MULTIOBJECTIVE SIMULATION OPTIMIZATION USING ENHANCED EVOLUTIONARY ALGORITHM APPROACHES

by

HAMIDREZA ESKANDARI,
B. S., Electrical Engineering, University of Tehran, Tehran, Iran, 1998
M. S., Socio-Economic Systems Engineering, Iran University of Science and Technology, Tehran, Iran, 2001

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the Department of Industrial Engineering and Management Systems
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2006

Major Professor: Christopher D. Geiger

ABSTRACT

In today's competitive business environment, a firm's ability to make the correct, critical decisions can be translated into a great competitive advantage. Most of these critical real-world decisions involve the optimization not only of multiple objectives simultaneously, but also conflicting objectives, where improving one objective may degrade the performance of one or more of the other objectives. Traditional approaches for solving multiobjective optimization problems typically try to scalarize the multiple objectives into a single objective. This transforms the original multiple optimization problem formulation into a single objective optimization problem with a single solution. However, the drawbacks to these traditional approaches have motivated researchers and practitioners to seek alternative techniques that yield a set of Pareto optimal solutions rather than only a single solution.

The problem becomes much more complicated in stochastic environments when the objectives take on uncertain (or "noisy") values due to random influences within the system being optimized, which is the case in real-world environments. Moreover, in stochastic environments, a solution approach should be sufficiently robust and/or capable of handling the uncertainty of the objective values. This makes the development of effective solution techniques that generate Pareto optimal solutions within these problem environments even more challenging than in their deterministic counterparts. Furthermore, many real-world problems involve complicated, "black-box" objective functions making a large number of solution evaluations computationally- and/or financially-prohibitive. This is often the case when complex computer simulation models are used to repeatedly evaluate possible solutions in search of the best solution (or set of

solutions). Therefore, multiobjective optimization approaches capable of rapidly finding a diverse set of Pareto optimal solutions would be greatly beneficial.

This research proposes two new multiobjective evolutionary algorithms (MOEAs), called fast Pareto genetic algorithm (FPGA) and stochastic Pareto genetic algorithm (SPGA), for optimization problems with multiple deterministic objectives and stochastic objectives, respectively. New search operators are introduced and employed to enhance the algorithms' performance in terms of converging fast to the true Pareto optimal frontier while maintaining a diverse set of nondominated solutions along the Pareto optimal front. New concepts of solution dominance are defined for better discrimination among competing solutions in stochastic environments. SPGA uses a solution ranking strategy based on these new concepts. Computational results for a suite of published test problems indicate that both FPGA and SPGA are promising approaches. The results show that both FPGA and SPGA outperform the improved nondominated sorting genetic algorithm (NSGA-II), widely-considered benchmark in the MOEA research community, in terms of fast convergence to the true Pareto optimal frontier and diversity among the solutions along the front. The results also show that FPGA and SPGA require far fewer solution evaluations than NSGA-II, which is crucial in computationally-expensive simulation modeling applications.

To my wife and family, for their love and support.

ACKNOWLEDGMENTS

I would like to thank all of my dissertation committee members, Dr. Mansooreh Mollaghasemi, Dr. José A. Sepúlveda, Dr. Annie S. Wu and Dr. Ferenc Szidarovszky for their useful comments, suggestions, and help in producing a high quality research document. I would like to thank Dr. Mansooreh Mollaghasemi and Dr. Luis C. Rabelo for their financial support during the early stages of this research investigation. To my research advisor, Dr. Christopher D. Geiger, without your guidance, work ethic, motivation, and support I would not be at this juncture in life. Your support in submitting papers and attending many conferences has helped me gain the insight and knowledge necessary to complete this research. I value your opinions and thank you for expanding my horizons.

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

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1. Multiobjective Optimization

In today's competitive global business environment, a firm's ability to make the most appropriate critical decisions can be translated into a great competitive advantage. Most of these critical decisions are multiple objective problems in which management should be able to handle the challenges of conflicting objectives. For example, in supply chain management, the objective of reducing total costs typically opposes the objective of decreasing lead times, and improving product quality. These conflicting objectives are also encountered in other problem settings including job shop scheduling, inventory control, facility location, portfolio management and project management. In recent years, multiple objective problems have begun to draw the attention of practitioners and academicians alike.

Several methods exist that one could use to solve problems involving multiple objectives (Szidarovszky *et al.*, 1986; Mollaghasemi and Pet-Edwards, 1992). A naïve way is to select the most important performance objective and ignore the other less important objectives. This treatment of neglecting some objectives will undoubtedly result in poor solutions. Another method is to select a single objective for optimization and constrain the values of the other objectives to be within certain levels. The main drawback of this method is that the constrained objectives usually restrict the feasible solution space resulting in no feasible solution being found.

Other more traditional approaches for solving multiobjective optimization problems (MOPs) typically try to scalarize the multiple objectives into a single objective. This transforms the original multiple objective optimization problem formulation into a single objective optimization problem with a single solution. The major drawbacks of traditional methods that serve as motivation for using these alternative techniques include:

- The priority (or weight) vector used in the scalarization process greatly influences the final solution;
- Alternative solutions will not be available to decision-makers without at least changing some parameters such as the priority vector;
- Some optimal solutions may never be found if the objective space is not convex for minimization problems (Szidarovszky *et al.*, 1986 pp. 34-39); real-world problems are seldom convex (Silva and Biscaia, 2003);
- There are implications in the homogenization of different performance measures (such as cost, quality of products, and cycle times) to a common unit of measure; and
- Traditional approaches may not work effectively if objectives are noisy or have discontinuous variable space.

For example, consider the first drawback. A small perturbation in the priority vector values can greatly influence the obtained solution. Each certain pair of weights w_1 and w_2 ($w_2 = 1 - w_1$ for biobjective problem) results in single nondominated point in the tradeoff curve. However, the drawbacks of this class of approaches have motivated researchers and practitioners to seek alternative techniques to find a set of Pareto optimal

(nondominated) solutions rather than just a single solution (*e.g.*, Srinivas and Deb 1994; Deb, 2001; Coello *et al.*, 2002; Silva and Biscaia 2003). A solution is Pareto optimal if there exists no feasible solution for which an improvement in one objective does not lead to a simultaneous degradation in one or more of the remaining objectives.

1.2. Solution Dominance in Multiobjective (Deterministic) Problem Environments

In deterministic problem environments, most multiobjective optimization applications are gravitating towards using the nondomination-based approaches due to the limitations of traditional multiobjective methods. Assume that $f_i(\mathbf{A})$ and $f_i(\mathbf{B})$ are the values of objective function i ($i \in \{1, \ldots, m\}$) for two solutions **A** and **B**, where **A** and **B** are p-dimensional vectors of the decision variables. The desire is to minimize each objective function. In a deterministic problem domain, solution **A** *strictly dominates* (is better than) solution **B** if $f_i(\mathbf{A})$ is less than $f_i(\mathbf{B})$ for each objective function i. Figure 1.1 illustrates the concept of strict dominance graphically for an optimization problem in which $m = 2$ and the goal is to minimize both functions f_1 and f_2. In the figure, it can be seen that solution **A** strictly dominates all solutions in the shaded region, including solution **B**. It must be noted that in stochastic problem environments where the objective function values are uncertain, the definition of strict solution dominance must be modified. Nondomination considers all possible tradeoffs of the priorities of the given objectives, as shown in Figure 1.2, which shows the problem of minimizing two objectives.

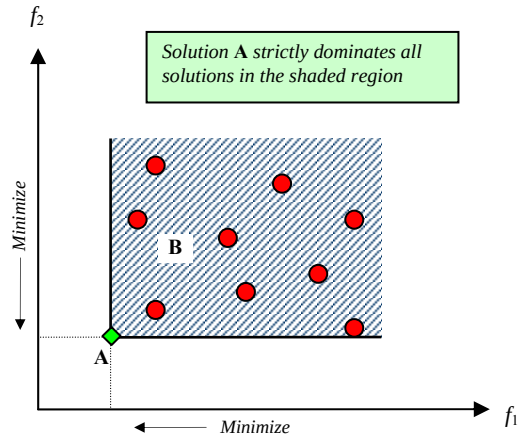

Figure 1.1: Illustration of strict dominance in a deterministic problem domain.

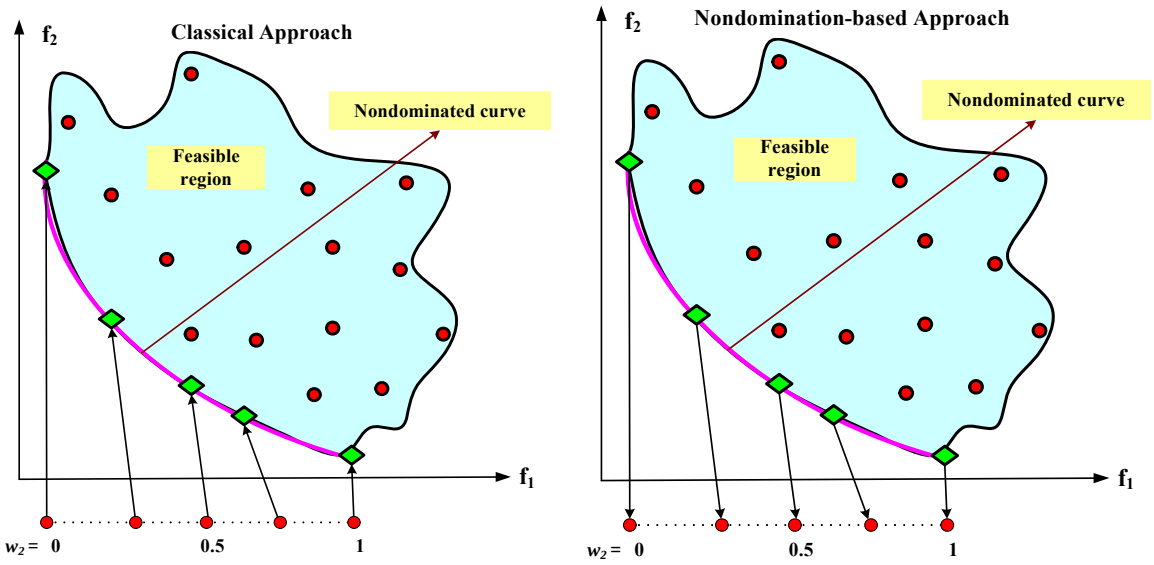

Figure 1.2: Illustration of the classical approach and nondomination-based approach for minimization problem with two objectives (Deb, 2001).

1.3. Systems Simulation Modeling

Due to the complexities and uncertainties existing in real-world problems, it is very difficult to solve single objective problems, let alone multiobjective problems, exactly using traditional analytical models. As an alternative to analytical methods,

computer simulation is an effective approach that can be used to model the complexities and uncertainties of the real-world problems without the limiting assumptions. Simulation can estimate the measures of the system performance. This is accomplished by performing *n* simulation replications. It is appropriate to note here that a known drawback of using simulation is that it can be computationally-expensive and time-consuming. When simulating realistic, large-scale stochastic systems, even a single replication can be computationally-prohibitive. Each replication is one sample observation (point estimate) for the performance measure. Then, the arithmetic mean of the *n* independent and identically distributed sample observations is used as an unbiased point estimate of the true population mean. Due to the randomness in the simulation model, a confidence interval is usually constructed for each system performance measure of interest. The analyst asserts that this confidence interval contains the true mean with a certain level of confidence. In most simulation studies, confidence intervals are employed in the output analysis of the model in addition to the sample means.

However, simulation modeling facilitates policy evaluation of a system and "what if" analyses. It alone lacks optimizing ability, and thus, should be combined with other analysis techniques to become most effective for optimization problems. The general approach to address this problem is the integration of an optimization subroutine or module with the simulation model.

1.4. Optimization via Simulation

In general, simulation optimization is the process of searching for the best set of model specifications, *i.e.*, input parameters and structural assumptions, where the objective value is the output performance of simulation model for the underlying system

(Olafsson and Kim, 2002). Figure 1.3 shows the general process of optimization via simulation. The optimization module uses the numerical values of the performance measures estimated by a simulation model (or set of models) to make decisions regarding the next set of candidate solutions. Thereafter, the optimization algorithm generates new model specifications through perturbations of existing solutions that are fed to the simulation model. This search continues until a user-specified stopping criterion is satisfied.

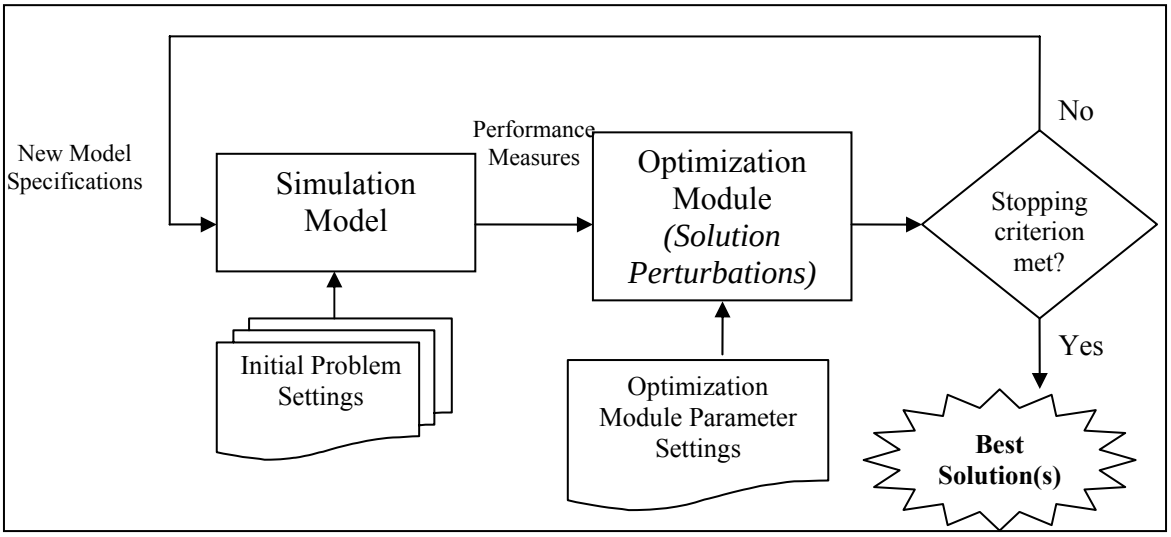

Figure 1.3: General process of simulation optimization.

Several simulation optimization approaches have been proposed by researchers. They differ primarily based on the problem settings and characteristics. Such settings and characteristics for simulation optimization problems include the nature of the solution space (continuous or discrete decision parameters), number of feasible solutions (relatively small, large but finite, or countably infinite), number of the performance measures (single objective or multiple objectives) (Andradóttir, 1998b; Azadivar, 1999; Swisher *et al.*, 2000; Olafsson and Kim, 2002). . It is also worthy to note that there is a

considerable gap between the approaches proposed in the research literature and those that are employed by commercial software packages for practical use.

Since evaluation of the measures of the system performance is performed by executing simulation runs that are often computationally-expensive and time-consuming, it is very important that an optimization algorithm be able to find optimal or near-optimal solutions in the early stages of the search process. The optimization algorithm should also be capable of effectively balancing the tradeoff between solution space exploration and solution space exploitation. In other words, an intelligent algorithm should search the feasible solution space thoroughly, and evaluate the regions around the local optima carefully in order to possibly find global optimal solution, which may be in another region. On the other hand, an optimization algorithm should be robust enough to handle the challenges of randomness and uncertainty involved in the estimated objective functions of the simulated model. In this case, the existing uncertainty and noise might hinder the optimization algorithm trying to move into improving directions.

1.5. Simulation Optimization of Multiple Stochastic Objectives

An issue that should be considered in the stochastic optimization context is the randomness effect of conflicting performance measures in the simulation models caused by the uncertain nature of different processes of the underlying system. The randomness effect of the performance measures plays an important role in the quality of the obtained results; thus, inefficient methods may lead to incorrect conclusions and improper decisions. The stochastic nature of simulation models together with costly simulation experimental runs makes the efficiency of the optimization methodology critical.

Due to the complexity and difficulty of dealing with these kinds of problems, few attempts have been made in multiobjective simulation optimization. The majority of these works focus on utility theory, interactive approaches, response surface methodology and goal programming (Mollaghasemi, 1994; Boyle and Shin, 1996; Lee *et al.*, 1996; Baesler and Sepulveda, 2000). To the best of the author's knowledge, the existing literature does not support an efficient approach for multiobjective simulation optimization to find Pareto optimal solutions.

1.6. Objectives of This Research

The primary objective of this study is to provide a modeling framework that integrates simulation models and nondomination-based multiobjective optimization methods. More specifically, in many applications of simulation modeling, the time to perform a single solution evaluation (replication) is of the order of minutes to hours, restricting the total number of solution evaluations needed for statistical precision. Additionally, many real-world problems often involve complicated stochastic (or noisy) multiple objective functions making a large number of the necessary replications computationally-prohibitive. Therefore, a multiobjective optimization algorithm capable of finding diverse Pareto optimal solutions and handling the uncertainty of stochastic multiple objective functions would be greatly beneficial. The purpose of this research is to propose such a stochastic multiobjective optimization methodology that finds evenly-distributed Pareto optimal solutions in a computationally-efficient manner.

1.7. Contributions of This Research

The contributions of this research are summarized in the following:

- This research proposes two new multiobjective evolutionary algorithms (MOEAs), called fast Pareto genetic algorithm (FPGA) and stochastic Pareto genetic algorithm (SPGA), for optimization problems with multiple deterministic objectives and stochastic objectives, respectively.
- New concepts of solution dominance are defined for better discrimination among competing solutions in stochastic environments. SPGA uses a solution ranking strategy based on these new concepts.
- New genetic operators are introduced to enhance both algorithms' performance in finding Pareto optimal solutions while minimizing computational effort. An elitism operator with high intensity is employed to ensure the quick propagation of the nondominated solutions, and a dynamic regulation operator to dynamically adapt the population size.
- In addition to distance and hypervolume ratio metrics, two new metrics, called diversity and delineation, are suggested to better discriminate among the MOEAs.
- New stopping criterion is introduced in which different numbers of solution evaluations are used for different test problems depending on the complexity of the problem.

1.8. Organization of This Dissertation

The remainder of this document is organized as follows. CHAPTER 2 briefly a reviews the existing literature in the area of simulation optimization. CHAPTER 3 presents the proposed framework for solving multiobjective simulation optimization problems. After introducing new solution dominance concepts for stochastic problem environments and a new solution ranking scheme, the logic of the proposed methodology,

which uses an evolutionary algorithm, is discussed. CHAPTER 4 summarizes the performance of the proposed framework for deterministic problem environments. It first discusses the experimental design followed by the computational results, including the comparison of the proposed methodology with another state-of-the-art algorithm to assess its performance. CHAPTER 5 discusses an enhancement of the proposed framework that makes it appropriate for stochastic problem environments. New dominance concepts are presented. Experimental results show the enhanced approach's competitiveness against a well-known state-of-the-art algorithm. This dissertation is concluded in CHAPTER 6 with a summary of the research and proposed directions for future study.

CHAPTER 2: OVERVIEW OF SIMULATION OPTIMIZATION THEORY AND APPROACHES

2.1. Introduction

In general, simulation optimization is the process of finding the best values of a set of decision variables, where the objective value is the output performance of simulation model for the underlying system. More specifically, in simulation optimization, one tries to find the best system design or solution to optimize the objective function

$$\min_{\theta \in \Theta} / \max f(\boldsymbol{\theta}), \qquad 2.1$$

where $\boldsymbol{\theta}$ denotes a k-dimensional vector of decision variables of the system, $\boldsymbol{\Theta}$ represents the constraint set on $\boldsymbol{\theta}$ (feasible region), and $f(\boldsymbol{\theta})$ is the real objective function, representing the expected system performance. There is no explicit analytical expression for the objective function f when f is stochastic, or it is very complicated if available (Law and Kelton, 2000, p. 646). Typically, this objective function is estimated using a function of the stochastic output $X(\boldsymbol{\theta})$, which might be an unbiased estimate for $f(\boldsymbol{\theta})$; that is, $f(\boldsymbol{\theta}) = E[X(\boldsymbol{\theta})]$ (Olafsson and Kim, 2002). There are other ways of formulating the simulation optimization problem. They can be found in Azadivar (1999).

Sections 2.2 and 2.3 present a brief review and several advantages, disadvantages, and applications of several simulation optimization techniques. Figure 2.1 shows the most popular simulation optimization approaches as categorized in this literature study. There are other ways to categorize simulation optimization approaches based on the

nature of the search space, *e.g.*, continuous decision variables versus discrete decision variables. Further material in this regard can be found in Fu (1994), Andradóttir (1998a), Andradóttir (1998b), Azadivar (1999), Swisher *et al.* (2000), April *et al.* (2001), Olafsson and Kim (2002), Fu (2002), and April *et al.* (2003).

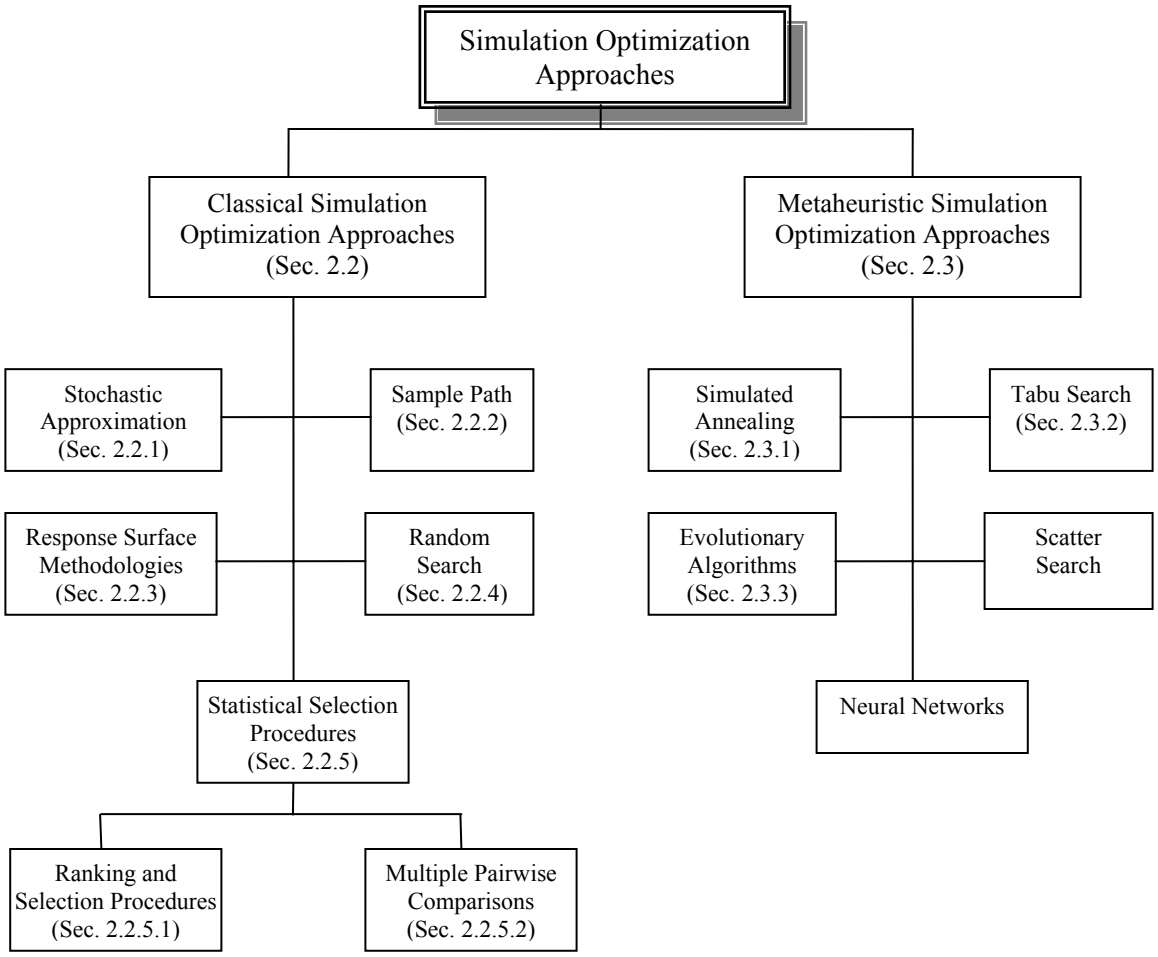

Figure 2.1. Taxonomy of existing simulation optimization approaches.

2.2. Classical Approaches for Simulation Optimization

Fu (2002) reports that research of classical approaches for simulation optimization includes five different categories:

1) stochastic approximation (*i.e.*, gradient-based approaches);
2) sample path optimization (also known as stochastic counterpart);
3) (sequential) response surface methodologies;
4) random search; and
5) statistical selection approaches (ranking and selection, multiple pairwise comparison).

2.2.1. Stochastic Approximation

Stochastic approximation (SA) is an iterative process that attempts to mimic the gradient search method used in deterministic optimization. The best known stochastic approximation algorithms are first introduced by Robbins and Monro (1951) and Keifer and Wolfowitz (1952). The general stochastic approximation methodology is based on the equality

$$\boldsymbol{\theta}_{n+1} = \Pi_{\boldsymbol{\theta}}\left(\boldsymbol{\theta}_n - a_n \hat{\nabla} f\left(\boldsymbol{\theta}_n\right)\right), \qquad 2.2$$

where $\hat{\nabla} f(\boldsymbol{\theta}_n)$ is the estimate of the gradient, $\Pi_{\boldsymbol{\theta}}$ denotes some projection back into the feasible region and a_n is the step size at iteration n. Under certain conditions, when the step size approaches zero with an slow enough rate, the asymptotic convergence of the SA algorithm can be guaranteed, *i.e.*, $\lim_{n\to\infty} a_n = 0$, and $\sum_n a_n = \infty$ according to the harmonic series $a_n = a / n$, where a is a positive scalar.

SA-based algorithms are generally used in simulation optimization problems with continuous decision variables. However, SA has also been applied to discrete variable problems (see, for example, Gerencser (1999)). Some of the major drawbacks of this approach are its slow convergence rate, its lack of an appropriate stopping rule and its difficulty in handling constraints (Shapiro, 1996). It has been found that, in practice, the performance of a SA-based algorithm strongly depends on the choice of the step size (Fu, 2002). Another disadvantage of this method is that it might find local optima, since it is based on gradient search method.

Many gradient estimation techniques have been developed to estimate the gradient in Eq. 2.2. One way for estimating the gradient in this equation is using either the naïve one-sided finite differences (FD) or two-sided symmetric differences (SD) given as

$$\hat{g}_i(\boldsymbol{\theta}_n) = \frac{\hat{f}(\boldsymbol{\theta}_n + c_n e_i) - \hat{f}(\boldsymbol{\theta}_n)}{c}, \text{ and} \qquad 2.3$$

$$\hat{g}_i(\boldsymbol{\theta}_n) = \frac{\hat{f}(\boldsymbol{\theta}_n + c_n e_i) - \hat{f}(\boldsymbol{\theta}_n - c_n e_i)}{2c}, \qquad 2.4$$

respectively, where e_i denotes the unit vector in the i^{th} direction and c_n represents a small change in each decision variable. One-sided finite differences and two-sided symmetric differences need k+1 and 2k simulation replications (k is the dimension of the vector $\boldsymbol{\theta}$) respectively, which require considerable computational effort. Spall (1992) proposes the simultaneous perturbations (SP) technique in order to increase computational efficiency. This technique, which perturbs the solution in all directions simultaneously, requires only two simulation replications regardless of the dimension of the search space. Spall (1992)

shows that the asymptotic convergence rate of the simultaneous perturbations is the same as that of the FD technique.

In order to improve the computational effort and convergence rate of the SA technique, many researchers focus on the direct estimation of the gradient. The best known techniques for direct gradient estimation are perturbation analysis (PA) (Glasserman, 1991; Ho and Cao, 1991) and likelihood ratio (LR) (Rubinstein, 1991; Rubinstein and Shapiro, 1993). Infinitesimal perturbation analysis (IPA) is a widely-used variant of the PA techniques. Kapuscinski and Taylor (1999) report that they successfully use IPA for optimization of capacitated production inventory systems. Fu (2002) summarizes the advantages and disadvantages of the main gradient estimation techniques, as shown in Table 2.1.

Table 2.1: Gradient estimation techniques for stochastic approximation (summarized from Fu (2002)).

Technique	Number of Simulations	Advantages	Disadvantages
IPA	1	Highly efficient, easy to implement	Limited applicability
Other PA	Usually > 1	Model specific implementation	Difficult to apply
LR	1	Requires only model input distributions	Possibly high variance
SD	$2k$	Widely applicable, model free	Generally noisier
FD	k+1	Widely applicable, model free	Generally noisier
SP	2	Widely applicable, model free	Generally noisier

2.2.2. Sample Path Optimization

Sample path optimization includes methods that attempt to approximate the original simulation optimization problem with a set of deterministic continuous optimization problems. To demonstrate the framework, suppose that $Y_1, \ldots, Y_N$ are N

independent random variables, where N is the size of the sample path (simulation replication), and a function h such that $X_i(\boldsymbol{\theta}) = h(\boldsymbol{\theta},Y_i)$ has the cumulative distribution function $F_{\boldsymbol{\theta}}$ for $i = 1, \ldots, N$. Then, the objective function is approximated by the sample mean over the N sample paths as $\hat{f}_N(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} h(\boldsymbol{\theta},Y_i)$, for all $\boldsymbol{\theta} \in \boldsymbol{\Theta}$. If each of the $h(\boldsymbol{\theta},Y_i)$ are independent and identically-distributed (IID) unbiased estimates of $f(\boldsymbol{\theta})$, then, for a sufficiently large N, the deterministic objective function $\hat{f}_N(\boldsymbol{\theta})$ approximates the expected objective function $\hat{f}_N(\boldsymbol{\theta})$ of the original simulation problem (Andradóttir, 1998b). In the simulation context, the common random numbers (CRNs) variance reduction technique provides the same sample path to calculate $\hat{f}_N(\boldsymbol{\theta})$ over different values of $\boldsymbol{\theta}$.

The main advantage of the sample path optimization methodology relative to gradient-based methods is that it is capable of handling optimization problems with complicated constraints (Fu, 2002). Rubinstein and Shapiro (1993) introduce the stochastic counterpart (SC) method, a variant of the sample path optimization method, to overcome the slow convergence rate, lack of robust stopping rule, and difficulties for handling constraints characteristic of SA-based methodologies. In this approach, $f(\boldsymbol{\theta})$ is approximated using the likelihood ratio method.

2.2.3. Response Surface Methodology

Generally, response surface methodology (RSM) is based on the principle of building metamodels that attempt to obtain an approximate functional relationship between the input decision variables and output objective function. RSM attempts to fit a

polynomial of appropriate degree to the response surface formed by different input decision variables. There is a variety of metamodeling approaches, and the two best known approaches are regression models and neural networks. Other metamodeling approaches include multivariate adaptive regression splines, radial basis functions, frequency domain approximations, spatial correlation models, and interpolative models known as kriging. Detailed discussions of these types of metamodels can be found in Barton (1998).

In the most recent research literature, metamodeling is performed in a more localized way called sequential response surface methodology, or sequential metamodeling. Sequential RSM procedures avoid exploring the entire search space, which can be costly and often impractical. Rather, it employs linear polynomials to approximate the response surface in small sub-regions of the feasible region. Thereafter, the gradient estimation and steepest decent method is used to move to a new sub-region. This exploration process continues until the linear model becomes inadequate as indicated by the approximated response surface with slope of zero. This implies that the sub-region includes the optimal point and higher order of response surface is required for appropriate fitness. Canonical and ridge analysis is usually employed to examine this sub-region thoroughly with regression models.

Safizadeh (2002) shows that, under certain conditions, the smaller size of the RSM sub-region reduces both the bias and variance of the gradient estimate. He provides guidelines for manipulating the size of the sub-region. In the study, a strong assumption is made that the positive correlation between the performance measures of two simulation

replications decreases when the difference of the values of input decision variables increases.

Keys and Rees (2004) propose a new sequential metamodeling strategy based on the nonparametric thin-plate splines. In their proposed procedure, the exploration starts with a uniform grid of points, and then the location of next system design point is found from the solution of a mathematical programming model. This solution is based on the distribution of the quantiles of estimated second derivatives of the response function.

It is worthy to note that even sequential RSMs require a substantial amount of computational effort, particularly when the number of decision variables is large. If the response surface of any sub-region has multiple optima, then the linear polynomial is not necessarily a good approximate. To obtain a better approximate, replications with a smaller sub-region is required to provide more accurate information. This problem makes the search for optimal solutions dramatically slow and increases the computational time.

2.2.4. Random Search Method

Random search method typically involves an iterative process in which the search moves successively from the current solution to a randomly-selected new (possibly better) solution in the neighborhood of that solution. This implies that the neighborhood structure must be well-connected in a certain precise mathematical sense so that the search may converge for all initial solutions (Andradóttir, 1998b). Random search methods have been mainly used for discrete variable optimization problems, though there is no particular theoretical reason that prevents applying them to continuous optimization problems. Random search methods are of special appeal for their generality and existence

of theoretical convergence proofs (Fu, 2002). The general random search, also summarized by Olaffson and Kim (2002), is as follows:

(0) Set iteration index $i = 0$; Select an initial solution $\boldsymbol{\theta}_i$ and perform the simulation to obtain expected value $X(\boldsymbol{\theta}_i)$.

(1) Select a candidate solution $\boldsymbol{\theta}_c$ from the neighborhood of the current solution $N(\boldsymbol{\theta}_i)$ according to some pre-specified probability distribution and perform the simulation to obtain expected value $X(\boldsymbol{\theta}_c)$.

(2) If the candidate satisfies the acceptance criterion based on the simulated performance, then $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_c$; otherwise $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i$.

(3) If the termination criterion is satisfied, then terminate the search; otherwise $i = i+1$ and go back to Step 1.

Different random search methods found in the literature primarily vary in the choice of the neighborhood structure, the method of candidate selection, the acceptance and termination criteria (Olafsson and Kim, 2002). The best known variants of the random search methods are the stochastic ruler algorithms, originally proposed by Yan and Mukai (1992), and those based on the simulated annealing approach. Detailed discussions on random search methods can be found in Andradóttir (1998b).

2.2.5. Statistical Selection Procedures

Statistical selection procedures are designed to distinguish the best solution(s) from among a given finite set of feasible solutions, that is $\boldsymbol{\Theta} = \{\theta_1, \theta_2, \ldots, \theta_n\}$, where n is relatively small. In the simulation context, statistical analysis is required to evaluate each feasible solution and compare their performance measure in order to consider uncertainty

surrounding the stochastic output. Several statistical procedures have been developed including ranking and selection (R&S) procedures and multiple pairwise comparison (MPC) procedures to address this problem. The primary difference between these two classes of procedures is that R&S procedures ensure the correct selection of the best solutions thar are within user-specified confidence and precision levels. MPC procedures make certain pairwise comparisons among feasible solutions in order to provide some inferences in the form of confidence intervals. In other words, R&S procedures help the analyst make a decision, whereas the latter only provide some statistical inferences for system performance between each pair of system design alternatives. A brief review of commonly used statistical selection procedures is now given.

2.2.5.1. Ranking and Selection

Ranking and selection is a practical tool for selecting the best solutions among a given set of competing solutions. The most popular R&S method is indifference zone ranking and selection. Assuming that decision-makers are indifferent to performance measure differences less than precision level $\delta > 0$, one can follow a procedure to make the right selection with a certain probability of correct selection, P^*. In other words, with at least a certain probability P^* the performance of the selected solution $\boldsymbol{\theta}'$ is within the δ interval of the performance of the best solution $\boldsymbol{\theta}^*$

$$\text{Prob}\left(\left| f\left(\boldsymbol{\theta}'\right) - f\left(\boldsymbol{\theta}^*\right)\right| < \delta\right) \geq P^* . \qquad 2.5$$

The precision level δ is called the indifference zone and probability of correct selection P^* is actually confidence level (1-α), and both should be pre-specified by the user. The two-stage procedure developed by Dudewicz and Dalal (1975) estimates the

appropriate number of simulation replications in order to guarantee the desired confidence of correct selection. In the first stage of sampling, the means and variances of each of the n feasible solutions are evaluated using $r_0 \geq 2$ replications. Then, the variances obtained in the first stage are used to determine the number of additional replications required for each solution in the second stage, say r_i (i = 1, …, n). Specifically,

$$r_i = \max\left\{r_0 + 1, \left\lceil \frac{h^2 S_i^2(r_0)}{\delta^2} \right\rceil\right\}, \qquad 2.6$$

where $\lceil x \rceil$ is the smallest integer that is greater than or equal to the real number x, and h is a constant that depends on r, P^*, and δ, which can be found in a given table. Finally, the weighted sample means is estimated and the best solution is selected. Intuitively, the higher confidence level P^*, or the more precision level (the smaller δ), the more replications is required which corresponds to Eq. 2.6. Values for P^* and δ should be selected depending on the goal of study and the system of interest (Law and Kelton, 2000). The initial number of replications r_0 plays an important role in determining the required computational time of the underlying system. It is advised that r_0 be at least 20 so that poor first stage variances are minimized, which usually results in a large number of required additional replications.

Ranking and selection procedures are easy to implement and interpret. This makes them popular when the number of system design alternatives n are relatively small, *i.e.*, 2 to 20. However, when the number of design alternatives is quite large, these procedures are inefficient and even impractical in terms of computation time. The reason is: (1) in Eq. 2.6, the constant h is an increasing function of n, and (2) Eq. 2.6 is derived based on

the worse case scenario, which means that the best design is exactly δ better than the other (n-1) system designs, which are all viewed as the second best designs. Thus, when the number of alternatives is large, a great amount of computational effort is required for the inferior alternatives making the analysis quite time-consuming and maybe computationally-prohibitive. Nelson *et al.* (2001) address this problem by proposing procedures for selecting the best design alternative, and they argue that these procedures are statistically more efficient than conventional procedures. They use the data provided in the first stage sampling to screen out the inferior alternatives, and the second stage sampling, which usually requires more computational effort, only includes the superior alternatives.

Another R&S procedure called subset selection attempts to find a subset of alternatives containing the best system design to screen out the inferior alternatives. This approach is useful when specification of several good alternatives is desired in the sense that the best alternative might be rejected for any reason. The first subset selection procedure is suggested by Gupta (1956) in which a random size subset containing the best design alternative is selected with user specified correct selection P^*, and without any indifference zone specification, *i.e.*, $\delta = 0$. This original procedure requires the assumptions of normality and equal and known variances among alternatives that are rarely satisfied in simulation optimization problems. Koenig and Law (1985) develop the indifference zone procedure, suggested earlier by Dudewicz and Dalal (1975), for R&S problems in application of selecting a subset of the given size m containing the best of n alternatives. The only difference between these two is that the indifference zone subset selection procedure takes on different values of h depending on m, n, δ, and P^*. However,

it is reported in the literature that the goal of most simulation studies is to specify the best system design rather than produce a subset of designs containing the best (Swisher and Jacobson, 1999).

2.2.5.2. Multiple Pairwise Comparisons

The main goal of the multiple pairwise comparison (MPC) procedures is to gain some statistical insight in the form of confidence intervals about the differences of each pair of design alternatives without guaranteeing any decision. There are several types of MPC procedures, including paired-*t*, Bonferroni, all-pairwise comparisons, all-pairwise multiple comparison (MCA), multiple comparison with a control (MCC), and multiple comparison with the best (MCB) (Swisher and Jacobson, 1999).

In paired-*t*, Bonferroni and all-pairwise multiple comparison (also called brute force), all possible pairwise comparison are performed constructing confidence interval for n system design alternatives. Using the Bonferroni approach, $n(n\text{-}1)/2$ confidence intervals are constructed at the confidence level of $\left(1-\alpha/\left[n\left(n-1\right)/2\right]\right)$ in order to provide the overall confidence level of $(1\text{-}\alpha)$ for all intervals simultaneously. This method is useful when the number of alternatives is quite small; otherwise, individual confidence intervals become wide and do not provide useful inferences. MCA methods are similar to the brute force approach except it constructs a simultaneous set of confidence intervals with the same half-width at an overall confidence level of $(1\text{-}\alpha)$. MCC techniques are typically used in the case when the analyst wishes to compare a set of design alternatives to a pre-specified system design such as to an existing design. The MCB approach is the most popular MPC procedure that attempts to identify the best

design from a set of alternatives. It requires constructing only n-1 simultaneous confidence intervals as $\mu_i - \min_{i \neq j} \mu_j$ for i = 1, 2, …, n, which is significantly less than those of the brute force approach.

Applying MCA, MCC, or MCB procedures requires IID with (approximately) normal distributions as well as equal variances. Yang and Nelson (1991) consider this requirement and modify MCA, MCC, and MCB procedures by incorporating two variance reduction techniques – common random numbers and control variates. They report that using variance reduction techniques achieves better statistical precision and ensures more confident decisions.

2.3. Metaheuristic Search Approaches for Simulation Optimization

Metaheuristic approaches have drawn considerable attention from many researchers in the last decade. The most popular metaheuristics are simulated annealing, tabu search, evolutionary algorithms, scatter search and neural networks. Each of these search heuristics has its own set of search features that makes them capable of escaping local optima. These approaches are all considered global search strategies in that they are capable of find optimal or near-optimal solutions in relatively short amounts of time. Originally designed for combinatorial optimization problems in the deterministic environment, these methods have been adapted for the stochastic environment associated with discrete simulation optimization, and they have been successfully applied to many real-world simulation problems.

It is important to note that several commercial software implementations currently incorporate these metaheuristic approaches. Although classical approaches for simulation

optimization account for a substantial amount of the research literature, none of them have been used in optimization modules embedded in the available commercial simulation software packages. Law and Kelton (2000) summarize the optimization methodologies utilized in the more popular commercial packages. This summary is given in Table 2.2. A brief discussion of the three best known metaheuristics follow. These are simulated annealing, tabu search and evolutionary algorithms.

Table 2.2. Commercial implementation of metaheuristic search strategies for simulation optimization (obtained from Law and Kelton (2000)).

Optimizer	Vendor	Simulation Software	Optimization Technique(s)
AutoStat	AutoSimulation, Inc.	AutoMod, AutoSched	Evolution Strategies, Genetic Algorithms
OptQuest	Optimization Technologies, Inc.	Arena, Micro Saint, Quest, Taylor Enterprise	Scatter Search, Tabu Search, Neural Networks
OPTIMIZ	Visual Thinking International, Ltd.	SIMUL8	Neural Networks
Sim Runner2	ProModel Corp.	MedModel, ProModel, Service Model	Evolution Strategies, Genetic Algorithms
Witness Optimizer	Lanner Group, Inc.	Witness	Simulated Annealing, Tabu Search

2.3.1. Simulated Annealing

Simulated annealing is a random local search technique that mimics the physical annealing process for crystalline solids. In this process, the molten solid is cooled very slowly from a high temperature until it reaches the ground temperature with a low energy state. If the cooling process occurs too quickly, the crystal is trapped in a much higher energy state than that of perfect crystal. In this analogy, state, energy, ground state, rapid quenching, temperature and careful annealing in the physical system correspond to feasible solution, evaluation function, optimal solution, local search, control temperature parameter and simulated annealing in the optimization problem (Michalewicz and Fogel, 2000).

In this method, search starts from an initial solution and moves from one solution to the candidate solution from its neighborhood, which is randomly selected. In order to overcome being trapped at local optima, simulated annealing allows acceptance (with certain probability) of inferior candidate solutions, or (for minimization problem)

$$\text{Prob}\left(\text{Accept}\,\boldsymbol{\theta}_C\right)=\begin{cases}1, & \text{if } X\left(\boldsymbol{\theta}_C\right)<X\left(\boldsymbol{\theta}_i\right)\\ e^{\frac{X(\boldsymbol{\theta}_C)-X(\boldsymbol{\theta}_i)}{T_i}}, & \text{otherwise}\end{cases}, \qquad 2.7$$

where T_i is the temperature parameter at iteration i that usually decreases during the run. This means that the candidate solution is certainly accepted if it is superior to the current solution. Otherwise, it is accepted with certain probability at which higher difference in their performance makes it less likely of accepting new solution. Simulated annealing is generally different from stochastic hill climbing only in temperature parameter, which is kept fixed in the latter method (Michalewicz and Fogel, 2000). The search usually starts with high values of T, and then it gradually decreases when the search progresses according to a function commonly referred to as the cooling, or annealing, schedule. This implies that the procedure starts with purely random search and ends in ordinary hill climbing approach with the hope of not being trapped at a local optimum and converging to the global optimum. Various cooling schedules have been proposed in the literature, including monotonic schedules, geometric schedules, and adaptive schedules. Alrefaei and Andradóttir (1999) report that they successfully use a constant temperature parameter in a specific simulation optimization problem.

The main advantage of simulated annealing relative to other metaheuristic approaches is that it has been shown to guarantee convergence in many settings (Jeon and Kim, 2004). On the other hand, it requires excessive computation time in practice, and it

is relatively slow in converging to good solutions in comparison to other metaheuristic approaches. Further, simulated annealing cannot perform intensification and diversification in an efficient manner (Jeon and Kim, 2004). Intensification reinforces attributes historically found good in order to return towards attractive regions to explore them carefully, while diversification drives the search into perhaps more promising regions.

Many different variants of simulated annealing have been suggested within the last decade in order to improve the drawbacks of the conventional version. For example, Azizi and Zolfagari (2004) propose two variations of the simulated annealing algorithm and successfully use them to minimize the makespan of a set of *n* jobs in the job shop scheduling problem. They note in their work that if some local optima are located at the relatively low temperature towards the end of the search, the search becomes trapped at a local optimum and the global optimal solution cannot be obtained. In their first approach, called adaptive simulated annealing, they consider the characteristics of the search trajectory in which adaptive cooling schedule is used that adjusts the temperature dynamically based on the number of consecutive improving moves. In this adaptive cooling schedule, the temperature is controlled by a single function at which temperature is kept above a minimum level. The temperature increases when any uphill move occurs. Such an improvement addresses the limitation of the traditional simulated annealing algorithm of having significantly low transition probability toward the end of the search.

2.3.2. Tabu Search

Tabu search is a metaheuristic first introduced by Glover (1997). It is a memory-based search strategy to guide the local search and avoid entrapment at local optima by

forbidding (or penalizing) moves that take a solution located in the previously visited region of the search space. The main idea of tabu search is that the memory enforces the search to deeply explore new areas of the search space within a single execution. The search keeps track of the sequence of recent moves or visited solutions in a memory list. A standard form of tabu list records the solutions that have been visited in the *n* last moves, where *n* is the tabu tenure. The best solution located in the neighborhood of the current solution is selected as the candidate solution if this move is not forbidden (*i.e.*, not in the tabu list). If this move is forbidden, the next best candidate solution from the neighborhood is selected that is not classified as tabu. However, in order to avoid not selecting a superior tabu solution found during the search, the tabu classification can be overridden when a predetermined aspiration criterion is satisfied. One popular aspiration criterion is to select the tabu move if it is the best ever solution found in the search that has not been visited before (Ho and Haugland, 2004). Finally, as with all metaheuristic search techniques, a stopping criterion determines when the search process halts. Usually, the search stops when a prespecified number of iterations has been completed, or when the current best solution has not been improved above a specified percentage within a certain number of consecutive iterations.

The type of memory described above is called short-term memory, or recency-based memory. Another type of memory, called frequency-based memory, or long-term memory, encourages moves that have led to solutions whose attribute have rarely been seen before (diversification). It also encourages moves that have historically led to improvements by reinforcing the consideration of special attributes of previously found good solutions in the remaining exploration (intensification) (Glover *et al.*, 1999). For

instance, long-term memory may allow the search to restart from a previously seen good solution with a different tabu list that guides the search in another direction from the good initial solution (Olafsson and Kim, 2002). One popular approach for long-term memory implementation is to measure the absolute frequency of a selected move during the search. An efficient implementation of long-term memory can balance between the diversification and intensification functions and increase the performance of the algorithm considerably.

Tabu search is a deterministic search approach and cannot guarantee the convergence. However, exploitation of the adaptive memory strategy is the unique feature of this search method that distinguishes it from other metaheuristic approaches. Many researchers have recently incorporated the adaptive memory feature of tabu method in their proposed metaheuristic algorithms. For example, Azizi and Zolfaghari (2004) incorporate a tabu list to their adaptive simulated annealing algorithm in order to improve the performance of their methodology by taking advantage of the tabu memory structure in order to keep track of recently visited solutions and prevent cycling. It has been reported that the combination of the tabu method with the complementary population-based approach of scatter search is a considerably powerful tool for simulation optimization problems (Glover *et al.*, 1999). Factors that affect the performance of tabu search include proper appropriate selection of the neighbor of a solution, the number of moves classified as tabu in the memory list, proper combination and management of the short-term and long-term memory, and efficient implementation of the intensification and diversification mechanisms (Jeon and Kim, 2004).

Tabu search has been widely used in a variety of applications ranging from job shop scheduling to power systems. For instance, Ho and Haugland (2004) use a tabu search algorithm to solve the vehicle routing problem with time windows and split deliveries. Time windows means that each customer has their own time interval in which to receive the service, and split deliveries means that the demand of a customer may be met by more than one vehicle, when the demand size exceeds the vehicle capacity. They apply tabu search successfully to minimize the number of vehicles, and the total distance traveled. They use a unique neighborhood structure that is defined by a union of four move operators including relocate, exchange, 2-opt, and relocate split operators.

2.3.3. Evolutionary Algorithms

Evolutionary algorithms are nature-inspired heuristics based on the Darwinian evolution theory on survival of the fittest (Holland, 1975; Goldberg, 1989; Mitchell, 1996). The main idea behind this family of algorithms is a population of individuals (solutions) with certain attributes is exposed to an environment. Some of the individuals are better suited to satisfy the requirement of the environment (*i.e.*, survive) and thus have more chance to be selected for populating the next generation of solutions. Their attributes are inherited by their offspring in the next generation. As a consequence, over several generations, inferior individuals with undesired attributes are gradually eliminated and the superior individuals evolve and eventually dominate the population. Such evolution is accomplished through different biological reproduction operations on the current individuals (parents) to generate the offspring for the new population. The most common operators include crossover and mutation. The crossover operator typically selects two individuals from the current population (usually superior individuals have

more chance to be selected) and combine them to make two new individuals. Thereafter, the mutation operator takes each individual and changes it slightly.

Evolutionary algorithms have many advantages over classical optimization approaches. One of the main advantages is that it is a population-based approach, which implies that if an optimization problem has multiple optimal solutions, an evolutionary algorithm is capable of finding multiple optimal solutions in its final population, whereas a classical optimization approach may find only a single optimal solution.

Another advantage of evolutionary approaches over those based on the locally searching the neighborhood of each single solution is their capability of more thoroughly exploring the feasible solution space in an efficient manner in terms of computational time (April *et al.*, 2003). The performance of the local search approaches based on the neighborhood sampling strongly depends on the distance of the optimal solution from the starting point as well as the appropriate definition of neighborhood because move operations can direct the search towards the optimal solution. Given the fact that in the stochastic simulation optimization context the fitness functions are estimated by running the expensive simulation models, finding a near-optimal or even good solution in an acceptable short period is considerably preferential.

Lacksonen (2001) performs an empirical study to compare the Hooke-Jeeve pattern search, Nelder-Mead simplex, simulated annealing, and genetic algorithm optimization approaches on variations of four industrial case study simulation models with 25 different test problems. Combinations of real variables, integer variables, non-numeric variables, deterministic constraints, and stochastic constraints are considered in the test problems. Based upon the general results regarding solution quality, the

decreasing order of the optimization approaches in terms of robustness of performance is genetic algorithms, pattern search, simulated annealing and simplex method. Genetic algorithms are found to be the most robust approach, since it finds near optimal solutions for all 25 test problems. The pattern search appears to be robust for small and medium size problems (less than 12 variables) with numeric variables. Simulated annealing and the simplex method are not found to be very robust approaches. However, it is important to note that these results are based on only 25 test problems in four application areas, and the performance of the approaches might be different on other test problems.

2.4. Evolutionary Algorithms for Multiobjective Optimization

As previously mentioned, evolutionary algorithms (EAs) are population-based search algorithms inspired by Darwinian evolutionary theory. It has been shown that EAs are intelligent optimization algorithms that are able to balance exploration and exploitation of the solution search space (Goldberg, 1989; Mitchell, 1996). In recent years, several variations of multi-objective evolutionary algorithms (MOEAs) have been developed to handle MOPs (Deb, 2001; Coello *et al.*, 2002). Many of the suggested MOEAs have been employed in a variety of real-world applications (Coello and Lamont, 2004). Some major advantages of using EAs for multiple objective optimization problems include the following:

- EA-based approaches are capable of finding a set of good solutions rather than a single solution (Srinivas and Deb, 1994; Deb, 2001).
- EA-based approaches are capable of exploring the search space more thoroughly with the smaller number function evaluations than other point-to-point local

search procedures such as simulated annealing and tabu search (April *et al.*, 2003).

- EA-based approaches are less dependent on the selection of the starting solutions, and they do not require neighborhood definition (April *et al.*, 2003).

2.4.1. Vector Evaluated Genetic Algorithm

The first multiobjective GA developed by Schaffer (1984) is called Vector Evaluated Genetic Algorithm (VEGA). Schaffer expands Grefenstette's GENESIS program in order to make it applicable for the problems with multiple objective functions by modifying the conventional selection method. VEGA is only different from the simple Genetic Algorithm (SGA) in the way selection procedure is implemented. In VEGA, assuming the population size is N, the population at each generation is divided into M equal sub-populations of size N/M. M is the number of the objective functions. The individuals are randomly placed in a sub-population. Then, individuals in each sub-population are assigned fitness according to a particular objective function. In this way, all M objective functions are considered in the selection operation of the whole population. Schaffer uses a fitness-proportionate selection operator. It should be noted that the entire population should be shuffled completely together before applying the usual crossover and mutation operations. Despite its simple implementation, this algorithm has a problem of biasedness towards some champion individuals and regions, as found by Schaffer (1984). This phenomenon, in genetics, is known as speciation, which means "…the evolutionary formation of new species among solutions that excels in some respect…" (Coello, 1999). VEGA works well during early generations. However, during later generations, the entire population usually converges towards some

regions that may not be the Pareto optimal front. This problem arises because particular solutions with better individual objective function values are emphasized without considering compromised solutions with average performance for each objective functions. Schaffer (1985) suggests two heuristics to resolve the speciation problem – the nondomination selection heuristic and the mate selection heuristic. It is worth mentioning that Schaffer found that his algorithm had a better performance in comparison to the adaptive random search method. The computational complexity of VEGA is the same as that of SGA because the selection operation is just repeated for each individual objective.

Tamaki *et al.* (1996) suggest a new algorithm called the Pareto reservation strategy in which they incorporate VEGA with the Pareto optimality concept. In this strategy, nondominated individuals in the current population are reserved and transferred to the next population so as to minimize the influence of the particular solutions with good individual objective function values. They also use a sharing technique to preserve diversity among solutions in the Pareto front.

2.4.2. Multiple Objective Genetic Algorithm

The first multiobjective GA based on the nondominated classification of the individuals, called Multiple Objective Genetic Algorithm (MOGA), is proposed by Fonseca and Fleming (1993). In this approach, the rank of each individual, say i, is determined by one plus the number of individuals in the current population that dominates it, *i.e.*, $r_i = 1+n_i$. Individuals are sorted in ascending order based on their rank. Thereafter, fitness values are assigned to individuals by using usually (but not necessarily) a linear mapping function. Then, their fitness values are averaged to ensure that the same rank individuals have identical fitness. In MOGA, the sharing function uses

the objective function values, instead of parameter values, which implies that the coverage quality of individuals' density in the search space might be poor. It is worth mentioning that no requirement in the approach enforces using the sharing function on the objective values. However, Coello (1996) reports that this is a theoretical problem and the algorithm works practically well. Fonseca and Fleming (1993) suggest a good method for updating the sharing parameter σ_{share} dynamically. This dramatically increases the performance of the algorithm. The overall computational complexity of MOGA is $O(MN^2)$.

MOGA has been used by many researchers in a variety of applications, particularly in control systems design. For example, Fonseca and Fleming (1998) apply MOGA using a multiple constraint handling strategy to solve a design problem for the low pressure spool speed governor of a Pegasus gas turbine engine. They consider several real nonlinear objective functions in their design including maximization of the stability of the closed-loop system, gain and phase margins, and minimization of the output error, while maintaining the rise time, settling time and overshoot in their desired levels. Oyama and Liou (2001) and in their follow-up work Oyama and Liou (2002) use MOGA for the design of cryogenic rocket engine turbopumps. They use this approach using floating-point representation, instead of binary representation, to optimize a single-stage centrifugal pump design as well as multi-stage pump design.

2.4.3. Nondominated Sorting Genetic Algorithm

Srinivas and Deb (1994) propose an interesting approach called Nondominated Sorting Genetic Algorithm (NSGA). This approach classifies the population into a number of layers of nondominated fronts. The first layer of nondominated individuals is

assigned the highest fitness value. Before identifying the second layer of nondominated individuals, sharing is done among the first front individuals to ensure a better spread of the individuals. This process is repeated for the remaining individuals until all individuals in the population belong to one front. For better understanding, the flowchart of NSGA is given in the Figure 2.2. Assigning a higher fitness value to individuals in the frontier layer increases convergence pressure on the population. This helps the layer move towards the true Pareto front. NSGA can be used for problems with any number of objectives as well as for maximization and minimization objectives (Srinivas and Deb, 1994). This approach uses a sharing function on the parameter values resulting in a noticeable uniform spread of individuals over the Pareto front. The overall computational complexity of NSGA is the maximum of the $O(MN^2)$ and $O(nN^2)$ which might be more than that of MOGA, $O(MN^2)$. The value n is the number of input parameters. It has been reported that NSGA is less efficient in finding the Pareto front in terms of quality of solutions and more sensitive to the sharing parameter σ_{share} than MOGA (Coello, 1996; Van Veldhuizen, 1999).

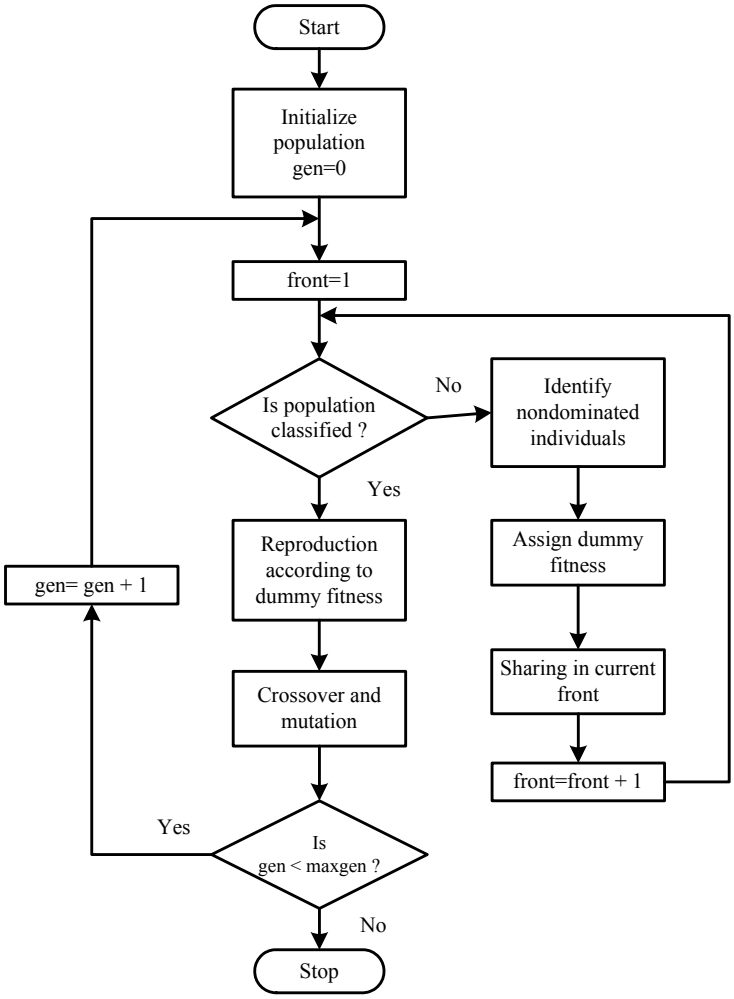

Figure 2.2. Flow diagram of NSGA (obtained from Srinivas and Deb (1994)).

NSGA has been used to obtain Pareto optimal solutions in a variety of applications. For example, Michielssen and Weile (1995) use NSGA for the design of an electromagnetic system. The NSGA is used by Vedarajan *et al.* (1997) for portfolio investment optimization. Mitra *et al.* (1998) use NSGA to solve biobjective optimization problems for three grades of Nylon 6 being produced in an industrial semi-batch reactor in which the total reaction time and the concentration of an undesirable cyclic dimmer in the product are to be minimized. NSGA has also been used by others to solve flowshop

and job shop scheduling problems (*e.g.*, Bagchi, 2001; Talbi *et al.*, 2001). Bagchi (2001) introduces an enhanced version of NSGA, called Elitist Nondominated Sorting GA (ENGA), which statistically improves the convergence speed to find out Pareto front by elitist selection pressure. Yee *et al.* (2003) solve several MOPs for both adiabatic and steam-injected styrene reactors successfully using NSGA with appropriate values for its parameters through several trials.

2.4.4. Niched Pareto Genetic Algorithm

Horn and Nafpliotis (1993) propose Niched Pareto Genetic Algorithm (NPGA), a multiobjective GA based on Pareto dominance using binary tournament selection, unlike VEGA, NSGA and MOGA that use fitness proportionate selection. In this scheme, two individuals i and j compete with respect to the number of individuals in a sub-population of size t_{dom} from the parent population that dominate them. If both individuals i and j are either dominated by at least one individual or not dominated by any individual, the tournament result will be determined through the calculation of their niche counts. The overall computational complexity of NPGA is the larger of $O(MNt_{dom})$ or $O(N^2)$. Dealing with problems with high number of objectives, if the sub-population size of comparison set t_{dom} is kept much smaller than N, then NPGA is much more efficient than other methods. But, if the sub-population size t_{dom} is equal to N, its overall complexity is the same as that of MOGA, *i.e.*, $O(MN^2)$. One of the attractive features of NSGA is that there is no need for fitness assignments unlike other methods (*e.g.*, VEGA, NSGA and MOGA) that particular fitness value should be assigned to each individual. NPGA uses tournament selection, which has better growth and convergence properties in comparison to fitness proportionate selection (Goldberg and Deb, 1991). However, NPGA requires

not only an appropriate selection of the sharing factor σ_{share} but also a good choice of the sub-population size t_{dom}. Moreover, the performance of NPGA is more sensitive to the right value of σ_{share} than NSGA, because in NPGA, unlike NSGA, the niche count of individual *i* is calculated when the individuals are located within the distance of σ_{share} regardless of how far they are from an individual *i* (Deb, 2001).

Schott (1995) uses NPGA for the design of a fault tolerant system to minimize objectives, unavailability and purchase cost. He compares NSGA with the ε-constraint method where he reports the superiority of NSGA. Abido (2003) uses the traditional NPGA with some basic modifications to solve the nonlinear constrained multiobjective environmental/economic dispatch problem. The problem is formulated to minimize fuel cost and emission, while satisfying the generation capacity, power balance and security constraints. He implements a real-coded GA with a blend crossover operator and a non-uniform mutation operator in order to overcome the difficulties of binary representation for large dimensioned problems with continuous search space (Herrera *et al.*, 1998). In this study, an average linkage-based hierarchical clustering algorithm is employed to reduce the size of nondominated set to provide the decision-maker with the manageable Pareto optimal set.

2.5. Multiobjective Evolutionary Algorithms under Uncertainty

A review of the literature reveals that only a few attempts have been made in the area of multiobjective evolutionary algorithms in stochastic environments. This undoubtedly is due to the existing uncertainties and complexities involved in the nature of the problems within this context. Moreover, to the best of our knowledge, no

multiobjective evolutionary algorithm exists that is capable of effectively dealing with uncertain and noisy objective functions that has been extensively tested.

Hughes (2001) presents a new approach for probabilistic ranking and selection for both single objective and multiobjective optimization problems accounting for uncertainties and noise present in the objective functions. Unlike the conventional ranking processes, his approach provides a statistical basis for addressing uncertainties and noise in the ranking and selection process. He experiments how the noise affects the assigned ranks within a population of solutions of an EA and finds that the probabilistic ranking process outperforms the ranking processes of MOGA and NSGA in the presence of high levels of noise. Further research to employ the suggested probabilistic ranking approach in evolutionary algorithms has not been found.

Teich (2001) introduces the concept of probabilistic dominance in multiobjective evolutionary algorithms when the objective values are uncertain but constrained within certain intervals. This is an extension to the definition of Pareto-based dominance. He modifies strength Pareto EA (SPEA) by updating the external set in order to handle estimated objective values bounded by intervals. Teich (2001) applies the modified strength Pareto EA (or SPEA2) to a hardware/software partitioning problem in order to minimize execution time and cost.

A large body of MOEA research focuses on algorithms that are modifications of NSGA-II. This is largely due to the influence of the exceptional work of Deb *et al.* (2002). Singh *et al.* (2001), for instance, propose a number of modifications to GAs to tackle some of the problems in noisy MOPs. They suggest improving the performance of original NSGA-II in noisy environments by modifying the ranking, selection and

diversity preservation schemes. Their suggested modifications are tested on a continuous, multi-modal problem that has more than one local Pareto optimal front. Noise is considered in the objective, decision and parameter space. However, they find that the performance of the modified NSGA-II is not significantly improved, particularly for solution diversity preservation. Babbar *et al*. (2003) suggest several modifications to the original NSGA-II ranking scheme to improve the performance of the algorithm in noisy environments. They test the modified NSGA-II on two test problems. It is worth mentioning that, in order to make a fairer comparison, only real nondominated solutions rather than rank-1 frontiers at the final generation should be benchmarked. Poles *et al.* (2003) propose a new EA for multiobjective optimization, called MOGA-II, which is different from the MOGA of Fonseca and Fleming (1993). They test the robustness of MOGA-II on noisy single-objective problems and compare its performance to that of two other algorithms.

In noisy genetic algorithms, Goldberg *et al.* (1992) find that, when dealing with noisy and uncertain objective functions, a large population size should be considered. This helps to prevent premature convergence in noisy and stochastic environments. Miller (1997) suggests that, under certain assumptions, there is a good approximation to estimate population size depending on the noise level. He also proposes some approximations to estimate the lower and upper bounds of the appropriate number of samplings for each solution.

2.6. Multiobjective Simulation Optimization

Due to the uncertainties existing in the nature of stochastic simulation problems, considering the additional complexity of dealing optimizing multiple objectives makes

solving this type of problem very challenging. Most likely, this is the main reason that only limited works have been done in this research area compared to using stochastic simulation for the optimization of a single objective. It is worth mentioning that most of these attempts use response surface methodologies, goal programming, and/or interactive multiobjective algorithms that the decision-maker uses to direct the search. These methods typically suffer from local optimality, absence of pre-knowledge on underlying system and individual objective ranges, and lack of an automated search process. Additionally, most of these methods disregard the stochastic nature of the output responses and perform the search deterministically.

Mollaghasemi *et al.* (1991) propose an approach in which they integrate the gradient search method and multiple attribute value function. Evans *et al.* (1991) review some of the best-known multiobjective optimization techniques categorized based on the three types of approaches: prior, progressive, and posterior articulation preferences that can be used for stochastic simulation models. They describe some important problem characteristics that should be considered in the selection of an appropriate multiobjective optimization technique for integration with simulation models. Mollaghasemi and Evans (1994) introduce an interactive approach based on the multiobjective optimization approach, called STEP method. Briefly, the STEP method is a multiobjective programming algorithm which attempts to minimize the maximum deviation of objectives from the ideal solution using relative weight of deviations. The decision-maker is then provided with the obtained solution, and asked to identify the satisfactory and unsatisfactory objectives in order to direct the search for accomplishing improvement. Mollaghasemi and Evans (1994) modify the STEP method for applications to simulation

models by using gradient search for each objective to find an ideal solution. A job shop model is used for application of the proposed interactive algorithm with six decision variables, the number of machines at each of six job stations, and four objectives including average time in system for three different part types and average machine utilization for all machine groups.

Baesler and Sepulveda (2000) suggest a new approach for multiobjective simulation optimization by using the GA within goal programming. This approach, unlike previous approaches that disregard the stochastic nature of output responses, employs the variances of the responses in order to perform the search stochastically towards the solution with the minimum weighted deviation from the target levels. They use a statistical grouping procedure based on Tukey's method to cluster the individuals in a population where there is a statistical difference between individuals of two different groups, but not between individuals within a group. A fitness-proportionate scheme is used to select a group from which an individual is randomly chosen. They implement a real coded-GA using blend crossover and uniform mutation operators. The same authors apply their proposed methodology to design a cancer treatment center facility. The decision variables of the underlying system include the number of treatment chairs at ambulatory treatment center, number of blood nurses, laboratory capacity, and pharmacy capacity. In this study, they consider four measures of system performance including minimization of patient's waiting time and closing time as well as maximization of nurse utilization and chair utilization. They show that the configurations found using the proposed methodology are all better than the existing configuration ranging from 18 to 25 percent improvement.

Joines *et al.* (2002) introduce a GA-based multiobjective simulation optimization approach using a modified version of the original NSGA-II. They apply their methodology to a real-world supply chain optimization problem with two objectives, gross margin return on investment and customer service level. They find Pareto optimal solutions for different levels of customer service, which provide valuable information for the decision-maker. In single objective simulation optimization, Hedlund and Mollaghasemi (2001) develop an optimization framework by incorporating an indifference zone ranking and selection procedure into a GA and using common random numbers to reduce the disturbance caused by the effect of noise.

2.7. Summary

This chapter provides a review of the existing relevant literature in the area of multiobjective optimization. We then discuss simulation optimization and applications of genetic algorithms for multiobjective optimization. Table 2.3 summarizes the key features of the best known simulation optimization approaches.

At the time of writing this dissertation document, no commercial simulation software uses classical optimization approaches, because they usually require not only a considerable amount of computational effort but also a great deal of technical sophistication on the part of the user (Andradóttir, 1998b; April *et al.*, 2003). Leading commercial simulation software employ metaheuristic approaches in their optimization modules. Moreover, there is a significant trend into population-based evolutionary approaches including genetic algorithm and scatter search (or hybrid approaches). These advantages include finding a set of good solutions rather than a single solution (Deb, 2001), exploring the search space more thoroughly with the smaller number function

evaluations, being less dependent on the selection of the starting solutions, and not requiring neighborhood definition (April *et al.*, 2003).

Table 2.3. Summary of simulation optimization approaches (obtained from Fu (2002)).

Approach	Key Features
Gradient Search	Move locally in most promising direction, according to gradient
Random Search	Move randomly to new point, no information used in search
Simulated Annealing	Sometimes move in locally worse directions, to avoid being trapped in local optima
Genetic Algorithms & Scatter Search	Population based, generates new members using (local) operations on attributes on current members
Tabu Search	Use memory (search history) to avoid tabu moves
Neural Networks	(Nonlinear) Function approximation
Math Programming	Powerful arsenal of rigorously tested software

On the other hand, any proposed methodology should be able to handle the uncertainty and noise involved in the objective functions and avoid drawbacks of traditional multiobjective optimization techniques mentioned in CHAPTER 1.

CHAPTER 3: PROPOSED METHODOLOGY

3.1. Introduction

A review of literature reveals that limited work has been done in the area of multiobjective simulation optimization, most likely because of the existing uncertainties and complexities involved in the nature of the problems. The suggested approaches are typically suffering from local optimality, absence of pre-knowledge on underlying system and solution objective ranges, and/or lack of automated search process. Additionally, most of these methods disregard the stochastic nature of the output responses and perform the search deterministically without providing any statistical guarantee that the search is progressing in the right direction.

The primary purpose of this research is to develop a GA-based stochastic multiobjective optimization methodology to find Pareto optimal solutions for simulation models in a short period of time. This chapter presents a proposed modeling framework for multiobjective optimization in deterministic problem environments integrating nondomination-based multiobjective optimization methods and evolutionary algorithms. The proposed multiobjective evolutionary algorithm, which is described in detail, uses a newly introduced ranking strategy and new search operators. This chapter is concluded by a brief discussion of the proposed MOEA's computational complexity.

3.2. A Proposed Methodology – Fast Pareto Genetic Algorithm (FPGA)

The proposed framework named fast Pareto genetic algorithm (FPGA) utilizes a population-based evolutionary algorithm. However, more importantly, this framework incorporates a new solution ranking strategy into an MOEA. A real-coded GA is implemented to avoid the difficulties associated with binary representation and bit operations, particularly when dealing with continuous search spaces with large dimension. Recall that each solution to a MOP is represented by an n-dimensional vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, where a decision variable x_i is a real number bounded by a lower limit a_i and upper limit b_i, *i.e.*, $x_i \in [a_i, b_i]$. The dimension of the vector is equal to the number of decision variables of the problem under study. Figure 3.1 gives the pseudocode for FPGA and Figure 3.2 shows the logic flow of FPGA.

```
Initialize user decision parameters (numvars, numobjs, maxpopsize, maxsoleval, p_c, p_m, …)
t := 0
create initial random population P_t = {x_1^t, x_2^t, x_3^t, …}
evaluate(P_t)
do while (stopping criterion is not met)
{
    t := t +1
    P'_t := select(P_{t-1}) // select pairs of solutions for reproduction
    O_t := crossover( P'_t )
    O_t := mutate(O_t)
    evaluate(O_t)
    CP_t := P_{t-1} ∪ O_t // form composite population
    rank(CP_t)
    regulate(CP_t)
    P_t := generate(CP_t)
}end do
```

Figure 3.1. Pseudocode of the proposed fast Pareto genetic algorithm (FPGA).

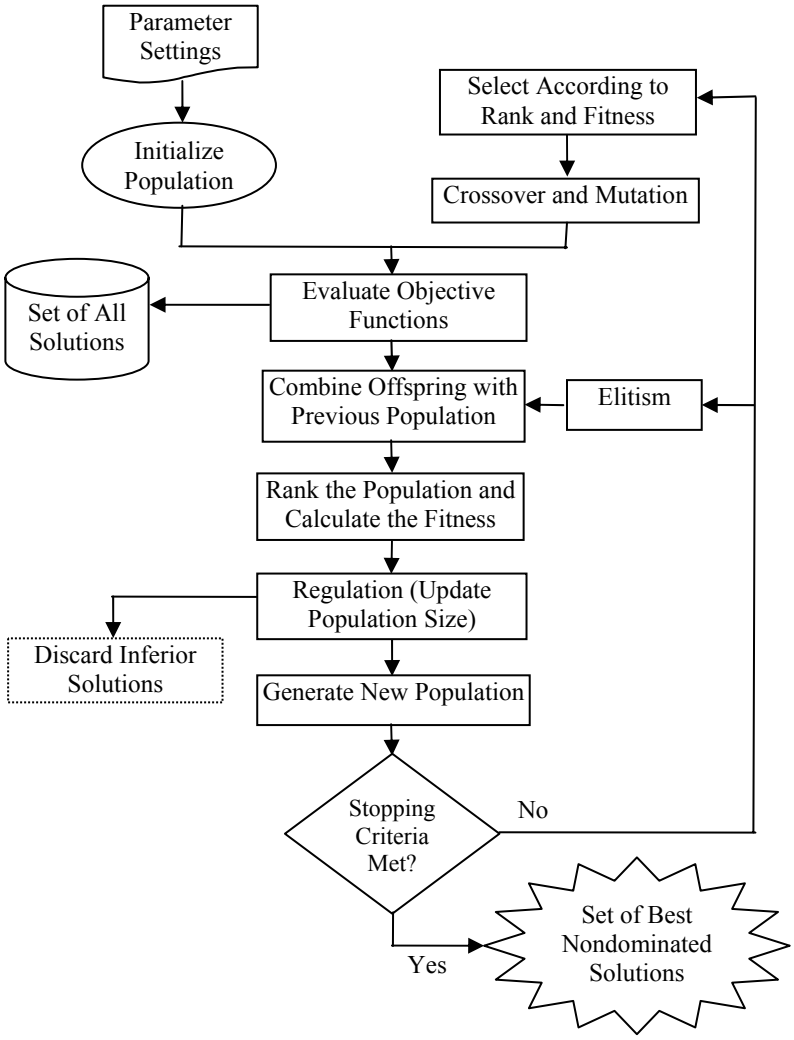

Figure 3.2. Logic flow of the fast Pareto genetic algorithm (FPGA).

The major steps of FPGA are as follows:

1. Initialize all decision parameters to user-specified values;
2. Create an initial population of candidate solution vectors randomly at the first generation; however, FPGA can be easily modified to generate the initial population heuristically, seeded with user-defined solution vectors, or using a combination of these approaches;

3. If it is the first generation, go to Step 5; otherwise, increment the generation number and select pairs of solutions as parents from the previous population in the reproduction operation using binary tournament selection;
4. Perform the crossover and mutation operations to generate candidate solutions (offspring);
5. Evaluate the candidate solution vectors for the *m* objective functions and record them;
6. Combine generated candidate solutions with the previous population to form a composite population;
7. Rank the composite population of solutions based on the new ranking strategy using their fitness values;
8. Regulate the population size according to the number of nondominated solutions and generate a new population from the composite population by discarding the inferior (dominated) solutions; and
9. Terminate the search if the stopping criterion is met; otherwise, return to Step 3.

In the proposed methodology, no input preferences are required from the decision-maker, neither any interaction during the search which even provides more information for the decision-maker. At the end of the search, it is expected that a large set of nondominated solutions are found. Using an appropriate screening algorithm, this large set of Pareto optimal solutions reduced to a manageable size of optimal solutions.

3.2.1. FPGA Initialization and Solution Evaluation

After initializing the user-specified parameter settings (*e.g.*, number of decision variables, number of objectives, maximum population size, maximum number of solution

evaluations, *etc.*), the initial population is created by random sampling of each decision variable within its defined range of variation. The user can also include some promising solutions if prior knowledge about the problem under study is available. Another approach is to take initial solutions from either the boundary of the search space or scan the search space with equal intervals as a grid. An initial population can also be generated by combination of the methods described above. In this research, the initial population is generated randomly. The evaluation of new solutions in terms of the objective functions is accomplished by calculating the corresponding evaluation function (*e.g.*, a mathematical closed-form expression, a computer simulation model) specified by the underlying problem. At each generation, the obtained solutions with their corresponding objective values are all recorded. If a solution advances to subsequent generations, its corresponding attributes are retrieved and copied to the new generations. In FPGA, before ranking and fitness assignment is performed, the new solution set $\mathbf{O}_t$ generated by crossover and mutation operations are combined with previous population $\mathbf{P}_{t-1}$ to form the composite population $\mathbf{CP}_t$, *i.e.*, $\mathbf{CP}_t = \mathbf{P}_{t-1} \cup \mathbf{O}_t$, where $\cup$ denotes the union of the two sets.

3.2.2. Solution Ranking and Fitness Assignment

The new ranking strategy is based on the classification of candidate solutions of the composite population $\mathbf{CP}_t$ into two different categories (ranks) according to solution dominance. All dominated solutions are identified as the second rank. These ranks are used to evaluate solution fitness for the purpose of solution reproduction. It is important to note here that a solution with a larger fitness value is preferred.

Rank-1 Solutions

Firstly, all nondominated solutions are identified as the first rank, which implies that there is no solution that is better than these solutions with respect to all objectives simultaneously. The fitness of the nondominated solutions in the first rank is calculated by comparing each nondominated solution with one another and assigning a fitness value. These values are computed using the crowding distance approach suggested by Deb *et al.* (2002), which has been shown to help maintain diversity among the nondominated solutions on the Pareto optimal front. The larger a solution's fitness value, the greater the distance that solution is from its neighboring nondominated solutions along the Pareto front.

Rank-2 Solutions

All dominated solutions are identified as the second rank. Each dominated solution in the second rank is compared with all other solutions and assigned a fitness value depending on the number of solutions they dominate. The idea here is similar to the strength concept employed in SPEA and SPEA2; however, it has been generalized. The fitness assignment takes into account both dominating and dominated solutions for any dominated solution. Here, each solution in the composite population $\mathbf{CP}_t$ is assigned a net strength value $S(\mathbf{x}_i)$, indicating the number of solutions it dominates, where

$$S\left(\mathbf{x}_i\right) = \left|\left\{\mathbf{x}_j \mid \forall \mathbf{x}_j \in \mathbf{CP}_t \wedge \mathbf{x}_i \succ \mathbf{x}_j \wedge j \neq i\right\}\right|. \tag{3.1}$$

The cardinality of a set is denoted as $|\cdot|$ and the expression $\mathbf{x}_i \succ \mathbf{x}_j$ means solution $\mathbf{x}_i$ dominates solution $\mathbf{x}_j$. Then, the fitness value of each dominated solution is calculated by

$$F(\mathbf{x}_i) = \sum_{\mathbf{x}_i \succ \mathbf{x}_j} S(\mathbf{x}_j) - \sum_{\mathbf{x}_j \succ \mathbf{x}_i} S(\mathbf{x}_j), \ \forall \mathbf{x}_j \in \mathbf{CP}_t \wedge j \neq i. \tag{3.2}$$

In other words, a fitness value is assigned to each dominated solution $\mathbf{x}_i$ is equal to the summation of the strength values of all solutions it dominates minus the summation of the strength values of all solutions by which it is dominated. In contrast to SPEA and SPEA2 where the strength values of only the solutions by which $\mathbf{x}_i$ is dominated (*i.e.*, the second term in Eq. 3.2) is considered. This strategy provides more information on Pareto dominance and niching relations among solutions in the composite population and reduces the chance that two solutions have the same fitness value. Thus, no additional diversity preservation mechanism is used among the dominated solutions in the second rank requiring less computation (unlike the SPEA2 which requires much higher computation for the density estimator). It is interesting to note that if most solutions do not dominate one another, it is implied that they belong to the first rank where crowding distance operator is invoked to maintain the diversity among them (discussed in detail in the next section).

After the fitness values of all candidate solutions in $\mathbf{CP}_t$ are calculated, the solutions are compared. Three different scenarios might occur. In the first scenario, two selected solutions have different ranks in which the solution with the better rank is preferred. In the second scenario, two solutions have the same rank but different fitness values in which the solution with larger fitness value is preferred. In the last scenario, two solutions have the same rank and fitness value where one of them is randomly preferred.

3.2.3. Distance Crowding Operation

In order to take the right proportion of nondominated solutions to maintain an even distribution of solutions along the Pareto optimal front, the crowded tournament selection operator originally introduced by Deb (2002) in NSGA-II is employed. This new approach does not have the challenges of using the sharing function method including selecting the value of the sharing parameter σ_{share} and the large computational complexity. Briefly explained, the crowding distance of a set of solutions estimates the density of the solutions surrounding any one particular solution in the population. It is determined by calculating the average distance of two solutions on either side of the solution in question along each of the objectives. Crowding distance is used as an estimate of the normalized perimeter of the cuboids formed by using the nearest neighbors as the vertices. Figure 3.3 shows how the crowding distance of a solution *p* is calculated as half of the perimeter of the cuboid. The interested reader is referred to Deb (2001) and Deb *et al.* (2002) for a more detailed discussion of this operator.

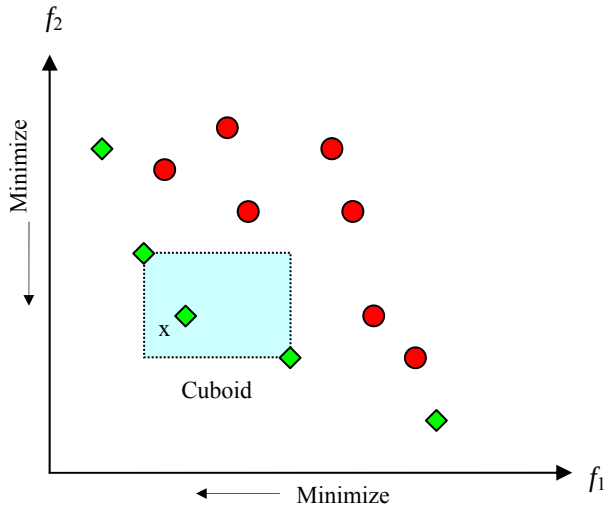

Figure 3.3: Illustration of crowding distance calculation.

However, in many applications, particularly in real-world problems, to emphasize the tradeoff among the objectives, the normalized area of the cuboid is suggested rather than the normalized perimeter. After all nondominated solutions in the population are assigned distance crowding values, the solutions are then compared to one another. In order to maintain diversity along the Pareto optimal front, the distance tournament selection operator is performed both to assign higher priority to less crowded nondominated solutions and to select the right subset of the nondominated set to copy to the next generation when the size of the nondominated set exceeds half of the pre-specified maximum population size.

3.2.4. Elitism and Expansion Operations

An elitism operator with relatively high intensity is implemented to ensure propagation of the nondominated solutions (*i.e.*, elite solutions) to subsequent generations. This is accomplished by copying all solutions in the population in the previous generation $\mathbf{P}_{t\text{-}1}$ to the composite population $\mathbf{CP}_t$. Combination of previous generation $\mathbf{P}_{t\text{-}1}$ with generated offspring $\mathbf{O}_t$ provides an opportunity to preserve the superior solutions in the next generation and discard the inferior solutions depending on the number of nondominated solutions obtained in the composite population.

The number of nondominated solutions usually increases over generations resulting in low elitism intensity in early generations if the population size is quite large and kept fixed. Moreover, the fluctuations of the number of nondominated solutions over generations demand an adaptive population sizing strategy to place appropriate emphasis of elitism intensity on nondominated solutions. If elitism intensity is too high, premature convergence might occur and if elitism intensity is too low, convergence might be too

slow and computationally-expensive. Therefore, FPGA employs a regulation operator to dynamically adjust the population size until it reaches a user-specified maximum population size as calculated by

$$\left|\mathbf{P}_t\right| = \min\left\{a_t + \left\lceil b_t \times \left|\left\{\mathbf{x}_i \mid \mathbf{x}_i \in \mathbf{CP}_t \wedge \mathbf{x}_i \text{ is nondominated}\right\}\right| \right\rceil, \textit{maxpopsize}\right\}, \tag{3.3}$$

where $\left|\mathbf{P}_t\right|$ is the population size at generation *t*, a_t is a positive integer variable that might change over generations, b_t is a positive real variable that might change over generations, $\lceil x \rceil$ is the smallest integer that is greater than or equal to the real number *x*, and *maxpopsize* is the user-specified maximum population size.

FPGA, unlike many of the other existing MOEAs, benefits the dynamic small number of offspring created by crossover and mutation operations over generations as calculated by

$$\left|\mathbf{O}_t\right| = \max\left\{c_t + \left\lceil d_t \times \left|\left\{\mathbf{x}_i \mid \mathbf{x}_i \in \mathbf{CP}_t \wedge \mathbf{x}_i \text{ is nondominated}\right\}\right| \right\rceil, \textit{maxsoleval}\right\}, \tag{3.4}$$

where $|\mathbf{O}_t|$ is the number of offspring created at generation *t*, c_t is a positive integer variable that might change over generations, d_t is a positive real variable that might change over generations, and *maxsoleval* is the user-specified maximum number of solution evaluations at each generation. It is interesting to note that this feature makes FPGA capable of saving a significant number of solution evaluations early in the search and utilizes the exploitation in a more efficient manner at later generations. Creating large number of offspring at early generations consumes considerable number of solution evaluations limiting the total number of generations, which results in no extensive utilization of exploitation, especially if the number of solution evaluations is restricted. Bear in mind that in expensive MOPs, where a small number of solution evaluations is

desired, more emphasis on exploitation and less emphasis on exploration could be extremely beneficial.

The suggested values for a_t, b_t, c_t and d_t are obtained by performing several pilot runs. In this study, we set $a_t = 20$, $b_t = 1$ and *maxpopsize* = 100. Thus, substituting these values into Eq. 3.3, we get

$$|\mathbf{P}_t| = \min\left\{20 + \left|\left\{\mathbf{x}_i \mid \mathbf{x}_i \in \mathbf{CP}_t \wedge \mathbf{x}_i \text{ is nondominated}\right\}\right|, 100\right\}. \tag{3.5}$$

In other words, the population size at generation t is 20 plus the number of nondominated solutions in the composite population if it is not larger than the pre-specified maximum population size. Otherwise, it is kept (truncated) equal to the maximum population size. Also, we set $c_t = 20$, $d_t = 0$ and *maxsoleval* = 100. Thus, substituting these values into Eq. 3.4, we get $|\mathbf{O}_t| = 20$, which means that the number of offspring created at each generation is small, but constant through the search process.

As the intent of this research is to introduce a novel strategy that addresses adaptive population sizing and conservative offspring generation in order to improve the efficiency and effectiveness of the search, the attempt to determine more appropriate (and perhaps more robust) values for a_t, b_t, c_t and d_t parameters is left for future study.

3.2.5. Crossover and Mutation Operations

The pairs of selected solutions in the reproduction process undergo crossover and mutation operations to produce offspring for the population at the next generation. The crossover operator exchanges information between selected solution pairs with a probability of occurrence p_c. The simulated binary crossover (referred to in the literature as SBX) operator introduced by Deb and Agrawal (1995) is performed in this algorithm.

This operator requires two parent solutions and creates two offspring, and it preserves the common interval schemata between the parent solutions in the offspring. Another interesting aspect of the SBX operator is that the absolute difference in offspring values is that of their parents.

After the crossover operation, the newly-obtained solutions undergo a mutation operation with a probability of occurrence p_m. The polynomial mutation operator introduced by Deb and Goyal (1996) is employed in which the probability distribution is polynomial. This operator is very similar to non-uniform mutation, but here the shape of the probability distributions is not dynamically changed over generations.

The interested reader is referred to the work of Herrera *et al.* (1998) or Deb (2001) for detailed discussions on different crossover and mutation operators for real-coded GAs.

3.2.6. Stopping Criterion

Different approaches have been used to stop the search process of EAs including those that consider the landscape of the response surface, the desired solutions quality, the specific number of solution evaluations and the required computation time. Designed for dealing with expensive MOPs, FPGA uses a new stopping criterion that considers the convergence speed towards the true Pareto optimal front. Here, when the number of nondominated solutions reaches the pre-specified maximum population size, and thereafter, no changes are made in the number of nondominated solutions within a certain number of solution evaluations[1], the search stops. For better understanding of the

[1] The expression "solution evaluations" could be replaced by "generations" if the MOEA has identical population size over generations.

suggested stopping criterion for expensive MOPs, a new convergence velocity measure is defined.

Definition 3.1: The Pareto production rate (*PPR*) is the rate at which a particular MOEA produces nondominated solutions per population and is calculated as

$$PPR_t = \frac{|\mathbf{NP}_t|}{|\mathbf{P}_t|}, \tag{3.6}$$

where $\mathbf{P}_t$ is the population at generation t, and $|\mathbf{NP}_t|$ denotes the number of nondominated solutions belonging to population $\mathbf{P}_t$.

When PPR_t reaches one (*i.e.*, all solutions in the population are nondominated) and it does not make any changes over a pre-specified number of solution evaluations implying no promising nondominated solutions are found within this period, the search stops.

This new stopping criterion has a few advantages over many other suggested stopping criteria, particularly when solving MOPs where each solution evaluation is computationally- and/or financially-expensive. First, it does not require the knowledge about the true Pareto optimal front of the problem under study. This is often the case when addressing real-world problems. If the approximate set of the true Pareto optimal front is not known, determination of sufficient number of solution evaluations for successful convergence is virtually impossible. Secondly, a sufficient number of solution evaluations for any problem is different depending on the number of decision variables, variables' domains, number of objectives and optimality characteristics. Therefore, for benchmarking and comparative analysis, the number of solution evaluations for each test problem is set to different values to allow for convergence to the true Pareto optimal

front. Moreover, some algorithms might converge faster to the true Pareto optimal front, even when an excessive number of generations is assigned. Finally, it is possible to evaluate this measure during the entire search process at any given generation thus providing valuable information about the convergence behavior of an algorithm.

It is important to note that this stopping criterion allows a measure to evaluate the capability of a multiobjective optimization algorithm to produce nondominated solutions at any given generation rather than a measure to evaluate the convergence of the obtained nondominated solutions to the true Pareto optimal front. Therefore, a careful monitoring of the algorithm during pilot runs is crucial to ensure that a sufficient number of generations (or solution evaluations) is assigned for successful convergence. Here, since the population size of FPGA is not fixed and is changing over the search, a variation of the *PPR* is employed in which the number of obtained nondominated solutions in terms of the total number of solution evaluations is calculated.

3.2.7. Screening Nondominated Solutions Set by Clustering

Since in most nondomination-based multiobjective problems the size of the Pareto optimal set becomes extremely large, some tools should be employed to prune it to manageable size for the decision-maker. A review of the literature on cluster analysis reveals that there are several methods available for this purpose. For example, Morse (1980) provides comprehensive review of different clustering methods including two general forms of direct and hierarchical clustering. The basic idea is to portray the nondominated set by a representative subset that reflects the characteristics of the main set without destroying attributes of the obtained curve. In general, cluster analysis

partitions a collection of N elements into P groups of relatively homogeneous elements, where $P < N$.

In this methodology, an average linkage hierarchical clustering is adopted to screen out the potentially large Pareto optimal set obtained at the end of the search process. The mechanism is that the two clusters with minimum average distance are combined together into a larger cluster. This process continues until the desired number of clusters is formed. Then, the nearest solution to the centroid of each cluster is selected and the remainders are removed.

3.3. Computational Complexity of FPGA

To determine the computational complexity of FPGA, consider the worst case complexity at generation t of the search process. The key operations of FPGA with respect to complexity include the new ranking strategy, fitness assignment, and crowding distance computation. The complexity of FPGA's ranking strategy that determines the nondominated solutions and dominated solutions is $O(mN_t\log N_t)$ for $m = 2$ and 3 and $O(mN_t\log N_t^{m-2})$ for $m \geq 4$. The complexity of the crowding distance computation performed for fitness assignment of the nondominated solutions is $O(mN_t\log(N_t))$. Sorting of the nondominated solutions based on their fitness assignments obtained from crowding distance needs $O(m\ N_t\log N_t)$ computations. Fitness assignment of dominated solutions requires $O(m\,N_t^2)$ computations. Thus, the overall complexity of FPGA is at most $O(m\,N_t^2)$. If the maximum population size of FPGA is the static population size N of most other MOEAs, the overall complexity of FPGA is $O(mN^2)$, which is no more than that of other popular MOEAs such as NSGA-II, SPEA2 and PAES.

CHAPTER 4: FPGA COMPUTATIONAL RESULTS

4.1. Introduction

The previous chapter describes a new multiobjective evolutionary algorithm approach, called fast Pareto genetic algorithm (FPGA), for expensive MOPs. In this chapter, we evaluate the performance of FPGA on a suite of published benchmark test problems having two objectives and no coupled constraints. In all test problems, the functions are to be minimized. These problems consider deterministic objective functions. The results of FPGA are also benchmarked against one of the state-of-the-art MOEAs – real-coded NSGA-II of Deb *et al.* (2002). It has been reported that NSGA-II outperforms most of its competitors including SPEA and PESA, and it competes closely with SPEA2 in terms of convergence to the true Pareto optimal front while maintaining the diversity (Deb *et al.*, 2002; Zitzler *et al.*, 2001; Erbas *et al.*, 2006). However, SPEA2 requires higher computational complexity of $O(mN^2 \log N)$ (Zitzler *et al.*, 2001) compared to that of NSGA-II, $O(mN^2)$, raising the question of whether the computationally-expensive fitness assignment strategy and truncation operator in SPEA2 pays off. Some studies report that there is no significant difference between the performance of SPEA2 and NSGA-II, although SPEA2 requires significantly higher computational time (Zitzler *et al.*, 2001; Deb *et al.*, 2005; Bui *et al.*, 2005; Erbas *et al.*, 2006).

4.2. Benchmark Test Problems

The suite of test problems consists of seven well-known benchmark problems. Table 4.1 summarizes the number of decision variables and their bounds, the true Pareto optimal front and optimality characteristics for the seven problems. The first test problem, referred to as FON (Fonseca and Fleming, 1993), has a nonconvex Pareto optimal front. The second problem, referred to as KUR (Kursawe, 1990), has three discontinuous Pareto optimal front regions, which are an isolated point, a concave region and a convex region. Problems three through seven are well-known ZDTs real-variable problems, except ZDT5, which is a discrete-variable problem designed for binary strings suggested by (Zitzler *et al.*, 2000). The test problems ZDT1 and ZDT2 have 30 decision variables each and the former has a convex Pareto optimal front and the latter has a concave Pareto optimal front. The 30-decision variable problem ZDT3 has five discontinuous Pareto optimal front regions. The 10-decision variable test problem ZDT4 is a multi-frontal (multi-modal) problem having a large number of local Pareto optimal fronts and a single global Pareto optimal front. The test problem ZDT6 has 10 decision variables and a nonconvex Pareto optimal front. Moreover, the density of solutions across its Pareto optimal front is non-uniform and the density towards the Pareto optimal front gets thin. Many researchers have used these problems as benchmarks for evaluating their proposed algorithms (*e.g.*, Deb *et al.*, 2002; Zitzler *et al.*, 2001).

Table 4.1: Benchmark test problems.

Test Problem	Number of Variables *n*	Variable Bounds	Objective Functions	Pareto Optimal Solutions	Optimality Characteristics
FON	3	[-4, 4]	$f_1(\mathbf{x}) = 1 - \exp\left(-\sum_{i=1}^{3}\left(x_i - \frac{1}{\sqrt{3}}\right)^2\right)$ $f_2(\mathbf{x}) = 1 - \exp\left(-\sum_{i=1}^{3}\left(x_i + \frac{1}{\sqrt{3}}\right)^2\right)$	$x_1 = x_2 = x_3$ $\in [-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}]$	Nonconvex
KUR	3	[-5, 5]	$f_1(\mathbf{x}) = \sum_{i=1}^{n-1}\left(-10\exp\left(-0.2\sqrt{x_i^2 + x_{i+1}^2}\right)\right)$ $f_2(\mathbf{x}) = \sum_{i=1}^{n}\left(\lvert x_i\rvert^{0.8} + 5\sin x_i^3\right)$	Refer to Deb (2001)	Nonconvex Discontinuous Non-uniformly spaced Isolated point
ZDT1	30	[0, 1]	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - \sqrt{x_1 / g(\mathbf{x})}\right]$ $g(\mathbf{x}) = 1 + 9(\sum_{i=2}^{n} x_i) \Big/ (n-1)$	$x_1 \in [0,1]$ $x_i = 0,\ i = 2,\ldots,n$	Convex
ZDT2	30	[0, 1]	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - \left(x_1 / g(\mathbf{x})\right)^2\right]$ $g(\mathbf{x}) = 1 + 9(\sum_{i=2}^{n} x_i) \Big/ (n-1)$	$x_1 \in [0,1]$ $x_i = 0,\ i = 2,\ldots,n$	Nonconvex

Table 4.1 (cont'd): Benchmark test problems.

Test Problem	Number of Variables *n*	Variable Bounds	Objective Functions	Pareto Optimal Solutions	Optimality Characteristics
ZDT3	30	[0, 1]	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - \sqrt{x_1/g(\mathbf{x})} - \frac{x_1}{g(\mathbf{x})}\sin(10\pi x_1)\right]$ $g(\mathbf{x}) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)\Big/(n-1)$	$x_1 \in [0,1]$ $x_i = 0,\ i = 2,\ldots,n$	Nonconvex Discontinuous Non-uniformly spaced
ZDT4	10	$x_1 \in [0,1]$ $x_i \in [-5,5]$, $i = 2,\ldots,n$	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - \sqrt{x_1/g(\mathbf{x})}\right]$ $g(\mathbf{x}) = 1 + 10(n-1) + \sum_{i=2}^{n}\left[x_i^2 - 10\cos(4\pi x_i)\right]$	$x_1 \in [0,1]$ $x_i = 0,\ i = 2,\ldots,n$	Nonconvex Multimodal
ZDT6	10	[0, 1]	$f_1(\mathbf{x}) = 1 - \exp(-4x_1)\sin^6(6\pi x_1)$ $f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - \left(f_1(x)/g(\mathbf{x})\right)^2\right]$ $g(\mathbf{x}) = 1 + 9\left[\sum_{i=2}^{n} x_i\Big/(n-1)\right]^{0.25}$	$x_1 \in [0,1]$ $x_i = 0,\ i = 2,\ldots,n$	Nonconvex Non-uniformly spaced

4.3. MOEA Parameter Settings

For both FPGA and NSGA-II, all of the parameter settings, except the maximum number of solution evaluations, are used according to the suggested values in the original study of Deb *et al.* (2002) as summarized in Table 4.2. In order to make better comparisons, the maximum population size for FPGA is set to the suggested population size used by Deb *et al.* (2002). The number of solution evaluations shown in Table 4.2 depends on the characteristics and complexity of the underlying problem. The number of solution evaluations is kept small to evaluate the performance of each algorithm more effectively for the expensive, real-world MOPs that may only allow a small number of solution evaluations.

Table 4.2: Parameter settings for FPGA and NSGA-II.

Algorithm Parameter	**FPGA and Real-Coded NSGA-II**						
Test Problem	FON	KUR	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
Number of Solution Evaluations	1500	2000	6500	7000	6000	10000	10000
Initial Population Size	100						
Maximum Population Size	100						
Crossover Probability	1.0						
Mutation Probability	1/n (where n is the number of variables)						
Crossover Type	Simulated Binary Crossover (η_c = 15)						
Mutation Type	Polynomial Mutation (η_m = 20)						
Selection Scheme	Binary Tournament						

4.4. Performance Metrics

In MOPs, there are three primary goals: 1) *fast convergence* to the true Pareto frontier solution set in the objective space, 2) *close proximity* to the true Pareto frontier solution set, and 3) *diversity and even dispersion* of the obtained nondominated solutions along the true Pareto optimal front. Many performance metrics have been introduced within the last decade (*e.g.*, Srinivas and Deb, 1994; Zitzler *et al.*, 1999; Van Veldhuizen and Lamont, 2000; Deb *et al.*, 2002; Collette and Siarry, 2005; Erbas *et al.*, 2006). Few performance metrics have been suggested to simultaneously consider the above goals. Most previous studies emphasize only the closeness and diversity measures. Fast convergence to optimal solutions for computationally-expensive MOPs is very important. This is especially the case in real-world problems where finding the optimal or even near-optimal solutions is often computationally-prohibitive.

In this study, four performance metrics are used to measure the convergence behavior and diversity of FPGA and NSGA-II, two of which are newly introduced. They are the diversity metric and the delineation metric. Two of the four metrics, delineation and hypervolume, are employed for simultaneous evaluation of closeness and diversity of the obtained solutions to gain a more thorough overall evaluation. For each test problem, each algorithm is run with 30 different seed values and the mean, standard deviation and 95% confidence interval are computed. The lower and upper bounds of the 95% confidence interval are calculated by $\bar{x} \pm t_{\alpha/2,n-1}\, s/\sqrt{n}$, where $\bar{x}$ is the sample mean, s is sample standard deviation, α is the significance level and is equal to 5% and n is the sample size and is equal to 30. Given the fact that in expensive MOPs the time required for solution evaluations significantly dominates the actual CPU time of any approach, no

attempt is made to measure the computation time needed to run each algorithm. Moreover, equality of the computational complexity of FPGA and NSGA-II indicates that there should be no appreciable difference between their computation times.

4.4.1. Distance from the Pareto Optimal Front

Deb *et al.* (2002) suggest the distance metric Υ, which evaluates the extent of convergence to a known Pareto optimal front. To calculate Υ, a set of H evenly-spaced solutions from the true Pareto optimal set in the objective space must be known. The set of H solutions should be large enough such that it reflects the true Pareto optimal front well. In this study, a set of 500 solutions from the true Pareto optimal frontier set is used for each of the seven test problems. The minimum Euclidean distance from each obtained nondominated solution to the H solutions is calculated and the average of these distances is used as the distance metric Υ. It is important to note that all solutions obtained by an algorithm including those that are dominated are considered for the calculation of this metric. The distance metric Υ returns a value in the range of $[0, \infty)$. The smaller the value of this metric, the closer the solutions are to the true Pareto optimal front. Ideally, this metric is zero, where each obtained solution falls exactly on one of the H solutions. However, the likelihood of this happening is rare.

4.4.2. Diversity of Nondominated Solutions

We define the diversity metric Δ to evaluate the extent of dispersion of the obtained nondominated solutions in the objective space. Here, the goal is to obtain a set of nondominated solutions that are both widely- and uniformly-distributed along the

Pareto optimal front at the end of the search. To compute the diversity metric Δ, the Euclidean distance d_i between consecutive nondominated solutions is calculated in the objective space, as shown in Figure 4.1, where $i = 1, ..., |\mathbf{NP}_t|-1$ and $|\mathbf{NP}_t|$ is the number of nondominated solutions at the end of the search. Then, the standard deviation of these distances σ_d is calculated representing the degree of non-uniformity of the nondominated solutions. The minimum Euclidean distance of the two extreme Pareto solutions of the true Pareto optimal set from the nondominated solutions, denoted by d_p and d_q, is calculated. Note that the distances d_p and d_q are the distances from the closest nondominated solutions, not necessarily the endpoints of nondominated solutions, to the two extreme Pareto solutions. Finally, the diversity of the set of nondominated solutions is

$$\Delta(\mathbf{NP}_t) = d_p + d_q + \sqrt{\frac{1}{|\mathbf{NP}_t|-1}\sum_{i=1}^{|\mathbf{NP}_t|-1}\left(d_i - \bar{d}\right)^2}\,. \tag{4.1}$$

The first two terms of Eq. 4.1 measure the spread of the nondominated solutions and the last term measures their uniform spacing.

The diversity metric Δ returns a value in the range of $[0, \infty)$. Small values of this metric mean the nondominated solutions are well spread and distributed. Ideally, this metric takes a value of zero. This happens when each end nondominated solution falls exactly on the extreme Pareto optimal solutions and all Euclidean distance d_i between consecutive nondominated solutions are equal in the objective space. However, similar to the distance metric Υ, that rarely happens.

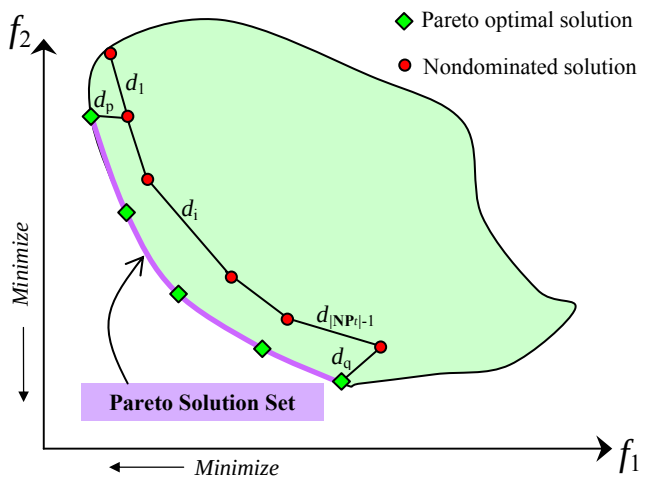

Figure 4.1. Diversity metric Δ.

4.4.3. Delineation of Pareto Optimal Front

The delineation metric Φ is introduced to evaluate simultaneously the extent of both convergence and diversity to the true Pareto optimal front. A goal of this research is to propose a MOEA that identifies a set of solutions that well represent the Pareto optimal set. The idea behind the delineation metric is how well each solution on the Pareto optimal front is represented by the obtained nondominated solutions. To calculate Φ, a large set of H evenly-spaced solutions from the Pareto optimal set of each test problem that well reflects the true Pareto optimal front must be known. The same set of H solutions used in calculating the distance metric ϒ is used here. The minimum Euclidean distance from each Pareto optimal solution to the obtained solutions l_i is calculated, and the average of these distances is used as the delineation metric Φ, *i.e.*,

$$\Phi(\mathbf{P}_t) = \frac{1}{H}\sum_{i=1}^{H} l_i \,. \tag{4.2}$$

Figure 4.2 shows the calculation procedure of this metric. It is important to note that all solutions obtained by an algorithm including those that are dominated are also considered for the calculation of this metric. The delineation metric Φ returns a value in the range of [0, ∞). The smaller the value of this metric, the better the Pareto optimal solutions are represented by the obtained solutions. Ideally, this metric is zero, where population size is adequately large (≥ H) and each H selected Pareto solution is exactly overlapped by one of the nondominated solutions. The likelihood of this happening is zero, especially when population size is smaller than H, which is the case in most applications.

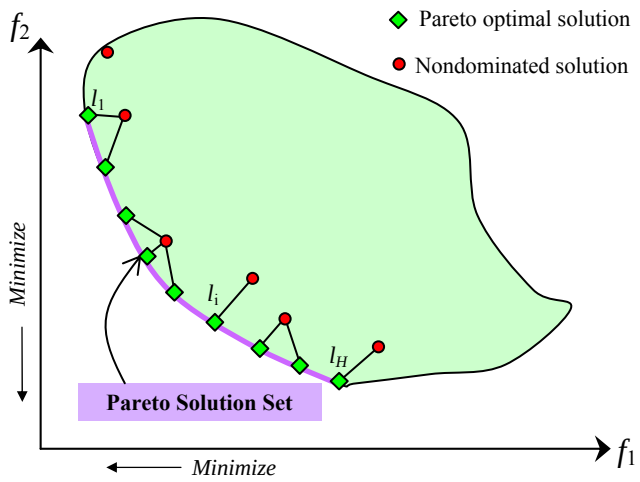

Figure 4.2. Delineation metric Φ.

4.4.4. Hypervolume

The hypervolume metric *HV*, originally suggested by Zitzler and Thiele (1999), calculates the volume of the objective space dominated by the nondominated solutions having the reference point **R**. Mathematically stated, the function $HV(\mathbf{NP}_t)$ calculates the volume enclosed by the union of the hypercubes h_i (i = 1, …, $|\mathbf{NP}_t|$), where each hypercube h_i is built with the reference point **R** and solution $\mathbf{x}_i$, that is,

$HV(\mathbf{NP}_t) = \text{volume}\left(\bigcup_{i=1}^{|\mathbf{NP}_t|} h_i\right)$. In the biobjective case, each hypercube is represented by a rectangle with vertices **R** and $\mathbf{x}_i$. This measure considers simultaneously the extent of convergence and diversity to a known Pareto optimal front. The goal of this measure is to identify the proportion of the volume enclosed by the reference point and Pareto optimal front covered by the nondominated solutions obtained at the end of the search. To be consistent with other performance metrics used in this study (*i.e.*, the smaller value of the metric, the better), a modification of the hypervolume metric is employed. We call the modified *HV* metric the hypervolume ratio *HVR* metric. The *HVR* represents the proportion of the volume enclosed by reference point and true Pareto optimal front that is not covered by the nondominated solutions, and is give by

$$HVR(\mathbf{NP}_t) = 1 - \frac{HV(\mathbf{NP}_t)}{HV(\mathbf{PF})}, \tag{4.3}$$

where **PF** is the set of solutions on the true Pareto optimal front. The hypervolume ratio *HVR* returns a value in the range of [0, 1]. The smaller the value of this metric, the less portion of the volume is not covered by nondominated solutions. Ideally, as in delineation metric this metric is zero, where population size is adequately large ($\geq H$) and each nondominated solutions falls on one of the *H* Pareto solutions.

4.5. FPGA Computational Results

In this section, the computational results of FPGA and the real-coded NSGA-II are presented. We first illustrate the suggested stopping criterion for expensive MOPs. Discussion of the computational results is then given followed by explanation of the effect of adaptive population sizing strategy employed in FPGA.

4.5.1. Termination of the Search

Recall that according to the suggested stopping criterion for expensive MOPs, the search terminates when the number of nondominated solutions reaches the pre-specified maximum population size, and no changes are made in the number of nondominated solutions within a certain number of solution evaluations thereafter. In order to better evaluate the performance of these algorithms in terms of the velocity measure *PPR*, sample simulation results on KUR and ZDT6 are shown in Figure 4.3 and Figure 4.4, respectively. The number of nondominated solutions fluctuates (slightly on KUR and greatly on ZDT6) during the early and middle stages of the search. However, after a point, no considerable changes occur resulting in termination of the search. Although the number of nondominated solutions is gradually increases through the search, it does not have monotonically increasing behavior. It decreases at some points when a promising nondominated solution in the objective space is found, which converts some of the nondominated solutions in the previous generation into dominated solutions in the current generation. It can be seen that FPGA is capable of producing nondominated solutions faster than NSGA-II and reaches the maximum population size in a significantly fewer number of solution evaluations. This unique property of fast convergence makes FPGA an appropriate approach for dealing with MOPs that are computationally- and/or financially-expensive.

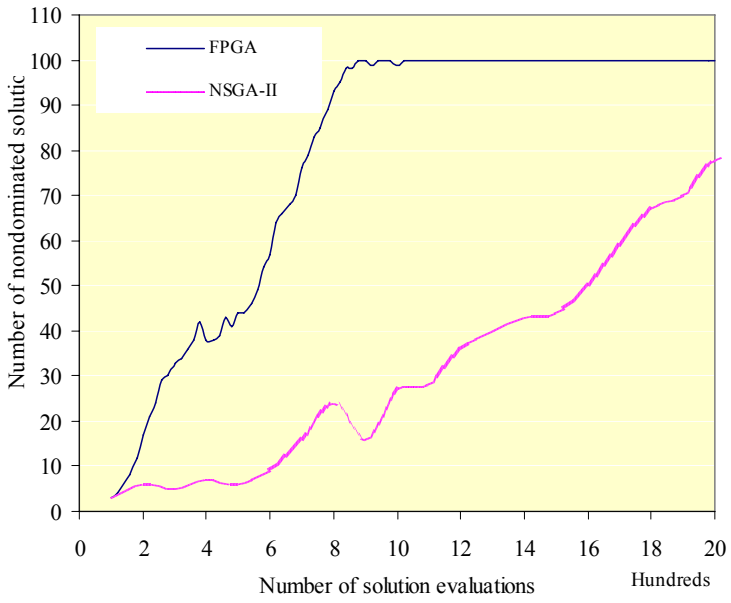

Figure 4.3. The velocity measure *PPR* on KUR.

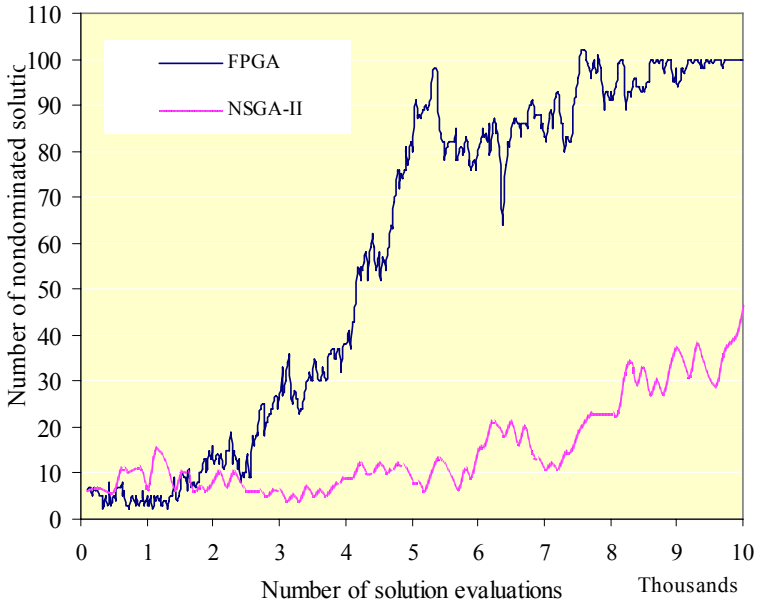

Figure 4.4. The velocity measure *PPR* on ZDT6.

4.5.2. Discussion of the Results

Table 4.3 and Table 4.4 show the output statistics including mean, standard deviation and 95% confidence interval (CI) of the four aforementioned performance metrics obtained from generating 30 random replications for each test problem using FPGA and NSGA-II. The distance ϒ and diversity Δ metrics are shown in Table 4.3 and the delineation metric Φ and hypervolume ratio *HVR* metric are shown in Table 4.4. Recall that lower values are preferred for all four metrics. In both Table 4.3 and Table 4.4, the first column shows the test problem and the second column presents the MOEA.

The results shown in Table 4.3 indicate that FPGA significantly outperforms NSGA-II with respect to the convergence to the Pareto optimal front. There is no overlap between the confidence intervals of the distance metric ϒ for FPGA and NSGA-II in all problems. Compared with FPGA, NSGA-II exhibits poor convergence in the ZDT4 and ZDT6 test problems. Both MOEAs have acceptable standard deviations for ϒ-metric on most problems. An exception occurs on ZDT4, where NSGA-II has very high standard deviation for ϒ-metric. To illustrate the convergence behavior of FPGA and NSGA-II, the sample obtained populations at the end of the search together with the Pareto optimal front for KUR, ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6 are shown in Figure 4.5, Figure 4.6, Figure 4.7, Figure 4.8, Figure 4.9 and Figure 4.10, respectively. These figures show the superiority of FPGA over NSGA-II in rapidly converging to the true Pareto optimal solution set while preserving a diverse set of nondominated solutions. Within the given number of solution evaluations, FPGA obtains the population of nondominated solutions while a significant proportion of solutions in NSGA-II are dominated solutions,

indicating that FPGA has a much faster convergence. It is interesting to note that all obtained nondominated solutions yielded by NSGA-II at the end of the search are dominated by the nondominated solutions of FPGA in most problems. The favorable performance of FPGA is most likely due to high elitism intensity and regulation operator employment. These settings help to improve search space exploitation and to save a considerable number of solution evaluations for further exploitation at later generations.

Table 4.3. Mean, standard deviation and 95% confidence interval of distance and diversity metrics for FPGA and NSGA-II over the 30 random replications.

Test Problem	Algorithm	Distance ϒ			Diversity Δ		
		Avg.	Std. Dev.	95% CI	Avg.	Std. Dev.	95% CI
FON	FPGA	0.0048	0.0007	[0.0045, 0.0050]	0.0765	0.0251	[0.0672, 0.0859]
	NSGA-II	0.0077	0.0014	[0.0072, 0.0083]	0.1324	0.0220	[0.1242, 0.1406]
KUR	FPGA	0.0048	0.0009	[0.0044, 0.0051]	0.0705	0.0179	[0.0638, 0.0771]
	NSGA-II	0.0086	0.0012	[0.0081, 0.0090]	0.1209	0.0710	[0.0945, 0.1474]
ZDT1	FPGA	0.0210	0.0110	[0.0169, 0.0251]	0.0769	0.0296	[0.0659, 0.0879]
	NSGA-II	0.0659	0.0128	[0.0612, 0.0707]	0.1324	0.0220	[0.1242, 0.1406]
ZDT2	FPGA	0.0075	0.0044	[0.0059, 0.0092]	0.4436	0.3415	[0.3163, 0.5709]
	NSGA-II	0.0933	0.0241	[0.0844, 0.1023]	0.3263	0.0858	[0.2943, 0.3583]
ZDT3	FPGA	0.0200	0.0092	[0.0166, 0.0235]	0.2017	0.1036	[0.1631, 0.2403]
	NSGA-II	0.0297	0.0091	[0.0263, 0.0331]	0.1968	0.0233	[0.1882, 0.2055]
ZDT4	FPGA	0.0332	0.0262	[0.0234, 0.0430]	0.3812	0.1804	[0.3140, 0.4485]
	NSGA-II	0.7677	0.3414	[0.6404, 0.8950]	1.5111	0.5797	[1.2950, 1.7273]
ZDT6	FPGA	0.0445	0.0082	[0.0414, 0.0475]	0.1393	0.0256	[0.1297, 0.1488]
	NSGA-II	0.2647	0.0380	[0.2506, 0.2789]	0.7239	0.1063	[0.6843, 0.7636]

Table 4.4. Mean, standard deviation and 95% confidence interval of delineation Φ and hypervolume ratio *HVR* metrics for FPGA and NSGA-II over the 30 random replications.

Test Problem	Algorithm	Delineation Φ			Hypervolume Ratio *HVR*		
		Avg.	Std. Dev.	95% CI	Avg.	Std. Dev.	95% CI
FON	FPGA	0.0087	0.0019	[0.0080, 0.0094]	0.0293	0.0046	[0.0276, 0.0310]
	NSGA-II	0.0108	0.0015	[0.0102, 0.0113]	0.0411	0.0057	[0.0390, 0.0433]
KUR	FPGA	0.0056	0.0008	[0.0053, 0.0059]	0.0101	0.0059	[0.0079, 0.0123]
	NSGA-II	0.0086	0.0012	[0.0081, 0.0090]	0.0148	0.0078	[0.0119, 0.0177]
ZDT1	FPGA	0.0208	0.0097	[0.0172, 0.0244]	0.0443	0.0198	[0.0369, 0.0517]
	NSGA-II	0.0599	0.0111	[0.0557, 0.0640]	0.1259	0.0226	[0.1175, 0.1343]
ZDT2	FPGA	0.1050	0.1234	[0.0590, 0.1510]	0.1653	0.1603	[0.1055, 0.2251]
	NSGA-II	0.0899	0.0232	[0.0812, 0.0985]	0.3087	0.0679	[0.2833, 0.3340]
ZDT3	FPGA	0.0269	0.0255	[0.0174, 0.0364]	0.0850	0.0345	[0.0722, 0.0979]
	NSGA-II	0.0286	0.0084	[0.0255, 0.0318]	0.1086	0.0252	[0.0992, 0.1180]
ZDT4	FPGA	0.0701	0.0457	[0.0531, 0.0872]	0.0910	0.0479	[0.0732, 0.1089]
	NSGA-II	0.6557	0.3128	[0.5391, 0.7724]	0.8173	0.2123	[0.7381, 0.8964]
ZDT6	FPGA	0.0415	0.0079	[0.0385, 0.0444]	0.1083	0.0190	[0.1012, 0.1154]
	NSGA-II	0.2538	0.0396	[0.2391, 0.2686]	0.5731	0.0690	[0.5473, 0.5988]

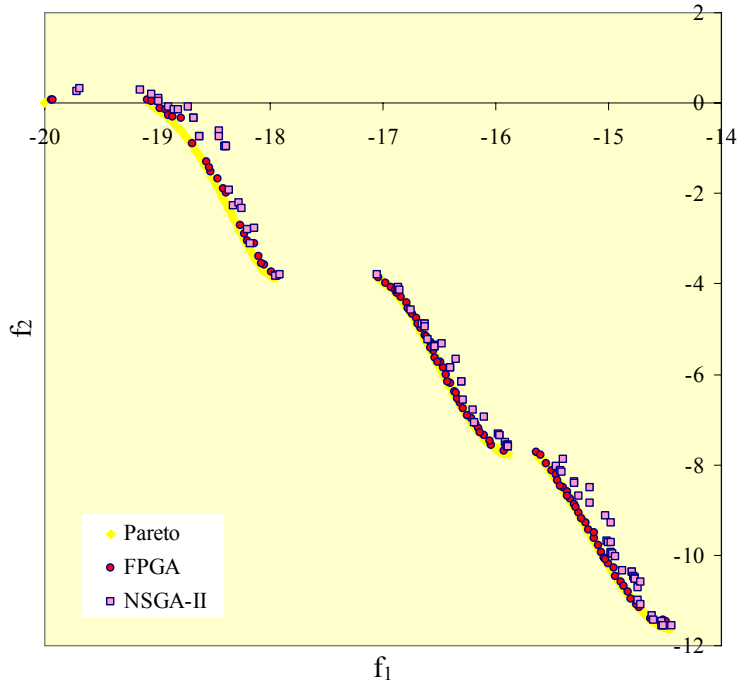

Figure 4.5. The populations with FPGA and NSGA-II on KUR.

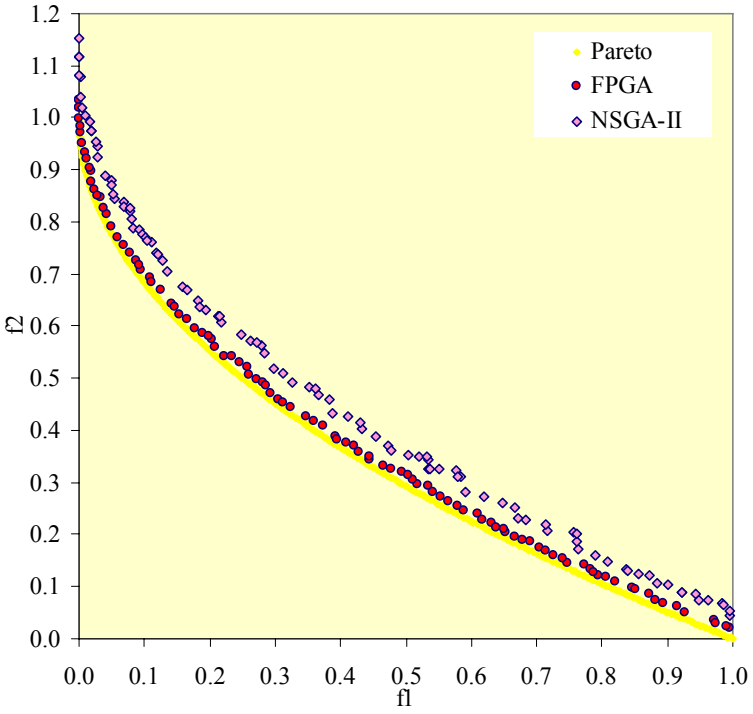

Figure 4.6. The populations with FPGA and NSGA-II on ZDT1.

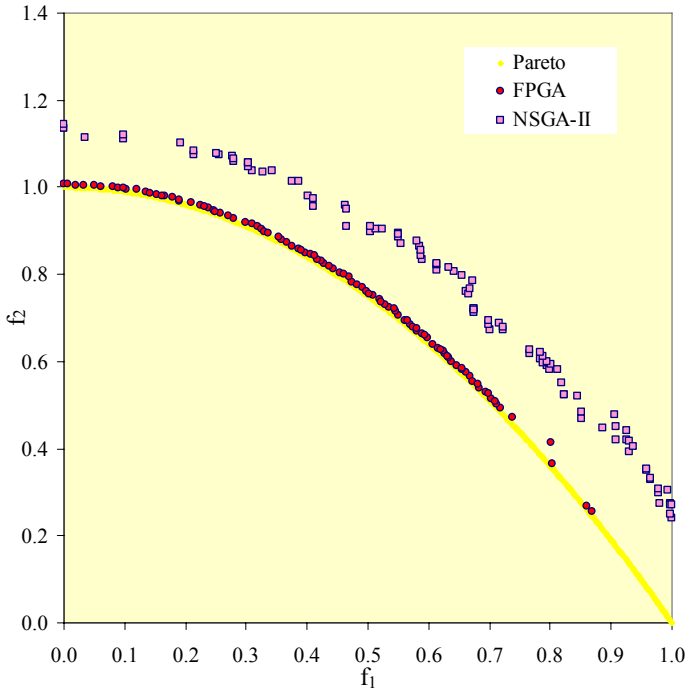

Figure 4.7. The populations with FPGA and NSGA-II on ZDT2.

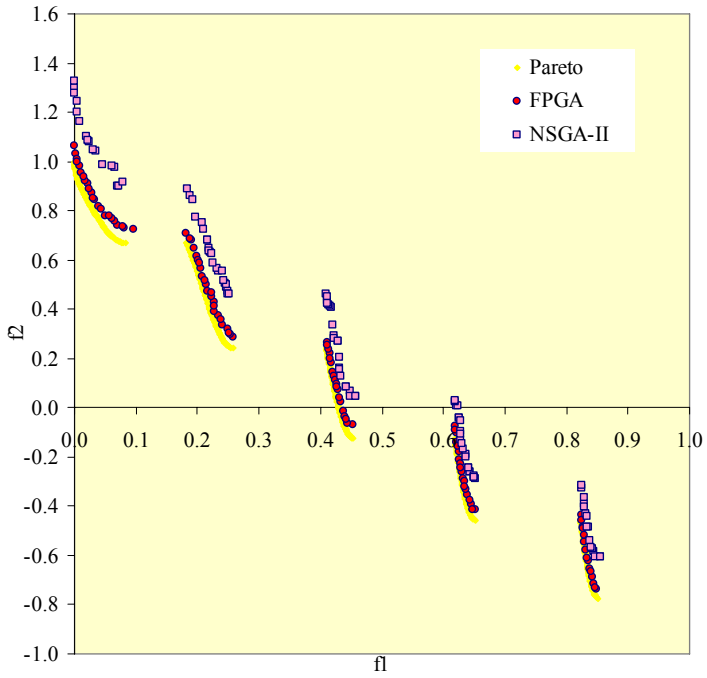

Figure 4.8. The populations with FPGA and NSGA-II on ZDT3.

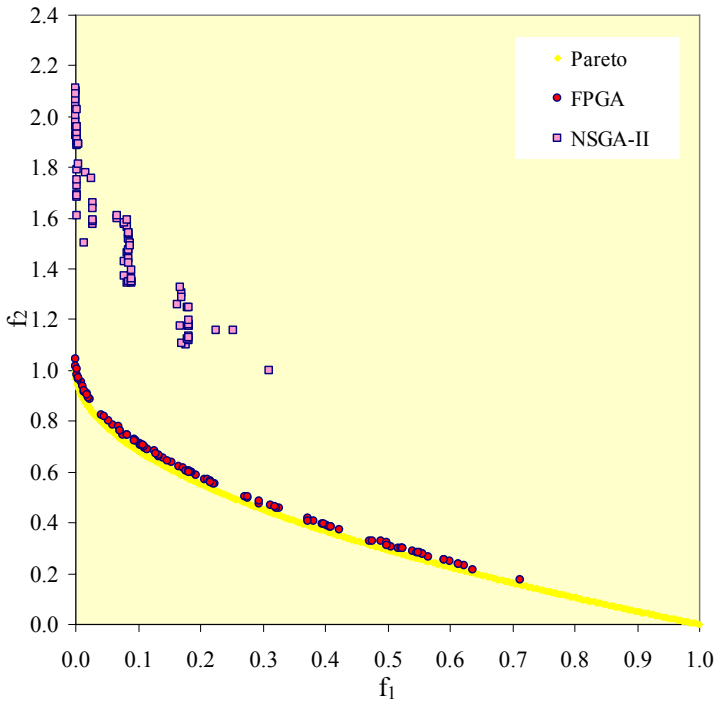

Figure 4.9. The populations with FPGA and NSGA-II on ZDT4.

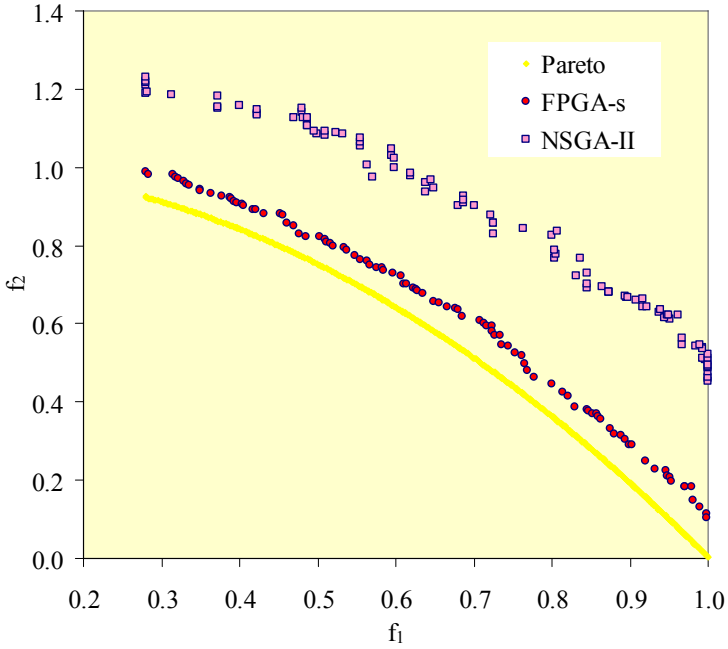

Figure 4.10. The populations with FPGA and NSGA-II on ZDT6.

Table 4.3 shows that FPGA has significantly better performance than NSGA-II in terms of the diversity metric Δ for most problems. There is no overlap between the confidence intervals of the Δ-metric for FPGA and NSGA-II in FON, KUR, ZDT1, ZDT4 and ZDT6 problems. NSGA-II performs only slightly better than FPGA on ZDT2 and ZDT3 with respect to this metric. It is interesting to note that FPGA has a better Δ-metric than NSGA-II in many replications on ZDT2 and ZDT3, but its performance is actually poorer in a few replications. Figure 4.11 shows the sample obtained population with FPGA having poor diversity together with NSGA-II and the true Pareto optimal front for ZDT3. Here, the top three disconnected Pareto front regions are covered quite well by obtained solutions with FPGA, whereas no solution is found in the other two Pareto front regions resulting in large value for distance d_q and consequently poor Δ-metric. The reason for this happening is also most likely due to the employment of high elitism intensity resulting in biasedness towards some particular regions of the Pareto front in few replications. This undesired biasedness with FPGA is also realized on ZDT2, ZDT3 and ZDT4 problems having relatively large standard deviation. NSGA-II has good standard deviations for Δ-metric on all problems, except in ZDT4, where it has very high standard deviation.

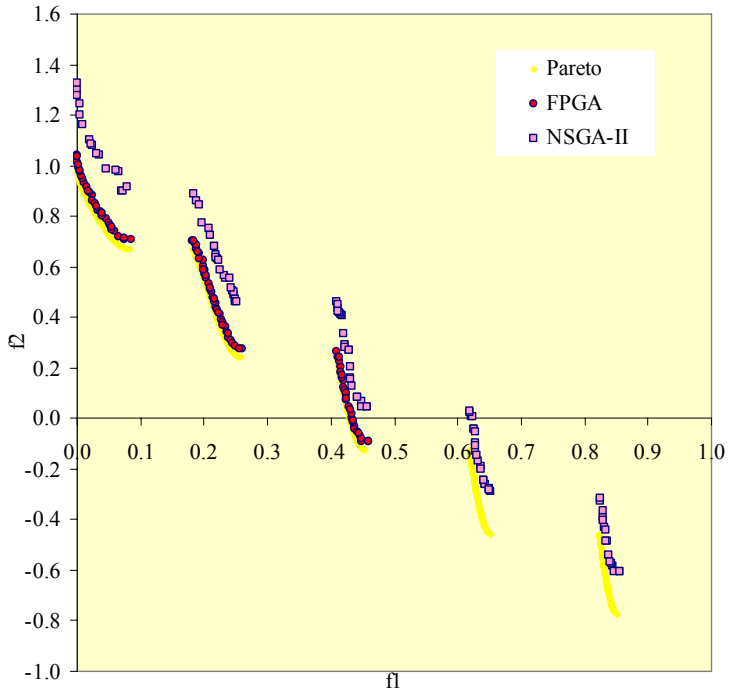

Figure 4.11. The populations with FPGA having poor diversity in few replications and NSGA-II on ZDT3.

Table 4.4 shows that FPGA has better performance than NSGA-II in terms of the delineation metric Φ for most problems. There is no overlap between the confidence intervals of the Φ-metric for FPGA and NSGA-II in FON, KUR, ZDT1, ZDT4 and ZDT6 problems. FPGA has slightly better mean performance than NSGA-II on ZDT3, but there is a considerable overlap between their confidence intervals. On the other hand, NSGA-II performs just slightly better than FPGA on ZDT3, but there is a considerable overlap between their confidence intervals. The standard deviations of the Φ-metric across all problems for both MOEAs are small, except for FPGA on ZDT2 and ZDT3 (due to the poor diversity in a few replications) and for NSGA-II on ZDT4.

For the hypervolume ratio *HVR* metric the reference point **R** is set at (1, 1.1) for all test problems, except for KUR where it is set at (-14.3, 0.1). For each test problem, the reference point is determined as a point with a little higher than the maximum value of optimal Pareto solution set for each objective. However, if in any test problem an objective is equal to one of the variables, the maximum value of this variable is taken since the value of this objective never exceeds the maximum value of the variable. The results shown in Table 4.4 indicate that FPGA outperforms NSGA-II with respect to the hypervolume ratio *HVR* measure. There is no overlap between the confidence intervals of the *HVR*-metric for FPGA and NSGA-II in all problems, except in KUR where there is a little bit overlap. It is interesting to note that although NSGA-II has better mean performance than FPGA on ZDT2, and there is considerable overlap between their confidence intervals on ZDT3 with respect to delineation metric Φ, FPGA outperforms NSGA-II with respect to the *HVR*-metric. Regarding the obtained results, it is implied that although nondominated solutions with FPGA in few replications do not represent the Pareto fronts of ZDT2 and ZDT3 pretty well, they dominate a considerable portion of the hypervolumes enclosed by Pareto fronts and reference point **R**. The standard deviations of the *HVR*-metric for NSGA-II are small on all problems, except in ZDT4.

4.5.3. A Discussion on FPGA Population Regulation

The regulation operator employed in FPGA improves its performance for all three goals: 1) fast convergence, 2) proximity to the Pareto optimal front, and 3) diversity maintenance. This operator monitors the population and adjusts the population size accordingly. When the number of nondominated solutions increases (or decreases) at any generation, an increase (or a decrease) in the population size is triggered and the

population size becomes 20 plus the number of nondominated solutions in the composite population. This process continues until the population size reaches the pre-specified maximum population size when 80% of the population size is populated with nondominated solutions. Then, the population size is kept fixed at maximum population size and the more diverse nondominated solutions, which reflect the Pareto optimal front, are preserved using the distance crowding operator if their number exceeds the maximum population size. This operator balances the proportion of nondominated solutions in the population by adjusting the population size adaptively during the search process. This dynamic adjustment enhances FPGA's convergence behavior and maintains diversity in larger populations at later generations.

Figure 4.12 shows the number of nondominated solutions, population size, *PPR* and the number of solution evaluations at each generation for FPGA within the search process on the 30-variable ZDT6 problem. We multiply *PPR* by 100 so that the same scale for the *y*-axis can be used for better illustration. The initial population size and the initial number of solution evaluations are kept at 100 to make sure that FPGA and NSGA-II both start from identical initial populations. After the initial generation, the regulation operator is invoked and adjusts the population size. As mentioned earlier, in this study, the number of solution evaluations at each generation, except the initial generation, is 20. This suggests that an evolving population size with small number of solution evaluations at each generation ensures the algorithm's search in the early and middle stages is performed to conserve solution evaluations for more search space exploitation in later generations. Figure 4.12 shows that, after about 2,700 solution evaluations, consistently more than 20 nondominated solutions are obtained resulting in

PPR of more than 0.50. This high level of elitism intensity puts more pressure on the search to converge faster towards promising regions, requiring fewer number of solution evaluations.

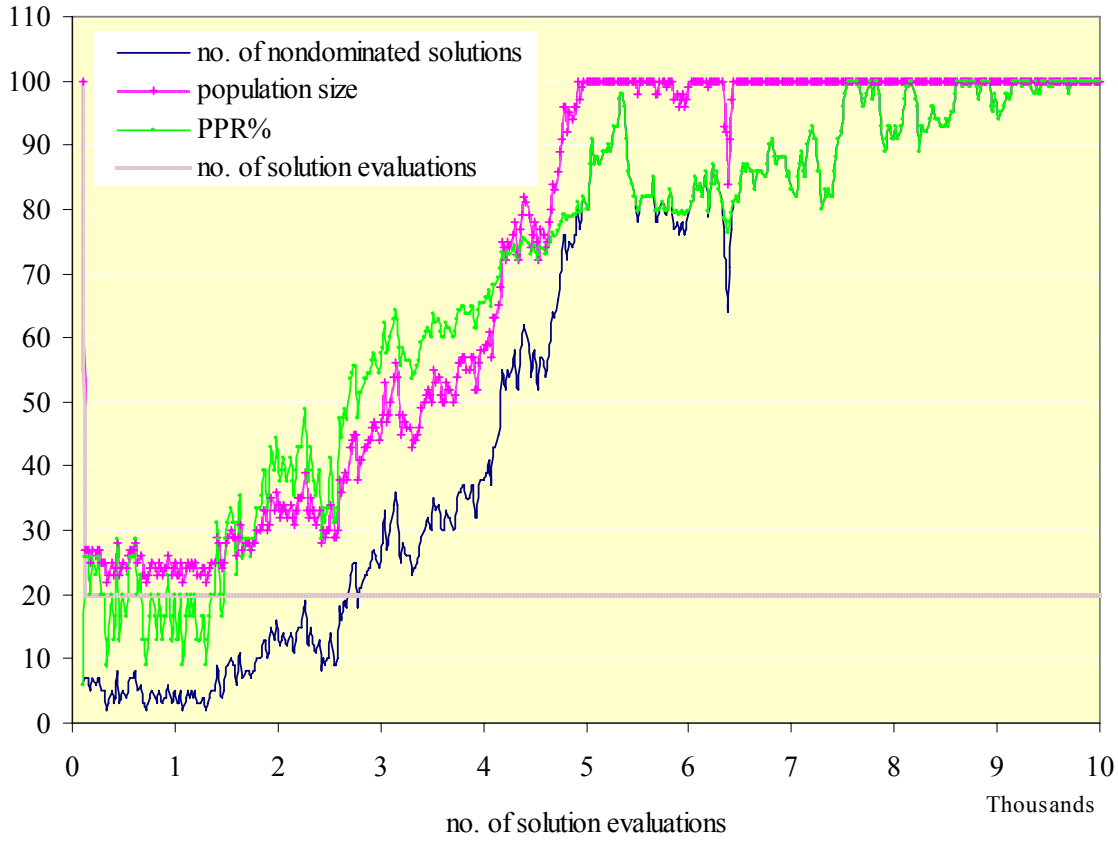

Figure 4.12. Population regulation behavior of FPGA on ZDT6.

CHAPTER 5: PROPOSED METHODOLOGY FOR STOCHASTIC ENVIRONMENTS

5.1. Introduction

In CHAPTER 4, validation and benchmarking of the proposed MOEA methodology, FPGA, in deterministic problem environments is accomplished. Originally designed for solving deterministic MOPs, FPGA requires being equipped with some stochastic procedures to be able to deal with MOPs with stochastic and noisy objective functions. This chapter demonstrates some modifications and enhancements made to the FPGA to enable it to better discriminate among the competing solutions in stochastic problem environments. The modified algorithm is called stochastic Pareto genetic algorithm (SPGA).

5.2. Redefinition of Solution Dominance in Multiobjective Stochastic Environments

CHAPTER 1 discusses the concept of dominance in deterministic problem environments. Recall that in a deterministic problem domain, solution **A** *strictly dominates* (is better than) solution **B** if $f_i(\mathbf{A})$ is less than $f_i(\mathbf{B})$ for each objective function *i.* The strict dominance definition must be modified for multiobjective stochastic problem environments in which the objective functions do not take on certain values but they are described with the expected values and variances (or half-widths). This uncertainty typically results from either the randomness effect involved in the simulation modeling or incomplete knowledge about the underlying optimization problem. Given the fact that in

the stochastic simulation models, objective functions are random and take on uncertain values, new definitions to compare two different solutions are proposed.

In the simulation context, it is a reasonable assumption that the objective values of solutions are approximately normally-distributed. Suppose that $\bar{f}_i(\mathbf{A})$, $s_{\mathbf{A}}^2$ and $\bar{f}_i(\mathbf{B})$, $s_{\mathbf{B}}^2$ are the expected values and variances of each objective function i for two solutions **A** and **B**, respectively. The objective function expected values and variances are calculated after a number of function evaluations n. Half-widths of solutions **A** and **B** are calculated by

$$hw_i(\mathbf{A}) = t_{1-\alpha/2,n-1}\frac{s_{\mathbf{A}}}{\sqrt{n}} \text{ and } hw_i(\mathbf{B}) = t_{1-\alpha/2,n-1}\frac{s_{\mathbf{B}}}{\sqrt{n}}. \quad (5.1)$$

where α is the significance level ($0 \le \alpha \le 1$) and parameter $t_{1-\alpha/2,n-1}$ is the critical value for t-distribution based on n-1 degrees of freedom. Now, it is assumed that each objective function f_i has truncated normal distribution and is represented by its confidence interval $[f_i - hw_i, f_i + hw_i]$, where $f_i - hw_i$ and $f_i + hw_i$ are the lower and upper bounds of the interval at significance level α, respectively.

Definition 5.1: Solution **A** *probabilistically dominates* solution **B** with a probability of $\prod_{i=1}^{m} P(f_i(\mathbf{A}) < f_i(\mathbf{B}))$ if $f_i(\mathbf{A}) - hw_i(\mathbf{A}) < f_i(\mathbf{B}) + hw_i(\mathbf{B})$ for each objective function i ($i \in \{1, \ldots, m\}$).

In this case, due to the uncertainty surrounding the objective function values, it is not certain that solution **A** strictly dominates solution **B**. As a result, the strict dominance

definition must be modified to account for this uncertainty. Further, in stochastic environments, it is necessary to know if there is a significant difference between two solutions. The following revised definition is proposed.

Definition 5.2: Solution **A** *significantly dominates* (is better than) solution **B** with a confidence level of about $(1 - m\alpha)$ if $f_i(\mathbf{A}) + hw_i(\mathbf{A}) < f_i(\mathbf{B}) - hw_i(\mathbf{B})$ for each objective function i ($i \in \{1, \ldots, m\}$).

If two solutions **A** and **B** with their corresponding confidence intervals are compared, three different cases can occur for calculating the probability that solution **A** dominates solution **B**, *i.e.*, $P(\mathbf{A} \succ \mathbf{B})$. First, solution **A** does not dominate solution **B** when at least one lower bound of the solution **A** confidence interval is larger than the corresponding upper bound of solution **B**. Second, solution **A** significantly dominates solution **B** when all upper bounds of the solution **A** confidence interval are less than the corresponding upper bound of solution **B**. In the third case, solution **A** probabilistically dominates solution **B** with a certain probability when all lower bounds of the solution **A** confidence intervals are less than the corresponding upper bounds of solution **B**. Therefore, the probability that solution **A** dominates solution **B** is given by

$$P(\mathbf{A} \succ \mathbf{B}) = \begin{cases} 0, & \text{if } \exists i : f_i(\mathbf{A}) - hw_i(\mathbf{A}) > f_i(\mathbf{B}) + hw_i(\mathbf{B}), \\ 1, & \text{if } \forall i : f_i(\mathbf{A}) + hw_i(\mathbf{A}) < f_i(\mathbf{B}) - hw_i(\mathbf{B}), \\ \prod_{i=1}^{m} P\left(f_i(\mathbf{A}) < f_i(\mathbf{B})\right), & \text{otherwise.} \end{cases} \tag{5.2}$$

Now, regarding that each objective function f_i follows a normal distribution with a known mean and variance, the question is how to calculate the probability $P\left(f_i(\mathbf{A}) < f_i(\mathbf{B})\right)$. If x and y are independent random variables, it can be proved that

$$P(x < y) = \int_{-\infty}^{\infty} f_x(t) F_y(t) d_t \,, \tag{5.3}$$

where $f_x(t)$ and $F_y(t)$ are probability density function of variable x and cumulative density function of variable y, respectively.

According to Eq. 5.3, we get

$$P(x < y) = \int_{-\infty}^{\infty} \left(\frac{1}{\sigma_x\sqrt{2\pi}} e^{-\frac{(x-\mu_x)^2}{2\sigma_x^2}} \int_{-\infty}^{x} \frac{1}{\sigma_y\sqrt{2\pi}} e^{-\frac{(y-\mu_y)^2}{2\sigma_y^2}} d_y\right) d_x \,. \tag{5.4}$$

It is realized that Eq. 5.4 is very complicated to integrate directly, and it does not have a closed-form expression. Therefore, an alternative approach is suggested knowing that the difference between two independent normal distributions is also normal distribution.

Theorem 5.1: If x and y are independent normal random variables with means μ_x and μ_y ($\mu_x < \mu_y$), and variances σ_x^2 and σ_y^2, the probability

$$P(x < y) = 1 - Q\left(\frac{\mu_y - \mu_x}{\sqrt{\sigma_x^2 + \sigma_y^2}}\right), \tag{5.5}$$

where the Gaussian error integral $Q(x) = 1 - \Phi(x) = 1/\sqrt{2\pi} \int_x^{\infty} e^{-t^2}$.

Proof: If x and y are independent normal random variables with means μ_x and μ_y and variances σ_x^2 and σ_y^2, the probability of x being less than y is $P(x < y) = P(0 < y - x)$.

Now, assuming $\mu_x < \mu_y$, the change $t = y - x$ results in $P(x < y) = P(0 < t)$, where t is a normal random variable with mean $\mu_t = \mu_y - \mu_x$ and variance $\sigma_t^2 = \sigma_x^2 + \sigma_y^2$, as shown in Figure 5.2.

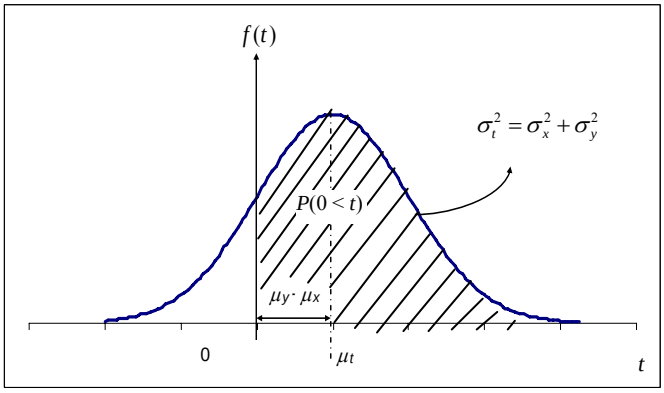

Figure 5.1. Plot of normally-distributed random variable t.

Now, the probability of $P(x < y) = P(0 < t)$ is

$$P(0 < t) = Q\left(\frac{0 - \mu_t}{\sigma_t}\right) = Q\left(\frac{-(\mu_y - \mu_x)}{\sqrt{\sigma_x^2 + \sigma_y^2}}\right).$$

Since $Q(-x) = 1 - Q(-x)$, then

$$P(0 < t) = 1 - Q\left(\frac{\mu_y - \mu_x}{\sqrt{\sigma_x^2 + \sigma_y^2}}\right).$$

□

The integral described for $Q(x)$ does not have a closed-form expression. However, an excellent closed-form approximation is suggested by Borjesson and Sundberg (1979) to estimate $Q(x)$ with an acceptable error. The next section describes how this probability is calculated and how it can be employed to improve the concepts of the stochastic dominance and significant dominance in the stochastic problem domain.

It is interesting to note that although two new definitions for dominance have been suggested, it is still difficult to discriminate which solutions should be considered as nondominated at any generation. The following definition helps better identification of nondominated solutions.

Definition 5.3: Solution **A** *stochastically dominates* (is better than) solution **B** if $\overline{f}_i(\mathbf{A})$ is less than $\overline{f}_i(\mathbf{B})$ for each objective function i ($i \in \{1, \ldots, m\}$).

It is clear that if solution **A** stochastically dominates solution **B**, denoted by $\mathbf{A} \succ \mathbf{B}$, $P(\mathbf{A} \succ \mathbf{B})$ is larger than $P(\mathbf{B} \succ \mathbf{A})$. This implies that the expectation that solution **A** is a nondominated solution in any given generation is higher that of solution **B**.

5.3. Noise

Noise is introduced in the objective space as

$$f_i'(\mathbf{x}) = f_i(\mathbf{x}) + s_i N(0,1), \tag{5.6}$$

where $f_i'(\mathbf{x})$ is a noisy objective function of solution **x**, $f_i(\mathbf{x})$ is the real value of objective function, and s_i is the standard deviation of normal distribution of noise (or uncertainty) effect with mean zero. In most noisy GA studies, the standard deviations s_i is kept fixed over all possible values of objective functions (Bui *et al.*, 2005; Fieldsend and Everson, 2005). This assumption is not reasonable in many stochastic problem environments, particularly in the stochastic simulation context. In most stochastic real-world MOPs, the higher objective values are usually expected to have more errors than lower ones. In

minimization problems, if the objective values are quite large with respect to the standard deviation s_i, an employed algorithm is not challenged during the search until the objective values become relatively small so that the standard deviation significantly affects the real values of objective functions.

To model the noise in stochastic environments more accurately, it is suggested that the standard deviation s_i is composed of two components – variable error λ_i and constant error ε_i – over all possible objective function values as follows

$$s_i = \lambda_i f_i(\mathbf{x}) + \varepsilon_i, \tag{5.7}$$

where λ_i is a coefficient that makes the standard deviation able to change corresponding to its objective value, and ε_i is the constant error along all objective values.

5.4. Stochastic Solution Ranking Strategy and Fitness Assignment

The new ranking strategy is based on the classification of candidate solutions of the composite population $\mathbf{CP}_t$ into two different categories (ranks) according to solution dominance similar to FPGA. Firstly, all stochastically nondominated solutions are identified as the first rank, which implies that there is no solution that is stochastically better than these solutions with respect to all objectives simultaneously. All stochastically dominated solutions are identified as the second rank. These ranks are used to evaluate solution fitness for the purpose of solution reproduction. It is important to note that a solution with larger fitness value is preferred.

The fitness of the stochastically nondominated solutions in the first rank is calculated by comparing each nondominated solution with one another and assigning a

fitness value. These values are computed using the crowding distance approach suggested by Deb *et al.* (2002), which has been shown to help maintain diversity among the nondominated solutions in the Pareto optimal front. The larger a solution's fitness value, the greater the distance that solution is from its neighboring nondominated solutions along the Pareto front.

Each stochastically dominated solution in the second rank is compared with all other solutions and assigned a fitness value depending on the probabilities that solutions dominate one another similar to FPGA. The idea here is similar to the strength concept employed in SPEA and SPEA2; however, it has been generalized and developed for the stochastic problem domain. Here, each solution, say $\mathbf{x}_i$, in the composite population $\mathbf{CP}_t$ is assigned a net strength value $S(\mathbf{x}_i)$, indicating the summation of the probabilities that it dominates other solutions, where

$$S(\mathbf{x}_i) = \sum_j P(\mathbf{x}_i \succ \mathbf{x}_j), \ \forall \mathbf{x}_j \in \mathbf{CP}_t \wedge j \neq i. \tag{5.8}$$

The expression $\mathbf{x}_i \succ \mathbf{x}_j$ represents that solution i dominates solution $\mathbf{x}_j$. Then, the fitness value of each dominated solution is calculated using Eq. 3.2

$$F(\mathbf{x}_i) = \sum_{\mathbf{x}_i \rangle \mathbf{x}_j} S(\mathbf{x}_j) - \sum_{\mathbf{x}_j \rangle \mathbf{x}_i} S(\mathbf{x}_j), \ \forall \mathbf{x}_j \in \mathbf{CP}_t \wedge j \neq i, \tag{5.9}$$

where expression $\mathbf{x}_i \rangle \mathbf{x}_j$ denotes that solution $\mathbf{x}_i$ stochastically dominates solution $\mathbf{x}_j$. In other words, a fitness value is assigned to each dominated solution $\mathbf{x}_i$ is equal to the summation of the strength values of all solutions it stochastically dominates minus the summation of the strength values of all solutions by which it is stochastically dominated. SPGA takes into account both dominating and dominated solutions with respect to

solution $\mathbf{x}_i$. This strategy provides more information on Pareto dominance and niching relations among solutions in the composite population and reduces the chance that two solutions have the same fitness value. Thus, no additional diversity preservation mechanism is used among the dominated solutions in the second rank requiring less computation. It is interesting to note that if most solutions do not dominate one another, it is implied that they belong to the first rank where crowding distance operator is invoked to maintain the diversity among them.

After the fitness values of all candidate solutions in $\mathbf{CP}_t$ are calculated, the solutions are compared, where three different scenarios might occur. In the first scenario, two selected solutions have different ranks in which the solution with the better rank is preferred. In the second scenario, two solutions have the same rank but different fitness values in which the solution with larger fitness value is preferred. In the last scenario, two solutions have the same rank and fitness value where one of them is randomly preferred.

5.5. Sampling Operator

In most research studies on MOEAs with noisy objective functions, the number of samplings is arbitrarily taken and kept fixed for all solutions, say 10 or 20 (Babbar *et al*., 2003; Bui *et al*., 2005). There are very few studies that address the optimal sampling problem in noisy genetic algorithms (*e.g.*, Miller, 1997; Gopalakrishnan *et al*., 2001). However, the number of samplings could be different for each solution. One approach is to reduce or remove the overlap of stochastically (not significantly) dominated solutions' confidence intervals from those of the stochastically nondominated solutions. However, overlap removal is a difficult task with respect to multiple objectives. In addition, a higher number of samplings requires a great deal of computational effort restricting the

search to explore more solutions with a smaller number of samplings. The determination of the appropriate number of samplings for each solution or resampling strategy in stochastic MOPs could provide significant performance improvement of any proposed method.

For SPGA, the number of samplings for each solution is determined by the proposed sampling operator as follows

$$\text{Sampling}(insam, maxsam, incsam), \tag{5.10}$$

where *insam* is a small positive integer representing the initial number of samplings, *maxsam* is a positive integer representing the maximum number of samplings allowed for each solution and *incsam* is a small positive integer value representing the increment for the number of samplings.

At each generation, first the population of solutions is evaluated using the initial number of samplings *insam*. Thereafter, the population is classified into three different categories: 1) stochastically nondominated solutions, 2) stochastically dominated solutions and 3) significantly dominated solutions. The solutions in the first and second category, *i.e.*, stochastically nondominated and dominated solutions, are evaluated for *incsam* additional times to obtain better estimates for the real values of their objective functions. No additional samplings for significantly dominated solutions is required, since at a certain confidence level, they are dominated and computational effort of additional samplings could be used for better estimate of exact values of competing solutions at the tradeoff curve. The population with updated objective values (more accurate mean and variance) for solutions in the first and second categories is reclassified. This process continues until all stochastically nondominated and dominated

solutions are evaluated for the maximum number of samplings *maxsam.* When sampling operator is executed, the population with more reliable nondominated solutions is passed for ranking and fitness assignment operation.

The proposed resampling strategy for stochastic MOPs could potentially save extra number of samplings assigned to significantly dominated solutions and provide a higher number of samplings for stochastically nondominated and dominated solutions to better identify the actual nondominated solutions at each generation. After the initial sampling of the population, the stochastically nondominated solutions at the Pareto frontier are not reliable. As more sampling is performed on potential nondominated solutions, the disturbance of noise is reduced, and the more reliable solutions are identified as Pareto frontier.

5.6. SPGA Computational Study

We evaluate the performance of SPGA on a number of test problems with different Pareto optimality characteristics including KUR, ZDT1, ZDT4 and ZDT6 (refer to Section 4.2 for more information on these test problems). The performance of SPGA is also benchmarked against the real-coded NSGA-II of Deb *et al.* (2002). For both SPGA and NSGA-II, all of the parameter settings, except the maximum number of solution evaluations, are used according to the suggested values in the original study of Deb *et al.* (2002) as summarized in Table 5.1. The maximum population size for SPGA is set to the suggested population size used by Deb *et al.* (2002). The number of solution evaluations shown in Table 5.1 depends on the characteristics and complexity of the underlying problem in the stochastic environment estimated by the stopping criterion suggested in CHAPTER 4. The small number of solution evaluations helps us evaluate the

performance of each algorithm more effectively for the expensive, real-world MOPs that may only allow a small number of solution evaluations.

Table 5.1: Parameter settings for SPGA and NSGA-II.

Algorithm Parameter	**SPGA and Real-Coded NSGA-II**			
Test Problem	KUR	ZDT1	ZDT4	ZDT6
Number of Solution Evaluations	1500	7000	12000	10000
Initial Population Size	100			
Maximum Population Size	100			
Crossover Probability	1.0			
Mutation Probability	$1/n$ (where n is number of variables)			
Crossover Type	Simulated Binary Crossover (η_c = 15)			
Mutation Type	Polynomial Mutation (η_m = 20)			
Selection Scheme	Binary Tournament			

For comparative analysis, four performance metrics described in CHAPTER 4 (distance, diversity, delineation and hypervolume ratio metrics) are used to measure the convergence behavior and diversity of SPGA and NSGA-II. Note that the observed values of objective functions are noisy, and they might provide misleading results and improper conclusion. Therefore, to calculate the performance metrics, the real values of objective functions of the obtained population at the end of the search are taken into consideration, since the exact equations of objective functions are known.

For each test problem, each algorithm is run with 50 different initial random seed values and the mean, standard deviation and 95% confidence interval are computed. The

lower and upper bounds of the 95% confidence interval are calculated by $\bar{x} \pm t_{\alpha/2,n-1}\, s/\sqrt{n}$, where $\bar{x}$ is the sample mean, s is sample standard deviation, α is the significance level and is equal to 5% and n is the sample size. To make a more precise statistical comparative analysis and benchmarking, the sample size is set quite large (equal to 50) so that the 95% confidence intervals are considerably reduced.

In this study, we set $\lambda_i = 0.04$ and $\varepsilon_i = 0.02$ to enforce artificial noise around each objective function i (i = 1 and 2). This amount of noise is significant for KUR, ZDT1, ZDT2 and ZDT6 problems and creates some difficulty for an algorithm to converge to the true Pareto optimal front. The experiments on the noisy functions are implemented using random sampling, where the number of samplings is 15, *i.e.*, n = 15. For both SPGA and NSGA-II, the mean of the obtained noisy objective values for each objective function is taken as an estimate for expected objective value. The advantage of making this estimate is to reduce the disturbance of the noise. Obviously, making better estimates requires a higher number of samplings resulting in a larger evaluation computation cost per solution. For the sampling operator employed in SPGA, we set the parameters *insam* = 5, *maxsam* = 15 and *incsam* = 1. This setting means that, at each generation, all solutions are initially evaluated five times, and at any step of the resampling process solutions, which are either stochastically nondominated or stochastically dominated re-evaluated. The re-evaluation of stochastically nondominated or dominated solutions is repeated 10 times ($maxsam - insam$).

5.7. Discussion of Computational Results

In this section, the computational results of SPGA and the real-coded NSGA-II in the stochastic problem environments are presented. Table 5.2 and Table 5.3 show the output statistics including mean, standard deviation and 95% confidence interval (CI) of the four performance metrics obtained from the 50 replications using the following three algorithms: regular SPGA without sampling operator (referred to as SPGA-r), SPGA with sampling operator (referred to as SPGA-s) and real-coded NSGA-II. To illustrate the convergence behavior of SPGA-r, SPGA-s and NSGA-II, the sample populations at the end of the search together with the Pareto optimal front for KUR, ZDT1, ZDT4 and ZDT6 are shown in Figure 5.2, Figure 5.3, Figure 5.4 and Figure 5.5, respectively. These figures show the superiority of SPGA-s and SPGA-r over NSGA-II in rapidly converging to the true Pareto optimal solution set, while maintaining a diverse set of nondominated solutions. Within the given number of solution evaluations, both SPGA-s and SPGA-r obtain the population of nondominated solutions, while a significant proportion of solutions in NSGA-II are dominated solutions, indicating that SPGA-s and SPGA-r have a much faster convergence. It is interesting to note that all obtained nondominated solutions yielded by NSGA-II at the end of the search are dominated by the nondominated solutions of SPGA-s and SPGA-r in most problems.

Table 5.2. Mean, standard deviation and 95% confidence interval of distance ϒ and diversity Δ metrics for SPGA-s, SPGA-r and NSGA-II over 50 random replications.

Test Problem	Algorithm	Distance ϒ			Diversity Δ		
		Avg.	Std. Dev.	95% CI	Avg.	Std. Dev.	95% CI
KUR	SPGA-s	0.0081	0.0014	[0.0077, 0.0085]	0.0709	0.0337	[0.0616, 0.0802]
	SPGA-r	0.0059	0.0011	[0.0056, 0.0062]	0.0794	0.0406	[0.0682, 0.0907]
	NSGA-II	0.1575	0.0211	[0.1516, 0.1633]	0.1176	0.0282	[0.1098, 0.1254]
ZDT1	SPGA-s	0.0690	0.0259	[0.0618, 0.0761]	0.0772	0.0357	[0.0673, 0.0871]
	SPGA-r	0.0690	0.0322	[0.0601, 0.0779]	0.0908	0.0655	[0.0726, 0.1089]
	NSGA-II	0.2416	0.0937	[0.2156, 0.2675]	0.1337	0.0278	[0.1259, 0.1414]
ZDT4	SPGA-s	0.2097	0.2416	[0.1427, 0.2767]	0.0673	0.0495	[0.0536, 0.0810]
	SPGA-r	0.5064	0.3841	[0.3999, 0.6128]	0.3124	0.1152	[0.2805, 0.3443]
	NSGA-II	19.7773	4.5576	[18.5140, 21.0405]	1.8428	1.2563	[1.4946, 2.1911]
ZDT6	SPGA-s	0.1208	0.0614	[0.1038, 0.1378]	0.1347	0.0298	[0.1264, 0.1429]
	SPGA-r	0.1498	0.0631	[0.1323, 0.1673]	0.1499	0.0445	[0.1375, 0.1622]
	NSGA-II	2.087	0.3232	[1.9974, 2.1766]	0.8919	0.1759	[0.8431, 0.9406]

Table 5.3. Mean, standard deviation and 95% confidence interval of delineation Φ and hypervolume ratio *HVR* metrics for SPGA-s, SPGA-r and NSGA-II over 50 random replications.

Test Problem	Algorithm	Delineation Φ			Hypervolume Ratio *HVR*		
		Mean	**Std. Dev.**	**95% CI**	**Mean**	**Std. Dev.**	**95% CI**
KUR	SPGA-s	0.0118	0.0018	[0.0113, 0.0123]	0.0389	0.0055	[0.0374, 0.0405]
	SPGA-r	0.0074	0.0019	[0.0069, 0.0079]	0.0313	0.0058	[0.0297, 0.0329]
	NSGA-II	0.0269	0.0042	[0.0258, 0.0281]	0.1014	0.0155	[0.0971, 0.1057]
ZDT1	SPGA-s	0.0484	0.0139	[0.0445, 0.0522]	0.0927	0.0238	[0.0861, 0.0993]
	SPGA-r	0.0491	0.0199	[0.0436, 0.0546]	0.0880	0.0281	[0.0802, 0.0957]
	NSGA-II	0.1046	0.0251	[0.0976, 0.1116]	0.1962	0.0437	[0.1841, 0.2083]
ZDT4	SPGA-s	0.0251	0.0130	[0.0215, 0.0287]	0.0456	0.0185	[0.0405, 0.0507]
	SPGA-r	0.1337	0.0661	[0.1154, 0.1521]	0.1707	0.0913	[0.1454, 0.1960]
	NSGA-II	0.9111	0.4104	[0.7973, 1.0248]	0.8450	0.1165	[0.8127, 0.8773]
ZDT6	SPGA-s	0.0620	0.0104	[0.0591, 0.0649]	0.1472	0.0218	[0.1412, 0.1532]
	SPGA-r	0.0799	0.0157	[0.0756, 0.0843]	0.1762	0.0313	[0.1675, 0.1848]
	NSGA-II	0.6508	0.1286	[0.6152, 0.6865]	0.8241	0.4974	[0.6862, 0.9620]

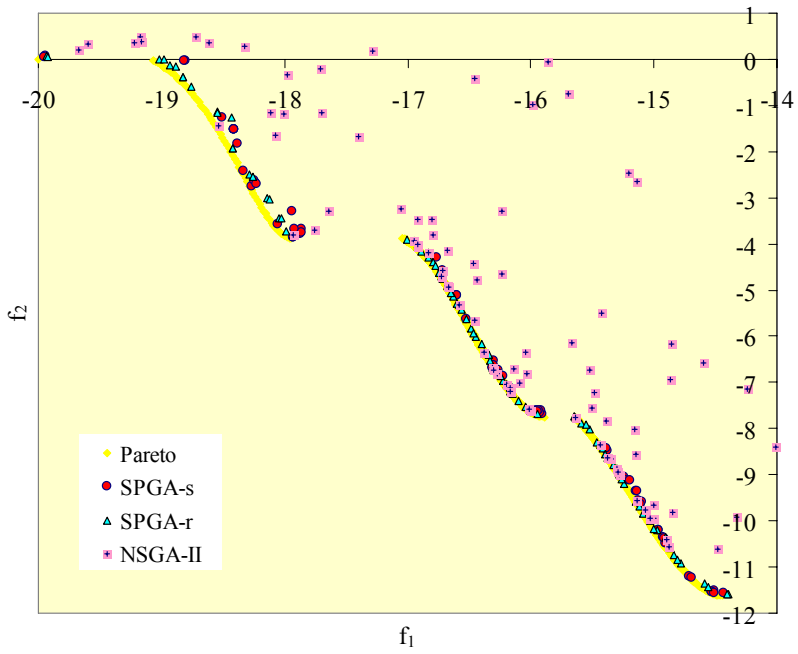

Figure 5.2. The populations with SPGA-s, SPGA-r and NSGA-II on KUR.

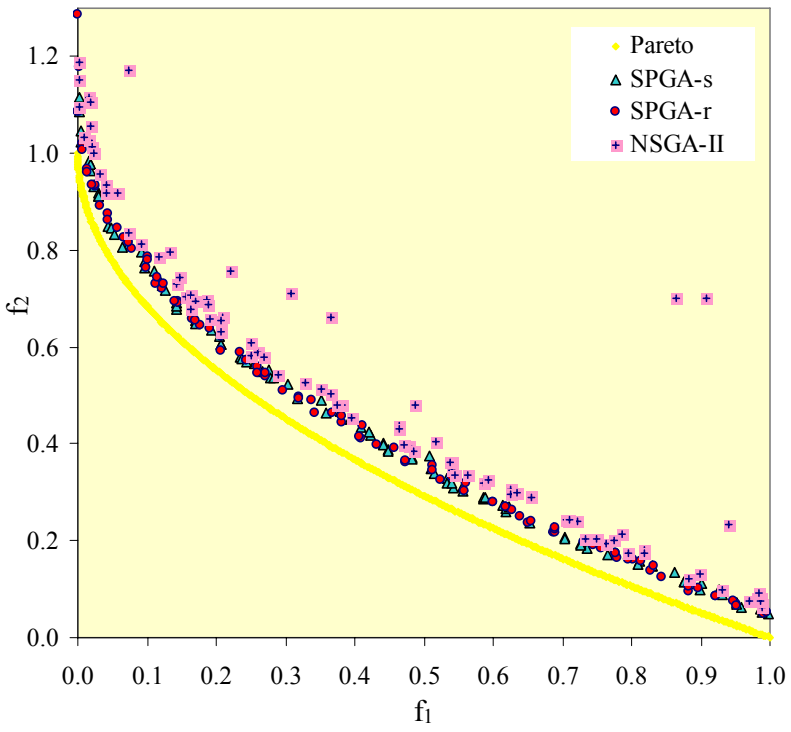

Figure 5.3. The populations with SPGA-s, SPGA-r and NSGA-II on ZDT1.

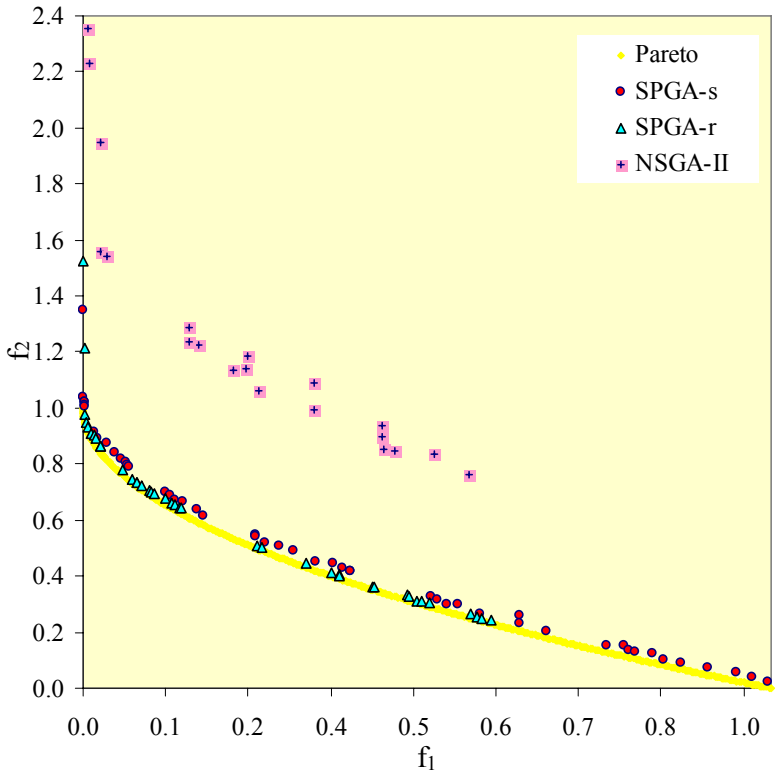

Figure 5.4. The populations with SPGA-s, SPGA-r and NSGA-II on ZDT4.

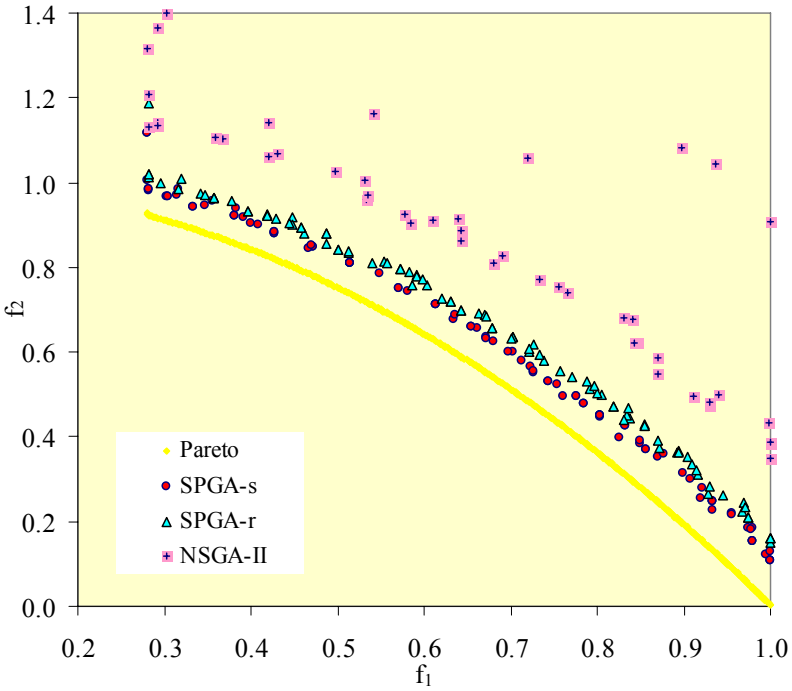

Figure 5.5. The populations with SPGA-s, SPGA-r and NSGA-II on ZDT6.

5.7.1. KUR Test Problem

The results shown in Table 5.2 and Table 5.3 for KUR problem indicate that both SPGA-s and SPGA-r perform quite significantly better than NSGA-II with respect to all metrics. SPGA-r also performs significantly better than SPGA-s with respect to distance, delineation and hypervolume ratio metrics. SPGA-s has slightly, but not significantly, lower value for diversity. The standard deviations of all metrics for three algorithms are small. On KUR problem, sampling operator with the given parameters does not apparently help the SPGA, since SPGA-r provides better overall performance than SPGA-s.

5.7.2. ZDT1 Test Problem

The results shown in Table 5.2 and Table 5.3 for ZDT1 problem indicate that both SPGA-s and SPGA-r perform quite significantly better than NSGA-II with respect to all metrics. SPGA-s and SPGA-r have similar distance metric values and delineation metric values. SPGA-s has slightly, but not significantly, lower value for diversity, whereas it has slightly, but not significantly, higher value for *HVR*. On ZDT1, the sampling operator does not significantly help SPGA to accomplish better performance. The standard deviations of distance metric in all problems for three algorithms are small, except for SPGA-r with respect to diversity.

5.7.3. ZDT4 Test Problem

The obtained results for ZDT4 problem indicate that both SPGA-s and SPGA-r significantly outperform NSGA-II with respect to all metrics. SPGA-s also performs significantly better than SPGA-r for all metrics. The sampling operator significantly helps SPGA improve its performance in terms of all metrics. This significant improvement is obtained by saving considerable number of samplings for significantly dominated solutions during the search and exploring the solution space more thoroughly. The standard deviations of distance and diversity metrics for three algorithms are not relatively small implying that quite different populations are obtained at the end of the search on ZDT4.

5.7.4. ZDT6 Test Problem

The obtained results for ZDT6 problem indicate that both SPGA-s and SPGA-r significantly outperform NSGA-II with respect to all metrics. SPGA-s also performs significantly better than SPGA-r for all metrics. As in ZDT4 problem, the sampling operator significantly helps SPGA improve its performance. The standard deviations of all metrics for three algorithms are relatively small.

CHAPTER 6: SUMMARY AND FUTURE RESEARCH DIRECTIONS

6.1. Introduction

This chapter provides a summary of the research, conclusions and future research directions.

6.2. Summary and Conclusions

It has been shown that evolutionary algorithms, the focus of this study, are powerful, intelligent optimization algorithms that are able to balance exploration and exploitation of the solution search space. The drawbacks of traditional approaches, which typically try to scalarize the multiple objectives into a single objective, have motivated researchers and practitioners to seek alternative techniques to find a set of Pareto optimal solutions rather than just a single solution.

This research presents two new multiobjective evolutionary algorithms, called fast Pareto genetic algorithm (FPGA) and stochastic Pareto genetic algorithm (SPGA) for dealing with multiobjective optimization problems, where each solution evaluation is computationally- and/or financially-expensive. FPGA is designed for handling MOPs with deterministic objective values, whereas SPGA is equipped with an enhanced stochastic ranking procedure and resampling strategy to be a robust approach for solving MOPs with uncertain, normally-distributed objective function values, particularly in stochastic simulation context. Both approaches are Pareto-based multiobjective optimization methods using genetic algorithms. New genetic operators are introduced to enhance both algorithms' performance in finding Pareto optimal solutions while

minimizing computational effort. An elitism operator with high intensity is employed to ensure the quick propagation of the nondominated solutions, and a dynamic regulation operator to dynamically adapt the population size. In addition to distance and hypervolume ratio metrics, two new metrics, called diversity and delineation, are defined to better discriminate among the MOEAs.

Computational results for seven well-known test problems with different Pareto optimality characteristics indicate that FPGA is capable of efficiently and effectively direct the search toward Pareto optimal front. Statistical analyses show that, within a relatively small number of solution evaluations, FPGA outperforms NSGA-II in most problems in terms of rapidly converging to the true Pareto optimal solution set while preserving a diverse, evenly-distributed set of nondominated solutions. Adaptive population sizing is most likely one of the main factors resulting in the superiority of FPGA over NSGA-II in this benchmark environment. It is also believed that FPGA benefits its own unique feature of small number of solution evaluations at each generation which saves a significant number of solution evaluations early in the search and utilizes the exploitation in a more efficient manner at later generations. However, FPGA could be more effective if it incorporates a diversity preservation mechanism into its fitness assignment strategy to emphasize the less crowded dominated solutions. Incorporation of a diversity preservation mechanism or reduction of high elitism intensity might help FPGA not to bias towards some regions found as a diversification maintenance problem in few replications in ZDT2 and ZDT3 problems.

In the stochastic problem environment, computational results on four test problems indicate the superiority of SPGA over NSGA-II in terms of all performance

metrics. Within the given number of solution evaluations, both SPGA with the resampling operator (SPGA-s) and SPGA without resampling operator (SPGA-r) obtain the population of nondominated solutions, while a significant proportion of solutions in NSGA-II are dominated solutions, indicating that SPGA-s and SPGA-r have much faster convergence. It is interesting to note that all obtained nondominated solutions yielded by NSGA-II at the end of the search are dominated by the nondominated solutions of SPGA-s and SPGA-r in most problems. Results obtained from a little experimentation presented in CHAPTER 5 imply that sampling operator could help SPGA in many MOPs. However, in some MOPs, it might not be helpful or even worsen SPGA's performance if appropriate selection of resampling operator parameters is not carried out. Furthermore, any strong conclusion about the practicality and usefulness of resampling operator demands further experimentation of SPGA on a larger suite of test problems with several different levels of noise.

6.3. Future Research Directions

There are several additional aspects that need to be addressed and investigated for providing FPGA and SPGA as more robust multiobjective simulation optimization tools. The proposed future research directions are outlined in the following sections.

6.3.1. Expanded Suite of Test Problems with Different Properties

Although FPGA and SPGA have been tested on a suite of well-known test problems with different optimality characteristics, they can be tested and benchmarked on several other test problems, different in dimension of search space, higher in dimension of objective space, constraint and different optimality characteristics. For example, Deb

et al. (2001) suggest well-known suite of DTLZ problems that are scalable to any number of decision variables and objectives with provided knowledge of exact shape and location of the resulting Pareto optimal front. In addition, since for each DTLZ problem difficulties in both converging to the Pareto optimal front and maintaining a diverse set of solutions are known, they provide very useful validation and benchmarking environment for better understanding of the working principles of FPGA and SPGA. For the constrained test problems, OSY and TNK problems, suggested by Osyczka and Kundu (1995) and Tanaka (1995), respectively, are among the more popular ones.

On the other hand, the proposed optimization algorithms should be evaluated on a number of discrete variable test problems including Boolean functions defined over bit-strings. The multiobjective 0-1 knapsack problem is a very good test problem in this case, since it is simply described but very difficult to solve, as it is a well-known NP-Hard problem. Moreover, it is a very practical problem investigated in various fields including project selection, finance and portfolio investment. As an example, Zitzler and Thiele (1999) introduce a suite of nine multiobjective 0-1 knapsack problems with the number of items as 250, 500 and 750, and the number of knapsacks as 2, 3 and 4 for comparative analysis of five different MOEAs.

6.3.2. Parameter Settings

Without any doubt, an appropriate selection of parameter settings for any MOEAs could significantly improve the performance of the algorithm. As the primary intent of this research is to introduce a novel approach that addresses solving expensive MOPs, the attempt to determine more appropriate (and perhaps more robust) parameter settings for FPGA and SPGA is left for future study. The suggested values for most of the parameters

used for FPGA and SPGA are obtained by either performing several pilot runs or taking them from the literature, particularly from Deb *et al.* (2002) for benchmarking. It is important to note that the best parameter settings are typically problem-dependent and may vary over different problems. Therefore, an investigation of some intelligent GA operators in the MOEA field that are capable of automatically adjusting their rates is recommended. An algorithm equipped with the feature of self-adjustment operation rates may have the benefit of higher convergence velocity by searching the solution space in a more efficient manner.

6.3.3. Additional MOEA Performance Metrics

Two complimentary performance metrics can be used to compare nondominated solutions produced by the various MOEAs. The first measure, called attainment surface, calculates a frequency distribution for intersection points of each cross-line with attainment surfaces obtained from nondominated solutions sets (Fonseca and Fleming, 1999). Then, it compares statistically the frequency distributions of intersection points for all cross-lines for two MOEAs head to head. The second measure, called *C* metric, compares the coverage of nondominated solution sets of two different MOEAs by measuring the percentage of the solutions in one set is dominated by the solutions in another set (Zitzler and Thiele, 1999). This measure presents the superiority of one MOEA over another MOEA by comparing the coverage of their nondominated solution sets. Each of these two metrics considers simultaneously both goals in multiobjective optimization, convergence to the Pareto optimal set and maintenance of diversity.

6.3.4. Statistical Comparative Analysis of Performance Metrics

It is surprising to note that although a significant amount of research has been carried out and many MOEAs, test problems and performance metrics are introduced in the MOEA area in the last decade, very little statistical analysis of results are employed to perform extensive comparative analysis among the proposed MOEAs. Since EAs are random search approaches and a few experiments with different seeds are run for each instance, using appropriate statistical tools are advisable to be employed for more precise comparative analyses.

6.3.5. Integration of the Proposed Methodology with Commercial Simulation Software

A very interesting and practical task is to integrate the proposed optimization methodology, SPGA, with simulation software package like ARENA. At this time, there is no interface between SPGA and simulation software and the search cannot be performed automatically. If we would like to apply SPGA to a simulation model, it requires a great deal of effort to manually import the objective values and variances of solutions to SPGA and export the values of the decision variables of candidate solutions to simulation software. However, it is possible to integrate the SPGA and simulation software and perform the search automatically.

LIST OF REFERENCES

Abido, M.A. (2001). "A new multiobjective evolutionary algorithm for environmental/economic power dispatch." *Power Engineering Society Summer Meeting*, Vol. 2, pp. 1263-1268.

Alrefaei, M.H. and S. Andradóttir. (1999) "A simulated annealing algorithm with constant temperature for discrete stochastic optimization." Management Science. Vol 45, No. 5. pp. 748-764.

Andradóttir, S. (1998a). "Simulation Optimization." In J. Banks (Ed.) Handbook of Simulation. John Wiley and Sons, New York, NY.

Andradóttir, S. (1998b). "A Review of Simulation Optimization Techniques." Proceedings of the 1998 Winter Simulation Conference, D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannan (eds.), pp. 151-158.

April, J., F. Glover, J. Kelly, and M. Laguna. (2001). "Simulation/ Optimization using "Real-World" Applications," *Proceedings of the 2001Winter Simulation Conference*, pp. 134-138.

April, J., F. Glover, J. Kelly and M. Laguna. (2003). "Practical Introduction to Simulation Optimization." Proceedings of the 2003 Winter Simulation Conference. S. Chick, T. Sanchez, D. Ferrin and D.Morrice, eds., pp. 71-78, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Azadivar, F. (1999). "Simulation Optimization Methodologies." In P.A. Farrington, H.B. Nembhard, D.T. Sturrock, and G.W. Evens (Eds.) Proceedings of the 1999 Winter Simulation Conference. pp. 93-100.

Azadivar, F. and G. Tompkins. (1999). "Simulation Optimization with Qualitative Variables and Structural Model Changes: A Genetic Algorithm Approach." European Journal of Operational Research. Vol. 113. Pp. 169-812.

Azadivar, F. and J. Wang. (2000). "Facility Layout Optimization Using Simulation and Genetic Algorithms." International Journal of Production Research. Vol. 38, No. 17. pp. 4369-4383.

Azizi, N., S. Zolfaghari. 2004. "Adaptive temperature for simulating annealing: a comparative study." Computers & Operations Research, Vol. 31, pp. 2439-2451.

Babbar, M., A. Lakshmikantha and D.E. Goldberg. 2003. A Modified NSGA-II to Solve Noisy Multiobjective Problems, in James Foster (editors), 2003 Genetic and

Evolutionary Computation Conference. Late-Breaking Papers, pp. 21--27, AAAI, Chicago, Illinois, USA, July.

Baesler, F.F. and J.A. Sepúlveda. (2000). "Mutli-Response Simulation Optimization using Stochastic Genetic Search within a Goal Programming Framework." In J.A. Joines, R.R. Barton, K.Kang, and P.A. Fishwick (Eds.) Proceedings of the 2000 Winter Simulation Conference. pp. 788-794.

Bagchi, T. P. (2001). "Pareto-Optimal Solutions for Multi-objective Production Scheduling Problems." In Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, First International Conference on Evolutionary Multi-Criterion Optimization, pp. 458-471. Springer-Verlag. Lecture Notes in Computer Science No. 1993.

Barton, R.R. (1998). "Simulation Metamodels." Proceedings of the 1998 Winter Simulation Conference, pp. 167–176, Washington, USA .

Borjesson, P.O. and C.E.W. Sundberg. (1979). "Simple approximation of the error function Q(x) for communications applications." *IEEE Transactions on Communications*, Vol. 27, No. 3, pp. 639-643.

Boyle, C.R. and W.S. Shin, (1996). "An interactive multiple-response simulation optimization method." *IIE Transactions*, Vol. 28, pp. 319-331.

Bui, L. T., H. A. Abbass and D. Essam. (2005). "Fitness Inheritance For Noisy Evolutionary Multi-Objective Optimization." in Hans-Georg Beyer et al. (editors), *2005 Genetic and Evolutionary Computation Conference* (*GECCO'2005*), Vol. 1, pp. 779-785, ACM Press, New York, USA.

Coello, C. A. C. (1996). "An Empirical Study of Evolutionary Techniques for Multiobjective Optimization in Engineering Design." PhD thesis, Department of Computer Science, Tulane University, New Orleans, Louisiana.

Coello, C. A. C. (1999). "A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques." *Knowledge and Information Systems*, Vol. 1, No. 3, pp. 269-308.

Coello, C.A.C., D.A. Van Veldhuizen and G.B. Lamont. (2002). Evolutionary Algorithms for Solving Multi-Objective Problems. 1st Ed. New York: Kluwer Academic Publishers.

Coello, C.A.C., and G.B. Lamont. 2004. Applications of Multi-Objective Evolutionary Algorithms. Singapore: World Scientific.

Collette, Y., P. Siarry. (2005). "Three new metrics to measure the convergence of metaheuristics towards the Pareto frontier and the aesthetic of a set of solutions in biobjective optimization." *Computers & Operations Research*, Vol. 32, pp. 773-792.

Deb, K. (2001). Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons, Chichester, UK.

Deb K. and R.B. Agrawal (1995). "Simulated Binary Crossover for Continuous Search Space." *Complex System*, Vol. 9, pp. 115-148.

Deb, K., and M. Goyal. (1996). "A Combined Genetic Adaptive Search (GeneAS) for Engineering Design." *Computer Science and Informatics*, Vol. 26, No. 4, pp. 30-45.

Deb, K., M. Mohan and S. Mishra. (2005). "Evaluating the epsilon-Domination Based Multi-Objective Evolutionary Algorithm for a Quick Computation of Pareto-Optimal Solutions." *Evolutionary Computation*, Vol. 13, No. 4, pp. 501-525.

Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan. (2002). "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA—II." IEEE Transactions on Evolutionary Computation, Vol. 6, No. 2, pp. 182-197.

Dudewicz, E.J. and S.R. Dalal. (1975). "Allocation of Observations in Ranking and Selection With Unequal Variances." Sankhya: The Indian Journal of Statistics. Vol. 37, No. 1. pp. 37-78.

Erbas, C., S. Cerav-Erbas and A. D. Pimentel. 2006. Multiobjective Optimization and Evolutionary Algorithms for the Application Mapping Problem in Multiprocessor System-on-Chip Design. To appear in *IEEE Transactions on Evolutionary Computation*.

Evans, G., B. Stuckman, and M. Mollaghasemi, (1991) "Multicriteria Optimization of Simulation Models." *Proceedings of the 1991 Winter Simulation Conference*, Phoenix, Arizona.

Fonseca, C. and P.J. Fleming. (1993). "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization." In Proceedings of the Fifth International Conference on Genetic Algorithms (University of Illinois at Urbana-Champaign), S. Forrest, Ed. Morgan Kaufmann, San Mateo, CA, pp. 416–423.

Fu, M.C. (1994). "A Tutorial Review of Techniques for Simulation Optimization." In J.D. Tew, S. Manivannan, D.A. Sadowski, and A.F. Seila (Eds.) Proceedings of the 1994 Winter Simulation Conference. pp. 149-156.

Fu, M. (2002). "Optimization for Simulation: Theory and Practice," INFORMS Journal on Computing, Vol. 14. No. 3, pp. 192-215.

Gerencs´er, L. (1999) “Rate of convergence of moments for a simultaneous perturbation stochastic approximation method for function minimization.” *IEEE Trans. Automat. Contr.*, Vol. 44, pp. 894–906.

Glasserman, P. (1991). *Gradient Estimation via Perturbation Analysis*. Kluwer Academic Publishers, Boston, Massachusetts.

Glover, F. and M. Laguna. (1997). *Tabu Search*. Kluwer Academic Publishers, Norwell, Massachusetts.

Glover, F., J.P. Kelly, and M. Laguna. (1999). “New Advances for Wedding Optimization and Simulation.” Proceedings of the 1999Winter Simulation Conference, pp. 255-260.

Goldberg, D.E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, Massachusetts.

Goldberg, D. E. and K. Deb. (1991). “A comparison of selection schemes used in genetic algorithms.” Foundations of Genetic Algorithms, I. pp. 69-93.

Goldberg, D.E., K. Deb, and J.H. Clark. (1992). “Genetic Algorithms, Noise, and the Sizing of Populations.” Complex Systems. Vol. 6. pp. 333-362.

Gopalakrishnan, G., B. Minsker and D. Goldberg. (2001). “Optimal Sampling in a Noisy Genetic Algorithm for Risk-Based Remediation Design”.

Gupta, S.S. (1956). “On a Decision Rule for a Problem in Ranking Means.” *Mimeograph Series No. 150, Institute of Statistics*. University of North Carolina, Chapel Hill.

Hedlund, H.E., M. Mollaghasemi. 2001. A genetic algorithm and an indifference-zone ranking and selection framework for simulation optimization. In Proceedings of Winter Simulation Conference, 417-421.

Herrera, F., M. Lozano, J.L. Verdegay. (1998). “Tackling real-coded genetic algorithms: operators and tools for behavioral analysis.” Artificial Intelligence Review. Vol. 12. No. 4. pp. 265–319.

Ho, Y.C. and X.R. Cao. (1991). *Perturbation Analysis of Discrete Event Dynamic Systems*. Kluwer Academic Publishers, Norwell, Massachusetts.

Ho, S.C., D. Haugland. 2004. “A tabu search heuristic for the vehicle routing problem with time windows and split deliveries” Computers & Operations Resaerch, Vol. 31, pp. 1947-1964.

Holland, J.H. (1975). Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, Michigan.

Horn, J. and N. Nafpliotis. (1993). "Multiobjective optimization using the niched Pareto genetic algorithm." IlliGAl Rep. 93005. University of Illinois at Urbana-Champaign, Champaign, IL.

Hughes E. J. (2001). "Evolutionary Multi-objective Ranking with Uncertainty and Noise." In Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, editors, First International Conference on Evolutionary Multi-Criterion Optimization, pp. 329-343. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

Jeon, Y.J, J.C. Kim. 2004. "Application of simulated annealing and tabu search for loss minimization in distribution systems." Electrical Power and Energy Systems, Vol. 26, pp. 9-18.

Joines, J., D. Gupta, M. A. Gokce, R. E. King and M. G. Kay. 2002. Supply Chain Multi-Objective Simulation Optimization. In Proceedings of the 2002 Winter Simulation Conference, ed. E. Yücesan, C.-H. Chen, J.L. Snowdon, and J.M. Charnes, 1306-1313. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Kapuscinski, R., S. Tayur. (1998). "A capacitated production-inventory model with periodic demand". *Operations Research*, Vol. 46, pp. 899–911.

Keys, A.C. and L.P. Rees. (2004). "A sequential-design metamodeling strategy for simulation optimization." International Journal of Computers and Operations Research. Vol. 31. pp. 1911-1932.

Kiefer, J. and J. Wolfowitz. (1952). "Stochastic Estimation of the Maximum of a Regression Function." Annals of Mathematical Statistics. Vol. 23. pp. 446-462.

Koenig, L.W. and A.M. Law. (1985). A procedure for selecting a subset of size m containing the l best of k independent normal populations, with applications to simulation. *Communications in Statistics* B14:719 734.

Kursawe, F. 1990. A Variant of Evolution Strategies for Vector Ooptimization. eds. H.P. Schwefel and R. Männer, In Proceedings of the 1st Parallel Problem Solving from Nature Workshop, 496, 193-197, Springer-Verlag, Berlin, Germany.

Lacksonen, T. (2001). "Empirical comparison of search algorithms for discrete event simulation". Computers and industrial engineering. Vol. 40. pp. 133-148.

Law, A.M. and W.D. Kelton. (2000). Simulation Modeling and Analysis. 3rd Edition. McGraw-Hill, New York, New York.

Lee, Y.H., H.M. Shin and B.-H. Yang. (1996). “An Approach for Multiple Criteria Simulation Optimization with Application to Turning Operation", *Journal of Computers and Industrial Engineering*, Vol. 30, No. 3, pp. 375-386.

Michalewicz Z., and D. B. Fogel. (2000). *How to solve it: Modern Heuristics*. Springer.

Michielssen, E. and D.S. Weile. (1995). “Electromagnetic system design using genetic algorithms.” In Genetic Algorithms and Evolution Strategies in Engineering and Computer Science: Recent Advances and Industrial Applications, D. Quagliarella, J. Périaux, C. Poloni, and G. Winter, Eds. John Wiley and Sons Ltd., Chichester, UK, pp. 267–288.

Miller, B.L. (1997). “Noise, Sampling, and Efficient Genetic Algorithms.” IlliGAL Report No. 97001. University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL.

Mitchell, M. (1996). An Introduction to Genetic Algorithms. The MIT Press, Cambridge, MA.

Mitra, K. K. Deb, and S. K. Gupta. (1998). “Multiobjective Dynamic Optimization of an Industrial Nylon 6 Semibatch Reactor Using Genetic Algorithm.” Journal of Applied Polymer Science. Vol. 69 No.1. pp. 69-87.

Mollaghasemi, M. (1994) “Multiple Response Optimization of Simulation Models.” Transactions of the Society for Computer Simulation. Vol. 11, No. 3.

Mollaghasemi, M., G. Evans, and W. Biles, (1991). “An Interactive Approach for Optimizing Multi-Response Simulation Models," *Proceedings of the 13th Annual Conference for Computers and Industrial Engineering*, Orlando, Florida.

Mollaghasemi, M. and G. Evans. (1994) “Multicriteria Design of Manufacturing Systems through Simulation Optimization.” IEEE Transactions on System, Man, and Cybernetics. Vol. 24, No. 9.

Mollaghasemi, M. and J. Pet-Edwards. (1992). Technical Briefing: Making Multiple-Objective Decisions. IEEE Computer Society Press, Los Alamitos, CA.

Morse, J.N. (1980). “Reducing the size of nondominated set: pruning by clustering.” Computer and Operations Research. Vol. 7. No. 1–2. pp. 55–66.

Nelson, B. L., J. Swann, D. Goldsman, and W. Song. (2001). “Simple procedures for selecting the best system when the number of alternatives is large.” Operations Research, Vol. 49, No. 6, pp. 950–963.

Olafsson, S. and Kim, J. (2002). Simulation optimization. Proceedings of the 2002 Winter Simulation Conference, pp. 79-84.

Osyczka, A. and S. Kundu. (1995). "A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm." *Structural Optimization*, Vol. 10, pp. 94-99.

Oyama A an.d M.S. Liou. (2001). "Multiobjective Optimization of Rocket Engine Pumps using Evolutionary Algorithm." In Proceedings of the 15th AIAA Computational Fluid Dynamic Conference, American Institute of Aeronautics and Astronautics, Anaheim, California, June 2001.

Oyama A an.d M.S. Liou. (2002). "Multiobjective Optimization of Rocket Engine Pumps using Evolutionary Algorithm." Journal of Propulsion and Power, Vol. 18. No. 3. pp. 528-535.

Poles, S., E. Rigoni and T. Robic. (2004). "MOGA-II Performance on Noisy Optimization Problems.) in Bogdan Filipic and Jurij Silc (editors), Bioinspired Optimization Methods and Their Applications. Proceedings of the International Conference on Bioinspired Optimization Methods and their Applications, BIOMA 2004, 51-62, Jozef Stefan Institute, Ljubljana, Slovenia.

Robbins, H. and S. Monroe. (1951). "A Stochastic Approximation Method." Annals of Mathematical Statistics. Vol. 22. pp. 400-407.

Rubinstein, R.Y. (1991). "How to optimize discrete-event systems from a single sample path by the score function method." *Annals of Operations Research*, Vol. 27, pp. 175-212.

Rubinstein, R.Y. and A. Shapiro. (1993). *Discrete Event Systems: Sensitivity Analysis and Stochastic Approximation Using the Score Function Method*. John Wiley & Sons, Inc., Chichester.

Safizadeh, M.H. (2002). "Minimizing the bias and variance of the gradient estimate in RSM simulation studies." European Journal of Operational Research. Vol. 136 pp. 121-135.

Schaffer, J.D. 1987. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms, 93-100. Lawrence Erlbaum.

Schott, J. R., (1995). "Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization." Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, May 1995.

Shapiro, A. (1996). Simulation based optimization. *Proceedings of the 1996 Winter Simulation Conference*, pp. 332-336.

Shen, Z.J. and M. Daskin. 2005. "Tradeoffs between Customer Service and Cost in Integrated Supply Chain Design." Manufacturing and Service Operations Management. Working paper.

Silva C.M., and E.C. Biscaia. 2003. Genetic Algorithm Development for Multi-objective Optimization of Batch Free-Radical Polymerization Reactors. Computers and Chemical Engineering 27, 1329-1344.

Singh, A. B. Minsker and D. E. Goldberg. (2003). "Combining Reliability and Pareto Optimality-An Approach Using Stochastic Multi-Objective Genetic Algorithms." In *American Society of Civil Engineers (ASCE) Environmental & Water Resources Institute (EWRI) World Water & Environmental Resources Congress 2003 & Related Symposia*, Philadelphia, PA.

Spall, J.C. (1992). "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation." *IEEE Transactions on Automatic Control*, Vol. 37, pp. 332-341.

Srinivas N. and K. Deb 1994. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. International Journal of Evolutionary Computation 2(3), 221-248.

Swisher, J.R., P.D. Hyden, S.H. Jacobson, and L.W. Schruben. (2000). "A Survey of Simulation Optimization Techniques and Procedures." in J.A. Jones, R.R. Barton, K. Kang, and P.A. Fishwick (eds.), Proceedings of the 2000 Winter Simulation Conference. pp. 119-128.

Swisher, J.R., S.H. Jacobson. (1999). "A survey of ranking, selection, and multiple comparison procedures for discrete-event simulation." Proceedings of the 1999 Winter Simulation Conference, pp. 492-501.

Szidarovszky, F., M.E. Gershon, L. Duckstein. (1986). "Techniques for multiobjective decision making in systems management". Elsevier science publishers. Amsterdam, The Netherlands. pp. 34-39.

Talbi E.G., M. Rahoual, M. H. Mabed, and C. Dhaenens. (2001). "A Hybrid Evolutionary Approach for Multicriteria Optimization Problems: Application to the Flow Shop." In E. Zitzler, K. Deb, L. Thiele, C. A. C. Coello, and D. Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, pp. 416-428. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

Tamaki, H., H. Kita and S. Kobayashi. (1996). "Multi-objective optimization by genetic algorithms: A review." In Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (Nagoya, Japan), T. Fukuda and T. Furuhashi, Eds. 517–522. Inc., San Francisco, CA, pp. 658–665.

Tanaka, M., H. Watanabe, Y. Furukawa, and T. Tanino. (1995). "GA-Based Decision Support System for Multicriteria Optimization." *Proceedings of the International Conference on Systems, Man, and Cybernetics*, Vol. 2, pp. 1556-1561, Piscataway, NJ.

Teich, J. 2001. Pareto-Front Exploration with Uncertain Objectives. Proceedings of the First Conference on Evolutionary Multi-Criterion Optimization.

Teleb, R. and F. Azadivar. (1994). "A Methodology for Solving Multi-Objective Simulation-Optimization Problems." European Journal of Operational Research. Vol. 72. pp. 135-145.

Van Veldhuizen, D. A. and G. B. Lamont. (2000) "On Measuring Multiobjective Evolutionary Algorithm Performance." In *2000 Congress on Evolutionary Computation*, Vol. 1, pp. 204-211, Piscataway, New Jersey.

Vedarajan G., L.C. Chan and D.E. Goldberg. (1997). "Investment portfolio optimization using genetic algorithms." Proceedings of the 1997 Conference on Genetic Programming: Late Breaking Papers, J. R. Koza, Ed. pp. 255–263.

Yan, D., and H. Mukai. (1992). "Stochastic discrete optimization." *SIAM Journal on Control and Optimization*, Vol. 30, pp. 594-612.

Yang, W. and B.L. Nelson. (1991). "Using Common Random Numbers and Control Variates in Multiple-Comparison Procedures." Operations Research. Vol. 39. pp. 583-591.

Yee, A.K.Y., A.K. Ray, G.P. Rangaiah. (2003). "Multiobjective optimization of an industrial styrene reactor." *Computers and Chemical Engineering*, Vol. 27, pp. 111-130.

Zitzler, E., K. Deb and L. Thiele. (2000). "Comparison of Multiobjective Evolutionary Algorithms: Empirical Results." *Evolutionary Computation*, Vol. 8, No. 2, pp. 173-195.

Zitzler, E., M. Laumanns and L. Thiele. 2001. SPEA2: Improving the Strength Pareto Evolutionary Algorithm, Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland.

Zitzler, E. and L. Thiele. (1999). "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach." IEEE Transactions on Evolutionary Computation. Vol. 3. No. 4. pp. 257-271.