

© [2007]

HEIDI ARLENE TABOADA JIMENEZ

ALL RIGHTS RESERVED

**MULTI-OBJECTIVE OPTIMIZATION ALGORITHMS CONSIDERING
OBJECTIVE PREFERENCES AND SOLUTION CLUSTERS**

by

HEIDI ARLENE TABOADA JIMENEZ

A Dissertation submitted to the
Graduate School-New Brunswick
Rutgers, The State University of New Jersey
in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Industrial and Systems Engineering

written under the direction of

Dr. David W. Coit

and approved by

New Brunswick, New Jersey

October, 2007

ABSTRACT OF THE DISSERTATION

Multi-objective Optimization Algorithms Considering Objective Preferences and Solution Clusters

by HEIDI ARLENE TABOADA JIMENEZ

Dissertation director: Dr. David W. Coit

This thesis presents the development of new methods for the solution of multiple objective problems. One of the main contributions of this thesis is that it presents new approaches that provide a balance between the determination of single solutions and a set of Pareto-optimal solutions. Existing solution methodologies for multiple objective problems can generally be categorized as single solution methods or Pareto optimality methods. However, for many problems and decision-makers, a balanced approach is more appropriate, and this thesis provides new approaches to meet those needs. Other main contributions are that several novel multi-objective evolutionary algorithms are presented, which offer distinct advantages compared to existing algorithms.

Two different new approaches are introduced which can efficiently determine an attractive Pareto set or organize and reduce the size of the Pareto-optimal set. This makes it easier for the decision-maker to comparatively analyze a smaller set of solutions, and finally, select the most desirable one for system implementation.

In the first approach, the developed algorithm has the capability to automatically identify an optimal number of clusters in the Pareto-optimal set and provide the decision-

maker with representative solutions of each cluster. The second approach is a method that yields efficient results for any user who can prioritize the objective functions. In this method, the objective functions are ranked ordinally based on their importance to the decision-maker, and a reduced Pareto set is determined based on randomly generated weight sets, reflecting the decision-maker preferences.

Different new multiple objective evolutionary algorithms (MOEAs) are designed as the result of this research and they are described and tested. New ideas have been incorporated into these MOEAs to provide the research community with new alternatives. One of the developed MOEAs is MoPriGA, a multi-objective prioritized genetic algorithm. MoPriGA incorporates the knowledge of the decision-maker objective function preferences directly within the evolutionary algorithm. The idea behind this algorithm is to more intensely focus on the region of the Pareto set of interest to the decision-maker.

Acknowledgments

Pursuing a PhD is a life changing experience, and I cannot properly express my happiness to have gotten to this stage. This is by far the most important part of my thesis, so it is ironic that I am having difficulty expressing my thoughts. Perhaps, because I know that words will fail to express my deep appreciation that I feel for some of the people who have helped me in all kinds of ways throughout these years. However, it is with tremendous gratitude that I write these acknowledgements.

First, I would like to thank my PhD advisor, Dr. David W. Coit, for his great supervision, support, and encouragement during this work. Throughout my thesis-writing period, he not only provided encouragement, sound advice and good teaching but also good company.

I would like to express my special gratitude to Dr. James T. Luxhøj. I deeply appreciate his enthusiasm, encouragement, and valuable comments throughout the course of my PhD study. My gratitude is extended to all members of my PhD committee for their help and support. I also wish to express my gratitude to all my professors in the IE department. Thank you for sharing your experience, knowledge, and expertise.

Life blessed me with the opportunity to meet an exceptional woman, Dr. Susan Albin, her courage, determination and strength with which she responds to life's challenges has led me to profoundly value life and its many gifts. I am deeply thankful for her encouragement during the times when my own faith grew thin. Thank you for always believing in me.

I am very grateful to have gained some extraordinary friends during these years. Especially, I would like to thank Abdullah Karaman, Erol Zeren and Frank Chen. I have no words to thank all your love, moral support, and friendship during all these years. In the same way, my appreciation is extended to all my colleagues of the IE department. Thanks for the warm atmosphere and support that you always provided.

Funding for this work was provided by the Mexican National Council of Science and Technology (CONACYT) and is gratefully acknowledged.

I am forever grateful to my family. My appreciation goes to my parents, whose foresight and values paved the way for a privileged education. Thank you for your unconditional support at each turn of the road. I also want to thank my brother for being such a wonderful friend. I only hope that you realize how much I love you. In the same manner, I would like to thank my extended family, especially my parents-in-law. They have always believed in me. Despite the distance, we were always very close to each other.

Finally, but most of all, I wish to thank the person who has been closest to my endless questions, tears and joy, my husband José Espíritu, coauthor of a number of papers in this thesis. Thank you for always taking my problems as your own, and help me to overcome them. Thank you so much for being my constant source of love and encouragement. I am so lucky to have such a good friend and partner at home as well as at work. With all the ‘cells’ passing in this world it is a fortune that ours ‘collided’.

From the deepest of my heart, I thank all of you,

Heidi A. Taboada

Table of Contents

Abstract	ii
Acknowledgements	iv
Chapter 1 Introduction	1
1.1 Multiple objective problems	2
1.1.1 Existing methods for the solution of multiple objective problems	5
1.1.2 Research contributions	5
1.2 Thesis organization	7
Chapter 2 Metaheuristics literature review	11
2.1 Combinatorial optimization problems	12
2.2 Metaheuristic optimization approaches	13
2.2.1 Population-based methods	15
2.2.1.1 Ant colony optimization (ACO)	15
2.2.1.2 Evolutionary computation (EC)	16
2.2.1.3 Genetic algorithms (GAs)	17
2.2.1.3.1 Genetic algorithms: basic principles and design issues	19
2.2.1.3.1.1 Encoding or chromosome implementation	19
2.2.1.3.1.2 Selection and fitness assignment	22
2.2.1.3.1.3 Elitism	27
2.2.1.3.1.4 Crossover and mutation	27
2.2.1.3.1.4.1 Crossover and mutation for binary encoded individuals	28
2.2.1.3.1.4.2 Crossover and mutation for permutation encoded individuals	30

2.2.1.3.1.4.3 Crossover and mutation for value encoded individuals	31
2.2.1.3.1.4.4 Crossover and mutation for tree encoded individuals	32
2.2.1.3.1.5 Optimal crossover and mutation rates in genetic search	33
2.2.1.3.1.6 Reinsertion	34
2.2.2 Trajectory methods	34
2.2.2.1 Simulated annealing (SA)	34
2.2.2.2 Tabu search (TS)	36
2.3 Summary	38
Chapter 3 Multi-objective optimization	41
3.1 Multi-objective optimization solution methods	41
3.1.1 Single-objective approaches	42
3.1.2 Multiple objective evolutionary algorithms (MOEAs)	43
3.1.3 Differences between MOGAs and single GAs	44
3.2 State-of-the-art multi-objective evolutionary algorithms	45
3.2.1 Vector Evaluated Genetic Algorithm (VEGA)	46
3.2.2 Multi-Objective Genetic Algorithm (MOGA)	47
3.2.3 Niched Pareto Genetic Algorithm (NPGA)	48
3.2.4 Nondominated Sorting Genetic Algorithm (NSGA)	51
3.2.5 Strength Pareto Evolutionary Algorithm (SPEA)	53
3.2.6 Fast Elitist Nondominated Sorting Genetic Algorithm (NSGA-II)	55
3.2.7 Pareto-Archived Evolutionary Strategy (PAES)	58
3.3 Summary	59

4.6.2.1.3 Pruned results by using data clustering: example 1	94
4.6.2.2 Scheduling of unrelated parallel machines: example 2	96
4.6.2.2.1 Pruned results by using the non-numerical ranking preference method: example 2	97
4.6.2.2.2 Pruned results by using data clustering: example 2	100
4.7 Summary	101
Chapter 5 Developed MOEA for design allocation problems	104
5.1 Description of the problem addressed	105
5.2 Multi-criteria formulation of the RAP using GA's	107
5.3 The proposed algorithm: MOEA-DAP	108
5.3.1 Chromosomal representation	110
5.3.2 Constraint-handling method	112
5.3.3 Determination of the initial generation	112
5.3.4 Aggregated fitness function	115
5.3.4.1 Fitness metric 1: distance- based, $f_1(i)$	116
5.3.4.2 Fitness metric 2: dominance count- based, $f_2(i)$	120
5.3.5 Selection step	122
5.3.6 Crossover operator	124
5.3.7 Mutation	127
5.3.8 Elitist reinsertion	129
5.4 Performance Comparison of MOEA-DAP	129
5.5 Summary	135

Chapter 6 MOMS-GA: An extension to MOEA-DAP to consider

multi-state system performance	136
6.1 Introduction	136
6.2 Previous research on multi-state systems (MSS)	137
6.3 Evolutionary approaches in multi-objective optimization	138
6.4 Multi-state system availability estimation method	139
6.4.1 Parallel components	140
6.4.2 Series components	141
6.4.3 Total system reliability evaluation	142
6.5 Multi-objective multi-state genetic algorithm (MOMS-GA)	143
6.6 Numerical examples	145
6.6.1 Example 1	146
6.6.2 Example 2	149
6.7 Summary	152

Chapter 7 A multi-objective evolutionary algorithm for determining

optimal configurations of multi-task production systems	153
7.1 Introduction	153
7.2. Problem description	156
7.3 Multi-state system availability estimation method	157
7.4 Description of the multi-objective evolutionary algorithm	161
7.5 Examples	163
7.5.1 Example 1	163
7.5.2 Example 2	167

7.5.3 Example 3	171
7.6 Summary	175
Chapter 8 A multi-objective prioritized evolutionary algorithm (MoPriGA)	176
8.1 Introduction	176
8.2 Previous research on post-optimality selection	178
8.3 Defining preferences in MOEAs	179
8.4 Description of the multi-objective prioritized GA (MoPriGA)	181
8.5 Examples	185
8.5.1 Case when $f_1 \succ f_2 \succ f_3$	186
8.5.2 Case when $f_2 \succ f_1 \succ f_3$	187
8.5.3 Case when $f_1 \equiv f_2 \succ f_3$	188
8.6 Summary	190
Chapter 9 Future Research	191
9.1 Development of a multi-purpose MOEA	191
9.2 Handling uncertainties in MOEAs	192
Appendix A: Runs for Performance Comparison	195
References	211
Vita	225

List of tables

Table 2.1	Selection probability and fitness value	23
Table 4.1	Component choices for each subsystem	81
Table 4.2	Summary of results obtained with the clustering analysis	85
Table 4.3	Processing times for PWB scheduling problem	88
Table 4.4	Processing costs for PWB scheduling problem	88
Table 4.5	Pruned solutions	90
Table 4.6	Solutions found in the pruned Pareto set in ten simulation runs	91
Table 4.7	Analytical pruning results	94
Table 4.8	Results obtained with the cluster analysis	96
Table 4.9	Processing times for PWB scheduling problem	97
Table 4.10	Processing costs for PWB scheduling problem	97
Table 4.11	Pruned solutions	98
Table 4.12	Solutions found in the pruned Pareto set in ten simulation runs	100
Table 4.13	Results obtained with the cluster analysis	101
Table 5.1	Dominance count in initial population	114
Table 5.2	Dominance count in the first nondominated set	114
Table 5.3	Initial solutions	115
Table 5.4	Nondominated solutions	115
Table 5.5	Standardized nondominated set	116
Table 5.6	Distance from each solution to the rest of the solutions	117
Table 5.7	Fitness value 1 for the nondominated set	118
Table 5.8	Fitness value 1 for the nondominated set (standardized space)	119

Table 5.9	Dominance count in the nondominated set	121
Table 5.10	Fitness value 2 bounds for the nondominated set	122
Table 5.11	Fitness value 2 for the nondominated set (standardized space)	122
Table 5.12	Total Fitness value of nondominated solutions	123
Table 5.13	Ranked nondominated individuals	123
Table 5.14	Component choices for each subsystem	129
Table 5.15	Performance comparison	131
Table 5.16	Maximum and minimum values found in \mathbf{Y}_{true}	132
Table 6.1	Characteristics of the system elements available	147
Table 6.2	Parameters of the cumulative demand curve	147
Table 6.3	Example design configurations of Example 1	149
Table 6.4	Characteristics of the system elements available	150
Table 6.5	Parameters of the cumulative demand curve	150
Table 6.6	Example design configurations of example 2	152
Table 7.1	Characteristics of the machines available	164
Table 7.2	Parameters of the cumulative demand curve	164
Table 7.3	Chosen design configuration for example 1	167
Table 7.4	Chosen design configuration for example 1 when considering $f_2 \succ f_1 \succ f_3$	167
Table 7.5	Characteristics of the machines available	168
Table 7.6	Parameters of the cumulative demand curve	168
Table 7.7	Compromised example design configuration for example 2	171
Table 7.8	Characteristics of the system elements available	172

Table 7.9	Parameters of the cumulative demand curve	173
Table 7.10	Example design configurations of Example 3	174
Table 8.1	Component choices for each subsystem	186
Table 8.2	Example design configurations	190

List of illustrations

Figure 1.1	Pareto Front of a bi-objective minimization problem	3
Figure 1.2	Achieving a balance between single solutions and Pareto-optimal solutions	5
Figure 2.1	Local optima and global optima	13
Figure 2.2	Binary encoding example	20
Figure 2.3	Permutation encoding example	21
Figure 2.4	Value encoding example	21
Figure 2.5	Example of a chromosome with tree encoding	22
Figure 2.6	Roulette-wheel selection	24
Figure 2.7	Stochastic universal sampling	25
Figure 2.8	Tournament selection, $k=2$	26
Figure 2.9	Single-point crossover	28
Figure 2.10	Double-point crossover	29
Figure 2.11	Multi-point crossover	29
Figure 2.12	Mutation in binary encoded individuals	30
Figure 2.13	Single-point crossover for permutation encoded individuals	30
Figure 2.14	Order changing mutation for permutation encoded individuals	31
Figure 2.15	Double-point crossover	31
Figure 2.16	Real value mutation	32
Figure 2.17	Crossover for tree encoded individuals	33
Figure 3.1	Population diversity is achieved	45
Figure 3.2	Population diversity is not achieved	45

Figure 3.3	The VEGA selection mechanism	47
Figure 3.4	Population ranking according to MOGA	48
Figure 3.5	MOGA overview	48
Figure 3.6	Niche count in NPGA	49
Figure 3.7	NPGA overview	50
Figure 3.8	NPGA 2 overview	51
Figure 3.9	Nondominated fronts according to NSGA	52
Figure 3.10	NSGA overview	53
Figure 3.11	Population ranking according to SPEA	54
Figure 3.12	SPEA overview	54
Figure 3.13	Crowding distance calculation	56
Figure 3.14	Working mechanism of the NSGA-II algorithm	57
Figure 3.15	NSGA-II overview	57
Figure 3.16	PAES pseudocode	59
Figure 4.1	Achieving a balance between single solutions and Pareto-optimal solutions	62
Figure 4.2	Methods to prune the Pareto-optimal set	64
Figure 4.3	Plane containing set of possible weights	66
Figure 4.4	Weight region for the $f_1 \succ f_2 \succ f_3$ objective function preference	66
Figure 4.5	Distribution of random weights used for a three objective problem	69
Figure 4.6	Overview of clustering algorithmic implementation	75
Figure 4.7	General series-parallel redundancy system	76
Figure 4.8	Pareto-optimal set	82

Figure 4.9	Comparing pruned Pareto solution with the Pareto-optimal solution set for reliability versus cost	83
Figure 4.10	Comparing pruned Pareto solution with the Pareto-optimal solution set for reliability versus cost	83
Figure 4.11	Clustered Pareto-optimal set	84
Figure 4.12	Pareto-optimal set of Example 1 (PWB) in a three-dimensional space	89
Figure 4.13	Pruned solutions for the $w_1 > w_2 > w_3 > w_4$ objective function preference in a two-dimensional space	90
Figure 4.14	Schedule for solution 2	90
Figure 4.15	Algorithm to prune the Pareto set given objective functions preferences	92
Figure 4.16	Clustered data in a three-dimensional space	95
Figure 4.17	Clustered data in a three-dimensional space	95
Figure 4.18	Schedule for solution number 6	96
Figure 4.19	Pareto-optimal set of Example 2	98
Figure 4.20	Pruned solutions for the $f_1 \succ f_3 \succ f_2$ objective function preference in a two-dimensional space	99
Figure 4.21	Schedule for solution 48	99
Figure 4.22	Clustered data for the second PWB example	100
Figure 5.1	Flowchart of MOEA-DAP	110
Figure 5.2	Dominance in a bi-objective problem	115
Figure 5.3	First nondominated set in a normalized space	120

Figure 5.4	First nondominated set in a bi-objective normalized space	120
Figure 5.5	Sub-chromosome representation of individuals	125
Figure 5.6	Crossover operation example	126
Figure 5.7	Example of mutation	128
Figure 5.8	Nondominated solutions in Y_{true} obtained from NSGA-II algorithm	132
Figure 5.9	Nondominated solutions in Y_{true} obtained from NSGA-II algorithm. Rel vs Cost	133
Figure 5.10	Nondominated solutions in Y_{true} obtained from NSGA-II algorithm. Rel vs Weight	133
Figure 5.11	Nondominated solutions in Y_{true} obtained from NSGA-II algorithm. Cost vs Weight	133
Figure 5.12	Nondominated solutions in Y_{true} obtained from MOEA-DAP algorithm	133
Figure 5.13	Nondominated solutions in Y_{true} obtained from MOEA-DAP. Rel vs Cost	133
Figure 5.14	Nondominated solutions in Y_{true} obtained from MOEA-DAP. Rel vs Weight	133
Figure 5.15	Nondominated solutions in Y_{true} obtained from MOEA-DAP. Cost vs Weight	133
Figure 6.1	Pareto front of example 1	148
Figure 6.2	Pareto front of example 1 in a two dimensional space	148
Figure 6.3	Pareto front of example 2	151

Figure 6.4	Pareto front of example 2 in a two dimensional space	151
Figure 7.1	Representation of solutions	161
Figure 7.2	Pareto front of example 1	165
Figure 7.3	Pareto front of example 1 in a two dimensional space	166
Figure 7.4	Pareto front of example 2 in a three dimensional space	169
Figure 7.5	Pareto front of example 2 in a two dimensional space	170
Figure 7.6	Pareto front of example 3 in a three dimensional space	173
Figure 7.7	Pareto front of example 3 in a two dimensional space	174
Figure 8.1	Approach to obtain solutions that reflect DM objective function preferences (after the search)	177
Figure 8.2	New approach to incorporate DM objective function preferences within the EA (during the search)	178
Figure 8.3	Working mechanism of MoPriGA	185
Figure 8.4	Preferred solutions found in the final Pareto set: Case when $f_1 \succ f_2 \succ f_3$	187
Figure 8.5	Preferred solutions found in the final Pareto set in a two-dimensional perspective	187
Figure 8.6	Preferred solutions found in the final Pareto set: Case when $f_2 \succ f_1 \succ f_3$	188
Figure 8.7	Preferred solutions found in the final Pareto set in a two-dimensional perspective	188
Figure 8.8	Preferred solutions found in the final Pareto set: Case when $f_1 \equiv f_2 \succ f_3$	189

Figure 8.9	Preferred solutions found in the final Pareto set in a two-dimensional perspective	189
Figure 9.1	Variance as a Risk measure	193

1. Introduction

This thesis is focused on the development of new methods for the solution and analysis of multiple objective optimization problems. These new methods provide a balance between commonly existing methods, and for decision-makers, these new methods more appropriately offer practical solutions. In the same way, novel evolutionary algorithms are presented. These algorithms offer distinct advantages compared to existing algorithms. Moreover, the developed algorithms are intended to be used on a wide range of optimization problems rather than any specific one.

Most real-world engineering optimization problems involve the achievement of several objectives, normally conflicting with each other. These problems are called “multi-objective,” “multi-criteria,” or “vector” optimization problems, and were originally studied in the context of economics. However, scientists and engineers soon realized the importance of solving multi-objective optimization problems, and the development of techniques to model and solve such problems became an important area within operations research.

Because of the conflicting nature of their objectives, multi-objective optimization problems do not normally have a single optimal solution, and in fact, they even require the definition of a new notion or interpretation of “optimum.” The most commonly adopted notion of optimality in multi-objective optimization is that originally proposed

by Edgeworth (1881) and later generalized by Pareto (1896). Such a notion is called *Edgeworth-Pareto optimality* or, more commonly, *Pareto optimality*.

Classical methods are often not efficient in solving multiple objective problems because they require repetitive applications to find multiple Pareto-optimal solutions, and in some occasions, repetitive applications do not guarantee finding distinct Pareto-optimal solutions. These methods are also susceptible to the shape or continuity of the Pareto-optimal set, and therefore, their applicability may be severely limited in many real-world applications. This context gives the main motivation for using evolutionary algorithms for solving multi-objective optimization problems, which provide an efficient approach to find multiple Pareto-optimal solutions simultaneously in a single run.

Recently, several methods for solving multi-objective optimization problems have been developed and studied. However, relatively little prior work has been done on the evaluation of the solutions obtained by these algorithms. These algorithms give as a result, a non-dominated set of solutions. If this set has a relatively small number of solutions, then standard decision-making tools, such as the Analytic Hierarchy Process (AHP), exist. However, this set often contains a large number of solutions, from which the decision-maker has to finally select one solution for system implementation and the selection of one solution over the others may become an arduous task.

1.1 Multiple objective problems

The complexity of solving multi-objective problems involves two types of problem difficulties: *i)* multiple, conflicting objectives, and *ii)* a highly complex search space. For instance, consider a production planning example with two objectives, cost (f_1) and makespan (f_2), to be minimized under a set of constraints. For this bi-objective problem,

an optimum design should ideally be a solution that achieves the minimum makespan at the minimum cost without violating the constraints. If such a solution exists, then it is necessary to solve just a single-objective optimization problem, because the optimal solution for the first objective is also optimal for the second objective. However, this rarely happens in real multi-objective problems.

Multi-objective optimization refers to the solution of problems with two or more objectives to be satisfied simultaneously. Often, such objectives are in conflict with each other and are expressed in different units. Because of their nature, multi-objective optimization problems normally have not one, but a set of solutions, which are called Pareto-optimal solutions or nondominated solutions (Chankong & Haimes 1983; Hans 1988). When such solutions are represented in the objective function space, the graph produced is called the Pareto front of the problem, as shown in Figure 1.1.

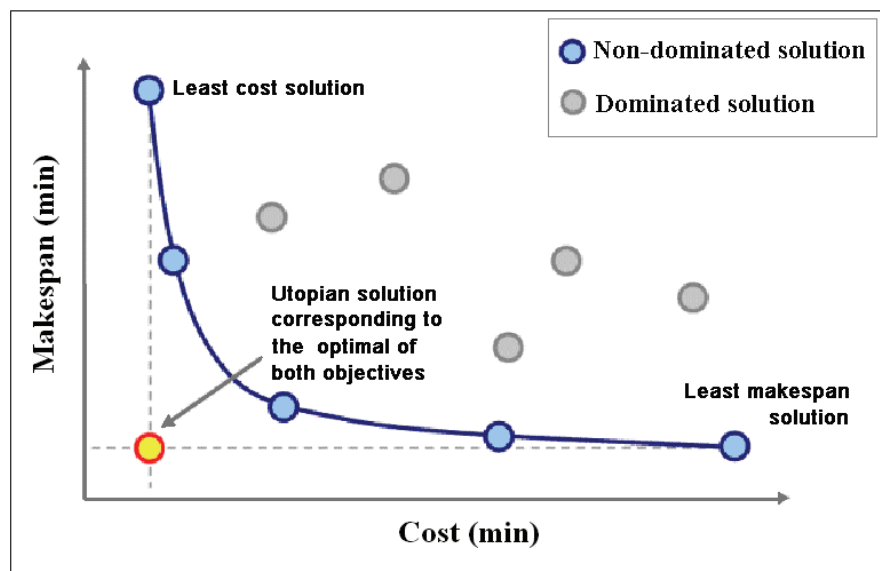


Figure 1.1 Pareto Front of a bi-objective minimization problem

A general formulation of a multi-objective optimization problem consists of a number of objectives and is associated with a number of inequality and equality constraints. Mathematically, the problem can be written as follows (Rao, 1991):

Minimize/Maximize $f_i(\mathbf{x})$ for $i = 1, 2, \dots, n$

Subject to:

$$g_j(\mathbf{x}) \leq 0 \quad j = 1, 2, \dots, J$$

$$h_q(\mathbf{x}) = 0 \quad q = 1, 2, \dots, Q$$

In the vector function, $f_i(\mathbf{x})$, some of the objectives are generally in conflict with others, and some may have to be minimized while others are maximized. Thus, the multi-objective optimization problem is defined as the problem to find the vector $\mathbf{x} = (x_1, x_2, \dots, x_n)$, i.e., solution which optimize the vector function, $f_i(\mathbf{x})$.

The constraints define some feasible region X , and any point $\mathbf{x} \in X$ defines a feasible solution. Normally, we rarely have a situation in which all the $f_i(\mathbf{x})$ values have an optimum in X at a common point \mathbf{x} . Therefore, it is necessary to establish certain criteria to determine what is considered as an optimal solution, and this criteria is nondominance. Thus, solutions to a multi-objective optimization problem are mathematically expressed in terms of nondominance.

Without loss of generality, for a minimization problem for all objectives, a solution \mathbf{x}_1 dominates a solution \mathbf{x}_2 , if and only if, the two following conditions are true:

- \mathbf{x}_1 is no worse than \mathbf{x}_2 in all objectives, i.e., $f_i(\mathbf{x}_1) \leq f_i(\mathbf{x}_2) \quad \forall i, i \in \{1, 2, \dots, n\}$
- \mathbf{x}_1 is strictly better than \mathbf{x}_2 in at least one objective, i.e., $f_i(\mathbf{x}_1) < f_i(\mathbf{x}_2)$ for at least one i .

Then, the optimal solutions to a multi-objective optimization problem are the set of nondominated solutions X and they are usually known as Pareto-optimal set (Zeleny, 1982).

1.1.1 Existing methods for the solution of multiple objective problems

Existing methods require either the aggregation of the objectives into an overall objective function or the determination of a Pareto set. The first method, in which one single solution is obtained, requires precise knowledge of the objective function priorities and relative importance, and thus, very broad and very detailed knowledge of the system is demanded, i.e., systems usage, customer priorities and tendencies management priorities, etc. Some methods that belong to this first approach are the weighted sum method, goal programming, and utility theory among others. These methods are further discussed in Chapter 3. In contrast, in the second method a Pareto-optimal set is obtained. This set usually contains a large number (in some cases, thousands) of solutions and from the decision-maker's perspective, consideration of all the nondominated solutions can be prohibitive and inefficient. Generally, Multiple Objective Evolutionary Algorithms (MOEAs) are used to determine a Pareto-optimal set.

1.1.2 Research contributions

From the discussion above, it can be seen that there is a need for the development of new methods that provide a balance between single solutions and Pareto optimality, as shown in Figure 1.2.

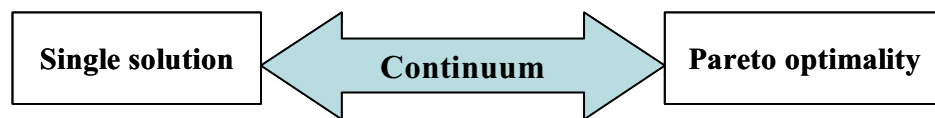


Figure 1.2 Achieving a balance between single solutions and Pareto-optimal solutions

The contributions of this thesis are summarized as follows:

- Development of new methods for the solution of multi-objective optimization problems that achieve a balance between single solutions and Pareto-optimal solutions.
- A method developed for decision-makers that can prioritize the objective functions to find appropriate smaller sets of solutions that clearly reflect his/her objective function preferences.
- Development of a practical method that allows the decision-maker to obtain a comparably smaller set of solutions when he/she does not have any *a priori* information of the objective function preferences. This method represents the integration of existing data mining techniques into a new approach for analyzing Pareto-optimal sets.
- A new multiple objective evolutionary algorithm that, when tested on design allocation problems, is observed to be superior to one of the most successful evolutionary algorithms that currently exists.
- New multi-objective evolutionary algorithms that consider, not only multi-state, but also multi-state multi-task systems.
- A new multi-objective evolutionary algorithm that incorporates the knowledge of the decision-maker objective function preferences directly into the search process for the first time.

1.2 Thesis organization

This thesis is organized as follows:

In **Chapter 2**, a brief overview of some of the most well-known metaheuristic algorithms and their classification is presented. Also, the basic design issues in single-objective genetic algorithms are introduced.

In **Chapter 3**, a description of the two primary approaches to identify solution(s) to multiple objective problems is introduced. Since in Chapter 2, the basic design issues in single-objective genetic algorithms were introduced, in Chapter 3, some of the aspects that make single-objective genetic algorithms different from multiple-objective genetic algorithms are reviewed. Due to their importance, most of the effort is concentrated on an extensive overview of some of the currently most successful state-of-the-art MOEAs.

In **Chapter 4**, several methods are presented, which can efficiently organize and reduce the size of the Pareto-optimal set, and thus, make it easier for the decision-maker to comparatively analyze a small set of solutions, and finally, select the most desirable one for system implementation.

The primary idea is based on an unsupervised data mining technique, in which the solutions in the Pareto-optimal set are clustered so that the Pareto-optimal front is reduced to a set of k disjoint clusters or to the solutions contained in the “knee cluster”. These are likely to be the most interesting solutions, instead of having to analyze the entire Pareto set.

The second idea is based on an approach analogous to the weighted sum method on the Pareto-optimal set except that specific numerical objective function weights or penalties are not required. The objective functions are ranked ordinally based on their

importance to the decision-maker (ties allowed). To determine a promising set of recommended solutions, a pruning process is introduced. In this process, the objective functions are scaled and repeatedly combined into a single objective function using numerous randomly generated weight sets. Each random weight set adheres to the decision-maker preferences and is generated from a multi-dimensional weight function analogous to a multiple dimension probability density function. This is a simple method that yields efficient results for any user who can prioritize the objective functions to find appropriate solutions.

In the present work, different multi-objective optimization problems are presented which (in different ways) incorporate the ideas that are outlined above.

Chapter 5 presents a newly developed multi-objective evolutionary algorithm for solving system design allocation problems, MOEA-DAP. Because EAs are appropriate for high-dimension stochastic problems with many nonlinearities or discontinuities, they are suited for solving many different problems, including reliability design problems. This new algorithm uses a genetic algorithm based on rank selection and elitist reinsertion, and a modifying genetic operator constraint handling method. MOEA-DAP, mainly differs from other MOEAs in the type of crossover operator used, that appears to encourage the exploration of the search space. A comparison between one of the most successful evolutionary algorithms that currently exists, NSGA-II, and the new algorithm, indicates that MOEA-DAP is more powerful to solve multi-objective design allocation problems based on the example problems considered.

Chapter 6 introduces a multi-objective multi-state genetic algorithm (MOMS-GA) to solve multiple objective multi-state reliability and availability optimization design

problems. MOMS-GA was developed as an extension of MOEA-DAP, which was developed to consider binary-state reliability. That is, the evolutionary algorithm assumed that the system and its components could be in either a working or a failed state only. In contrast, the developed MOMS-GA works under the assumption that both, the system and its components, experience more than two possible states of performance. The universal moment generating function (UMGF) approach was implemented in the algorithm to obtain the system availability.

Analogous to the allocation of redundant components to meet high reliability specifications in reliability optimization, there are many other engineering design and development projects that require the allocation of redundant components such as in the machine allocation phase in production systems. Despite the clear relationship between the two types of allocation problems, production scheduling and reliability optimization are typically treated independently in the research literature and in practice. **Chapter 7** shows how system availability can be used within the context of multi-task production systems. For this purpose, this chapter presents a new multiple objective evolutionary algorithm to determine optimal configurations of multi-state, multi-task production systems based on availability analysis. The performance of a manufacturing system is greatly influenced by its configuration. In the algorithm, availability is used in the context of multi-task production systems to select a particular configuration that maximizes the probability of meeting a required demand for each specific task, or the expected productivity for each task.

Chapter 8 introduces a multiple objective prioritized genetic algorithm (MoPriGA). MoPriGA conceptually combines the idea of the working mechanism of MOEA-DAP

and post-Pareto pruning. This algorithm incorporates the knowledge of the decision-maker objective function preferences based on the inclusion of the uncertain weight function, $f_w(\mathbf{w})$, into the search process. MoPriGA is a powerful algorithm that searches extensively in the region of interest without reducing the capability of the search, but simply focusing intensely on the region of the Pareto set of most interest to the decision-maker.

There are numerous opportunities for developing novel and original research in the evolutionary multi-objective optimization area. **Chapter 9** presents a discussion of some of these research opportunities to extend the current research. One of them is the development of new multipurpose MOEAs with the ability of finding homogeneously distributed solutions in the final Pareto front with the robustness of balancing proximity and diversity during the searching process.

One of the most important tasks in the future research is the incorporation of uncertainties in MOEAs. Several risk measures, currently used to manage uncertainties in portfolio optimization, such as value-at-risk, conditional-value-at risk, and expected shortfall are intended to be adapted in the searching process in order to incorporate risk in the solution of multi-objective optimization problems. This is an entirely new research area, and if successful, it will be a great contribution to the research community.

2. Metaheuristics literature review

This dissertation involves the use of metaheuristics to obtain solutions for multiple objective optimization problems. Solutions from existing metaheuristics are pruned to obtain smaller, promising Pareto sub-sets and new metaheuristics are designed. In this chapter, an overview of existing metaheuristic methods is given.

Metaheuristics can be considered as high level strategies for exploring search spaces by using different methods. This chapter introduces four of these metaheuristic optimization techniques: Genetic Algorithms (GAs), Ant Colony Optimization (ACO), Simulated Annealing (SA) and Tabu Search (TS). GAs locate optima using processes similar to those in natural selection and genetics. The ACO algorithm consists of a set of artificial “ants” that incrementally construct solutions by adding components to their solutions. SA operates analogously to the searching of minimum energy configurations in metal annealing. TS is a metaheuristic procedure that employs dynamically generated constraints or tabus to guide the search for optimum solutions (Pham & Karaboga, 2000).

This thesis is concerned with the development of new multiple objective genetic algorithms (MOGAs) as solution methods to multiple objective optimization problems. Therefore, a more detailed description of GAs is presented throughout this chapter. Although single-objective GAs are different from multiple objective GAs, they both share basic design characteristics such as the crossover and mutation operators among others.

2.1 Combinatorial optimization problems

Many optimization problems consist of the search for a “best” configuration of a set of variables to achieve some goals. Optimization problems generally divide naturally into two categories: those with continuous decision variables and those with discrete decision variables, which we call *combinatorial*. According to Papadimitriou & Steiglitz (1982), in Combinatorial Optimization (CO) problems, we are searching for an object from a finite - or possibly countably infinite - set. This object is typically an integer number, a subset, a permutation, or a graph structure.

A single objective combinatorial optimization problem, $P=(S, f)$, can be mathematically defined by:

- a set of variables $X=(x_1, \dots, x_n)$;
- variable domains D_1, \dots, D_n ;
- constraints among variables;
- an objective function f to be minimized

The set S of all possible feasible assignments is usually called the *search (or solution) space*, as each element of the set can be seen as a candidate solution. To solve a combinatorial optimization problem, one has to find a solution $s^* \in S$ with minimum objective function value, that is, $f(s^*) \leq f(s) \forall s \in S$. s^* is called a global optimal solution of (S, f) .

Finding a *global optimal* solution to an instance of some problems can be prohibitively difficult, but it is often possible to find a solution s' which is best in the sense that there is nothing better in its neighborhood $N(s')$, such a solution is called *local optimal* with respect to $N(s')$.

To illustrate the difference between local optima and global optima, consider the instance of an optimization problem (S, f) defined by

$$F=[0,1] \subseteq R$$

And the cost function c sketched in Figure 2.1. Let the neighborhood be defined simply by closeness in Euclidean distance for some $\varepsilon > 0$.

$$N(s') = \{x: x \in F \text{ and } |x - s'| \leq \varepsilon\}$$

Then, if ε is suitably small, the points A , B and C are all local optima, but only B is global optima (Papadimitriou & Steiglitz, 1982).

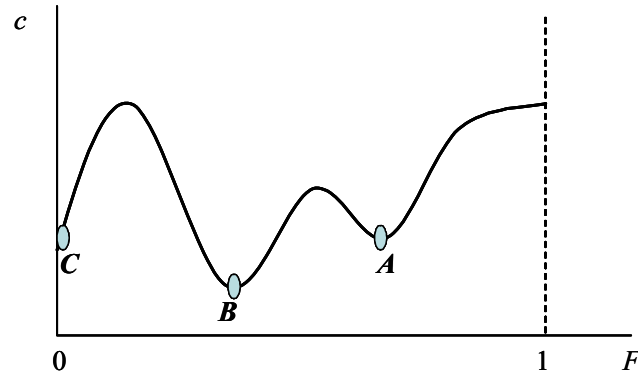


Figure 2.1 Local optima and global optima

2.2 Metaheuristic optimization approaches

Due to the practical importance of CO problems, many algorithms have been developed to provide solutions for these types of problems. These algorithms can be classified as either *complete* or *approximate* algorithms. Complete algorithms are guaranteed to find, for every finite size instance of a CO problem, an optimal solution in bounded time (see Papadimitriou & Steiglitz (1982) and Nemhauser & Wolsey (1988)).

Complete methods might need exponential computation time which may be too computationally expensive for practical purposes. Thus, the use of approximate methods to solve CO problems has received more and more attention during the last years. In

approximate methods, we sacrifice the guarantee of finding optimal solutions for the sake of getting good solutions in a significantly reduced amount of time.

In the last 20 to 25 years, a new kind of approximate algorithm has emerged which basically tries to combine basic heuristic methods in higher level frameworks aimed at efficiently and effectively exploring a search space (Blum & Roli, 2003). These methods are commonly called *metaheuristics*. The term *metaheuristic*, first introduced in Glover (1986), derives from the composition of two Greek words. *Heuristic* derives from the verb *heuriskein* which means “to find”, while the suffix *meta* means “beyond, in an upper level.”

In general, metaheuristics can be loosely defined as high level strategies for exploring search spaces by using different methods. Some of the fundamental properties which characterize metaheuristics are:

- Metaheuristics are strategies that “guide” the search process.
- The goal is to efficiently explore the search space in order to find (near) optimal solutions.
- Metaheuristics are not problem-specific.
- Techniques which constitute metaheuristics algorithms range from simple local search procedures to complex learning processes.
- Metaheuristic algorithms are approximate and usually non-deterministic.
- They may incorporate mechanisms to avoid getting trapped in confined areas of the search space.

There are different ways to classify metaheuristics. One of the most common ways is to classify them as *single point local-search methods* (or trajectory methods) in which a

single solution is improved incrementally by making small changes to it; and *population-based methods*, where a population of solutions is evolved in parallel, describing the evolution of a set of points in the search space.

Some of the most traditional and important metaheuristics representative of each class are:

- *Population-based methods*: Ant Colony Optimization (ACO) and Evolutionary Computation (EC) including Genetic Algorithms (GAs)
- *Trajectory methods*: Simulated Annealing (SA) and Tabu Search (TS)

Since this thesis proposal is concerned with the solutions and applications of the most recent versions of the last method, more emphasis is devoted to its explanation and understanding, although a brief overview of the other algorithms is also described to outline the different concepts that are used in the trajectory vs. population-based search methods.

2.2.1 Population-based methods

2.2.1.1 Ant colony optimization (ACO)

Ant colony optimization (ACO) is a metaheuristic approach firstly proposed by Dorigo (1992) for solving discrete combinatorial optimization problems. ACO is inspired by the behavior of ants when finding the shortest path between a food source and their nest. Ants deposit a substance called pheromone while exploring paths and also use the level of concentration of pheromone to decide which path to follow. Since the pheromone evaporates as time passes, the concentration is strongest in the shortest paths, making them more attractive for other ants that also contribute to enhance the attractiveness of

the path. The ACO algorithm consists of a set of artificial ants that incrementally construct solutions by adding components to their solutions.

ACO has been applied successfully to a large number of difficult combinatorial optimization problems such as the traveling salesman problem, as in Dorigo & Gambardella (1997), scheduling problems, and routing problems in telecommunication networks. Further research on the ACO metaheuristic can be found in Dorigo *et al.* (1996, 1999).

2.2.1.2 Evolutionary computation (EC)

Evolutionary computation (EC) algorithms are inspired by nature's capability to evolve well adapted living beings to their environment. There has been a variety of slightly different EC algorithms proposed over the years. Basically they fall into three different categories which have been developed independently from each other. These are Evolutionary Programming (EP) developed by Fogel (1962) and Fogel *et al.* (1966), Evolutionary Strategies (ES) proposed by Rechenberg (1973) and Genetic Algorithms (GAs) initially proposed by Holland (1975).

Evolutionary computation algorithms have been successfully applied to numerous problems from different domains, including optimization, automatic programming, machine learning, economics, operations research, ecology, population genetics, studies of evolution and learning, and social systems (Mitchell, 1996).

Since, in part, this thesis is concerned with the development of new multiple objective genetic algorithms (MOGAs) a more detailed description of Genetic Algorithms will be presented, since they represent the basis for a strong understanding of the current state-of-the-art multiple objective evolutionary algorithms (MOEAs).

2.2.1.3 Genetic algorithms (GAs)

GAs, developed by Holland (1975), are nondeterministic stochastic search/optimization methods that simulate the process of natural evolution to solve problems with a complex solution space. GAs are computer-based algorithms that mimic some of the known mechanisms in evolution, as key elements in their design and implementation.

In its general form, a GA works as follows: an initial population of individuals is generated at random or heuristically. At every generation, the individuals in the current population are decoded and evaluated according to some predefined quality criterion, referred to as the fitness function.

Creation of new members is done by *crossover* and *mutation* operations. The effectiveness of the *crossover* operator dictates the rate of convergence; while the *mutation* operator prevents the algorithm from prematurely converge to a local optimum.

During the selection procedure, individuals are chosen according to their fitness value. Individuals with high-fitness values have better chances of reproducing, while low-fitness ones are more likely to disappear. The procedure is terminated either when the search process stagnates or when a predefined number of generations is reached.

Genetic algorithms are advanced search mechanisms ideal for exploring large and complex problem spaces. However, it is important to not forget that GAs are stochastic iterative processes and they are not guaranteed to converge to the global optimal solution. Hence, the termination condition may be specified as some fixed maximal number of generations or as the attainment of an acceptable fitness level.

Some of the essential differences between GAs and other forms of optimization, according to Goldberg (1989) are:

- *GAs search a population of points in parallel, not a single point.* This gives the GAs the power to search “noisy” spaces. That is, instead of relying on a single point to search through the space, the GAs look at many different areas of the problem space at once.
- *GAs use probabilistic transition rules, not deterministic ones.* This is a direct result of the randomization techniques used by GAs.
- *GAs work on an encoded form of the solution parameters rather than their actual values.* Besides the computational advantage that this represents, it also provides the possibility of crossover and mutation.
- *GAs use only payoff information to guide themselves to the problem space.* GAs do not require derivative information, or other auxiliary knowledge; only the objective function and corresponding fitness levels influence the direction of search.

A pseudocode of a basic GA is as follows:

1. [*Start*] Generate random population of n chromosomes
2. [*Fitness*] Evaluate the fitness $f(\mathbf{x})$ of each chromosome \mathbf{x} in the population
3. [*New population*] Create a new population by repeating following steps until the new population is complete
 - 3.1 [*Selection*] Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
 - 3.2 [*Crossover*] With a crossover probability, crossover the parents to form new offspring (children) solutions. If no crossover was performed, offspring is the exact copy of parents.

- 3.3 [*Mutation*] With a mutation probability, mutate new offspring at each locus (position in chromosome).
- 3.4 [*Accepting*] Place new offspring in the new population
4. [*Replace*] Use new generated population for a further run of the algorithm
5. [*Test*] If a defined stopping criterion is satisfied, stop, and return the best solution in current population
6. [*Loop*] Go to Step 2

2.2.1.3.1 Genetic algorithms: basic principles and design issues

GAs remain the most recognized form of evolutionary computation algorithms. In GA terminology, a solution vector $x \in X$ is called an individual or a *chromosome*. Chromosomes are made of discrete units called *genes*. Each gene controls one or more features of the chromosome. In the original implementation of GA by Holland (1975), genes are assumed to be binary digits. In later implementations, more varied gene types have been introduced. Normally, a chromosome corresponds to a unique solution x in the solution space. This requires a mapping mechanism between the solution space and the chromosomes. This mapping is called an encoding. In fact, GA works on the *encoding* of a problem, not on the problem itself (Konak *et al.*, 2006).

2.2.1.3.1.1 Encoding or chromosome implementation

One of the first issues that must be resolved when designing the GA is to decide the type of encoding to use, which is simply the form of the basic chromosome. This determines the requirements and complexity of the genetic operators and directly affects the performance of those operators. In part, the chromosome implementation will be driven by the type of problem to be solved. (Is it number based; integer or real?, what are

the precision requirements? Is it symbolic; arbitrary symbol length or fixed, encoded or pre-decoded? etc.)

There are many other ways of encoding: binary encoding, permutation encoding, value encoding, and tree encoding are among the most used encoding systems. These encoding schemes are discussed below.

Binary encoding

Binary encoding is the most common encoding scheme, primarily because the first research of GA used this type of encoding and because of its relative simplicity. In binary encoding, every chromosome is a string of bits - 0 or 1.

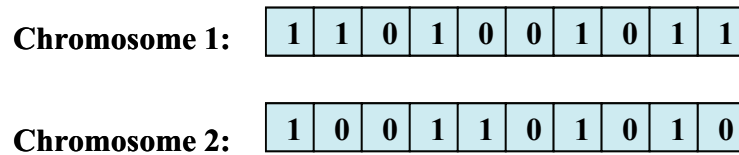


Figure 2.2 Binary encoding example

Each chromosome has one binary string and each bit in this string can represent some characteristic of the solution or the whole string can represent a number.

Permutation encoding

In permutation encoding, every chromosome is a string of numbers that represent a position in a sequence. Sometimes, the use of this representation provides a convenient and natural way of expressing the mapping from representation to problem domain.

For instance, consider the traveling salesman problem, the task being to find the shortest route visiting all the cities from a given set exactly once. By using integer labels, each candidate solution can be uniquely represented as a permutation of these elements. For example, in a seven-city tour $\{2, 7, 1, 3, 5, 6, 4\}$ and $\{6, 4, 7, 1, 5, 3, 2\}$ represent

paths between the cities. Thus the chromosomes used in a GA to solve this problem would contain seven integers, each integer corresponding to a city in the tour.

Chromosome 1:

2	7	1	3	5	6	4
---	---	---	---	---	---	---

Chromosome 2:

6	4	7	1	5	3	2
---	---	---	---	---	---	---

Figure 2.3 Permutation encoding example

Note that permutation encoding and value (integer) encoding differ in the aspect that in permutation encoding there are no genes with the same value in a given chromosome, while in value encoding such a situation is allowed.

Value encoding

Value encoding can be used in problems where some complicated values, such as real numbers, are used and where binary encoding would not suffice. While value encoding is very good for some problems, it is often necessary to develop some specific crossover and mutation techniques for these chromosomes.

Under the value encoding scheme, every chromosome is a sequence of some values. Values can be anything connected to the problem, such as (real) numbers, characters, or any objects.

Chromosome 1:

1.23	2.15	4.92	2.20	3.41	5.68
------	------	------	------	------	------

Chromosome 2:

A	B	E	D	B	C
---	---	---	---	---	---

Chromosome 3:

N	W	S	N	E	N
---	---	---	---	---	---

Figure 2.4 Value encoding example

In chromosome 1 (Figure 2.4), A represents a real value for specific variables; in chromosome 2, A could represent a particular task, B another, etc. In chromosome 3, N could be north, S south, etc.

Tree encoding

Tree encoding is useful for evolving programs (i.e. genetic programming) or any other structures that can be encoded in trees. In the tree encoding every chromosome is a tree of some objects, such as functions or commands in the programming language.

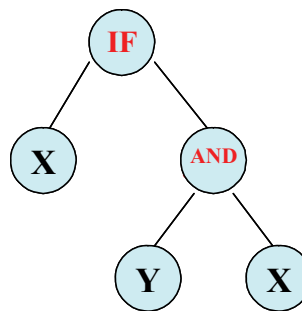


Figure 2.5 Example of a chromosome with tree encoding

Tree encoding has been used, for example, in heuristics for computing constrained minimum spanning trees – minimum-weight spanning trees satisfying an additional constraint, such as on the number of leaves, maximum degree, or diameter of the tree (Edelson & Gargano, 2000; Zhou & Gen, 1997).

2.2.1.3.1.2 Selection and fitness assignment

According to Darwin's evolution theory, the fittest members of a population should survive to create new offspring. The selection operator is intended to improve the average quality of the population by giving higher quality individual a higher probability of survival. There are many methods in selecting the best chromosomes. Examples are roulette wheel selection, stochastic universal sampling selection, rank selection, tournament selection, steady-state selection and some others.

Roulette wheel selection

First, parents are selected according to their fitness. The better the chromosomes are, the more chances to be selected they have. The simplest selection scheme is roulette-wheel selection, also called stochastic sampling with replacement (Baker, 1987). This is a stochastic algorithm and involves the following technique:

1. The individuals are mapped to contiguous segments of a line, such that each individual's segment is equal in size to its fitness.
2. A random number is generated and the individual whose segment spans the random number is selected.
3. The process is repeated until the desired number of individuals is obtained (called mating population).

This technique is analogous to a roulette wheel with each slice proportional in size to the fitness function of every chromosome. The bigger the value is, the larger the section is.

As an example, Table 2.1 shows the selection probability for seven individuals. Individual 1 is the fittest individual and occupies the largest interval, whereas individual 6, as the second least fit individual, has the smallest interval on the line (see Figure 2.6). Individual 7, the least fit interval, has a fitness value of 0 and get no chance for reproduction.

Table 2.1 Selection probability and fitness value

Individual	1	2	3	4	5	6	7
Fitness value	3.0	2.5	2.0	1.5	1.0	0.5	0
Selection probability	0.2857	0.2381	0.1905	0.1429	0.0952	0.0476	0

For selecting the mating population, the appropriate number of uniformly distributed random numbers (uniform distributed between 0.0 and 1.0) is independently generated, i.e. consider a sample of 5 random numbers: 0.42, 0.86, 0.99, 0.05, 0.78.

Figure 2.6 shows the selection process of the individuals for the example in table together with the above sample trials. After selection, the mating population consists of the individuals: 2, 5, 6, 1, 4. The roulette-wheel selection algorithm provides a zero bias but does not guarantee minimum spread.

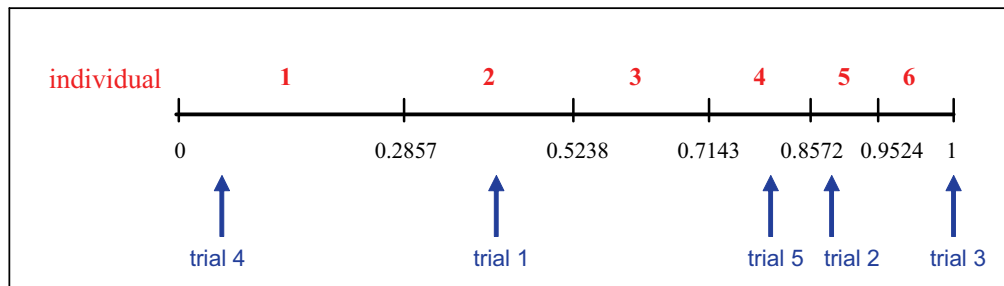


Figure 2.6 Roulette-wheel selection

Stochastic universal sampling

Stochastic universal sampling provides zero bias and minimum spread (Baker, 1987). The individuals are mapped to contiguous segments of a line, such that each individual's segment is equal in size to its fitness exactly as in roulette-wheel selection. Here equally spaced pointers are placed over the line as many as there are individuals to be selected. Consider $NPointer$ to be the number of individuals to be selected. Then, the distance between the pointers are $1/NPointer$ and the position of the first pointer is given by a randomly generated number in the range $[0, 1/NPointer]$.

Using the information from Table 2.2, consider four individuals to be selected. Thus, the distance between the pointers is $1/4=0.25$. Figure 2.7 shows the selection for the above example.

Randomly sample 1 number in the range $[0, 0.25]$: 0.22. After selection the mating population consists of the individuals: 1, 2, 4, 6.

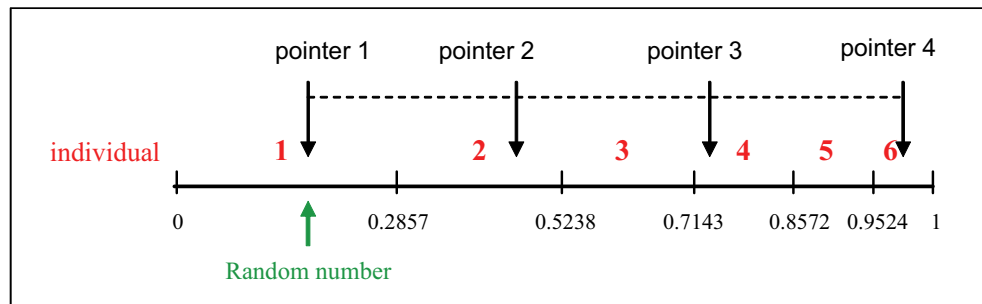


Figure 2.7 Stochastic universal sampling

Rank selection

Rank selection ranks the population first and then every chromosome receives fitness value determined by this ranking. The worst will have the fitness 1, the second worst 2, etc. and the best will have fitness N (number of chromosomes in the population). The fitness assigned to each individual depends only on its position in the individuals rank and not on the actual objective value.

Rank-based fitness assignment overcomes the scaling problems of the proportional fitness assignment. (Stagnation in the case where the selective pressure is too small, or premature convergence where selection has caused the search to narrow down too quickly.) The reproductive range is limited, so that no individuals generate an excessive number of offspring. Ranking introduces a uniform scaling across the population and provides a simple and effective way of controlling selective pressure.

Rank-based fitness assignment behaves in a more robust manner than proportional fitness assignment and, thus, is a good method to choose. (Whitley, 1989 and Bäch & Hoffmeister, 1991).

Tournament selection

Tournament selection involves randomly choosing two candidates from the current population. Then, the fitness of these two candidates are compared and the one with the highest one is selected for mating. Tournament selection can be generalized to include more than two individuals being chosen for competition and the best of this group is selected to reproduce.

To illustrate, consider one round of tournament selection with $k=2$ as in Figure 2.8, in which individuals 132 and 28 are chosen for competition. Individual 28 would then be selected since its fitness (0.125) is larger than the fitness of individual 132 (0.056). Thus, individual 28 would be considered for mating purposes. This process is repeated a specified number of times or until the mating pool is complete.

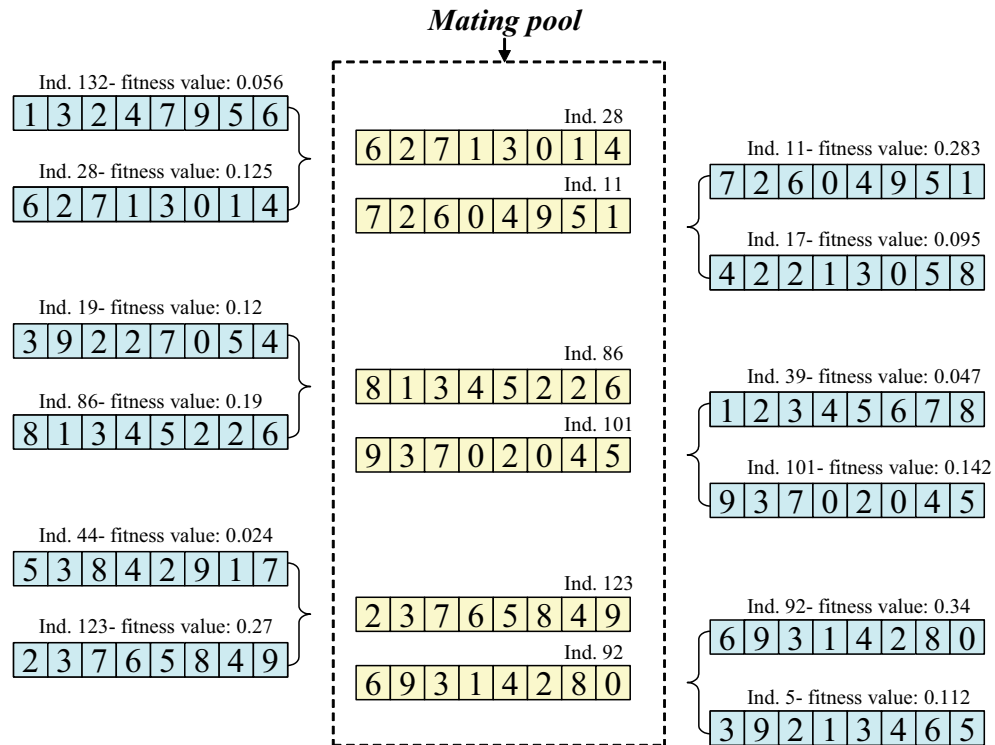


Figure 2.8 Tournament selection, $k=2$

Steady-state selection

The steady-state selection GA works in the following way. In every generation a few good (with higher fitness) chromosomes are selected for creating new offspring. Then some bad (with lower fitness) chromosomes are removed and the new offspring is placed in their place. The rest of population survives to new generation.

2.2.1.3.1.3 Elitism

Elitism is the name of the method that first copies the best chromosome (or few best chromosomes) to the new population. The rest of the population is constructed in ways described above. Elitism can rapidly increase the performance of GA, because it prevents the loss of the best found solutions.

2.2.1.3.1.4 Crossover and mutation

Crossover and mutation are the most important part of a GA. The performance of the algorithm is mainly influenced by these two operators. The primary purpose of the crossover operator is to get genetic material from the previous generation to the subsequent generation, while the main purpose of the mutation operator is to introduce a certain amount of randomness to the search. It can help in the search to find solutions that crossover alone might not encounter.

Usually, there is a predefined probability of procreation via each of these operators. Traditionally, these probability values are selected such that crossover is the most frequently used, with mutation being resorted to only relatively rarely. This is because the mutation operator is a random operator and serves to introduce diversity in the population.

Crossover is made in the hope that new chromosomes contain good parts of old chromosomes, and therefore, the new chromosomes are better. However, it is good to leave some part of the old population to survive to the next generation.

Mutation generally prevents the GA from falling into local extremes. Mutation should not occur very often, because then GA will in fact change to *random search*.

There are many different types of crossover operators presented in the literature. Some of them are: single-point crossover, two-point crossover, uniform crossover, order crossover, position based crossover (Syswerda, 1990), partially mapped crossover (Goldberg & Lingle, 1985), etc. However, the type of encoding in most cases dictates the type of crossover to utilize. Some of the most common crossover operators and type of mutation used in binary, permutation, value, and tree encoded individuals are presented next:

2.2.1.3.1.4.1 Crossover and mutation for binary encoded individuals

For binary encoded chromosomes, there can be distinguished several types of crossover techniques, some of them are:

Single-point crossover: a crossover position is selected at random, then the first child contains the first part of parent 1 and, the rest is copied from the other parent as shown in Figure 2.9.

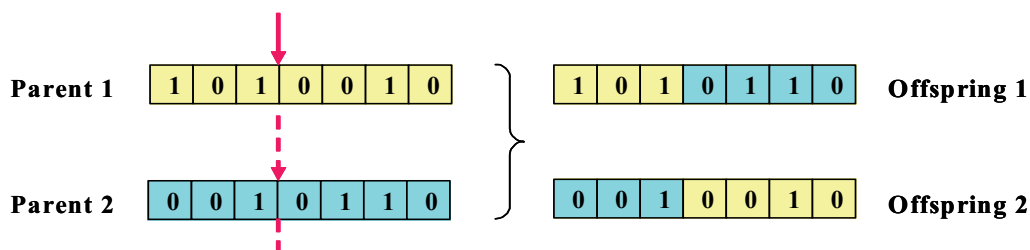


Figure 2.9 Single-point crossover

Double-point crossover: in double-point crossover two crossover positions are selected uniformly at random and the variables exchanged between the individuals between these points, then two new offspring are produced as seen in Figure 2.10.

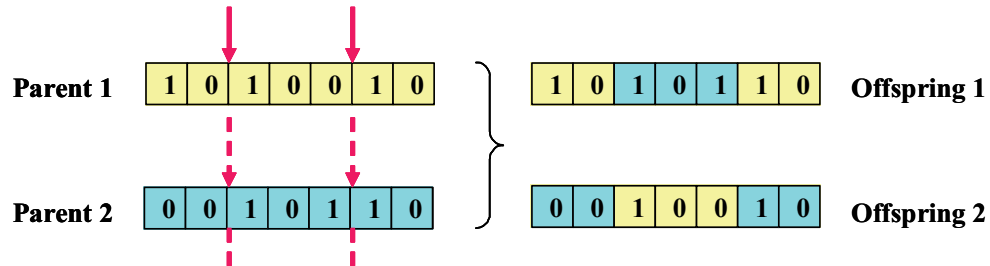


Figure 2.10 Double-point crossover

Multi-point crossover: For multi-point crossover, k crossover positions are selected, then the variables between successive crossover points are exchanged between the two parents to produce two new offspring. Figure 2.11 illustrates an example of multi-point crossover.

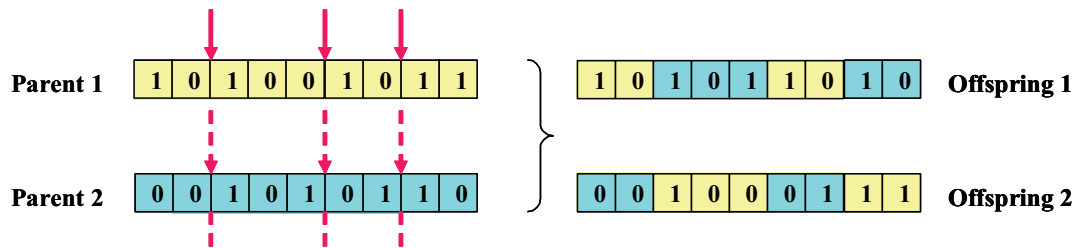


Figure 2.11 Multi-point crossover

The idea behind multi-point, and indeed many of the variations on the crossover operator, is that parts of the chromosome representation that contribute most to the performance of a particular individual may not necessarily be contained in adjacent substrings (Booker, 1987). Furthermore, the disruptive nature of multi-point crossover appears to encourage the exploration of the search space, rather than favoring the convergence to highly fit individuals early in the search, thus making the search more robust (Spears & De Jong, 1991).

Mutation for binary encoded individuals:

Bit inversion is one mutation technique for binary encoded chromosomes.

Bit inversion - Selected bits are inverted (one or more). For binary valued individuals mutation means the flipping of variable values, because every variable has only two states. Thus, the size of the mutation step is always one. For every individual, the variable value to change is chosen (mostly uniform at random). In Figure 2.12, the individual is mutated in position $k=4$.



Figure 2.12 Mutation in binary encoded individuals

2.2.1.3.1.4.2 Crossover and mutation for permutation encoded individuals

Single point crossover and order changing are presented as the crossover and mutation techniques for permutation encoded individuals.

Single point crossover – as shown in Figure 2.13, one crossover point is selected, the permutation is copied from the first parent till the crossover point, then the other parent is scanned and if the number is not yet in the offspring, it is added.

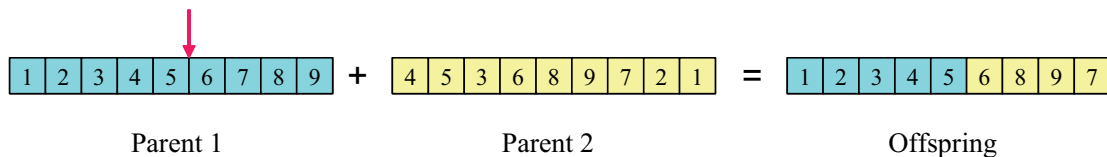


Figure 2.13 Single-point crossover for permutation encoded individuals

Mutation for permutation encoded individuals:

Order changing - Two numbers in the string are selected and exchanged as shown in Figure 2.14.



Figure 2.14 Order changing mutation for permutation encoded individuals

2.2.1.3.1.4.3 Crossover and mutation for value encoded individuals

All crossovers from binary encoding can be used here as well. Figure 2.15 shows an example of double-point crossover for chromosomes encoded with integer variables.

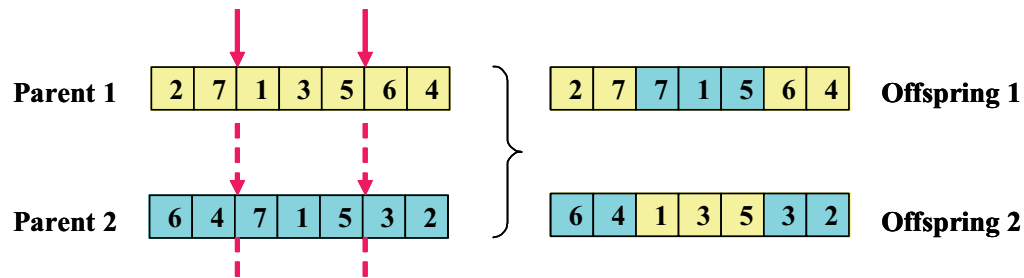


Figure 2.15 Double-point crossover

Mutation for value encoded individuals:

In the case of having integer encoded chromosomes, mutation can be similar to the one for permutation encoded individuals (order changing) if the problem allows this type of change.

For the case of having real encoded individuals, a small number is added to (or subtracted from) selected values with a low probability. Thus, the probability of mutating a variable (mutation rate) and the size of the changes for each mutated variable (mutation step) must be defined. Figure 2.16 shows the case in which randomly selected, positions 3 and 4 of the chromosome are selected to be mutated. For the first selected gene, a small number (0.12) was subtracted from the current value in that gene, and for the second selected gene, the same quantity was added, instead of subtracted, to the current value.

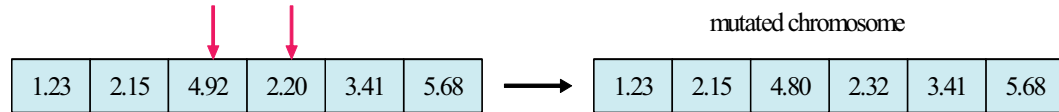


Figure 2.16 Real value mutation

The mutation rate is independent of the size of the population, and the size of the mutation step, is usually difficult to choose. The optimal step-size depends on the problem considered and may even vary during the optimization process. It is known that small steps (small mutation steps) are often successful, especially when the individual is already well adapted. However, larger changes (large mutation steps) can, when successful, produce good results much quicker. Thus, a good mutation operator should often produce small step-sizes with a high probability and large step-sizes with a low probability.

Generally, the probability of mutating a variable is inversely proportional to the number of variables (dimensions). The more dimensions one individual has, the smaller is the mutation probability. Different papers reported results for the optimal mutation rate. In Mühlenbein & Schlierkamp-Voosen (1993), it is reported that a mutation rate of $1/n$ (n = number of variables of an individual) produced good results for a wide variety of test functions. Similar results are reported in (Bäck, 1993) and (Bäck, 1996) for a binary valued representation.

2.2.1.3.1.4.4 Crossover and mutation for tree encoded individuals

Tree crossover – One crossover point is selected in both parents, and parents are divided at that point and the parts below the crossover points are exchanged to produce new offspring.

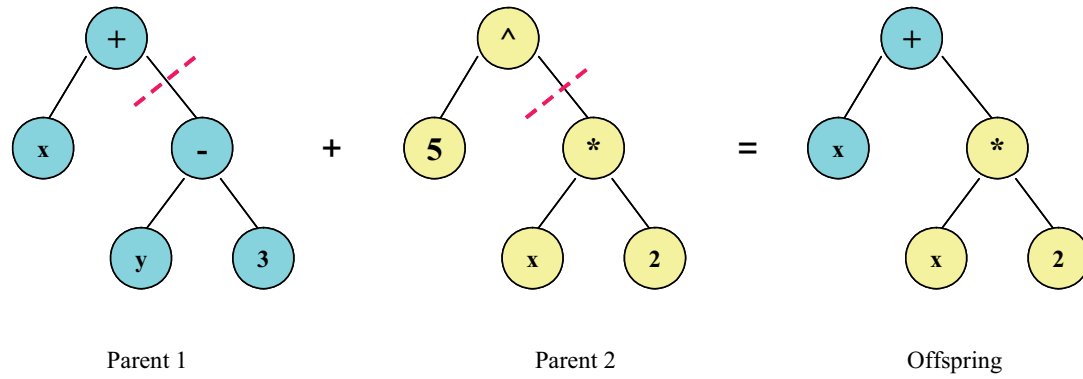


Figure 2.17 Crossover for tree encoded individuals

Mutation for tree encoded individuals:

Changing operator (number) mutation – in this type of mutation, some nodes are selected to be changed.

2.2.1.3.1.5 Optimal crossover and mutation rates in genetic search

There are no universally accepted general rules to choose the values of basic GA operators for solving specific optimization problems. The best way to determine the proper combination of these values is by experimental comparison between GAs with different parameters (Lisnianski & Levitin, 2003).

However, numerous experimental studies have developed some rules of thumb concerning ranges of GA parameters. For example, De Jong (1975) suggests that the mutation probability, which is a bit reversal event, should occur with small probability, $p_{mut} \approx 0.001$. Grefenstette (1986) suggests a $p_{mut} \approx 0.01$, while in Schaffer *et al.* (1989) a range is considered, $p_{mut} \in [0.005, 0.01]$. Analogously, for the crossover rate, De Jong (1975) suggests that the crossover should occur with probability, $p_{cross} \approx 0.6$. Grefenstette (1986) suggests a $p_{cross} \approx 0.95$, while in Schaffer *et al.* (1989) a range is again considered $p_{cross} \in [0.75, 0.95]$.

2.2.1.3.1.6 Reinsertion

When less offspring are produced than the size of the original population, then to maintain the size of the original population, the offspring have to be reinserted into the old population. Similarly, if not all offspring are to be used at each generation or if more offspring are generated than the size of the old population, then a reinsertion scheme must be used to determine which individuals are to exist in the new population. There are different schemes of global reinsertion:

- Pure reinsertion: Produce as many offspring as parents and replace all parents by the offspring.
- Uniform reinsertion: Produce less offspring than parents and replace parents uniformly at random.
- Elitist reinsertion: Produce less offspring than parents and replace the worst parents.
- Fitness-based reinsertion: Produce more offspring than needed for reinsertion and reinsert only the best offspring.

Pure reinsertion is the simplest reinsertion scheme. Every individual lives one generation only. This scheme is used in the simple genetic algorithm. However, it is very likely, that very good individuals are replaced without producing better offspring, and thus, good information is lost.

2.2.2 Trajectory methods

2.2.2.1 Simulated annealing (SA)

Simulated Annealing (SA) is commonly said to be the oldest among the metaheuristics and surely one of the first algorithms that had an explicit strategy to escape from local minima. The origins of the algorithm are in statistical mechanics

(Metropolis algorithm) and it was first presented as a search algorithm for CO problems in Kirkpatrick *et al.* (1983) and Cerny (1985). The fundamental idea is that improving candidate solutions are always accepted while non-improving solutions are accepted with a certain probability. The probability of accepting non-improving solutions is calculated according to the current temperature of the algorithm.

This process is analogous to the annealing process of metals and glass, which assume a low energy configuration when cooled with an appropriate cooling schedule. Regarding the search process, this means that the algorithm is the result of two combined strategies: random walk and iterative improvement.

The algorithm starts by generating an initial solution (either randomly or heuristically constructed) and with a high initial temperature, T , which corresponds to a high probability of accepting non-improving solutions. The temperature is gradually decreased as the search progresses so that the probability of accepting non-improving solutions is also reduced. At temperature zero, T_0 , the algorithm operates like an improving heuristic, i.e., only improving solutions are accepted. The pseudo-code of the SA metaheuristic is shown below:

1. Generate initial solution x
2. Set initial temperature
3. Generate candidate solution x' from current solution x
4. If $\text{fitness}(x') > \text{fitness}(x)$, then $x = x'$
5. If $\text{fitness}(x') \leq \text{fitness}(x)$, then calculate Acceptance Probability
 - 5.1 If $\text{Acceptance Probability} > \text{random}[0,1]$ then $x = x'$
6. Update temperature according to cooling schedule

7. If stopping condition is reached, stop, otherwise go to step 3.

The choice of an appropriate cooling schedule is crucial for the performance of the algorithm. The cooling schedule defines the value of T at each iteration k . The cooling schedule and the initial temperature should be adapted to the particular problem instance, since the cost of escaping from local minima depends on the structure of the search landscape. A simple way of empirically determining the starting temperature T_0 is to initially sample the search space with a random walk to roughly evaluate the average and the variance of objective function values.

Generally, SA can find good solutions for a wide variety of problems, it is easy to implement and is capable of handling almost any optimization problem and any constraint. However, some of the difficulties reported with this method are long run times. SA is nowadays used as a component in more advanced metaheuristics, rather than applied as stand-alone search algorithm.

2.2.2.2 Tabu search (TS)

Tabu search is in many ways similar to simulated annealing: they both move from one solution to another with the next solution being possibly worse than the one before. However, the basic difference between SA and TS lies in the mechanism used for approving a candidate solution. In TS the mechanism is not probabilistic, but rather of a deterministic nature (Pinedo & Chao, 1999).

Generally speaking, TS is a meta-heuristic that guides a local heuristic search strategy to explore the solution space beyond local optimality. It was originally proposed by Glover (1989, 1990).

The local procedure is a search that uses an operation called a move to define the neighborhood of any given solution. The neighborhood of the current solution is explored and the best solution is selected as the new current solution. The best solution in the neighborhood is selected, even if it is worse than the current solution.

In TS the *Tabu List* plays an important role. It keeps track of previously explored solutions and prohibits TS from revisiting them again. In this way, TS can overcome local minima by forcing the acceptance of solutions worse than the current solution.

The general framework of TS consists of several steps which are shown below:

1. *Initialization*: a starting solution s is generated by choosing random values for \mathbf{x} . This solution is evaluated by the evaluation function, and solution s is stored in the algorithm's memory. This memory is called the *Tabu List*.
2. *Neighborhood exploration*: all possible neighbors of solution s are generated and evaluated. Neighboring solutions are solutions which can be reached from the current solution by a simple, basic transformation of the current solution. Solutions which are present in the *Tabu List* are considered unreachable neighbors.
3. *New current solution*: a new current solution is chosen from the explored neighborhood. This solution cannot be in the *Tabu List* and has to have the best evaluation value from all reachable neighbors. The evaluation value can be worse compared with the current solution. In this way the algorithm is able to overcome local minima. The new current solution is added to the *Tabu List*.

4. *Stop*: if no more neighbors are present (all are tabu) or a certain evaluation value or a predetermined number of iterations is reached, the algorithm stops, otherwise the algorithm continues with step 2.

The flexibility and variety of principles that are incorporated in the Tabu Search have made this Metaheuristic very appealing to the research community. A wide range of combinatorial problems have been solved using TS (Glover & Laguna, 1997). Since Tabu Search can be conceptualized as a framework rather than a method, many of its components can be designed specifically for target applications just by following its principles. Therefore, recently, more powerful versions of TS have been proposed, these versions retain more information.

2.3 Summary

Metaheuristics can be loosely defined as high level strategies for exploring search spaces by using different methods.

This section provided a brief **overview** of some of the most well-known **metaheuristic algorithms** and their classification. Also, a **general introduction to single objective GAs** was given, as well as a reasonable **description of their design issues**.

There are several different philosophies apparent in the existing metaheuristics. Some of them can be seen as “intelligent” extensions of local search algorithms. The goal of this kind of metaheuristic is to escape from local minima in order to proceed in the exploration of the search space and to move on to find other hopefully better local minima. This is, for example, the case in **Tabu Search** and **Simulated Annealing**. These

metaheuristics (also called trajectory methods) work on one or several neighborhood structure(s) imposed on the members (the solutions) of the search space.

We can find a different philosophy in algorithms like **Ant Colony Optimization** and **Evolutionary Computation**. They incorporate a learning component in the sense that they implicitly or explicitly try to learn correlations between decision variables to identify high quality areas in the search space. In Evolutionary Computation algorithms, a population of individuals is modified by recombination and mutation operators, and in Ant Colony Optimization a colony of artificial ants is used to construct solutions guided by the pheromone trails and heuristic information.

The most recognized form of evolutionary computation algorithms are **Genetic Algorithms** (GAs) which in general, can be more efficient and out-perform gradient search methods if your search space has many local optima. Since the genetic algorithm traverses the search space using the genotype rather than the phenotype, it is less likely to prematurely converge to a local high or low.

Although GAs use the idea of randomness when performing a search, it must be clearly understood that GAs are not simply random search algorithms. Random search algorithms can be inherently inefficient due to the directionless nature of the search. The GAs are not directionless. They utilize knowledge from previous generations of strings in order to construct new strings that will approach the optimal solution. Thus, GAs are a form of a randomized search and the way that the strings are chosen and combined comprise a stochastic process (Lisnianski & Levitin, 2003).

In Chapter 4, a multi-purpose multiple objective evolutionary algorithm (NSGA-II) is used to solve two-well known multiple objective problems and, in Chapters 5 through 8,

new multiple objective evolutionary algorithms to solve different multiple objective optimization problems are developed.

3. Multi-objective optimization

In this chapter, the two primary approaches to identify solution(s) to multiple objective problems are reviewed. The first approach involves determining the relative importance of the attributes, and aggregating the attributes into some kind of overall composite objective function; while the second approach involves populating a number of feasible solutions along a Pareto frontier and the final solution is a set of non-dominated solutions. Multi-objective evolutionary algorithms (MOEAs) are the most notable methods of this second approach.

In Chapter 2, the basic design issues in single-objective GAs were introduced, and thus, some of the aspects that make single-objective genetic algorithms different from multiple-objective genetic algorithms are also reviewed. However, due to their importance, most of the effort is concentrated to present an extensive overview of some of the currently state-of-the-art MOEAs.

3.1 Multi-objective optimization solution methods

Although there are several approaches to solve multi-objective problems, the two most common are: 1) combine them into a single objective function such as the weighted sum method, goal programming or utility functions and apply methods for single objective optimization, or 2) obtain a set of non-dominated Pareto-optimal solutions. For the first approach, a single “optimal” solution is generally found, whereas in the second approach, a potentially large Pareto-optimal set is identified.

3.1.1 Single-objective approaches

The presence of several conflicting objectives is typical for engineering problems. The most common approach for multi-objective optimization is by aggregating the different objectives into one composite objective function. Optimization is then conducted with one optimal solution as the result. The weighted sum method, goal programming, utility theory, etc., are examples of this approach.

The *weighted sum method* consists of combining all the objective functions together using different weighting coefficients for each one. This method is the simplest possible approach to solve the multi-objective problem, but the challenge with this approach is determining the appropriate set of weights when the user does not have enough information about the problem or has only an intuition of the importance of one objective over the other. In practice, it is difficult to establish a relationship between these weights and the real outcome in terms of objective functions values.

Goal programming deals with the achievement of prescribed goals or targets. In this method, the user has to assign targets or goals that he/she wishes to achieve for each objective. This technique yields a dominated solution if the goal point is chosen in the feasible domain. However, the decision-maker must devise the appropriate weights for the objectives. This can also be a difficult task in many cases, unless there is prior knowledge about the shape of the search space, the relative importance of the objectives and meaningful goals.

For modeling designer's preference structure, one of the commonly used methods is based on the *utility theory* (Keeney and Raifa, 1976). A utility or value function combines all objectives into one composite function, and then any appropriate single

objective function method can be used. Although utility functions offer the ideal way to solve a multiple objective problem (Steuer 1989), one difficulty associated with using the utility function approach is that, in practice, no precise approach exists to obtain the mathematical representation of the decision-maker's true preference or utility function in a multi-objective setting. This can be problematic for the non-specialist.

3.1.2 Multiple objective evolutionary algorithms (MOEAs)

Evolutionary algorithms are *the* standard tool for many multi-objective optimization problems. Their parallel search leads to an approximation of the Pareto front in a single optimization run. This is a major advantage compared to traditional optimization algorithms like gradient-based methods that converge to a single Pareto solution. Furthermore, traditional methods require an aggregation of all objectives to a single objective. This is difficult if the shape of the Pareto front is unknown before optimization.

Evolutionary algorithms can exploit the population-based feature and converge in parallel to the Pareto front. While optimizing, different solutions in the population converge to different areas of the Pareto front, and thus an approximation of the Pareto front can be obtained in a single optimization run. The research interest has increased over the past twenty years on the development and application of evolutionary algorithms for Pareto optimization. Several promising methods have been proposed and compared by several researchers, e.g.:

- VEGA (vector evaluated genetic algorithm) by Shaffer (1985)
- MOGA (multi-objective genetic algorithm) by Fonseca and Flemming (1993)
- NPGA (niched-Pareto genetic algorithm) by Horn *et al.* (1994).

- NSGA (nondominated sorting genetic algorithm) developed by Srinivas & Deb (1995).
- SPEA (strength Pareto evolutionary algorithm) by Zitzler & Thiele (1999).
- NSGA-II by Deb *et al.* (2002).
- PAES (Pareto-Archived Evolutionary Strategy) by Knowles & Corne (2000).

In general, MOEAs are suited to multi-objective optimization because they are able to capture multiple Pareto-optimal solutions in a single simulation run and may exploit similarities of solutions by recombination. Summaries and comparisons of different MOEAs are described by Konak *et al.* (2006).

3.1.3 Differences between MOGAs and single GAs

In single-objective GAs, individual performance, as measured by the objective function and individual fitness, are so closely related that the objective function is sometimes referred to as the fitness function. The two are, however, not the same. In fact, whereas the objective function characterizes the problem and cannot be changed at will, assigned fitness is a direct measure of individual reproductive ability, forming an integral part of the GA search strategy.

Generally, multi-objective evolutionary algorithms use standard genetic operators as described in Section 2 and the differences between these algorithms concentrates on the strategies used for selection and diversification. The alternative approaches employ neighborhood search, which needs to be specifically designed according to the problem.

One of the desirable characteristics that all MOGAs try to achieve is diversity. Diversity is the term used to describe the relative uniqueness of each individual in the population. Typically diversity refers to genetic variation, such that evenly-spaced

solutions in the final Pareto front are obtained. Figures 3.1 and 3.2 illustrate the cases in which diversity of solutions is achieved and not achieved, respectively.

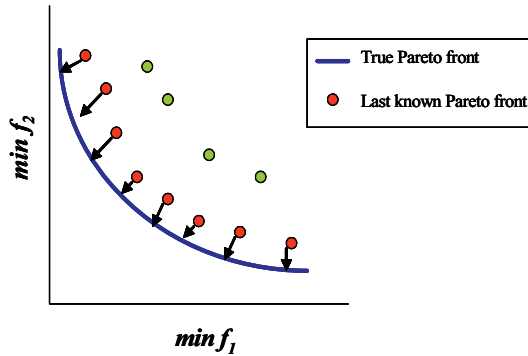


Figure 3.1 Population diversity is achieved

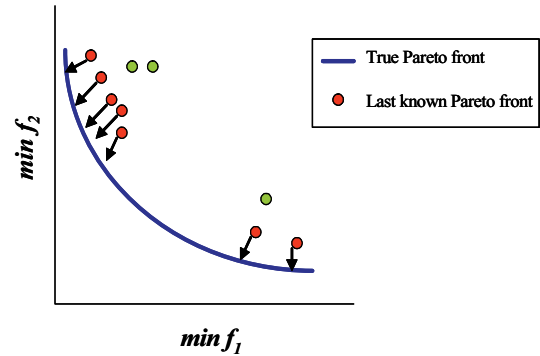


Figure 3.2 Population diversity is not achieved

Diversity is considered favorable as the greater the variety of genes available to the genetic algorithm the greater the likelihood of the system identifying alternate solutions. Moreover, maintaining diversity of individuals within a population is necessary for the long term success of any evolutionary system. Genetic diversity helps a population adapt quickly to changes in the environment, and it allows the population to continue searching for productive niches (neighborhoods), avoiding becoming trapped at local optima.

3.2 State-of-the-art multi-objective evolutionary algorithms

Hertz & Klobner (2000) state that there is not a clear and widely accepted definition of evolutionary algorithms. However, they suggest that in a strict sense, an evolutionary algorithm involves a population of solutions, evolves this population by means of cooperation (recombination) and self-adaptation (mutation), and uses a coded representation of the solutions.

A number of different MOEAs have been proposed in recent years and the increasing interest on these methods has motivated the extension of evolutionary algorithms

originally proposed for single-objective optimization to multi-objective variants. Some of these MOEAs are described next.

3.2.1 Vector Evaluated Genetic Algorithm (VEGA) [Schaffer 1984, 1985]

VEGA is perhaps the first genetic algorithm in which the concept of dominance was implemented for the evaluation and selection of individuals. The name of the algorithm results from the optimization of a vector of objectives instead of a scalar in single objective optimization. The VEGA algorithm divides the population into k subpopulations according to k objective functions. The individuals in each subpopulation are assigned a fitness value based on the corresponding objective function. In this algorithm, selection is done for each of the k objectives separately, filling equally sized portions of the mating pool. Afterwards, the mating pool is shuffled, and crossover and mutation are performed.

A drawback of this algorithm is that it tends to bias selection in favor to those individuals at the extreme (that solely minimize/maximize one objective), and thus, the algorithm fails to sustain diversity among the Pareto-optimal solutions and converges near one of the individual solutions. Figure 3.3 shows the VEGA selection mechanism considering two objective functions.

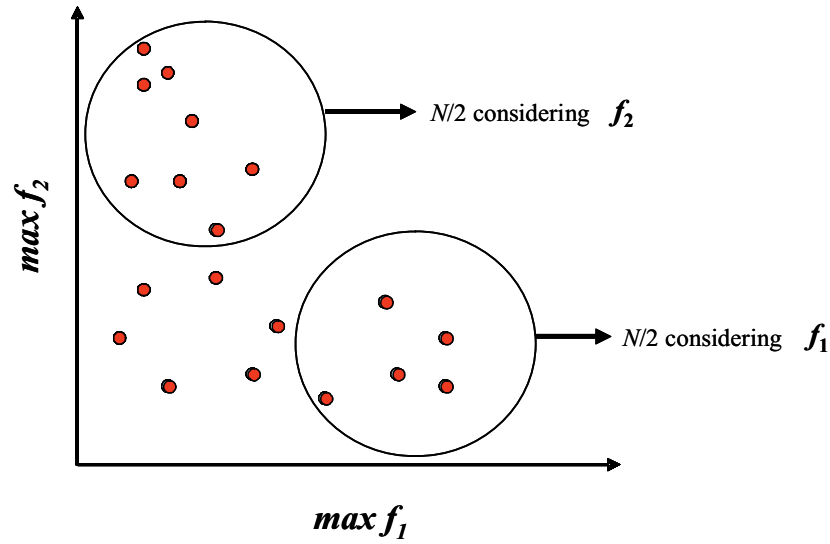


Figure 3.3 The VEGA selection mechanism

3.2.2 Multi-objective Genetic Algorithm (MOGA) [Fonseca & Fleming 1993]

In the Fonseca & Fleming MOGA, each individual is ranked according to their degree of dominance. The more population members that dominate an individual, the higher the ranking for the individual. An individual's ranking equals the number of individuals that it is dominated by plus one (as in Figure 3.4). Thus, individuals on the Pareto front have a ranking of one, as they are non-dominated. The rankings are then scaled to score individuals in the population. The fitness is assigned to each individual using an interpolation between the best and the worst rank. A scheme for niche (neighborhood) formation is used in which fitness in the objective domain is shared among non-dominated individuals in order to maintain a uniform distribution of individuals over the trade-off surface. The fitness of all individuals in the same rank is averaged and this value is assigned to all of them. A summary of MOGA as presented in Coello Coello *et al.* (2002) is presented in Figure 3.5.

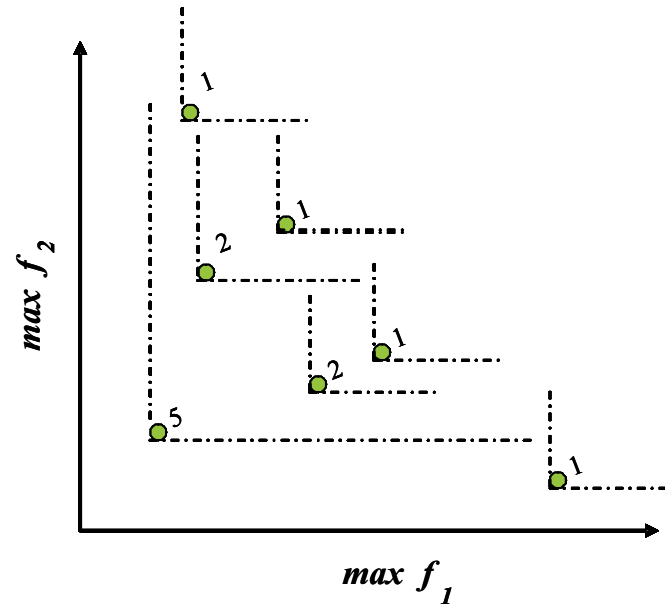


Figure 3.4 Population ranking according to MOGA

1. Initialize Population
2. Evaluate objective values
3. Assign rank based on Pareto dominance
4. Compute Niche count
5. Assign linearly scaled fitness
6. Assign Shared fitness
7. For $i=1$ to G
 - Selection via stochastic universal sampling
 - Single point crossover
 - Mutation
 - Evaluate objective values
 - Assign rank based on Pareto dominance
 - Compute niche count
 - Assign linearly scaled fitness
 - Assign shared fitness
8. End loop

Figure 3.5 MOGA overview

3.2.3 Niche Pareto Genetic Algorithm (NPGA) [Horn *et al.* 1994]

The Niche Pareto Genetic Algorithm (NPGA) (Horn *et al.*, 1994; Horn, 1997) uses the concept of Pareto dominance and tournament selection to solve multiple objective optimization problems. This was one of the first algorithms to directly address the

diversity of the approximation set. The main difference between NPGA and traditional GAs is localized in the selection mechanism. In this algorithm the selection of individuals is conducted using a Pareto domination tournament selection in conjunction with fitness sharing to maintain a diverse population.

Pareto domination tournaments are binary tournaments in which the domination of each candidate is assessed with respect to a randomly chosen sample, T_{dom} , typically 10% of the population. The two individuals competing for selection are compared against this subset, T_{dom} , of the population, and if one of the competing individuals is dominated by any member of the set and the other is not, then the latter is chosen as winner of the tournament. If both individuals are dominated (or not dominated), the result of the tournament is decided by sharing: i.e., the individual that has the least individuals in its niche (defined by σ_{share}) is selected for reproduction.

In Figure 3.6, individuals in a niche “share” the niche fitness and Figure 3.7 presents a summary of NPGA as presented in Coello Coello *et al.* (2002). NPGA has been shown to be inferior to most of the more recent MOEAs. A comparison by Zitzler *et al.* (2000) ranked NPGA fifth out of six considered MOEAs.

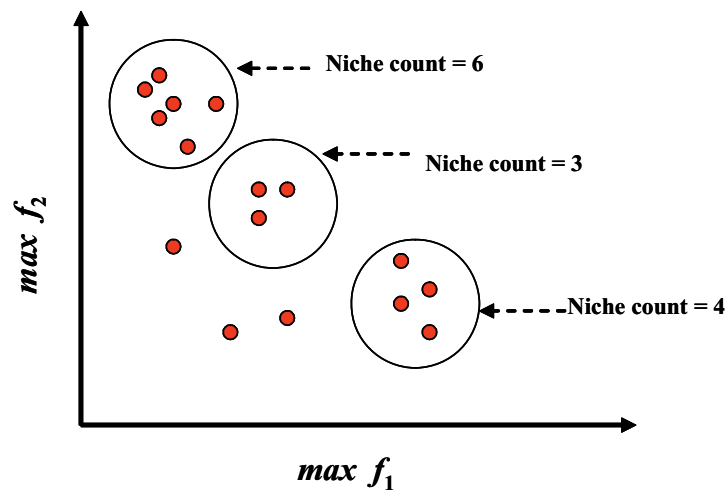


Figure 3.6 Niche count in NPGA

1. Initialize Population
2. Evaluate objective values
3. For $i = 1$ to G
 - Specialized Binary Tournament selection
 - Only Candidate 1 Dominated: Select Candidate 2
 - Only Candidate 2 Dominated: Select Candidate 1
 - Both Candidates Dominated or Both Not Dominated:
 - Perform Specialized fitness Sharing
 - Return Candidate with Lower Niche count
 - Single point crossover
 - Mutation
 - Evaluate objective values
4. End loop

Figure 3.7 NPGA overview

An improved version of NPGA, called NPGA II, was presented by Erickson *et al.* (2001). They use Pareto ranking but keep tournament selection. Niche counts in NPGA II are calculated using individuals in the partially filled next generation which is known as continuously updated fitness sharing, and was proposed by Oei *et al.* (1991). Figure 3.8 presents a summary of NPGA 2, as in Coello Coello *et al.* (2002).

1. Initialize Population
2. Evaluate objective values
3. For $i = 1$ to G
 - Specialized Binary Tournament Selection
 - Using Degree of Domination as Rank
 - Only Candidate 1 Dominated: Select Candidate 2
 - Only Candidate 2 Dominated: Select Candidate 1
 - Both Candidates Dominated or Both Not Dominated:
 - Perform Specialized fitness Sharing
 - Return Candidate with Lower Niche count
 - Single point crossover
 - Mutation
 - Evaluate objective values
4. End loop

Figure 3.8 NPGA 2 overview

3.2.4 Nondominated Sorting Genetic Algorithm (NSGA) [Srinivas & Deb 1995]

NSGA also classifies individuals according to dominance in a ranking scheme similar to the one used in by Fonseca & Flemming (1993). NSGA was proposed by Srinivas & Deb (1995). The idea behind NSGA is that a ranking selection method is used to emphasize good solutions and a niche method is used to maintain stable subpopulations. While it follows the standard GA for parent selection and offspring generation, it varies in the manner in which the selection operator works. In NSGA the fitness of the individuals is determined by using the concept of Pareto dominance as follows. At the beginning of the search, an initial population is created, and then, the non-dominated individuals in the current population are identified. All of these non-dominated solutions belong to the first rank and the same high fitness value is assigned to them to ensure that they have equal reproductive potential. Figure 3.9 depicts the nondominated fronts according to NSGA for a two-objective problem to minimize f_1 and to maximize f_2 .

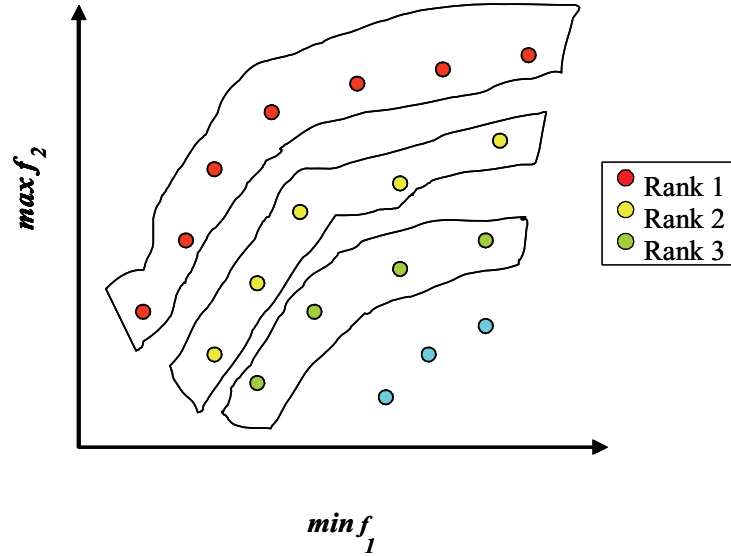


Figure 3.9 Nondominated fronts according to NSGA

To maintain diversity, solutions found in the first front, undergo a fitness sharing procedure. Fitness sharing encourages the search in unexplored sections of a Pareto front. Sharing is achieved by performing selection operation using degraded fitness values obtained by dividing the original fitness value of an individual by a quantity proportional to the number of individuals around it. After sharing their fitness value, the individuals are temporarily ignored, and the rest of the population is processed in the same way to identify a new set of non-dominated individuals. A fitness value that is smaller than the previous one is assigned to all the individuals belonging to the second non-dominated front. This process continues until the whole population is classified into non-dominated fronts with different fitness values.

Once fitness has been assigned, the population is reproduced according to the fitness values. Since individuals in the first front have the maximum fitness value, they receive more copies than the rest of the population. The efficiency of NSGA mainly is due to the way multiple objectives are reduced to a dummy fitness function using non-dominated sorting procedures.

The parameter, σ_{share} , can be calculated as follows, as in Deb & Goldberg (1989).

$$\sigma_{share} = \frac{0.5}{\sqrt[p]{q}}$$

Where q is the desired number of distinct Pareto-optimal solutions and p is the number of decision variables. A summary of the NSGA algorithm is shown in Figure 3.10.

1. Initialize population
2. Evaluate objective values
3. Assign rank based on Pareto dominance
4. Compute niche count
5. Assign shared fitness
6. For $i=1$ to G
 - Selection via stochastic universal sampling
 - Single point crossover
 - Mutation
 - Evaluate objective values
 - Assign rank based on Pareto dominance
 - Compute Niche count
 - Assign shared fitness
7. End loop

Figure 3.10 NSGA overview

3.2.5 Strength Pareto Evolutionary Algorithm (SPEA) [Zitzler & Thiele 1998, 1999]

SPEA was proposed as an approach to incorporate several of the desirable features of other multi-objective evolutionary algorithms. SPEA uses two populations, P and P' . Throughout the process, copies of all non-dominated individuals are stored in P' . Each individual is given a fitness value, f_i , based on Pareto dominance. The fitness of the members of P' is calculated as a function of how many individuals in P they dominate. That is, each solution i in P' is assigned a real value $s_i \in [0,1)$, called strength. Let n denote the number of individuals in P that are dominated by i and assume N is the size of P . Then s_i is defined by:

$$s_i = \frac{n}{N + 1}$$

Thus, the fitness f_i of i is equal to its strength i.e., $f_i = s_i$. Then, those individuals in P' or those that are nondominated are ranked as indicated in Figure 3.11. A summary of the SPEA algorithm is shown in Figure 3.12.

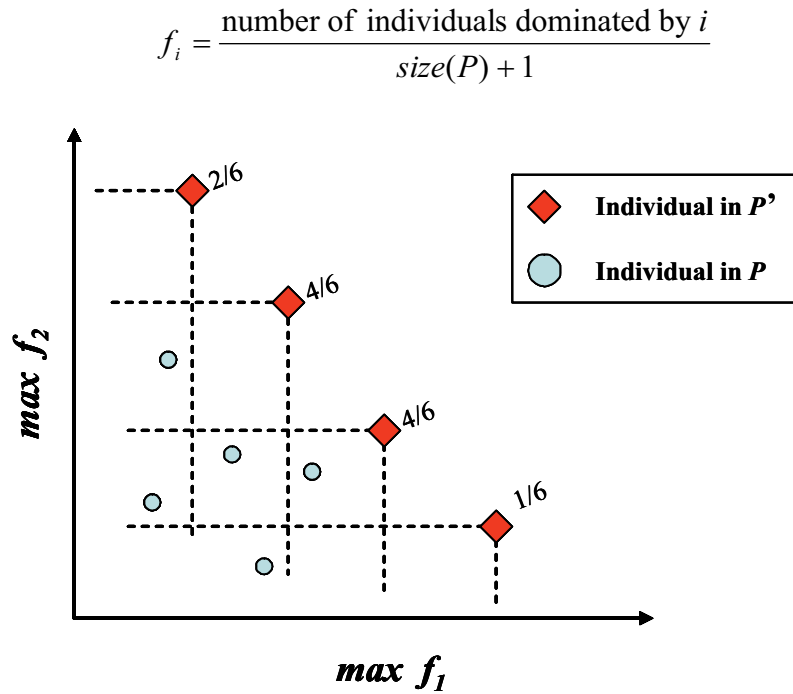


Figure 3.11 Population ranking according to SPEA

1. Initialize Population P
2. Create empty external set P'
3. For $i = 1$ to G
 - Copy nondominated members of P to P'
 - Remove elements from P' which are covered by any other member of P'
 - prune P' (using clustering) when the maximum capacity of P' has been exceeded
 - Compute fitness of each individual in P and in P'
 - Use binary tournament selection with replacement to select individuals from $P+P'$ (multiset union) until the mating pool is full
 - Apply Crossover and Mutation
4. End loop

Figure 3.12 SPEA overview

The improved version of this technique, called SPEA2 was proposed by Zitzler *et al.* (2001). The main differences of SPEA2 in comparison to SPEA are:

- An improved fitness assignment scheme is used, which takes into account for each individual how many other individuals it dominates and it is dominated by.
- A nearest neighbor density estimation technique is incorporated which allows a more precise guidance of the search process.
- A new archive truncation method to guarantee the preservation of boundary solutions.

3.2.6 Fast Elitist Nondominated Sorting Genetic Algorithm (NSGA-II) [Deb *et al.*, 2002]

The NSGA-II algorithm is an improved version of the NSGA. This algorithm does not have the problems of using the sharing function method, including the appropriate selection of the sharing parameter σ_{share} .

In NSGA-II, the selection of individuals is performed as follows: First, an offspring population Q_t is created by using the parent population P_t . However, instead of finding the nondominated front of Q_t only, the two populations are first combined together to form R_t of size $2N$. Then, a nondominated sorting is used to classify the entire population R_t . Although this requires more effort compared to performing a nondominated sorting on Q_t alone, it allows a global nondomination check among the offspring and parent solutions. Once the nondominated sorting is over, the new population is filled by solutions of different nondominated fronts, one at a time.

The filling starts with the best nondominated front and continues with solutions of the second nondominated front, to be followed by the third nondominated front and so on.

After all solutions have been assigned a rank based on the nondomination criterion, a niching strategy, called crowding distance, is employed to estimate the distance between the closest two members for each solution.

The crowding distance parameter that is incorporated in this algorithm serves as an estimate of the perimeter of the cuboids formed by using the nearest neighbors as the vertices. Figure 3.13 shows how the crowding distance of an individual is calculated. It is performed by obtaining the average Euclidean distance of two points in either side of the point in question along each of the objectives.

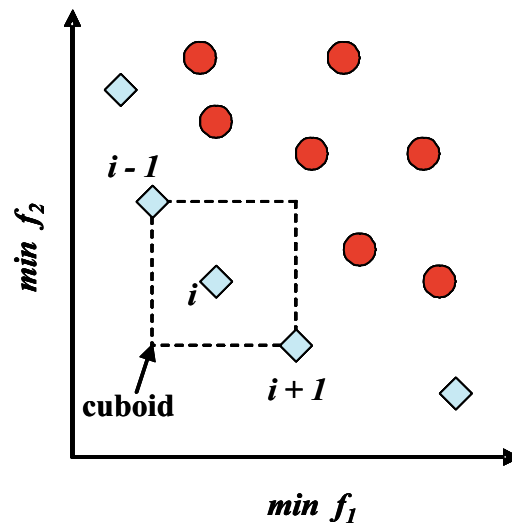


Figure 3.13 Crowding distance calculation

NSGA-II uses the concept of controlled elitism to tune the mutation rate and the elitism rate to attain equilibrium between the two. Controlled elitism limits the maximum number of individuals in the population. This algorithm is efficient in obtaining good Pareto-optimal fronts for any number of objectives and can accommodate any number of constraints as well. A schematic representation of the NSGA-II algorithm is shown in Figure 3.14, and a general summary for the algorithm is shown in Figure 3.15.

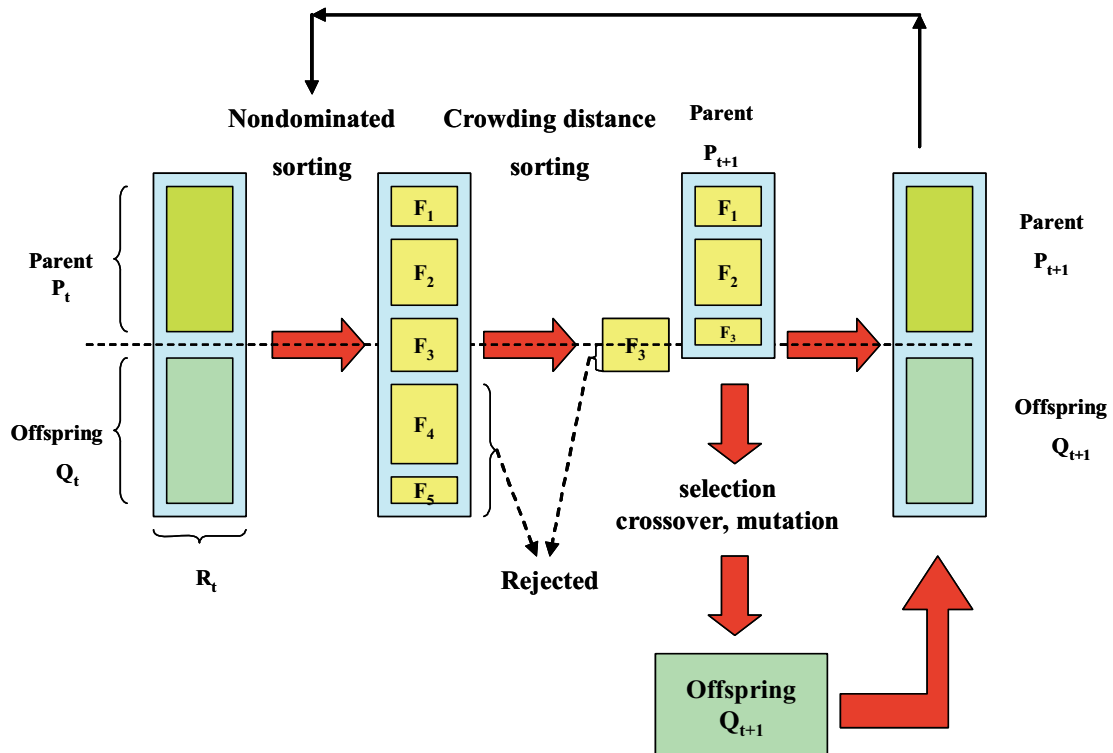


Figure 3.14 Working mechanism of the NSGA-II algorithm

1. Initialize Population P
2. Generate random population – size M
3. Evaluate objective values
4. Assign rank based on Pareto dominance – “sort”
5. Generate child population
 - Binary tournament selection
 - Recombination and Mutation
6. For $i = 1$ to G
 - With parent and child population
 - Assign rank based on Pareto dominance – “sort” starting from the first front until M individual found
 - Determine crowding distance between points on each front
 - Select points (elitism) on the lower front (with lower rank) and that are outside the crowding distance
 - Create next generation
 - Binary tournament selection
 - Recombination and Mutation
 - Increment generation index
7. End loop

Figure 3.15 NSGA-II overview

3.2.7 Pareto-Archived Evolutionary Strategy (PAES) [Knowles & Corne, 1999; 2000]

The algorithm PAES was proposed by Knowles & Corne (1999, 2000). PAES is a multi-objective optimizer which uses $(\mu \text{ “population size”} + \lambda \text{ “number of solutions per generation”})$ local search evolution strategy. PAES has three variants, which are $(1+1)$ -PAES, $(1+\lambda)$ -PAES and $(\mu+\lambda)$ -PAES.

The algorithm in its simplest form is a $(1+1)$ evolution strategy employing local search but using a reference archive of previously found solutions in order to identify the approximate dominance ranking of the current candidate solution vectors. Nonetheless, it is capable of finding diverse solutions in the Pareto optimal set because it maintains an archive of non-dominated solutions which it exploits to accurately estimate the quality of new candidate solutions.

At any iteration t , a candidate solution c_t and a mutated solution m_t must be compared for dominance. Acceptance is simple if one solution dominates the other. If neither solution dominates the other, the new candidate solution is compared with the reference population of previously archived non-dominated solutions. If the comparison fails to favor one solution over the other, the chosen solution is the one which resides in the least crowded region of the space. The PAES algorithm has three main parts; the candidate solution generator, the candidate solution acceptance and the non-dominated solutions archive. Figure 3.16 presents a summary of PAES algorithm.

1. Initialize Single Population parent c and add to Archive
2. Mutate c to produce child m and evaluate fitness
 - If (c dominates m) discard m
 - else if (m dominates c): replace c with m , and add m to Archive
 - else if (m is dominated by any member of the Archive) discard m
 - else apply test ($c, m, \text{Archive}$) to determine which becomes the new current solution and whether to add m to the Archive
3. Until a termination criterion has been reached, return to 2

Figure 3.16 PAES pseudocode

The main feature of PAES is the use of an adaptive grid on which the objective function space is located using a coordinate system. Such a grid is the diversity maintenance mechanism of PAES and it is the main feature of this algorithm. The grid is created by bisecting k times the function space of dimension $d = g + 1$. The control of 2^{kd} grid cells means the allocation of a large amount of physical memory for even small problems. For instance, 10 functions and 5 bisections of the space produce 250 cells.

3.3 Summary

A description of the two primary approaches to identify solution(s) to this particular type of problems was briefly introduced in this section. However, this chapter was more devoted to present the **working mechanism of several state-of-the-art multi-objective evolutionary algorithms**.

The MOEAs described in this chapter are just a sample of the vast number of algorithms proposed in the literature in recent years. Other approaches not discussed in this section include the multi-objective messy genetic algorithm (MOMGA) developed by Van Valduizen & Lamont (2000b). A revised extension of MOMGA (called

MOMGA-II) has been proposed by Zydallis *et al.* (2001). Essentially, in the recent years, many other extensions of evolutionary algorithms for multi-objective optimization have been proposed. For example, variants of micro-genetic algorithms, cellular genetic algorithms, particle swarm optimization methods, agent-based algorithms among others. For a more detailed review of the principles of evolutionary multi-objective optimization and recent developments in this field the reader may refer to Coello Coello *et al.* (2002) and Van Valduizen & Lamont (2000a).

In this thesis, several MOEAs are used and some developed to solve engineering optimization problems and we extend MOEAs to achieve balance between single solutions and Pareto-optimal solutions. In next chapter, two new approaches are presented which offer distinct benefits, which are pruning by using data clustering and the non-numerical ranking preferences method.

4. Post-Pareto optimality

For multi-objective optimization problems, there are two primary approaches to identify solution(s) to these particular types of problems. The first involves determining the relative importance of the attributes and aggregating the attributes into some kind of overall objective function, e.g., utility or value function. Then, any appropriate single-objective optimization or mathematical programming algorithm can be applied. Solving the optimization problem with this approach generates an “optimal” solution, but only for a specified set of quantified weights or specific utility function. Unfortunately, the precise value of the objective function weights used or the form of the selected utility function dictates the final solution, and thus, broad and detailed knowledge of the system is demanded.

The second approach involves determining a number of feasible solutions along a Pareto frontier and the final solution is a set of non-dominated solutions. In this case, the Pareto set can contain a large number (in some cases, thousands) of solutions. From the decision-maker’s perspective, consideration of all the nondominated solutions can be prohibitive and inefficient.

The methods developed and presented in this chapter take the view that, for many multi-objective engineering design optimization problems, a balance between single solutions and Pareto-optimal sets can be advantageous. Thus, the post-Pareto optimality analysis methods proposed in this chapter represent a compromise between the two

extremes in Figure 4.1 with the aim to achieve a smaller practical set, called the pruned Pareto set that can be more easily analyzed by the decision maker.

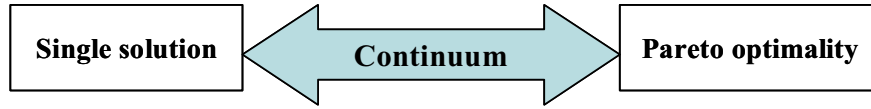


Figure 4.1 Achieving a balance between single solutions and Pareto optimal solutions

The two pruning methods presented in this Chapter are applied after the determination of Pareto-optimal sets by a MOEA. However, in Chapter 8, one of the pruning techniques presented in this chapter is incorporated as part of the MOEA body. This aspect makes the analysis of the solution of multiple objective problems more efficient.

4.1 Post-Pareto optimality analysis

Although, several methods for solving multi-objective optimization problems have been developed and studied as seen in Chapter 3, little prior work has been done on the evaluation of results obtained in multi-objective optimization. Korhonen & Halme (1990) suggested the use of a value function to help the decision-maker identify the most preferred solution in multi-objective optimization problems. Venkat *et al.* (2004) introduced and analyzed the Greedy Algorithm (GR) to obtain a sub-set of Pareto optima from a large set of the Pareto set. The selection of the sub-set was based on maximizing a scalarized function of the vector of percentile ordinal rankings of the Pareto optima within the large set.

The two main objectives of the post-Pareto optimality analysis are: *i)* obtain a smaller sub-set of preferred solutions from the large Pareto-optimal set, and *ii)* the evaluation and interpretation of the results obtained from any optimization method.

The post-Pareto analysis and the selection of one solution over the others can be quite a challenging problem since, in the absence of subjective or judgmental information, none of the corresponding trade-offs can be said to be better than the others. Thus, the motivation for the work presented next stems from challenges encountered during the post-Pareto analysis phase. To reduce or limit intelligently the size of the Pareto-optimal set, we proposed the following two methods: 1) pruning by using non-numerical objective function ranking preferences method, and 2) pruning by using data clustering, as presented in Taboada *et al.* (2005, 2007a).

As in Taboada *et al.* (2007a), Figure 4.2 shows how to select the preferred Pareto optimal set pruning procedure once the Pareto-optimal set or sub-set has been obtained. The decision-maker should select the first method if he/she knows in advance the objective function preferences as shown in Figure 4.2. Essentially, this method should be chosen by more experienced decision-makers that are familiar with the importance of the objective functions. On the other hand, if the decision-maker does not know *a priori* the objective function preferences, he/she may prefer to use the second method to cluster solutions in regions, and then, just analyze k solutions or focus on the most interesting regions to concentrate his/her efforts.

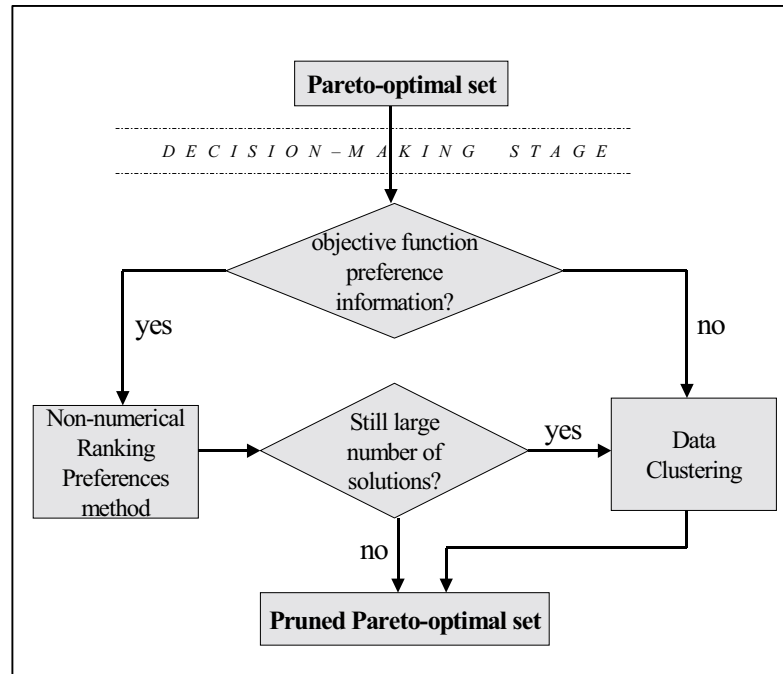


Figure 4.2 Methods to prune the Pareto-optimal set

The combination of the two proposed methods is ideally suited to address complex multi-objective optimization problems in which the Pareto-optimal set is very large. For this type of problem, where the Pareto-optimal set can contain thousands of solutions, the combination of the two pruning methods is preferred. In such cases, the pruning by using the non-numerical objective function ranking preferences method should be initially applied to obtain a Pareto sub-set that reflects the decision-maker's objective function preference, and then, the pruning by using data clustering can be applied to further reduce the size of the Pareto sub-set. Thus, the decision-maker gets a smaller set of solutions to analyze and select one solution for implementation (Taboada & Coit, 2007).

4.2 Pruning by using the non-numerical ranking preferences method

The first method is based on a non-numerical ranking of the objective functions based on their relative importance. The strength of this method is precisely that the decision-maker only ranks non-numerically (in order of relative importance) the objective

functions but does not have to select specific weight values. Instead, he/she prioritizes or ranks the objective functions (ties allowed) based on their relative importance. This pruning method helps the decision-maker select solutions that reflect his/her preferences. In a broader sense, this method is a pseudo-ranking scheme that accommodates preferences but it is different from assigning preselected weights or utility functions. This method allows objectives to have the same rank. One example of ranking objective functions is:

Objective $f_1(\mathbf{x})$ is more important than objective $f_3(\mathbf{x})$

Objective $f_3(\mathbf{x})$ is more important than objective $f_2(\mathbf{x})$

Ranked objectives = $\{f_1(\mathbf{x}), f_3(\mathbf{x}), f_2(\mathbf{x})\} : f_1(\mathbf{x}) \succ f_3(\mathbf{x}) \succ f_2(\mathbf{x})$

Based on the objective function rankings, a weight function $f_w(\mathbf{w})$ is developed, indicating the likelihood of different weight combinations. The weight function $f_w(\mathbf{w})$ is derived from a region where all the weights in this set sum up to one as shown in Figure 4.3 for a case with three objective functions.

To illustrate, consider a case where the objective function preference is $f_1 \succ f_2 \succ f_3$, and the objectives have all been similarly scaled. The exact value of the weights is not known but we know that $w_1 > w_2 > w_3$. The resulting region where the weights are sampled, and then combined with the objective functions, is shown in Figure 4.4.

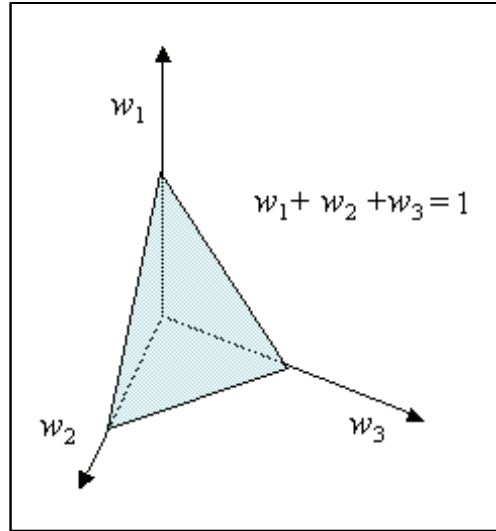
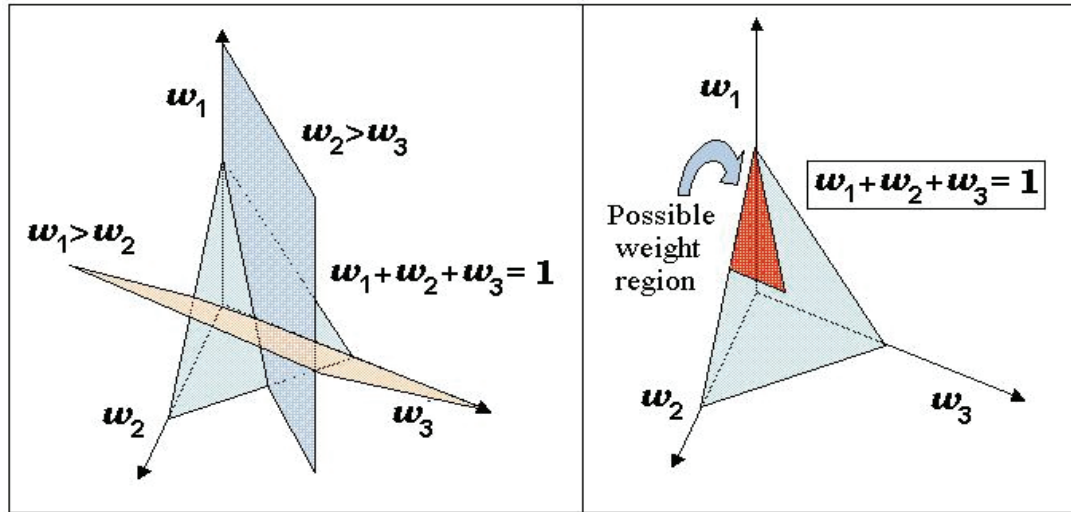


Figure 4.3 Plane containing set of possible eights

Figure 4.4 Weight region for the $f_1 \succ f_2 \succ f_3$ objective function preference

The weights are uniformly sampled from the region of interest with a weight function that is defined as follows:

$$f_w(\mathbf{w}) = \begin{cases} c & \forall \ w_1 > w_2 > w_3 \\ 0, & \text{elsewhere} \end{cases}$$

The marginal distribution of w_1 (for $w_1 > w_2 > w_3$) has been derived, in which $f_w(\mathbf{w})$ is integrated over w_2 , where c is a constant. For three objective functions,

$$f_{w_1}(w_1) = \int_{w_2} f_w(w_1, w_2) dw_2 = \begin{cases} 0, & 0 \leq w_1 < \frac{1}{3} \\ 12 \left(\frac{3}{2} w_1 - \frac{1}{2} \right), & \frac{1}{3} \leq w_1 < \frac{1}{2} \\ 12 \left(\frac{1}{2} - \frac{1}{2} w_1 \right), & \frac{1}{2} \leq w_1 < 1 \end{cases}$$

Knowing that:

$$\int_{1/3}^1 f_{w_1}(w_1) dw_1 = 1$$

Then,

$$\int_{1/3}^{1/2} c \left(\frac{3}{2} w_1 - \frac{1}{2} \right) dw_1 + \int_{1/2}^1 c \left(\frac{1}{2} - \frac{1}{2} w_1 \right) dw_1 = 1$$

Solving for c we get that $c = 12$, thus the probability density function of w_1 is given by:

$$f_{w_1}(w_1) = \int_{w_2} f_w(w_1, w_2) dw_2 = \begin{cases} 0 & 0 \leq w_1 < \frac{1}{3} \\ 12 \left(\frac{3}{2} w_1 - \frac{1}{2} \right) & \frac{1}{3} \leq w_1 < \frac{1}{2} \\ 12 \left(\frac{1}{2} - \frac{1}{2} w_1 \right) & \frac{1}{2} \leq w_1 < 1 \end{cases}$$

The marginal cumulative distribution function of w_1 is:

$$F_{w_1}(w_1) = \int_{w_2} f_w(w_1, w_2) dw_2 = \begin{cases} 0 & 0 \leq w_1 < \frac{1}{3} \\ 9w_1^2 - 6w_1 + 1 & \frac{1}{3} \leq w_1 < \frac{1}{2} \\ -3w_1^2 + 6w_1 - 2 & \frac{1}{2} \leq w_1 < 1 \end{cases}$$

The distribution function for w_2 conditioning on w_1 is given by:

$$f_{w_2|w_1}(w_2 | w_1) = \frac{f_w(w_1, w_2)}{f_{w_1}}$$

For $\frac{1}{3} \leq w_1 < \frac{1}{2}$, we get:

$$f_{w_2|w_1}(w_2 | w_1) = \begin{cases} 0 & 0 \leq w_2 \leq \frac{1}{2} - \frac{1}{2} w_1 \\ \frac{3}{2} w_1 - \frac{1}{2} & \frac{1}{2} - \frac{1}{2} w_1 \leq w_2 \leq w_1 \end{cases}$$

And for $\frac{1}{2} \leq w_1 < 1$:

$$f_{w_2|w_1}(w_2 | w_1) = \begin{cases} 0 & 0 \leq w_2 \leq \frac{1}{2} - \frac{1}{2} w_1 \\ \frac{1}{2} - \frac{1}{2} w_1 & \frac{1}{2} - \frac{1}{2} w_1 \leq w_2 \leq w_1 \end{cases}$$

Finally, by knowing the values of w_1 and w_2 , the value of w_3 is just $1 - w_1 - w_2$.

Then, random, but ranked, weights sets can be generated using Monte Carlo simulation methods. These weights adhere to the ranking pattern used for the objective functions. A substantially large set of weights is generated, with each set containing one weight for each objective. As an example, Figure 4.5 shows the distribution of 5,000 randomly generated weights. The y -axis represents the frequency or number of times a specific weight was generated and the x -axis represents the value of each individual weight. As can be seen, the possible values for the weights in the case $f_1 \succ f_2 \succ f_3$ are:

$$\frac{1}{3} < w_1 < 1, 0 < w_2 < \frac{1}{2} \text{ and } 0 < w_3 < \frac{1}{3}.$$

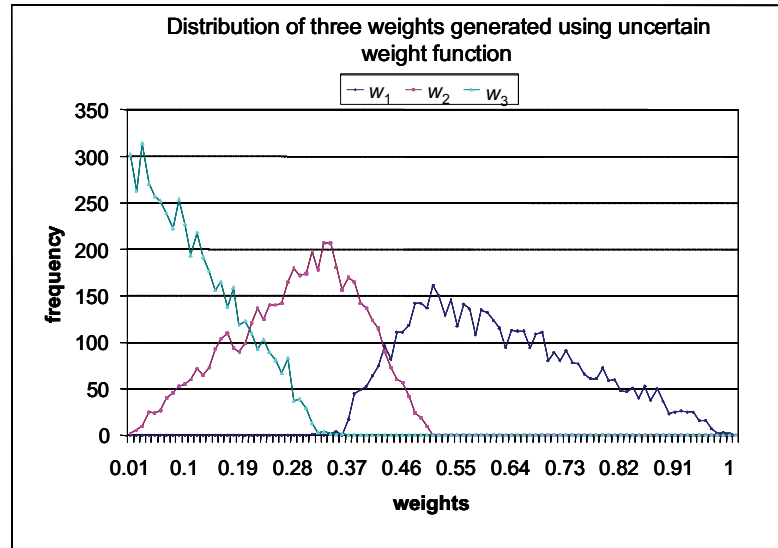


Figure 4.5 Distribution of random weights used for a three objective problem

These weight sets are used to repeatedly combine the scaled objectives into a single objective function using the randomly generated weight sets. For all minimization objectives, without loss of generality, the solution that yields the minimum value for f' is recorded and gets a counter of 1. This is repeated with the next set of weights, and the best solution for that combination is identified. This process is repeated many times (e.g., several thousand), and at the end, the solutions that have non-zero counter values will be those solutions that form the pruned Pareto set.

This method has been observed to achieve as much as a 90% reduction of the entire Pareto-optimal set (Taboada *et al.*, 2005; 2007a; Taboada & Coit, 2006a). In Section 4.6.2.1.2, a more formally mathematical formulation is presented to demonstrate the strength of the non-numerical ranking preferences method and, to show that some of the Pareto-optimal solutions will never be preferred given the objective function preferences.

This approach is an extension of earlier research considering multi-criteria decision making with a finite set of alternatives. Lahdelma *et al.* (1998) considered uncertainty in weight selection similar to the uncertain weight function proposed here. Rietveld &

Ouwensloot (1992) and Hinloopen *et al.* (2004) also describe solution methods where solutions must be selected based on ordinal data. The uncertain weight function combined with Tabu search was demonstrated by Kultural-Konak *et al.* (2006). This is an effective approach but it is potentially inefficient, and later, in Chapter 8, a new integrated algorithm is presented.

4.3 Pruning by using data clustering

This second method is a new approach based on the concepts of data clustering after determination of a Pareto-optimal set. This new approach offers benefits compared to previous approaches because it provides practical support to the decision-maker during the selection step. The main idea of this approach is to systematically assist the decision-maker during the post-Pareto analysis stage to select his/her choice without precise quantified knowledge of the relative importance of the objective functions.

In multicriteria optimization, data clustering can be a useful exploratory technique in knowledge discovery. Since it groups similar solutions together, it allows the decision-maker to identify potentially meaningful trade-offs among the solutions contained in the Pareto-optimal set without requiring the decision-maker to explicitly define objective function weights or utility functions.

4.3.1 Data clustering background

Cluster analysis is a multivariate analysis technique that is defined as the process of organizing objects in a database into clusters/groups such that objects within the same cluster have a high degree of similarity, while objects belonging to different clusters have a high degree of dissimilarity (Kaufman & Rousseeuw, 1990).

Probably, the most popular nonhierarchical partitioning method is the “ k -means” clustering algorithm. The general algorithm was introduced by Cox (1957) and MacQueen (1967) first named it “ k -means.” Since then it has become widely used and is classified as a partitional or non-hierarchical clustering method (Jain & Dubes, 1988).

The k -means algorithm is well known for its efficiency in clustering data sets. The grouping is done by calculating the centroid for each group, and assigning each observation to the group with the closest centroid. For the membership function, each data point belongs to its nearest center, forming a partition of the data. The objective function that the k -means algorithm optimizes is:

$$KM(X, C) = \sum_{i=1}^n \min_{j \in \{1, \dots, k\}} \| \mathbf{v}_i - \mathbf{c}_j \|^2$$

where:

$\mathbf{v}_i = i^{\text{th}}$ data vector

$\mathbf{c}_j = j^{\text{th}}$ cluster centroid

X = set of data vectors

C = set of centroids

This objective function is used in the algorithm to minimize the within-cluster variance (the squared distance between each center and its assigned data points). This algorithm involves the iterative assignment of cluster membership and re-calculation of centroids. The membership function for k -means is as follows, where $m_{KM}(\mathbf{c}_l | \mathbf{v}_i)$ is a 0-1 function, taking the value of 1 if data vector \mathbf{v}_i is assigned to cluster \mathbf{c}_l , and 0 otherwise.

$$m_{KM}(\mathbf{c}_l | \mathbf{v}_i) = \begin{cases} 1; & \text{if } l = \arg \min_{\forall j} \| \mathbf{v}_i - \mathbf{c}_j \|^2 \\ 0; & \text{otherwise} \end{cases}$$

The performance of the k -means clustering algorithm may be improved by estimating the ideal number of clusters represented in the data. Thus, different cluster validity indices have been suggested to address this problem. A cluster validity index indicates the quality of a resulting clustering process. Then, the clustering partition that optimizes the validity index under consideration is chosen as the best partition. The silhouette plot method is one of these cluster validity techniques.

Rousseeuw (1987) and Rousseeuw *et al.* (1989) suggested a graphical display, the silhouette plot, to evaluate the quality of a clustering allocation, independently of the clustering technique that is used. The silhouette value for each point is a measure of how similar that point is to points in its own cluster compared to points in other clusters. $s(i)$ is known as the silhouette width. This value is a confidence indicator on the membership of the i^{th} sample in cluster X_j and it is defined as:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

where $a(i)$ is the average distance from the i^{th} point to all the other points in its cluster, and $b(i)$ is the average distance from the i^{th} point to all the points in the nearest neighbor cluster.

The value of $s(i)$ ranges from +1 to -1. A value of +1, indicates points that are very distant from neighboring clusters; a value of 0, indicates points that are not distinctly in one cluster or another, and a value of -1, indicates points that are probably assigned to the wrong cluster.

For a given cluster, X_j , it is possible to calculate a *cluster silhouette* S_j , which characterizes the heterogeneity and isolation properties of such a cluster. It is calculated as the average of the sum of all samples' silhouette widths in X_j .

Moreover, for any partition, a global silhouette value or silhouette index, GS_u , can be used as an effective validity index for a partition U .

$$S_j = \frac{\sum_{i \in X_j} s(i)}{|X_j|}$$

$$GS_u = \frac{1}{c} \sum_{j=1}^c S_j$$

It has been demonstrated that this equation can be applied to estimate the “optimal” or preferred number of clusters for a partition U (Rousseeuw, 1987). In this case the partition with the maximum silhouette index value is taken as the optimal partition.

4.3.2 Description of the new approach

The developed approach is based on the following steps:

1. Obtain the entire Pareto-optimal set or sub-set of Pareto solutions by using a multi-purpose MOEA (such as NSGA-II).
2. Apply the k -means algorithm to form clusters on the solutions contained in the Pareto set. The solution vectors are defined by the specific objective function values, $f_i(\mathbf{x})$, for each prospective solution. Normalization of the objective function values is recommended to have comparable units. Several replicates are needed to avoid local optima. The solution to consider is the one with the lowest total sum of distances over all replicates.
3. To determine the “optimal” or preferred number of clusters, k , in this set, silhouette plots are used. A value of the silhouette width, $s(i)$, is obtained for several values of k . The clustering with the highest average silhouette width, GS_u , is selected as the “optimal” or preferred number of clusters in the Pareto-optimal set.

4. For each cluster, select a representative solution. To do this, the solution that is closest to its respective cluster centroid is chosen as a good representative solution. This results in a dramatic reduction in the number of solutions that the decision-maker must consider.
5. Analyze the representative solutions based on the priorities and preferences of the decision-maker. At this stage, the decision-maker can either select one solution among the k representative solutions, or he/she can decide to perform further investigation on the cluster that he/she is most interested. An unbiased suggestion is to focus on the cluster that has the solutions that conform to the “knee” region (Das, 1999; Branke *et al.*, 2004). The “knee” is formed by those solutions of the Pareto-optimal front where a small improvement in one objective would lead to a large deterioration in at least one other objective.
6. Then, Steps 2, 3 and 4 are applied again on this reduced space formed by the solutions in the selected “knee” cluster.

By following this approach, one systematically contracts the subspace in the direction of the most relevant solutions for the decision-maker until a unique selection can be made.

4.3.3 MATLAB® implementation

After obtaining the Pareto set from a particular MOEA, e.g., from the NSGA-II in the preliminary research results, a MATLAB® code was developed to perform the steps of the proposed technique. From normalized data, the code runs the k -means algorithm, from two to a specified number of means; it calculates average silhouette values and the clustering with the highest average silhouette width, GS_u , is selected as the “optimal”

number of clusters in the Pareto-optimal set. An overview of the algorithmic implementation is shown in Figure 4.6.

```

For  $C=2$  to  $MC$     *maximum number of centroids*
  For  $Z=1$  to  $R$     *number of replicates*
    Randomly select initial values for  $C$ 
    For each  $\mathbf{v}_i \in X$ , assign all  $\mathbf{v}_i$  to  $\mathbf{c}_j \in C$  according to nearest  $\mathbf{c}_j$ 
    Recompute  $\mathbf{c}_j$ 
    Until no change in  $\mathbf{c}_j$ 
    Return  $C$ ,  $KM(X,C)$  and membership
    Store values for  $C$ ,  $KM(X,C)$  and membership
     $Z=Z+1$ 
  end
end
Select the minimum  $KM(X,C)$  obtained for all replicates
end
Obtain silhouette values,  $s(i)$ 
Choose the cluster with the maximum silhouette width,  $GS_u$ , of all centroids considered

```

Figure 4.6 Overview of clustering algorithmic implementation

Notice that k -means can converge to a local optimum, in this case, a partition of points in which moving any single point to a different cluster increases the total sum of distances. This problem can be solved by performing several replicates, each with a new set of initial cluster centroid positions. That is, each of the replicates begins from a different randomly selected set of initial centroids. The final solution is the one with the lowest total sum of distances over all replicates.

4.4 Numerical examples

Three examples of two different multi-objective problems are used to illustrate the two proposed methods to narrow the search space. The first example presented is the well-known redundancy allocation problem (RAP) which was formulated as a multi-objective problem to maximize the system reliability and to minimize cost and weight of the system. The second and third examples address the scheduling of a Printed Wiring Board (PWB) manufacturing line (Yu *et al.*, 2002) formulated with four objectives.

4.5 Redundancy allocation problem (RAP)

4.5.1 Description of the RAP

The RAP is a system design optimization problem. This system has a total of s subsystems arranged in series. For each subsystem, there are n_i functionally equivalent components arranged in parallel, thus n_i is a decision variable. Each component has potentially different levels of cost, weight, reliability and other characteristics. The n_i components are to be selected from m_i available component types, where multiple copies of each type can be selected. An example of a series-parallel system is depicted in Figure 4.7.

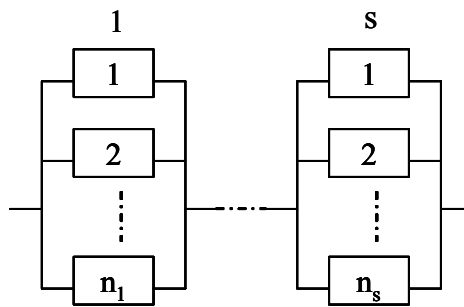


Figure 4.7 General series-parallel redundancy system

The use of redundancy improves system reliability but also adds cost, weight, etc., to the system. There are generally system-level constraints and the problem is to select the design configuration that maximizes some stated objective functions.

4.5.2 Previous research

Solving the redundancy allocation problem has been shown to be NP-hard by Chern (1992). Different optimization approaches have been previously used to determine optimal or good solutions to this problem.

It has been solved using dynamic programming by Bellman (1957) and Bellman & Dreyfus (1958) to maximize reliability for a system given a single cost constraint. For

each subsystem, there was only one component choice so the problem was to identify the optimal levels of redundancy, n_i . Fyffe *et al.* (1968) also used a dynamic programming approach and solved a more difficult design problem. They considered a system with 14 subsystems, and constraints on both cost and weight. For each subsystem, there were three or four different component choices each with different reliability, cost and weight. Bulfin & Liu (1985) used integer programming and they formulated the problem as a knapsack problem using surrogate constraints.

Unfortunately, the mathematical programming approaches are only applicable to a limited or restricted problem domain and require simplifying assumptions, which limits the search to an artificially restricted search space. In these formulations, once a component selection is made, only the same component type can be used to provide redundancy. This restriction is required so the selected mathematical programming tool can be applied, but it is not an actual imposition of the engineering design problem. Thus, the resulting solution is only “optimal” for a restricted solution space, and better solutions can be found if the restriction is no longer imposed.

GAs offer many advantages compared to alternative methods used to solve the RAP. Coit & Smith (1996a, 1996b) used GAs to obtain solutions to the redundancy allocation problem. In their research, they solved 33 variations of the Fyffe’s problem using a GA.

Several techniques considering multiple criteria have been presented in the literature. A multi-objective formulation of a reliability allocation problem to maximize system reliability and minimize the system cost was considered by Sakawa (1978) using the surrogate worth trade-off method. Inagaki *et al.* (1978) used interactive optimization to design a system with minimum cost and weight and maximum reliability.

Dhingra (1992) used goal programming and goal-attainment to generate Pareto-optimal solutions to solve a special case of a multi-objective RAP. Busacca *et al.* (2001) proposed a multi-objective GA approach that was applied to a design problem with the aim to identify the optimal system configuration and components with respect to reliability and cost objectives.

Kulturel-Konak *et al.* (2003) solved this problem using Tabu Search method considering three objective functions; maximization of system reliability and minimization of cost and weight of the system.

The following notation is used throughout the remainder of this example:

Notation:

R, C, W = system level reliability, cost and weight or constraint limits

s = number of subsystems

x_{ij} = quantity of the j^{th} available component used in subsystem i

$\mathbf{x}_i = (x_{i,1}, x_{i,2}, \dots, x_{i,m_i})$

m_i = total number of available components for subsystem i

$n_{\max,i}$ = user defined maximum number of components in parallel used in subsystem i

$R_i(\mathbf{x}_i)$ = reliability of subsystem i

c_{ij}, w_{ij}, r_{ij} = cost, weight and reliability for the j^{th} available component for subsystem i

ω_i = weight used for objective i in the weighted sum method

It is important to highlight that ω_i and w_{ij} , are conceptually very different and only for this current example (RAP) the w 's are used to refer to component weights and the ω 's to refer to objective function weights.

4.5.3 Problem formulation

It is well known that redundant elements increase system reliability, but also increase the procurement cost and system weight. A non-trivial question arises then regarding how

to optimally allocate redundant elements. The answer depends on the criterion of optimality and on the structure of the designed system.

Different problem formulations of the RAP have been presented in the literature. For instance, Problem P1 maximizes the system reliability given restrictions on the system cost, C , and the system weight, W . Alternatively, Problem P2 is formulated as a multi-objective optimization problem by using a weighted sum approach. Problem P3 is a multi-criteria formulation of the redundancy allocation problem, in which a Pareto-optimal set of solutions is obtained. The formulation of the three problems is shown below:

Problem P1: Reliability maximization

$$\max \prod_{i=1}^s R_i(\mathbf{x}_i)$$

Subject to:

$$\sum_{i=1}^s \sum_{j=1}^{m_i} c_{ij} x_{ij} \leq C$$

$$\sum_{i=1}^s \sum_{j=1}^{m_i} w_{ij} x_{ij} \leq W$$

$$1 \leq \sum_{j=1}^{m_i} x_{ij} \leq n_{\max,i} \quad \text{for } \forall i = 1, 2, \dots, s$$

$$x_{ij} \in \{0, 1, 2, \dots\}$$

Given the overall restrictions on system cost of C and weight of W , the problem is to determine which design alternative to select with the specified level of component reliability, and how many redundant components to use in order to achieve the maximum reliability.

Problem P2: Weighted sum method formulation

$$\max \omega_1 R - \omega_2 C - \omega_3 W = \omega_1 \left[\prod_{i=1}^s R_i(x_i) \right] - \omega_2 \left[\sum_{i=1}^s \sum_{j=1}^{m_i} c_{ij} x_{ij} \right] - \omega_3 \left[\sum_{i=1}^s \sum_{j=1}^{m_i} w_{ij} x_{ij} \right]$$

Subject to:

$$1 \leq \sum_{j=1}^{m_i} x_{ij} \leq n_{\max,i} \quad \text{for } \forall i = 1, 2, \dots, s$$

$$\sum_{i=1}^n \omega_i = 1$$

$$x_{ij} \in \{0, 1, 2, \dots\}$$

The solution for this multi-objective formulation is determined by combining the three objectives into a single objective problem. This requires the user to *a priori* specify objective function weights to represent the relative importance to the individual objective function and one single set of weighting coefficients yields only one “optimal” solution. Therefore, choosing the correct set of weights, that provide the decision-maker an attractive set of solutions can be highly challenging, yet it will dictate the final solution. Often, decision-makers lack the training or detailed knowledge to precisely select the weights. This is undesirable because even small alterations to the weights can lead to very different final solutions.

Problem P3: Multi-objective formulation

$$\max \left[\prod_{i=1}^s R_i(x_i) \right], \min \left[\sum_{i=1}^s \sum_{j=1}^{m_i} c_{ij} x_{ij} \right], \min \left[\sum_{i=1}^s \sum_{j=1}^{m_i} w_{ij} x_{ij} \right]$$

Subject to

$$1 \leq \sum_{j=1}^{m_i} x_{ij} \leq n_{\max,i} \quad \text{for } \forall i = 1, 2, \dots, s$$

$$x_{ij} \in \{0, 1, 2, \dots\}$$

For the multi-objective RAP, the objectives are to determine the optimal design configuration that will maximize system reliability, minimize the total cost and minimize the system weight, for a series-parallel system. A Pareto-optimal set of solutions can be obtained by using any multi-objective evolutionary algorithm (MOEA) available. However, there may be too many prospective solutions for the decision-maker to fully consider before ultimately select a unique design configuration to implement. This P3 problem formulation is the one addressed in this chapter.

4.5.4 Multi-objective RAP example

To illustrate how pruning can be of great aid for the decision-maker on the post-Pareto analysis stage, a RAP was solved. For this example, the configuration selected consists of 3 subsystems, with an option of 5, 4 and 5 types of components in each subsystem, respectively. The optimization involves selection from among these component types. The minimum number of components in each subsystem is 1, for the system to function, and the maximum number of components is 8 in each subsystem.

Table 4.1 defines the component choices for each subsystem.

Table 4.1 Component choices for each subsystem

Design Alternative j	Subsystem i								
	1			2			3		
	r_{ij}	c_{ij}	w_{ij}	r_{ij}	c_{ij}	w_{ij}	r_{ij}	c_{ij}	w_{ij}
1	0.95	2	5	0.99	4	4	0.90	6	5
2	0.93	1	4	0.98	3	6	0.85	5	4
3	0.91	2	2	0.97	1	5	0.82	3	3
4	0.90	1	3	0.96	2	7	0.79	3	5
5	0.95	2	8				0.99	2	4

For this case, NSGA was solved with a population size of 100. There were 46 solutions in the Pareto-optimal set. This is likely to be too many possibilities for a decision-maker to select one preferred solution. This set is shown in Figure 4.8. The

Pareto set was then pruned using both methods previously described.

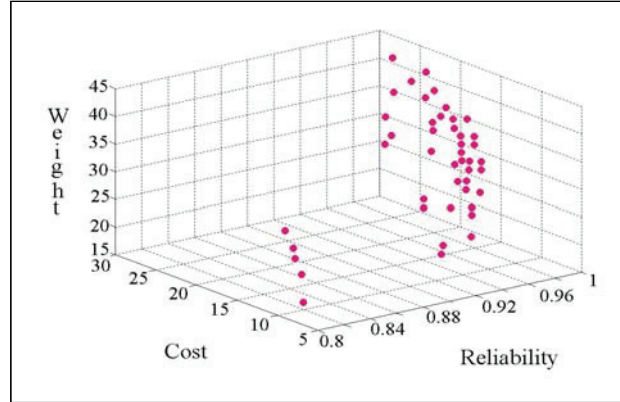


Figure 4.8 Pareto-optimal set

4.5.4.1 Pruned results by using the non-numerical ranking preferences method

Pareto optimal solutions were obtained using NSGA and the pruned solutions identified by using the proposed method. The objective function priorities used on these solutions were: $(R \succ C \succ W)$, $(C \succ R \succ W)$, $(R \succ W \succ C)$ and $(W \succ R \succ C)$. Figure 4.9 shows the pruned solution set for $w_1 > w_2 > w_3$ and $w_2 > w_1 > w_3$, compared to the original Pareto optimal set (obtained by using NSGA), and Figure 4.10 shows the pruned solution set for $w_1 > w_3 > w_2$ and $w_3 > w_1 > w_2$, compared to the original Pareto-optimal set. Considering two objective functions at a time, the charts maps reliability versus cost. The pruned solution sets for all four possibilities are shown in the figures. Pruning the solutions caused almost a 90% reduction in the size of the Pareto-optimal set.

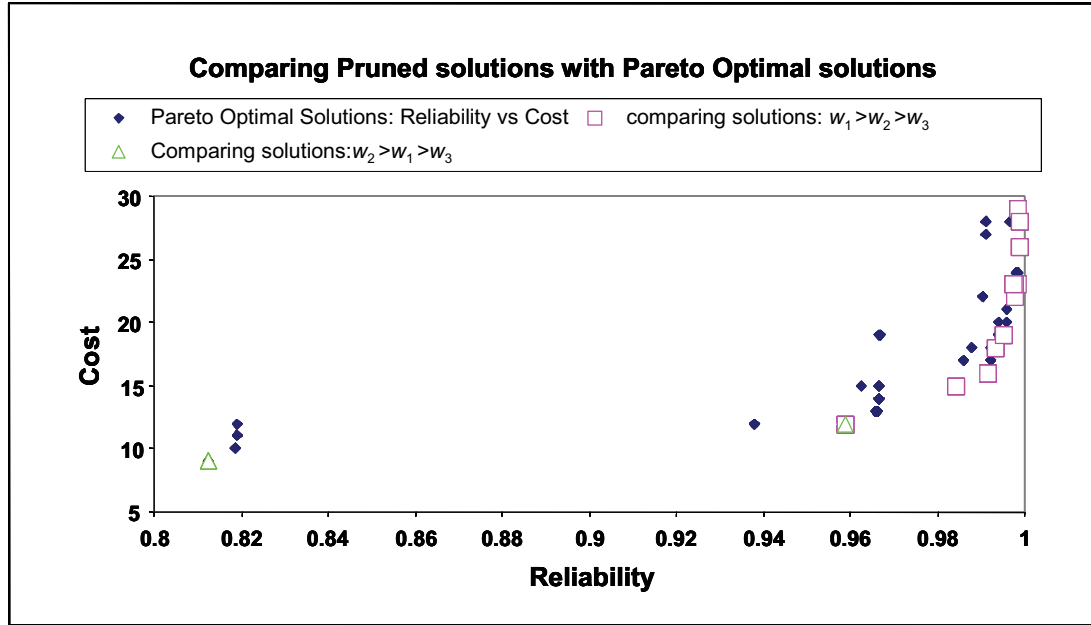


Figure 4.9 Comparing pruned Pareto solution with the Pareto-optimal solution set for reliability versus cost

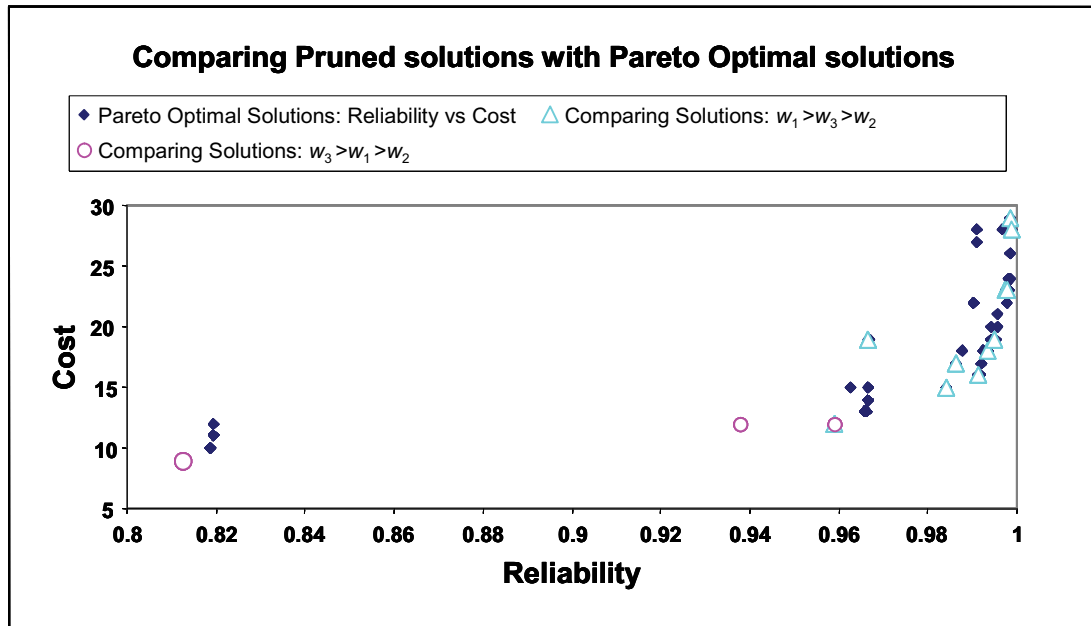


Figure 4.10 Comparing pruned Pareto solution with the Pareto-optimal solution set for reliability versus cost

4.5.4.2 Pruned results by using data clustering

The k -means algorithm was then used to cluster the original 46 solutions found in the Pareto set. Normalization of the objective function values was performed to have comparable units. Thus, the three objective functions were normalized using the

following linear normalization equation; although other types of normalizing equations can be used, i.e., logarithmic.

$$\frac{f_i(\mathbf{x}) - f_i^{\min}}{f_i^{\max} - f_i^{\min}} \quad \forall i = 1, 2, \dots, n$$

where f_i^{\min} = minimum value for $f_i(\mathbf{x})$ found in the Pareto optimal set.

f_i^{\max} = maximum value for $f_i(\mathbf{x})$ found in the Pareto optimal set.

To use the above equation, all the objective functions were considered to be minimized, thus reliability was multiplied by -1. It is important to remark that by using a different type of normalization function, the clustering outcome is potentially different.

To determine the optimal number of clusters, silhouette plots were used as suggested by Rousseeuw (1987), and several runs were performed for different values of k with several replicates for each value of k . For this particular data set, we found three to be the optimum number of clusters. The three clusters are shown in Figure 4.11 from normalized data. Cluster 1 contained 22 solutions; there were 19 solutions in cluster 2 and five in cluster 3.

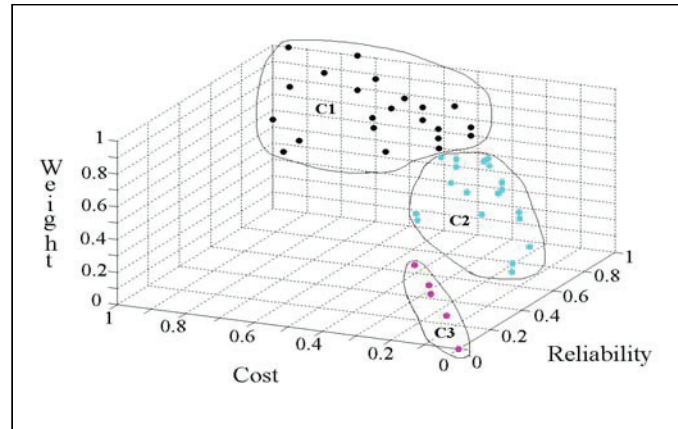


Figure 4.11 Clustered Pareto-optimal set

The clusters formed are highly internally homogeneous. That is, members within a cluster are similar to one another. One way for the decision maker to select one solution,

among the solutions contained in a cluster, is to identify what solution is the closest to its centroid. Table 4.2 shows the summary of results obtained by the cluster analysis. The representative solutions are those that are closest to their corresponding centroid; each solution is shown with its corresponding reliability, cost and weight. With the information from Table 4.2, the decision maker now has a small set of solutions, and it is thus, easier to make his/her choice regarding the importance of the different objectives.

Table 4.2 Summary of results obtained with the clustering analysis

	# of solutions	Representative solutions	Reliability	Cost	Weight
Cluster 1	22	#39	0.9978541	22	34
Cluster 2	19	#91	0.984265	15	25
Cluster 3	5	#87	0.819216	11	24

Another way to take advantage of this method is that, once having the optimal number of clusters selected (three in our case), then we looked for the cluster that contained the most interesting solutions of the Pareto optimal set. These are the solutions where a small improvement in one objective would lead to a large deterioration in at least one other objective. These solutions are sometimes called “knees.” In this case, as we can see from Figure 4.11, solutions in cluster 2 are likely to be more relevant to the decision maker. Thus, solution #91 can be chosen as a good representative solution of this mentioned knee region.

4.6 Scheduling of unrelated parallel machines

4.6.1 Multiple objective scheduling problems

There are numerous cases where scheduling problems have more than one, often conflicting objectives. For instance, flow-shop scheduling (minimizing completion time, tardiness, mean-flow-time), project scheduling (minimizing the projects completion time, the tardiness of orders), examination time-tabling (minimizing total number of violations

of each type of constraints) and JIT sequencing (minimizing total utility work, minimizing set-up cost) and many others. Recent and comprehensive surveys on theory and applications of multi-criteria scheduling are provided by T'kindt & Billaut (2002) and Hoogeveen (2005).

Over the years, there have been several approaches used to model the various objectives in such problems. MOGAs have been recognized to be well-suited for solving multiple objective optimization problems because of their abilities to exploit and explore multiple solutions in parallel and find a widespread set of non-dominated solutions in a single run. Thus, scheduling with multiple objectives has been one of the most attractive applications of the MOGAs.

Murata *et al.* (1996) proposed a MOGA and applied it to a flowshop scheduling problem with two objectives; minimization of makespan and minimization of the total flowtime. Ponnambalam *et al.* (2001) proposed a MOGA for scheduling job shops. The performance criterion considered was the weighed sum of the multiple objectives minimization of makespan, minimization of total idle time of machines and minimization of total tardiness. Arroyo & Armentano (2005) proposed a multi-objective local search (MOGLS) genetic algorithm, which was applied to the flowshop scheduling problem for the following two pairs of objectives; (i) makespan and maximum tardiness; (ii) makespan and total tardiness.

4.6.2 Scheduling of unrelated parallel machines: multi-objective formulation

The second and third examples addressed the drilling of Printed Wiring Boards (PWBs) which is performed by a group of unrelated parallel machines (Yu *et al.*, 2002) which must be scheduled. The processing time of each lot may be different for different

machines, and a machine that has a shorter processing time for a particular lot may have a longer processing time for another lot. There are multiple criteria that need to be considered to determine the best schedule. Parallel machine scheduling problems are generally NP-hard problems (Karp, 1972). In terms of the complexity hierarchy of deterministic scheduling, unrelated machines scheduling problems are some of the most difficult to solve.

Yu *et al.* (2002) proposed a Lagrangian Relaxation Heuristic (LRH) method to solve the PWB scheduling problem. They constructed an integer programming model with a special structure called unimodularity. In order to account for multiple objectives of the scheduling system, they introduced preference constraints and brought them into the objective function by using Lagrangian relaxation.

This PWB scheduling problem was formulated as a multi-objective problem considering four objectives to be minimized; minimize overtime, minimize average finish time, minimize the variance of the finish time and minimize the total cost. The multi-objective formulation is as follows:

$$\min \sum_{i=1}^m O_i, \quad \min \mu_c, \quad \min \sum_{i=1}^m \frac{(C_i - \mu_c)^2}{m}, \quad \min \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

subject to:

$$\begin{aligned} \sum_{i=1}^m x_{ij} &= 1 \\ O_i &= \max \left(\sum_{j=1}^n p_{ij} x_{ij} - T, 0 \right) \\ C_i &= \sum_{j=1}^n p_{ij} x_{ij} \\ \mu_c &= \frac{\sum_{i=1}^m C_i}{m} \end{aligned}$$

$$x_{ij} \in \{0, 1\}$$

where:

$$x_{ij} = \begin{cases} 1, & \text{if lot } j \text{ is assigned to machine } i \\ 0, & \text{otherwise} \end{cases}$$

O_i = Overtime on machine i

m = number of parallel machines

n = number of lots to schedule

p_{ij} = processing time of lot j on machine i

c_{ij} = cost of processing a lot j on machine i

T = lot release interval time

4.6.2.1 Scheduling of unrelated parallel machines: example 1

The processing times and the processing costs are shown in Tables 4.3 and 4.4 respectively (Yu *et al.*, 2002). The release interval time, T , is equal to 3 time units. To satisfy feasibility, a large cost is assumed for processing a lot that cannot be processed by certain machines, such as in the cases of lot 1 on machine 1 and machine 2, forcing them to be scheduled on a machine that can perform the job.

Table 4.3 Processing times for PWB scheduling problem

Machine	Lot 1	Lot 2	Lot 3	Lot 4	Lot 5	Lot 6
M1	∞	1.7	2.4	1.3	∞	3.5
M2	∞	1.5	2.2	0.8	3.2	1.9
M3	1.1	0.7	3.2	1.8	3.1	0.4

Table 4.4 Processing costs for PWB scheduling problem

Machine	Lot 1	Lot 2	Lot 3	Lot 4	Lot 5	Lot 6
M1	6000	14	28	29	6000	13
M2	6000	23	16	13	25	10
M3	11	27	10	12	24	11

The multi-objective scheduling of PWB problem was initially solved, using the NSGA-II algorithm, to determine a Pareto optimal set. The algorithm was run for 150 generations with a population size of 500, with the probability of crossover as 0.7, and taking the probability of mutation to be 0.03. There were 28 solutions in the Pareto-

optimal set. The post-Pareto analysis was then performed on these 28 solutions to provide the decision-maker a workable sub-set of solutions by using the two proposed methods.

4.6.2.1.1 Pruned results by using the non-numerical ranking preferences method

The non-numerical ranking preference combination selected to illustrate this example is the case in which overtime (O) is more important than average finish time (AFT) , which is more important than variance of finish time (VFT), which is more important than cost (C) ($O \succ AFT \succ VFT \succ C$: $w_1 > w_2 > w_3 > w_4$). Figure 4.12 shows, in a three-dimensional space, the 28 solutions contained in the Pareto-optimal set.

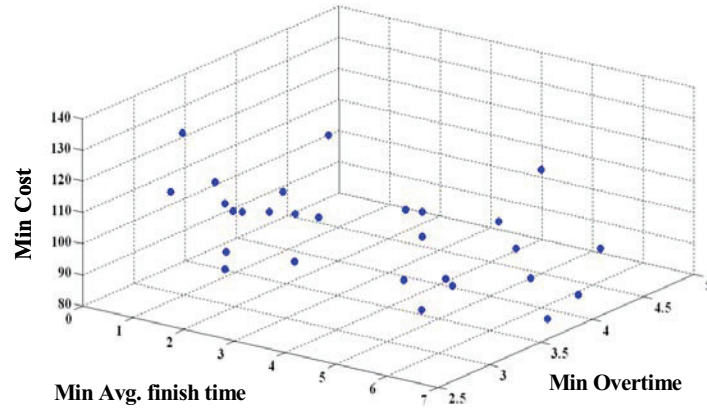


Figure 4.12 Pareto-optimal set of Example 1 (PWB) in a three-dimensional space

Table 4.5 shows the pruned solutions obtained by applying the proposed method to narrow the search space using 5000 randomly generated weight sets. Of the original 28 solutions, the pruned set only includes three; solution 1 has the minimum overtime but it is achieved at a higher cost than solutions 2 and 5. On the other hand, solution 5 presents the minimum cost but it has the highest average finish time as well as the highest variance of the average finish time.

Table 4.5 Pruned solutions

Ranking preferences: $w_1 > w_2 > w_3 > w_4$				
Sol No.	Minimize			
	Overtime	Avg. finish time	Var. Avg. finish time	Cost
2	1.0000000	2.8333370	0.6488890	115
5	1.6000000	3.1000000	1.4066670	89
1	0.9000000	3.0333330	0.3888890	131

The pruned solutions obtained considering the $w_1 > w_2 > w_3 > w_4$ objective function preference are shown as triangles in Figure 4.13. In this case, by using this method, a 89.2% reduction was achieved in the solutions obtained from the Pareto-optimal set. These pruned solutions would then be further analyzed by the decision-maker. Solution 2 is shown as an example of a schedule for the PWB scheduling problem in Figure 4.14.

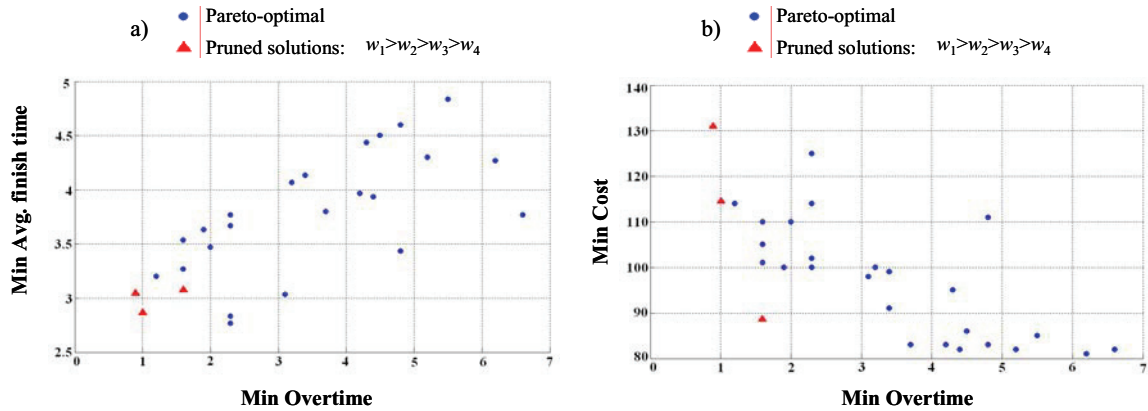
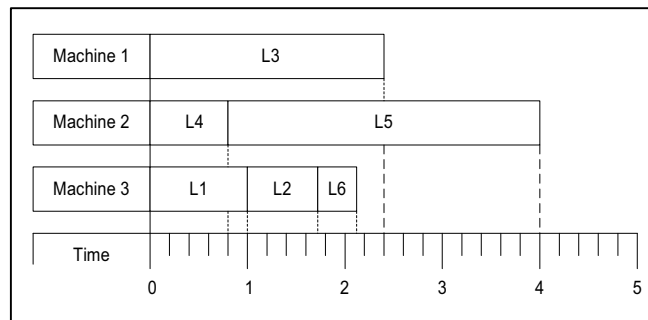
Figure 4.13 Pruned solutions for the $w_1 > w_2 > w_3 > w_4$ objective function preference in a two-dimensional space

Figure 4.14 Schedule for solution 2

This pruning method gives as a result the solutions, contained in the Pareto-optimal set, that clearly reflect the decision-maker objective function preferences. For this

example, these three solutions obtained are the best solutions that reflect the $w_1 > w_2 > w_3 > w_4$ decision-maker's objective function preference.

To empirically demonstrate that the pruning method is repeatable and reliable, and there is no sacrifice for the non-numerically ranking preferences method, ten simulation runs have been performed. In each of the ten runs, 5000 different weights sets, randomly selected from $f_w(\mathbf{w})$, were used. Table 4.6 shows the results for the ten simulation runs. In this table, we can observe that given this objective function preference, besides the three solutions found in the pruning set, there is no other solution in the Pareto-optimal set that satisfies this objectives preference. The other solutions that do not appear on the table it is because they had a counter value of zero, meaning that they did not minimize the value of f' : $f' = w_1 f_1(x) + w_2 f_2(x) + w_3 f_3(x) + w_4 f_4(x)$. However, a more formal mathematical formulation to demonstrate the accuracy and optimality of the non-numerical ranking preferences method is presented in the next section.

Table 4.6 Solutions found in the pruned Pareto set in ten simulation runs

Solution #	Counter on Simulation Run #									
	1	2	3	4	5	6	7	8	9	10
1	156	155	146	143	142	156	156	156	134	149
2	4545	4538	4539	4577	4559	4558	4525	4568	4579	4539
5	299	307	315	280	299	286	319	276	287	312

4.6.2.1.2 Mathematical demonstration of the non-numerical ranking preferences method: example 1

A more formally mathematical formulation was developed to demonstrate the strength of the non-numerical ranking preferences method. For this pruning method, the objective to minimize (for all objectives converted to minimization) the following combined objective function. Remember that the weights have not been selected, but there ordering has been specified. The objective is to find all solutions that could

conceivably minimize this function for some feasible weight combination.

$$\begin{aligned} \min \quad & \sum_i w_i f_i \\ \text{s.t.} \quad & w_{(1)} > w_{(2)} > \dots > w_{(n)} \\ & w_{(1)} + w_{(2)} + \dots + w_{(n)} = 1 \\ & w_{(i)} \geq 0 \text{ for } i = 1, 2, \dots, n \end{aligned}$$

Given the objective function preferences, we developed the algorithm shown in Figure 4.15 to analytically determine the pruned Pareto set for Example 1 (PWB). This set represents all solutions that possibly could be preferred given the new constraints caused by the prioritization.

```

Initialize:  $PP = \{\emptyset\}$ 
For  $l=1$  to  $c$ 
  Solve
    
$$z = \min_{\mathbf{w}} \max_{j \in P - \{l\}} \left\{ \sum_{i=1}^n w_i (f_{il} - f_{ij}) \right\}$$

    s.t.
      
$$w_{(1)} > w_{(2)} > \dots > w_{(n)}$$

      
$$\sum_{i=1}^n w_i = 1 \quad i \in \{0, 1, \dots, n\}$$

    If  $z \leq 0$ , then  $l$  joins  $PP$ 
    If  $z > 0$ , remove  $l$ , does not join  $PP$ 
  end
set  $PP$  is the final pruned set

```

Figure 4.15 Algorithm to prune the Pareto set given objective functions preferences

Where,

P = Pareto set

PP = pruned Pareto set

l = individual solution from P

c = total number of solutions in P

W = set of weights

This works as follows. For a solution l to be a member of the pruned-Pareto set, there must be some weight combination (adhering to the prioritization constraints) where the

combined objective function is lower than all other solutions, i.e., all differences in objective functions must be negative. Therefore, if there exists some weight combination where even the maximum difference is negative, then the solution belongs in the pruned-Pareto set. Therefore we minimize this maximum objective function difference to see whether there is some weight combination where it is the best. This then must be repeated individually for all solutions in the Pareto-optimal set. Example 1 only had 28 solutions so we can check the pruned Pareto set and compare it to the results from the simulation.

Table 4.7 shows the solutions for the four objective functions in a normalized space. Column 6 shows the z values obtained by applying algorithm shown in Figure 4.15 As can be seen, only solutions 1, 2 and 5 have z values less than zero, and thus, only these solutions can minimize the objective function given the $f_1 \succ f_2 \succ f_3$ objective function preferences. Therefore, there is no sacrifice for this example. This confirms that, with the non-numerical ranking preferences method, we are only obtaining those solutions that clearly satisfy the decision maker's objective function preferences.

Table 4.7 Analytical pruning results

sol. #	Min Overtime	Min AFT	Min Var(AFT)	Min cost	z
1	0	0.129029	0.019612	1	-0.01478
2	0.017544	0.048387	0.034512	0.68	-0.04979
3	0.052632	0.209676	0.021778	0.66	0.037485
4	0.122807	0.161289	0.061895	0.58	0.056338
5	0.122807	0.161289	0.077942	0.16	-0.05884
6	0.122807	0.241936	0.054382	0.48	0.059696
7	0.122807	0.370966	0.006619	0.40	0.060775
8	0.175439	0.419353	0.030311	0.38	0.096721
9	0.192982	0.338711	0.048651	0.58	0.12384
10	0.245614	0	0.267192	0.42	0.060139
11	0.245614	0.032255	0.196127	0.66	0.096737
12	0.245614	0.435485	0.024196	0.38	0.13085
13	0.245614	0.483873	0.003565	0.88	0.21081
14	0.385965	0.129029	0.261077	0.34	0.13743
15	0.403509	0.629034	0.007387	0.38	0.22623
16	0.438596	0.661289	0.040243	0.36	0.24681
17	0.438596	0.661289	0.043682	0.20	0.20909
18	0.491228	0.5	0.227711	0.04	0.18913
19	0.578947	0.580647	0.169002	0.04	0.21771
20	0.596491	0.806451	0.036042	0.28	0.30407
21	0.614035	0.564515	0.351249	0.02	0.26302
22	0.631579	0.838711	0.012224	0.10	0.27202
23	0.684211	0.322578	0.551450	0.04	0.27472
24	0.684211	0.887098	0	0.60	0.41233
25	0.754386	0.741936	0.200586	0.02	0.30690
26	0.807018	1	0.081507	0.08	0.37054
27	0.929825	0.725809	0.695110	0	0.46593
28	1	0.483873	1	0.02	0.50352

4.6.2.1.3 Pruned results by using data clustering: example 1

Using the k -means algorithm, clustering analysis was performed on the 28 solutions contained in the Pareto-optimal set and $k = 3$ was found to be the optimal number of clusters with the aid of the silhouette plots. Figure 4.16 shows the three clusters in a three-dimensional space, for minimizing overtime, minimizing average finish time and minimizing cost. Figure 4.17 shows the same clusters in a three-dimensional space for minimizing overtime, minimizing average finish time and minimizing the variance of the average finish time. In Figures 4.16 and 4.17, the fourth objective (minimize variance of average finish time and minimize cost, respectively) is not shown but it is still considered in the analysis. The objective functions have been normalized from 0 to 1.

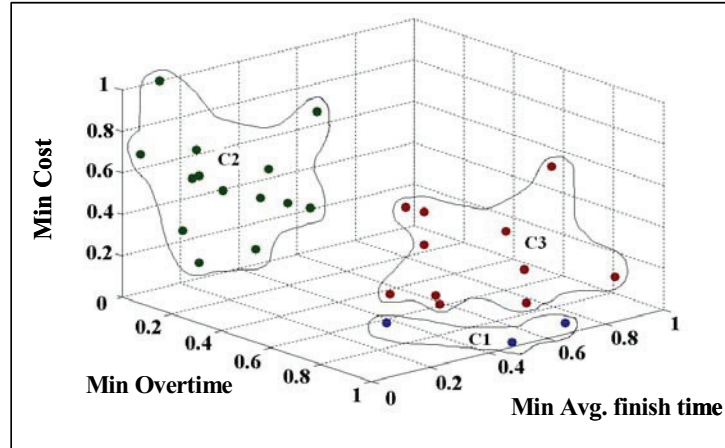


Figure 4.16 Clustered data in a three-dimensional space

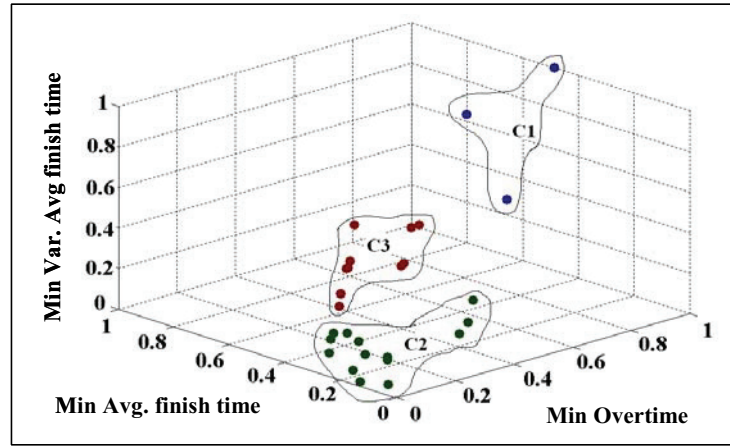


Figure 4.17 Clustered data in a three-dimensional space

Table 4.8 shows the summary of the results obtained by using the k -means algorithm. This table includes the representative solutions that were closest to their corresponding centroid. As we can see from Table 4.8, solution number 6 from cluster 2 gives the minimum overtime, average finish time and variance of the average finish time but it has the highest cost. On the other hand, solution 28 from cluster 1 has the lowest cost but it also gives the highest overtime and variance of the average finish time. Figure 4.18 shows, as an example, the schedule for solution 6.

Table 4.8 Results obtained with the cluster analysis

	# of solutions	Representative Solution	Min Overtime	Min Avg. Finish time	Min Var. Avg. finish time	Min Cost
Cluster 1	3	28	6.2000000	4.2666670	12.1755550	81
Cluster 2	14	6	1.5000000	3.2333333	0.9955560	105
Cluster 3	11	21	4.3000000	4.4333330	0.6755560	95

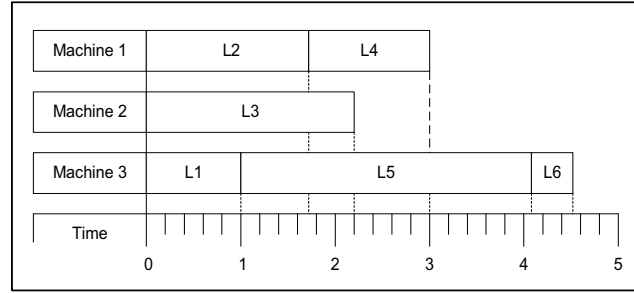


Figure 4.18 Schedule for solution number 6

For both pruning methods, the decision-maker only needs to consider three solutions (instead of 28). It is much easier and convenient to select from among 3. The first pruning method is most appropriate when the decision-maker can prioritize their objectives, while the second does not require the decision-maker to *a priori* specify any objective function preference.

4.6.2.2 Scheduling of unrelated parallel machines: example 2

In this case, the PWB scheduling problem was formulated as a multi-objective problem considering three objectives to be minimized: minimize overtime, minimize average finish time and minimize the total cost. The multi-objective formulation is as follows:

$$\min \sum_{i=1}^m O_i, \quad \min \mu_c, \quad \min \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij}$$

subject to:

$$\sum_{i=1}^m x_{ij} = 1$$

$$O_i = \max \left(\sum_{j=1}^n p_{ij} x_{ij} - T, 0 \right)$$

$$C_i = \sum_{j=1}^n p_{ij} x_{ij}$$

$$\mu_C = \frac{\sum_{i=1}^m C_i}{m}$$

The processing times and the processing costs are shown in Tables 4.9 and 4.10 respectively. The release interval time, T , is equal to 4 time units.

Table 4.9. Processing times for PWB scheduling problem

Machine	Lot 1	Lot 2	Lot 3	Lot 4	Lot 5	Lot 6	Lot 7
M1	2.5	1.3	2.7	4.3	3.5	1.8	1.7
M2	1.8	1.4	1.2	5.2	3.0	2.4	1.6
M3	3.8	2.2	3.4	6.8	4.4	6.2	4.4
M4	3.0	1.0	1.4	4.6	2.8	2.2	1.2
M5	5.2	2.4	6.2	8.4	6.6	7.0	4.8

Table 4.10 Processing costs for PWB scheduling problem

Machine	Lot 1	Lot 2	Lot 3	Lot 4	Lot 5	Lot 6	Lot 7
M1	16	24	18	22	26	20	22
M2	22	18	30	20	28	18	26
M3	12	12	15	18	22	12	16
M4	18	28	16	26	34	18	18
M5	14	14	11	16	18	10	12

The multi-objective scheduling of PWB problem was initially solved using the NSGA-II algorithm, to determine a Pareto-optimal set; with a population size of 500 and the algorithm was run for 150 generations, with a probability of crossover of 0.8, and probability of mutation of 0.02. There were 48 solutions in the Pareto-optimal set. The post-Pareto analysis was then performed on these 48 solutions to provide the decision-maker a workable sub-set of solutions by using the two proposed methods.

4.6.2.2.1 Pruned results by using the non-numerical ranking preference method: example 2

Three objectives were considered to be minimized in this example: overtime (O), average finish time (AFT) and cost (C). The non-numerical ranking preference combination selected to illustrate this example is the case in which overtime is more

important than cost, which is more important than average finish time ($O \succ C \succ \text{AFT}$: $w_1 > w_3 > w_2$). These preferences were selected to demonstrate the pruning process. Figure 4.19 shows the 48 solutions contained in the Pareto-optimal set.

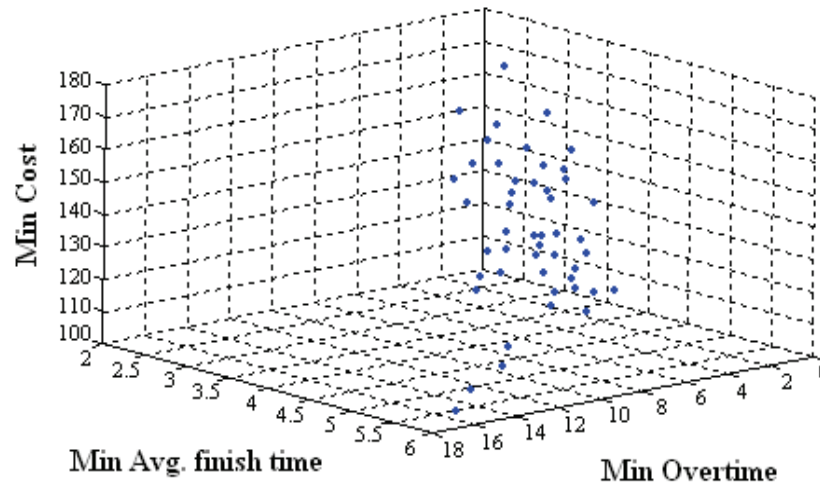


Figure 4.19 Pareto-optimal set of Example 2

Table 4.11 shows the pruned solutions obtained by applying the proposed method to narrow the search space using 5000 weights sets randomly selected from $f_w(\mathbf{w})$. Of the original 48 solutions, the pruned set only includes 2. Of the two solutions found in the pruned set, solution 48 has the lowest overtime; however it is achieved at the highest cost. In contrast, solution 26 presents the minimum cost but it possesses the highest overtime.

Table 4.11 Pruned solutions

Ranking preferences: $w_1 > w_3 > w_2$			
Sol No.	Minimize		
	Overtime	Avg. finish time	Cost
26	4	4.62	116
48	0.9	3.58	132

The pruned solutions obtained, considering the $f_1 \succ f_3 \succ f_2$ objective function preference, are shown in triangles in Figure 4.20 in two different two-dimensional

perspectives. By using this method, a 95.83% reduction was achieved in the solutions obtained from the Pareto-optimal set. These pruned solutions would then be further analyzed by the decision-maker. Solution 48 is shown as an example of a schedule for the PWB scheduling problem in Figure 4.21.

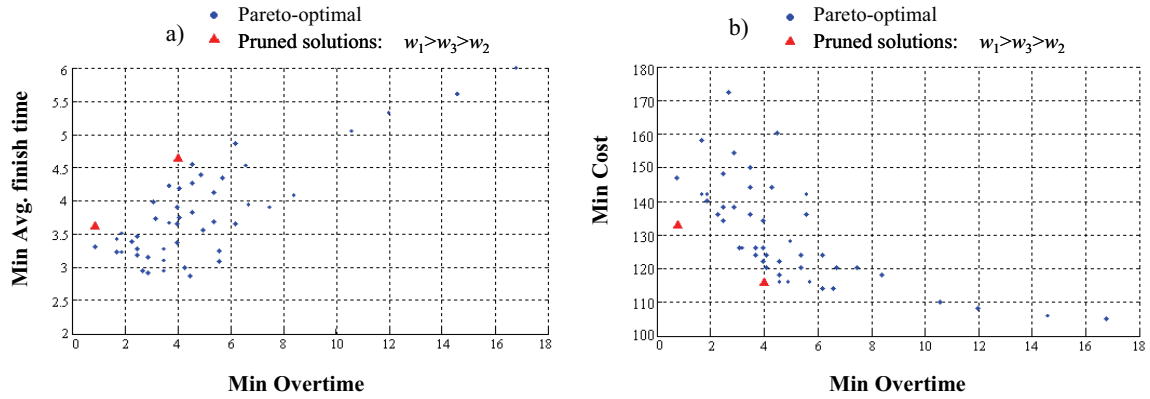


Figure 4.20 Pruned solutions for the $f_1 \succ f_3 \succ f_2$ objective function preference in a two-dimensional space

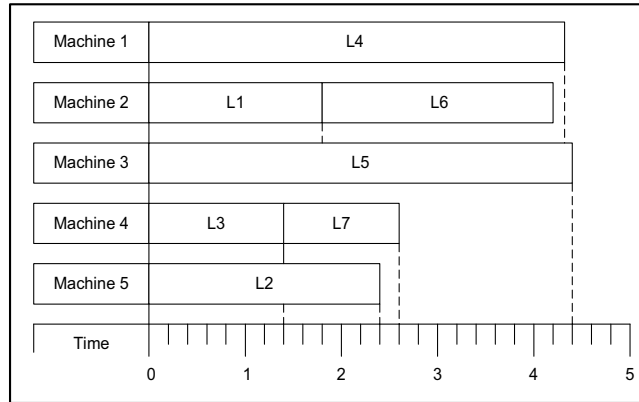


Figure 4.21 Schedule for solution 48

For this example, the two solutions found in the pruned Pareto set are the best solutions that reflect the decision-maker's $f_1 \succ f_3 \succ f_2$ objective function preference. Ten simulation runs were performed using the developed pruning algorithm. In each of the ten runs, 5000 different weights sets, randomly selected from $f_w(\mathbf{w})$, were used. Table 4.12 shows the results for the ten simulation runs. In this table, we can observe that given this objective function preference, besides the two solutions found in the pruning set, there is

no other solution in the Pareto-optimal set that satisfies this objectives preference. The other solutions that do not appear on the table it is because they had a counter value of zero, meaning that they did not minimize the value of f' : $f' = w_1f_1(x) + w_2f_2(x) + w_3f_3(x)$.

Table 4.12 Solutions found in the pruned Pareto set in ten simulation runs

Solution #	Counter on Simulation Run #									
	1	2	3	4	5	6	7	8	9	10
26	97	93	86	88	94	101	90	92	95	98
48	4903	4907	4914	4912	4906	4899	4910	4908	4905	4902

4.6.2.2.2 Pruned results by using data clustering: example 2

Using the k -means algorithm, clustering analysis was performed on the 48 solutions contained in the Pareto-optimal set and we found $k = 3$ to be the optimal number of clusters with the aid of the silhouette plots. Figure 4.22 shows the three clusters in a normalized space.

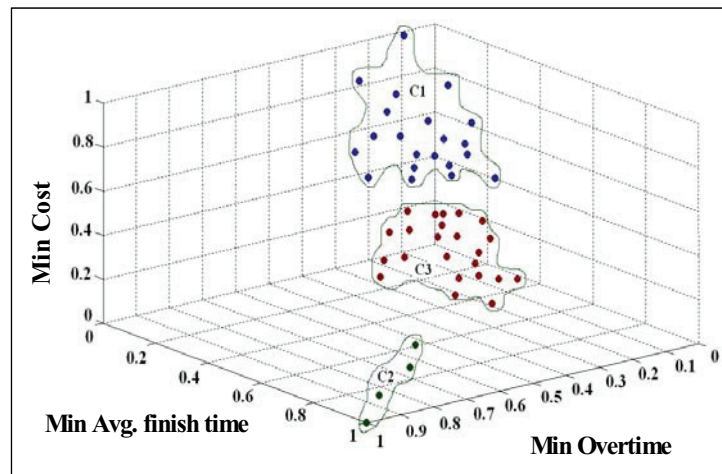


Figure 4.22 Clustered data for the second PWB example

Table 4.13 shows the summary of the results obtained by using the k -means algorithm. This table includes the representative solutions that were closest to their corresponding centroid. As we can see from Table 4.13, solution number 31 from cluster 1 gives the minimum overtime and average finish time but it has the highest cost. On the other hand, solution 2, from cluster 2, has the lowest cost but it also gives the highest

overtime and average finish time. In this case, solution 15, which is found in the “*knee cluster*”, seems to be a good compromise.

Table 4.13 Results obtained with the cluster analysis

	# of solutions	Representative Solution	Min Overtime	Min Avg. Finish time	Min Cost
Cluster 1	21	31	3.5	3.1	144
Cluster 2	4	2	14.6	5.6	106
Cluster 3	23	15	5.4	4.12	120

For this second example, in the first pruning method, the decision-maker only needs to consider two solutions (instead of 48). On the other hand, in the second pruning method, the decision-maker only needs to analyze three solutions. However, as it was mentioned earlier, for the first pruning method the decision-maker needs to be able to state the objective function preferences, while for the second pruning method there is no such need. Thus, this second method is helpful for those decision-makers with less knowledge about the problem domain. However, in the later method, once clustered the Pareto set, the decision-maker can analyze only the solutions found in the “*knee cluster*” and/or directly choose the representative solution of this cluster. In this manner, both methods, offer different advantages as reduction or pruning techniques.

4.7 Summary

In this chapter we saw that a popular method of “solving” multi-objective problems is to determine a Pareto-optimal set or sub-set. However, this then requires the decision-maker to select from among this set of solutions, which is often large when there is more than two objective functions. **To reduce or limit intelligently the size of the Pareto-optimal set, two methods were presented: 1) pruning by using non-numerical objective function ranking preferences method, and 2) pruning by using data clustering.**

The **main difference between the two pruning methods** is that the first pruning method is most appropriate when the decision-maker can prioritize their objectives, while the second does not require the decision-maker to *a priori* specify any objective function preference.

The first method, **pruning by using non-numerical ranking preferences**, provides the decision-maker a set of solutions that match his/her preferences and compare solutions with different objective function combinations. This method, in contrast with the weighted sum method, does not require the decision-maker to specify precise weight values or equivalent cost metrics. The different weight combinations are generated using the weight function, $f_w(\mathbf{w})$, which reflects the decision-maker preferences.

In the second method, **pruning by using data clustering**, we made use of clustering techniques used in data mining. In this case, we grouped the Pareto-optimal solutions by using the k -means algorithm to find groups of similar solutions. A clustering validation technique has been integrated into the k -means clustering algorithm to give a relatively automatic clustering process. The only parameters defined by the user are the maximum number of clusters to be analyzed and the desired number of replicates. This is to avoid the bias due to the selection of the initial centroids that has been observed. This was performed by selecting different values as initial centroids, and then, comparing the results obtained until a minimum was found. To determine the “optimal” or preferred number of clusters, k , in the set, the silhouette method was applied. A value of the silhouette width, $s(i)$, was obtained for the several values of k desired to investigate. The clustering with the highest average silhouette width, GS_{μ} , was selected as the “optimal” or preferred number of clusters in the Pareto-optimal set.

With this second approach, the decision-maker obtained a pruned Pareto-subset of just k particular solutions. Moreover, clustering analysis was useful to focus search on the

“knee” region of the Pareto front. The “knee” region is characterized by those solutions of the Pareto-optimal set where a small improvement in one objective would lead to a large deterioration in at least one other objective. The clusters formed in this region contain those solutions that are likely to be more relevant for the decision-maker. By using this approach the decision-maker is not required to specify any objective function preferences

The **two methods were demonstrated on two well-known multi-objective problems**: the redundancy allocation problem (RAP) and the scheduling of the bottleneck operation of a Printed Wiring Board (PWB) manufacturing line.

The two pruning methods presented in this Chapter were applied after the determination of Pareto-optimal sets by a MOEA. However, in Chapter 8, the non-numerical ranking preferences method is incorporated within the MOEA code. This aspect makes the analysis of the solution of multiple objective problems more efficient.

5. Developed MOEA for design allocation problems

In this chapter, a new MOEA for solving system design allocation problems is developed and tested. The algorithm uses a GA based on rank selection and elitist reinsertion, and a modifying genetic operator constraint handling method. Because GAs are appropriate for high-dimension stochastic problems with many nonlinearities or discontinuities, they are suited for solving reliability design problems. This new MOEA will be combined with the post-Pareto screening methods to develop a new approach to multiple objective optimization and it will be extended to develop an entirely new MOEA with integrated solution screening to achieve a balance between existing methods.

The developed algorithm, MOEA-DAP (Taboada & Coit, 2006b), mainly differs from other MOEAs in the crossover operation performed and in the selection procedure. In the crossover step, several offspring are created through multi-parent recombination. As a result, from n parents, denoted as N_{par} , considered for mating, in our algorithm there are $s \times [N_{par}(N_{par}-1)]$ number of children produced, where s is the number of subsystems considered. Thus, the mating pool contains a great amount of diversity of solutions. This disruptive nature of our proposed type of crossover, called subsystem rotation crossover (SURC) appears to encourage the exploration of the search space.

Additionally, for the selection step, the *fitness metric 1* proposed is based on the cumulative Euclidean distance from one solution to the rest of the nondominated solutions. In this way, we assign the highest fitness to those nondominated solutions that

are the farthest away with respect to the rest of the solutions. This is also to promote a good spread along the Pareto front.

To validate the performance of the developed MOEA-DAP, ten experimental runs were obtained from our algorithm and they are compared against ten runs obtained from one of the most successful evolutionary algorithms that currently exists: the NSGA-II algorithm. The NSGA-II algorithm is noticeably more efficient than its previous version (NSGA), but there are possible concerns regarding its exploratory capability. Although the algorithm tends to spread quickly and appropriately when a certain nondominated region is found, it seems to have difficulties in generating nondominated solution vectors that lie in certain (isolated) regions of the search space (Coello Coello & Toscano Pulido, 2001).

As it will be described in Section 5.9, the results of the performance comparison based on two different performance metrics, Overall Nondominated Vector Generation (*ONVG*) and Overall true Nondominated Vector Generation (*OTNVG*), indicate that our proposed algorithm is more effective for solving this type of problem. The MOEA-DAP algorithm obtains more solutions that contribute to the true Pareto-optimal front, and the solutions obtained are also more uniformly distributed along the Pareto frontier than those solutions coming from the NSGA-II.

5.1 Description of the problem addressed

This chapter describes the use of a multiple objective evolutionary algorithm to solve engineering design allocation problems. The problem addressed in this chapter arises in many real engineering optimization problems, where managers and/or decision-makers have to efficiently allocate components from among of a set of predefined component

choices to determine the optimal configuration to be implemented. There are numerous application areas of the redundancy allocation problem, such as in the case of electrical power systems (Ouidir *et al.*, 2004), transportation systems (Levitin & Lisnianski, 2001), telecommunications (Lyu *et al.*, 2002), among others.

This chapter addresses the problem of designing a hardware system structure. In the problem formulation presented, there is a specified number of subsystems and, for each subsystem, there are multiple component choices which can be selected and used in parallel. This formulation pertains to the well-known redundancy allocation problem (RAP). In this chapter, the RAP is modeled as a multi-objective problem with the system reliability to be maximized, cost and weight of the system to be minimized, and no constraints limiting the possible values of reliability, making this problem a multiple objective combinatorial optimization (MOCO) problem.

MOEA-DAP represents a new alternative for the solution of a difficult MOCO reliability-design problem. This new approach has the strength of a problem-oriented technique. The selection of components is advantageously combined to create a multiple objective evolutionary algorithm (MOEA) which can tackle the problem in the most efficient way. Since MOCO problems contain information derived from their specific combinatorial structure, this can be advantageously exploited during the search. To take advantage of the combinatorial structure within the search algorithm, a problem dependent customized crossover operator is used, called the subsystem rotation crossover (SURC)

To be most efficient, the solution of a multiple objective problem seems to necessarily require a hybrid algorithm, i.e., an integration of standard evolutionary

algorithms and problem dependent components. This is particularly true for MOCO problems, where the adaptation of a universal method to a problem can not compete with a method specifically designed for this problem.

5.2 Multi-criteria formulation of the RAP using GAs

Multicriteria formulations using GAs can be found over the Literature. Busacca *et al.* (2001) proposed a multi-objective GA approach that was applied to a safety system of a nuclear power plant. Huang *et al.* (2006) proposed a new method of system reliability multi-objective optimization using GAs. They considered a reliability optimization model obtained from a transformation of Dhingra's over-speed protection system model (1992) which contains two objective functions, simultaneously maximizing system reliability and minimizing system cost subject to limits on weight and volume.

As shown in Chapter 4, Taboada & Coit (2007) and Taboada *et al.* (2007a) formulated the redundancy allocation problem (RAP) as a multi-objective problem with the system reliability to be maximized, and cost and weight of the system to be minimized. The Pareto-optimal set was initially obtained using the fast elitist nondominated sorting genetic algorithm (NSGA-II) originally proposed by Deb *et al.* (2002). Then, the decision-making stage was performed by applying two proposed pruning methods to reduce the size of the Pareto-optimal set and obtain a smaller representation of the multi-objective design space. For those studies, NSGA-II was effective. However, NSGA-II is a general MOEA for any type of problem. This implies that the problem formulation needs to be adapted. Moreover, in these studies, the final Pareto front found by NSGA-II contained many repeated solutions, so to obtain a large number of solutions; several runs had to be performed. Thus, if a decision-maker must

solve many similar RAP problems, then a custom MOEA, especially designed to solve multi-objective design allocation problems, offers great advantage.

5.3 The proposed algorithm: MOEA-DAP

MOEA-DAP is a multiple objective evolutionary algorithm specifically designed for solving design allocation problems. The multi-objective formulation that we considered is shown in Equation 5.1, with the system reliability to be maximized, cost and weight of the system to be minimized, and no constraints in the possible values of reliability. In practice, MOEA-DAP could be generalized to accommodate any number of constraints.

$$\max \left[\prod_{i=1}^s R_i(\mathbf{x}_i) \right], \min \left[\sum_{i=1}^s \sum_{j=1}^{m_i} c_{ij} x_{ij} \right], \min \left[\sum_{i=1}^s \sum_{j=1}^{m_i} w_{ij} x_{ij} \right] \quad (5.1)$$

Subject to:

$$1 \leq \sum_{j=1}^{m_i} x_{ij} \leq n_{\max,i} \quad \text{for } \forall i = 1, 2, \dots, s$$

$$x_{ij} \in \{0, 1, 2, \dots\}$$

Where:

s = number of subsystems

$\mathbf{x}_i = (x_{i,1}, x_{i,2}, \dots, x_{i,m_i})$

x_{ij} = quantity of the j^{th} available component used in subsystem i

m_i = total number of available components for subsystem i

$n_{\max,i}$ = user defined maximum number of components in parallel used in subsystem i

$R_i(\mathbf{x}_i)$ = reliability of subsystem i

c_{ij}, w_{ij}, r_{ij} = cost, weight and reliability for the j^{th} available component for subsystem i

For the multi-objective RAP, the objectives for MOEA-DAP are to determine the optimal design configuration that maximizes system reliability, minimizes the total cost, and minimizes the system weight, for a series-parallel system. The pseudo-code and the

flowchart of the developed MOEA-DAP are shown below and in Figures 5.1, respectively. The specific steps are further explained in the following sections.

1. *[Start]* Generate random population of n chromosomes
2. *[Fitness]* Evaluate the aggregated fitness function of each chromosome x in the population
3. *[Selection]* With a given crossover probability select individuals with the highest aggregated fitness to perform recombination.
4. *[Crossover]* With a pre-defined crossover probability, cross-over the parents to form new offspring (children).
5. *[Mutation]* With a pre-defined mutation probability, mutate new offspring at a random position in the chromosome
6. *[Reinsertion]* Place new offspring + a specified percentage of the most elite parents to form the new population
7. *[Replace]* Use new generated population for a further run of the algorithm
8. *[Test]* If the Generation $i = \text{Generation 'max'}$, **stop**, and return the best solutions in current population, otherwise return to step 2.

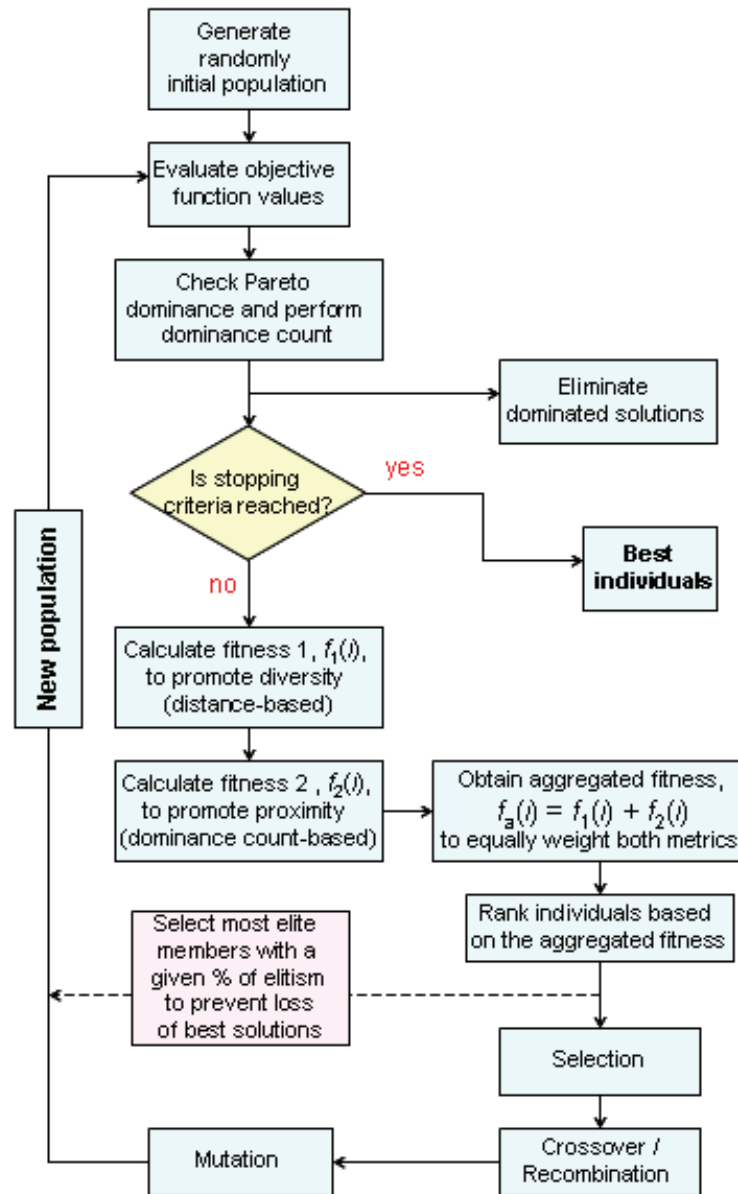


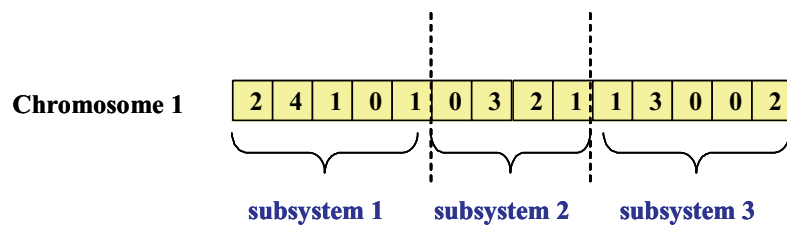
Figure 5.1 Flowchart of MOEA-DAP

5.3.1 Chromosomal representation

Although binary-coded GAs are commonly used, there is an increasing interest in alternative encoding strategies, such as integer and real-valued representations. For some problem domains, it is argued that the binary representation is in fact deceptive since it obscures the nature of the search (Bramlette, 1991).

Thus, in our MOEA we used an integer chromosomal representation. For instance, consider the following example to illustrate a chromosome used in our algorithm to solve the multi-objective-RAP problem.

The following chromosome contains fourteen integers for a configuration which consists of 3 subsystems, with an option of 5, 4 and 5 types of components in each subsystem, respectively, with 1 as the minimum number of components in each subsystem, for the system to function, and 8 as the maximum number of components in each subsystem. Each integer corresponds to the number of redundant components of that type. For example, for subsystem 1; two copies of the first component type, four components of the second component type, one copy of the third component type and one copy of the fifth component type are used in parallel. For subsystem 2; three copies of the second component type, two copies of the third component type and one copy of the fourth component type are used in parallel. And, finally, for subsystem 3; one copy of the first component type, three copies of the second component type and two copies of the fifth component type are used in parallel.



From above, it can be seen that for the multi-objective-RAP, the use of an integer representation provides a convenient and natural way of expressing the mapping from chromosomal representation to the problem domain. Alternatively, a binary representation would require 126 different 0-1 values.

5.3.2 Constraint-handling method

Our proposed MOEA uses the *modifying genetic operator strategy*. In this approach the genetic operators are crafted to always produce feasible solutions. The main reason we used this constraint handling strategy is because evolutionary computation techniques have huge potential for incorporating specialized operators that search the boundary of both, feasible and infeasible, regions in an efficient way. However, it is commonly acknowledged that restricting the size of the search space in evolutionary algorithms (as in most search algorithms) is generally beneficial. Hence, it seems natural in the context of constrained optimization to restrict the search for the solution to the boundary of the feasible part of the space (Michalewicz & Shoenauer, 1996).

The most common approach for GAs to accommodate constraints is to use penalty functions to penalize constraint violations. Although conceptually very simple, in practice it is quite difficult to implement, because the exact location of the boundary between the feasible and infeasible regions is unknown in most problems (Coello Coello, 2002).

The formulation addressed in the MOEA-DAP development had s constraints to limit the total components within each subsystem. If additional constraints are imposed (e.g., fuel consumption, volume), then the existing modifying genetic operator strategy could be readily expanded to accommodate these constraints by using repair operators.

5.3.3 Determination of the initial generation

EAs are population-based algorithms, thus, MOEA-DAP begins its search with a population of random solutions. Immediately, thereafter, objective function values are evaluated and the Pareto dominance criterion is checked in the initially created solutions. Those solutions that are dominated by other solutions are eliminated. Thus, in this way,

MOEA-DAP ensures that the resulting population will only contain Pareto-optimal solutions.

For instance, consider an example configuration which consists of four subsystems, with an option of three types of components in each subsystem, respectively, with one as the minimum number of components in each subsystem, for the system to function, and five as the maximum number of components in each subsystem.

First, a random initial population, N_{pop} , is created. That is, n number of strings are created. And, as explained in Section 5.2, only feasible solutions are created. The length of each string will depend on the number of options of different components in each subsystem, that is:

$$L = m(1) + m(2) + \dots + m(i) = \sum_i m(i)$$

Where:

L = length of string

$m(i)$ = number of available components for subsystem i

For the example that we are considering, say that 12 individuals are randomly created. Then, the objective functions values are calculated for the three objective functions. Next, **Pareto dominance criterion** is checked in this initial population. The problem is treated as a minimization problem, thus, reliability is multiplied by -1. Then, without loss of generality, for this minimization problem for all objectives, a solution \mathbf{x}_1 dominates a solution \mathbf{x}_2 , if and only if, the two following conditions are true:

- \mathbf{x}_1 is no worse than \mathbf{x}_2 in all objectives, i.e., $f_i(\mathbf{x}_1) \leq f_i(\mathbf{x}_2) \quad \forall i, \quad i \in \{1, 2, \dots, n\}$
- \mathbf{x}_1 is strictly better than \mathbf{x}_2 in at least one objective, i.e., $f_i(\mathbf{x}_1) < f_i(\mathbf{x}_2)$ for at least one i .

Before obtaining a count of how many of these 12 individuals are dominated, we count how many individuals are dominated by each of the 12 individuals forming the initial population. Table 5.1 shows the objective function values for the first 12 chromosomes created, and in the last column, the number of individuals that each individual dominates is presented. This is often referred as ‘dominance count.’

Table 5.1 Dominance count in initial population

First population	Reliability	Cost	Weight	Dominance count
1	-0.90112	76	81	0
2	-0.89779	63	78	0
3	-0.98734	60	75	3
4	-0.68228	69	73	0
5	-0.94814	52	75	3
6	-0.96703	55	97	0
7	-0.98699	68	62	3
8	-0.95267	52	80	1
9	-0.79624	29	41	1
10	-0.84937	68	76	0
11	-0.89678	42	56	2
12	-0.92673	46	68	4

As we can see from Table 5.1, solution number 12 is the most dominating solution, with a dominance count of 4. Then after this, we copy the nondominated solutions to a separate set and we discard those solutions that are dominated. Thus, Table 5.1, is now reduced to Table 5.2.

Table 5.2 Dominance count in the first nondominated set

Nondominated set	Reliability	Cost	Weight	Dominance count
1	-0.98734	60	75	3
2	-0.94814	52	75	3
3	-0.96703	55	97	0
4	-0.98699	68	62	3
5	-0.95267	52	80	1
6	-0.79624	29	41	1
7	-0.89678	42	56	2
8	-0.92673	46	68	4

As we can see from Table 5.2, solution number 3 does not dominate any of the solutions but it still is a nondominated solution. To easily explain this phenomenon,

consider the case of a bi-objective problem presented in Figure 5.2, in which we try to minimize weight and minimize cost. Certainly, from the Figure 5.2, and from the information in Tables 5.3 and 5.4, we can observe that solution number 1 pertains to the nondominated set of solutions although it does not dominate any solution.

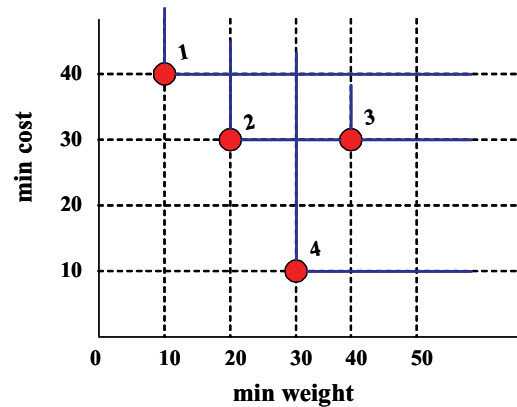


Figure 5.2 Dominance in a bi-objective problem

Table 5.3. Initial solutions

solution	weight	cost	Dominance count
1	10	40	0
2	20	30	1
3	40	30	0
4	30	10	1

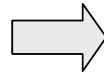


Table 5.4. Nondominated solutions

solution	weight	cost	Dominance count
1	10	40	0
2	20	30	1
4	30	10	1

5.3.4 Aggregated fitness function

After obtaining the nondominated solutions and having each of these marked with its corresponding dominance count, we proceed to assign fitness to these solutions. In our algorithm, we used two different methods to assign fitness to the solutions. The first fitness, $f_1(i)$, is intended for maintaining population diversity. The second fitness, $f_2(i)$, aims to select those individuals which are more dominating. The two different fitness metrics are then aggregated weighting each of the fitness metrics equally, aiming to achieve proximity and diversity, which are the two most common desirable

characteristics in MOEAs. The two fitness metrics used in the algorithm are described next.

5.3.4.1 Fitness metric 1: distance-based, $f_1(i)$

This fitness metric gives highest fitness to those solutions that are farther away from other solutions in the Pareto front, giving those solutions a greater possibility to be chosen later for reproduction. With this fitness function, we aim to maintain diversity of the Pareto optimal solutions. To illustrate how this fitness works, let's continue considering the information from Table 5.2, which contains only those solutions that were nondominated. The next step consists in standardizing the solutions. The three objective functions were scaled from 0 to 1 using the following equation:

$$\frac{f_i(\mathbf{x}) - f_i^{\min}}{f_i^{\max} - f_i^{\min}} \quad \forall i = 1, 2, \dots, n$$

where f_i^{\min} = minimum value for $f_i(\mathbf{x})$ in the nondominated set.

f_i^{\max} = maximum value for $f_i(\mathbf{x})$ in the nondominated set.

As we can notice, we need to multiply column 2 by (-1) to be able to use the above equation properly. Now, the 8 nondominated solutions from Table 5.2 are presented as Table 5.5:

Table 5.5. Standardized nondominated set

Nondominated set	Reliability	Cost	Weight	Dominance count
1	1	0.79487	0.60714	3
2	0.79487	0.58974	0.60714	3
3	0.89369	0.66667	1	0
4	0.99813	1	0.375	3
5	0.81856	0.58974	0.69643	1
6	0	0	0	1
7	0.52609	0.33333	0.26786	2
8	0.68281	0.4359	0.48214	4

From here, we compute the Euclidean distance from each solution to the others, i.e., from solution 1 to the rest of the solutions, from solution 2 to the rest of the solutions, and so on. Next, the sum of the distances from each solution to the rest of the solutions is obtained, and the maximum and minimum value of all the sums is determined. Table 5.6 presents these calculations. Since for calculating fitness 1, we do not use the fifth column of Table 5.5, we can just keep it aside.

Table 5.6. Distance from each solution to the rest of the solutions

solution	1	2	3	4	5	6	7	8
1	0	0.290090	0.42670	0.30979	0.288050	1.41440	0.74345	0.49507
2	0.29009	0	0.41233	0.51334	0.092373	1.16110	0.50310	0.22771
3	0.42670	0.412330	0	0.71599	0.322050	1.49770	0.88447	0.60490
4	0.30979	0.513340	0.71599	0	0.551250	1.46180	0.82386	0.65507
5	0.28805	0.092373	0.32205	0.55125	0	1.22590	0.57875	0.29667
6	1.41440	1.161100	1.49770	1.46180	1.22590	0	0.67796	0.94271
7	0.74345	0.503100	0.88447	0.82386	0.57875	0.67796	0	0.28461
8	0.49507	0.227710	0.60490	0.65507	0.29667	0.94271	0.28461	0
$\sum_i d(i, j)$	3.9675	3.2001	4.8642	5.0311	3.3551	8.3816	4.4962	3.5067
$\max_j \sum_i d(i, j)$						8.3816		
$\min_j \sum_i d(i, j)$		3.2001						

To calculate the fitness that we are going to assign to each solution, from Table 5.6, the difference (*Diff*) between maximum and minimum value of the sum of distances is initially obtained.

$$Diff = \max_j \sum_i d(i, j) - \min_j \sum_i d(i, j)$$

Where $d(i, j)$ is the Euclidean distance between i and j .

Our algorithm considers five possible intervals for each of this fitness 1 metric, then this difference is divided by five and we obtain the increment, denoted by I .

$$I = Diff / \text{number of intervals}$$

Thus, the values in which each of the fitness 1, $f_1(i)$, ranges can be calculated as follows:

$$f_1(i) = \begin{cases} 1, & \text{if } \min_j \sum_i d(i, j) \leq \sum_i d(i, j) < \min_j \sum_i d(i, j) + I \\ 2, & \text{if } \min_j \sum_i d(i, j) + I \leq \sum_i d(i, j) < \min_j \sum_i d(i, j) + 2I \\ 3, & \text{if } \min_j \sum_i d(i, j) + 2I \leq \sum_i d(i, j) < \min_j \sum_i d(i, j) + 3I \\ 4, & \text{if } \min_j \sum_i d(i, j) + 3I \leq \sum_i d(i, j) < \min_j \sum_i d(i, j) + 4I \\ 5, & \text{if } \min_j \sum_i d(i, j) + 4I \leq \sum_i d(i, j) \leq \max_j \sum_i d(i, j) \end{cases}$$

In this example case, the difference is 5.1815, thus the increment is 1.0363. Table 5.7 shows the fitness 1 metric bounds, as well as the assigned fitness to each of the 8 nondominated solutions. This means that, from the nondominated set, now we have the following fitness 1 values for each of the solutions presented in Table 5.8.

Table 5.7. Fitness value 1 for the nondominated set

Fitness 1 value	Fitness 1 value ranges	solution
1	$3.2001 \leq \sum_i d(i, j) < 4.2364$	1, 2, 5 and 8
2	$4.2364 \leq \sum_i d(i, j) < 5.2727$	3, 4 and 7
3	$5.2727 \leq \sum_i d(i, j) < 6.3090$	0
4	$6.3090 \leq \sum_i d(i, j) < 7.3453$	0
5	$7.3453 \leq \sum_i d(i, j) \leq 8.3816$	6

Table 5.8. Fitness value 1 for the nondominated set (standardized space)

Nondominated set	Reliability	Cost	Weight	Fitness value 1
1	1	0.79487	0.60714	1
2	0.79487	0.58974	0.60714	1
3	0.89369	0.66667	1	2
4	0.99813	1	0.375	2
5	0.81856	0.58974	0.69643	1
6	0	0	0	5
7	0.52609	0.33333	0.26786	2
8	0.68281	0.4359	0.48214	1

From Table 5.8, it can be noticed that we are assigning the highest fitness to those nondominated solutions that are the farthest away with respect to the rest of the solutions. This is to promote diversity in the population. This is critical to prevent premature convergence and to assure that the solution space is thoroughly considered. In this way, during the selection step, solution number 6 has highest possibilities of being chosen for reproduction, directing the search to the least crowded areas of the search space.

In Figure 5.3, it is observed that, as expected, solution number 6 is the farthest of all solutions. It can be noticed also that the rest of the solutions are almost equally separated. In Figure 5.4, the solutions are plotted in a bi-objective space to better visualize the solutions.

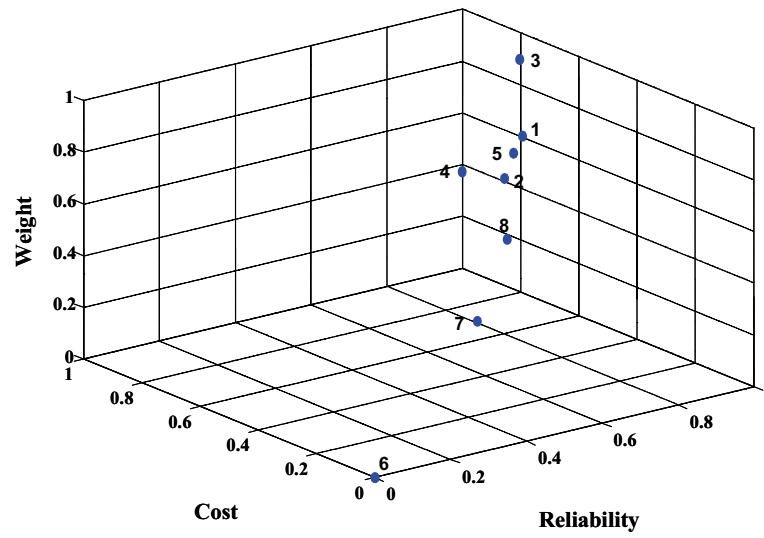


Figure 5.3. First nondominated set in a normalized space

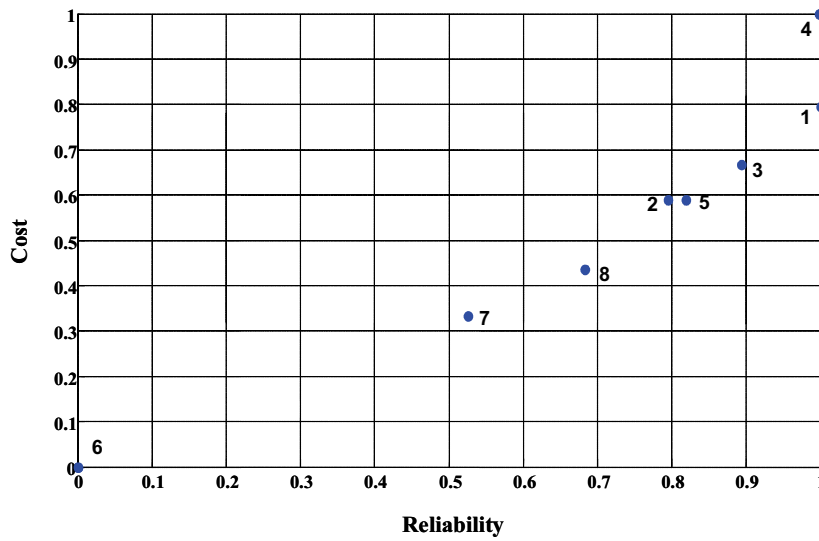


Figure 5.4. First nondominated set in a bi-objective normalized space

5.3.4.2 Fitness metric 2: dominance count-based, $f_2(i)$

The second fitness metric, $f_2(i)$, is based on the dominance count concept. Determination of the intervals for this metric is similar to the one explained previously for Fitness Metric 1. Using the information from Table 5.5, maximum and minimum values of dominance count are calculated for the nondominated solutions. Table 5.9 presents the data for the example.

Table 5.9. Dominance count in the nondominated set.

Nondominated set	Reliability	Cost	Weight	Dominance Count $DC(i)$
1	1	0.79487	0.60714	3
2	0.79487	0.58974	0.60714	3
3	0.89369	0.66667	1	0
4	0.99813	1	0.37500	3
5	0.81856	0.58974	0.69643	1
6	0	0	0	1
7	0.52609	0.33333	0.26786	2
8	0.68281	0.43590	0.48214	4
$\max_i DC(i)$				4
$\min_i DC(i)$				0

In this case, the difference and the increment are calculated as follows:

$$Diff = \max_i DC(i) - \min_i DC(i)$$

For this metric we also consider discrete intervals. Five intervals were chosen for the example. The increment (I) is calculated in the same way as in fitness metric 1. Thus, the values in which each of the fitness i ranges can be calculated as follows:

$$f_2(i) = \begin{cases} 1, & \text{if } \min_i DC(i) \leq DC(i) < \min_i DC(i) + I \\ 2, & \text{if } \min_i DC(i) + I \leq DC(i) < \min_i DC(i) + 2I \\ 3, & \text{if } \min_i DC(i) + 2I \leq DC(i) < \min_i DC(i) + 3I \\ 4, & \text{if } \min_i DC(i) + 3I \leq DC(i) < \min_i DC(i) + 4I \\ 5, & \text{if } \min_i DC(i) + 4I \leq DC(i) \leq \max_i DC(i) \end{cases}$$

In this case, the difference is 4, thus the increment is 0.8. Table 5.10 shows the fitness 2 metric bounds, as well as the assigned fitness to each of the 8 nondominated solutions.

This means that, from the nondominated set now we have, as in Table 5.11, the following fitness 2 values for each of the solutions:

Table 5.10. Fitness value 2 bounds for the nondominated set

Fitness 1 value	Fitness 2 value ranges	solution
1	$0 \leq DC(i) < 0.8$	3
2	$0.8 \leq DC(i) < 1.6$	5 and 6
3	$1.6 \leq DC(i) < 2.4$	7
4	$2.4 \leq DC(i) < 3.2$	1, 2 and 4
5	$3.2 \leq DC(i) \leq 4.0$	8

Table 5.11. Fitness value 2 for the nondominated set (standardized space)

Nondominated set	Reliability	Cost	Weight	Fitness value 2
1	1	0.79487	0.60714	4
2	0.79487	0.58974	0.60714	4
3	0.89369	0.66667	1	1
4	0.99813	1	0.375	4
5	0.81856	0.58974	0.69643	2
6	0	0	0	2
7	0.52609	0.33333	0.26786	3
8	0.68281	0.4359	0.48214	5

In this way, the highest fitness is assigned to those solutions that are more dominating. Now, that the two fitness metrics that our multi-objective optimization have been explained, the selection step takes place.

5.3.5 Selection step

In the selection step, rank selection was used. In Whitley (1989) and Bäch & Hoffmeister (1991), it has been observed that rank-based fitness assignment overcomes the scaling problems of the proportional fitness assignment. Ranking introduces a uniform scaling across the population and provides a simple and effective way of controlling selective pressure. Moreover, rank-based fitness assignment behaves in a more robust manner than proportional fitness assignment, and thus, is an appropriate method to choose.

In our multi-objective optimization algorithm, values assigned in fitness metric 1 (Table 5.8) and values in fitness metric 2 (Table 5.11) are summed to form the total fitness value, as shown in Table 5.12.

Table 5.12. Total Fitness value of nondominated solutions

Solution	Fitness metric 1 $f_1(i)$	Fitness metric 2 $f_2(i)$	Aggregated Fitness value $f_a(i) = f_1(i) + f_2(i)$
1	1	4	5
2	1	4	5
3	2	1	3
4	2	4	6
5	1	2	3
6	5	2	7
7	2	3	5
8	1	5	6

Then, in rank selection every nondominated individual receives an aggregated fitness value, $f_a(i)$, determined by their ranking from fitness metric 1 and 2. From here, the nondominated solutions are ranked in descending order, according to their aggregated fitness value $f_a(i)$ as in Table 5.13.

Table 5.13. Ranked nondominated individuals

Aggregated Fitness value $f_a(i)$	Ranked Solutions
7	6
6	4
6	8
5	1
5	2
5	7
3	3
3	5

As can be seen, solutions with the same aggregated fitness value, $f_a(i)$, are allowed, and thus, the worst individuals have the lowest total fitness values, and the best have the highest total fitness value. Then, the number of individuals to be selected for the recombination step is dictated by the desired crossover probability. For instance, in Table 5.13, we have 8 nondominated solutions (*Nnon-dom*). If the crossover probability,

P_{cross} , is 0.7, then the number of solutions to be selected to perform recombination will be $N_{parents} = \text{round}(P_{cross} \times N_{non-dom})$. In this case, there are: $\text{round}(0.7 \times 8) = 6$ parents.

5.3.6 Crossover operator

To be efficient, an approximation method for solving a MOP seems to be necessarily an hybrid algorithm, i.e., a combination of evolutionary algorithms and problem dependent components. This is particularly true for MOCO problems, where the adaptation of a universal method to a problem can generally not compete with a method specifically designed for this problem. For the exploitation of the combinatorial structure within the search algorithm, the problem dependent component used in this paper is the specific crossover operator: subsystem rotation crossover (SURC). In this step, multi-parent recombination is allowed. This action, and the way that SURC works, produces a large number of children in the mating pool, creating a large number of diverse solutions to choose from. Diversity is considered favorable, as the greater the variety of genes available to the genetic algorithm, the greater the likelihood of the system identifying alternate solutions. Moreover, maintaining diversity of individuals within a population is necessary for the long term success of any evolutionary system. This customized crossover operator is fully described next.

In MOEA-DAP, each solution, represented as a chromosome, has s number of sub-chromosomes. That is, if we want to solve a problem with three subsystems, as in Figure 5.5, then each individual that can be selected for recombination has three sub-chromosomes.

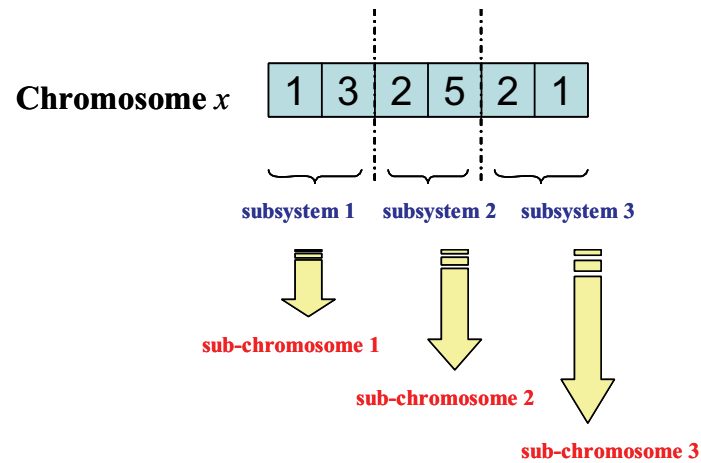


Figure 5.5. Sub-chromosome representation of individuals

To illustrate how the recombination operation was performed, consider the case in which three parents were selected for cross-over. As explained above, each parent has, in this case, three sub-chromosomes. Then to produce the children, we kept fixed subchromosomes 2 and 3 in the three parents and we perform circular rotation of sub-chromosomes 1, as shown in Figure 5.6.

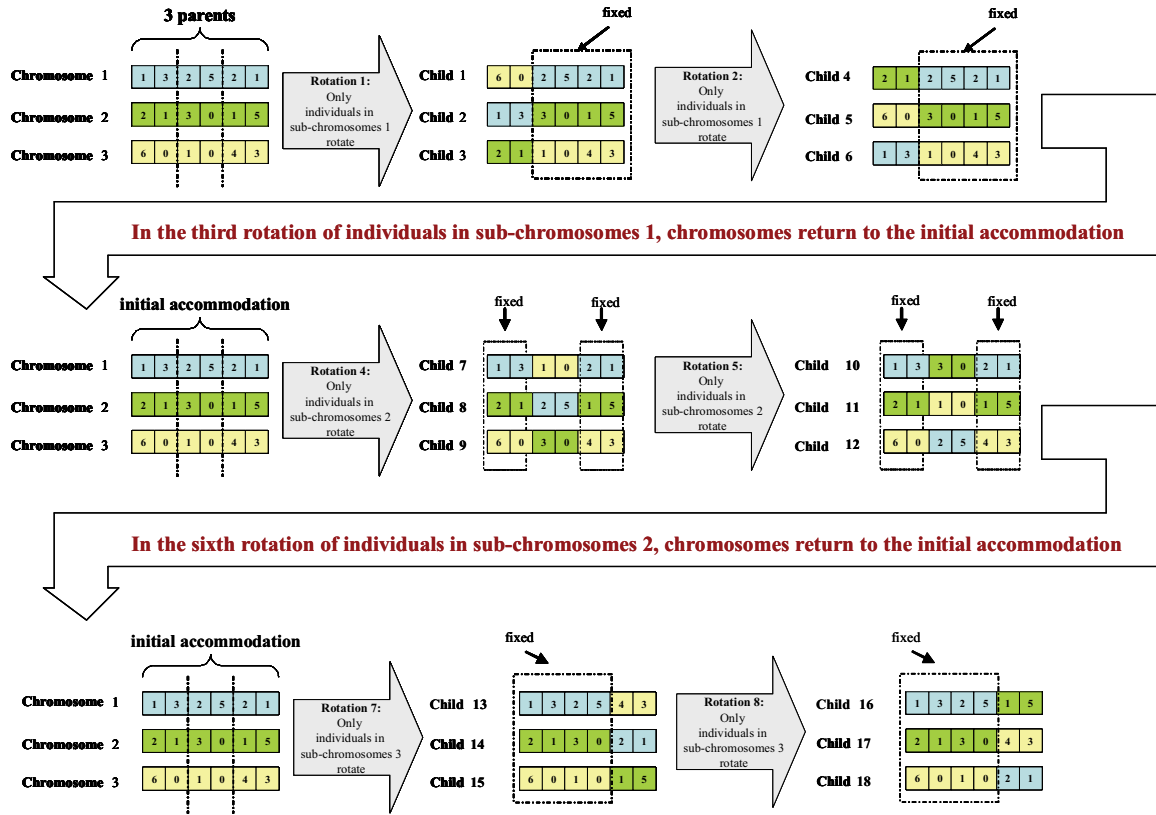


Figure 5.6 Crossover Operation Example

Thus, as we can see, from only three parents, there are $(3)(2)(3) = 18$ children in the mating pool. To calculate the number of children in the mating pool, we use the following equation:

$$Nmp = s [Npar (Npar - 1)]$$

Where:

Nmp = number of children in the mating pool

$Npar$ = number of parents

s = number of subsystems

The next decision is to determine how many of these children in the mating pool are to be selected to undergo mutation. This number is going to depend on the crossover probability and on the percentage of elitism ($\%elitism$) desired. To prevent the loss of the best found solutions and to increase the performance of our algorithm, we keep a percentage of the best nondominated solutions (those with the highest aggregated fitness,

$f_a(i)$ for next generation. These individuals are directly copied to the elite list and they form part of the population in next generation.

For instance, consider a case in which the input parameters are as follows: $N_{pop}=20$, $P_{cross}=0.75$, $\%elitism=25\%$, $s=3$. Let's say that after checking Pareto dominance, only 16 individuals are nondominated. Then these individuals are ranked in descending order of aggregated fitness value, $f_a(i)$, and with 25% of elitism, the best $0.25 \times 16 = 4$ individuals, denoted as N_{elite} , are selected to form the next population. With the specified crossover probability of 0.75, $16 \times 0.75 = 12$ individuals are selected as parents to perform crossover. After the procedure explained above, the recombination is carried out, and now we have in the mating pool $3(12 \times 11) = 396$ children produced. Thus, the number of children selected from the mating pool to form the next population is:

$$N_{csm} = N_{pop} - N_{elite}$$

Where:

N_{csm} = number of children selected from the mating pool

N_{pop} = population size

N_{elite} = number in the elite list

In this case, 16 children are randomly selected to undergo mutation, and finally, these individuals plus the ones already in the elite list will form the next population.

5.3.7 Mutation

There are no universally accepted general rules to choose the values of basic GA operators for solving specific optimization problems. However, numerous experimental studies have developed some rules of thumb concerning ranges of GA parameters. For example, in the case of choosing the mutation rate, De Jong (1975) suggests that the mutation probability, which is a bit reversal event, should occur with small probability,

$p_{mut} \approx 0.001$. Grefenstette (1986) suggests a $p_{mut} \approx 0.01$, while in Schaffer *et al.* (1989) a range is considered, $p_{mut} \in [0.005, 0.01]$. Hence, in MOEA-DAP, with a specified probability of mutation, one gene in the chromosome is selected at random and that gene undergoes mutation. Figure 5.7 shows the genetic string for an example child, before and after mutation.

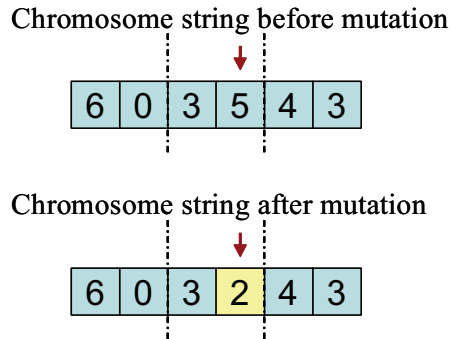


Figure 5.7. Example of mutation

Since our proposed MOEA uses the modifying genetic operator strategy as a constraint handling method, we are always interested in producing only feasible solutions. Consider, for instance, if the maximum number of components in each subsystem is 8 and if the gene selected to mutate is gene number 4, as shown in Figure 5.7, then we can only change that value (currently 5) to a smaller number and not for a higher one, in order to not violate the constraint on the maximum number of components allowed per subsystem. Notice that, if the gene selected to mutate had been gene number 1, and if after mutating, the value randomly generated would have been a zero, then subsystem number 1 would no longer have any component. Thus, when this is the case, the algorithm creates another random number to ensure that all subsystems have components, and thus the evaluation of reliability can properly be obtained.

5.3.8 Elitist reinsertion

As explained in the crossover step, our algorithm uses elitist reinsertion in the aim of preventing the loss of the best solutions. The user specifies the desired percentage of elitism, $\%elitism$, and individuals from the previous population that contain the ranked nondominated individuals (see Table 5.13) are chosen to pertain to an elite list according with this percentage.

5.4 Performance comparison of MOEA-DAP

To validate the performance of the developed MOEA-DAP, ten experimental runs are obtained from our algorithm and they are compared against ten runs obtained from one of the most successful evolutionary algorithms that currently exists: NSGA-II. For each run, both algorithms used the following input parameters: $Npop=50$, $Generations=100$, $P_{cross}=0.8$, $P_{mut}=0.007937$; and additionally our algorithm considered $\%elitism=60\%$. The output of the individual runs can be found in Appendix A.

The example considered consists of a configuration of 3 subsystems, with an option of 5, 4 and 5 types of components in each subsystem, respectively. The optimization involves selection from among these component types. The minimum number of components in each subsystem is 1, for the system to function, and the maximum number of components is 8 in each subsystem. Table 5.14 defines the component choices for each subsystem.

Table 5.14. Component choices for each subsystem

Design Alternative j	Subsystem i								
	1			2			3		
	R	C	W	R	C	W	R	C	W
1	0.94	9	9	0.97	12	5	0.96	10	6
2	0.91	6	6	0.86	3	7	0.89	6	8
3	0.89	6	4	0.70	2	3	0.72	4	2
4	0.75	3	7	0.66	2	4	0.71	3	4
5	0.72	2	8				0.67	2	4

As happens in most multi-objective optimization problems, the true Pareto front is not known, and then a good approximation of the true Pareto-optimal front, Y_{true} , can be built by gathering all non-dominated individuals from all sets or runs. In other words, for the performance comparison that is presented next, the real Pareto-optimal front is approximated by the best known solutions of all our experiments.

Two metrics are used to compare the performance of the MOEA-DAP algorithm against the NSGA-II. The following two metrics, as in Van Veldhuisen (1999), are the Overall Nondominated Vector Generation (*ONVG*) and Overall true Nondominated Vector Generation (*OTNVG*).

1. Overall Nondominated Vector Generation (*ONVG*): simply counts the number of solutions in the Pareto front Y_{known}

$$ONVG = |Y_{known}|$$

where $||$ denotes cardinality, i.e., the number of elements or objects in a set.

2. Overall true Nondominated Vector Generation (*OTNVG*): counts the number of solutions in the Pareto front Y_{known} that are also in the true Pareto-optimal front Y_{true} .

$$OTNVG = | \{ y \mid y \in Y_{known} \wedge y \in Y_{true} \} |$$

Table 5.15 presents the performance comparison summary of both algorithms. As can be seen, from the 20 runs carried out, 389 nondominated solutions were obtained, which is denoted by the sum in *ONVG* obtained by NSGA-II and MOEA-DAP. As can be observed, only 96 nondominated solutions were obtained when using the NSGA-II algorithm; in contrast to the 293 solutions obtained by the new MOEA-DAP algorithm.

These 389 solutions are nondominated in their individual runs, thus to form the true Pareto-optimal front, Y_{true} , these solutions are joined together and the Pareto dominance criterion is checked to eliminate dominated solutions, as shown in Section 5.3.

Table 5.15 Performance comparison

Algorithm	Population Size (A)	Run	<i>ONVG</i>	<i>OTNVG</i>
NSGA-II	50	1	12	5
		2	13	4
		3	11	0
		4	10	2
		5	4	0
		6	13	2
		7	6	0
		8	12	2
		9	3	0
		10	12	5
			$\Sigma = 96$	$\Sigma = 20$
Our algorithm MOEA-RAP	50	1	27	15
		2	28	13
		3	36	17
		4	36	11
		5	28	20
		6	29	13
		7	25	11
		8	37	20
		9	26	12
		10	21	13
			$\Sigma = 293$	$\Sigma = 145$

After checking the Pareto dominance criterion, the true Pareto front is formed by 139 solutions, that is $Y_{true} = |139|$. To obtain the values in the fifth column of Table 5.15, we count the number of nondominated solutions in each individual run, Y_{known} , and check how many of these are also in the true Pareto-optimal front, Y_{true} (see Table B.21 in Appendix B).

The true Pareto-optimal front is formed by 139 nondominated solutions. From these, 124 solutions were obtained with our algorithm (MOEA-DAP), and only 15 solutions

were obtained by using the NSGA-II algorithm. The maximum and minimum values found in Y_{true} are presented in Table 5.16.

Table 5.16. Maximum and minimum values found in Y_{true}

	Reliability	Cost	Weight
max	0.999999 ≈ 1	129	129
min	0.33768	6	11

To make a visual comparison of the solutions in Y_{true} obtained by both algorithms, Figure 5.8 shows the 15 solutions obtained by the NSGA-II that contributed to the true Pareto-optimal front and to better visualize the solutions Figures 5.9, 5.10 and 5.11 plot reliability vs. cost, reliability vs. weight and cost vs. weight, respectively. In the same way, Figure 5.12 shows the 124 solutions found in Y_{true} and obtained by the MOEA-DAP algorithm. Figures 5.13, 5.14 and 5.15 show also the two dimensional representation of the solutions.

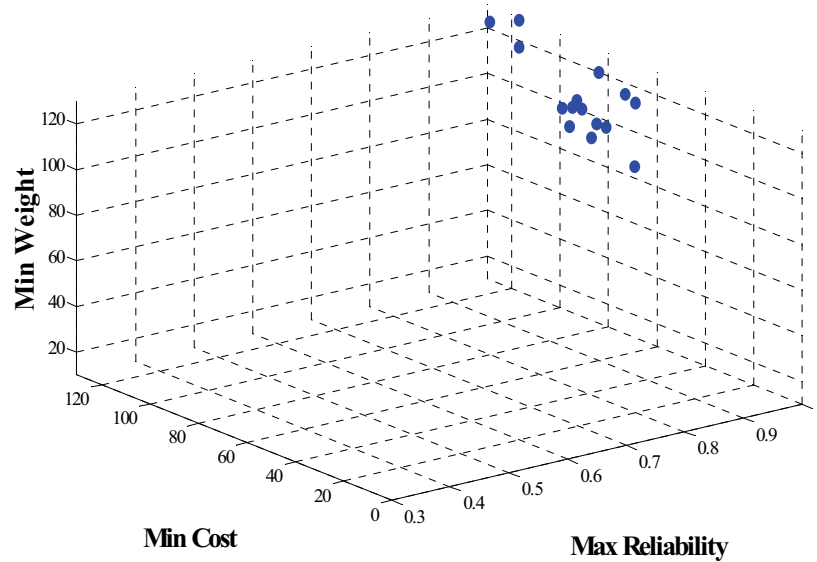


Figure 5.8 Nondominated solutions in Y_{true} obtained from NSGA-II algorithm

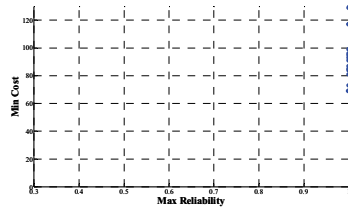


Figure 5.9 Nondominated solutions in Y_{true} obtained from NSGA-II algorithm. Rel vs Cost

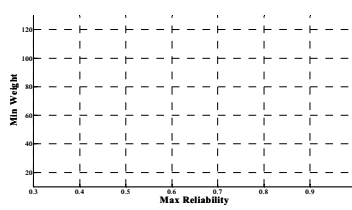


Figure 5.10 Nondominated solutions in Y_{true} obtained from NSGA-II algorithm. Rel vs Weight

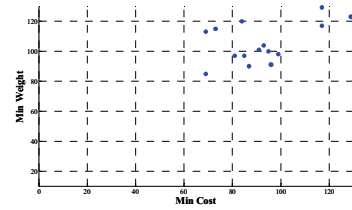


Figure 5.11 Nondominated solutions in Y_{true} obtained from NSGA-II algorithm. Cost vs Weight

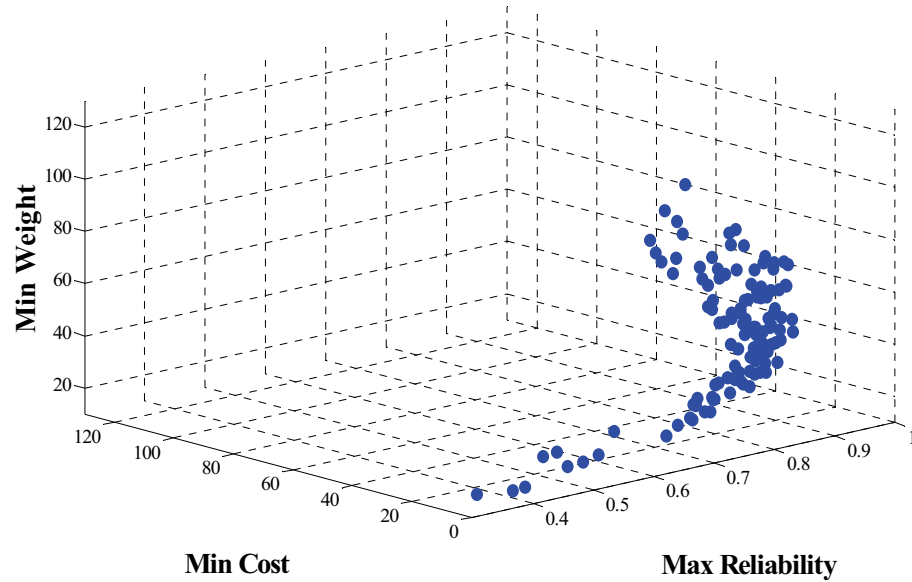


Figure 5.12 Nondominated solutions in Y_{true} obtained from MOEA-DAP algorithm

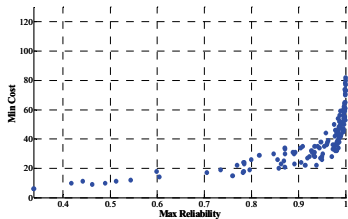


Figure 5.13 Nondominated solutions in Y_{true} obtained from MOEA-DAP. Rel vs Cost

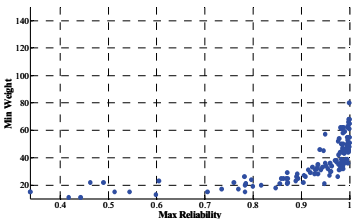


Figure 5.14 Nondominated solutions in Y_{true} obtained from MOEA-DAP. Rel vs Weight

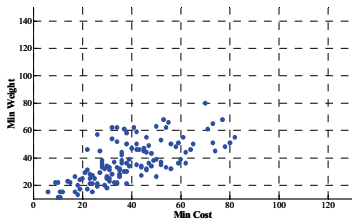


Figure 5.15 Nondominated solutions in Y_{true} obtained from MOEA-DAP. Rel vs Weight

As can be seen in Figures 5.8, through 5.11 the nondominated solutions obtained by the NSGA-II algorithm are not uniformly distributed along the Pareto front and mostly are extreme solutions, with very high reliability but also with large cost and weight. In

contrast, we can observe that our new MOEA-DAP algorithm is able to obtain a set of non-dominated solutions fairly uniformly spreading along the Pareto frontier (Y_{true}).

Thus, as stated in Coello Coello & Toscano Pulido (2001), although the NSGA-II algorithm tends to spread quickly and appropriately when a certain nondominated region is found, it seems to have difficulties to generate nondominated solution vectors that lie in certain regions of the search space. For this particular problem, the MOEA-DAP offers distinct advantages and in the example, the performance was better. However, it can not be concluded that the performance will be better for all problems.

The results of the performance comparison were based on two different performance metrics, *ONVG* and *OTNVG*. The results indicate that MOEA-DAP is more effective for solving this type of problem. Since our algorithm obtains more solutions that contribute to the true Pareto-optimal front, the solutions are also more uniformly distributed along the Pareto frontier than those solutions coming from the NSGA-II.

The performance comparison show that MOEA-DAP, which is a problem specific multi-objective algorithm, is superior at obtaining diverse solutions and a uniform spread along the Pareto front. In contrast, the solutions obtained by the NSGA-II algorithm are not uniformly distributed along the Pareto front and are mainly extreme solutions. This, however, is not surprising, since NSGA-II is a general multi-objective algorithm it can hardly compete with a method specifically designed to solve this particular MOCO reliability-design problem. As mentioned before, MOEA-DAP has the strength of a problem-oriented technique.

5.5 Summary

In this section, **the methodology** involving the sequential operations of a **newly developed multi-objective evolutionary algorithm for solving system design allocation problems is presented**. Our algorithm (MOEA-DAP) uses a GA based on rank selection and elitist reinsertion, and a modifying genetic operator constraint handling method.

MOEA-DAP, mainly differs from other MOEAs in the type of crossover operation performed. In this step, several offspring are created through **multi-parent recombination**. As a result, from n parents, denoted as $Npar$, considered for mating, in our algorithm there will be $s [Npar (Npar-1)]$ number of children produced, where s is the number of subsystems considered. Thus, the mating pool contains a great amount of diversity of solutions. This disruptive nature of our proposed type of crossover, subsystem rotation crossover (SURC), appears to encourage the exploration of the search space.

A **performance comparison** between one of the most successful evolutionary algorithms that currently exists: NSGA-II and our algorithm, shows that our algorithm is more powerful to solve multi-objective redundant design allocation problems. This shows that one challenge still remains for current multi-purpose MOEAs, which is the scalability problem, that is, the efficient solution of large scale problem instances.

6. MOMS-GA: an extension to MOEA-DAP to consider multi-state system performance

In this chapter, a custom genetic algorithm was developed and implemented to solve multiple objective multi-state reliability optimization design problems. Many real-world engineering design problems are multi-objective in nature, and among those, several of them have various levels of system performance ranging from perfectly functioning to completely failed. This multi-objective genetic algorithm uses the universal moment generating function approach to evaluate the different reliability or availability indices of the system. The components are characterized by having different performance levels, cost, weight and reliability. The solution to the multi-objective multi-state problem is a set of solutions, known as the Pareto-front, from which the analyst may choose one solution for system implementation. Two illustrative examples are presented to show the performance of the algorithm, and the multi-objective formulation considered for both of them, is the maximization of system availability and the minimization of both system cost and weight.

6.1. Introduction

Most realistic optimization problems, particularly those in system design, require the simultaneous optimization of more than one objective function. In this chapter, I present a multi-objective multi-state genetic algorithm (MOMS-GA) to solve multiple objective multi-state reliability and availability optimization design problems (Taboada *et al.*,

2006). The objectives considered are the maximization of the system availability, and the minimization of system cost and weight. The components and the system considered have a range of different states and the universal moment generating function (UMGF) approach is used to obtain the system availability.

Reliability is defined as the probability that a device or system is able to perform its intended functions satisfactorily under specified conditions for a specified period of time. However, traditional reliability assumes that a system and its components can be in either a completely working or a completely failed state only (Birnbaum *et al.*, 1961), i.e., no intermediate states allowed. This condition has facilitated the development of a robust and extensive theory to analyze system performance. However, in some cases, traditional reliability theory fails to represent the true behavior of the system. Failure to acknowledge this situation can represent a major deficiency when systems have a range of intermediate states that are not accounted for by traditional reliability estimation.

To describe the satisfactory performance of a device or system, we may need to use more than two levels of satisfaction, for example, excellent, average, and poor. Multi-state reliability (El-Newehi *et al.*, 1978; Barlow & Wu, 1978; Lisniaski & Levitin, 2003) has been proposed as a complementary theory to cope with the problem of analyzing systems where traditional reliability theory and models become insufficient. Then, in a multi-state system, both the system and its components are allowed to experience more than two possible states, e.g., completely working, partially working or partially failed, and completely failed.

6.2 Previous research on multi-state systems (MSS)

When considering multi-state systems (MSS), there are generally four methods for

MSS reliability assessment, which are, (1) the structure function approach (Brunelle & Kapur, 1998; Pourret *et al.*, 1999), (2) the stochastic processes “Markov” approach (Xue & Yang, 1995), (3) The Monte Carlo simulation technique (Ramirez-Marquez & Coit, 2005) and (4) the universal moment generating function approach (Levitin & Lisnianski, 2001; Ushakov 1986, 1988).

Research that considers the RAP for MSS considering one objective and several constraints have been presented recently. Ramirez-Marquez & Coit (2004), proposed a heuristic to solve a multi-state series-parallel system with binary capacitated components. In their study, the RAP is formulated with the objective of minimizing the total cost associated with a system design constrained by a reliability performance index. In their heuristic, once a component selection is made, only the same component type can be used to provide redundancy. Levitin *et al.* (1998) used a GA for solving the multi-state RAP, where the system and its components have a range of performance levels. Based on the UMGF, they determined the system availability. Levitin (2000) addressed the multi-stage expansion problem for multi-state series-parallel systems. In this problem, the system-study period is divided into several stages. Later, Levitin (2001) solved a redundancy optimization problem for multi-state systems with fixed resource-requirements and unreliable sources, subject to availability constraints. Later, Tian & Zuo (2006) applied GA together with physical programming to solve the RAP.

6.3 Evolutionary approaches in multi-objective optimization

Evolutionary algorithms, as shown in Chapter 3, have been recognized to be well-suited to solve multi-objective optimization problems. Their ability to accommodate complex problems, involving features such as discontinuities, multimodality, disjoint

feasible spaces, etc., reinforces the potential effectiveness of EAs in multi-objective search and optimization.

These universal methods, although capable of solving many multi-objective problems, are not specifically designed to be efficient in the solution of large-scale multi-objective system design combinatorial problems. Therefore, in this chapter, a specific MOEA, called MOMS-GA, is presented as a method exclusively designed to solve multiple objective multi-state reliability-design optimization problems. Thus, MOMS-GA has the strength of a problem-oriented technique, in which the selection of components is advantageously combined to create a MOEA which can undertake the problem in the most efficient way. This is the first reported multi-objective evolutionary framework for solving multiple objective multi-state reliability-design optimization problems. The fundamental operations of MOMS-GA are presented in Section 6.5.

6.4 Multi-state system availability estimation method

The procedure used in this chapter for system-availability evaluation is based on the universal z -transform, originally introduced by Ushakov (1986). In the literature, the universal z -transform is also called universal moment generating function (UMGF) or simply u -transform, which has proven to be very effective for high dimension combinatorial problems. The UMGF represents an extension of the widely known moment generating function (Ross, 1993). The UMGF of a discrete random variable G is defined as a polynomial

$$u(z) = \sum_{j=1}^J p_j z^{g_j} \quad (1)$$

Where the discrete random variable G has J possible values and p_j is the probability that G is equal to g_j .

The probabilistic characteristics of the random variable G can be found using the function $u(z)$. In particular, if the discrete random variable G is the MSS stationary output performance, then availability A is given by the probability $P(G \geq D)$, which can be defined as:

$$P(G \geq D) = \delta(u(z)z^{-D}) \quad (2)$$

Where δ is the disruptive operator defined by the following expressions:

$$\delta(p_j z^{g_j - D}) = \begin{cases} p_j, & \text{if } g_j \geq D \\ 0, & \text{if } g_j < D \end{cases} \quad (3)$$

$$\delta\left(\sum_{j=1}^J p_j z^{g_j - D}\right) = \sum_{j=1}^J \delta(p_j z^{g_j - D}) \quad (4)$$

It can be easily shown that equations (1)-(4) meet condition $P(G \geq D) = \sum_{g_j \geq D} p_j$. By using the operator δ , the coefficients of polynomial $u(z)$ are summed for every term with $g_j \geq D$, and the probability that G is not less than some specified value D is systematically obtained.

Consider single components with total failures and each component i has nominal performance G_i and availability A_i . The UMGF of such a component has only two terms and can be defined as:

$$u_i(z) = (1 - A_i)z^0 + A_i z^{G_i} = (1 - A_i) + A_i z^{G_i} \quad (5)$$

To evaluate the MSS availability of a series-parallel system, two basic composition operators are introduced. These operators determine the polynomial $u(z)$ for a group of components.

6.4.1 Parallel components

The systems considered in this section pertain to flow transmission multi-state

systems, in which the flow can be dispersed and transferred by parallel components simultaneously. Therefore, for a system containing n elements connected in parallel, the total capacity is equal to the sum of capacities of all its elements. Therefore, its u -function can be calculated using the π operator:

$$u_p(z) = \pi(u_1(z), u_2(z), \dots, u_n(z))$$

where

$$G = \sum_{i=1}^n g_i$$

Therefore for a pair of components connected in parallel we have:

$$\pi(u_1(z), u_2(z)) = \pi\left(\sum_{i=1}^{k_1} p_i z^{a_i}, \sum_{j=1}^{k_2} q_j z^{b_j}\right) = \sum_{i=1}^{k_1} \sum_{j=1}^{k_2} p_i q_j z^{a_i + b_j} \quad (6)$$

The parameters a_i and b_j are physically interpreted as the performances of the two components, k_1 and k_2 are numbers of possible performance levels for these components, while p_i and q_j are steady-state probabilities of possible performance levels for the components. One can see that the π operator is simply a product of the individual u -functions. For a system with multiple components, the operator can for two components can be iteratively applied to accommodate any number of components.

6.4.2 Series components

When the components are connected in series in flow transmission multi-state systems, the component with the least performance becomes the bottleneck of the system. This component, therefore, defines the total system productivity. To calculate the u -function for a system with m elements connected in series, the σ operator should be used:

$$u_s(z) = \sigma(u_1(z), u_2(z), \dots, u_m(z))$$

For which

$$G = \min\{g_1, g_2, \dots, g_m\}$$

So that,

$$\sigma(u_1(z), u_2(z)) = \sigma\left(\sum_{i=1}^n p_i z^{a_i}, \sum_{j=1}^m q_j z^{b_j}\right) = \sum_{i=1}^n \sum_{j=1}^m p_i q_j z^{\min\{a_i, b_j\}} \quad (7)$$

Using π and σ operators, the u -function of the entire system can be defined. To do this, we must first determine the individual u -functions of each element.

6.4.3 Total system reliability evaluation

Let us consider the general case where failures may either cause total failure or reduction of the component capacities, and therefore, different capacity degradation levels must be considered. In this case, the u -function of such a component is:

$$u_i^l(z) = \sum_{j=1}^J p_{ij}^l z^{g_{ij}}$$

Where the index l represents the subsystem, i denotes the component (within subsystem l) and j the component state. g_{ij} is the capacity of the element in state j , and p_{ij}^l is the probability of this state.

We obtain the UMGF of the l^{th} subsystem containing H_i parallel components of different versions by,

$$u_i(z) = \prod_{i=1}^{H_i} u_i^l(z) = \prod_{i=1}^{H_i} \sum_{j=1}^{J_i} p_{ij}^l z^{g_{ij}} \quad (8)$$

Where the i^{th} component in subsystem l has J_i different states, each state has a probability p_{ij}^l .

Thus, the UMGF of the entire system containing m subsystems connected in series is:

$$u_s(z) = \sigma\left(\prod_{i=1}^{H_1} u_i^1(z), \dots, \prod_{i=1}^{H_l} u_i^l(z), \dots, \prod_{i=1}^{H_m} u_i^m(z)\right) = \sum_{i=1}^N p_i z^{a_i} \quad (9)$$

Once all terms are considered and terms with the same exponents are grouped together, N represents the total number of possible system states, a_i represents the

different possible performance levels with probability p_i .

To evaluate the availability A of the entire system, $P(G \geq D)$ considering the cumulative demand curve is given by Equation (9). The corresponding UMGF, $u_d(z)$, for the random demand load is defined as:

$$u_d(z) = \sum_{s=1}^S q_s z^{-D_s}$$

q_s is the vector of the steady-state probabilities of the corresponding load demand level D_s and S is the maximum number of different intervals from the cumulative demand curve.

$$A = P(G_n \geq D_s) = \delta(u_s(z)u_d(z)) = \delta\left(\sum_{i=1}^n p_i z^{a_i} \sum_{s=1}^S q_s z^{-D_s}\right) = \delta\left(\sum_{i=1}^n \sum_{s=1}^S p_i q_s z^{a_i - D_s}\right) \quad (10)$$

6.5 Multi-objective multi-state genetic algorithm (MOMS-GA)

MOMS-GA was developed as an extension of MOEA-DAP (Taboada & Coit, 2006b), a multi-objective evolutionary algorithm for design allocation problems, introduced in last chapter. In MOEA-DAP, the multi-objective formulation was to maximize system reliability, minimize the total cost, and minimize the system weight, for a series-parallel system. However, MOEA-DAP was developed to consider binary-state reliability. That is, the evolutionary algorithm assumed that the system and its components could be in either a working or a failed state only. Thus, MOMS-GA, is a natural extension of MOEA-DAP. The developed MOMS-GA works under the assumption that both the system and its components experience more than two possible states of performance. Thus, in general, MOMS-GA differs from MOEA-DAP in the evaluation of the first objective function. MOEA-DAP evaluated system reliability (binary-state), while in MOMS-GA, the evaluation of the first objective function is

system availability (multi-state). The UMGF approach was implemented in the algorithm code to obtain the system availability.

A detailed explanation of the characteristics of the solution encoding, evolution parameters and genetic operators are as described in Chapter 5 (Taboada & Coit, 2006b).

However, the fundamental operations of MOMS-GA are summarized next.

1. [**Start**] Generate random population of n chromosomes. MOMS-GA uses an integer chromosomal representation.
2. [**Objective function values evaluation**] Evaluate system availability using the UMGF. Evaluate system cost and system weight.
3. [**Pareto dominance evaluation**] Pareto dominance criterion is checked in the initially created solutions. Those solutions that are dominated by other solutions are eliminated. Thus, in this way, MOMS-GA ensures that the resulting population only contains Pareto-optimal solutions.
4. [**Fitness evaluation**] Evaluate the following fitness functions of each chromosome \mathbf{x} in the population.
 - 4.1 Fitness Metric 1: Distance-based, $f_1(i)$. It gives highest fitness to those solutions that are farther away from other solutions in the Pareto front. It is intended for maintaining population diversity.
 - 4.2 Fitness Metric 2: Dominance count-based, $f_2(i)$. It aims to select those individuals which are more dominating (intended to achieve proximity).
 - 4.3 Aggregated Fitness Metric, $f_a(i)$: Fitness Metric 1 + Fitness Metric 2, $f_a(i) = f_1(i) + f_2(i)$. It aims to weight both metrics equally.
5. [**Selection**] Rank selection is used. With a given crossover probability, select individuals with the highest aggregated fitness to perform recombination.
6. [**Crossover**] With a pre-defined crossover probability, crossover the parents to form new offspring (children). For the exploitation of the combinatorial structure within the search algorithm, a problem-dependent component is developed in MOMS-GA: a specific crossover operator called subsystem rotation crossover (SURC). In this step, multi-parent recombination is allowed. This action, and the way that SURC works, produces a large number of children in the mating pool,

creating a large number of diverse solutions to choose from. Diversity is considered favorable, as the greater the variety of genes available to the genetic algorithm, the greater the likelihood of the system identifying good alternate solutions.

7. [**Mutation**] Single-point mutation is used. With a pre-defined mutation probability, mutate new offspring at a random position in the chromosome.
8. [**Reinsertion**] MOMS-GA uses elitist reinsertion in the aim of preventing the loss of the best-found solutions. New offspring plus a specified percentage of the most elite individuals from the previous population are chosen to form the new population.
9. [**Replace**] Use new generated population for a further run (generation) of the algorithm
10. [**Test**] If the Generation i = Generation 'max', **stop**, and return the best solutions in current population, otherwise return to step 2.

6.6 Numerical examples

Two examples are considered. They pertain to the type of flow transmission multi-state systems with flow dispersion. The main characteristic in these systems is that the parallel elements in each subsystem can transmit the flow simultaneously. The first example considers binary capacitated components and multi-state system performance, while the second example considers multi-state components and multi-state system performance.

The first example consists of five main units connected in series. For each unit, there are several components available to choose from to provide redundancy. Each component of the system is binary capacitated. This problem has been previously solved as a single objective problem considering the minimization of total system cost, subject to a desired level of reliability by using a GA in Levitin & Lisnianski (2001), and later by Ramirez-Marquez & Coit (2004) using a heuristic. Recently, Gupta & Agarwal (2006) considered the same example using a GA which incorporates a dynamic adaptive penalty function.

The second example presented consists of three main units connected in series. For each unit, there are several components available that can be chosen to provide redundancy. Each component of the system can have different levels of performance, which range from maximum capacity to total failure.

6.6.1 Example 1

Table 6.1 shows the example considered, consisting of five main units connected in series. For each unit, there are several components available in the market that can be chosen to provide redundancy. Each component of the system is considered to be binary capacitated, meaning that it can have only two states, functioning with the nominal capacity or total failure, corresponding to capacity 0. The collective performance of these binary components leads to multi-state system behavior. Each component is characterized by its availability, nominal capacity, cost and weight. Without loss of generality, component capacities can be measured as a percentage of the maximum demand. Table 6.2 presents different demand levels for a given period, known as the cumulative demand curve.

Table 6.1. Characteristics of the system elements available

Subsystem	Component Type	Availability	Feeding Capacity (%)	Cost	Weight
1	1	0.980	120	0.590	35.4
	2	0.977	100	0.535	34.9
	3	0.982	85	0.470	34.1
	4	0.978	85	0.420	33.9
	5	0.983	48	0.400	34.2
	6	0.92	31	0.180	34.3
	7	0.984	26	0.220	32.6
2	1	0.995	100	0.205	26.5
	2	0.996	92	0.189	22.4
	3	0.997	53	0.091	20.3
	4	0.997	28	0.056	21.7
	5	0.998	21	0.042	25.2
3	1	0.971	100	7.525	42.1
	2	0.973	60	4.720	41.7
	3	0.971	40	3.590	40.8
	4	0.976	20	2.420	39.6
4	1	0.977	115	0.180	25.4
	2	0.978	100	0.160	23.9
	3	0.978	91	0.150	24.7
	4	0.983	72	0.121	24.6
	5	0.981	72	0.102	23.6
	6	0.971	72	0.096	26.2
	7	0.983	55	0.071	25.5
	8	0.982	25	0.049	22.6
	9	0.977	25	0.044	24.8
5	1	0.984	128	0.986	15.4
	2	0.983	100	0.825	15.3
	3	0.987	60	0.490	14.9
	4	0.981	51	0.475	15.0

Table 6.2. Parameters of the cumulative demand curve

Demand (%)	100	80	50	20
Duration (h)	4203	788	1228	2536
Duration (%)	0.48	0.09	0.14	0.29

The problem was solved using the developed algorithm, MOMS-GA, with a population size of 200 and 50 generations. MOMS-GA, fully coded in MATLAB[®] 7.0, was run on a Sony VAIO computer, with an Intel Pentium processor operating at 1.86 GHz and 1 GB of RAM. The computation time was 595.25 seconds. The problem

considered was a multi-objective problem with system availability to be maximized and, cost and weight of the system to be minimized.

Figure 6.1 shows the 118 solutions found in the Pareto-front. To better visualize the solutions obtained, Figure 6.2 show the two dimensional representation of the same solutions.

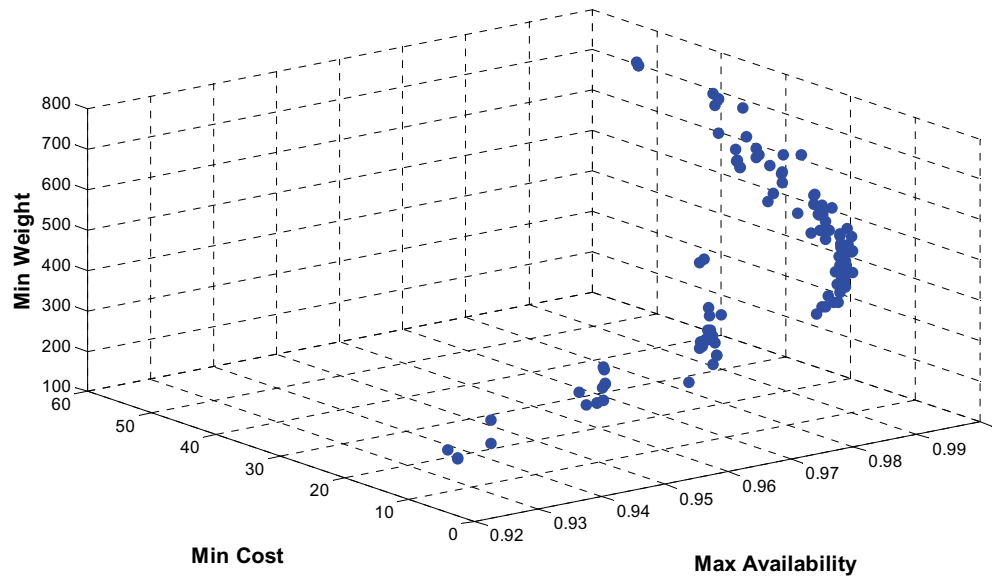


Figure 6.1 Pareto front of example 1

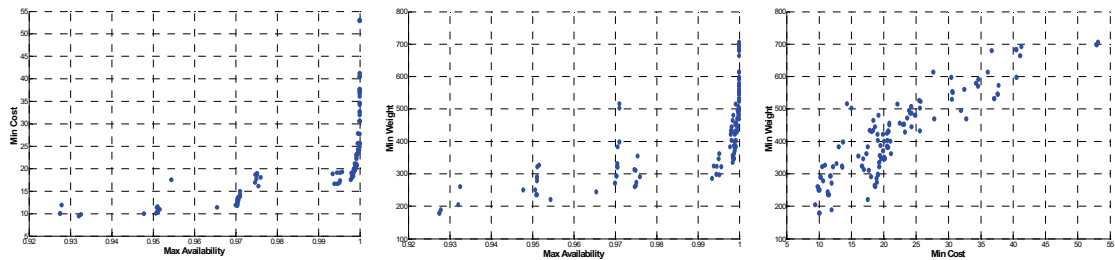
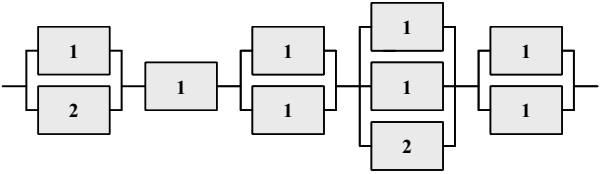
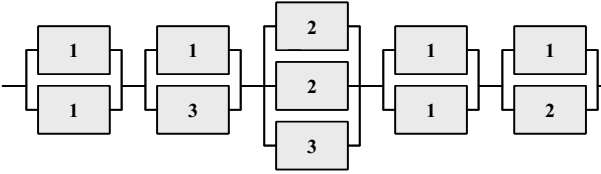
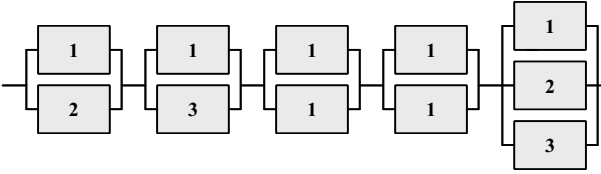


Figure 6.2 Pareto front of example 1 in a two dimensional space

Once the Pareto-optimal set is obtained, the decision-maker has to decide which of the non-dominated points to choose as the solution to the problem. For instance, the regions of the Pareto set which express good compromises according to problem-specific knowledge can be identified. More detail on methods to be applied in the decision-

making stage to reduce the size of the Pareto-optimal set, and obtain a smaller representation of the multi-objective design space can be found in Taboada & Coit (2007) and Taboada *et al.* (2007a). In this case, example solutions from the “*knee*” region (Das, 1999; Branke *et al.*, 2004) are presented as good compromises. The “*knee*” is formed by those solutions of the Pareto-optimal front where a small improvement in one objective would lead to a large deterioration in at least one other objective. Table 6.3 shows three example design configurations from this region with their respective system availability, cost and weight.

Table 6.3. Example design configurations of Example 1

Sol. No.	System Design Configuration Diagram	Availability	Cost	Weight
31		0.993439	18.872	286.5
34		0.994562	16.677	323.7
56		0.995164	19.132	297.7

6.6.2 Example 2

Table 6.4 shows the second example considered, which consists of three main units connected in series. For each unit, there are several components available in the market that can be chosen to provide redundancy. Each component of the system can have different levels of performance, which range from maximum capacity to total failure.

Each component is characterized by its availability (p_{ij}), nominal capacity, cost and weight. Table 6.5 presents the system cumulative demand curve.

Table 6.4. Characteristics of the system elements available

Subsystem	Component Type	Availability (p_{ij})	Feeding Capacity (%)	Cost	Weight
1	1	0.70	130	65	80
		0.20	100		
		0.10	0		
	2	0.65	100	60	70
		0.25	80		
		0.10	0		
	3	0.60	95	50	75
		0.30	90		
		0.10	0		
	4	0.90	135	80	100
		0.05	80		
		0.05	0		
2	1	0.50	200	120	70
		0.25	140		
		0.20	100		
		0.05	0		
	2	0.60	220	130	100
		0.30	140		
		0.10	0		
	3	0.90	300	200	100
		0.10	0		
		0.10	0		
3	1	0.80	160	200	60
		0.15	90		
		0.05	0		
	2	0.85	140	160	100
		0.15	0		
	3	0.90	200	250	90
		0.10	0		
	4	0.65	100	100	70
		0.30	80		
		0.05	0		
	5	0.50	130	60	50
		0.30	100		
		0.15	50		
		0.05	0		

Table 6.5. Parameters of the cumulative demand curve

Demand (%)	100	80	60	20
Duration (h)	4380	2628	876	876
Duration (%)	0.5	0.3	0.1	0.1

MOMS-GA was run considering a population size of 100 and 50 generations. The computation time was 606.20 seconds. The multi-objective formulation seeks to maximize system availability, while minimizing system cost and weight. Figure 6.4 shows the 57 solutions found in the Pareto front. To better visualize the solutions obtained, Figure 6.4 show the two dimensional representation of the same solutions. Table 6.6 shows three example design configurations with its respective system availability, cost and weight. These three solutions were selected as good compromise solutions by considering the “*knee*” of the Pareto-front.

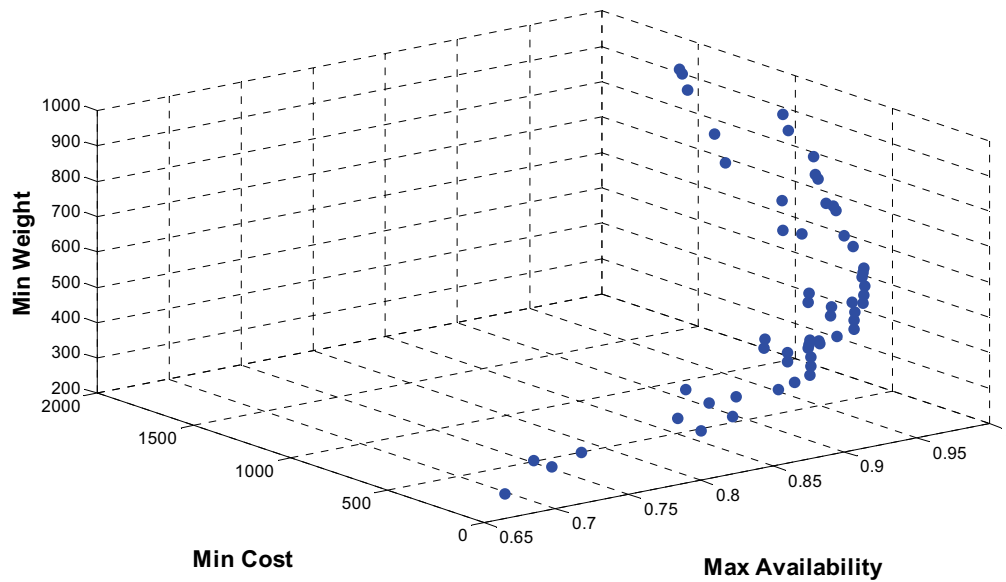


Figure 6.3 Pareto front of example 2

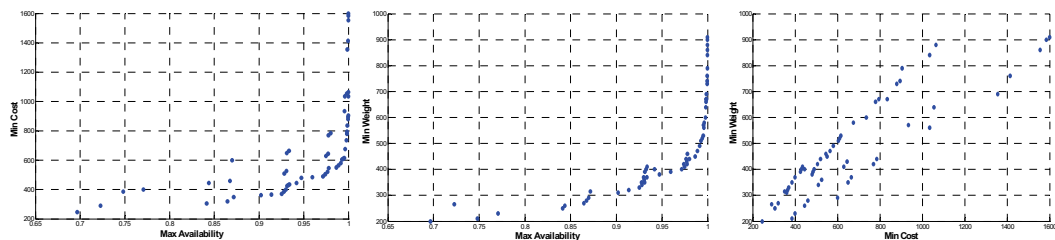
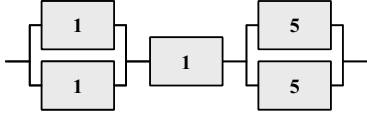
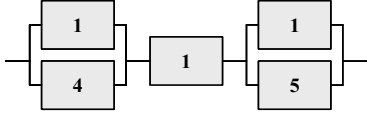
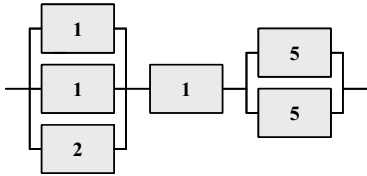


Figure 6.4 Pareto front of example 2 in a two dimensional space

Table 6.6 Example design configurations of example 2

Sol. No.	System Design Configuration Diagram	Availability	Cost	Weight
24		0.925452	370	330
8		0.930628	525	360
12		0.932698	430	400

6.7 Summary

MOMS-GA was developed to solve multiple objective multi-state reliability optimization design problems. Many real-world engineering design problems are multi-objective in nature, and among those, several of them have various levels of system performance. The multi-objective GA developed uses the UMGF to evaluate the different reliability indices of the system. The use of a fast UMGF-based procedure for system availability evaluation within a multi-objective evolutionary algorithm allows identification of the entire multi-state performance distribution based on the performance of its components. The components are characterized for having different performance levels, cost, weight, and availability. The solution to the MOMS problem is a set of solutions, known as the Pareto-front, from which the analyst may choose one solution for system implementation.

7. A multi-objective evolutionary algorithm for determining optimal configurations of multi-task production systems

This chapter presents a new multiple objective evolutionary algorithm to determine optimal configurations of multi-state, multi-task production systems based on availability analysis. A multi-task production system is one in which different subsets of machines can be used to perform distinct functions or tasks. The performance of a manufacturing system is greatly influenced by its configuration. Availability can be used in the context of multi-task production systems to select a particular configuration that maximizes the probability of meeting a required demand for each specific task, or the expected productivity for each task. A particular configuration may not simultaneously maximize the probability of meeting demand for each of the individual tasks, and thus, the problem is treated as a multi-objective optimization problem. The solution to this problem is a set of promising solutions that provides a trade-off among the different objective functions considered.

7.1 Introduction

Many modern systems operate in a multi-task mode. A multi-task production system is one in which different subsets of machines can be used to perform distinct functions or tasks. Multi-task machining has been adopted in an increasing number of manufacturing job shops, especially by companies facing competition from lower cost markets. Some of the most interesting and important examples of multi-task systems occur in flexible

manufacturing systems (FMSs). Flexibility is a major consideration in the design of manufacturing systems, and FMSs have been developed over the last two decades to help manufacturing industries move towards the goal of flexibility. Many examples can be found in flexible production facilities and flexible assembly systems. For instance, in a flexible assembly system, there are typically a limited number of different product types, and the system has to produce a given quantity of each product type (Pinedo & Chao, 1999).

The development of effective and efficient FMS scheduling strategies remains an important and active research area. However, the selection of a system configuration is a frequent difficulty which arises during the early stages of the manufacturing system development (during the machine allocation phase). At this stage, manufacturers must choose, not only machine specifications and vendors, but also the configuration of the system. However, because of their nature, multi-task manufacturing systems can be designed in many different ways. The chosen configuration has a profound impact on the overall performance of the system in terms of reliability, productivity, cost, etc.

Significative research has been done in the area of configuration selection for manufacturing systems. For instance, Koren *et al.* (1998) analyzed how reliability, productivity, and quality were affected by different system configurations assuming known machine level reliability and process capability. Later, Altumi *et al.* (2001) presented a model to determine the spare tooling allocation requirement for the tooling system in a FMS, so that the desired system reliability is achieved and the cost is minimized. Cochran *et al.* (2001) analyzed how the selection of a manufacturing system configuration can impact the ability to meet different types of objectives such as cost,

performance, and quality. By considering several design configurations, Freiheit *et al.* (2004) examined the importance of design configuration on system productivity. They showed how improvements can be obtained by using bufferless series-parallel configuration arrangements. Later, Youssef *et al.* (2006) used the universal generating function for evaluating the availability of multi-state manufacturing systems capable of producing more than one part type. Seward & Nachlas (2004) considered availability in the analysis of manufacturing systems and developed models for the operational reliability and availability of multi-task systems.

Analogous to the machine allocation phase in production systems, there are many other engineering design and development projects that require the allocation of redundant components to meet high reliability specifications. Perhaps, the most representative problem in reliability design is the well-known redundancy allocation problem (RAP). In the RAP, as shown in previous chapters, one considers a system with a total of m subsystems arranged in series. For each subsystem, there are n functionally equivalent components arranged in parallel, with potentially different levels of cost, weight, reliability and other characteristics. The n components are to be selected from several available component types, where multiple copies of each type can be selected. While there are many forms of the RAP, it generally involves the selection of components and redundancy levels with the objective to maximize the overall system reliability or availability while satisfying a constraint for some other system characteristics such as system cost and system weight. The RAP has been solved using dynamic programming (Fyffe *et al.*, 1968; Nakagawa & Miyazaki, 1981), integer programming (Bulfin & Liu, 1985; Gen *et al.*, 1990), genetic algorithms (Ida *et al.*, 1994;

Painton & Campbell, 1995; Coit & Smith, 1996a; Tian & Zuo, 2006; Taboada & Coit, 2006b), among others.

Despite the clear relationship between the two types of allocation problems, production scheduling and reliability optimization are typically treated independently in the research literature and in practice. This chapter shows how system availability can be used in the context of multi-task production systems to select a particular configuration that maximizes the probability of meeting a given demand for each of the individual tasks. The problem is treated as a multi-objective problem, and it is solved using a multi-objective evolutionary algorithm.

7.2. Problem description

The problem addressed in this chapter is one that pertains to a flexible flow shop environment with L stages arranged in series. At each stage l ($l=1, \dots, L$) several machines work in parallel, such that the total performance of the stage is equal to the sum of performances of the available machines. The system is aimed at performing K different tasks. Any task k ($k=1, \dots, K$) can be processed at each stage on any machine. For each stage l , there are I_l types of machines available in the market. Each type i ($i=1, \dots, I_l$) is characterized by its cost c_i^l , nominal performance g_{ik}^l and availability p_{ik}^l when performing task k . For example, a particular manufacturing stage may involve processing on a lathe, and the nominal performance is the cutting speed and there may be several alternative lathes with different cutting speeds.

Any possible system structure can be represented by a matrix of integer numbers $\mathbf{h}=\{h_{il}, i=1, \dots, I_l, l=1, \dots, L\}$, where h_{il} is the number of machines of type i chosen for the stage l . The problem is to build a series-parallel system (by choosing the machines of

available types for each stage) that maximizes the probability of meeting a required system performance level (demand) D_k for each task k and minimizes the total system cost:

$$\text{maximize } A_k(\mathbf{h}, D_k) \quad \text{for } k=1, \dots, K,$$

$$\text{minimize } C(\mathbf{h}) = \sum_{l=1}^L \sum_{i=1}^{I_l} h_{il} c_i^l$$

The system is assumed to be a bufferless manufacturing system. The presence of multiple parallel machines per stage reduces the effect of breakdown of any of the machines, and thus, the use of buffers is not always necessary. Since a particular configuration may not simultaneously maximize the probability of meeting demand for each of the individual tasks, the problem becomes a multi-objective problem with K objectives to be maximized, i.e., the performance level for each task. Another objective that was additionally considered in two examples presented in this paper is the minimization of the overall system configuration cost. This multi-objective problem has a set of Pareto-optimal solutions.

7.3 Multi-state system availability estimation method

As shown in chapter 6, many practical systems can perform their intended functions at more than two different levels of performance. These kinds of systems are known as multi-state systems. Within the context of multi-task production systems, system availability is used as criteria to select a particular configuration that maximizes the probability of meeting a given demand for each of the individual tasks.

The universal moment generating function (UMGF) is used again to evaluate the availability of the multi-task multi-state manufacturing system. Therefore, in this chapter,

the UMGF introduced in the previous chapter, is adapted to be used within the context of multi-task production systems.

The u -function representing the probability mass function (pmf) of a discrete random variable Y is defined as a polynomial

$$u(z) = \sum_{j=1}^J \alpha_j z^{y_j}, \quad (1)$$

where the variable Y has J possible values and, α_j is the probability that Y is equal to y_j .

To obtain the u -function representing the pmf of a function of two independent random variables, composition operators are introduced. Considering a function $\varphi(Y_m, Y_n)$, the composition operators determine the u -function for $\varphi(Y_m, Y_n)$ using simple algebraic operations on the individual u -functions of the variables. All of the composition operators take the following form,

$$U(z) = u_m(z) \otimes_{\varphi} u_n(z) = \sum_{j=1}^{J_m} \alpha_{mj} z^{y_{mj}} \otimes_{\varphi} \sum_{i=1}^{J_n} \alpha_{ni} z^{y_{ni}} = \sum_{j=1}^{J_m} \sum_{h=1}^{J_n} \alpha_{mj} \alpha_{nh} z^{\varphi(y_{mj}, y_{nh})} \quad (2)$$

The u -function, $U(z)$, represents all of the possible mutually exclusive combinations of realizations of the variables by relating the probabilities of each combination to the value of function $\varphi(Y_m, Y_n)$ for this combination. For example, for functions $Y_m + Y_n$ and $\min(Y_m, Y_n)$ operator (2) takes the form

$$u_m(z) \otimes_{+} u_n(z) = \sum_{j=1}^{J_m} \sum_{h=1}^{J_n} \alpha_{mj} \alpha_{nh} z^{y_{mj} + y_{nh}} = u_m(z) u_n(z) \quad (3)$$

and

$$u_m(z) \otimes_{\min} u_n(z) = \sum_{j=1}^{J_m} \sum_{h=1}^{J_n} \alpha_{mj} \alpha_{nh} z^{\min(y_{mj}, y_{nh})} \quad (4)$$

Note that in the case of summation (3), the composition operator constitutes simple

product of polynomials.

Consider u -function representing the pmf of random performance of a single machine used within a specific manufacturing stage. When the system performs task k , any machine of type i belonging to stage l can be in one of two states: normal functioning with nominal performance g_{ik}^l (probability of this state is equal to the machine availability, p_{ik}^l) and total failure with performance 0 (probability of this state is equal to $1 - p_{ik}^l$). The UMGF representing the performance distribution of this machine is

$$u_{ik}^l(z) = (1 - p_{ik}^l)z^0 + p_{ik}^l z^{g_{ik}^l}. \quad (5)$$

Having the u -function representing the pmf of any random variable Y in the form (1), the probability that Y is not less than any fixed value D can be easily determined by summing the coefficients of polynomial $u(z)$ for every term with $y_j \geq D$. This can be done by applying the following operator δ to $u(z)$.

$$P(Y \geq D) = \delta(u(z), D) = \sum_{j=1}^J \alpha_j 1(y_j \geq D). \quad (6)$$

Applying operator $\delta(U_k(z), D_k)$ to the u -function $U_k(z)$, representing the pmf of system performance for task k , availability can be determined, i.e., probability that the demand D_k is met.

The multi-task production system considered in this chapter pertains to flow transmission multi-state systems (Levitin, 2005), in which the product can be dispersed and processed by parallel machines simultaneously. Therefore, for a production system containing several elements connected in parallel, the total capacity is equal to the sum of the capacities of all its elements. Therefore, its u -function can be calculated using the \otimes_+

operator.

The u -function representing the pmf of the cumulative performance of h_{il} identical machines of type i (when performing task k) can be obtained by applying operator (3) over h_{il} identical u -functions $u_{ik}^l(z)$. The resulting u -function takes the form

$$U_{ik}^l(z) = (u_{1k}^l(z) \otimes_{+} u_{2k}^l(z) \otimes_{+} \dots \otimes_{+} u_{ik}^l(z) \otimes_{+}) = (u_{ik}^l(z))^{h_{il}}. \quad (7)$$

Having the number of machines of each type in the stage l , one can obtain the u -function representing the pmf of the cumulative performance of all of the machines in this stage (when performing task k) as

$$U_k^l(z) = \otimes_{+} (U_{1k}^l(z), \dots, U_{l_k}^l(z)) = \prod_{i=1}^{I_l} (u_{ik}^l(z))^{h_{il}}. \quad (8)$$

When the stages are connected in series in flow transmission multi-state systems, the stage with the least performance becomes the bottleneck of the system. This machine, therefore, dictates the total system productivity. To calculate the u -function representing the pmf of the performance of the entire system performing task k , the \otimes_{\min} operator (4) should be used.

$$U_k(z) = \otimes_{\min} (U_k^1(z), \dots, U_k^m(z)) = \otimes_{\min} \left(\prod_{i=1}^{I_1} (u_{ik}^1(z))^{h_{i1}}, \dots, \prod_{i=1}^{I_m} (u_{ik}^m(z))^{h_{im}} \right). \quad (9)$$

Having the u -function $U_k(z)$ that represents the pmf of the performance of the entire system performing task k , one can obtain the system availability for task k as,

$$A_k = \delta(U_k(z), D_k) \quad (10)$$

If the demand D_k is random, it can be represented by its pmf , $q_s = P(D_k = d_{ks})$, for $s=1, \dots, S$, and the availability A_k can be obtained as,

$$A_k = \sum_{s=1}^S q_s \delta(U_k(z), d_{ks}) \quad (11)$$

7.4 Description of the multi-objective evolutionary algorithm

The evolutionary algorithm developed uses an integer chromosomal representation. For instance, consider the following example to illustrate a particular chromosome generated by the algorithm. Each integer corresponds to the number of redundant machines of that type. For example, Figure 7.1 shows a chromosome (genotype) and the mapping to its corresponding system configuration (phenotype). In this chromosome, for subsystem 1, there are only two copies of the first machine type. For subsystem 2, 1 copy of the first machine type, two copies of the second machine type and one copy of the third machine type are used in parallel. Finally, for subsystem 3, one copy of the first machine type, one copy of the second machine type and one copy of the third machine type are used in parallel.

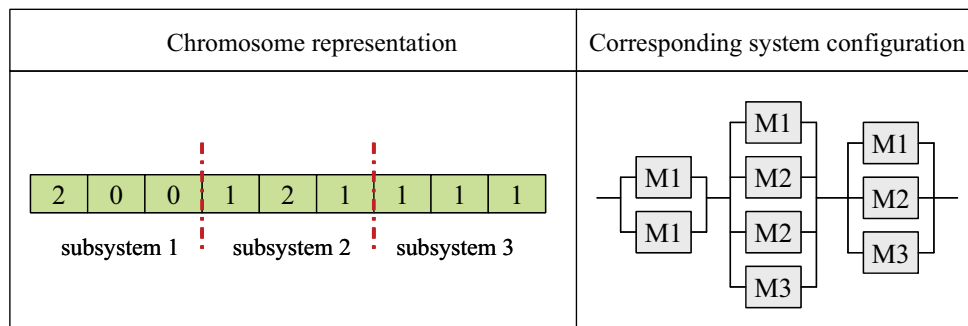


Figure 7.1 Representation of solutions

The general steps of the algorithm are briefly explained below.

Step 1. Randomly generate an initial population of solutions.

Step 2. Decode each solution and evaluate objective function values pertaining to the availability for each task. That is, for each of the solutions, the probability of meeting the required demand of each of the individual tasks is evaluated by using the UMGF; and,

finally, the overall system cost is computed.

Step 3. Check Pareto dominance criterion. Of the initial randomly generated solutions, eliminate those solutions that are dominated.

Step 4. Evaluate the following fitness functions of each chromosome \mathbf{x} in the population.

4.1. Fitness Metric 1: Distance-based, $f_1(i)$. It gives highest fitness to those solutions that are farther away from other solutions in the Pareto front. It is intended for maintaining population diversity.

4.2. Fitness Metric 2: Dominance count-based, $f_2(i)$. It aims to select those individuals which are more dominating (intended to achieve proximity to the true Pareto frontier).

4.3. Aggregated Fitness Metric, $f_a(i)$: Fitness Metric 1 + Fitness Metric 2, $f_a(i) = f_1(i) + f_2(i)$. It aims to weight both metrics equally.

Step 5. Rank selection is used. To perform recombination, with a given crossover probability, individuals with the highest aggregated fitness are selected.

5.1 The algorithm uses elitist reinsertion. A percentage of the best ranked (most elite) individuals are directly copied to the next population. This is done with the aim of preventing the loss of the best-found solutions.

Step 6. With a pre-defined crossover probability, new offspring (children) are generated using a problem-specific crossover operation. For the exploitation of the combinatorial structure within the search algorithm, a specific crossover operator denominated subsystem rotation crossover, SURC, was used (as shown in Chapter 5). In this step, multi-parent recombination is allowed. This action, and the way that SURC

works, produces a large number of children in the mating pool, creating a large number of diverse solutions to choose from.

Step 7. Single-point mutation is used. With a pre-defined mutation probability, mutate new offspring at a random position in the chromosome.

Step 8. The algorithm uses elitist reinsertion in the aim of preventing the loss of the best-found solutions. New offspring plus a specified percentage of the most elite individuals from the previous population are chosen to form the new population.

Step 9. Use new generated population for a further run (generation) of the algorithm

Step 10. If the Generation i = Generation 'max', stop, and return the best solutions in current population, otherwise return to step 2.

7.5 Examples

Three examples are presented to illustrate the problem addressed. The three examples consider three main manufacturing stages connected in series.

7.5.1 Example 1

This example considers three main units connected in series. For each series subsystem, there are several machines available in the market that can be chosen to provide redundancy. Each of the machines can perform three different tasks. Each task is considered to be binary capacitated, meaning that it can have only two states, functioning with the nominal capacity or total failure, corresponding to capacity 0. The collective performance of these binary components leads to a multi-state multi-task system behavior. Each task is characterized by its availability and nominal capacity (production rate in parts/hour). Table 7.1 shows the characteristics of the available machines. Without loss of generality, task capacities can be measured as a percentage of the maximum

demand. Table 7.2 presents different demand levels for each task, for a given period, known as the cumulative demand curve.

Table 7.1. Characteristics of the machines available

Subsystem l	Machine i	Task k	Availability p_{ik}^l	Production rate (parts/hour) g_{ik}^l
1	1	1	0.50	30
		2	0.80	20
		3	0.60	40
	2	1	0.80	30
		2	0.45	16
		3	0.60	50
	3	1	0.90	25
		2	0.85	18
		3	0.65	44
2	1	1	0.80	40
		2	0.90	24
		3	0.60	60
	2	1	0.70	32
		2	0.80	28
		3	0.60	50
	3	1	0.50	24
		2	0.60	12
		3	0.90	52
3	1	1	0.85	28
		2	0.80	28
		3	0.75	42
	2	1	0.65	38
		2	0.70	20
		3	0.60	48
	3	1	0.90	30
		2	0.60	28
		3	0.70	56

Table 7.2. Parameters of the cumulative demand curve

Task 1		Task 2		Task 3	
Demand (%)	Duration (%)	Demand (%)	Duration (%)	Demand (%)	Duration (%)
100	0.60	70	0.65	140	0.50
60	0.30	50	0.25	95	0.35
40	0.10	20	0.10	40	0.16

The multi-objective formulation considers three objectives to be satisfied simultaneously: maximization of the availability of producing task 1, maximization of the availability of producing task 2 and maximization of the availability of producing task 3.

Since a particular configuration may not simultaneously maximize the probability of meeting demand for each of the individual tasks, the problem becomes a multi-objective problem with K objectives to be maximized.

The problem was solved using the developed algorithm, with a population size of 100 and 20 generations, and a restriction that the maximum number of machines to be used was six. The algorithm was fully coded in MATLAB® 7.0 and run on a Sony VAIO computer, with an Intel Pentium processor operating at 1.86 GHz and 1 GB of RAM. The computation time was 1,115.73 seconds.

Figure 7.2 shows the 29 solutions found in the Pareto front. To better visualize the solutions obtained, Figure 7.3 shows different two dimensional representations of the same solutions.

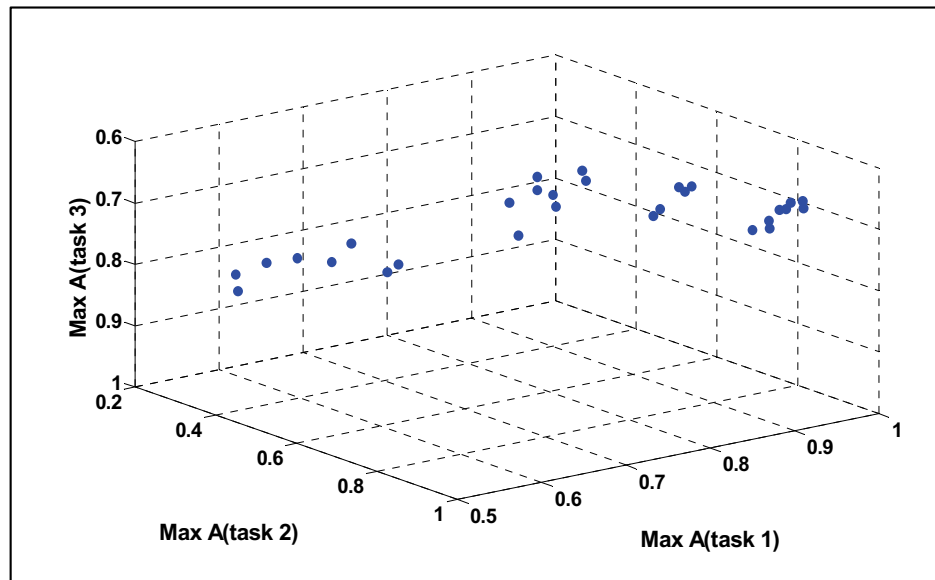


Figure 7.2. Pareto front of example 1

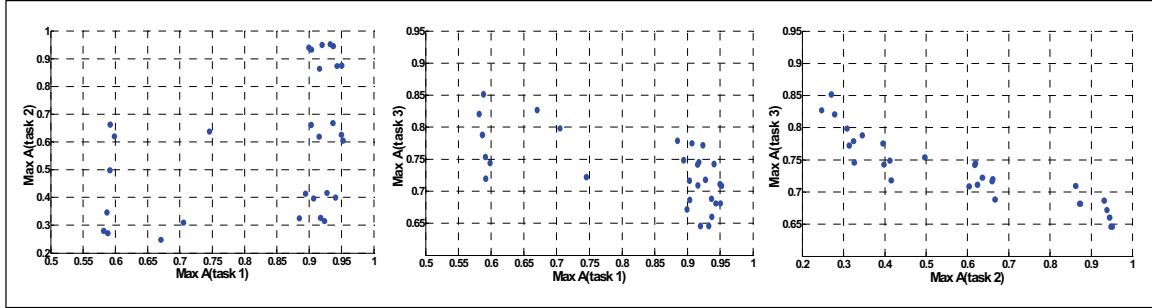


Figure 7.3. Pareto front of example 1 in a two dimensional space

Once the Pareto-optimal set is obtained, the decision-maker has to decide which of the non-dominated points to choose as the solution to the problem. For this example, if the three tasks are considered to be equally important, among the 29 solutions found in the final Pareto-optimal set, the decision-maker can choose the solution that is closest to the ideal vector. In multi-objective optimization, for each of the objectives there exists an ideal value in the objective value search space (Deb, 2002). Since in this case, we want to maximize simultaneously the three objective functions (task availability), the ideal vector would be $\mathbf{z}^{ideal} = (1, 1, 1)$. For this example, solution number 7 is the closest solution to this ideal vector. The system design configuration corresponding to solution number 7 is shown in Table 7.3.

However, in many multi-objective problems, there exists the case in which the achievement of one objective is more important than the others. If, for this particular example, the maximization of task 2 is considered to be more important than the maximization of task 1, and if the maximization of task 1 is more important than the maximization of task 3; that is, $f_2 \succ f_1 \succ f_3$, then by applying the non-numerical ranking preferences method (Taboada & Coit, 2006a; Taboada *et al.*, 2007a), a preferred sub-set of Pareto solutions can be selected. Based on this analysis, a solution that would clearly

reflect these objective function preferences would be solution number 5 as shown in Table 7.4.

Table 7.3. Chosen design configuration for example 1

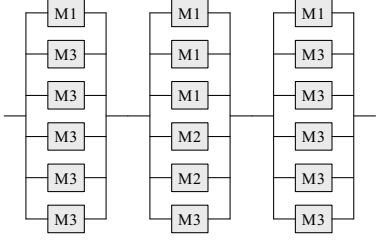
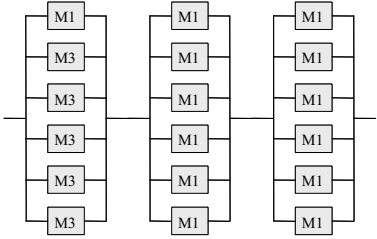
Sol No.	System configuration diagram	Availability task 1	Availability task 2	Availability task 3
7		0.91584	0.86234	0.70937

Table 7.4. Chosen design configuration for example 1 when considering $f_2 \succ f_1 \succ f_3$

Sol No.	System configuration diagram	Availability task 1	Availability task 2	Availability task 3
5		0.93295	0.95100	0.64605

7.5.2 Example 2

The second example is similar except that system cost has been added as an objective to be minimized. This example also consists of three main units connected in series. For each series subsystem, there are several machines available in the market that can be chosen to provide redundancy. Each of the machines can perform three different tasks. Each task is considered to be binary capacitated, meaning that it can have only two states, functioning with the nominal capacity or total failure, corresponding to capacity 0. The collective performance of these binary components leads to multi-state multi-task system behavior. Each task is characterized by its availability, nominal capacity (production rate in parts/hour), and cost. Table 7.5 shows the characteristics of the machines available.

Table 6 presents different demand levels for each individual task, for a given period.

Table 7.5. Characteristics of the machines available

Subsystem l	Machine i	Task k	Availability p_{ik}^l	Production rate, (parts/hour) g_{ik}^l	Machine Cost
1	1	1	0.80	100	65
		2	0.60	200	
		3	0.80	120	
	2	1	0.65	90	60
		2	0.75	160	
		3	0.77	180	
	3	1	0.90	75	50
		2	0.85	180	
		3	0.65	200	
	4	1	0.60	100	80
		2	0.90	150	
		3	0.75	160	
2	1	1	0.70	120	120
		2	0.65	160	
		3	0.87	150	
	2	1	0.77	100	150
		2	0.58	178	
		3	0.76	100	
	3	1	0.60	130	200
		2	0.80	150	
		3	0.53	180	
3	1	1	0.85	400	200
		2	0.86	200	
		3	0.85	150	
	2	1	0.65	450	190
		2	0.78	220	
		3	0.61	270	
	3	1	0.90	300	240
		2	0.63	280	
		3	0.70	220	

Table 7.6. Parameters of the cumulative demand curve

Task 1		Task 2		Task 3	
Demand (%)	Duration (%)	Demand (%)	Duration (%)	Demand (%)	Duration (%)
250	0.60	190	0.70	280	0.70
220	0.20	150	0.20	200	0.20
170	0.15	100	0.10	150	0.10
150	0.05				

In this case, the multi-objective formulation considered four objectives to be satisfied simultaneously: maximization of the availability of producing task 1, maximization of the availability of producing task 2, maximization of the availability of producing task 3, and

minimization of the overall system cost. The problem was solved using the developed algorithm, with a population size of 200 and 20 generations, and 5 as the maximum number of machines to be used. The computation time was 1,270.35 seconds.

Figure 7.4 shows, in a three dimensional perspective, the 134 solutions found in the Pareto front. To better visualize the solutions obtained, Figure 7.5 shows different representations of the two dimensional plots of the same solutions.

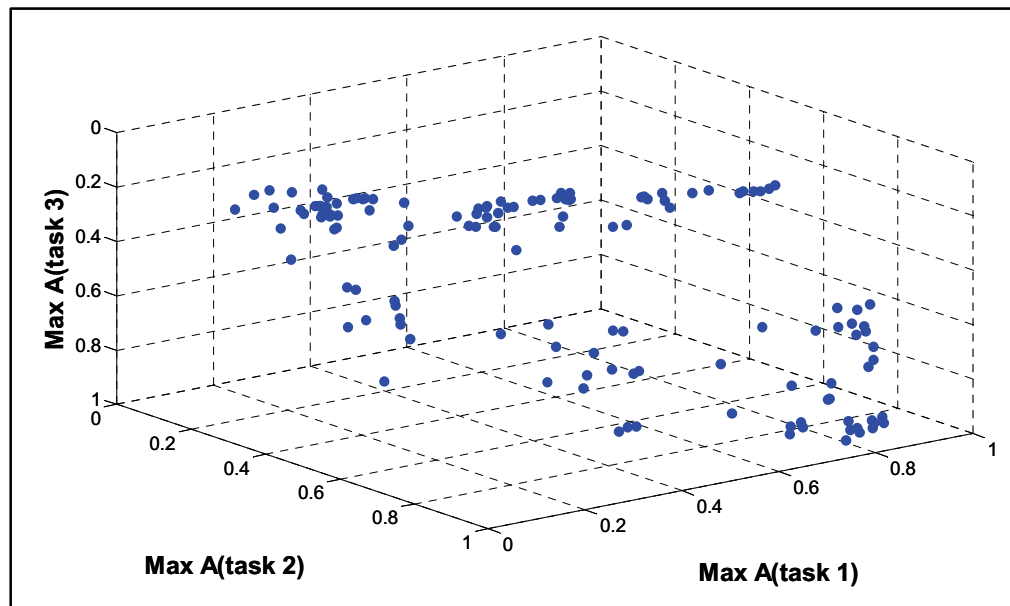


Figure 7.4. Pareto front of example 2 in a three dimensional space

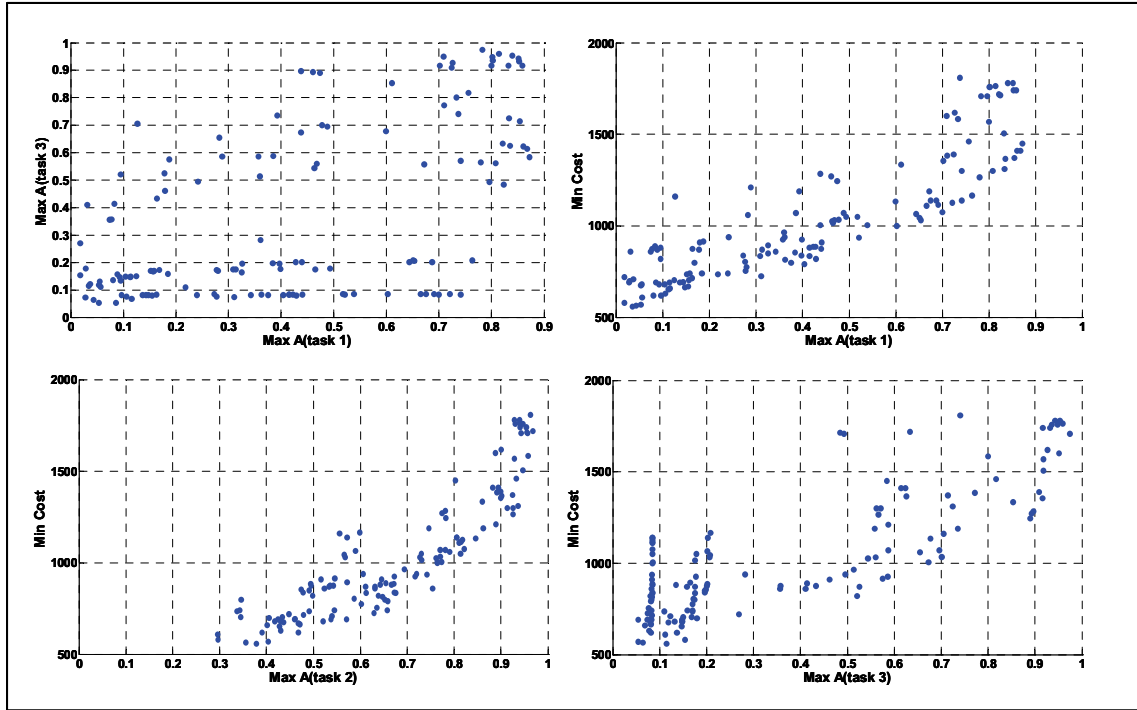


Figure 7.5. Pareto front of example 2 in a two dimensional space

To choose a solution from this Pareto set, the decision-maker may have a more difficult decision because now, with four objectives, the solutions are harder to visualize. However, according to problem-specific knowledge, there are regions in the Pareto set which express good compromises and these regions can be identified. More detail on methods to be applied in the decision-making stage to reduce the size of the Pareto-optimal set, and obtain a smaller representation of the multi-objective design space can be found in Taboada & Coit (2007) and Taboada *et al.* (2007a). In this case, an example solution from the “*knee*” region (Das, 1999; Branke *et al.*, 2004; Taboada & Coit, 2007) is presented as a good compromise. The “*knee*” is formed by those solutions of the Pareto-optimal front where a small improvement in one objective would lead to a large deterioration in at least one other objective. Table 7.7 shows an example design configuration from this region with its respective objective function values obtained.

Table 7.7. Compromised example design configuration for example 2

Sol No.	System configuration diagram	Availability task 1	Availability task 2	Availability task 3	Cost
85		0.83353	0.93668	0.72435	1310

7.5.3 Example 3

The third example considers more complex system behavior because each available machine can exhibit degraded system performance. This is also a realistic formulation because many actual components degrade with time. The example consists of three main units connected in series. For each unit, there are several machines available to choose from to provide redundancy. Three different tasks need to be completed. For each unit, there are several components available in the market that can be chosen to provide redundancy. Each machine can perform three different tasks. Each of these tasks can have different levels of performance, which range from maximum capacity to total failure. Each component is characterized by its availability (p_{ik}^l), nominal capacity (production rate in parts/hour), and cost. Table 7.8 shows the characteristics of the machines available and Table 7.9 presents the system cumulative demand curve.

Table 7.8. Characteristics of the system elements available

Subsystem 1					Subsystem 2					Subsystem 3				
Machine i	Task k	Availability p'_{ik}	Production rate (parts/hour) g'_{ik}	Cost	Machine i	Task k	Availability p'_{ik}	Production rate (parts/hour) g'_{ik}	Cost	Machine i	Task k	Availability p'_{ik}	Production rate (parts/hour) g'_{ik}	Cost
1	1	0.70	90	60	1	1	0.75	80	100	1	1	0.80	120	100
		0.20	30				0.15	50				0.15	60	
		0.10	20				0.10	10				0.05	20	
	2	0.60	100			2	0.65	110			2	0.80	60	
		0.30	40				0.25	20				0.15	40	
		0.10	10				0.10	10				0.05	10	
	3	0.80	120			3	0.85	90			3	0.75	95	
		0.10	20				0.10	60				0.15	50	
		0.10	15				0.05	10				0.10	25	
2	1	0.90	110	80	2	1	0.80	110	70	2	1	0.85	90	150
		0.05	40				0.15	30				0.10	50	
		0.05	20				0.05	5				0.05	10	
	2	0.80	80			2	0.78	90			2	0.75	120	
		0.15	30				0.12	15				0.20	30	
		0.05	10				0.10	10				0.05	10	
	3	0.70	90			3	0.65	130			3	0.65	100	
		0.20	60				0.25	40				0.30	50	
		0.10	5				0.10	15				0.05	30	
3	1	0.60	80	90	3	1	0.90	130	80	3	1	0.75	100	130
		0.30	50				0.05	50				0.20	40	
		0.10	20				0.05	10				0.05	30	
	2	0.85	120			2	0.85	90			2	0.85	70	
		0.10	20				0.10	20				0.10	40	
		0.05	10				0.05	5				0.05	20	
	3	0.75	70			3	0.70	80			3	0.80	140	
		0.15	30				0.20	50				0.15	20	
		0.10	10				0.10	10				0.05	10	

Table 7.9. Parameters of the cumulative demand curve

Task 1		Task 2		Task 3	
Demand (%)	Duration (%)	Demand (%)	Duration (%)	Demand (%)	Duration (%)
160	0.60	130	0.65	140	0.50
120	0.20	90	0.15	100	0.20
80	0.15	60	0.10	80	0.15
40	0.05	30	0.10	50	0.10
				20	0.05

The multi-objective formulation again considered four objectives to be satisfied simultaneously: maximization of the availability of producing task 1, maximization of the availability of producing task 2 and maximization of the availability of producing task 3, and minimization of the system cost. The problem was solved using the developed algorithm, with a population size of 200 and 20 generations. 110 solutions were found in the final Pareto-optimal set. Figure 6 shows the Pareto front obtained in a three dimensional perspective, and to better visualize the solutions obtained, Figure 7.7 shows different representations of the two dimensional plots of the same solutions. Table 7.10 shows three example design configurations chosen from the “*knee*” region with their respective objective function values.

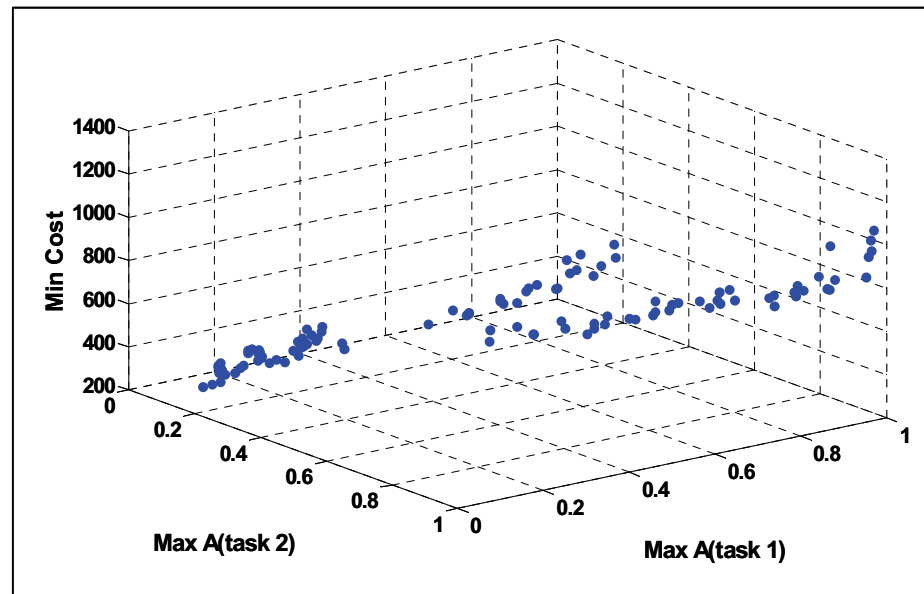


Figure 7.6. Pareto front of example 3 in a three dimensional space

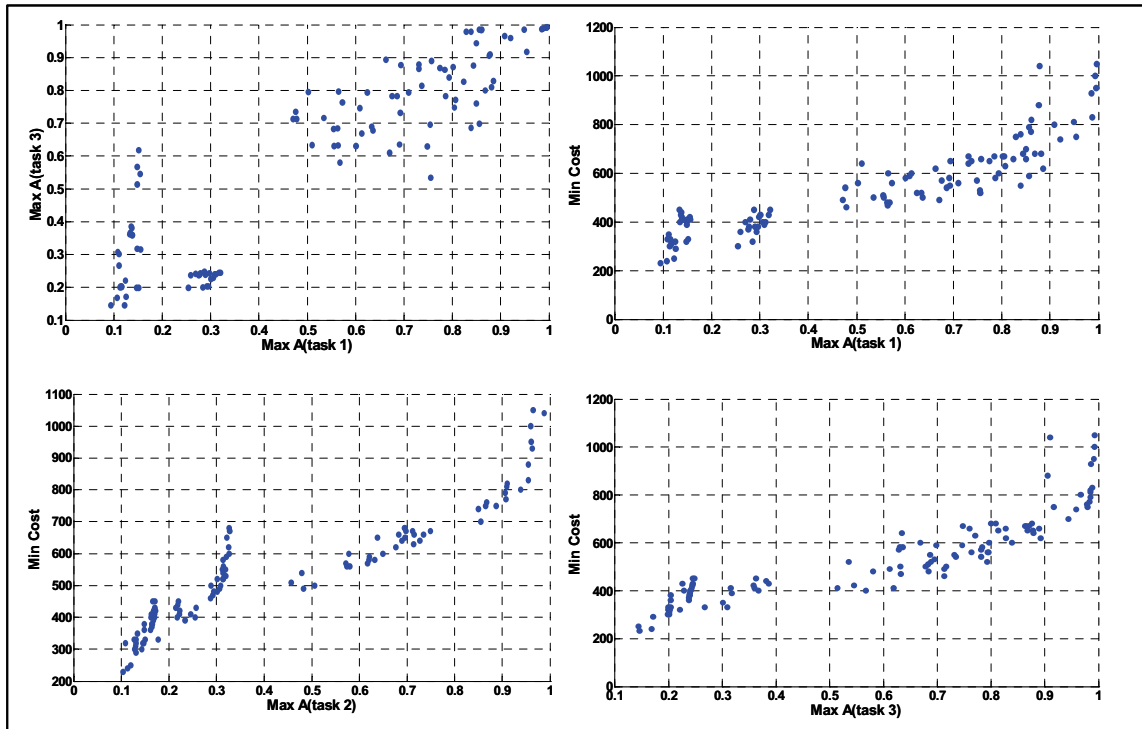


Figure 7.7. Pareto front of example 3 in a two dimensional space

Table 7.10. Example design configurations of Example 3

Sol No.	System configuration diagram	Availability task 1	Availability task 2	Availability task 3	Cost
22		0.92104	0.85079	0.95908	740
101		0.80647	0.71388	0.77134	630
107		0.84999	0.85571	0.94427	700

7.6 Summary

This chapter presented a multiple objective evolutionary algorithm to determine manufacturing system configurations of multi-state multi-task production systems based on availability analysis. Availability was used in the context of multi-task production systems to select a particular configuration that maximized the probability of meeting a required demand for each specific task, or the expected productivity for each task. A particular configuration may not simultaneously maximize the probability of meeting demand for each of the individual tasks, and thus, the problem was treated as a multi-objective optimization problem. A multi-objective evolutionary algorithm was developed to solve the problem. Three different examples were presented to illustrate the problem.

8. A multi-objective prioritized evolutionary algorithm (MoPriGA)

This chapter presents a new MOEA that conceptually combines the idea of the working mechanism of MOEA-DAP and post-Pareto pruning, introduced in Chapters 5 and 4, respectively. MoPriGA, incorporates the knowledge of the DM objective function preferences based on direct exploitation of the uncertain weight function, $f_w(\mathbf{w})$, into the search process. The initial pruning selection criterion, as well as the two different fitness metrics that are incorporated in the algorithm, enable the search process to explore the most promising region of the solution space based on the DM objective function preferences. This MOEA directly searches in the most promising region, and thus, no pruning is required, resulting in a much more efficient search for good solutions.

8.1 Introduction

A new MOEA that conceptually combines the MOEA-DAP and post-Pareto pruning has been developed. In Chapters 5 through 7, different MOEAs were developed to solve a wide variety of multi-objective optimization problems that share some similar characteristics. In these MOEAs, the optimization was initially performed without the input of the DM. That is, the solutions obtained in the final Pareto front were independent from the DM objective function preferences. MOEAs, as shown in previous chapters, offer many advantages to solve MOPs. However, the biggest disadvantage of these methods can be that the DM has potentially too many solutions in the final Pareto set. Therefore, he/she needs to perform additional steps before obtaining or selecting a trade-

off solution. To satisfy this new requirement, post-Pareto analysis was introduced. Chapter 4 presented two different techniques to perform post-Pareto analysis. Both methods presented in that chapter supported screening in the final Pareto set and, in different manners, efficiently determined a smaller and an attractive Pareto sub-set from which later, the DM could easily select the most desirable solution for system implementation.

In the first method, the non-numerical ranking preferences method, the DM was only asked to rank non-numerically (in order of relative importance) the objective functions but did not have to select specific weight values. Based on the DM objective function preferences, an uncertain weight function was generated. Then, different weight combinations reflecting the DM preferences were generated numerous times from the uncertain weight function. The solutions that this method yielded are those that clearly satisfied the given objective function preferences (see Figure 8.1). The second method, post-Pareto clustering, showed the capacity to automatically identify an optimal number of clusters in the Pareto-optimal set, clustering optimal solutions that shared similar properties and, providing the DM with representative solutions of each cluster. Although both of these methods to perform post-Pareto analysis have shown to be effective, their use, in practice, can be potentially inefficient.

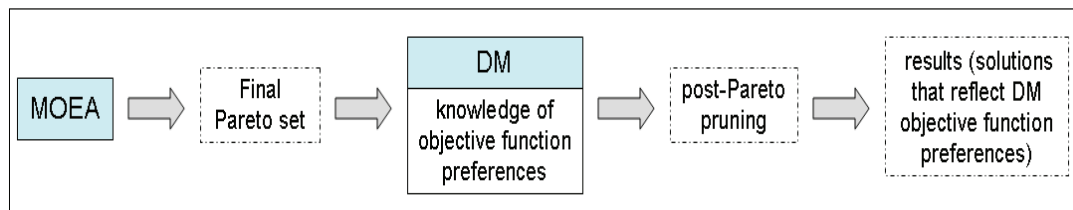


Figure 8.1 Approach to obtain solutions that reflect DM objective function preferences (after the search)

This chapter presents a new MOEA that conceptually combines the working mechanism of MOEA-DAP (introduced in Chapter 5) and post-Pareto pruning (using the

non-numerically ranking preferences method). This newly developed algorithm enables the search process to move along predefined objective function preferences without the burden of having to select specific weight values. An early pruning selection criterion, as well as the two different fitness metrics that are incorporated into the algorithm, enable the search process to move according to the DM preferred solutions (see Figure 8.2). Consequently, the biggest advantage of this algorithm is that reducing the size of the solution set does not really require higher level decision making to be incorporated into the algorithm to direct the search, such as in the case of having to specify a reliable utility function.

Moreover, the idea of this algorithm is not to reduce the capability of the search, but simply to more intensely focus on the region of the Pareto set of interest to the DM. This is accomplished by providing external initial objective function preferences information but still ensuring that the preference relationships introduced in the MOEA preserve existing dominance relationships. Otherwise, the search would be biased towards undesired, or sub-optimal, regions of the search space.

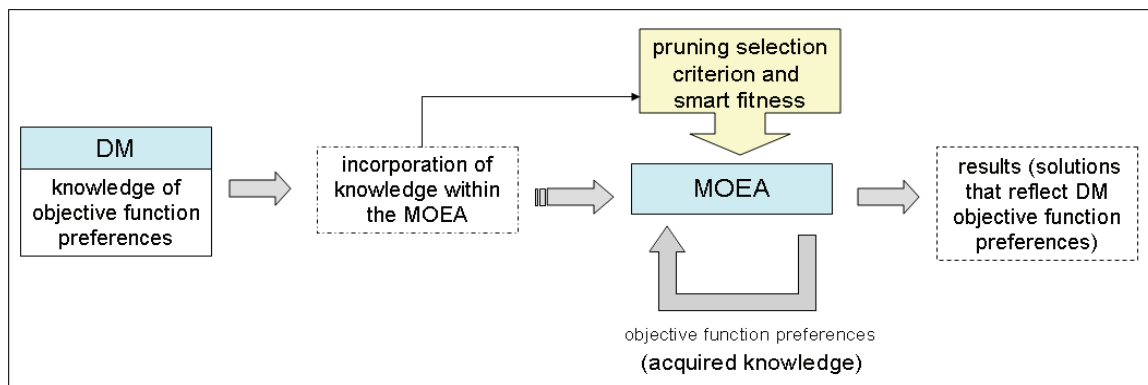


Figure 8.2 New approach to incorporate DM objective function preferences (during the EA search)

8.2 Previous research on post-optimality selection

Most of the current research on evolutionary multi-objective optimization has

concentrated on issues related to the design of evolutionary algorithms to search for nondominated solutions. However, these nondominated solutions do not provide any insight into the process of decision making itself. Once the Pareto set is obtained, post-Pareto analysis has been proposed as an afterward required step to help in the decision-making process. In this area, little, but significative, research has been done. When performing post-Pareto analysis, two different areas can be distinguished. The first one involves the analysis of the Pareto set when all the attributes are considered to have equal importance or, in other words, when the DM does not express any preferences of the attributes. In this case, the research efforts have been focused in locating the region(s) of the solution space where balanced trade-offs can be found, either by locating the “knee” solutions or by presenting to the DM a considerably smaller number of solutions that are still representative of all of the solution space. The second area directs its efforts to find those solutions of the Pareto set that satisfy objective function preferences. In practice, if objective function preferences are known, the decision-maker wishes to evaluate a limited number of Pareto-optimal solutions. In theory, these solutions should be a sub-set of the Pareto front that satisfies the DM objective function preferences.

8.3 Defining preferences in MOEAs

As presented by Horn (1997), when using EAs, preferences can be expressed *a priori*, *a posteriori*, or *during* the search. If preferences are expressed *a priori*, the DM has to define his/her preferences in advance (before actually performing the search). The most classical examples of this category are the aggregating approaches in which weights are specified beforehand to combine all the objectives into a single objective function. The second approach pertains to the *search-first-and-decide later* case. This is the category in

which most evolutionary multi-objective approaches can be classified. In this case, EAs are used to search for the “best possible” solutions, and this normally implies that the EA attempts to find the nondominated or Pareto-optimal solutions. Then, as presented in Section 8.2, the DM has to further investigate these solutions to do post-optimal selection. As explained before, post-Pareto optimality is a challenging problem by itself. The third category is the least common in the EA literature. The articulation of preferences *during* the search (information incorporated within the EA) can be further divided into two sub-classes:

- (i) approaches that allow guiding the search of the EA using preferences from the DM but require interaction with the DM, and
- (ii) approaches that allow guiding the search of the EA using preferences from the DM without the need of interaction with the DM.

The proposed algorithm, MoPriGA, pertains to this second sub-class of algorithms. In reality, little attention has been devoted to the development of methods that incorporate the DM objective preferences within (during) the EA. However, a brief overview on these methods is presented next.

As described in Coello Coello (2000), Fonseca & Fleming (1993) presented the earliest attempt to incorporate preferences from the DM into an EA. They basically extended their developed MOGA to accommodate goal information as an additional criterion for non-dominance to assign ranks to the population. The goal attainment method was used for this purpose, so that the DM could specify goals at each generation. As can be noticed, this is an interactive approach. Shaw & Fleming (1997) discussed how a similar approach could be used to incorporate preferences into a production

schedule algorithm, but in their case, the preferences were defined *a priori*. As discussed in Coello Coello (2000), the main disadvantage of this approach is that it requires the user to know beforehand the ranges of variation of each objective.

Cvetkovic & Parmee (2000) developed a preference algorithm to transform linguistic (qualitative) information into real numbers to obtain the ranking of several objectives. Their method helps to find a specific set of weights that, in theory, satisfies the ranking of preferences by the DM. Then, this specific set of weights can be used in a GA to guide the search to preferred solutions. Wang *et al.* (2005) proposed a method to solve multi-objective and multi-constraint problems. In their method, the DM has to specify his/her objective function preferences by means of weights since the very start of the process. Their algorithm considers the satisfaction of the constraints as a new objective and uses a multi-criteria method to rank the members of the EA population at each generation based on the weights specified by the DM. These two methods are considered to be also *a priori* approaches, since the weights are assumed constant throughout the optimization process; however nothing in these approaches really excludes their use in an interactive way.

8.4 Description of the multi-objective prioritized GA (MoPriGA)

MoPriGA begins its search with a population of random, but feasible, solutions. This initial set of solutions is called the **W** set (for simple explanation of the working mechanism of this algorithm, consider that the **W** set, has a population size of 20 individuals). Immediately, thereafter, objective function values are evaluated. Then, the number of individuals that each individual dominates (dominance count) is recorded, and the Pareto dominance criterion is evaluated for the initially created solutions. Thus, the

solutions that are dominated by other solutions are eliminated. In MoPriGA, this set of non-dominated solutions is called the **X** set. To continue with on this example, consider that after applying the non-dominance criterion there are only 10 solutions in the **X** set. Then, pruning selection is applied to the initial set based on DM preferences. This pruning selection uses the uncertain weight function $f_w(\mathbf{w})$, initially introduced in Chapter 4, that is generated based on the DM objective function preferences. As mentioned in that chapter, the strength of this method is precisely that the DM only ranks non-numerically (in order of relative importance) the objective functions but does not have to select specific weight values or value or utility functions. Then, a large number of random but ranked weights are generated, with each set containing one weight for each objective. These weights are uniformly sampled from the region of interest that satisfies the following: $w_{(1)} > w_{(2)} > \dots > w_{(n)}$, $w_{(1)} + w_{(2)} + \dots + w_{(n)} = 1$ and $w_{(i)} \geq 0$ for $i = 1, 2, \dots, n$. After obtaining the set of ranked weights (for instance, consider 5,000), the first set of weights is multiplied by each of the nondominated solutions in a normalized space: $f' = w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x}) + \dots + w_n f_n(\mathbf{x})$. The solution that yields the minimum value for f' , for the first random weight set is recorded, and gets a counter of 1. Thus, we do the same with the remaining set of weights. At the end, the solutions that have non-zero counter values are those solutions that form the pruned Pareto set. In MoPriGA, this set of preferred solutions is called the **Z** set and, for purposes of this example assume that this set contains three solutions. The rest of the solutions (7), which are not in the preferred set, are placed in the **Y** set. Then, MoPriGA uses the **Z** set for two different purposes. The first purpose is to place this set to the very top of the list of ranked solutions. In this way, once selection is performed, the solutions in the **Z** set are always chosen for reproduction.

The second purpose of the \mathbf{Z} set is to be used as a form of elitism. That is, to prevent the loss of the best found solutions (solutions that clearly reflect the DM objective function preferences), the \mathbf{Z} set is directly copied to be part of the population in next generation.

As described earlier, the \mathbf{Y} set contains the nondominated and non-preferred solutions (7 solutions in this example), and, for each of these, its corresponding dominance count has been retained from a previous step. With this \mathbf{Y} set, we proceed to assign fitness to these solutions. In MoPriGA, two different methods are used to assign fitness to the solutions. In this case, the first fitness metric, $f_1(i)$, aims to select those individuals which are more dominating, and they are classified into discrete intervals. The way that this is done follows the same steps explained in Chapter 5. But in simple words, we can say that solutions with highest dominance count receive highest fitness. Then, for the second fitness metric, the Euclidean distance, $d(\mathbf{z}^{\text{best}}, \mathbf{Y})$, is calculated between the “best” solution, \mathbf{z}^{best} , (the solution with the largest non-zero counter value) in the \mathbf{Z} set and, the rest of the solutions in the \mathbf{Y} set. In this way, the solutions in \mathbf{Y} that are closest to \mathbf{z}^{best} receive highest fitness. For this second fitness metric, we again considered discrete intervals. Next, the two different fitness metrics are then aggregated weighting each of the fitness metrics equally. Based on this aggregated fitness, the strongest solutions are copied to complete the list of ranked solutions, which already contains the \mathbf{Z} set. Thus, following our example, in the list of ranked solutions, there are three solutions from the \mathbf{Z} set and, the seven strongest solutions (based on the aggregated fitness values) from the \mathbf{Y} set.

The subsequent steps (selection, recombination, and mutation) in the MoPriGA algorithm follow the same behavior explained in Chapter 5 for the construction of

MOEA-DAP. As it can be noticed, from Figure 8.3, for the selection step, rank selection was used. The number of individuals to be selected for the recombination step is dictated by the desired crossover probability. For instance, in our example, the ranked list has 10 solutions (N_{ranked}), and if the specified crossover probability, P_{cross} , is 0.7, then the number of parents to be selected to perform recombination is $N_{parents} = \text{round}(P_{cross} \times N_{ranked})$. In this specific example, there are $\text{round}(0.7 \times 10) = 7$ parents. Then the crossover step takes place. In this step, multi-parent recombination is allowed. This action produces a large number of children in the mating pool, creating a large number of diverse solutions to choose from. Diversity is considered favorable, as the greater the variety of genes available to the GA, the greater the likelihood of the system identifying alternate solutions. Moreover, maintaining diversity of individuals within a population is necessary for the long term success of any evolutionary system. Finally, in this case, 17 children are randomly selected from the mating pool to undergo mutation, and these individuals, plus the ones already in the elite list, form the next population.

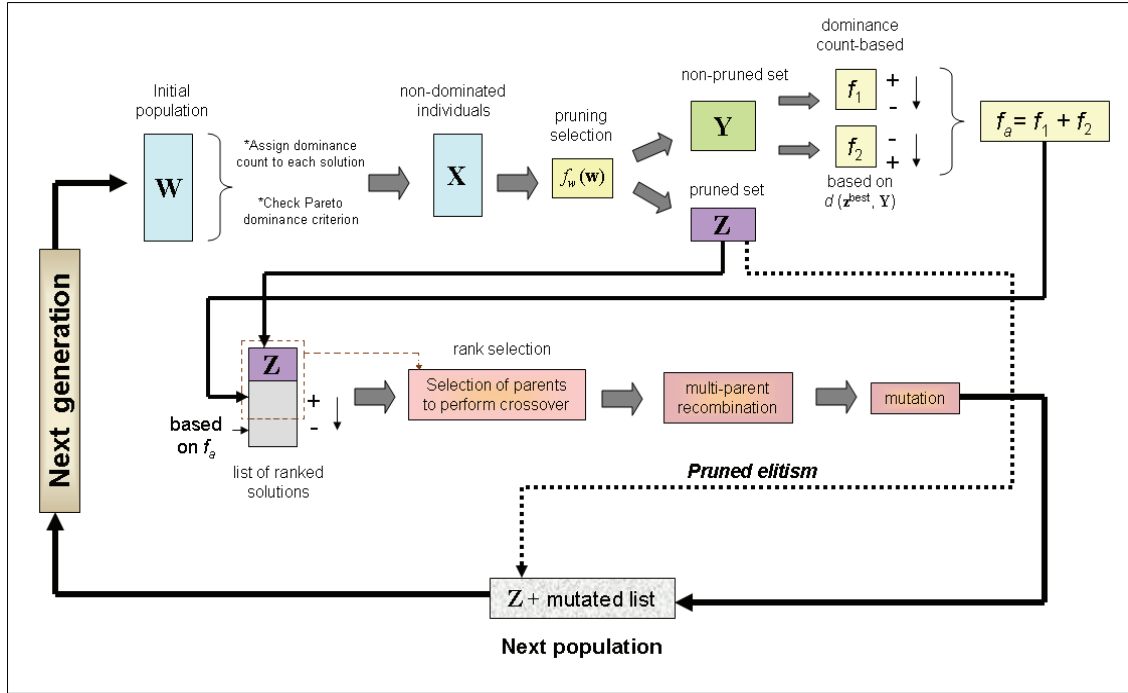


Figure 8.3 Working mechanism of MoPriGA

In this way, MoPriGA, incorporates the knowledge of the DM objective function preferences, based on the formulation of the uncertain weight function, $f_w(\mathbf{w})$, into the search process. That is, this newly developed algorithm enables the search process to move according to the DM preferred solutions without asking the DM to select specific weight values. The initial pruning selection criterion, as well as the both fitness metrics incorporated in the algorithm guide the search considering the DM objective preferences. The examples in Section 8.5 show that MoPriGA is a powerful algorithm that searches extensively in the region of interest without reducing the capability of the search, but simply intensifying the search on the region of the Pareto set of interest to the DM.

8.5 Examples

The multi-objective formulation that we considered is the same that was used in Chapter 5 and, it is again shown in Equation 8.1, with the system reliability to be

maximized, cost and weight of the system to be minimized, and no constraints in the possible values of reliability.

$$\max \left[\prod_{i=1}^s R_i(\mathbf{x}_i) \right], \min \left[\sum_{i=1}^s \sum_{j=1}^{m_i} c_{ij} x_{ij} \right], \min \left[\sum_{i=1}^s \sum_{j=1}^{m_i} w_{ij} x_{ij} \right] \quad (8.1)$$

Subject to:

$$2 \leq \sum_{j=1}^{m_i} x_{ij} \leq n_{\max,i} \quad \text{for } \forall i = 1, 2, \dots, s$$

$$x_{ij} \in \{0, 1, 2, \dots\}$$

The example considered consists of a configuration of 3 subsystems, with an option of 5, 4 and 5 types of components in each subsystem, respectively. The optimization involves selection from among these component types. For the three cases presented next, 2 was considered to be the minimum number of components per subsystem and, the maximum possible number of components is 4 in each subsystem. Table 8.1 defines the component choices for each subsystem. For the three examples presented next, MoPriGA was run considering a population size of 150, and it was run for 100 generations.

Table 8.1 Component choices for each subsystem

Design Alternative j	Subsystem i								
	1			2			3		
	R	C	W	R	C	W	R	C	W
1	0.80	24	30	0.95	26	30	0.70	20	25
2	0.92	30	40	0.75	10	28	0.67	20	20
3	0.60	15	20	0.80	15	35	0.85	30	40
4	0.70	28	26	0.85	18	40	0.75	40	20
5	0.90	30	45				0.80	35	30

8.5.1 Case when $f_1 \succ f_2 \succ f_3$

The example in this case considers that $f_1 \succ f_2 \succ f_3$; that is, reliability is more important than cost, and cost is more important than weight. In this case, 11 solutions were found in final preferred Pareto set. These solutions are shown in Figure 8.4. The

same solutions are shown in Figure 8.5 in a two-dimensional perspective. The solution that had the maximum counter for the given preferences is emphasized in these figures.

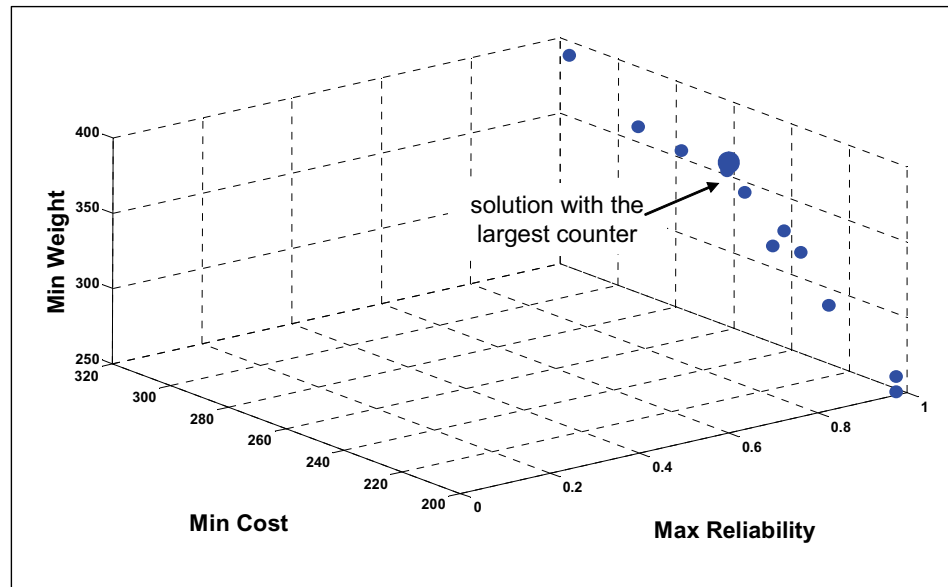


Figure 8.4 Preferred solutions found in the final Pareto set: Case when $f_1 \succ f_2 \succ f_3$

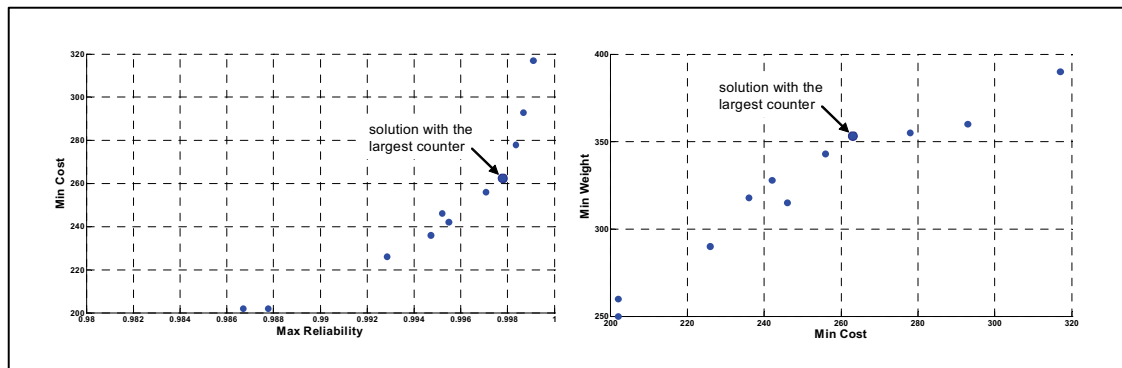


Figure 8.5 Preferred solutions found in the final Pareto set in a two-dimensional perspective

8.5.2 Case when $f_2 \succ f_1 \succ f_3$

The example in this case considers that $f_2 \succ f_1 \succ f_3$; that is, cost is more important than reliability, and reliability is more important than weight. In this case, four solutions were found in final preferred Pareto set. These solutions are shown in Figure 8.6. The same solutions are shown in Figure 8.7 in a two-dimensional perspective. The solution that had the maximum counter value for the given preferences is emphasized in these figures.

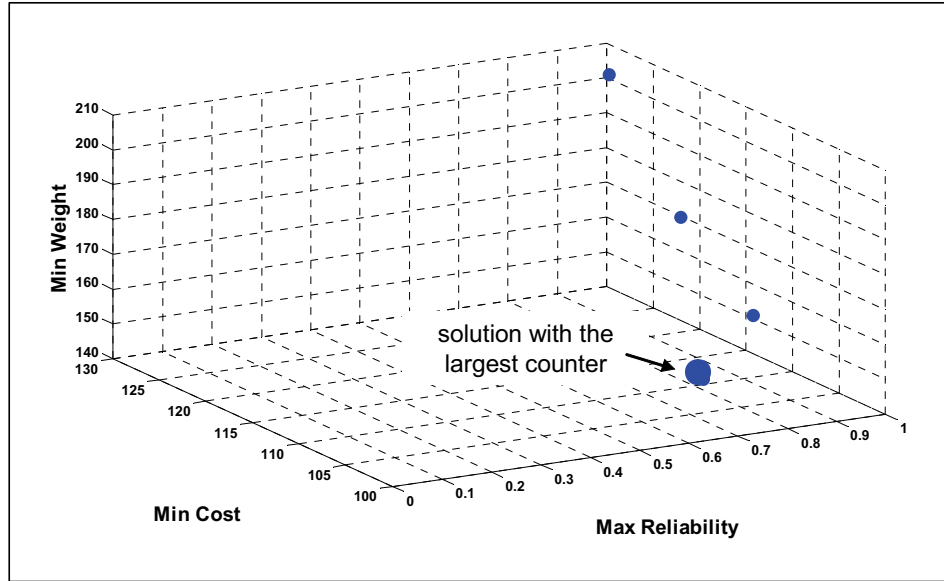


Figure 8.6 Preferred solutions found in the final Pareto set: Case when $f_2 \succ f_1 \succ f_3$

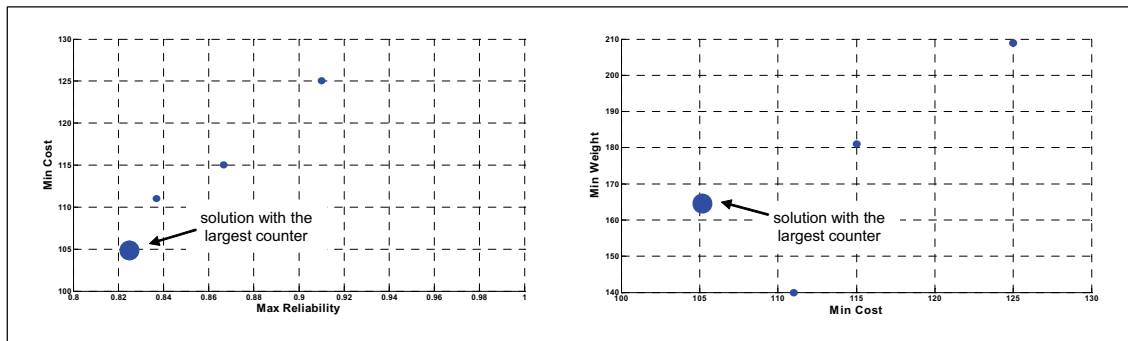


Figure 8.7 Preferred solutions found in the final Pareto set in a two-dimensional perspective

8.5.3 Case when $f_1 \equiv f_2 \succ f_3$

The example in this case considers that $f_1 \equiv f_2 \succ f_3$; that is, reliability is equally important than cost, and reliability and cost are both more important than weight. In this case, three solutions were found in final preferred Pareto set. These solutions are shown in Figure 8.8. The same solutions are shown in Figure 8.9 in a two-dimensional perspective. The solution that had the maximum counter value for the given preferences is emphasized in these figures.

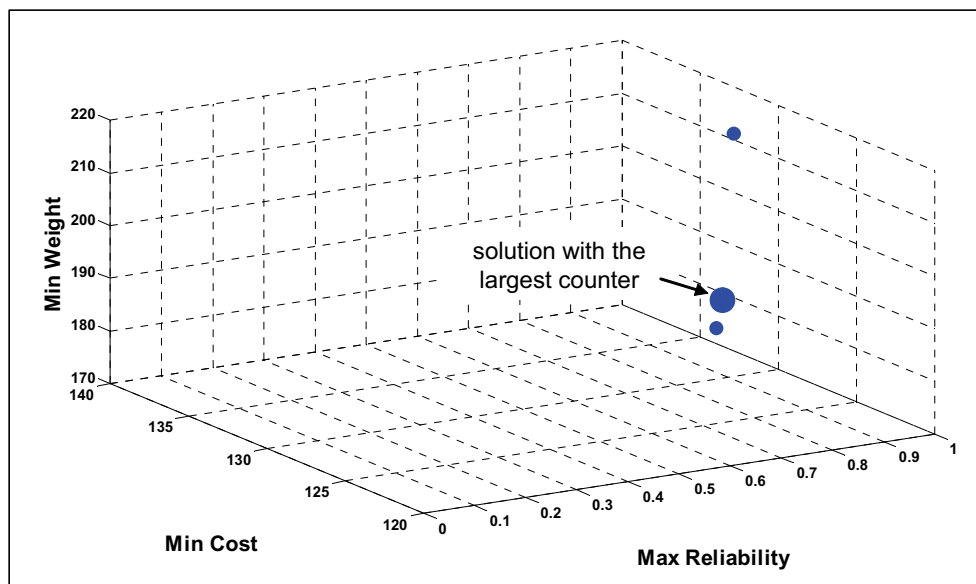


Figure 8.8 Preferred solutions found in the final Pareto set: Case when $f_1 \equiv f_2 \succ f_3$

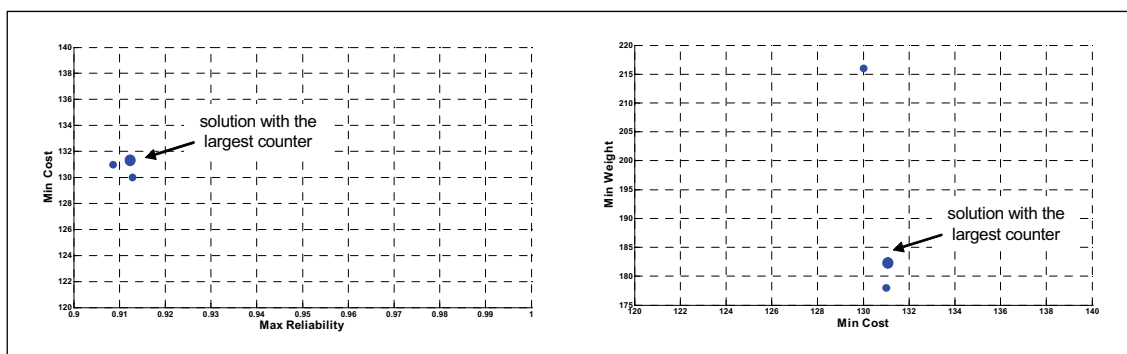
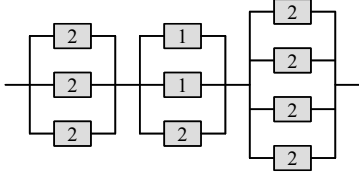
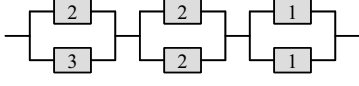
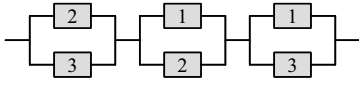


Figure 8.9 Preferred solutions found in the final Pareto set in a two-dimensional perspective

Table 8.2 shows three example design configurations with its respective system reliability, cost, and weight. Each of the presented design configurations corresponds to the solution with the largest counter value in each specific case.

Table 8.2 Example design configurations

Case when:	System Design Configuration Diagram	Reliability	Cost	Weight
$f_1 \succ f_2 \succ f_3$		0.997852	262	353
$f_2 \succ f_1 \succ f_3$		0.825820	105	166
$f_1 \equiv f_2 \succ f_3$		0.912880	131	183

8.6 Summary

This chapter presented a multi-objective prioritized GA (MoPriGA). MoPriGA is a powerful algorithm that searches extensively in the region of interest for the DM. MoPriGA incorporates the knowledge of the DM objective function preferences based on the inclusion of the uncertain weight function, $f_w(\mathbf{w})$, into the search process. This initial pruning selection criterion, as well as the two different fitness metrics that are incorporated into the algorithm, enable the search process to explore the most promising region of the solution space based on the DM objective function preferences. The strength of the algorithm is that it does not reduce the capability of the search, but simply intensifies the search on the region of the Pareto set that is of interest for the DM.

9. Future research

There are numerous opportunities for developing novel and original research in the evolutionary multi-objective optimization area. These opportunities can focus on (1) extending the existing results and models to address general problems, and (2) incorporating methods to accommodate uncertain problem decision variables.

9.1 Development of a multi-purpose MOEA

One challenge still remains for current multi-purpose MOEAs, which is the scalability problem, that is, the efficient solution of large scale problem instances. New multi-purpose MOEA are needed to analyze general combinatorial optimization problems.

Recently, various MOEAs have been proposed and applied, some of them are the non-dominated sorting genetic algorithms, (NSGA and NSGA-II), the strength Pareto evolutionary algorithm (SPEA), the Pareto archived evolutionary algorithm (PAES), among others. Although these MOEAs differ from each other, the common objective in all of them is to search for a near-optimal and uniformly diversified Pareto front. However, this ultimate goal is far from being accomplished by the existing MOEAs (Yang *et al.*, 2005). Thus, it is required to develop MOEAs with the ability of finding homogeneously distributed solutions in the final Pareto front with the robustness of balancing proximity and diversity during the searching process.

9.2 Handling uncertainties in MOEAs

In future research tasks, specific methods will be determined to accommodate uncertainties. Traditional MOEAs assume that information about the objectives can be obtained with total certainty. However, in real-world applications is very commonly the case that one must work with uncertainties (Parmee, 2001). This extension will pertain to the most realistic class of problems, multiple objective stochastic optimization problems. The new approach will incorporate additional risk metrics, to be minimized, into the multiple objective optimization framework.

Variance and standard deviation have been traditional risk measures in economics and finance since the pioneering work of Markowitz (1952). The two risk measures exhibit a number of nice technical properties. For example, the variance of a portfolio return is the sum of the variance and covariance of the individual returns. Furthermore, variance can be used as a standard optimization function. Finally, there is a well established statistical method to estimate variance and covariance. However, variance does not account for fat tails of the underlying distribution and therefore is inappropriate to describe the risk of low probability events, such as default risks.

Secondly, variance penalizes ups and downs equally. For instance, consider the example in Figure 9.1. In this case, reducing the standard deviation may happen but at the cost of eliminating good outcomes as well as bad ones

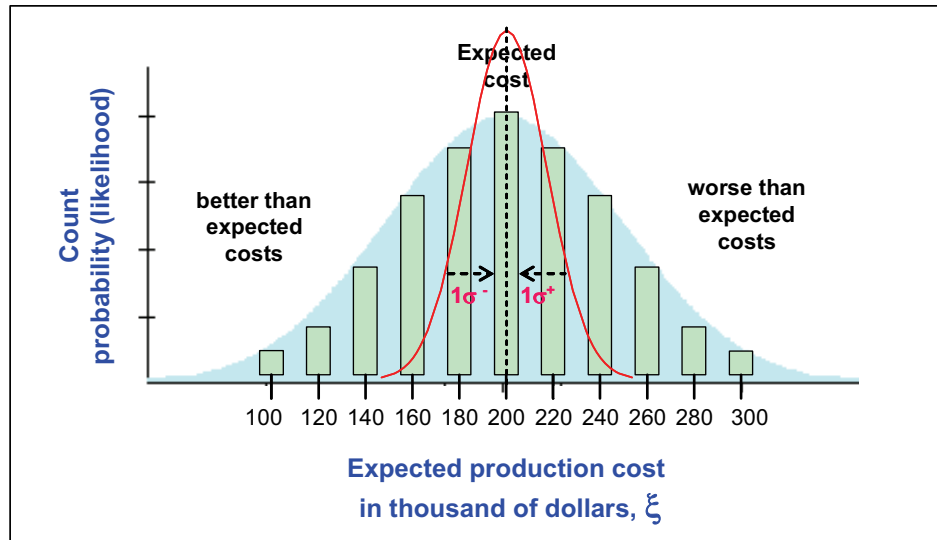


Figure 9.1 Variance as a Risk measure

Then, more appropriate measures of risk that recognize the logic of eliminating good outcomes from bad outcomes. Several risk measures have been developed in the past that overcome the deficiencies of variance or standard deviation to measure risk. However, they are mainly used in portfolio optimization, and one of the ideas is to develop or extend one risk measure that can be incorporated into a multiple objective evolutionary algorithm. This extension can be developed in such a way that this measure will be search-based, and the concept of risk will be incorporated in the search, i.e., a risk-based dominance criterion can be implemented in the selection step.

A prevalent method to measure risk is Value-at-Risk (VaR). The formal definition by Frey & McNeil (2002) which is derived from Artzner *et al.* (1999) is as follows: Given a loss L with probability distribution \mathbf{P} , the VaR of a portfolio at the given confidence level $\alpha \in [0,1]$ is represented by the smallest number l such that the probability that the loss L exceeds l is no larger than $(1-\alpha)$. Formally,

$$\text{VaR}_\alpha = \inf\{l \in \mathbf{R}, P(L > l) \leq 1 - \alpha\}$$

Moreover, VaR is a measure which captures the risk aspect of a low-probability/high-impact event. Thus, VaR may be a suitable risk measure, which can be adapted to the search in order to incorporate risk in the solution of multiple objective optimization problems.

Appendix A

This Appendix contains the runs performed to assess the performance of the developed MOEA-DAP algorithm. Tables A.1 through A.10 present the single runs from the NSGA-II algorithm. While Tables A.11 through A.20 present the single runs from the MOEA-DAP algorithm. Table A.21, contains the true Pareto-optimal front, Y_{true} .

Table A.1 Nondominated solutions in last Pareto front. Run 1, NSGA-II

Solution number	Original output			Non-duplicated nondominated solutions		
	Reliability	Cost	Weight	Reliability	Cost	Weight
1	0.99513	64	60	0.99513	64	60
2	0.99789	81	88			
3	0.99513	78	68			
4	0.99789	82	72	0.99789	82	72
5	0.99789	94	77			
6	0.99513	76	65			
7	0.9999	87	90	0.9999	87	90
8	0.99785	73	81	0.99785	73	81
9	0.99789	84	75			
10	0.99782	73	71	0.99782	73	71
11	0.99997	96	91	0.99997	96	91
12	0.99993	87	100			
13	0.99789	82	72			
14	0.99782	73	71			
15	0.99789	79	85	0.99789	79	85
16	0.99513	66	63			
17	0.99997	93	104	0.99997	93	104
18	0.99786	75	84	0.99786	75	84
19	0.99997	93	104			
20	0.99997	106	93			
21	0.99782	75	74			
22	0.99785	73	81			
23	0.99789	94	77			
24	0.99513	76	65			
25	0.99789	81	88			
26	0.9976	82	69	0.9976	82	69
27	0.99789	96	80			
28	0.99996	91	101	0.99996	91	101
29	0.99513	76	65			
30	0.99782	75	74			
31	0.99997	106	93			
32	0.99997	106	93			
33	0.99997	108	96			
34	0.99513	78	68			
35	0.99997	106	93			
36	0.99513	66	63			
37	0.99996	91	101			
38	0.99993	85	97	0.99993	85	97
39	0.99786	75	84			
40	0.99789	79	85			
41	0.9999	87	90			
42	0.9976	82	69			
43	0.99789	84	75			
44	0.99513	78	68			
45	0.99993	87	100			
46	0.99993	85	97			
47	0.99997	93	104			
48	0.99789	96	80			
49	0.99789	84	75			
50	0.99996	91	101			
Non-duplicated nondominated solutions				12		

Table A.2 Nondominated solutions in last Pareto front. Run 2, NSGA-II

	Original output			Non-duplicated nondominated solutions		
Solution number	Reliability	Cost	Weight	Reliability	Cost	Weight
1	0.99773	75	95	0.99773	75	95
2	0.99971	81	113	0.99971	81	113
3	0.99983	93	119			
4	0.9999997 \approx 1	117	129	0.9999997 \approx 1	117	129
5	0.9999999 \approx 1	129	123	0.9999999 \approx 1	129	123
6	0.99983	93	107	0.99983	93	107
7	0.99999	120	124			
8	0.9999999 \approx 1	129	123			
9	0.9999997 \approx 1	117	129			
10	0.99999	120	124			
11	0.99999	117	117	0.99999	117	117
12	0.99985	84	120	0.99985	84	120
13	0.99997	96	114	0.99997	96	114
14	0.99997	96	126			
15	0.99997	96	126			
16	0.99773	75	95			
17	\approx 1	120	136			
18	0.9979	99	105	0.9979	99	105
19	0.99984	105	113	0.99984	105	113
20	0.99998	108	120	0.99998	108	120
21	0.99983	93	107			
22	0.99998	108	120			
23	0.99999	117	117			
24	0.99984	105	113			
25	0.99998	108	120			
26	0.99788	78	102	0.99788	78	102
27	\approx 1	132	130			
28	\approx 1	132	130			
29	0.99983	93	119			
30	0.9979	99	105			
31	\approx 1	128	142			
32	0.99985	84	120			
33	0.99984	105	113			
34	0.90415	63	89	0.90415	63	89
35	0.99997	96	114			
36	0.99788	78	102			
37	0.99999	120	124			
38	\approx 1	128	142			
39	0.99983	93	107			
40	0.99971	81	113			
41	0.99983	93	107			
42	0.9999999 \approx 1	129	123			
43	\approx 1	128	142			
44	0.99773	75	95			
45	\approx 1	120	136			
46	0.99999	120	124			
47	0.99984	105	113			
48	0.99999	117	117			
49	0.99773	75	95			
50	0.99788	78	102			
Non-duplicated nondominated solutions				13		

Table A.3 Nondominated solutions in last Pareto front. Run 3, NSGA-II

Solution number	Original output			Non-duplicated nondominated solutions		
	Reliability	Cost	Weight	Reliability	Cost	Weight
1	0.99991	104	100	0.99991	104	100
2	0.99997	106	108	0.99997	106	108
3	0.9989	88	92	0.9989	88	92
4	0.99995	102	106	0.99995	102	106
5	0.99359	79	71	0.99359	79	71
6	0.99986	96	108	0.99986	96	108
7	0.99986	96	108			
8	0.99991	104	100			
9	0.99989	100	98	0.99989	100	98
10	0.99896	90	100	0.99896	90	100
11	0.99995	102	106			
12	0.99991	104	100			
13	0.99997	106	108			
14	0.9946	97	85	0.9946	97	85
15	0.99896	90	100			
16	0.99997	106	108			
17	0.9998	94	100	0.9998	94	100
18	0.9998	94	100			
19	0.99986	96	108			
20	0.99896	90	100			
21	0.99997	106	108			
22	0.99989	100	98			
23	0.9989	88	92			
24	0.99989	100	98			
25	0.9998	94	100			
26	0.99995	102	106			
27	0.9946	97	85			
28	0.99359	79	71			
29	0.99995	102	106			
30	0.9989	88	92			
31	0.99997	106	108			
32	0.9989	88	92			
33	0.99997	108	114			
34	0.9989	88	92			
35	0.99989	100	98			
36	0.99995	102	106			
37	0.9998	94	100			
38	0.99989	100	98			
39	0.9946	97	85			
40	0.99997	106	108			
41	0.99989	100	98			
42	0.99995	102	106			
43	0.99986	96	108			
44	0.9989	88	92			
45	0.99896	90	100			
46	0.9946	97	85			
47	0.99458	91	77	0.99458	91	77
48	0.99896	90	100			
49	0.99991	104	100			
50	0.99997	108	114			
Non-duplicated nondominated solutions				11		

Table A.4 Nondominated solutions in last Pareto front. Run 4, NSGA-II

Solution number	Original output			Non-duplicated nondominated solutions		
	Reliability	Cost	Weight	Reliability	Cost	Weight
1	0.85988	59	98	0.85988	59	98
2	0.99973	83	108	0.99973	83	108
3	0.99952	69	113	0.99952	69	113
4	0.99985	89	120	0.99985	89	120
5	0.99973	83	108			
6	0.99962	73	115	0.99962	73	115
7	0.99446	69	112	0.99446	69	112
8	0.85996	63	100	0.85996	63	100
9	0.99962	73	115			
10	0.99962	73	115			
11	0.99973	83	108			
12	0.99952	69	113			
13	0.85996	63	100			
14	0.99436	65	110	0.99436	65	110
15	0.99983	87	110	0.99983	87	110
16	0.85988	59	98			
17	0.99985	89	120			
18	0.99973	83	108			
19	0.85996	63	100			
20	0.99983	87	110			
21	0.99962	73	115			
22	0.99985	89	120			
23	0.99995	93	122	0.99995	93	122
24	0.99973	83	108			
25	0.99446	69	112			
26	0.99995	93	122			
27	0.99952	69	113			
28	0.99973	83	108			
29	0.99983	87	110			
30	0.99995	93	122			
31	0.99962	73	115			
32	0.99436	65	110			
33	0.99985	89	120			
34	0.99446	69	112			
35	0.85988	59	98			
36	0.99983	87	110			
37	0.99962	73	115			
38	0.85996	63	100			
39	0.85996	63	100			
40	0.85996	63	100			
41	0.99995	93	122			
42	0.99983	87	110			
43	0.99436	65	110			
44	0.99446	69	112			
45	0.99436	65	110			
46	0.99962	73	115			
47	0.99962	73	115			
48	0.85988	59	98			
49	0.85996	63	100			
50	0.99952	69	113			
Non-duplicated nondominated solutions				10		

Table A.5 Nondominated solutions in last Pareto front. Run 5, NSGA-II

	Original output			Non-duplicated nondominated solutions		
Solution number	Reliability	Cost	Weight	Reliability	Cost	Weight
1	0.96945	67	88	0.96945	67	88
2	0.99141	71	90	0.99141	71	90
3	0.96949	69	91	0.96949	69	91
4	0.99141	71	90			
5	0.99141	71	90			
6	0.96945	67	88			
7	0.99141	71	90			
8	0.99141	71	90			
9	0.96945	67	88			
10	0.99141	71	90			
11	0.99141	71	90			
12	0.96949	69	91			
13	0.96949	69	91			
14	0.99141	71	90			
15	0.99145	73	93	0.99145	73	93
16	0.99141	71	90			
17	0.99145	73	93			
18	0.99141	71	90			
19	0.96949	69	91			
20	0.99145	73	93			
21	0.99141	71	90			
22	0.96945	67	88			
23	0.99145	73	93			
24	0.99141	71	90			
25	0.96949	69	91			
26	0.96949	69	91			
27	0.99145	73	93			
28	0.99141	71	90			
29	0.99145	73	93			
30	0.99145	73	93			
31	0.99141	71	90			
32	0.96949	69	91			
33	0.99145	73	93			
34	0.99145	73	93			
35	0.99145	73	93			
36	0.96949	69	91			
37	0.99141	71	90			
38	0.99145	73	93			
39	0.99141	71	90			
40	0.96945	67	88			
41	0.99145	73	93			
42	0.96949	69	91			
43	0.96949	69	91			
44	0.96949	69	91			
45	0.96949	69	91			
46	0.99145	73	93			
47	0.99145	73	93			
48	0.96945	67	88			
49	0.99141	71	90			
50	0.96949	69	91			
Non-duplicated nondominated solutions				4		

Table A.6 Nondominated solutions in last Pareto front. Run 6, NSGA-II

Solution number	Original output			Non-duplicated nondominated solutions		
	Reliability	Cost	Weight	Reliability	Cost	Weight
1	0.99082	63	82	0.99082	63	82
2	0.99996	95	100	0.99996	95	100
3	0.91581	53	68	0.91581	53	68
4	0.9999	93	92	0.9999	93	92
5	0.99998	101	106			
6	0.99998	99	98	0.99998	99	98
7	0.99989	96	91	0.99989	96	91
8	0.99996	95	100			
9	0.99082	63	82			
10	0.99981	90	85	0.99981	90	85
11	0.99998	99	98			
12	0.99989	96	91			
13	0.91581	53	68			
14	0.9908	59	84	0.9908	59	84
15	0.99065	54	69	0.99065	54	69
16	0.99074	57	76	0.99074	57	76
17	0.91581	53	68			
18	0.99082	65	90			
19	0.99082	63	82			
20	0.9999	93	92			
21	0.9999	93	92			
22	0.9158	56	67	0.9158	56	67
23	0.99998	105	114			
24	0.99073	60	75	0.99073	60	75
25	0.9908	59	84			
26	0.99065	54	69			
27	0.99074	57	76			
28	0.99082	65	90			
29	0.99073	60	75			
30	0.99989	96	91			
31	0.99074	57	76			
32	0.99989	96	91			
33	0.99996	95	100			
34	0.99998	101	106			
35	0.99981	90	85			
36	0.9999	93	92			
37	0.99998	101	106			
38	0.99065	54	69			
39	0.99996	95	100			
40	0.91581	53	68			
41	0.91573	50	61	0.91573	50	61
42	0.99996	95	100			
43	0.9158	56	67			
44	0.99981	90	85			
45	0.99998	99	98			
46	0.99998	101	106			
47	0.9908	59	84			
48	0.99998	101	106			
49	0.99082	65	90			
50	0.9908	59	84			
Non-duplicated nondominated solutions				13		

Table A.7 Nondominated solutions in last Pareto front. Run 7, NSGA-II

Solution number	Original output			Non-duplicated nondominated solutions		
	Reliability	Cost	Weight	Reliability	Cost	Weight
1	0.9732	79	85	0.9732	79	85
2	0.99222	82	89	0.99222	82	89
3	0.99223	85	96	0.99223	85	96
4	0.99223	85	96			
5	0.99222	82	89			
6	0.99223	85	96			
7	0.99223	85	96			
8	0.99789	87	89	0.99789	87	89
9	0.99938	90	93	0.99938	90	93
10	0.99222	82	89			
11	0.99938	90	93			
12	0.99223	85	96			
13	0.99789	87	89			
14	0.99789	87	89			
15	0.99938	90	93			
16	0.99222	82	89			
17	0.99223	85	96			
18	0.99789	87	89			
19	0.99222	82	89			
20	0.99939	93	100	0.99939	93	100
21	0.99222	82	89			
22	0.99938	90	93			
23	0.99222	82	89			
24	0.99939	93	100			
25	0.99222	82	89			
26	0.99223	85	96			
27	0.99938	90	93			
28	0.99222	82	89			
29	0.99939	93	100			
30	0.99938	90	93			
31	0.99222	82	89			
32	0.99939	93	100			
33	0.99223	85	96			
34	0.99938	90	93			
35	0.9732	79	85			
36	0.99789	87	89			
37	0.99939	93	100			
38	0.99222	82	89			
39	0.9732	79	85			
40	0.99223	85	96			
41	0.99222	82	89			
42	0.99223	85	96			
43	0.99789	87	89			
44	0.99939	93	100			
45	0.99222	82	89			
46	0.99938	90	93			
47	0.99939	93	100			
48	0.9732	79	85			
49	0.99938	90	93			
50	0.99939	93	100			
Non-duplicated nondominated solutions				6		

Table A.8 Nondominated solutions in last Pareto front. Run 8, NSGA-II

Solution number	Original output			Non-duplicated nondominated solutions		
	Reliability	Cost	Weight	Reliability	Cost	Weight
1	0.71265	49	41	0.71265	49	41
2	0.9988	77	89	0.9988	77	89
3	0.98871	61	69	0.98871	61	69
4	0.9988	77	89			
5	0.9988	77	89			
6	0.99654	65	61	0.99654	65	61
7	0.98871	61	69			
8	0.71989	57	57	0.71989	57	57
9	0.71914	53	49	0.71914	53	49
10	0.98754	61	53	0.98754	61	53
11	0.99772	65	77	0.99772	65	77
12	0.99876	69	85	0.99876	69	85
13	0.98754	61	53			
14	0.71989	57	57			
15	0.71989	57	57			
16	0.99876	69	85			
17	0.99881	89	77	0.99881	89	77
18	0.99876	69	85			
19	0.99758	69	69	0.99758	69	69
20	0.99876	69	85			
21	0.99772	65	77			
22	0.98871	61	69			
23	0.99772	65	77			
24	0.99758	69	69			
25	0.71914	53	49			
26	0.71989	57	57			
27	0.99654	65	61			
28	0.71265	49	41			
29	0.99772	65	77			
30	0.71914	53	49			
31	0.99772	65	77			
32	0.71914	53	49			
33	0.99984	81	97	0.99984	81	97
34	0.99984	81	97			
35	0.9988	77	89			
36	0.71265	49	41			
37	0.99772	65	77			
38	0.99984	81	97			
39	0.99758	69	69			
40	0.99876	69	85			
41	0.98871	61	69			
42	0.99772	65	77			
43	0.99654	65	61			
44	0.71914	53	49			
45	0.9988	77	89			
46	0.98871	61	69			
47	0.99876	69	85			
48	0.99984	81	97			
49	0.99881	89	77			
50	0.9988	77	89			
Non-duplicated nondominated solutions				12		

Table A.9 Nondominated solutions in last Pareto front. Run 9, NSGA-II

Solution number	Original output			Non-duplicated nondominated solutions		
	Reliability	Cost	Weight	Reliability	Cost	Weight
1	0.99875	82	95	0.99875	82	95
2	0.99984	106	119			
3	0.99984	106	119			
4	0.99984	106	119			
5	0.99984	106	119			
6	0.99984	106	119			
7	0.99978	91	104	0.99978	91	104
8	0.99984	106	119			
9	0.99978	91	104			
10	0.99978	91	104			
11	0.99984	106	119			
12	0.99984	106	119			
13	0.99875	82	95			
14	0.99984	106	119			
15	0.99984	106	119			
16	0.99984	106	119			
17	0.99984	106	119			
18	0.99978	91	104			
19	0.99984	106	111	0.99984	106	111
20	0.99984	106	111			
21	0.99984	106	111			
22	0.99984	106	119			
23	0.99984	106	111			
24	0.99984	106	111			
25	0.99978	91	104			
26	0.99984	106	119			
27	0.99984	106	111			
28	0.99984	106	111			
29	0.99984	115	120			
30	0.99984	115	120			
31	0.99984	106	119			
32	0.99978	91	104			
33	0.99875	82	95			
34	0.99875	82	95			
35	0.99984	115	120			
36	0.99978	91	104			
37	0.99984	106	111			
38	0.99875	82	95			
39	0.99978	91	104			
40	0.99984	115	120			
41	0.99984	106	119			
42	0.99978	91	104			
43	0.99984	106	111			
44	0.99984	106	119			
45	0.99978	91	104			
46	0.99984	115	120			
47	0.99875	82	95			
48	0.99984	115	120			
49	0.99984	106	119			
50	0.99978	91	104			
Non-duplicated nondominated solutions				3		

Table A.10 Nondominated solutions in last Pareto front. Run 10, NSGA-II

Solution number	Original output			Non-duplicated nondominated solutions		
	Reliability	Cost	Weight	Reliability	Cost	Weight
1	0.99513	64	60	0.99513	64	60
2	0.99789	81	88			
3	0.99513	78	68			
4	0.99789	82	72	0.99789	82	72
5	0.99789	94	77			
6	0.99513	76	65			
7	0.9999	87	90	0.9999	87	90
8	0.99785	73	81	0.99785	73	81
9	0.99789	84	75			
10	0.99782	73	71	0.99782	73	71
11	0.99997	96	91	0.99997	96	91
12	0.99993	87	100			
13	0.99789	82	72			
14	0.99782	73	71			
15	0.99789	79	85	0.99789	79	85
16	0.99513	66	63			
17	0.99997	93	104	0.99997	93	104
18	0.99786	75	84	0.99786	75	84
19	0.99997	93	104			
20	0.99997	106	93			
21	0.99782	75	74			
22	0.99785	73	81			
23	0.99789	94	77			
24	0.99513	76	65			
25	0.99789	81	88			
26	0.9976	82	69	0.9976	82	69
27	0.99789	96	80			
28	0.99996	91	101	0.99996	91	101
29	0.99513	76	65			
30	0.99782	75	74			
31	0.99997	106	93			
32	0.99997	106	93			
33	0.99997	108	96			
34	0.99513	78	68			
35	0.99997	106	93			
36	0.99513	66	63			
37	0.99996	91	101			
38	0.99993	85	97	0.99993	85	97
39	0.99786	75	84			
40	0.99789	79	85			
41	0.9999	87	90			
42	0.9976	82	69			
43	0.99789	84	75			
44	0.99513	78	68			
45	0.99993	87	100			
46	0.99993	85	97			
47	0.99997	93	104			
48	0.99789	96	80			
49	0.99789	84	75			
50	0.99996	91	101			
Non-duplicated nondominated solutions				12		

Table A.11 Nondominated solutions in final
Pareto front. Run 1, MOEA-DAP

Solution number	Original output = Non- duplicated nondominated solutions		
	Reliability	Cost	Weight
1	0.99981	77	68
2	0.90598	48	32
3	0.96026	40	37
4	0.87971	36	33
5	0.99819	74	54
6	0.99894	78	48
7	0.87959	36	27
8	0.99799	74	52
9	0.95929	42	35
10	0.99906	73	74
11	0.9992	73	80
12	0.80704	36	20
13	0.78353	24	15
14	0.98734	52	37
15	0.99762	61	64
16	0.99675	62	44
17	0.973	38	41
18	0.99177	50	39
19	0.98313	39	39
20	0.99885	70	80
21	0.99429	45	47
22	0.94609	27	45
23	0.98342	44	34
24	0.9516	30	34
25	0.99514	56	50
26	0.87106	34	21
27	0.97596	50	26
Non- duplicated nondominated solutions	27		

Table A.12 Nondominated solutions in final
Pareto front. Run 2, MOEA-DAP

Solution number	Original output = Non- duplicated nondominated solutions		
	Reliability	Cost	Weight
1	0.78353	24	15
2	0.99642	62	36
3	0.98734	52	37
4	0.87106	34	21
5	0.97745	39	40
6	0.99799	74	52
7	0.99762	61	64
8	0.99006	51	68
9	0.97596	50	26
10	0.98714	52	35
11	0.984	45	64
12	0.98683	56	32
13	0.97967	42	39
14	0.95821	37	36
15	0.78499	24	21
16	0.99728	61	56
17	0.98142	46	39
18	0.87771	33	26
19	0.99064	56	40
20	0.87878	37	28
21	0.87134	30	25
22	0.95809	44	27
23	0.79952	26	19
24	0.98112	50	36
25	0.9877	55	52
26	0.98802	55	60
27	0.94753	38	21
28	0.76929	22	17
Non- duplicated nondominated solutions	28		

Table A.13 Nondominated solutions in final Pareto front. Run 3 MOEA-DAP

Solution number	Original output = Non-duplicated nondominated solutions		
	Reliability	Cost	Weight
1	0.97843	46	31
2	0.63179	19	25
3	0.9516	30	34
4	0.99514	56	50
5	0.97774	41	45
6	0.95826	31	54
7	0.97326	36	38
8	0.98549	48	34
9	0.96763	33	49
10	0.98559	47	36
11	0.97564	41	36
12	0.9949	46	55
13	0.94609	27	45
14	0.97829	43	43
15	0.98769	52	44
16	0.80586	30	26
17	0.98921	54	46
18	0.80541	28	28
19	0.8313	24	34
20	0.98342	44	34
21	0.96043	37	30
22	0.99305	52	50
23	0.98577	40	47
24	0.78749	22	24
25	0.96311	39	34
26	0.79773	24	26
27	0.97381	38	36
28	0.97531	40	38
29	0.96383	34	43
30	0.91857	27	29
31	0.96153	38	30
32	0.94001	35	31
33	0.95597	35	41
34	0.9871	49	38
35	0.98397	46	32
36	0.98227	42	46
Non-duplicated nondominated solutions	36		

Table A.14 Nondominated solutions in final Pareto front. Run 4, MOEA-DAP

Solution number	Original output = Non-duplicated nondominated solutions		
	Reliability	Cost	Weight
1	0.89929	24	42
2	0.97632	34	52
3	0.94059	29	40
4	0.88833	23	42
5	0.874	22	38
6	0.87444	30	30
7	0.99182	50	46
8	0.9863	42	50
9	0.98556	42	46
10	0.99673	53	70
11	0.9444	34	33
12	0.97815	44	33
13	0.98172	39	55
14	0.78175	17	26
15	0.86837	25	32
16	0.9773	38	44
17	0.98847	44	49
18	0.94021	28	42
19	0.99664	50	63
20	0.98962	42	59
21	0.86747	27	26
22	0.97955	39	41
23	0.86335	21	38
24	0.98681	49	45
25	0.78339	18	20
26	0.97275	34	43
27	0.99549	52	53
28	0.98921	44	53
29	0.98571	44	48
30	0.80437	27	22
31	0.9678	31	44
32	0.97134	35	39
33	0.79623	19	24
34	0.99045	45	70
35	0.98275	37	61
36	0.99438	47	56
Non-duplicated nondominated solutions	36		

Table A.15 Nondominated solutions in final
Pareto front. Run 5 MOEA-DAP

Solution number	Original output = Non- duplicated nondominated solutions		
	Reliability	Cost	Weight
1	0.94977	28	36
2	0.99177	50	39
3	0.98313	39	39
4	0.99885	70	80
5	0.99429	45	47
6	0.69149	44	29
7	0.99957	80	51
8	0.98084	44	33
9	0.99954	73	65
10	0.99874	73	51
11	0.99538	64	45
12	0.95132	29	39
13	0.99975	82	55
14	0.99594	66	45
15	0.96192	35	45
16	0.99409	45	45
17	0.9961	58	48
18	0.98398	54	33
19	0.99484	47	49
20	0.99771	59	51
21	0.99858	74	45
22	0.99879	71	61
23	0.68897	30	23
24	0.99532	62	45
25	0.98154	38	36
26	0.99495	60	39
27	0.99846	61	55
28	0.96211	35	47
Non- duplicated nondominated solutions	28		

Table A.16 Nondominated solutions in final
Pareto front. Run 6, MOEA-DAP

Solution number	Original output = Non- duplicated nondominated solutions		
	Reliability	Cost	Weight
1	0.98046	44	44
2	0.98097	32	62
3	0.99662	52	64
4	0.98319	38	50
5	0.65037	24	38
6	0.9675	46	36
7	0.99664	51	66
8	0.98875	59	36
9	0.99692	64	46
10	0.98451	46	44
11	0.99657	50	66
12	0.98502	34	62
13	0.99802	65	50
14	0.9919	56	46
15	0.95698	38	35
16	0.64572	22	30
17	0.64916	28	26
18	0.93434	28	35
19	0.85736	30	31
20	0.97798	32	54
21	0.98976	52	40
22	0.97043	28	38
23	0.98926	47	54
24	0.93214	29	32
25	0.99853	53	68
26	0.65316	37	30
27	0.9862	38	58
28	0.64769	22	38
29	0.98269	50	32
Non- duplicated nondominated solutions	29		

Table A.17 Nondominated solutions in final Pareto front. Run 7 MOEA-DAP

	Original output = Non-duplicated nondominated solutions		
Solution number	Reliability	Cost	Weight
1	0.88673	23	28
2	0.86259	23	25
3	0.96315	50	34
4	0.96232	34	47
5	0.93853	22	46
6	0.94972	26	57
7	0.92266	28	33
8	0.91428	22	31
9	0.93422	35	28
10	0.90531	24	27
11	0.93887	27	46
12	0.75874	17	23
13	0.87114	21	29
14	0.85578	26	21
15	0.95947	32	39
16	0.61152	18	15
17	0.6128	17	22
18	0.96317	51	32
19	0.9892	48	43
20	0.85843	20	25
21	0.94259	37	37
22	0.96613	44	38
23	0.6513	20	21
24	0.70056	19	22
25	0.63183	16	25
Non-duplicated nondominated solutions	25		

Table A.18 Nondominated solutions in final Pareto front. Run 8, MOEA-DAP

	Original output = Non-duplicated nondominated solutions		
Solution number	Reliability	Cost	Weight
1	0.93806	32	37
2	0.85561	21	29
3	0.97708	42	35
4	0.8745	27	31
5	0.9827	40	39
6	0.95945	36	36
7	0.95482	35	32
8	0.90864	35	26
9	0.87176	33	22
10	0.77474	20	24
11	0.97859	61	35
12	0.70539	17	15
13	0.98849	68	37
14	0.85798	22	28
15	0.99076	46	47
16	0.97728	36	42
17	0.96048	30	40
18	0.78473	23	21
19	0.93354	31	33
20	0.99007	43	46
21	0.97286	33	37
22	0.9959	71	44
23	0.97893	43	37
24	0.89276	31	25
25	0.92876	31	31
26	0.88819	31	23
27	0.90426	34	22
28	0.92739	32	28
29	0.81678	29	20
30	0.89029	30	26
31	0.98016	36	44
32	0.90711	28	31
33	0.75652	22	21
34	0.8154	22	25
35	0.98304	64	40
36	0.81013	20	32
37	0.86904	25	25
Non-duplicated nondominated solutions	37		

Table A.19 Nondominated solutions in final
Pareto front. Run 9 MOEA-DAP

	Original output = Non-duplicated nondominated solutions		
Solution number	Reliability	Cost	Weight
1	0.9629	40	41
2	0.98279	42	41
3	0.99273	56	45
4	0.98746	41	62
5	0.9217	26	59
6	0.92274	27	57
7	0.96132	39	37
8	0.98907	42	66
9	0.90753	32	33
10	0.99582	54	62
11	0.86698	21	32
12	0.96181	39	43
13	0.88734	25	34
14	0.9233	34	31
15	0.91288	35	29
16	0.99745	55	66
17	0.93571	44	35
18	0.76929	22	17
19	0.97843	46	31
20	0.98397	46	32
21	0.97632	34	52
22	0.94977	28	36
23	0.99177	50	39
24	0.98319	38	50
25	0.9862	38	58
26	0.86259	23	25
Non-duplicated nondominated solutions	26		

Table A.20 Nondominated solutions in final
Pareto front. Run 10, MOEA-DAP

	Original output = Non-duplicated nondominated solutions		
Solution number	Reliability	Cost	Weight
1	0.95894	42	32
2	0.73478	19	17
3	0.48973	10	22
4	0.92701	33	34
5	0.33768	6	15
6	0.92441	29	36
7	0.51282	11	15
8	0.59808	18	13
9	0.95625	38	34
10	0.54343	12	15
11	0.84671	30	18
12	0.89135	23	28
13	0.41741	10	11
14	0.75977	15	22
15	0.60456	14	23
16	0.44233	11	11
17	0.76418	29	23
18	0.76203	25	25
19	0.46214	9	22
20	0.85126	31	26
21	0.62621	23	16
Non-duplicated nondominated solutions	21		

Table A.21. True Pareto-optimal front, Y_{true} .

Solution number	Reliability	Cost	Weight	Solution number	Reliability	Cost	Weight
1	0.9999	87	90	71	0.9961	58	48
2	0.99997	96	91	72	0.98398	54	33
3	0.99997	93	104	73	0.99484	47	49
4	0.99996	91	101	74	0.99771	59	51
5	0.99993	85	97	75	0.99858	74	45
6	0.9999997 ≈ 1	117	129	76	0.99879	71	61
7	0.9999999 ≈ 1	129	123	77	0.98154	38	36
8	0.99999	117	117	78	0.99495	60	39
9	0.99985	84	120	79	0.99846	61	55
10	0.99952	69	113	80	0.98097	32	62
11	0.99962	73	115	81	0.98319	38	50
12	0.99996	95	100	82	0.98875	59	36
13	0.99998	99	98	83	0.99692	64	46
14	0.99876	69	85	84	0.98451	46	44
15	0.99984	81	97	85	0.98502	34	62
16	0.99981	77	68	86	0.99802	65	50
17	0.99894	78	48	87	0.93434	28	35
18	0.78353	24	15	88	0.97798	32	54
19	0.98734	52	37	89	0.97043	28	38
20	0.99675	62	44	90	0.93214	29	32
21	0.99177	50	39	91	0.99853	53	68
22	0.98313	39	39	92	0.9862	38	58
23	0.99885	70	80	93	0.86259	23	25
24	0.99429	45	47	94	0.93853	22	46
25	0.94609	27	45	95	0.94972	26	57
26	0.98342	44	34	96	0.92266	28	33
27	0.9516	30	34	97	0.91428	22	31
28	0.99514	56	50	98	0.93422	35	28
29	0.87106	34	21	99	0.90531	24	27
30	0.97596	50	26	100	0.87114	21	29
31	0.99642	62	36	101	0.85578	26	21
32	0.98714	52	35	102	0.9892	48	43
33	0.98683	56	32	103	0.85843	20	25
34	0.78499	24	21	104	0.97708	42	35
35	0.87134	30	25	105	0.95945	36	36
36	0.95809	44	27	106	0.95482	35	32
37	0.79952	26	19	107	0.90864	35	26
38	0.94753	38	21	108	0.87176	33	22
39	0.76929	22	17	109	0.70539	17	15
40	0.97843	46	31	110	0.97728	36	42
41	0.97326	36	38	111	0.78473	23	21
42	0.98549	48	34	112	0.93354	31	33
43	0.98559	47	36	113	0.99007	43	46
44	0.9949	46	55	114	0.97286	33	37
45	0.96043	37	30	115	0.89276	31	25
46	0.98577	40	47	116	0.92876	31	31
47	0.96311	39	34	117	0.88819	31	23
48	0.91857	27	29	118	0.90426	34	22
49	0.96153	38	30	119	0.92739	32	28
50	0.94001	35	31	120	0.81678	29	20
51	0.9871	49	38	121	0.89029	30	26
52	0.98397	46	32	122	0.98016	36	44
53	0.97632	34	52	123	0.86904	25	25
54	0.9863	42	50	124	0.98746	41	62
55	0.98556	42	46	125	0.99582	54	62
56	0.9444	34	33	126	0.99745	55	66
57	0.78175	17	26	127	0.73478	19	17
58	0.99664	50	63	128	0.48973	10	22
59	0.98962	42	59	129	0.33768	6	15
60	0.78339	18	20	130	0.51282	11	15
61	0.99549	52	53	131	0.59808	18	13
62	0.79623	19	24	132	0.54343	12	15
63	0.98275	37	61	133	0.84671	30	18
64	0.94977	28	36	134	0.89135	23	28
65	0.99957	80	51	135	0.41741	10	11
66	0.98084	44	33	136	0.75977	15	22
67	0.99954	73	65	137	0.60456	14	23
68	0.99874	73	51	138	0.44233	11	11
69	0.99975	82	55	139	0.46214	9	22
70	0.99409	45	45				

References

1. Altumi, A. A., Philipose, A. M. and Taboun, S. M. (2001). Reliability Optimisation of Flexible Manufacturing Systems with Spare Tooling. *International Journal of Advanced Manufacturing Technology*, 17:69-77.
2. Arroyo, J. E. C. and Armentano, V.A. (2005). Genetic local search for multi-objective flowshop scheduling problems. *European Journal of Operational Research*, 167, 717–738.
3. Artzner, P., Delbaen, F., Eber, J.-M., and Heath, D. (1999). Coherent measures of risk. *Mathematical Finance*, 9, 203-228.
4. Bäck, T. and Hoffmeister, F. (1991). Extended Selection Mechanisms in Genetic Algorithms. *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Mateo, California, USA: Morgan Kaufmann Publishers
5. Bäck, T. (1993). Optimal Mutation Rates in Genetic Search. *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, California, USA: Morgan Kaufmann Publishers.
6. Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice - Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. New York, Oxford: Oxford University Press.
7. Baker, J. E. (1987). Reducing Bias and Inefficiency in the Selection Algorithm. *In Proceedings of the Second International Conference on Genetic Algorithms and their Application*, Hillsdale, New Jersey, USA: Lawrence Erlbaum Associates.
8. Barlow, R. E. and Wu, A. S. (1978) Coherent systems with multi-state components. *Mathematics of Operations Research*, 3:275-281.
9. Bellman, R. E. (1957). *Dynamic Programming*. Princeton University Press.
10. Bellman, R. E. and Dreyfus, E. (1958). Dynamic programming and reliability of multicomponent devices. *Operations Research*, 6, 200-206.
11. Birnbaum, Z. W., Esary, J.D. and Saunders, S.C. (1961) Multi-component systems and structures and their reliability. *Techno-metrics*, 3:55-77.
12. Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization: overview and conceptual comparison. *ACM Computing Surveys*, 35(3): 268-308.
13. Booker, L. (1987). *Improving search in genetic algorithms*. Genetic Algorithms and Simulated Annealing. San Mateo, California, USA: Morgan Kaufmann Publishers.

14. Bramlette, M. F. (1991). Initialization, mutation and selection methods in genetic algorithms for function optimization. In *Proceedings of the 4th international conference on Genetic algorithms*.
15. Branke, J., Deb, K., Dierolf, H. and Osswald, M. (2004). Finding Knees in Multi-objective Optimization. In the Eighth Conference on Parallel Problem Solving from Nature (PPSN VIII). Lecture Notes in Computer Science. pp. 722-731.
16. Brunelle R. D. and Kapur K. C. (1998). Continuous-State System-Reliability: An Interpolation Approach. *IEEE Transactions on Reliability*, 47(2):181-187.
17. Bulfin, R. L. and Liu, C. Y. (1985). Optimal allocation of redundant components for large systems. *IEEE Transactions on Reliability*, 34, 241-247.
18. Busacca, P.G., Marseguerra, M. and Zio, E. (2001). Multiobjective optimization by genetic algorithms: application to safety systems. *Reliability Engineering and System Safety*, 72, 59-74.
19. Cerny, V. (1985). A thermodynamical approach to the travelling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Application*, 45, 41-51.
20. Chern, M. S. (1992). On the computational complexity of reliability redundancy allocation in a series system, *Operations Research Letters*, 11, 309-315.
21. Chankong, V. and Haimes, Y. (1983). Multiobjective decision making theory and methodology. New York: North-Holland.
22. Cochran, D. S., Arinez, J. F., Duda, J. W. and Linck, J. (2001). A Decomposition Approach for Manufacturing System Design. *Journal of Manufacturing Systems*, 20(6):371-389.
23. Coello Coello, C. A. (2000). Handling preferences in evolutionary multiobjective optimization: a survey. In *Proceedings of the 2000 Congress on Evolutionary Computation*, 1:30-37.
24. Coello Coello, C. A. and Toscano, P. G. (2001). A Micro-Genetic Algorithm for multiobjective optimization. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*: 274-282. San Francisco, California. Morgan Kaufmann Publishers.
25. Coello Coello, C. A., Van Veldhuizen, D. A. and Lamont, G. B. (2002). Evolutionary Algorithms for solving Multiobjective Problems. Kluwer Academics Publishers.
26. Coello Coello, C. A. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245-1287.

27. Coit, D. W. and Smith A. (1996a). Reliability optimization of series-parallel systems using a genetic algorithm. *IEEE Transactions on Reliability*, 45(2): 254-260.
28. Coit, D. W. and Smith A. (1996b). Penalty guided genetic search for reliability design optimization. *Computers and Industrial Engineering*, 30(4): 895-904.
29. Cox D. (1957): Note on grouping. *Journal of the American Statistical Association*, **52**, 543–547.
30. Cvetkovic, D. and Parmee, I. C. (2000). Preferences and their Application in Evolutionary Multiobjective Optimisation. *IEEE Transactions on Evolutionary Computation*, 6(1):42-57.
31. Das, I. (1999). On characterizing the ‘knee’ of the Pareto curve based on normal-boundary intersection. *Structural Optimization*, 18(2/3):107-115.
32. De Jong, K. A. (1975). An analysis of the behaviour of a class of genetic adaptive systems. PhD thesis, University of Michigan. Diss. Abstr. Int. 36(10), 5140B, University Microfilms No. 76-9381.
33. Deb, K. and Goldberg, D. E. (1989). An investigation of niche and species formation in genetic function. In *Proceedings of 3rd International Conference on Genetic Algorithms*: 42–50.
34. Deb, K., Agarwal, S., Pratap, A. and Meyarivan, T. (2000a). A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. KanGAL Report Number 200001, Indian Institute of Technology. Kanpur, India.
35. Deb, K., Agarwal, S., Pratap, A. and Meyarivan, T. (2000b). A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. *Proceedings of the Parallel Problem Solving from Nature VI Conference*, 849-858. Paris, France.
36. Deb, K. Multi-objective optimization using evolutionary algorithms. (2002). John Wiley & Sons. Ltd., Chichester, England.
37. Deb, K., Agrawal, S., Pratap, A. and Meyarivan, T. (2002). A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182-197.
38. Dhingra, Anoop K. (1992). Optimal apportionment of reliability and redundancy in series systems under multiple objectives. *IEEE Transactions on Reliability*, 41(4), 576-582.

39. Dorigo, M. (1992). Optimization, learning and natural algorithms (*in italian*). Ph.D. thesis, DEI, Politecnico di Milano, Italy.
40. Dorigo, M., Di Caro, G., and Gambardella, L. M. (1999). Ant algorithms for discrete optimization. *Artificial Life*, 5(2), 137–172.
41. Dorigo, M. and Gambardella, L. M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation* 1, 1 (Apr.), 53–66.
42. Dorigo, M., Maniezzo, V., and Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics—Part B* 26, 1, 29–41.
43. Edelson, W. and Gargano, M. L. (2000). Feasible encodings for GA solutions of constrained minimal spanning tree problems. *Proceedings of GECCO- 2000*, Las Vegas, Nevada.
44. Edgeworth, F. Y. (1881). *Mathematical Physics*. London, U.K. P.Keagan.
45. El-Newehi, E., Proschan, F. and Sethuraman, J. (1978) Multi- state coherent systems. *Journal of Applied Probability*, 15: 675-688.
46. Erickson, M., Mayer, A. and Horn, J. (2001). The Niche Pareto Genetic Algorithm 2 Applied to the Design of Groundwater Remediation Systems. *First International Conference on Evolutionary Multi-Criterion Optimization*, 681-695.
47. Fogel, L. J. (1962). Toward inductive inference automata. *Proceedings of the International Federation for Information Processing Congress*. Munich, 395–399.
48. Fogel, L. J., Owens, A. J., and Walsh, M. J. (1966). *Artificial Intelligence through Simulated Evolution*. Wiley, New York.
49. Fogel, D. B. and Ghoseil, A. (1997). A note on representations and variation operators. *IEEE Transactions on Evolutionary Computation*, 1(2):159-161.
50. Fonseca, C. M. and Fleming, P. J. (1993). Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. *Proceedings of the Fifth International Conference on Genetic Algorithms*, 416-423. San Mateo, California.
51. Freiheit, T., Shpitalni, M. and Hu, S. J. (2004). Productivity of paced parallel-serial manufacturing lines with and without crossover. *Journal of Manufacturing Science and Engineering*, 126:361-367.

52. Frey, R. and McNeil, A. (2002). VaR and expected shortfall in portfolios of dependent credit risks: conceptual and practical insights. *Journal of Banking and Finance*, 1317-1334.
53. Fyffe, D. E., Hines, W. W. and Lee, N. K. (1968). System reliability allocation and a computational algorithm, *IEEE Transactions on Reliability*, 17, 64-69.
54. Gen, M., Ida, K. and Lee, J. U. (1990). A computational algorithm for solving 0-1 goal programming with GUB structures and its application for optimization problems in system reliability. *Electronics and Communications in Japan, Part 3*, 73, 88-96.
55. Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.* 13, 533-549.
56. Glover, F. (1989). Tabu search Part I. *ORSA Journal on Computing*, : 190-206.
57. Glover, F. (1990). Tabu search Part II. *ORSA Journal on Computing*, 2: 4-32.
58. Glover, F. and Laguna, M. (1997). *Tabu Search*. Kluwer Academic Publishers.
59. Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.
60. Goldberg, D. and Lingle, R. (1985). Alleles, loci, and the traveling salesman problem. *Proceedings of the International Conference of Genetic Algorithms and their Applications*, 154-159.
61. Grefenstette, J. J. (1986). Optimization of Control Parameters for Genetic Algorithms. *IEEE Transactions on Systems, Man and Cybernetics* SMC-16(1), 122-128.
62. Gupta, R. and Agarwal, M. (2006). Penalty guided genetic search for redundancy optimization in multi-state series-parallel power system. *Journal of Combinatorial Optimization*, 12:257-277.
63. Hans, A. E. (1988). Multicriteria optimization for highly accurate systems. *Multicriteria Optimization in Engineering and Sciences*. W. Stadler (Ed.), Mathematical concepts and methods in science and engineering, 19: 309-352.
64. Hertz, A. and Klover, D. (2000). A Framework for the Description of Evolutionary Algorithms. *European Journal of Operational Research*, 126 (1): 1-12.
65. Hinloopen, E., Nijkamp, P. and Rietveld P. (2004). Integration of ordinal and cardinal information in multi-criteria ranking with imperfect compensation. *European journal of Operational Research*, 158(2): 317-338.

66. Holland J. (1975). *Adaptation in natural and artificial systems*. U. Michigan Press, MI.
67. Hoogeveen, Han. (2005). Multicriteria Scheduling. *European Journal of Operational Research*, 167, 592–623.
68. Horn, J., Nafpliotis, N. and Goldberg, D. E. (1994). A Niche Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, IEEE World Congress on Computational Intelligence, 1, 82-87, Piscataway, New Jersey. IEEE Service Center.
69. Horn, J. (1997). Multicriterion decision making. In Thomas Bäch, David Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*, volume 1, pages F1.9:1-F1.9:15. IOP. Publishing Ltd. And Oxford University Press.
70. Huang, H-Z., Qu, J. and Zuo, M. J. (2006). A New Method of System Reliability Multi-objective Optimization using Genetic Algorithms. *RAMS conference proceedings*.
71. Ida, K., Gen M. and Yokota, T. (1994). System reliability optimization with several failure modes by genetic algorithm. In *Proceedings of 16th International Conference on Computers and Industrial Engineering*, 349-352.
72. Inagaki T., Inoue, K. and Akashi, H. (1978). Interactive optimization of system reliability under multiple objectives. *IEEE Transactions on Reliability*, 27, 264-267.
73. Jain A. K. and Dubes R. C. (1988). *Algorithms for Clustering Data*. Englewood Cliffs: Prentice Hall.
74. Karp, R. M. (1972) Reducibility among combinatorial problems, in *Complexity of Computer Computations*, Miller, R.E. and Thatcher, J.W. (eds.), Plenum Press, New York. 85– 103.
75. Kaufman L. and Rousseeuw P. J. (1990). *Finding groups in data. An introduction to Cluster Analysis*. Wiley-Interscience
76. Keeney, R. L. and Raiffa H. (1976). *Decisions with Multiple Objectives: Preferences and Tradeoffs*. John Wiley & Sons.
77. Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 13 May 1983 220, 4598, 671–680.
78. Knowles, J. D. and Corne, D. W. (1999) The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimization. In *Proc. Congress on Evolutionary Computation*. vol. 1, pp. 98–105. Piscataway, NJ: IEEE Press.

79. Knowles, J. D. and Corne, D. W. (2000). Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2): 149-172.
80. Konak, A., Coit, D. and Smith, A. (2006). Multi-Objective Optimization Using Genetic Algorithms: A Tutorial. *Reliability Engineering & System Safety*, 91(9).
81. Koren Y., Hu S. J. and Weber T. (1998). Impact of Manufacturing System Configuration on Performance. *Annals of the CIRP (College International pour la Recherche en Productique)*. 47(1):369-372.
82. Korhonen P. and Halme M. (1990). Supporting the decision maker to find the most preferred solutions for a MOLP-problem. *Proceedings of the 9th International Conference on Multiple Criteria Decision Making*. Fairfax, Virginia, USA, 173-183.
83. Kulturel-Konak, S., Smith, A. and Coit D. W. (2003): Efficiently Solving the Redundancy Allocation problem using Tabu Search. *IEE Transactions*, 35(6), 515-526.
84. Kulturel-Konak, S., Coit, D. and Baheranwala, F. (2006). Pruned Pareto-Optimal Sets for the System Redundancy Allocation Problem Based on Multiple Prioritized Objectives. *Journal of Heuristics* (in print).
85. Lahdelma, R., Hokkanen J. and Salminen P. (1998). SMAA- Stochastic Multiobjective Acceptability Analysis. *European Journal of Operational Research*, 106(1), 137-143.
86. Levitin G. and Lisnianski A. (1998). Structure Optimization of Power System with Bridge Topology. *Electric Power Systems Research*. 45(3):201-208.
87. Levitin, G., Lisnianski, A., Ben-Haim, H. and Elmakis, D. (1998). Redundancy Optimization for Series-Parallel Multi-State Systems. *IEEE Transactions on Reliability*, 47(2):165-172.
88. Levitin, G. (2000). Multi-state series-parallel system expansion-scheduling subject to availability constraints. *IEEE Transactions on Reliability*, 49(1):71-79.
89. Levitin, G. and Lisnianski A. (2000). Optimization of Imperfect Preventive Maintenance for Multi-state Systems. *Reliability Engineering and Systems Safety*. 67(2):193–203.
90. Levitin, G. (2001). Redundancy optimization for multi-state system with fixed resource-requirements and unreliable sources. *IEEE Transactions on Reliability*, 50(1):52-59.

91. Levitin G. and Lisninaski A. (2001). A New Approach to Solving Problems of Multistate System Reliability Optimization. *Quality and Reliability Engineering International*, 17:93-104.
92. Levitin, G. (2003). Linear Multi-state Sliding-window Systems. *IEEE Transactions on Reliability*. 52(2):263-269.
93. Levitin, G. (2005). Universal generating function in reliability analysis and optimization, Springer-Verlag, London, 2005.
94. Liang, Y-C and Smith, A. E. (2004). An ant colony optimization algorithm for the redundancy allocation problem (RAP). *IEEE Transactions on Reliability*, 53(3):417-423.
95. Lisnianski, A. and Levitin, G. (2003). Multi-state system reliability. Assessment, Optimization and Applications. Series on Quality, Reliability and Engineering Statistics, Vol. 6. Ed. World Scientific.
96. Lyu, R. M., Rangarajan, S., and Van Moorsel, P.A. (2001). Optimal Allocation of Test Resources for Software Reliability Growth Modeling in Software Development. *IEEE Transactions on Reliability*, 51(2):183-192.
97. MacQueen J. (1967): Some methods for classification and analysis of multivariate observations. In L. M. LeCam and J. Neyman, editors, *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1, 281–297.
98. Markowitz, H. M. (1952). Portfolio Selection. *Journal of Finance*, 7, 77-91.
99. Michalewicz, Z. and Schoenauer, M. (1996). Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, 4(1).
100. Misra K. B. (1971). Dynamic programming formulation of redundancy allocation problem. *International Journal of Mathematical Education in Science and Technology*, 2:207-215.
101. Misra K. B. and Sharma U. (1991). An Efficient Algorithm to Solve Integer Programming Problems Arising in System Reliability Design. *IEEE Transactions on Reliability*, 40:81-91.
102. Mitchell, M. (1996). An Introduction to Genetic Algorithms. MIT Press, Cambridge, MA.
103. Mühlenbein, H. and Schlierkamp-Voosen, D. (1993). Predictive Models for the Breeder Genetic Algorithm, I: Continuous Parameter Optimization. *Evolutionary Computation*, 1(1).

104. Murata, T., Ishibuchi, H. and Tanaka H. (1996). Multi-objective genetic algorithm and its applications to flowshop scheduling. *Computers in Industrial Engineering*, 30(4), 957-968.
105. Nakagawa, Y. and Miyazaki, S. (1981). Surrogate constraints algorithm for reliability optimization problems with two constraints. *IEEE Transactions on Reliability*, R-30, 175-180.
106. Nemhauser, G. L. and Wolsey, A. L. (1988). *Integer and Combinatorial Optimization*. Wiley, New York.
107. Oei, C. K., Goldberg, D. E. and Chang, S. (1991). Tournament selection, niching and the preservation of diversity. IlliGal Report No. 91011. Urbana-Champaign, IL. Department of general engineering, University of Illinois at Urbana Champaign.
108. Ouiddir, R., Rahli, M., Meziane, R., and Zebblah, A. (2004). Ant colony optimization for new redesign problem of multi-state electrical power systems. *Journal of Electrical Engineering*, 55(3-4):57-63.
109. Painton L. and Campbell, J. (1995). Genetic algorithms in optimization of system reliability. *IEEE Transactions on Reliability*, 44:172-178.
110. Papadimitriou, C. H. and Steiglitz, K. (1982). *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, Inc., New York.
111. Pareto, V. (1896). *Cours D'Economie Politique*. Lausanne, Switzerland: F. Rouge, vol. I and II.
112. Parmee, I. C. (2001). Poor-definition, uncertainty and human-factors: Satisfying multiple objectives in real-world decision-making environments. *First international conference on evolutionary multi-criterion optimization*. Springer-Verlag. Lecture notes in Computer Science No. 1993.
113. Pham, D. T. and Karaboga, D. (2000). *Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural networks*. Ed. Springer.
114. Pinedo, M. and Chao, X. (1999). *Operations scheduling with applications in manufacturing and services*. Irwin/McGraw-Hill.
115. Ponnambalam, S. G., Ramkumar V. and Jawahar N. (2001). A multiobjective genetic algorithm for job shop scheduling. *Production Planning and Control*, 12(8), 764-774.

116. Pourret O., Collet J. and Bon J-L. (1999). Evaluation of the unavailability of a multistate-component system using a binary model. *Reliability Engineering and System Safety*, 64(1):13–17.
117. Ramirez-Marquez, J. E. and Coit, D. (2004). A Heuristic for Solving the Redundancy Allocation Problem for Multistate Series-Parallel Systems. *Reliability Engineering & System Safety*, 83(3): 341-349.
118. Ramirez-Marquez, J. E. and Coit, D. W. (2005). A Monte-Carlo Simulation Approach for Approximating Multistate Two-Terminal Reliability. *Reliability Engineering and System Safety*, 87(2):253-264.
119. Rao, S. S. (1991). Optimization theory and application. New Delhi: Wiley Eastern Limited.
120. Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann- Holzboog.
121. Rietveld P. and Ouwersloot H. (1992). Ordinal data in multicriteria decision making, a stochastic dominance approach to siting nuclear power plants. *European Journal of Operational Research*, 56, 249-262.
122. Ross, S. M. (1993). Introduction to probability models. Academic Press, New York.
123. Rousseeuw, P. J. (1987): Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20, 53-65.
124. Rousseeuw P., Trauwaert E. and Kaufman L. (1989): Some silhouette-based graphics for clustering interpretation. *Belgian Journal of Operations Research, Statistics and Computer Science*, 29(3).
125. Sakawa, M. (1978). Multiobjective optimization by the surrogate worth trade-off method. *IEEE Transactions on Reliability*, 27, 311-314.
126. Schaffer, J. D., Caruana, R. A., Eshelman, L. J. and Das, R. (1989). A study of control parameters affecting online performance of genetic algorithms for function optimization. In Schaffer (1989), 51-60.
127. Schaffer, J. D. (1984). Some experiments in machine learning using Vector Evaluated Genetic Algorithms. Ph.D. dissertation, Vanderbilt Univ., Nashville, TN, 1984.
128. Schaffer, J. D. (1985). Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In Genetic Algorithms and their Applications: Proceedings of

- the First International Conference on Genetic Algorithms, 93-100. Hillsdale, New Jersey.
129. Seward L. E and Nachlas J. A. (2004). Availability Analysis for Multitask Production Systems. *The International Journal of Flexible Manufacturing Systems*. 16:91-110.
 130. Shaw, K. J. and Fleming, P. J. (1997). Including real-life preferences in genetic algorithms to improve optimization of production schedules. In *Proceedings of the GALESIA '97*. Glasgow, Scotland.
 131. Spears, W. M. and De Jong, K. A. (1991). An Analysis of Multi-Point Crossover. *Foundations of Genetic Algorithms*. San Mateo, California, USA: Morgan Kaufmann Publishers.
 132. Srinivas, N. and Deb, K. (1995). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248.
 133. Steuer, R. E. (1989). Multiple Criteria Optimization: Theory, Computation, and Application. Reprint Edition, Krieger Publishing Company, Malabar, Florida.
 134. Syswerda, G. (1990). Schedule Optimization using Genetic Algorithms. In *Handbook of genetic Algorithms*, Van Nostrand Reinhold, New York.
 135. Taboada, H., Baheranwala, F., Coit, D. W. and Wattanapongsakorn, N. (2005). Practical Solutions of Multi-objective System Reliability Design Problems using Genetic Algorithms. *Proceedings of the Fourth International Conference on Quality & Reliability (ICQR4)*, Beijing, China, August 2005.
 136. Taboada, H. and Coit, D. (2006a). Multiple Objective Scheduling Problems: Determination of Pruned Pareto Sets, Rutgers University IE Working Paper 06-02.
 137. Taboada, H. and Coit, D. (2006b). MOEA-DAP: A New Multiple Objective Evolutionary Algorithm for Solving Design Allocation Problems. Rutgers University IE Working Paper 06-026.
 138. Taboada, H., Espiritu, J. and Coit, D. (2006). MOMS-GA: A Multiobjective Multi-State Genetic Algorithm for System Reliability Optimization Design Problems. Rutgers University IE Working Paper 06-027.
 139. Taboada, H., Baheranwala, F., Coit, D. W. and Wattanapongsakorn, N. (2007a). Practical Solutions of Multi-Objective Optimization: An Application to System Reliability Design Problems. *Reliability Engineering & System Safety*, 92(3):314-322.

140. Taboada, H., Espiritu, J., Coit, D. and Levitin, G. (2007b). A multi-objective evolutionary algorithm for determining optimal configurations of multi-task production systems. Rutgers University IE Working Paper 07-014.
141. Taboada, H. A. and Coit, D. W. (2007) Data Clustering of Solutions for Multiple Objective System Reliability Optimization Problems, *Quality Technology & Quantitative Management Journal*, 4(2):35-54.
142. Tian, Z. and Zuo, M. (2006). Redundancy allocation for multi-state systems using physical programming and genetic algorithms. *Reliability Engineering and System Safety*, 91:1049-1056.
143. Tillman, F. A., Hwang, C. L., and Kuo, W. (1977). Determining component reliability and redundancy for optimum system reliability. *IEEE Transactions on Reliability*, 26(3):162–165.
144. T'kindt, V., Billaut, J.-C. (2002). Multicriteria Scheduling: Theory, Models and Algorithms. Springer, Berlin.
145. Ushakov, I. (1986). A Universal Generating Function. *Soviet Journal of Computer and System Sciences*, 24(5):37-49.
146. Ushakov, I. A. (1987). Optimal standby problems and a universal generating function. *Soviet Journal of Computing System Science*, 25(4):79-82.
147. Ushakov, I. (1988). Reliability Analysis of Multistate Systems by Means of a Modified Generating Function. *Journal of Information Process*, 24(3):131-135.
148. Van Veldhuisen, D. A. (1999). Multiobjective Evolutionary Algorithms: Classifications, Analyses and New Innovations. PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology., Ohio.
149. Van Veldhuizen, D. A. and Lamont, G. B. (2000a). Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolutionary computation*, 8(2):125-147.
150. Van Veldhuizen, D. A. and Lamont, G. B. (2000b). Multiobjective optimization with messy genetic algorithms. *In Proceedings of the 2000 ACM symposium on applied computing*: 470-476. Villa Olmo, Como, Italy. ACM.
151. Venkat V., Jacobson, S. and Stori J. (2004). A post-Optimality Analysis Algorithm for Multi-Objective Optimization. *Computational Optimization and Applications*, 28, 357-372.
152. Wang J-W., Zhang Q., Zhang J-M. and Wei X-P. (2005). An approach to Evolutionary Multiobjective Optimization Algorithm with Preference. *Proceedings of*

- the Fourth International Conference on Machine Learning and Cybernetics*, Guangzhou, China, 5:2966-2970.
153. Whitley, D. (1989) The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is best. *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, California, USA: Morgan Kaufmann Publishers.
 154. Xue J. and Yang K. (1995). Dynamic Reliability Analysis of Coherent Multistate Systems. *IEEE Transactions on Reliability*, 44(4):683-688.
 155. Yang, S. M., Shao, D. G. and Luo, Y. J. (2005). Dynamic Archive Evolution Strategy for Multiobjective Optimization. In Coello *et al.*, editors. *Evolutionary Multiobjective Optimization*. Springer-Verlag.
 156. Youssef A. M. A., Mohib A. and ElMaraghy H. A. (2006). Availability Assesment of Multi-State Manufacturing Systems Using Universal Generating Functions. *Annals of the CIRP (College International pour la Recherche en Productique)*. 55(1):445-448.
 157. Yu Lian, Shih H. M, Pfund M., Carlyle W. M. and Fowler J. W. (2002). Scheduling of Unrelated Parallel Machines: an Application to PWB manufacturing. *IIE Transactions*, 34, 921-931.
 158. Zeleny, M. (1982). Multiple Criteria Decision Making. McGraw-Hill series in Quantitative Methods for Management.
 159. Zhou, G. and Gen, M. (1997) A note on genetic algorithms for degree-constrained spanning tree problems. *Networks*, 30.
 160. Zitzler, E. and Thiele, L. (1998). An evolutionary algorithm for multiobjective optimization: The strength Pareto approach. Technical Report 43, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland.
 161. Zitzler, E. and Thiele, L. (1999). Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4), 257-271.
 162. Zitzler, E., Deb K. and Thiele L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173-195.
 163. Zitzler, E., Laumanns, M. and Thiele L. (2001). SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical report 103, Gloriastrasse 35, CH-8092 Zurich, Switzerland.

164. Zydallis, J. B, Van Veldhuizen, D. A. and Lamont, G. B. (2001). A statistical comparison of multiobjective evolutionary algorithms including the MOMGA-II. In Zitzler, E., Deb K., Thiele, L., Coello, C. A. C., and Corne, D., editors. First International conference on Evolutionary Multi-criterion optimization: 226-240. Springer-Verlag. Lecture notes in Computer science No. 1993.

Curriculum Vita

Heidi Arlene Taboada Jiménez

- 2007 Rutgers, The State University of New Jersey. New Brunswick, NJ.
January 2003 – June 2007. Ph D. in Industrial & Systems Engineering
- 2005 Rutgers, The State University of New Jersey. New Brunswick, NJ.
January 2003 – January 2005. MS. in Industrial & Systems Engineering
- 2002 Instituto Tecnológico de Celaya. Guanajuato, México.
September 2000 – June 2002. MS. in Industrial Engineering
- 2000 Instituto Tecnológico de Zacatepec. Morelos, México.
September 1994 – January 2000. B.S. in Biochemical Engineering

Publications

- 2007 Heidi Taboada, Fatema Baheranwala, David Coit and Naruemon Wattanapongsakorn. (2007). Practical Solutions for Multi-Objective Optimization: An Application to System Reliability Design Problems. *Reliability Engineering & System Safety*, 92(3):314-322.
- 2007 Heidi Taboada and David Coit. (2007). Data Clustering of Solutions for Multiple Objective System Reliability Optimization Problems. *Quality Technology & Quantitative Management Journal*, 4(2):35-54.
- 2007 David Coit and Heidi Taboada. (2007). Genetic Algorithms in Reliability. *In Encyclopedia of Statistics in Quality and Reliability*. John Wiley & Sons, Ltd. (to appear).
- 2007 Heidi Taboada, José Espiritu, David Coit and Gregory Levitin. (2007). A multi-objective evolutionary algorithm for determining optimal configurations of multi-task production systems. *European Journal of Operational Research* (under review).
- 2007 Heidi Taboada and David Coit. (2007). Multiple Objective Design Allocation Problems: Development of New Evolutionary Algorithms. *In Proceedings of the European Safety & Reliability Conference (ESREL)*, Stavanger, Norway, June 2007 (in print).

- 2006 Heidi Taboada and David Coit. (2006). Multiple Objective Scheduling Problems: Determination of Pruned Pareto Sets. *IIE Transactions* (under review).
- 2006 Heidi Taboada and David Coit. (2006). MOEA-DAP: A new Multiple Objective Evolutionary Algorithm for solving Design Allocation Problems. *Engineering Applications of Artificial Intelligence* (under review).
- 2006 Heidi Taboada, José Espiritu and David Coit. (2006). MOMS-GA: A Multiobjective Multi-State Genetic Algorithm for System Reliability Optimization Design Problems. *IEEE Transactions on Reliability* (under review).
- 2006 Heidi Taboada and David Coit. (2006). Data Mining Techniques to Facilitate the Analysis of the Pareto-Optimal Set for Multiple Objective Problems. *In Proceedings of the Industrial Engineering Research Conference (IERC)*, Orlando, Florida, May 2006.
- 2005 Heidi Taboada, Fatema Baheranwala, David Coit and Naruemon Wattanapongsakorn. (2005). Practical Solutions of Multi-objective System Reliability Design Problems using Genetic Algorithms. *In Proceedings of the Fourth International Conference on Quality & Reliability (ICQR4)*, Beijing, China, August 2005.