IEEE.

[85] Xiaojian Liu, D. W. Begg, and R. J. Fishwick. Genetic approach to optimal topology/controller design of adaptive structures. *International Journal for Numerical Methods in Engineering*, 41:815–830, 1998.

[86] Daniel H. Loughlin and S. Ranjithan. The Neighborhood constraint method: A Genetic Algorithm-Based Multiobjective Optimization Technique. In Thomas Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 666–673, San Mateo, California, July 1997. Michigan State University, Morgan Kaufmann Publishers.

[87] E. H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7(1):1–13, 1975.

[88] A. L. Medaglia and S.-C. Fang. A genetic-based framework for solving (multicriteria) weighted matching problems. Technical report, North Carolina State University, Raleigh, North Carolina, 2000. Submitted to the European Journal of Operational Research.

[89] S. Medasani, J. Kim, and R. Krishnapuram. An overview of membership function generation techniques for pattern recognition. *International Journal of Approximate Reasoning*, 19:391–417, 1998.

[90] M. Meketon. Optimization in simulation: A survey of recent results. In *Proceedings of the 1987 Winter Simulation Conference*, pages 58–67, 1987.

[91] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 3rd edition, 1996.

[92] D. C. Montgomery and V. M. Bettencourt Jr. Multiple response surface methods in computer simulation. *Simulation*, 29:113–121, 1977.

[93] R. H. Myers and D. C. Montgomery. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. Wiley, New York, 1995.

[94] J. Nash. The bargaining problem. *Econometrica*, 18:155–162, 1950.

[95] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[96] V. Norkin, Y. Ermoliev, and A. Ruszczynski. On optimal allocation of indivisibles under uncertainty. *Operations Research*, 46(3):381–395, 1998.

[97] A. M. Norwich and I. B. Turksen. A model for the measurement of membership and the consequences of its empirical implementation. *Fuzzy Sets and Systems*, 12:1–25, 1984.

[98] A. Nozari and J. S. Morris. Application of an optimization procedure to steady-state simulation. In *Proceedings of the 1984 Winter Simulation Conference*, pages 217–219, 1984.

[99] Andrzej Osyczka and Sourav Kundu. A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. *Structural Optimization*, 10:94–99, 1995.

[100] Jacques Périaux, Mourad Sefrioui, and Bertrand Mantel. GA Multiple Objective Optimization Strategies for Electromagnetic Backscattering. In D. Quagliarella, J. Périaux, C. Poloni, and G. Winter, editors, *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science. Recent Advances and Industrial Applications*, chapter 11, pages 225–243. John Wiley and Sons, West Sussex, England, 1997.

[101] Harry G. Perros. *Queueing Networks with Blocking*. Oxford University Press, New York, New York, 1994.

[102] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C - The Art of Scientific Computing*. Cambridge University Press, 2nd edition, 1992.

[103] A. Alan B. Pritsker, Jean J. O'Reilly, and David K. La Val. *Simulation with Visual SLAM and AweSim*. Wiley, 1997.

[104] Domenico Quagliarella and Alessandro Vicini. Coupling Genetic Algorithms and Gradient Based Optimization Techniques. In D. Quagliarella, J. Périaux, C. Poloni, and G. Winter, editors, *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science. Recent Advances and Industrial Applications*, chapter 14, pages 289–309. John Wiley and Sons, West Sussex, England, 1997.

[105] I. Rechenberg. *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog Verlag, Stuttgart, Germany, 1973.

[106] L. P. Rees, E. R. Clayton, and B. W. Taylor III. Solving multiple response simulation models using modified response surface methodology within a lexicographic goal programming framework. *IIE Transactions*, 17(1):47–57, 1985.

[107] J. T. Richardson, M. R. Palmer, G. Liepins, and M. Hilliard. Some guidelines for genetic algorithms with penalty functions. In J. D. Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 191–197. Morgan Kaufmann, San Mateo, CA, 1989.

[108] Brian J. Ritzel, J. Wayland Eheart, and S. Ranjithan. Using genetic algorithms to solve a multiple objective groundwater pollution containment problem. *Water Resources Research*, 30(5):1589–1603, may 1994.

[109] H. Robbins and S. Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[110] Matthew N. O. Sadiku. *Simulation of Local Area Networks*. CRC Press, Boca Raton, 1995.

[111] M. H. Safizadeh. Optimization in simulation - current issues and the future outlook. *Naval Research Logistics*, 37(6):807–825, 1990.

[112] Eric Sandgren. Multicriteria design optimization by goal programming. In Hojjat Adeli, editor, *Advances in Design Optimization*, chapter 23, pages 225–265. Chapman & Hall, London, 1994.

[113] C. Sauer. *Simulation of Computer Communication Systems*. Prentice-Hall, 1983.

[114] J. David Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 93–100, 1985.

[115] J. David Schaffer and John J. Grefenstette. Multiobjective Learning via Genetic Algorithms. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI-85)*, pages 593–595, Los Angeles, California, 1985. AAAI.

[116] J. W. Schmidt and R. E. Taylor. *Simulation and Analysis of Industrial Systems*. Richard D. Irwin, Homewood, IL, 1970.

[117] L. W. Schruben and V. J. Cogliano. An experimental procedure for simulation response surface model identification. *Communications of the Association for Computing Machinery*, 30:716–730, 1987.

[118] H. P. Schwefel. *Numerical optimization of computer models*. Wiley, Chichester, 1981.

[119] A. Shapiro. Simulation based optimization. In *Proceedings of the 1996 Winter Simulation Conference*, pages 332–336, 1996.

[120] N. Srinivas and Kalyanmoy Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, fall 1994.

[121] Timothy J. Stanley and Trevor Mudge. A Parallel Genetic Algorithm for Multiobjective Microprocessor Design. In Larry J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 597–604, San Mateo, California, July 1995. University of Pittsburgh, Morgan Kaufmann Publishers.

[122] Ralph E. Steuer. *Multiple Criteria Optimization*. Series in Probability and Mathematical Statistics. John Wiley & Sons, Inc., 1985.

[123] D. W. Sullivan and J. R. Wilson. Restricted subset selection procedures for simulation. *Operations Research*, 37:52–71, 1989.

[124] R. Suri. An overview of evaluative models for flexible manufacturing systems. In *Proceedings of the First ORSA/TIMS Conference on Flexible Manufacturing Systems*, pages 8–15, 1984. Edited by K. E. Stecke and R. Suri.

[125] Patrick D. Surry and Nicholas J. Radcliffe. The COMOGA Method: Constrained Optimisation by Multiobjective Genetic Algorithms. *Control and Cybernetics*, 26(3), 1997.

[126] Patrick D. Surry, Nicholas J. Radcliffe, and Ian D. Boyd. A Multi-Objective Approach to Constrained Optimisation of Gas Supply Networks : The COMOGA Method. In Terence C. Fogarty, editor, *Evolutionary Computing. AISB Workshop. Selected Papers*, Lecture Notes in Computer Science, pages 166–180, Sheffield, U.K., 1995. Springer-Verlag.

[127] Gilbert Syswerda and Jeff Palmucci. The Application of Genetic Algorithms to Resource Scheduling. In Richard K. Belew and Lashon B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 502–508, San Mateo, California, 1991. Morgan Kaufmann.

[128] T. Takagi and M. Sugeno. Fuzzy identification of systems and application to modeling and control. *Transactions on Systems, Man and Cybernetics*, 15:116–132, 1985.

[129] H. Tamaki, H. Kita, and S. Kobayashi. Multi-objective optimization by genetic algorithms: A review. In *Proceedings of the 1996 IEEE ICEC*, pages 517–522, 1996.

[130] Hisashi Tamaki, M. Mori, M. Araki, Y. Mishima, and H. Ogai. Multi-Criteria Optimization by Genetic Algorithms : A Case of Scheduling in Hot Rolling Process. In *Proceedings of the 3rd Conference of the Association of Asian-Pacific Operational Research Societies within IFORS (APORS'94)*, pages 374–381. World Scientific, 1995.

[131] G. Tompkins and F. Azadivar. Genetic algorithms in optimizing simulated systems. In *Proceedings of the 1995 Winter Simulation Conference*, pages 757–762, 1995.

[132] I. B. Turksen. Measurement of membership functions and their acquisition. *Fuzzy Sets and Systems*, 40:5–38, 1991.

[133] Manuel Valenzuela and Eduardo Uresti. A Non-Generational Genetic Algorithm for Multiobjective Optimization. In Thomas Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 658–665, San Mateo, California, July 1997. Michigan State University, Morgan Kaufmann Publishers.

[134] David A. Van Veldhuizen and Gary B. Lamont. Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolutionary Computation*, 8(2):1–26, 2000.

[135] V. Rao Vemuri and Walter Cede no. A New Genetic Algorithm for Multi Objective Optimization in Water Resource Management. In *Proceedings of the Second IEEE International Conference on Evolutionary Computation*, pages 495–500, Piscataway, New Jersey, 1995. IEEE Press.

[136] Mary Ann F. Wagner and James R. Wilson. Recent developments in input modeling with Bézier distributions. In *Proceedings of the 1996 Winter Simulation Conference*, pages 1448–1456, 1996.

[137] Mary Ann F. Wagner and James R. Wilson. Using univariate Bézier distributions to model simulation input processes. *IIE Transactions*, 28:699–711, 1996.

[138] D. R. Wallace, M. J. Jakiela, and W. C. Flowers. Design search under probabilistic specifications using genetic algorithms. *Computer-Aided Design*, 28:405–420, 1994.

[139] P. B. Wienke, C. Lucasius, and G. Kateman. Multicriteria target optimization of analytical procedures using a genetic algorithm. *Analytical Chimica Acta*, 265(2):211–225, 1992.

[140] J. R. Wilson. Future directions in response surface methodology for simulation. In *Proceedings of the 1987 Winter Simulation Conference*, pages 378–381, 1987.

[141] P. B. Wilson and M. D. Macleod. Low implementation cost IIR digital filter design using genetic algorithms. In *IEE/IEEE Workshop on Natural Algorithms in Signal Processing*, pages 4/1–4/8, Chelmsford, U.K., 1993.

[142] Xiaofeng Yang and Mitsuo Gen. Evolution program for bicriteria transportation problem. In M. Gen and T. Kobayashi, editors, *Proceedings of the 16th International Conference on Computers and Industrial Engineering*, pages 451–454, Ashikaga, Japan, 1994. Pergamon Press.

[143] James M. Yunker and Jeffrey D. Tew. Simulation optimization by genetic search. *Mathematics and Computers in Simulation*, 37(1):17–28, 1994.

[144] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.

[145] L. A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man and Cybernetics*, 3(1):28–44, 1973.

[146] Lotfi A. Zadeh. Fuzzy logic, neural networks, and soft computing. *Communications of the ACM*, 37(3):77–84, 1994.

[147] Lotfi A. Zadeh. Some reflections on soft computing, granular computing and their roles in the conception, design and utilization of information/intelligence systems. *Soft Computing*, 2(1):23–25, 1998.

[148] H. J. Zimmermann and P. Zysno. Quantifying vagueness in decision models. *European Journal of Operational Research*, 22:148–158, 1985.

[149] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.

[150] 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.

[151] P. Zysno. Modelling membership functions. In B. Rieger, editor, *Empirical Semantics*, pages 350–375, Bochum, 1981. Brockmeyer.

Appendix A

Membership Functions

A.1 State Variables

Table A.1 contains a complete list of all the fuzzy controller inputs for the flow line design application in Section 3.4. For every input (linguistic variable), the linguistic states and the number of the figure where the membership functions can be found are given.

Linguistic Variable	Linguistic Values	Membership Functions
Overall work-in-process (w)	Low	See Figure A.1
	Medium	
	High	
Work-in-process at stage 1 (w_1)	Low	See Figure A.2
	Medium	
	High	
Work-in-process at stage 2 (w_2)	Low	See Figure A.3
	Medium	
	High	
Work-in-process at stage 3 (w_3)	Low	See Figure A.4
	Medium	
	High	
Work-in-process at stage 4 (w_4)	Low	See Figure A.5
	Medium	
	High	
Time in system (T)	Short	See Figure A.6
	Medium	
	Long	
Utilization at stage 1 (ρ_1)	Low	See Figure A.7
	Medium	
	High	
Utilization at stage 2 (ρ_2)	Low	See Figure A.8
	Medium	
	High	
Utilization at stage 3 (ρ_3)	Low	See Figure A.9
	Medium	
	High	
Utilization at stage 4 (ρ_4)	Low	See Figure A.10
	Medium	
	High	

Table A.1: Flow line fuzzy controller inputs.

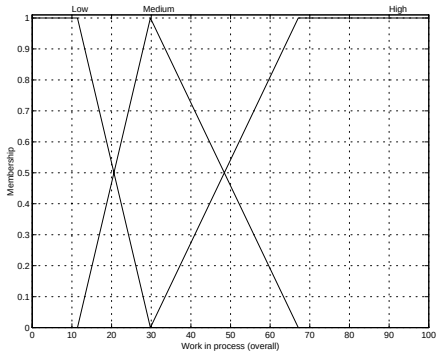


Figure A.1: Overall work-in-process (w)

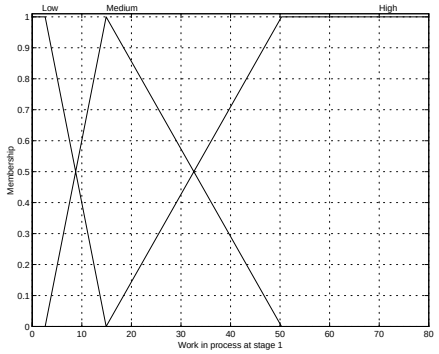


Figure A.2: Work-in-process at stage 1 (w_1)

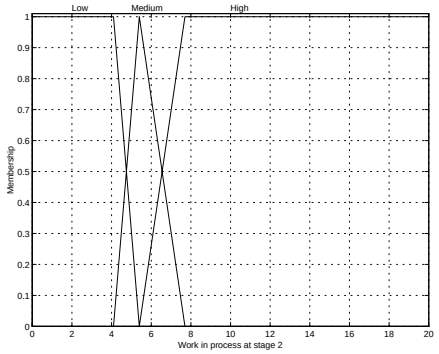


Figure A.3: Work-in-process at stage 2 (w_2)

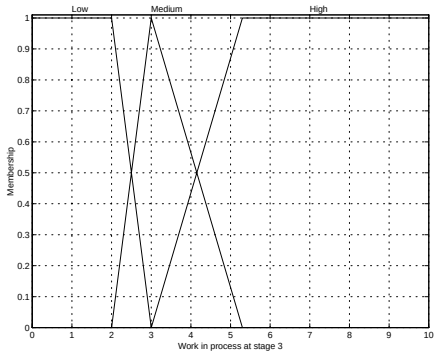


Figure A.4: Work-in-process at stage 3 (w_3)

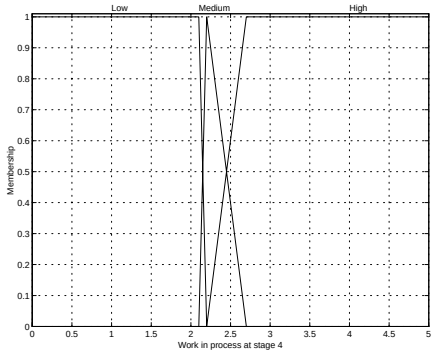


Figure A.5: Work-in-process at stage 4 (w_4)

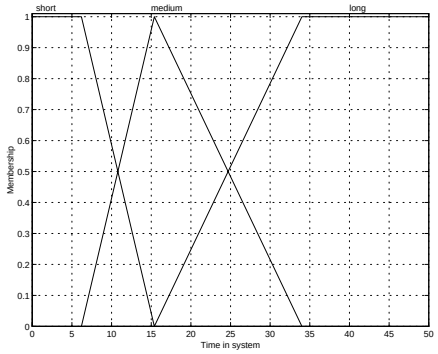


Figure A.6: Time in system (T)

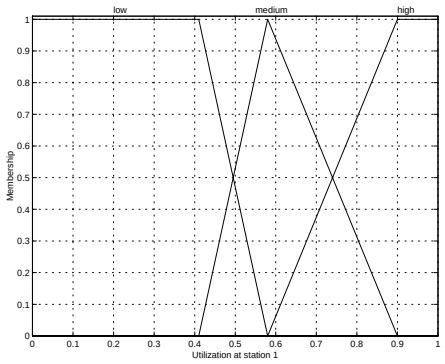


Figure A.7: Utilization at station 1 (ρ_1)

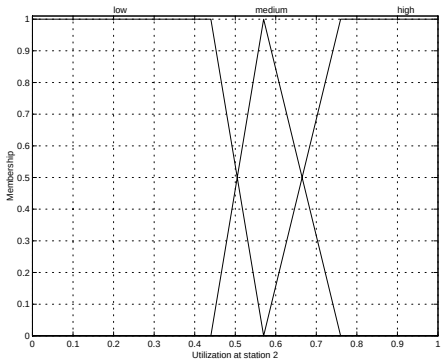


Figure A.8: Utilization at station 2 (ρ_2)

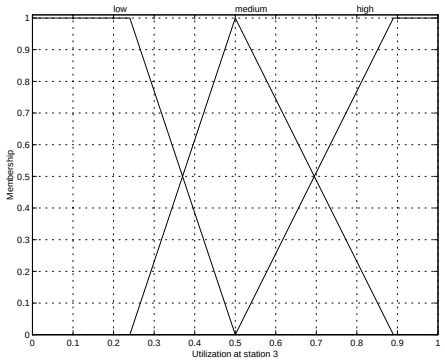


Figure A.9: Utilization at station 3 (ρ_3)

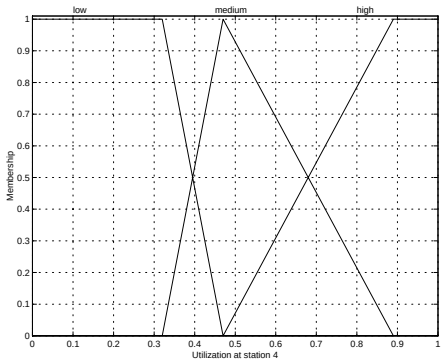


Figure A.10: Utilization at station 4 (ρ_4)

A.2 Control Variables

The fuzzy controller outputs for the flow line design application in Section 3.4 are displayed in Table A.2. For every output (linguistic variable), the associated linguistic states and the number of the figure where the membership functions can be found are given.

Linguistic Variable	Linguistic Values	Membership Functions
Change in server rate at stage 1 ($\Delta\mu_1$)	Negatively large	See Figure A.11.
	Negatively small	
	Zero	
	Positively small	
	Positively large	
Change in server rate at stage 3 ($\Delta\mu_3$)	Negatively large	See Figure A.12.
	Negatively small	
	Zero	
	Positively small	
	Positively large	
Change in number of servers at stage 2 (Δs_2)	Negatively large	See Figure A.13.
	Negatively small	
	Zero	
	Positively small	
	Positively large	
Change in number of servers at stage 3 (Δs_3)	Negatively large	See Figure A.14.
	Negatively small	
	Zero	
	Positively small	
	Positively large	

Table A.2: Flow line fuzzy controller outputs

Linguistic Variable	Linguistic Values	Membership Functions
Change in number of servers at stage 4 (Δs_4)	Negatively large	See Figure A.15.
	Negatively small	
	Zero	
	Positively small	
	Positively large	
Change in buffer space at stage 2 (Δb_2)	Negatively large	See Figure A.16.
	Negatively small	
	Zero	
	Positively small	
	Positively large	
Change in buffer space at stage 4 (Δb_4)	Negatively large	See Figure A.17.
	Negatively small	
	Zero	
	Positively small	
	Positively large	

Table A.2: Flow line fuzzy controller outputs (continued).

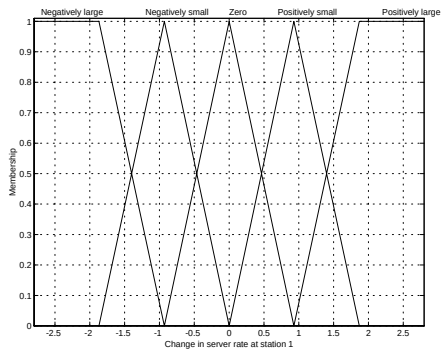


Figure A.11: Change in server rate at station 1 ($\Delta\mu_1$)

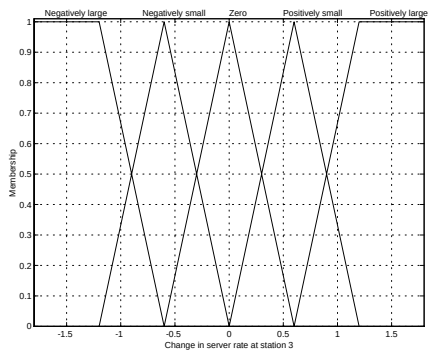


Figure A.12: Change in server rate at station 3 ($\Delta\mu_3$)

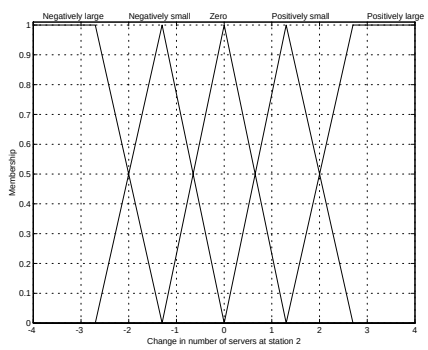


Figure A.13: Change in servers at station 2 (Δs_2)

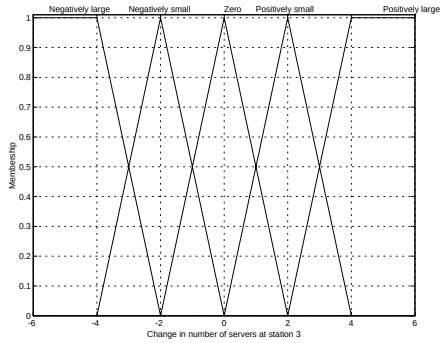


Figure A.14: Change in servers at station 3 (Δs_3)

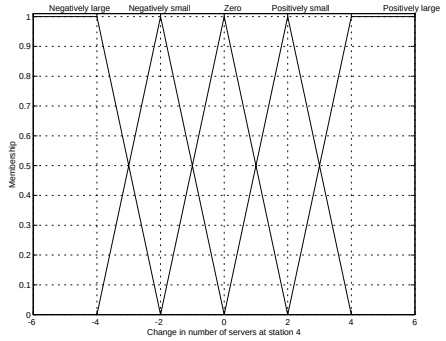


Figure A.15: Change in servers at station 4 (Δs_4)

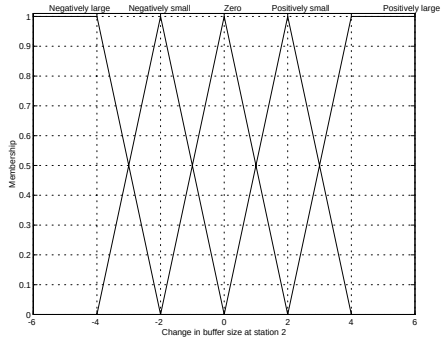


Figure A.16: Change in buffer space at station 2 (Δb_2)

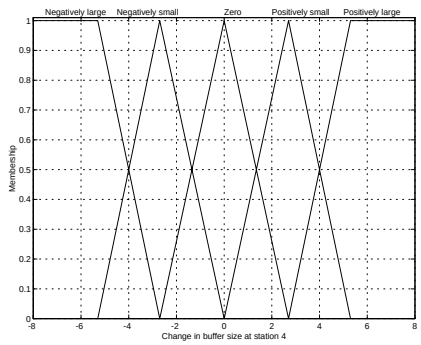


Figure A.17: Change in buffer space at station 4 (Δb_4)

Appendix B

Rule Base for the Multiple Objective Scenario

The following rule base is the one used in the experiment on the flow line design problem in Section 3.4.5.2 and Section 4.2.

Rule 1. If the overall work-in-process (ϖ) is high ($\tilde{H}_{\varpi}$) then the change in server rate at station 1 ($\Delta\gamma_1$) should be positively small ($\tilde{PS}_{\Delta\gamma_1}$).

Rule 2. If the overall work-in-process (ϖ) is medium ($\tilde{M}_{\varpi}$) then the change in server rate at station 1 ($\Delta\gamma_1$) should be positively small ($\tilde{PS}_{\Delta\gamma_1}$).

Rule 3. If the overall work-in-process (ϖ) is low ($\tilde{L}_{\varpi}$) then the change in server rate at station 1 ($\Delta\gamma_1$) should be zero ($\tilde{Z}_{\Delta\gamma_1}$).

Rule 4. If the overall work-in-process (ϖ) is high ($\tilde{H}_{\varpi}$) then the change in the number of servers at station 4 (Δs_4) should be positively small ($\tilde{PS}_{\Delta s_4}$).

Rule 5. If the overall work-in-process (ϖ) is medium ($\tilde{M}_{\varpi}$) then the change in the number of servers at station 4 (Δs_4) should be positively small ($\tilde{PS}_{\Delta s_4}$).

Rule 6. If the overall work-in-process (ϖ) is low ($\tilde{L}_{\varpi}$) then the change in the number of servers at station 4 (Δs_4) should be zero ($\tilde{Z}_{\Delta s_4}$).

Rule 7. If the overall work-in-process (ϖ) is high ($\tilde{H}_{\varpi}$) and the utilization at station 3 (φ_3) is high ($\tilde{H}_{\varphi_3}$) then the change in the server rate at station 3 ($\Delta\gamma_3$) should be positively large ($\tilde{PL}_{\Delta\gamma_3}$).

Rule 8. If the overall work-in-process (ϖ) is medium ($\tilde{M}_{\varpi}$) and the utilization at station 3 (φ_3) is high ($\tilde{H}_{\varphi_3}$) then the change in the server rate at station 3 ($\Delta\gamma_3$) should be positively small ($\tilde{PS}_{\Delta\gamma_3}$).

Rule 9. If the overall work-in-process (ϖ) is medium ($\tilde{M}_{\varpi}$) and the utilization at station 3 (φ_3) is medium ($\tilde{M}_{\varphi_3}$) then the change in the server rate at station 3 ($\Delta\gamma_3$) should be positively small ($\tilde{PS}_{\Delta\gamma_3}$).

Rule 10. If the overall work-in-process (ϖ) is high ($\tilde{H}_{\varpi}$) and the utilization at station 2 (φ_2) is high ($\tilde{H}_{\varphi_2}$) then the change in the number of servers at station 2 (Δs_2) should be positively large ($\tilde{PL}_{\Delta s_2}$).

Rule 11. If the overall work-in-process (ϖ) is medium ($\tilde{M}_{\varpi}$) and the utilization at station 2 (φ_2) is high ($\tilde{H}_{\varphi_2}$) then the change in the number of servers at station 2 (Δs_2) should be positively small ($\tilde{PS}_{\Delta s_2}$).

Rule 12. If the overall work-in-process (ϖ) is medium ($\tilde{M}_{\varpi}$) and the utilization at station 2 (φ_2) is medium ($\tilde{M}_{\varphi_2}$) then the change in the number of servers at station 2 (Δs_2) should be positively small ($\tilde{PS}_{\Delta s_2}$).

Rule 13. If the overall work-in-process (ϖ) is high ($\tilde{H}_{\varpi}$) and the utilization at station 3 (φ_3) is high ($\tilde{H}_{\varphi_2}$) then the change in the number of servers at station 3 (Δs_3) should be positively large ($\tilde{PL}_{\Delta s_3}$).

Rule 14. If the overall work-in-process (ϖ) is high ($\tilde{H}_{\varpi}$) and the utilization at station 3 (φ_3) is medium ($\tilde{M}_{\varphi_2}$) then the change in the number of servers at station 3 (Δs_3) should be positively small ($\tilde{PS}_{\Delta s_3}$).

Rule 15. If the overall work-in-process (ϖ) is medium ($\tilde{M}_{\varpi}$) and the utilization at station 3 (φ_3) is medium ($\tilde{M}_{\varphi_2}$) then the change in the number of servers at station 3 (Δs_3) should be positively small ($\tilde{PS}_{\Delta s_3}$).

Rule 16. If the utilization at station 1 (φ_1) is high ($\tilde{H}_{\varphi_1}$) then the change in server rate at station 1 ($\Delta\gamma_1$) should be zero ($\tilde{Z}_{\Delta\gamma_1}$).

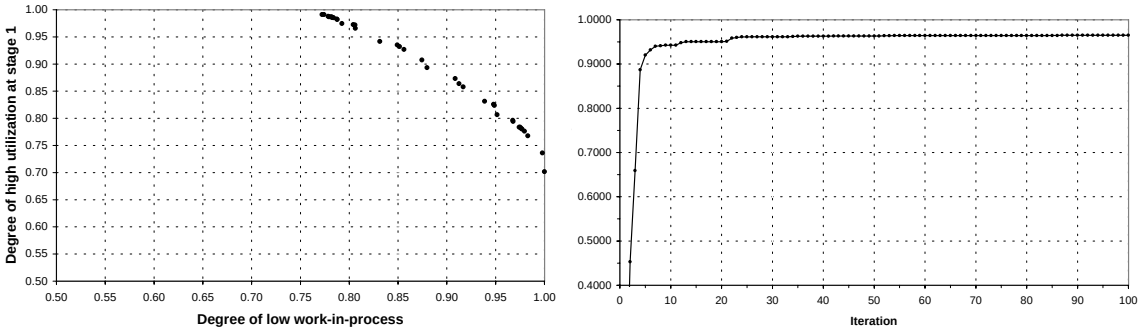
Rule 17. If the utilization at station 1 (φ_1) is medium ($\tilde{M}_{\varphi_1}$) then the change in server rate at station 1 ($\Delta\gamma_1$) should be negatively small ($\tilde{NS}_{\Delta\gamma_1}$).

Rule 18. If the utilization at station 1 (φ_1) is low ($\tilde{L}_{\varphi_1}$) then the change in server rate at station 1 ($\Delta\gamma_1$) should be negatively large ($\tilde{NL}_{\Delta\gamma_1}$).

Appendix C

Results for the Fuzzy Controlled Approach

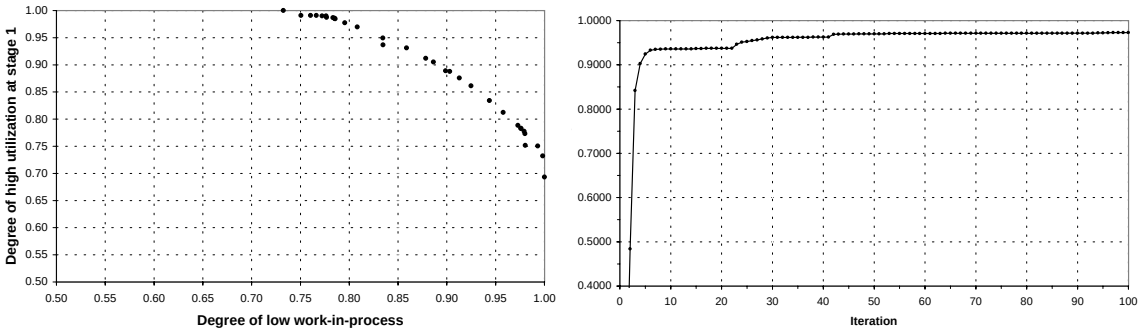
The results presented in this appendix detail the summarized results for the fuzzy controlled approach presented in Section 4.2.



(a) Approximate Pareto front.

(b) Zitzler and Thiele's dominated space metric.

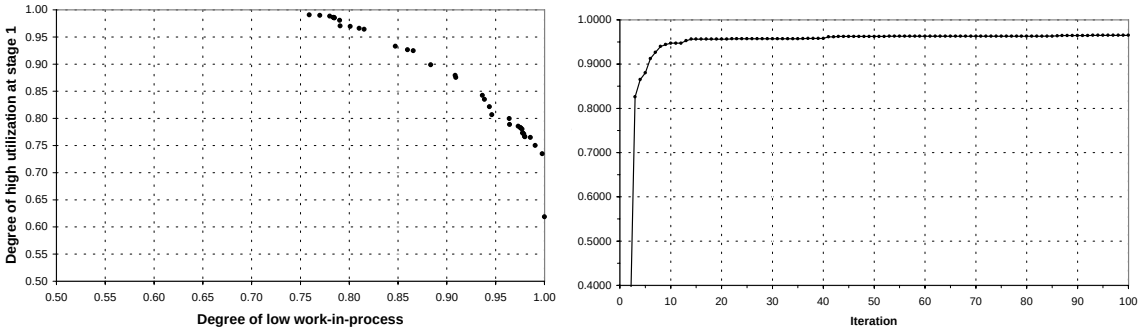
Figure C.1: Experiment # 1 of the two-goal scenario with the fuzzy controlled approach.



(a) Approximate Pareto front.

(b) Zitzler and Thiele's dominated space metric.

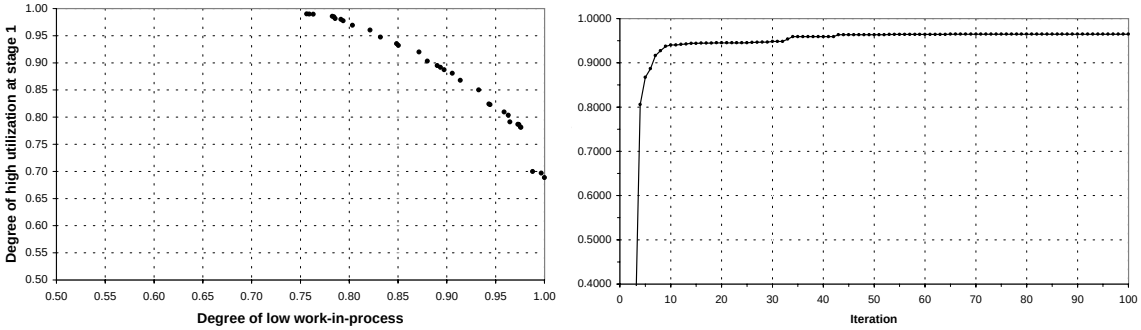
Figure C.2: Experiment # 2 of the two-goal scenario with the fuzzy controlled approach.



(a) Approximate Pareto front.

(b) Zitzler and Thiele's dominated space metric.

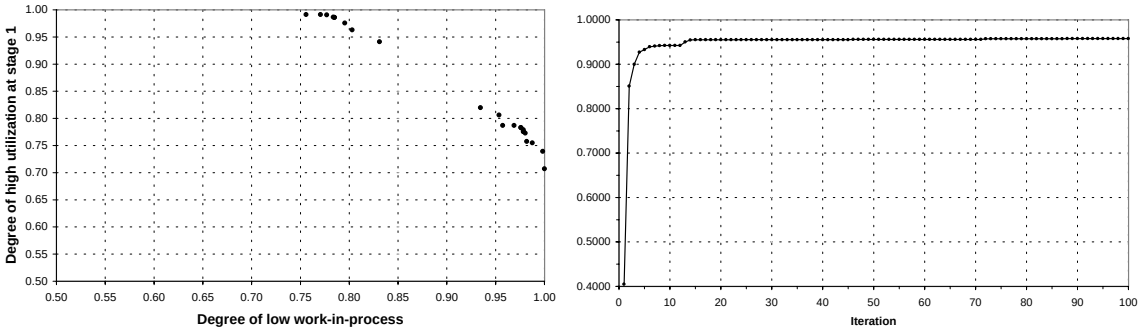
Figure C.3: Experiment # 3 of the two-goal scenario with the fuzzy controlled approach.



(a) Approximate Pareto front.

(b) Zitzler and Thiele's dominated space metric.

Figure C.4: Experiment # 4 of the two-goal scenario with the fuzzy controlled approach.



(a) Approximate Pareto front.

(b) Zitzler and Thiele's dominated space metric.

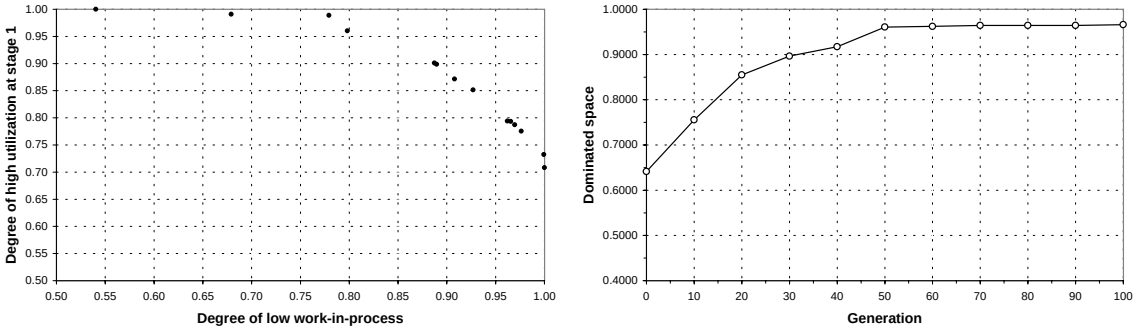
Figure C.5: Experiment # 5 of the two-goal scenario with the fuzzy controlled approach.

Appendix D

Results for OSEA

.

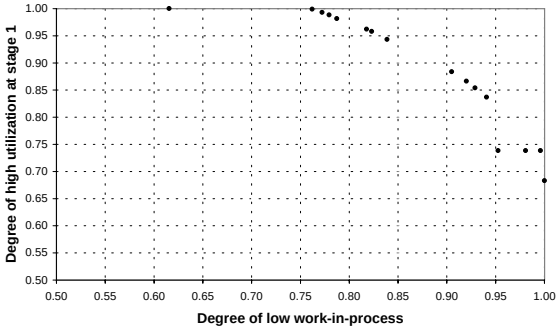
The results presented in this appendix detail the summarized results for OSEA presented in Section 4.2.



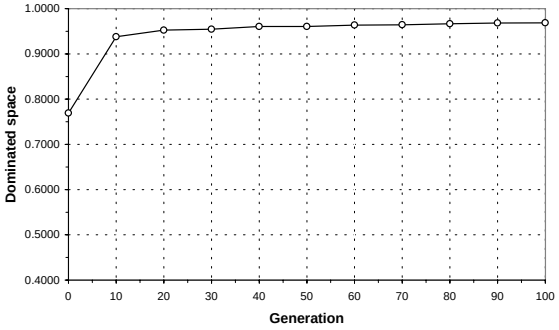
(a) Approximate Pareto front.

(b) Zitzler and Thiele's dominated space metric.

Figure D.1: Experiment # 1 of the flow line design problem with two goals solved by OSEA.

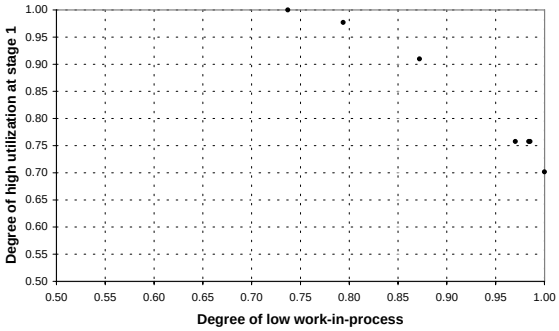


(a) Approximate Pareto front.

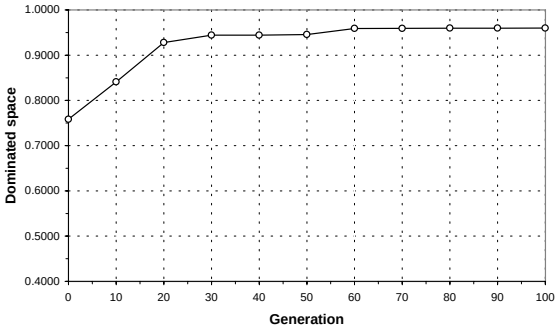


(b) Zitzler and Thiele's dominated space metric.

Figure D.2: Experiment # 2 of the flow line design problem with two goals solved by OSEA.

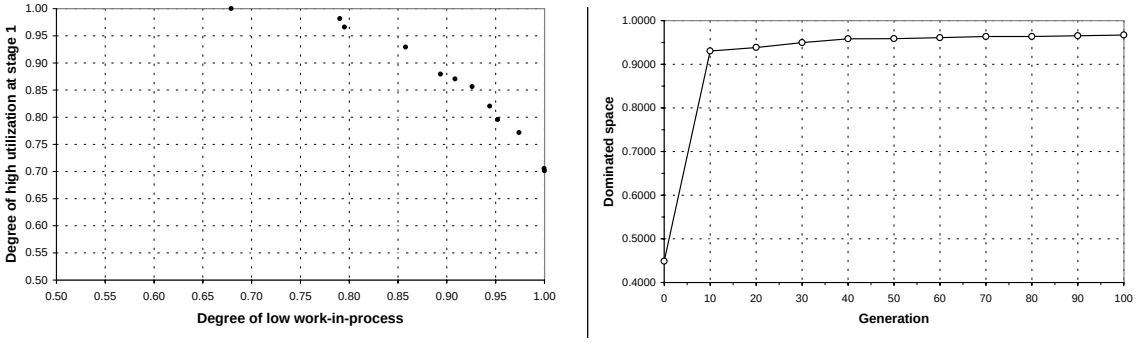


(a) Approximate Pareto front.



(b) Zitzler and Thiele's dominated space metric.

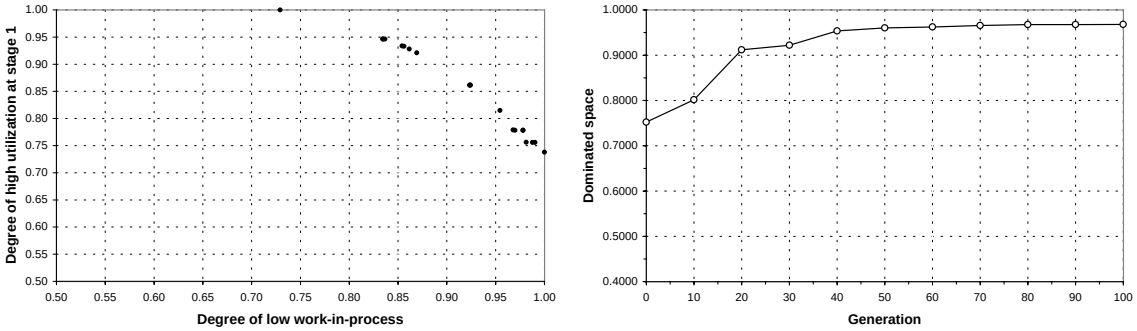
Figure D.3: Experiment # 3 of the flow line design problem with two goals solved by OSEA.



(a) Approximate Pareto front.

(b) Zitzler and Thiele's dominated space metric.

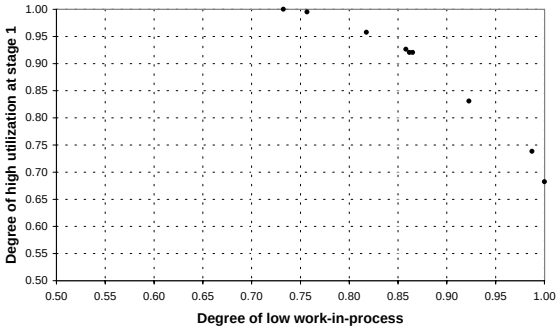
Figure D.4: Experiment # 4 of the flow line design problem with two goals solved by OSEA.



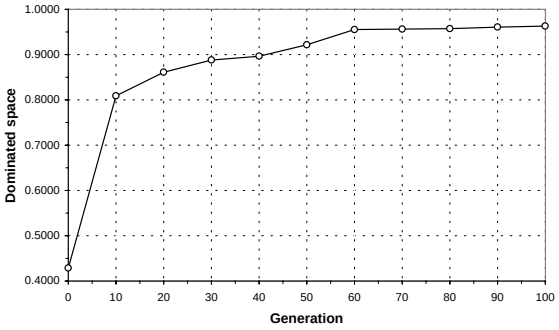
(a) Approximate Pareto front.

(b) Zitzler and Thiele's dominated space metric.

Figure D.5: Experiment # 5 of the flow line design problem with two goals solved by OSEA.



(a) Approximate Pareto front.



(b) Zitzler and Thiele's dominated space metric.

Figure D.6: Experiment # 6 of the flow line design problem with two goals solved by OSEA.