STATE-OF-THE-ART MULTIOBJECTIVE EVOLUTIONARY ALGORITHMS—PARETO PANKING, DENSITY ESTIMATION AND DYNAMIC POPULATION

By

HAIMING LU

Bachelor of Engineering

Tsinghua University

Beijing, China

1995

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirement for
the Degree of
DOCTOR OF PHILOSOPHY
August, 2002

STATE-OF-THE-ART MULTIOBJECTIVE

EVOLUTIONARY ALGORITHMS—PARETO

PANKING, DENSITY ESTIMATION

AND DYNAMIC POPULATION

Thesis Approved:

_____

Thesis Advisor

_____

_____

_____

Dean of the Graduate College

PREFACE

This study classifies existing Multiobjective Evolutionary Algorithms (MOEAs) and analyzes several state-of-the-art MOEAs based on different design procedures of those crucial building blocks. A Rank-Density based Genetic Algorithm (RDGA) is designed by synergistically integrates important features of existing MOEAs in a unique way. From the simulation results, RDGA has shown its capability in finding a *near-complete* and *near-optimal* Pareto set at the final and successfully applied in a neural network design problem. In addition, an MOEA with dynamic population size—Dynamic Multiobjective Evolutionary Algorithm (DMOEA) is derived from RDGA. Regulated by dynamic population strategies, DMOEA generation is found to be competitive with, or even superior to, other representative MOEAs in terms of keeping the diversity of the individuals along the trade-off surface, tending to extend the Pareto front to new areas, finding a well-approximated Pareto optimal front, and achieve optimal population size according to desired density value and approximated number of trade-off hyper-areas. Based on extensive studies on MOEAs, an MOEA Toolbox is designed to provide flexible choices to the users by combining different building blocks. To increase the convergence speed of DMOEA, a Particle Swarm Optimization (PSO) technique combined with genetic selection is proposed in a Dynamic Particle Swarm Evolutionary Algorithm (DPSEA). The comparison results show that DPSEA improves both efficiency and efficacy of evolutionary process and can be potentially applied to time varying multiobjective optimization problems in future work.

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to my academic advisor, Professor Gary G. Yen, for he has guided, instructed, encouraged, inspired and continually motivated me for four years. I feel fortunate to have him as a mentor and a friend. My genuine appreciation draws to my other committee members Professor Jong-Moon Chung, Professor Guoliang Fan, Professor R. Russell Rhinehart and Professor R. K. Yarlagadda whose direction, reassurance and acquaintance are also worthwhile.

I am indebted to all my friends and colleagues in Stillwater. Many thanks are due to all of them, especially the past and present members in Intelligent Systems and Control Laboratory at Oklahoma State University for their supportive help.

I would also like to thank to my family—my parents and brother, for encouraging me to do a good job all the time, to appreciate the value of hard work, and most importantly, for never giving up.

Finally, special honor and appreciation is given to my wife, Huan Wang. Without her understanding love, constant help, happy smile and especially joyful prayer, I would never have the moral and spiritual strength necessary for the completion of this research and dissertation.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

I. INTRODUCTION

1.1 Motivation

Many real-world problems in engineering, science, business, and natural and social sciences are largely characterized by the need to allocate limited resources to a collection of activities in application areas, such as inventory control, transportation networks, queuing analysis, task scheduling, capital investment, delivery of health services, water-resource management, and energy procurement programs. These problems involve multiple measures of performance, or objectives, which should be optimized simultaneously. In certain cases, objective functions may be optimized independently from each other to achieve the best result in each performance dimension. However, suitable solutions to the overall problem can hardly be found in this way. Optimal performance according to one objective, if such an optimum exists, may lead to unacceptably low performance in one or more of the other objectives. For example, in the design of an automobile, an engineer may wish to maximize crash resistance for safety and minimize weight for fuel economy. This is a multiobjective optimization problem with two conflicting goals, that is, a step towards improving one of the objectives, say enhancing crash resistance, is generally a step away from improving the other, increasing weight. Obviously, in this case, the notion of "optimum" has to be redefined since a single optimal point will not satisfy both objectives simultaneously.

In large-scale systems, these Multiobjective Optimization Problems (MOPs) are even more complicated. For instance, in a plant production study, one may not be

satisfied with only knowing what actions lead to minimizing production costs. Instead, the study may be taken so that it identifies additional objectives such as short-term and long-term capital gains, employee satisfaction and well-being, product diversification, and energy conservation managements. Obviously, some of these objectives are competing, or even conflicting, which cannot achieve an optimal solution at the same time. A suitable solution to such problems involving conflicting objectives should offer "acceptable" performance, though possibly sub-optimal in the single-objective sense, in all objective dimensions, where "acceptable" is problem-dependent and ultimately subjective.

1.2 Objective

The simultaneous optimization of multiple, possibly conflicting, objective functions deviates from single function optimization in that it seldom result in a single, global optimal solution. Instead, MOPs tend to be characterized by a family of alternatives that must be considered equivalent in the absence of information concerning the relative importance of each objective to the others. The family of solutions to a multiobjective optimization problem is composed of all those elements of the search space that are components of the corresponding objective vectors which cannot be all simultaneously improved. This is known as the concept of *Pareto optimality* [1]. A formal definition of Pareto optimality is given as follows [2]. Consider, without loss of generality, the minimization of the n components $f_k, k = 1,\ldots,n,$ of a vector function $\mathbf{f}$ of a vector variable $\mathbf{x}$ in a universe μ, where

$$\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}),\ldots, f_n(\mathbf{x})). \tag{1.1}$$

Then a decision vector $\mathbf{x}_\mu \in \mu$ is said to be Pareto-optimal if and only if there is no $\mathbf{x}_v \in \mu$ for which $\mathbf{v} = \mathbf{f}(\mathbf{x}_v) = (v_1, \ldots, v_n)$ dominates $\mathbf{u} = \mathbf{f}(\mathbf{x}_u) = (u_1, \ldots, u_n)$, that is, there is no $\mathbf{x}_v \in \mu$ such that

$$\forall i \in \{1, \ldots, n\}, v_i \le u_i \qquad \textit{and} \qquad \exists i \in \{1, \ldots, n\} \mid v_i < u_i . \tag{1.2}$$

The set of all Pareto-optimal decision vectors is called the *Pareto-optimal* set of the problem. The corresponding set of objective vectors is called the non-dominated set, or *Pareto front*. Apparently, the Pareto front dominates all other possible solutions and in most cases, it is located on the boundary of the objective vector space (i.e., feasible solution space) as shown in Figure 1.1 for a two-objective optimization problem (f_1 and f_2 refer to two cost functions of interest).

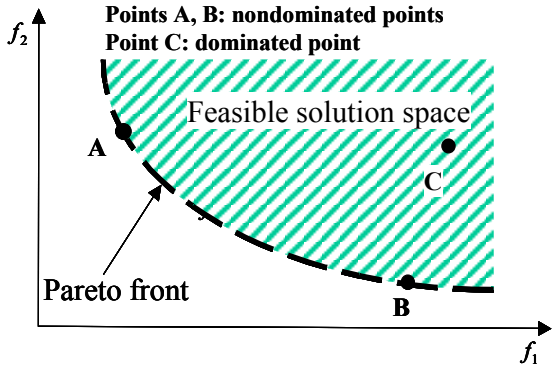


Figure 1.1 Graphical illustration of the Pareto optimality of a two-objective minimization problem

Conventional optimization techniques, such as gradient-based and simplex-based methods [3], and less conventional ones, such as simulated annealing [4] and tabu search [5], are difficult to extend to solve MOPs, because they were not designed with multiple solutions in mind. In practice, MOPs have to be reformulated as a single objective function prior to optimization, leading to the production of a single solution per run of the

optimizer. In literature, weighting objectives method [3,6], goal programming method [7-9] and Min-Max optimum method [10] are some representative decision making algorithms combined with conventional optimization techniques above to achieve a *single* solution in multiobjective optimization problems.

Evolutionary Algorithms (EAs) [11] have been recognized to be well suited to multiobjective optimization early in their development. In EAs, multiple individuals can search for multiple solutions in parallel, eventually taking advantage of any similarities available in the family of possible solutions to the problem. The ability to handle complex problems, involving features such as discontinuities, multimodality, disjoint feasible spaces and noisy function evaluations, reinforces the potential effectiveness of EAs in multiobjective search and optimization, which is perhaps *the* problem area where evolutionary computation really distinguishes itself from other algorithms.

Since the 1980's, several Multiobjective Evolutionary Algorithms (MOEAs) have been proposed and applied in MOPs [2]. These algorithms share the same purpose—approximate a uniformly distributed, *near-optimal* and *near-complete* Pareto front for a given MOP. However, this goal is very difficult to be achieved because the true Pareto front is a high-dimensional solution set instead of a single solution point, which is much more complicated than many single objective optimization problems. Generally, the approximation of the Pareto-optimal set involves two objectives: the distance to the true Pareto front is to be minimized while the diversity of the generated solutions is to be maximized [9]. Unfortunately, these two objectives are also contradictive. In one respect,

Evolutionary Algorithms encourage those better-fit individuals to restrict their searching efforts within local areas in order to search for solutions with even higher fitness values. On the other hand, most of the MOPs require the computational resources to be homogenously distributed in a high dimensional search space to maintain the diversity of resulting population. For this reason, a Pareto-based fitness assignment (ranking scheme) and a density estimation method are usually designed in some existing MOEAs [12-14] in order to guide the search towards a *near-complete* approximation of the ideal Pareto optimal front. Although some of the most advanced MOEAs have been shown to be able to solve some of the challenging multibojective optimization problems, several critical issues are still not well attended in both algorithm domain and problem domain. Therefore, the goal of this research is to study the characteristics of MOPs and exploit the advantages and disadvantages of the existing MOEAs; and propose some feasible innovations in MOEA designs in order to develop a *state-of-the-art* MOEA for practical uses in real-world multiobjective optimization applications.

The remainder of this dissertation is organized as follows. Chapter II introduces Evolutionary Algorithm (EAs) and its categories. As the most representative algorithm in EAs, genetic algorithm is reviewed in details. Its operation procedure, advantages over traditional heuristic optimization algorithms and open issues are also discussed in Chapter II. Chapter III defines multiobjective optimization functions and Pareto optimality. Three traditional decision making approaches for multiobjective optimization are highlighted therein. Chapter IV reviews existing literature on several well-regarded MOEAs and the incorporated characteristics applied by these MOEAs (e.g., fitness

assignment, diversity maintenance, and elitism). A Rand-Density based Genetic Algorithm (RDGA) is proposed and its main design procedures are discussed in Chapter V. In Chapter VI, based on the study of the challenging characteristics embedded in different types of MOPs, several representative MOEAs along with the proposed RDGA are examined by four benchmark MOP test functions. The results show that RDGA is competitive, or even superior to, the other MOEAs in terms of finding a near-complete and near-optimal set of Pareto points. Additionally, as a real application, a Radial-Basis Function Neural Network (RBFNN) design problem is formulated as a bi-objective MOP and an RDGA with hierarchical chromosome representation is implemented in order to search for a set of non-dominated neural network candidates to predict a chaotic time series. Chapter VII explores a study on dynamic population strategies in MOEA. Based on RDGA, a Dynamic Multiobjective Evolutionary Algorithm (DMOEA) is designed. In one aspect, a population growing strategy is proposed in order to encourage all of the created individuals contribute their valuable schemas adequately. On the other hand, those ill-performed and outdated individuals are eliminated from generation to generation to control the computation cost by preventing the explosion of the population size. By examining the selected performance indicators on a benchmark problem, DMOEA is found to be efficient and effective in regulating an optimal population size, keeping the diversity of the individuals along the trade-off surface, tending to extend the Pareto front to new areas, and finding a well-approximated Pareto optimal front. Additionally, dynamic population mechanism eliminates the guesswork from heuristically assigning an initial fixed population size. Based on the study of MOEAs, in Chapter VIII, a module-based, user-friendly MOEA toolbox is designed. Since an MOEA can be divided into

several crucial building blocks, such as ranking methods, density estimation approaches, fitness assignment strategies, elitism schemes and some other necessary routines. Synergistic combinations of these building blocks can result in different types of MOEAs existed, or even some novel ones. Therefore, a module-based toolbox can provide designers with flexibility in dealing with different types of MOPs with their favorite design procedures. In Chapter XI, a new class of evolutionary algorithm—Particle Swarm Optimization (PSO) is introduced. Based on PSO's characteristics of faster convergence, a Dynamic Particle Swarm Multiobjective Optimization (DPSMO) algorithm and a Dynamic Particle Swarm Evolutionary Algorithm (DPSEA) are devised. From simulation results, although DPSMO can significantly improve the efficiency of evolutionary process, it may also produce relatively poorer quality of final Pareto front comparing to DMOEA. However, DPSEA shows great potential in improving both efficiency and efficacy of evolutionary process, which makes DPSEA a potential approach for time varying or even real-time multiobjective optimization problems. Finally, Chapter X concludes this report with a few pertinent observations and proposes future research directions in the field of evolutionary algorithms in mutiobjective optimization problems.

II. EVOLUTIONARY ALGORITHMS

2.1 Overview of Optimization Algorithms

In general, optimization (or search) techniques can be classified into two categories [15]: enumerative (deterministic) and stochastic (random). Table 2.1 shows common examples of each type.

Table 2.1 General optimization approaches

Enumerative (Deterministic)	**Stochastic (Random)**
Greedy	Random Search (Walk)
Hill-Climbing	Simulated Annealing
Branch & Bound	Monte Carlo
Depth-First	Tabu Search
Breadth-First	Evolutionary Algorithms
Best-First	
Calculus-Based	
Mathematical Programming	

Enumerative schemes are perhaps the simplest search strategy—each possible solution is evaluated within some defined finite search space. However, it is apparent that this technique will be inefficient or even infeasible as search space becomes extremely large. Since many real world problems are computationally complex, some means of limiting the search space must be implemented to find "acceptable" solutions within "reasonable" time. Deterministic search attempts this by incorporating problem domain knowledge. Many of these are considered as graph/tree search algorithms, such as greedy algorithms, hill-climbing, branch & bound, etc [16-17,4,18]. Although these techniques had been successfully used in solving a wide variety of problems [16,19-20], they have

difficulty to deal with problems involving high-dimensionality, multi-modality, or NP-Complete characteristics According to [15], the problems exhibit one or more of these characteristics are termed *irregular* [21].

Because enumerative and deterministic techniques are unsuitable for the *irregular* optimization problems, stochastic search and optimization approaches are developed as alternative approaches for solving these irregular problems. These methods include Random Search, Simulated Annealing, Monte Carlo, Tabu Search and Evolutionary Algorithm (EA). Stochastic methods require a function assigning fitness values to possible solutions and an encode/decode mechanism between the problem and algorithm domains. In general, they provide good solutions to a wide range of optimization problems that traditional deterministic search methods find difficult [19].

2.1.1 Random Search (*Walk*)

A random search is the simplest stochastic search strategy, as it merely evaluates a given number of randomly selected solutions. A random walk is similar except that the next solution is randomly selected by using the last evaluated solution as a starting point [22]. Random searches can generally expect to do no better than enumerative ones [19].

2.1.2 Simulated Annealing

Simulated Annealing is an algorithm explicitly modeled on an annealing analogy. For example, a liquid is heated and then gradually cooled until it freezes and a “moving”

will be chosen randomly. If the "moving" improves the current optimal point, it is always executed; otherwise it will be executed with some probability**.** This probability exponentially decreases either by time or with the amount by which the current optimum is worsened [4]. If the liquid's temperature is cooled slowly enough, it will attain a lowest energy configuration. Therefore, basic mechanism of Simulated Annealing is to obtain the global optimum if the "moving" probability decreases slowly enough.

2.1.3 Monte Carlo

In general, Monte Carlo methods involve simulations dealing with stochastic events; they employ a pure random search where any selected trail solution is fully independent of any previous choice and its outcome [5]. The current "best" solution and associated decision variables are stored as a comparator. In the next step, the "best" solution may be updated, and so on.

2.1.4 Tabu Search

Tabu Search is a meta-strategy developed method in order to avoid getting "stuck" on local optima. It keeps a record of both visited solutions and the "path", which reached the solutions in different "memories". This information restricts the choice of solutions to evaluate in the next step. Tabu search is often integrated with other optimization methods [5].

All these approaches are single-point-based methods, which is significantly different from the population-based searching scheme used by Evolutionary Algorithm.

2.2 What is an Evolutionary Algorithm?

The principle of evolution is one of the most general conceptions of biology, which links every organism together in a historical chain of events. Every creature in the chain is the product of a series of "accidents" that have been sorted out thoroughly under selective pressure from the environment. Over many generations, random variation and natural selection modify the characteristics of individuals and species to fit the demands of their living environments. This fit has no intrinsic purpose—it is only the effect of natural variation acting upon and within populations and species and it makes evolution capable of "engineering" solutions to the problems of survival.

What advantages does the evolutionary process offer when applied to engineering problems? It could provide a means for solving problems that are difficult, if not impossible, to traditional algorithms. Indeed, the field of evolutionary computation is one of the fastest growing areas in computer science and engineering simply because of this reason [17]. Engineers and scientists with quite different backgrounds have come together to tackle some of the most difficult problems using this very promising set of stochastic search algorithms, Evolutionary Algorithms (EAs) [23,24].

2.3 Classification of Evolutionary Algorithms

There are three main types of EAs: Genetic Algorithm (GA) [11,25], Evolutionary Programming (EP) [26,27] and Evolutionary Strategies (ES) [28,5]. Each type has numerous variants due to different parameter settings and implementations. Which EA is the best depends upon the problem. There is no universally best algorithm

that can achieve optimal performance for all problems. Different representations or encoding schemes, selection schemes, and search operations will define different EA. For example, GA normally uses crossover and mutation as search operators, while ES only involves mutation. GA often emphasizes genetic evolution, while EP pays more attention to the evolution of behavior. Table 2.2 illustrates the key implementation differences among GA, ES and EP.

Table 2.2 Comparison of three major types of evolutionary algorithms

EA Type	Representation	Evolutionary Operatotors
GA	Normally binary; Real values can be adopted	Mutation, recombination, crossover and selection
ES	Real values and Strategy parameters	Mutation, and $(\mu + \lambda)$ or (μ, λ) selection [24]
EP	Real values	Mutation and $(\mu + \lambda)$ selection alone

In this study, due to its flexibility in solving complex optimization problems, genetic algorithm is chosen as a preferred searching algorithm. Moreover, as both GP and ES are originated from GA [23], we will mainly discuss the characteristics of genetic algorithms in this chapter.

2.4 Genetic Algorithm

The basic principles of Genetic Algorithm (GA) were first proposed by Holland [29] in 1970's. Thereafter, a series of literature becomes available [25,30-32]. GA is inspired by the mechanism of natural selection proposed by Darwin, in which better-fitted individuals are more likely to be the winners in a competing environment, or so called "survival of fittest law." GA uses a direct analogy to natural evolution

characteristics, where the optimal solutions can be evolved and represented by the final winners of the genetic process. Generally speaking, a GA is defined by the following four elements: representation, fitness evaluation, selection, and genetic operations. The whole process is described in Table 2.3.

Table 2.3 A standard genetic algorithm process

*1. Generate the initial **population** P(0) at random, and set iteration index i=0;*
2. REPAET
*(a) **Evaluate** the fitness of each individual in P(i);*
*(b) **Select** parents from P(i) based on their fitness in P(i);*
*(c) Apply **genetic operations** to the selected parents and obtain next generation P(i+1);*
UNTIL the stop criterion are meet.

2.4.1 Representation

GA presumes that the potential solution of any problem is an individual that can be represented by a set of parameters. These parameters are regarded as the genes of a chromosome and can be structured by a string of values in binary form. The chromosome representation that is encoded from the possible physical solution is called *genotype*; the corresponding physical representation is called *phenotype*. A suitable genetic representation for the given problem is always a critical part of genetic algorithms.

2.4.2 Fitness evaluation

A nonnegative value, generally known as a fitness value, is used to reflect the degree of "goodness" of a chromosome for the corresponding genotype, which would be highly related with its objective value. Fitness evaluation gives the performance of a

given chromosome for a specific objective in the phenotype. This is a very important link between GA and the system it represents.

2.4.3 Genetic selection

After a fitness evaluation, a better chromosome has a higher tendency to survive and reproduce good quality offspring. In a practical GA application, a population pool of chromosomes has to be built. These chromosomes can be randomly set initially. The size of the population varies based on the problem of interest. In each cycle of an evolving process, a given number of parents are selected by a selection routine to generate a mating pool for genetic reproduction.

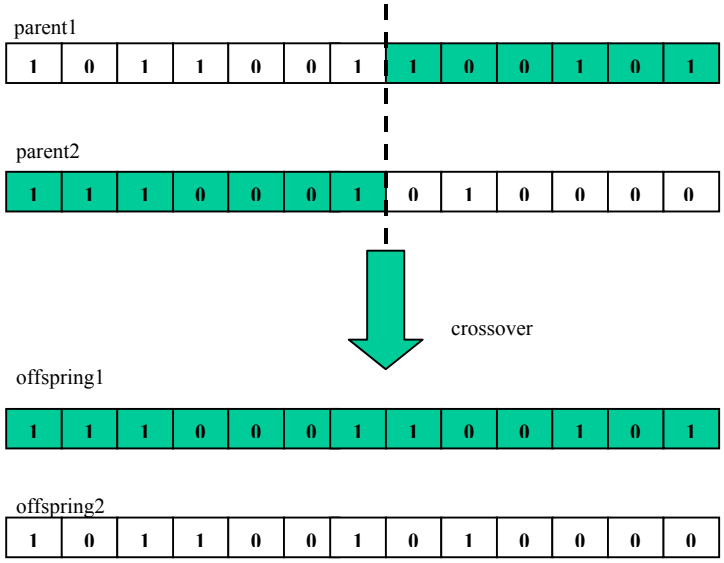


Figure 2.1 Illustration of crossover operation

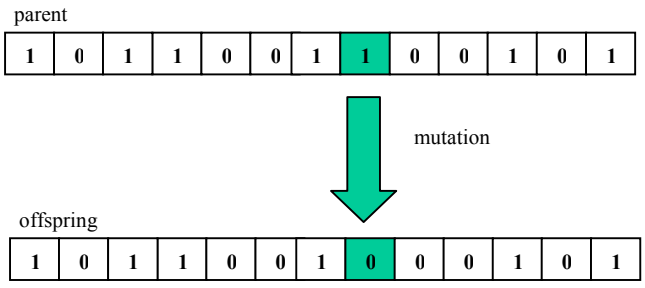


Figure 2.2 Illustration of mutation operation

2.4.4 Genetic operations

In mating pool, the genes of selected parents are mixed and recombined for the production of offspring for a given proportion of the next generation, which is called crossover (Figure 2.1). Mutation is occasionally applied (Figure 2.2), to introduce some new genes into the whole population. It is expected that from this process of evolution (manipulation of genes), the "better" chromosomes will create a larger number of offspring, having a higher chance of surviving in the next generation, and emulating the "survival-of-the-fittest" mechanism in nature.

2.4.5 Stopping criteria

The cycle of evolution is repeated until some desired termination criteria are reached. These criteria can be set by the number of evolution cycles (computational runs), the amount of variation of individuals between different generations, or a predefined value of fitness.

2.5 Difference between GA and Traditional Algorithms

Using GA to solve optimization problems is by far the most active area in evolutionary computation. Compare to those traditional algorithms, the benefits of applying GA in this field are mainly credited to "no assumption" and "parallel searching."

To be applicable, traditional algorithms for discovering the solutions for optimization problems require users to make many assumptions about how to evaluate

the fitness of a solution. For example, linear programming algorithms demand the cost functions to be linear, i.e., a sum of weighted individual cost terms. Another popular approach, the gradient-based search, by which we try to find the point of zero gradients, requires a smooth, differentiable cost function. In addition, it is unable to deal with a cost function having discontinuities. However, GA requires no such assumptions. In GA, the fitness of each individual solution in a population is evaluated and scored; it means one solution must be determined to be better than another in some way. This makes a broad range of problems that are outside the scope of traditional algorithms feasible to genetic algorithms.

Another attractive feature of GA is that it is population based. This makes GA to equip with the ability of parallel searching. In each generation, all the individuals of the population are trying to search in all the directions within the searching space, this allows GA to avoid entrapment in a local optimum and outperform the traditional pure hill-climbing algorithms.

2.6 GA Design and Open Problems

GA has the unique ability to search for and optimize a solution for a complex system. However, due to its evolutionary characteristics, a standard GA may not be flexible enough for practical applications which tend to be complicated, multi-tasking problems with various subgoals. Therefore, a means of modifying the GA structure needs to be made to meet the design criteria.

2.6.1 Chromosome representation

The coding of the chromosome representation may vary according to the nature of the problem. In general, bit string encoding is the most classic method used by GA because of its simplicity and traceability.

Recently, a direct manipulation of real-value chromosomes raised considerable interest. This representation was introduced especially to deal with problems with real parameters. In [33], the result indicated that floating point representation would be faster in computation and more consistent from the basis of run-to-run. At the same time, its performance can be enhanced to achieve a higher accuracy. However, the opinion given by [15] suggested that a real-value coded GA would not necessarily yield better result in some situations. By far, there is not sufficient consensus to support the superiority of either.

2.6.2 Objective and fitness function

An objective function is an assessment mechanism used to evaluate the goodness of a chromosome. Since each individual has a distinguished behavior, the evaluated values vary from one range to another. To maintain uniformity, the objective value, O, is mapped into a fitness value [25], shown in Equation (2.1), with a map Ψ where the domain of F is usually greater than zero.

$$\Psi : O \rightarrow F \tag{2.1}$$

Linear scaling

The fitness value of chromosome *i*, f_i, has a linear relationship with the objective value o_i as:

$$f_i = ao_i + b \tag{2.2}$$

where *a* and *b* are chosen to enforce the equality of the objective value and the average fitness value and cause the maximum scaled fitness to be a specified multiple of the average fitness. This method can reduce the effect of genetic drift to produce a very good chromosome. However, it may introduce a negative fitness value that must be avoided in the GA operation [32]. Thus, the choice of *a* and *b* depends upon the knowledge of the range of the objective values.

Sigma truncation

This method avoids the negative fitness value and incorporates the problem dependent information into the scaling mechanism. The fitness value f_i of chromosome *i* is calculated according to:

$$f_i = o_i - (\bar{o} - c\sigma) \tag{2.3}$$

where *c* is a small integer, $\bar{o}$ denotes the mean of the objective values, and σ is the standard deviation in the population. To prevent negative values of *f*, any negative result (i.e., $f < 0$) is set to zero. The chromosomes whose fitness values are less than $c\sigma$ will not be selected.

Power law scaling

The actual fitness values is taken as a specific power of the objective value, o_i:

$$f_i = o_i^k \tag{2.4}$$

where *k* is problem dependent or even varies during the evolution process [34].

2.6.3 Selection methods

To generate good offspring, an effective parent selection mechanism is essential. The chance of selecting one chromosome to be a parent should be directly proportional to the number of offspring produced. Baker [35] presented three measures of performance for the selection algorithms: *Bias, Spread* and *Efficiency*.

Bias defines the absolute difference between individuals in actual and expected probability of selection. Optimal zero bias is achieved when an individual's probability equals its expected number of trials.

Spread is a range of the possible number of trials that an individual may achieve. If $g(i)$ is the actual number of trials due to each individual *i*, then the "minimum spread" is the smallest spread that theoretically permits zero bias, i.e.

$$g(i) \in [\underline{et(i)},\ \overline{et(i)}] \tag{2.5}$$

where $et(i)$ is the expected number of trials of individual i, and underlined and overlined denote floor and ceiling operators, respectively. Thus, the spread of a selection method measures its consistency.

Efficiency is related to the overall time complexity of the algorithms.

Table 2.4 Rule of Roulette Wheel parent selection

1. *Sum the fitness of all the population members; named as total fitness (Fsum);*
2. *Generate a random number (n) between 0 and total fitness Fsum;*
3. *Return the first individual whose fitness, added to the fitness of the preceding individual, is greater than or equal to n*

By far, many selection techniques employ Roulette Wheel Mechanism as listed in Table 2.4 and shown in Figure 2.3. SSR (Stochastic Sampling with Replacement), SSPR (Stochastic Sampling with Partial Replacement) and SUS (Stochastic Universal Sampling) are three popular roulette wheel selection methods [25].

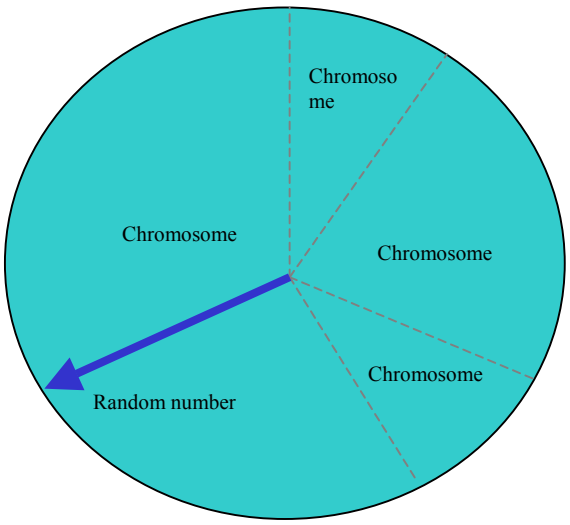


Figure 2.3 Illustration of random Roulette Wheel parent selection indicator

2.6.4 Genetic operation

Schema theory and building block hypothesis

Consider a simple three-dimensional space as shown in Figure 2.4, and assume that the searching space of the solution of a problem can be encoded with three bits; this can be represented as a simple cube with string "000" at the origin. The corners in this cube are numbered by bit strings and all adjacent corners are labeled by bit strings that differ by exactly 1 bit. If "*" represents a "don't care" or "wild card" match symbol, then the front plane of the cube can be represented by the special string "0**". Strings that contain "*" are referred to as schemata and each schema corresponds to a hyperplane in the searching space. A schema represents all strings which match it on all position other than "*". It is clear that each schema matches exactly 2^r strings, where r is the number of don't care symbols, '*', in the schema template. Every binary encoding is a "chromosome" which corresponds to a corner in the hypercube and is a member of the 2^L-1 different hyperplanes, where L is the length of the binary encoding.

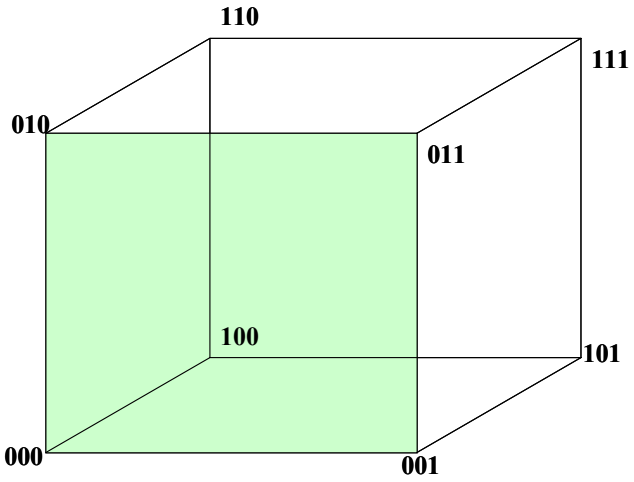


Figure 2.4 Three-dimensional cube to explain schemata

How can genetic algorithm be formulated to search for good schema? Michalewicz indicated, "A genetic algorithm seeks for near-optimal performance through the juxtaposition of short, low-order, high performance schemata, called the building block [31]."

Crossover and mutation

The genetic operations, which are generally referred to as crossover and mutation, have the ability to generate, promote and juxtapose (side by side) building blocks to form the optimal strings. Crossover tends to conserve the genetic information present in the parent strings. Thus, when these strings are similar, their capacity to generate new building blocks decreases. Mutation is not a conservative operator but is capable of generating new building blocks rapidly.

Although one-point crossover method was inspired by biological processes, it has one major drawback in that certain combinations of schema cannot be combined in some situations [25].

For example, assume that there are two high-performance schemata:

$$S_1 = 1\ \ 0\ \ 1\ \ *\ \ *\ \ *\ \ *\ \ 1$$

$$S_2 = *\ \ *\ \ *\ \ *\ \ 1\ \ 1\ \ *\ \ *.$$

There are two chromosomes I_1 and I_2 in the population matched by S_1 and S_2:

$$I_1 = 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 0\ \ 0\ \ 1$$

$$I_2 = 0\ \ 1\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 0.$$

If only one point crossover is performed, it is impossible to obtain the chromosome that can be matched by the following schema (S_3) as the first schema will be destroyed,

$$S_3 = 1\ 0\ 1\ *\ 1\ 1\ *\ 1.$$

A multi-point crossover can be introduced to overcome this problem. As a result, the performance of generated offspring is greatly improved. Another approach is the uniform crossover. This generates offspring from the parents, based on a randomly generated crossover mask. The operation is demonstrated in Figure 2.5. The resulting offspring contains a mixture of genes from each parent. The number of effective crossing points is not fixed, but will be averaged to *L/2* (where *L* is the chromosome length).

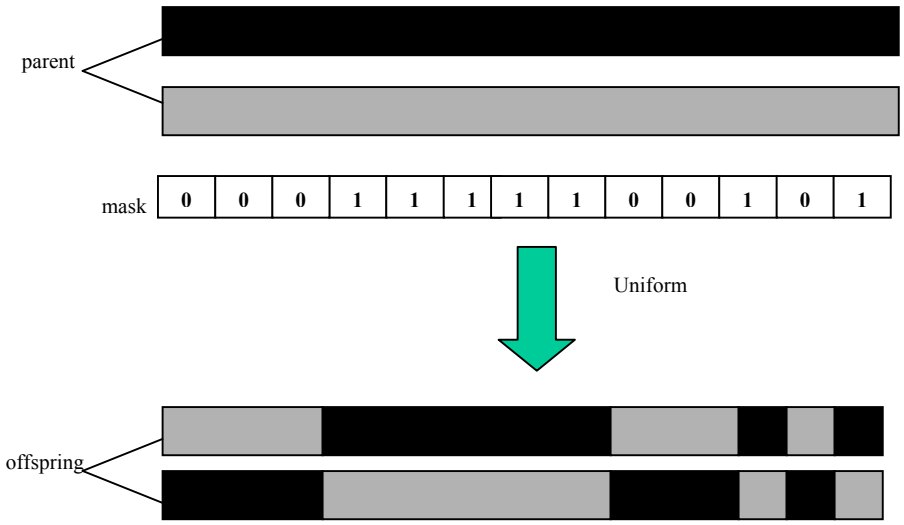


Figure 2.5 Example of uniform crossover

The preference of using which crossover techniques is still a debatable issue. DeJong [36] concluded that a two-point crossover seemed to be an optimal number for multi-point crossover. However, no analytical justification is given. Since the uniform crossover exchanges bits rather than segments, it can combine features regardless of their

relative location. This ability may outweigh the disadvantage of destroying building block solutions and make uniform crossover superior for some problems [37]. Therefore, the crossover technique used to improve offspring production is very much problem dependent. The basic concept in crossover is to exchange gene information between chromosomes. An effective crossover design would greatly increase the convergence rate of the evolutionary process.

Originally, mutation was designed only for the binary represented chromosomes. To adopt the concept of introducing variants into the chromosome, a random mutation [38] for the real number chromosome algorithm was proposed:

$$g = g + \psi(\mu, \sigma) \tag{2.6}$$

where g is the real value gene, ψ is a random function (Gaussian or normally distributed), and μ, σ denote the mean and variance related with the random function, respectively.

Operational rates setting

Another controversial debate for both analytical and empirical investigations is the choice of an optimal probability operation rate for crossover and mutation [31-33]. The increase of crossover probability would promote the recombination of building block and at the same time, it may disrupt the evolutionary process of good chromosomes. On the other hand, increasing the mutation probability would transform the genetic search into a random search, but would reintroduce the lost genetic material.

III. MULTIOBJECTIVE OPTIMIZATION

3.1 Introduction

In engineering practices, it is often a challenge to formulate a design when there are several criteria or design objectives to be met simultaneously. If the objectives are conflicting, then the problem becomes one of finding the best possible design that satisfies the conflicting objectives under different trade-off scenarios. With these multiple objectives and constraints taken into consideration, an optimum design problem can then be formulated. This type of problem is known as a *multiobjective*, *multicriteria*, or *vector optimization* problem.

Leibniz (1646-1716) and Euler (1707-1783) used infinitesimal calculus to find the extreme values of functions. This made it possible for researchers to study various new fields of mechanics. J. Bernoulli (1655-1705), D. Bernoulli (1700-1782), and Sir Isaac Newton (1643-1727) used these methods to lead them into their findings; Newton in minimizing the resistance of a revolving body while the Bernoulli's in solving isoperimetric problems. Lagrange (1736-1813) and Hamilton (1805-1865) developed several theorems that serve as the basis for the solution of all optimum design problems. Later, function approximations were developed by Rayleigh (1842-1919), Ritz (1878-1909), Galerkin (1871-1945) and others to solve complicated time-consuming functions, because they could be approximated relatively accurately.

A French-Italian economist named Pareto (1848-1923) first developed the principle of multiobjective optimization for use in economics. His theories became collectively known as Pareto's optimality concept.

3.1.1 Problem solution

SO solution

A Multiple-Objective (MO) optimum design problem is solved similarly to the Single-Objective (SO) problem. In a SO problem, the idea is to find a set of values for the design variables that, when subject to a number of constraints, yields an optimum value for the sole objective (or cost) function.

MOP ideal solution

In MOPs, the designer tries to find the values for the design variables, which optimize multiple objective functions simultaneously, in this manner the solution is chosen from a so-called Pareto optimal set. In general, for multiobjective problems the optimal solutions obtained by individual optimization of the objectives (i.e., SO optimization) is not a feasible solution to the multiobjective problem.

3.2 Definition

3.2.1 Design variables

The first step in the optimization process is the formulation of the problem. A mathematical model needs to be developed which will closely describe the behavior of the physical system in all possible situations.

A general multiobjective optimization problem can be described as a vector function **f** that maps a set of *m* parameters (*decision variables*) to a set of *n* objectives

$$\min/\max \ \mathbf{y} = \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_n(\mathbf{x})) \tag{3.1}$$

subject to $\mathbf{x} = (x_1, x_2, \ldots, x_m) \in X$

$$\mathbf{y} = (y_1, y_2, \ldots, y_n) \in Y ,$$

where $\mathbf{x}$ is called *decision vector* which includes *m decision variables*, X is the *parameter space*, $\mathbf{y}$ is the *objective vector* which includes *n objectives*, and Y is the *objective space*.

3.2.2 Constraints

The next step in the formulation of the problem is to identify the constraints. Constraints are conditions that must be satisfied, in order for the design to function according to the physical problems. Constraints are expressed as inequalities and/or equalities.

Inequality constraints

Inequalities are usually specified by $\mathbf{g}(\mathbf{x}) \le 0$ (where $\mathbf{g}$ is a vector representing the constraints g_j, $j = 1,...,J$**).** The standard form of an inequality constraint is shown below

$$g_j(\mathbf{x}) \le 0 , \qquad j = 1,...,J . \tag{3.2}$$

Equality Constraints

Equality constraints are shown as $\mathbf{h}(\mathbf{x}) = 0$. In a scalar form they are written as

$$h_k(\mathbf{x}) = 0, \qquad k = 1,...,K \text{ .} \tag{3.3}$$

3.2.3 Objective functions

The final step in the problem statement is to define the objective functions. These are the quantities that the designer wishes to optimize. These functions are expressed as

$$\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_n(\mathbf{x})). \tag{3.4}$$

Sometimes the functions may be defined so that they are all maximized.

$$\max f_i(\mathbf{x}) = -\min(-f_i(\mathbf{x})). \tag{3.5}$$

3.2.4 Standard form

The problem, when written in what is termed the standard form, will appear as follows

$$\min_{\mathbf{x}\in\Re^n} \{f(\mathbf{x}) : h(\mathbf{x}) = 0, g(\mathbf{x}) \le 0\}. \tag{3.6}$$

The above notation can be interpreted as follows: to find the real values of the design variables (i.e., that belong to $\Re^n$), which will result in the smallest values of the objective functions subject to both equality and inequality constraints.

3.3 Pareto Optimal and Traditional Decision Making Methods

3.3.1 Introduction

In a multiobjective optimization problem, we wish to find a set of values for the decision variables that optimizes a set of objective functions. The set of decision variables that produces the optimal result is designated to be the optimal set and is denoted by $\bar{x}^*$. The optimal set is referred to as the Pareto optimal set, and it yields a set of possible answers from which we may choose the desired values of the design variables.

3.3.2 Definition of a Pareto optimum

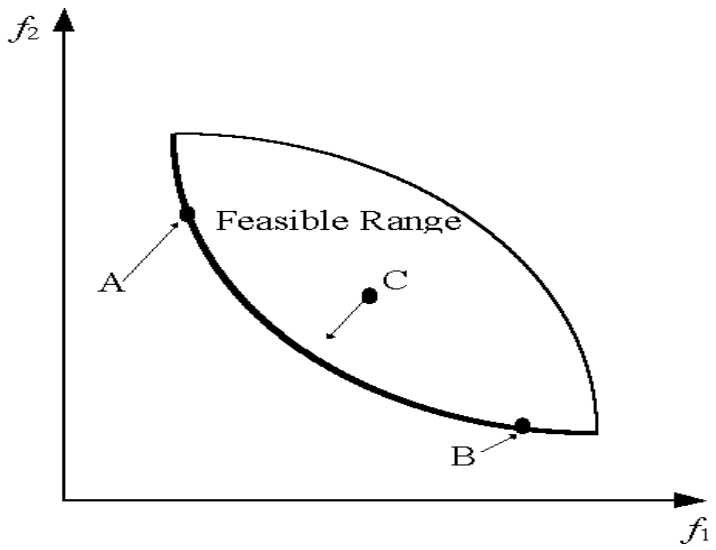


Figure 3.1 Graphical definition of the Pareto optimality

As shown in Figure 3.1, a set of points is said to be Pareto optimal if, moving from one point (e.g., point A) to another point (e.g., point B) in the set, any improvement in one of the objective functions from its current value would cause at least one of the

other objective functions to deteriorate from its current value. Note that, based on this definition, point C is not Pareto optimal.

A more formal definition of Pareto optimality is given as follows [2]. Consider, without loss of generality, the minimization of the n components $f_k, k = 1,\ldots,n,$ of a vector function $\mathbf{f}$ of a vector variable $\mathbf{x}$ in a universe μ, where

$$\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}),\ldots, f_n(\mathbf{x})). \tag{3.7}$$

Then a decision vector $\mathbf{x}_\mu \in \mu$ is said to be Pareto-optimal if and only if there is no $\mathbf{x}_v \in \mu$ for which $\mathbf{v} = \mathbf{f}(\mathbf{x}_v) = (v_1,\ldots,v_n)$ dominates $\mathbf{u} = \mathbf{f}(\mathbf{x}_u) = (u_1,\ldots,u_n)$, that is, there is no $\mathbf{x}_v \in \mu$ such that

$$\forall i \in \{1,\ldots,n\}, v_i \le u_i \qquad \textit{and} \qquad \exists i \in \{1,\ldots,n\} \mid v_i < u_i. \tag{3.8}$$

The set of all Pareto-optimal decision vectors is called the *Pareto-optimal* set of the problem. The corresponding set of objective vectors is called the non-dominated set, or *Pareto front*. Apparently, the Pareto front dominates all other possible solutions and in most cases, it is located on the boundary of the objective vector space (i.e., feasible solution space) as shown in Figure 1.1 for a two-objective optimization problem.

3.3.3 Popular decision making methods

Several methods have been recognized as popular decision-making methods for solving multiobjective optimization problem. Among all of these methods, weighting

objective method, goal programming method and Min-Max optimum method are the most representative ones.

3.3.4 Weighting objectives method

This method [3] takes each objective function and multiplies it by a fraction of one, the "weighting coefficient", which is represented by w_i. The modified functions are then added together to obtain a single cost function, which can be easily solved using any SO method. Mathematically, the new function is written as

$$f(\mathbf{x}) = \sum_{i=1}^{k} w_i f_i(\mathbf{x}) \tag{3.9}$$

where $0 \le w_i \le 1$, and $\sum_{i=1}^{k} w_i = 1$.

If the problem is convex, then a complete set of non-inferior or Pareto solutions can be iteratively found. However, if the problem is not convex, then there is no guarantee that this method will yield the entire Pareto set.

In this method, the weighting coefficients are determined beforehand. The coefficients are then varied to yield a set of feasible optima, the Pareto Optimal set. The designer is expected to pick the values of the variables from this set of solutions.

3.3.5 Goal programming method

This is perhaps the most well known method of solving MOPs [9]. This method was originally developed by Charnes and Cooper [3] and Ijiri [8]. In this method, the

designer must construct a set of goals (which may or may not be realistic) that should be obtained (if possible) for the objective functions. The user then assigns weighting factors to rank the goals in order of importance. Finally a single objective function is written as the minimization of the deviations from these goals.

A "goal constraint" is slightly different than a "real constraint" in goal programming problems. A "goal constraint" is a constraint that needs to be satisfied for the given MOP, but a slight deviation above or below this constraint is acceptable.

3.3.6 Min-max optimum

If one solves for the optimization of each of the objective functions individually, the min-max optimum is the set of points, which will give the smallest values of the relative deviations from the individual objective function [10]. This optimum assumes that each of the objective functions is equally important.

Before the min-max optimum can be defined mathematically, a number of functions must be defined first.

$$z_i(\mathbf{x}) = \frac{|f_i^0 - f_i(\mathbf{x})|}{|f_i^0|} \tag{3.10a}$$

$$z_i^*(\mathbf{x}) = \frac{|f_i^0 - f_i(\mathbf{x})|}{|f_i^*(\mathbf{x})|} \tag{3.10b}$$

$$Z_i(x) = \max\{z_i(\mathbf{x}), z_i^*(\mathbf{x})\} \tag{3.10c}$$

In the above equations, $f_i^0 = \min f_i(\mathbf{x})$ and $f_i^* = \min(\max f_i(\mathbf{x}))$. A point is a min-max optimum if for every $\mathbf{x}$ in the feasible region the following series of steps is satisfied.

Step 1:

$$v_1(x^*) = \min_{x \in X} \max_i \{z_i(x)\} \tag{3.11}$$

where X denotes the decision space. We also define I_1 as the index for the value of $z_i(x)$ which is maximized. If there is another set of solutions $X_1 \subset X$ that meets the requirements for the first step, proceed to the second step.

Step 2:

$$v_1(x^*) = \min_{x \in X_1} \max_{i \notin I_1} \{z_i(x)\} \tag{3.12}$$

Now, $I_1 = \{I_1, I_2\}$, where I_2 is the index at which the value of the z vector is maximized in step 2. The procedure continues on in an iterative manner until there is not a set of solutions which are feasible that satisfy the conditions established in the previous (the second to last) step.

Although these conventional algorithms have some differences in their design procedures, they all are based in a similar spirit that converts a multiobjective optimization problem into a single objective optimization problem. These conversions are always directed by the preferences of the decision-maker. However, from the definition of the Pareto optimality [2], "an MOP tends to be characterized by a family of trade-off solutions, which must be considered equivalent in the absence of the information of the

relevance of each objective relative to the others" [2]. Therefore, with this spirit in mind, Multiobjective Evolutionary Algorithms (MOEAs) have drawn more and more attentions from the researchers in this field.

IV. EVOLUTIONARY ALGORITHMS IN MULTIOBJECTIVE OPTIMIZATION

4.1 Introduction

In Chapter III, several traditional multiobjective optimization methods are introduced. All these methods try to either combine the multiple objectives in an *ad hoc* manner so that a scalar objective function is formed, or turn the objectives into constraints. The goal is to turn multiobjective problems into single-objective problems. Meanwhile, gradient-based or simplex-based optimization techniques are usually applied as a searching tool for the optimal solution, which may result in a local optimum solution for complicated optimization problems.

However, in many real-world multiobjective optimization problems, a suitable solution for the overall problem can hardly be found via the methods outlined in Chapter III since the objectives are different, sometimes even conflicting. Generally speaking, the simultaneous optimization of multiple, possibly competing, and conflicting objective functions are more attractive in that it seldom admits single, perfect solution. Instead, multiobjective optimization problems tend to be characterized by a family of alternatives that must be considered equivalent in the absence of information concerning the importance of each objective relative to others. A suitable solution to problems involving conflicting objectives should offer "acceptable" performance in all objective dimensions, although this solution is possibly sub-optimal for some objectives alone.

In their early development, Evolutionary Algorithms (EAs), a class of population-based optimization approaches, have been recognized to be well suited for multiobjective optimization. In EAs, multiple individuals search for multiple solutions in parallel, advantageously producing a family of feasible solutions to the problem. The ability to handle complex problems involving features such as discontinuities, multimodality and disjoint objective vector spaces, reinforces the potential effectiveness of EAs in multiobjective search and optimization, which is perhaps *the* problem area where EAs most distinguish themselves from the other algorithms [2].

Since the 1980's, several Multiobjective Evolutionary Algorithms (MOEAs) have been proposed and applied in Multiobjective Optimization Problems (MOPs) [13]. These algorithms share the same purpose—approximate a uniformly distributed, near-optimal and near-complete Pareto front for a given MOP. However, this purpose is very difficult to be achieved because the true Pareto front is a high-dimensional solution set, which is much more complicated than many single objective optimization problems combined together. Generally, the approximation of the Pareto-optimal set involves two conflicting objectives: the distance to the true Pareto front is to be minimized while the diversity of the evolved solutions is to be maximized [12]. For the first objective, a Pareto-based fitness assignment (ranking scheme) is usually designed in some state-of-the-art MOEAs [13] in order to guide the search towards the ideal Pareto optimal front. For the second objective, some successful MOEAs provide a density estimation method to preserve the population diversity. In addition, several other techniques have also been adopted such as: elitism scheme [12,14], crowded comparison [14], archive truncation [12] and etc.

Although all of these techniques are very important for MOEAs, the fitness assignment scheme, population density preservation method and elitism archive are considered the most crucial approaches, which have been applied in all the most successful MOEAs.

4.2 Fitness Assignment

In all the current studies of multiobjective evolutionary algorithms, assigning the fitness function is the critical part. Several MOEAs are categorized and different fitness assignment strategies are introduced. In particular, they are distinguished as plain aggregating approaches, population-based non-Pareto approaches, and Pareto-based approaches.

4.2.1 Aggregating methods

Similar to the linear weighting method introduced in the previous chapter, aggregating methods combine the objectives into a single scalar function that is used for fitness calculation. Linear weighting is still used when applying an EA and these aggregation approaches have the advantage of producing one single solution. However, three disadvantages exist in this kind of methods.

- If the objective functions are not commensurable with each other, the weighted combined objective function may cause difficulty to a user in choosing an appropriate set of weighting factors to derive a reasonable solution to the problem.
- Different objective functions may have different ranges of values, thus producing unequal importance to all objective functions. To avoid this issue,

we can normalize the objective functions before solving the optimization equations. However, this approach requires prior knowledge of the lower and upper bounds of each objective function. Unfortunately, this kind of domain knowledge is often not available.

- As mentioned in Chapter II, simple weighting techniques will not be able to respond to problems having non-convex feasible decision space.

The weighted sum approach, target vector optimization, and the method of goal attainment [39] are the most popular aggregation approaches.

4.2.2 Population-based non-Pareto approaches

These approaches are able to evolve multiple non-dominated solutions concurrently in a single simulation run. Known as the Vector Evaluated Genetic Algorithm (VEGA) (Figure 4.1), the method proposed by Schaffer [40] evolves the whole population to several sub-populations in the next generation according to each of the objectives, separately. Crossover and mutation are applied as usual after shuffling all the subpopulations together. Non-dominated individuals are identified by monitoring the population as it evolves. Shuffling and merging all subpopulations correspond to averaging the normalized fitness components associated with each of the objectives. The overall fitness corresponds to a linear function of the objectives where the weights depend on the distribution of the population at each generation. Therefore, different non-dominated individuals are generally assigned different fitness values, in contrast to what the definition of nondominance would suggest.

Fourman [41] proposed a method where selection is performed by comparing pairs of individuals with respect to one of the objectives. In this method, objectives are assigned different priorities by the user and individuals are compared according to the objective with the highest priority. If this results in a tie, the objective with the second highest priority is used, and so on. This is known as the lexicographic ordering, which is a type of goal programming method that was briefly introduced in Chapter III.

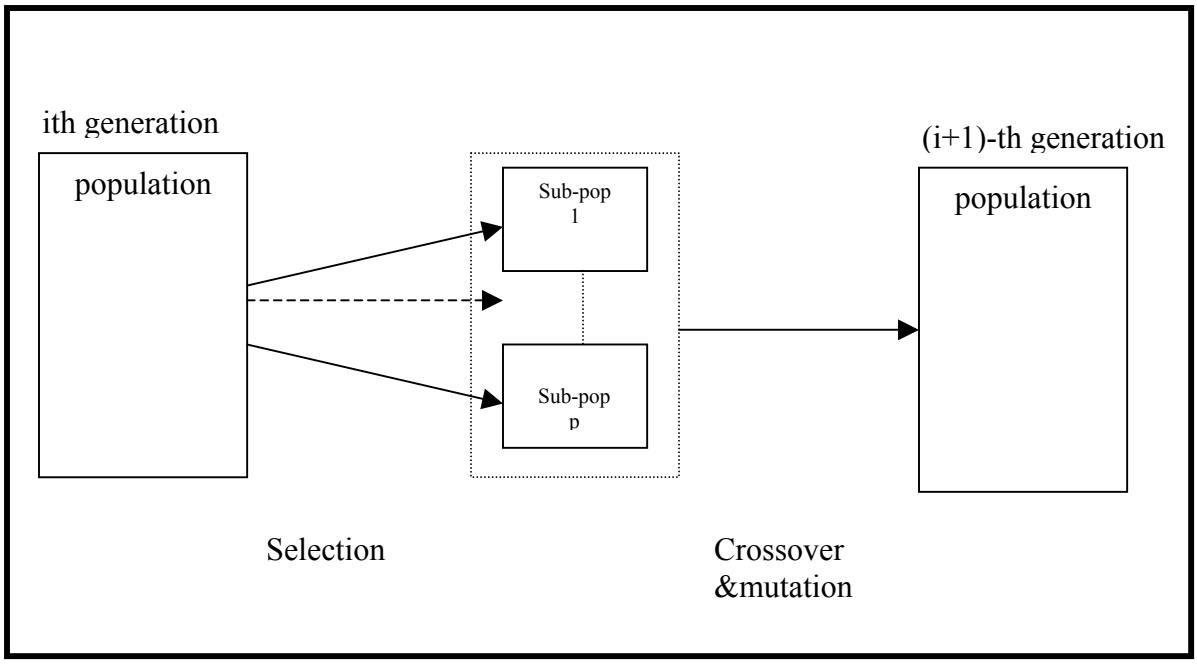


Figure 4.1 Outline of generation replacement of VEGA

VEGA is a pioneering work of multiobjective optimization by GA. However, this approach has difficulties in that it tends to generate the solutions that one of the objectives has extremely good performance at the cost of the others. Furthermore, VEGA can be shown to perform an implicitly weighted sum of the objectives [2]. This leads to the same difficulty found in aggregation genetic algorithms to search for a Pareto front when the problem involves a concave trade-off surface [2].

4.2.3 Population-based Pareto approaches

All the methods mentioned above attempt to promote the generation of multiple non-dominated solutions. However, none of them makes direct use of the actual definition of Pareto optimality. At most, the population is monitored for non-dominated solutions, as discussed in [40].

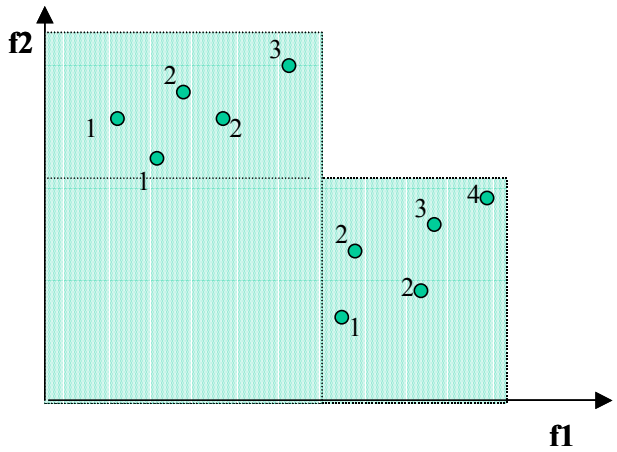


Figure 4.2 Illustration of Goldberg's Pareto-based ranking scheme

Pareto-based fitness assignment was first proposed by Goldberg [25], as a means of assigning equal probability of reproduction to all non-dominated individuals in the population. The method consisted of Pareto-based fitness ranking which assigns rank 1 to the non-dominated individuals and removing them from contention, then finding a new set of non-dominated individuals, ranked 2, and so on (Figure 4.2). This ranking approach was adopted by several MOEAs, including Niched Pareto Genetic Algotithm (NPGA) [42] and Non-dominated Sorting Genetic Algorithm I [43] and II [14] (NSGA I, II).

In the Multiobjective Genetic Algorithm (MOGA) proposed in [44], Fonseca further improved the ranking method by including the density information into the rank value—an individual's rank corresponds to how many individuals in the current population that dominate it. For example, consider an individual y at generation t, which is dominated by $p^{(t)}$ individuals in the current generation. Its rank value is given by [13],

$$rank(y,t) = 1 + p^{(t)}. \tag{4.1}$$

All the non-dominated individuals are assigned rank value 1, while dominated ones are penalized according to the population density of the corresponding region of the trade-off surface. Therefore, by this ranking method, an individual's rank value not only possesses its Pareto dominance status, but also incorporates its density information. This type of ranking scheme will be helpful in preserving the population diversity during the evolutionary process. Figure 4.3 shows the rank values resulted form this ranking method for the same population distribution as shown in Figure 4.2.

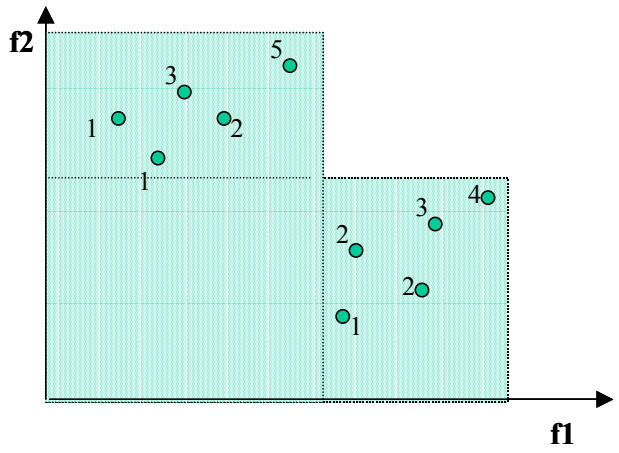


Figure 4.3 Illustration of Fonseca's Pareto-based ranking scheme

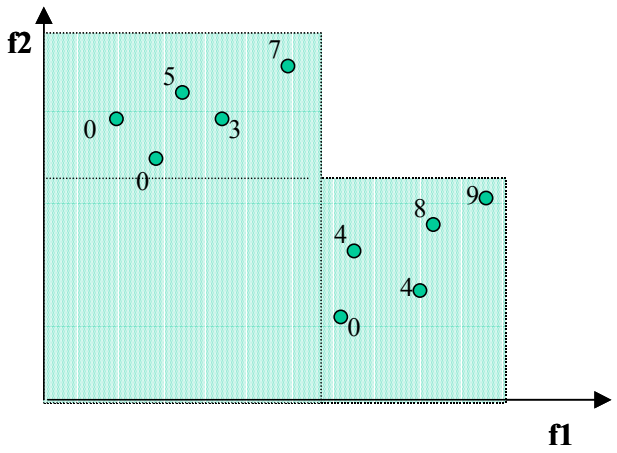


Figure 4.4 Illustration of the Pareto-based ranking scheme adopted by SPEA II

Another well-known MOEA is Strength Pareto Evolutionary Algorithm I [13] and II [12] (SPEA I, II), which devised a "strength" value instead of using the rank value. In SPEA II, a modified fitness assignment strategy based on strength values are proposed in order to overcome some difficulties the existing ranking approach has encountered. In detail, each individual i in the population P is assigned a strength value $S(i)$, representing the number of solutions it dominates:

$$S(i) = |\{j \mid j \in P \wedge i \succ j\}|, \tag{4.2}$$

where $|\cdot|$ denotes the cardinality of a set and the symbol $\succ$ corresponds to the Pareto dominance relation. On the basis of the S value, the raw fitness $R(i)$ of an individual i is calculated:

$$R(i) = \sum_{j \in P, i \succ j} S(j). \tag{4.3}$$

From Equation (4.3), the raw fitness $R(i)$ is determined by the strengths of its dominators in both archive and main population. In addition, similar to Fonseca's

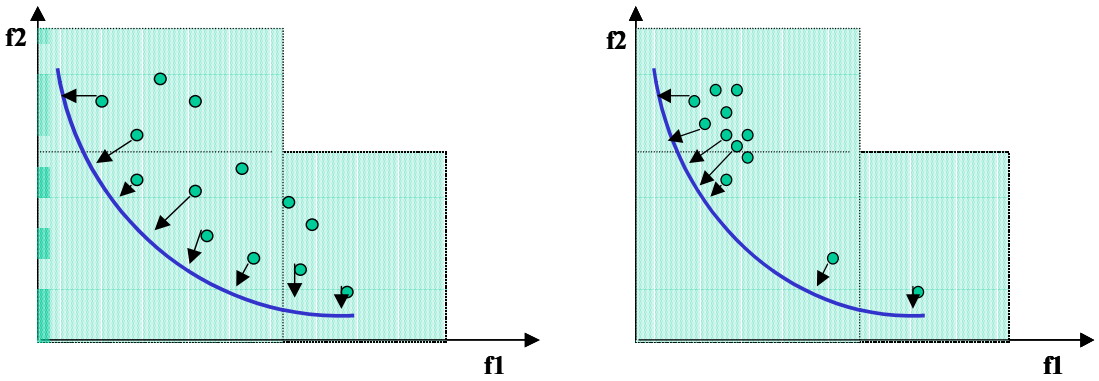
MOGA, the raw fitness values (rank values) produced by this algorithm also include some density information. The rank values resulted by this scheme is shown in Figure 4.4.

Therefore, according to how much preference information is incorporated into the fitness function, the approaches range from complete preference information given, as in combining objective functions directly or prioritizing them, to no preference information given, as in Pareto-based ranking. Which approach is best is determined by the problem to be solved. Although by now, non-informative Pareto-based ranking methods are at the dominant position in this research field, in some cases, partial preference information is also studied to restrict the searching to only one part of Pareto set. Although a specified ranking scheme can maintain the population diversity to some extent based on the concept of Pareto dominance, it may fail when most individuals do not dominate each other. For this reason, ranking scheme still cannot replace a real density preservation strategy. In most state-of-the-art MOEAs, a fitness sharing or density estimation method is always applied and the population density value is optimized as well.

4.3 Maintenance of Diversity

In solving multiobjective optimization problems, it is required that the solutions are Pareto-optimal, and at the same time they are uniformly sampled from the Pareto-optimal set. The Pareto-based approaches mentioned above achieve the first requirement. However, the approaches by themselves cannot meet the second criterion. In most evolutionary algorithms, it is known that the genetic diversity of the population is lost

due to their stochastic selection processes. This phenomenon is called "genetic drift" [45,46], by which genetic algorithms can exploit the "good" individuals and explore better ones by genetic operation. Although "genetic drift" effect has its advantages in single objective optimization, in MOEAs, loss of diversity due to the "genetic drift" needs to be restrained as shown in Figure 4.5.



(a) Population diversity is preserved **(b) Population diversity is not preserved**

Figure 4.5 Illustration of the effect of population diversity preservation

4.3.1 Niched fitness sharing technique

To maintain the diversity, a technique, so called "fitness sharing", is widely used [25]. In the fitness sharing method, the fitness value of each individual is reduced if there exists other individuals in its neighborhood. Therefore an individual located in a more crowded area leaves less offspring [42]. Thus, we can obtain a population distributed more uniformly over the Pareto-optimal set. Niche induction [42] technique is one of the representative fitness sharing methods that is adopted by Niched Pareto Genetic Algorithm (NPGA). In NPGA, a niche radius is chosen and individuals within the distance defined by the niche radius degrade each other's fitness, since they are in the same niche (shown in Figure 4.6). Thus the convergence occurs within a niche, but the

convergence of the whole population is avoided. Based on this fitness sharing technique, the more individuals a niche contains, the more its members' fitness values degrade.

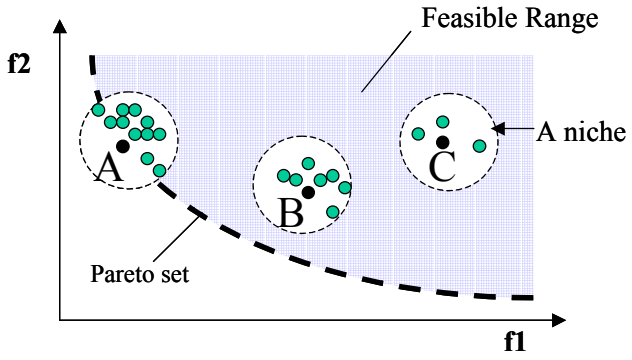


Figure 4.6 Illustration of niched fitness sharing technique

Since NPGA only applies Pareto selection to a portion of the entire population in each generation, it is relatively fast compared to the other Pareto-based approaches. In addition, it can produce good non-dominated solutions that can be kept for a large number of generations. Currently, many MOEAs implement niched fitness sharing strategies (e.g., [46-49]). The limitation of NPGA is that it requires heuristic choices of the sharing factor and the size of the tournament, which makes the process relatively complex in practice. Moreover, as the sharing technique degrades the fitness value, "harmful" individuals may be generated that may slow down the speed of the entire population to evolve in a correct direction to the Pareto front [50].

4.3.2 Density estimation technique

Some newly developed MOEAs apply a "density estimation" technique in order to provide a density value to each individual. The density value represents the crowdness of the area the interested individual located in. Crowding distance assignment and *k*-th

nearest neighbor methods belong to this category and have been used in NSGA-II and SPEA II, respectively.

In NSGA-II, to obtain an estimate of the density of individuals surrounding a particular point in the population, the average distance of two neighboring points on either side of the concerned individual along each dimension is taken. This quantity i_{dist} serves as an estimate of the size of the largest cuboid enclosing the individual i without including any other point in the population, which is called *crowding distance.* As shown in Figure 4.7, the crowding distance of the ith solution in its front (marked with dark points) is the average side length of the cuboid.

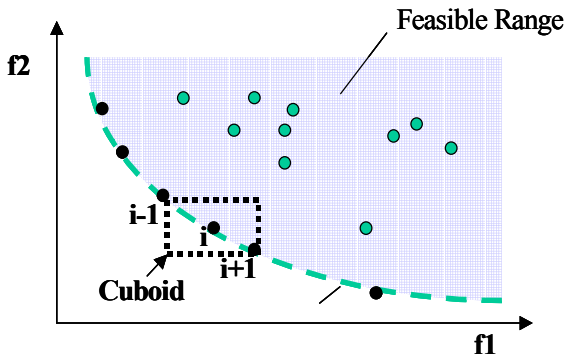


Figure 4.7 Illustration of crowding distance estimation approach

The density estimation technique used in SPEA II is an adapted *k*-th nearest neighbor method, where the density at any individual is a decreasing function of the distance to its *k*-th nearest neighbor (data point). The density estimate is taken as the inverse of the distance to the *k*-th nearest neighbor, which is denoted as σ_i^k. In SPEA II, k is set to be equal to the square root of a sample size N, thus, $k = \sqrt{N}$, and the density $D(i)$ corresponding to i is defined by

$$D(i) = \frac{1}{\sigma_i^k + 2}, \tag{4.4}$$

where two is added to ensure that $D(i)$ value is greater than zero and less than 1/2.

4.4 Fitness Assignment Scheme of NSGA-II and SPEA II

As two of the most recent and successful MOEAs, both NSGA-II [14] and SPEA II [12] clearly classified individual Pareto rank value and density value as two major fitness. However, their fitness assignment schemes are totally different. In fitness assignment, between two individuals, NSGA-II used a tournament scheme, by which NSGA-II prefers the point with a lower rank value, or the point located in a region with less numbers of points if both of the points belong to the same front. However, SPEA II calculates the fitness value for each individual by simply adding density value $D(i)$ to the raw fitness $R(i)$. Considering the ranking and density estimation schemes of different MOEAs, it is impossible to state that which ranking or density scheme is the best without synergistically integrating them together by an appropriate fitness assignment. On the other hand, according to the No Free Lunch (NFL) theorem [51], no formal assurances of an algorithm's general effectiveness exists if insufficient knowledge of the problem domain is incorporated into the algorithm domain. A study of benchmark MOP itself to exploit specific problem characteristics is also an important issue, which will be discussed in Chapter VI.

4.5 Other Significant Techniques Used in MOEAs

In order to improve the performance of an MOEA, several interesting techniques are also designed by different researchers. Among them, elitism scheme, mating restriction, and archive truncation are the most significant ones.

4.5.1 Elitism scheme

Originated from Evolutionary Strategy (ES), elitism scheme has been applied by almost all of the advanced MOEAs [12-14] in that it can further improve the performance of the resulting solutions. In detail, an archive with a fixed number of elitists will be set up besides the main population and the non-dominated individuals generated by the main population will be considered as a set of elitists and kept into the archive. Additionally, at each generation, a certain number of elitists will be copied into the main population to perform crossover. Therefore, by this two-way communication method, the elitist's archive will be updated generation by generation and the valuable schemas of an elitist can also be inherited by their offspring. For this reason, elitism scheme has the potential to help the entire population converge into a near-optimal Pareto front.

By now, Pareto Archive Evolutionary Strategy (PAES) is the one of the most successful MOEAs whose performance mainly depends on elitism. As a local search algorithm that simulates a random mutation hill-climbing strategy, PAES may represent the simplest possible, yet effective, nontrivial algorithm capable of generating diverse solutions in the Pareto optimal set [52]. In PAES, pure mutation operation is adopted to fulfill local search scheme. A reference archive of previously found non-dominated solutions is updated at each generation in order to identify the dominance ranking of all

the resulting solutions. Although (1+1)-PAES is originated as the simplest version, PAES can also generate λ mutants by mutating one of the μ current solutions, which is called $(\mu + \lambda)$-PAES [52]. Since PAES does not perform population-based search, only tournament selection can be applied to determine the survivors of the next generation. It is worthy to mention that although the archive size has to be pre-determined, PAES implements a population incrementing scheme by continuously adding new non-dominated individuals to the archive.

4.5.2 Mating restriction

The variability of mating is another important aspect as the population distributes itself around multiple regions of optimality. Different regions of the trade-off surface generally have very different genetic representations, which constrain mating to happen only locally to ensure viability [53]. So far, mating restriction has only been implemented based on the distance between individuals in the objective domain, either directly or indirectly. The use of mating restriction in multiobjective GAs does not appear to be widespread.

4.5.3 Archive truncation

In elitism scheme, an elitist's archive needs to be updated by comparing new introduced elitist with the existing ones in order to keep the archive size fixed. Therefore, an archive truncation technique is designed in SPEA II [12]. By this technique, an elitist that has minimum distance to another elitist is chosen at each stage as a member of the

archive, if there are several elitists with the same minimum distance, the tie is broken by considering the second smallest distances and so forth.

V. RANK DENSITY BASED MULTIOBJECTIVE GENETIC ALGORITHM

5.1 Introduction

From the literature review, the primary difficulty in the existing MOEAs lies on designing a suitable fitness assignment strategy in order to search for a *near-complete* and *near-optimal* approximated Pareto front for the given optimization problem. Unfortunately, these two objectives are contradictory. In one respect, the "genetic drift" character needs to be exploited to converge the solution to a nearly optimal point. On the other hand, the "genetic drift" phenomenon must be avoided in order to sketch a uniformly sampled trade-off surface for the final Pareto front. Based on these considerations, two of the best-known MOEAS [12-14], (i.e. NSGA-II and SPEA II) attempt to represent the fitness value of an individual by a Pareto rank value and a density value, and then optimize these two sub-fitness values using a specified assignment method. However, there remain several deficiencies in these algorithms. Especially, both NSGA-II and SPEA II do not treat rank value and density value equally in their selection process. In NSGA-II, Pareto rank value is considered more important than density value and the parent selection is mainly based on the rank value, whereas density value is merely treated as a reference in the tournament selection. SPEA II combines the rank and density values into a single fitness value by using a linear weighting method. Although the weights of rank and density are equal, there still exists a bias to rank value calculation because the maximum density value cannot be higher than 0.5 according to SPEA II. For this reason, the density value can hardly be minimized until the rank value has almost

converged. Therefore, both algorithms prefer taking advantages of "genetic drift" effect than controlling it, which may result into difficulties to find a uniformly distributed Pareto front.

To respond to these deficiencies, a Rank-Density based Genetic Algorithm (RDGA) [54], which synergistically integrates selected features of existing MOEAs in a unique way, is proposed. Although RDGA also converts a high dimensional MOP into a bi-objective optimization problem to minimize fitness rank values and cell densities, it adopts several additional techniques in order to achieve a *near-complete* and *near-optimal* Pareto front [55].

5.2 Critical Procedures of RDGA Design

There are five crucial procedures involved in RDGA design, which are discussed as follows.

5.2.1 Automatic Accumulated Ranking Strategy (AARS)

In RDGA, we propose an Automatic Accumulated Ranking Strategy (AARS). In AARS, an individual's rank value is defined as the summation of the rank values of the individuals that dominate it. For example, assuming at generation t, individual y is dominated by $p^{(t)}$ individuals $y_1, y_2, \cdots, y_{p^{(t)}}$, whose rank values are already known as $rank(y_1,t)$, $rank(y_2,t)$, $\cdots$, $rank(y_{p^{(t)}},t)$. Its rank value can be computed by

$$rank(y,t) = 1 + \sum_{j=1}^{p^{(t)}} rank(y_j,t). \tag{5.1}$$

By AARS, all the non-dominated individuals are still assigned rank value 1, while dominated ones are penalized to reduce the population density and redundancy. For instance, suppose we want to minimize two objectives, f_1 and f_2, and MOEAs generate eleven individuals, and their rank values based on four ranking techniques proposed by NSGA-II [14], MOGA [44], SPEA II [12] and AARS [54] are illustrated in Figure 5.1, where each dot represents a candidate phenotype solution. Considering all the individuals located in the lower-right area, AARS provides the exact same rank values as those computed by pure Pareto ranking method (adopted by NSGA-II [14]) since all the individuals are clearly aligned and not crowded at all. Therefore, adding extra density information (resulted by SPEA II) may not be necessary in this case. Meanwhile, AARS does impose penalty to the dominated individuals located in the upper-left area. The reason of penalizing all the dominated individuals in this area is because there exist several non-dominated individuals that can mostly represent the dominated points. Therefore, without increasing the population size, the population diversity will be maintained by penalizing those dominated individuals in AARS.

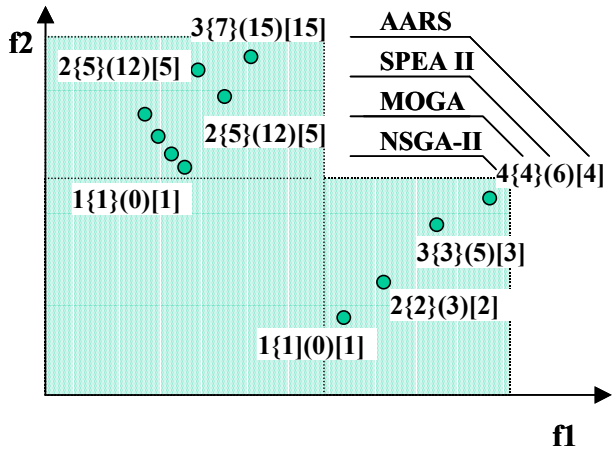


Figure 5.1 Individual rank values resulting from MOGA/NSGA-II/ SPEA II/ RDGA ranking methods

5.2.2 Adaptive density estimation

According to [13], although AARS and other ranking schemes [52,56] provide a sort of niching mechanism based on the concept of Pareto dominance, they may fail when most individuals do not dominate each other. Therefore, additional density information is incorporated to discriminate between individuals having identical raw fitness values. In RDGA, to deal with this problem, we adopt a modified adaptive cell density evaluation scheme originated from [52] as shown in Figure 5.2. The cell width in each objective dimension can be formed as

$$d_i = \frac{\max_{\mathbf{x}\in X} f_i(\mathbf{x}) - \min_{\mathbf{x}\in X} f_i(\mathbf{x})}{K_i}, i = 1,..., n, \tag{5.2}$$

where d_i is the width of the cell in the *ith* dimension, K_i denotes the number of cells designated for the *ith* dimension (i.e., in Figure 5.2, $K_1 = 6$ and $K_2 = 4$), and $\mathbf{x}$ is taken from the whole decision space X. As the maximum and minimum fitness values in objective space will change with different generations, the cell size will vary from generation to generation to maintain the accuracy of the density calculation. The density value of an individual is defined as the number of the individuals located in the same cell. Note that in PAES [52], the grid location of a solution in objective space is obtained by repeatedly bisecting the range in each objective and finding in which half the solution is. However, RDGA uses a different scheme to locate which cell an individual belongs to. First, the cells are created by dividing the range of current objective space based on K_i and given initial population. Second, the center position of each cell will be obtained and stored as a matrix. Third, each individual of initial population will search for its nearest

cell center and identify this cell as its “home address” and consider the other individuals who share the same “home address” as its “family members.” Then for each of these “homes,” the number of “family members” who dwell in it will be counted and saved as its density value. Fourth, when an offspring is generated and accepted, its “home address” can be easily located by following the third step and the density value of its home will increase by one. Meanwhile, if an old individual is removed, its “home” will be notified and the density value of its “home” will decrease by one. Therefore, at each generation, an individual can access its “home address” and then obtain the corresponding density value. The “home address” is merely a “pointer” to inform an individual where to find its density value. For instance, as shown in Figure 5.2, the “home address” and density value of individual **A** are (4,3) and 4, respectively. Therefore, if a new generated or a removed individual does not change the boundary of the range of current objective space, only the density value of its ”home” will changes, the density values of the other “homes” (cells) will not be affected. This setting can avoid the unnecessary recalculation of unchanged range of objective space and density values.

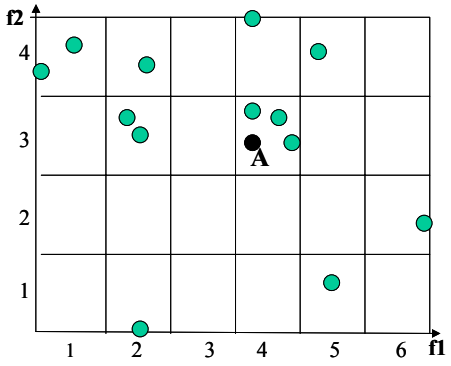


Figure 5.2 Illustration of density map and density grid applied by RDGA

5.2.3 Rank and density based fitness assignment

Because rank and density values represent fitness and population diversity, respectively, we assigned them as two important attributes to each individual. Therefore, any multiobjective optimization problem can be converted into a bi-objective optimization problem. On the other hand, since we need to minimize rank value together with density value, some further modifications need to be made to the original notation.

First, instead of minimizing the density value of an individual, we minimize the density value of the entire population. Based upon the definition of the cell density, an individual located in a crowded cell must have a relatively higher density value, which contributes much more to the population density value than an individual in the sparse area does. For example, a cell containing ten individuals will contribute $10 \times 10 = 100$ to the population density value, whereas a cell containing only one individual will contribute only 1 to the population density value.

Second, after the rank and density values of each individual have been extracted, a modified VEGA is applied to fulfill fitness assignment. As discussed in Chapter IV, VEGA possesses two deficiencies: 1) it does not have a scheme to maintain the diversity of the evolved Pareto front, and 2) it has difficulty in dealing with the problems with concave trade-off surfaces. As mentioned above, the goal of RDGA is to find the non-dominated individuals with the rank value equal to 1 and at the same time reduce the population density value to obtain a uniformly distributed trade-off surface. In this setting, there is no concern about keeping the population diversity in the rank-density

(algorithm) domain. Furthermore, whether the "Pareto front" in the rank-density domain is concave or not is not an issue since it is not a real Pareto front for the MO problem under consideration. Therefore, a simple VEGA is effective enough to fulfill fitness assignment after the original optimization problem has been transformed into the rank-density domain. It is worthy of noting that the idea of converting multiobjective into a domination measure function and neighboring density function was also adopted by Borges and Barbosa [57]. However, in their paper, two newly formulated objective functions were chosen from Goldberg's ranking scheme [25] and Horn's niche sharing method [42]. Afterwards, they combined two objective functions into one non-linear fitness function, which is the final fitness function. Because rank and density values have totally different characteristics, it is very difficult for this algorithm to designate a suitable coefficient in *ad hoc* to bias the preference during the evolutionary process.

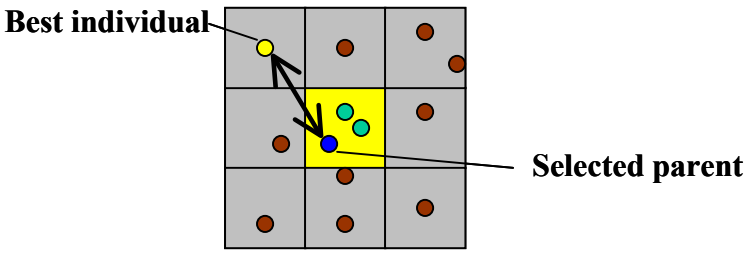


Figure 5.3 Illustration of the "diffusion" scheme

5.2.4 Crossover and mutation operations

For crossover, the parent selection and replacement schemes are borrowed from Cellular GA [53] to explore the new search area by "diffusion" (see Figure 5.3). For each subpopulation, a fixed number of parents are randomly selected for crossover. Then, each selected parent performs crossover with the best individual (the one with the lowest rank value) within the same cell and the nearest neighboring cells that contain individuals. If

one offspring produces better fitness (a lower rank value or a lower population density value) than its corresponding parent, it replaces its parent. The replacement scheme of the mutation operation is analogous.

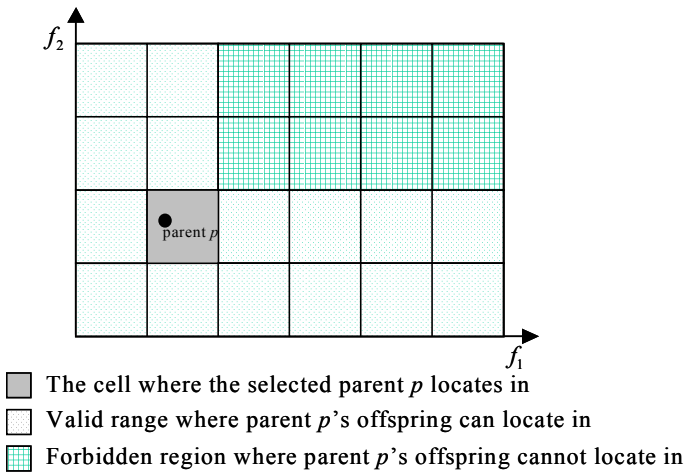


Figure 5.4 Illustration of the valid range and the forbidden region

Meanwhile, as RDGA takes the minimization of the population density value as one of the objectives, it is expected that the entire population may move toward an opposite direction to the Pareto front where the population density value is being minimized. Although moving away from the true Pareto front can reduce population density value, obviously, these individuals are harmful to the population to converge to the Pareto front. To prevent "harmful" offspring surviving and affecting the evolutionary direction and speed, a *forbidden region* concept is proposed in the replacement scheme for the density subpopulation, thereby preventing the "backward" effect. The *forbidden region* includes all the cells dominated by the selected parent. The offspring located in the forbidden region will not survive in the next generation, and thus the selected parent will not be replaced. As shown in Figure 5.4, suppose our goal is to minimize objectives f_1 and f_2, and a resulting offspring of the selected parent p is located in the forbidden

region. By RDGA, this offspring will be eliminated even if it reduces the population density because this kind of offspring has the tendency to push the entire population away from the desired evolutionary direction.

As discussed in Subsection 5.2.1, Automatic Accumulated Ranking Strategy (AARS) includes the scheme of punishing the individuals located in a crowded area, which means we add a bias to avoid the population density value from expanding too much when RDGA is implementing the minimization of population rank values. Meanwhile, a forbidden region is brought in to introduce another bias to prevent the offspring from having higher ranks than their parents when RDGA is evolving a lower population density value. Therefore, RDGA can be interpreted as trying to convert an MOP in problem domain into two new single objective optimization problems in algorithm domain—minimizing population rank and density values, and then performing an evolutionary process to optimize each of the objectives in turn. It is necessary to note that these two biases make two objectives of RDGA highly correlated. When one objective is being optimized, the corresponding bias will take the other objective as a constraint to keep the computation resources homogeneously distributed between two objectives.

5.2.5 Constraint handling

To handle the constraints, every new generated offspring will be tested against all the constraint functions in order to determine if it is a valid solution. If the offspring satisfies for all the constraints, it will be evaluated by the fitness function to obtain its fitness value, otherwise, it will be discarded.

5.2.6 Elitism strategy

The elitism scheme in [58] is also adopted in RDGA. At each generation, the non-dominated individuals generated from main population will be copied and stored to an archive. Meanwhile, a non-dominated solution in archive may also be selected with a certain probability as a parent to perform genetic operations. This probability p_e^t is called "elitism intensity" and according to [13], at each generation *t,* the probability of sampling an individual from the archive is given by

$$p_e^t = 1 - \left(\frac{|B|}{|A| + |B|}\right)^2, \tag{5.3}$$

where A and B represents archive of elitists and main population, respectively. After the evolution process has terminated, the resulting solutions in both main population and archive will be compared to derive the final Pareto front.

VI. BENCHMARK TEST FUNCTION STUDY AND EXPERIMENTAL RESULTS

6.1 Introduction

According to [15], in order to compare the performance of different MOEAs, the design of a variety of MOP benchmark problems and performance metrics is essential. Because a multiobjective optimization problem can be closely related to a combination of Single objective Optimization Problems (SOPs), some literature review on the features of SOP test functions can be helpful. In De Jong's SOP test bed study [36], he declared that six problem characteristics need to be examined: continuous and discontinuous, convex and non-convex, uni-modal and multi-modal, quadratic and non-quadratic, low and high dimensionality, and deterministic and stochastic. In addition, Michalewicz [59] addressed other issues that need to be considered for SOP test bed design, such as the number of constraints, type of constraints and ratio between the feasible and complete search space. Apparently, some of these properties are also valuable for an MOP and must be incorporated into the test bed design. Nevertheless, because the purpose of solving an MOP is to find a near-complete set of non-dominated solutions (Pareto front), the features that cause the true Pareto front difficult to be found are the primary concerns in MOP test function design. Therefore, we focus our investigation on five distinct features of a Pareto front. They are discontinuity, concavity, global/local optimality high-dimensional decision space and high dimensional objective space. In addition, since a neural network design problem can be considered as a bi-objective MOP, RDGA is applied to design a Radial Basis Function (RBF) neural network.

6.2 Performance Merit Indicator Design

Five MOEAs— MOGA, PAES, NSGA-II, SPEA II and the proposed RDGA are deployed in the simulation and run each of the algorithms for 50 times to obtain the statistical results. For each run, a new initial population with 100 individuals is randomly generated and used by each of four population-based MOEAs (i.e., MOGA, NSGA-II, SPEA II and RDGA), while only one initial individual is generated for PAES according to its design procedure [52] and the archive size is set to be 100 for all the selective MOEAs that involve elitism scheme. We use three indicators derived from final generation of 50 runs to benchmark the comparison results via statistical Box plots. They are: average individual rank value, average individual density value and average individual distance. As discussed in Chapter V, for an individual, different ranking schemes will produce different rank values, which will be used in respective fitness evaluations and selections. However, for a fair comparison in terms of ranking indicators of different MOEAs, we use Goldberg's pure Pareto ranking method [25] to recalculate the rank value for each individual resulted by each applied MOEAs. Meanwhile, as shown in Figure 5.2, the average individual density value is calculated as the mean value of all the individual density values. Here, according to the population size, we choose the number of grids for each objective dimension to be 20. This setting will not change the minimum and maximum individual density values, which are 1 and 100, respectively. Furthermore, because the rank is a relative value, it must be stated that we cannot guarantee the final population will be a true Pareto set even if all its individuals have rank values 1s as shown in Figure 5.3. For this reason, we use "final average individual distance" as the third indicator to measure how far the non-dominated points on the

resulting final Pareto front PF_{final} are away from the true Pareto front PF_{true} as shown in Figure 6.1, where PF_{true} is known in *a priori* for the given test functions in this paper. This indicator was originally introduced by Veldhuizen and Lamont [60], where the final individual distance G is defined as

$$G = \frac{(\sum_{i=1}^{m} d_i^2)^{1/2}}{m}, \tag{6.1}$$

where m is the number of individuals in PF_{final}, and d_i is the Euclidean distance between each of these individuals and a point on PF_{true} that is the closest to it. A result of $G = 0$ indicates the convergence $PF_{final} = PF_{true}$; any other value indicates PF_{final} deviates from PF_{true}.

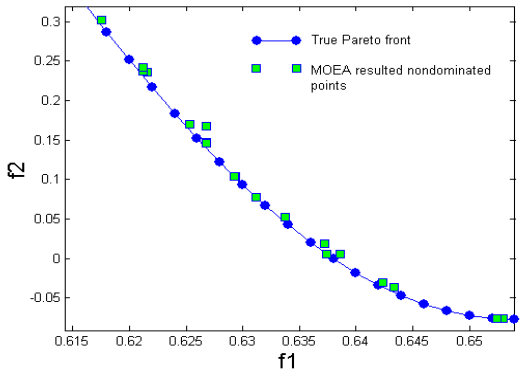


Figure 6.1 Difference between PF_{true} and PF_{final}

Moreover, in order to compare the dominance relationship between two populations resulted by two different MOEAs, the coverage of two sets (C value) [13] is measured to show how the final population of one algorithm dominate the final population of another. Function C maps the ordered pair (X_i, X_j) to the interval [0, 1],

where X_i and X_j denote the final populations resulted from algorithm *i* and *j*, respectively. The value $C(X_i, X_j) = 0$ means that all points in X_j are dominated by or equal to points in X_i. The opposite, $C(X_i, X_j) = 1$, represents the situation when none of the points in X_j are covered by the set X_i. Note that both $C(X_i, X_j) = 1$ and $C(X_j, X_i) = 1$ need to be considered independently since they have the distinct meanings.

Therefore, four indicators represent qualitative measures that describe the quality of the final result of selected MOEAs— the average individual rank value shows the dominated relationship between different individuals, the average individual density value illustrates how good the population diversity is preserved, the average individual distance measures distance between PF_{final} and PF_{true}, which provides the quality of the resulting Pareto front, and the C value compares the domination relationship of a pair of MOEAs. All values of four indicators generated at the final generation are illustrated by Box plots to show the statistical comparison results.

6.3 MOEA Comparison and Genetic Operator Design

To examine the performances of the selected MOEAs and the proposed RDGA on the test functions with different Pareto front features, we explore four numerical test functions in the simulation study. Function ***F1*** is advanced from an existing MOP to create discontinuous and concave Pareto front [61]. Functions ***F2-1*** and ***F2-2*** are designed to explore local and global Pareto optimality caused by objective function and

constraints, respectively. Function ***F3*** and ***F4*** has a high-dimensional decision space, while function ***F4*** involves a high-dimensional objective space. For a fair comparison, the stopping generation, the chromosome length of each decision variable, the crossover rate and the mutation rate are chosen to be 10,000, 15, 0.7, and 0.1, respectively for all population-based MOEAs considered. One point crossover is used for all the population based MOEAs. In addition, we select (1+10)-PAES and a bit flip mutation rate $1/k$ is used for a chromosome of k genes and the tournament size t_{dom} is chosen to be 2.

6.3.1 F1— MOP with discontinuous and concave Pareto front

The rationale of exploiting MOPs with discontinuous and concave Pareto fronts is that some MOEAs using plain aggregating schemes have been proven of having difficulty in finding the Pareto points on the discontinuous and concave segments. MOEA's ability of finding nonconvex Pareto front is one of the most important reasons of using EA's other than traditional gradient-based or simplex-based algorithms in multiobjective optimization.

Here, a modified Tanaka's MOP [61] is chosen to be the test function with a discontinuous and concave Pareto front.

Minimize $f_1(x_1, x_2)$ and $f_2(x_1, x_2)$, where

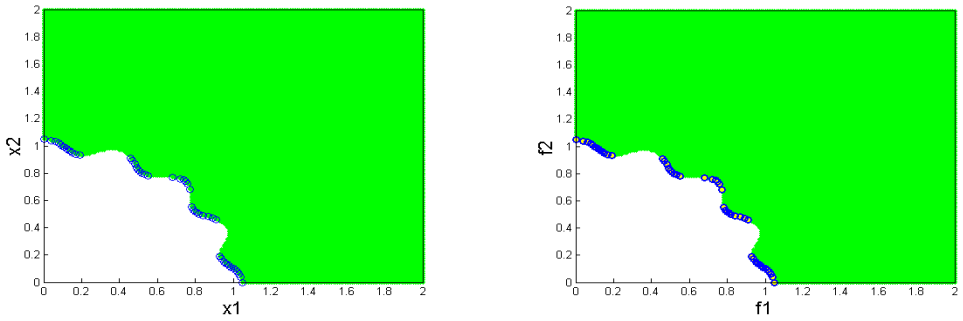
$$f_1(x_1, x_2) = x_1,$$

$$f_2(x_1, x_2) = x_2 \qquad (6.2)$$

subjecti to

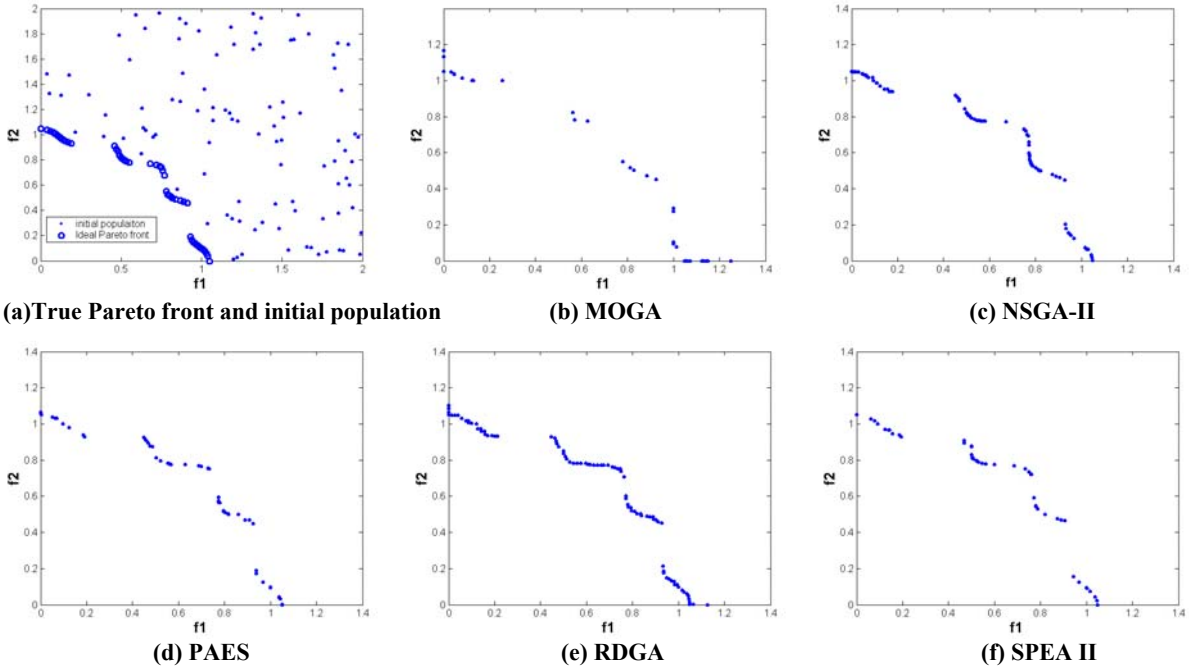
$$0 \le x_1, x_2 \le \pi\ , (x_1 - 0.5)^2 - 5(x_2 - 0.5)^2 < 0,$$

$$-(x_1^2 + x_2^2) + 1 + 0.1\cos(16\arctan(\frac{x_1}{x_2})) \le 0\ .$$



(a) Decision space and Pareto optimal set **(b) Objective space and true Pareto front**

Figure 6.2 (a) Decision space, objective space and Pareto front of Function *F1*



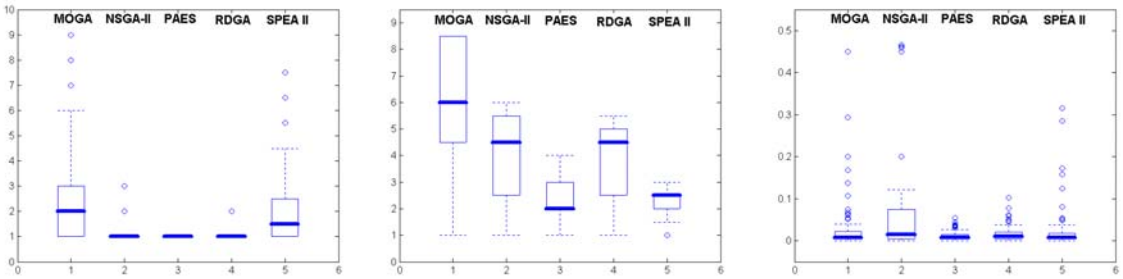
(a)True Pareto front and initial population **(b) MOGA** **(c) NSGA-II**

(d) PAES **(e) RDGA** **(f) SPEA II**

Figure 6.3 True Pareto front and Pareto fronts resulted by MOGA, NSGA-II, PAES, RDGA and SPEA II on Function *F1*

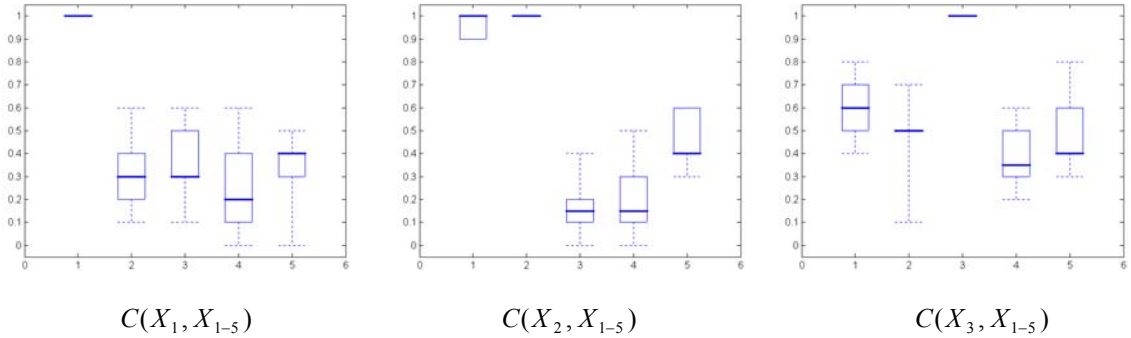
Indeed, the concave feature is created by the complicated constraints imposed in Equation (6.2). The Pareto optimal set and the true Pareto front are the same for this problem since each objective variable is equal to one decision variable. Figure 6.2(a) shows the Pareto optimal set and Figure 6.2(b) shows the corresponding Pareto front, which includes five discontinuous segments and all of them possess concavity features. Figure 6.3(a) shows the true Pareto front and a randomly generated initial population

using the same initial population for all population-based MOEAs. Figure 6.3(b) – (f) show the resulting Pareto fronts by five MOEAs. The Box plots for the average values of three indicators over 50 runs are illustrated in Figures 6.4(a), (b) and (c), respectively. The performance measures of $C(X_i, X_j)$ for the comparison sets between algorithms i and j are shown in Figure 6.5, where algorithms 1 – 5 represent MOGA, NSGA-II, PAES, RDGA and SPEA II in alphabetical order, respectively.

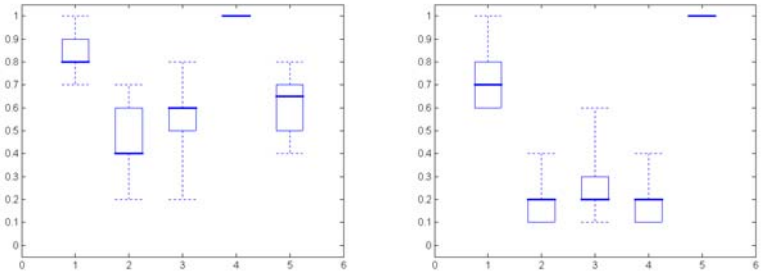


(a) average rank value **(b) average density value** **(c) average distance value**

Figure 6.4 Box plots of average individual rank, density and distance values on Function *F1*



$C(X_1, X_{1-5})$ $C(X_2, X_{1-5})$ $C(X_3, X_{1-5})$



$C(X_4, X_{1-5})$ $C(X_5, X_{1-5})$

Figure 6.5 Box plots using C measure on Function *F1*

Apparently, comparing the resulting Pareto fronts and indicator values in Figures 6.3 – 6.5, we can see that MOGA has the lowest performance in terms of all the indicator values, while the other four MOEAs provide competitive results. In particular, RDGA produces more complete Pareto fronts than the other four MOEAs and it also provides the highest $C(X_4, X_{1\sim5})$ values, which means the solution set resulted by RDGA most likely dominate the rest of the solution sets resulted by the other selective MOEAs. However, it is worthy to mention that the solution set resulted by RDGA also has relatively high density and distance values, which can be explained as RDGA creates more Pareto points than the other MOEAs and some of these points are not true non-dominated points. This problem can be solved if we let RDGA runs longer time instead of the predetermined 10,000 generations.

6.3.2 F2-1 & F2-2— Local and global Pareto optimality

Deb [48] proposed a multimodal two-objective optimization problem that possesses a local and a global Pareto front. He suggested that MOEAs might have a great tendency to converge to the local Pareto front instead of the global one if a certain kind of initial population was used. However, he did not elaborate the detail of the design procedure and how to make the problem more challenging. Moreover, a further study is needed if the local optimality is caused by constraints instead of objective functions, because two different rules behind each of them may result in dissimilar effects.

6.3.2.a F2-1—— Local optimality resulted by objective function

A two-variable, two-objective local-Pareto testing problem with a local Pareto front can be designed as:

Minimize $f_1(x_1, x_2)$ and $f_2(x_1, x_2)$, where

$$f_1(x_1, x_2) = R(x_1, x_2)$$

$$f_2(x_1, x_2) = \frac{T(x_1, x_2)}{S(x_1, x_2)} \tag{6.3}$$

where $T(x_1, x_2) = A - p_1 \times e^{-\frac{(x_2 - y_1)^2}{q_1}} - p_2 \times e^{-\frac{(x_2 - y_2)^2}{q_2}}$,

subject to $C(x_1, x_2)$.

From Equation (6.3), we can see in $T(x_1, x_2)$, parameter A affects the lowest bound of the feasible solution space and Pareto front; p_1 and p_2 determine the optimality of y_1 and y_2. If $p_1 > p_2$, y_1 will be the global optimal point, and y_2 will be the local optimal point. Otherwise, y_2 will be the global optimum, and y_1 will be the local optimum. Meanwhile, the deviation between y_1 and y_2 determines the distance of the gap between local and global optima. Parameters q_1 and q_2 determine how sharp the curves around the optimal points y_1 and y_2 will be. If $q_1 << q_2$, a global optimal point is created with a spike around y_1, and the sharper the spike is, the thinner the global Pareto optimal set will be.

A test function ***F2-1*** is created from the general model in Equation (6.3) as:

Minimize $f_1(x_1, x_2)$ and $f_2(x_1, x_2)$, where

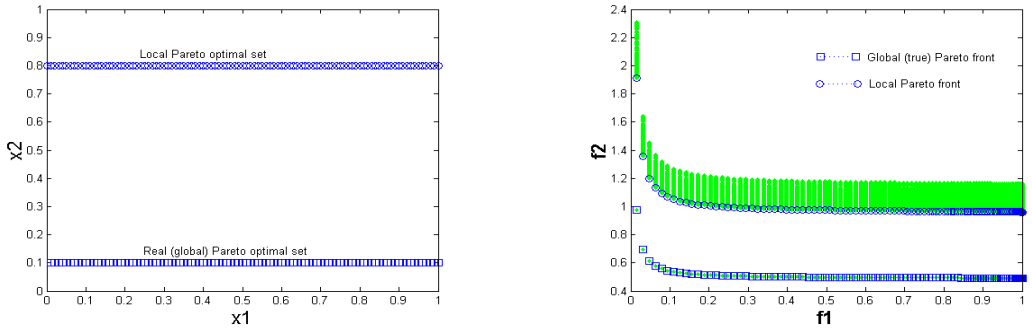
$$f_1(x_1, x_2) = \sin(\frac{\pi}{2} x_1)$$

$$f_2(x_1, x_2) = \frac{(1 - e^{-\frac{(x_2 - 0.1)^2}{0.0001}}) + (1 - 0.5e^{-\frac{(x_2 - 0.8)^2}{0.8}})}{\arctan(100 x_1)} \tag{6.4}$$

subject to $0 \le x_1, x_2 \le 1$.

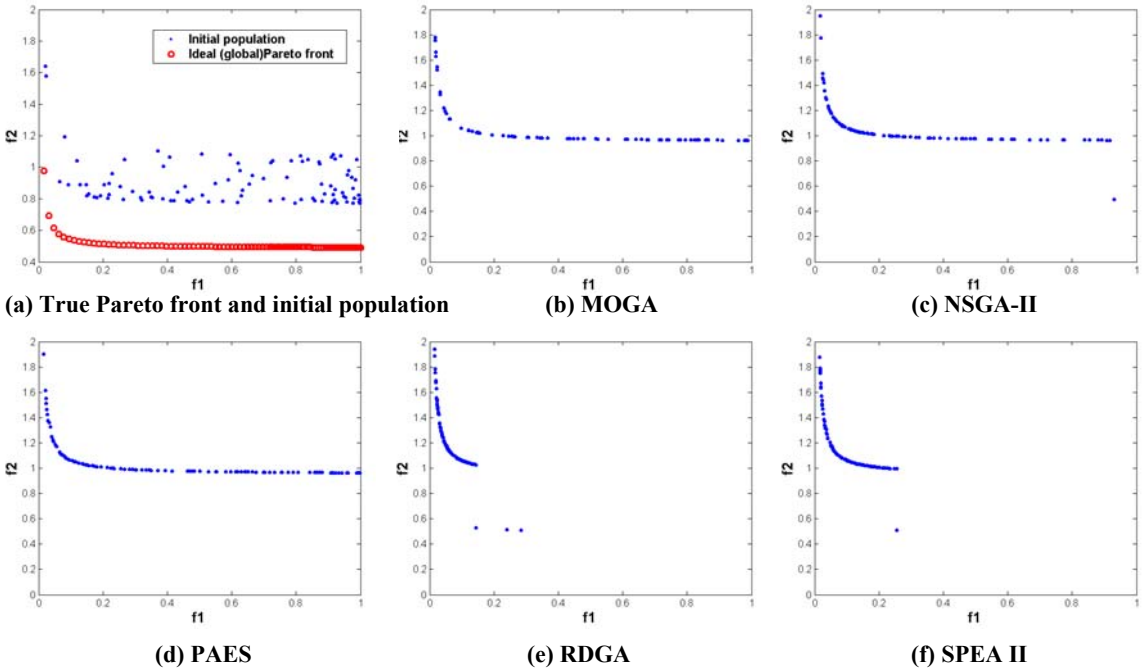
In Equation (6.4), there are two optimal values of x_2, $x_{2,global} = 0.1$ and $x_{2,local} = 0.8$, which are global optimum and local optimum for $f_2(x_1, x_2)$, respectively.

This effect will construct the final local and global Pareto fronts as shown in Figure 6.6(a) with a sampling rate equal to 0.01 for both decision variables. The true (global) Pareto front is a very thin curve, which is separated from the major range that contains the local Pareto front.



(a) Decision space and Pareto optimal set **(b) Objective space and true Pareto front**

Figure 6.6 Decision space, objective space and Pareto fronts of Function *F2-1*



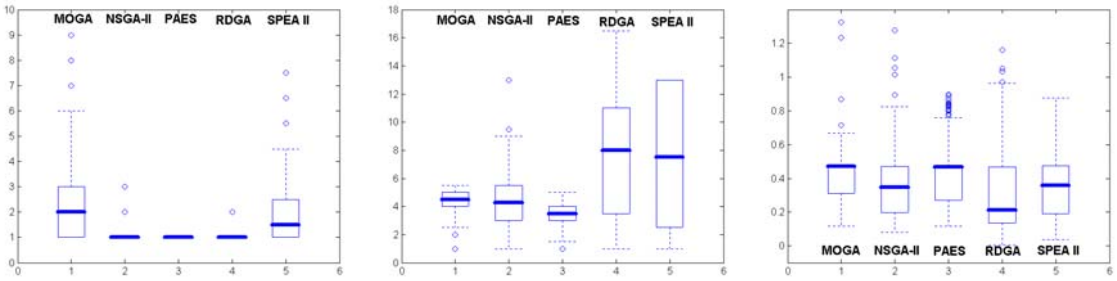
(a) True Pareto front and initial population **(b) MOGA** **(c) NSGA-II**

(d) PAES **(e) RDGA** **(f) SPEA II**

Figure 6.7True Pareto front Pareto fronts resulted by MOGA, NSGA-II, PAES, RDGA and SPEA II on Function *F2-1*

Figure 6.6(a) shows decision space and local and global Pareto optimal sets, while Figure 6.6(b) shows the objective space and local and global Pareto fronts for the test function ***F2-1***. Figure 6.7(b) – (f) show the resulting Pareto fronts by five MOEAs for a

randomly generated initial population, which is shown in Figure 6.7(a) with a true Pareto front. The Box plots for the average values of three indicators over 50 runs are illustrated in Figures 6.8(a), (b) and (c), respectively. The performance measures of $C(X_i, X_j)$ for the comparison sets between algorithms i and j are shown in Figure 6.9, where algorithms 1 – 5 represent MOGA, NSGA-II, PAES, RDGA and SPEA II in alphabetical order, respectively.



(a) average rank value **(b) average density value** **(c) average distance value**

Figure 6.8 Box plots of average individual rank, density and distance values on Function *F2-1*

From Figure 6.9, we can see that RDGA and SPEA II provide the best results. Particularly, RDGA's lowest C value is greater than 0.8, which means most of the solutions resulted by the other four MOEAs are dominated or equal to the solutions by RDGA. Moreover, RDGA produces the lowest rank and distance values. The highest density values generated by RDGA and SPEA II are caused by the partial local and partial global Pareto fronts as shown in Figure 6.7(e) and (f), which may result in a very crowded partial global segment. From Figure 6.7, it is obvious that the resulting Pareto front can be pure global, pure local or partial local and partial global. Indeed, the shapes of the resulting Pareto fronts significantly rely on different types of initial populations for this test function. Therefore, two sets of initial populations are used for comparison. Set 1 includes 50 initial populations where none of their individuals belongs to the global

Pareto front. For set 2, at least one individual is located on the global Pareto front for each of 50 initial populations.

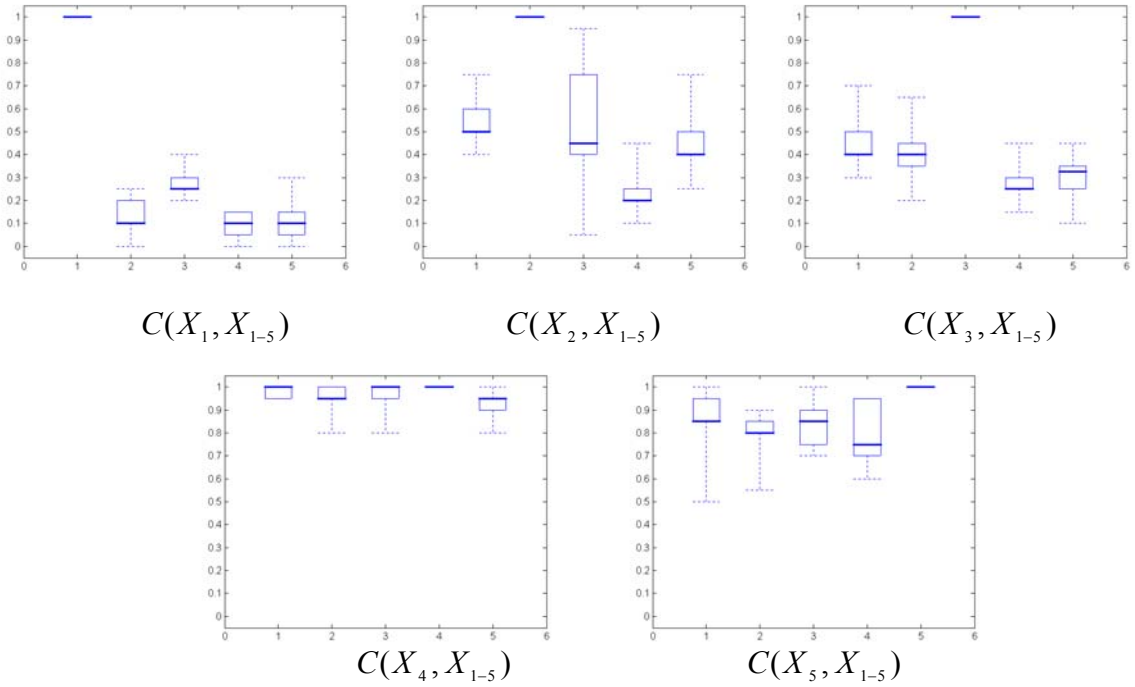


Figure 6.9 Box plots using C measure on Function *F2-1*

Tables 6.1 and 6.2 show the indicator values for set 1 and set 2 correspondingly. Comparing the observations from Table 6.1 with Table 6.2, we can see that all of the selected MOEAs are very sensitive to the initial population. When the initial population contains at least one individual that belongs to the global Pareto front, there will be a higher probability for the final population to converge to the global Pareto front, and otherwise it is most likely to converge to a local Pareto front. Moreover, different choices of parameters $A, p_1, p_2, q_1, q_2, y_1, y_2$ will produce various Pareto optimality characteristics. For instance, Figures 6.10(a) and (b) show how parameters q_1 and q_2 affect the selected MOEAs in finding a global Pareto front for the initial population Sets

1 and 2, respectively. When the ratio of q_2 / q_1 increases, the percentage that the final population is located on the global Pareto front will decrease correspondingly.

Table 6.1 Final simulation results for Function *F2-1* by five MOEAs using initial population set 1

	Number of runs	Stop generation	Final average individual rank value	Final average individual density value	Final average generation distance	Number of runs produce pure global Pareto front	Number of runs produce local Pareto front*	Number of runs produce partial global Pareto front
MOGA	50	10,000	1.02	3.21	0.59	0	49	1
NSGA-II	50	10,000	1	5.03	0.51	1	45	4
PAES	50	10,000	1	3.54	0.55	0	49	1
RDGA	50	10,000	1	6.15	0.43	2	40	8
SPEA II	50	10,000	1.01	5.32	0.46	0	42	8

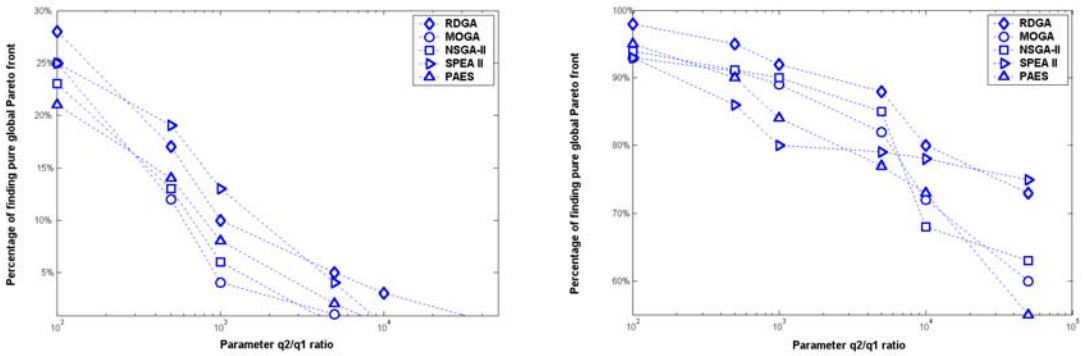
Table 6.2 Final simulation results for Function *F2-1* by five MOEAs using initial population set 2

	Number of runs	Stop generation	Final average individual rank value	Final average individual density value	Final average generation distance	Number of runs produce pure global Pareto front	Number of runs produce pure local Pareto front*	Number of runs produce partial global Pareto front
MOGA	50	10,000	1.03	3.74	0.14	37	0	13
NSGA-II	50	10,000	1.03	3.30	0.05	45	0	5
PAES	50	10,000	1	4.05	0.09	41	0	9
RDGA	50	10,000	1.12	3.44	0.07	44	0	6
SPEA II	50	10,000	1.15	3.21	0.06	44	0	6

***Note: In Table 1 and 2, we consider a pseudo-global Pareto front as a local Pareto front**

Indeed, when $q_2 / q_1 = 10{,}000$, the global Pareto optimal set is already very thin, which means there is only a very small deviation from $x_{2,\,global} = 0.1$ to produce global Pareto optimality. Even when x_2 takes a very close value to $x_{2,\,global} = 0.1$, such as $x_2 = 0.09995$, the resulting Pareto front will not be the global one, which is shown in Figure 6.11. From Figure 6.11, we also see that the gap between local and global Pareto

front is not empty. Some pseudo-global Pareto fronts will emerge when the y value is getting close to $x_{2\,global} = 0.1$. Therefore, instead of being trapped by the local Pareto front, the resulting non-dominated points may be stuck on a pseudo-global Pareto front as well. This effect becomes prominent when the ratio of q_2 / q_1 increases. In this scenario, although RDGA may perform better than the other selected MOEAs on average, it will still be difficult to find a global Pareto front if none of the individuals of the initial population are located exactly on the global Pareto front.



(a) using initial population set 1 **(b) using initial population set 2**

Figure 6.10 Illustration of q_2 / q_1 ratio affects MOEAs finding global Pareto front

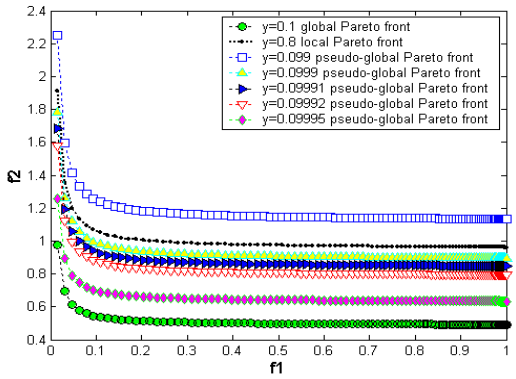


Figure 6.11 Pseudo-global Pareto fronts when x_2 approaches to $x_{2\,global} = 0.1$ ($q_2 / q_1 = 10{,}000$) ratio

6.3.2.b F2-2—*Local optimality resulted by constraint*

Applying constraints may also create the similar local and global optimal effect that is represented by

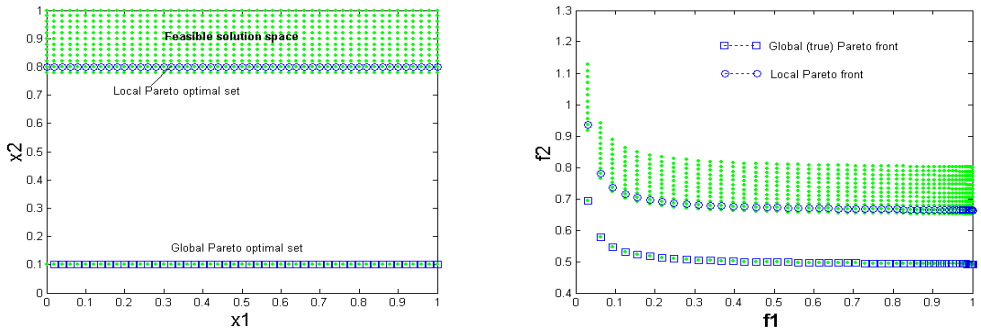
Minimize $f_1(x_1, x_2)$ and $f_2(x_1, x_2)$, where

$$f_1(x_1, x_2) = \sin(\frac{\pi}{2} x_1)$$

$$f_2(x_1, x_2) = \frac{(1 - e^{-\frac{(x_2 - 0.1)^2}{0.8}}) + (1 - 0.5e^{-\frac{(x_2 - 0.8)^2}{0.8}})}{\arctan(100x_1)} \quad (6.5)$$

subject to $0 \le x_1 \le 1$ and $0.0999 \le x_2 \le 0.1001$, or $0.79 \le x_2 \le 1$.

In Equation (6.5), parameter $q_1 = q_2$, implies there will not be any spike in the function $T(x, y)$, thus the search space will not be separated into two parts. Indeed, there is only one optimal point for $T(x, y)$ at $x_2 \approx 0.28$. However, as we designed a new constraint for the decision variables in Equation (6.5), we still can produce similar local-global optimality results shown in Figure 6.12. Under this scenario, the global Pareto front and local Pareto front still exists, except they are created by a strict constraint.



(a) Decision space and local and global Pareto optimal sets **(b) Objective space and Pareto fronts**

Figure 6.12 Decision space, objective space and local and global Pareto fronts of Function *F2-2*

Under the same conditions, we run four selected MOEAs and the proposed RDGA, given the initial population set 1 and set 2 for comparison. Tables 6.3 and 6.4 show the indicator values for Sets 1 and 2 correspondingly.

Table 6.3 Final simulation results for function *F2-2* by five MOEAs using initial population set 1

	Number of runs	Stop generation	Final average individual rank value	Final average individual density value	Final average generation distance	Number of runs produce pure global Pareto front	Number of runs produce pure local Pareto front	Number of runs produce partial global Pareto front
MOGA	50	10,000	1.21	3.33	0.32	4	18	28
NSGA-II	50	10,000	1	5.01	0.27	6	15	29
PAES	50	10,000	1	3.96	0.35	5	20	25
RDGA	50	10,000	1.13	5.61	0.22	9	13	28
SPEA II	50	10,000	1.08	5.05	0.24	10	15	25

Table 6.4 Final simulation results for function *F2-2* by five MOEAs using initial population set 2

	Number of runs	Stop generation	Final average individual rank value	Final average individual density value	Final average generation distance	Number of runs produce pure global Pareto front	Number of runs produce pure local Pareto front	Number of runs produce partial global Pareto front
MOGA	50	10,000	1.04	3.20	0.08	45	0	5
NSGA-II	50	10,000	1	4.61	0.03	48	0	2
PAES	50	10,000	1	3.83	0.08	44	0	6
RDGA	50	10,000	1	4.09	0.02	48	0	2
SPEA II	50	10,000	1	4.52	0.02	49	0	1

Comparing the indicator values in Tables 6.3 and 6.4 with those in Tables 6.1 and 6.2, we can see that for the function ***F2-2***, the global Pareto fronts, resulted by imposing constraints, are easier to be found by MOEAs than those resulted from objective functions. This occurrence can be explained as the local optimality represented in Equation (6.3) having multilayer pseudo-global Pareto fronts, each of which contributes a new local Pareto front. In this case, instead of finding the global Pareto front, MOEAs are easily trapped by a local or pseudo-global Pareto front. Nevertheless, the local optimality caused by constraints does not enclose these pseudo-global Pareto fronts. The gap

between local and global Pareto fronts is completely blank, which means the resulting non-dominated points are most likely located on either of them, thus simplifying the searching complexity.

For the local optimality created by Equation (6.5), the smaller the constraint range for $x_{2\,global}$ ($0.0999 \le x_{2\,global} \le 0.1001$ in Equation (6.5)) the more difficult for MOEAs to find a real Pareto front will be, because the global Pareto optimal set will be a thinner band when the constraint range is small.

6.3.3 F3—*MOP with high-dimensional decision space*

Minimize $f_1(x)$ and $f_2(x)$, where

$$f_1(x) = 1 - e^{-4x_1}\sin^6(6\pi x_1) \tag{6.6}$$

$$f_2(x) = g(x)\left(1 - \frac{f_1(x)}{g(x)}\right)^2 \quad g(x) = 1 + 4\left(\sum_{i=2}^{5} x_i / 4\right)^{0.25},$$

subject to $0 \le x_i \le 1, \quad i = 1, \ldots, 5.$

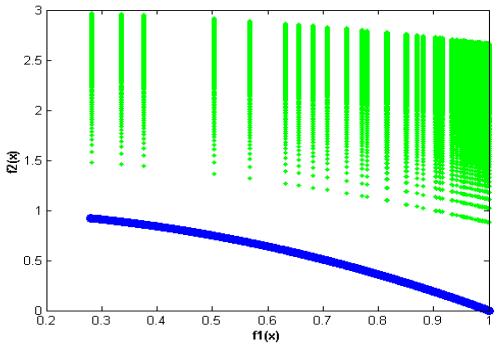
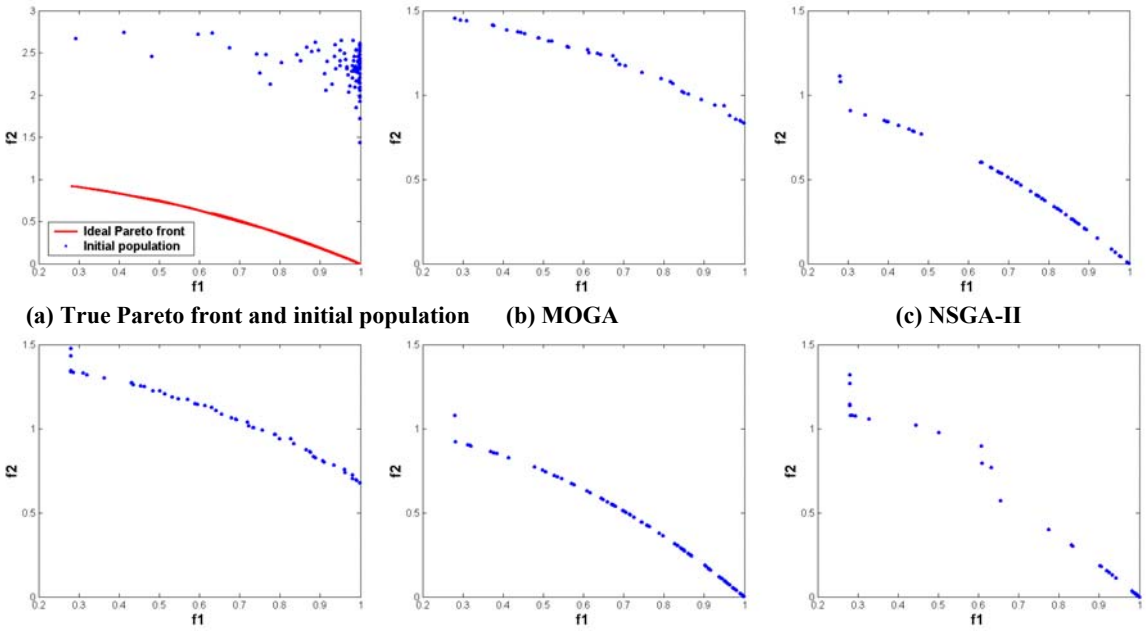


Figure 6.13 Objective space and Pareto front of Function *F3*

This test function is proposed in [14] as an MOP with high-dimensional decision space and local Pareto front in objective space as shown in Figure 6.13. Figure 6.14(b) – (f) show the resulting Pareto fronts by five chosen MOEAs for a randomly generated initial population, which is shown in Figure 6.14(a) with an ideal Pareto front. The Box plots for the average values of three indicators over 50 runs are illustrated in Figures

6.15(a), (b) and (c), respectively. The performance measures of $C(X_i, X_j)$ for the comparison sets between algorithms i and j are shown in Figure 6.16, where algorithms 1 – 5 represent MOGA, NSGA-II, PAES, RDGA and SPEA II in alphabetical order, respectively.

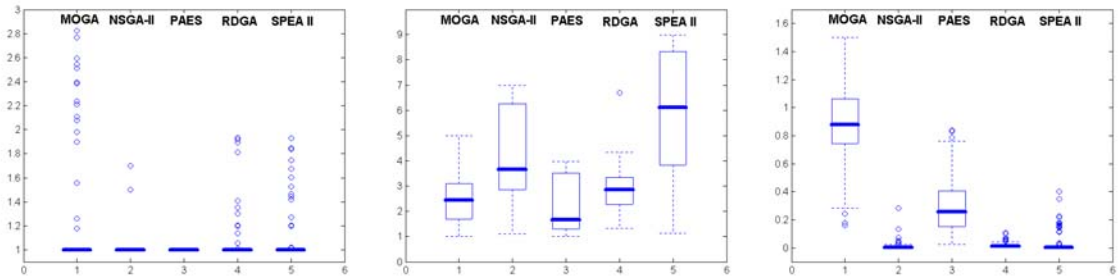


(a) True Pareto front and initial population **(b) MOGA** **(c) NSGA-II**

(d) PAES **(e) RDGA** **(f) SPEA II**

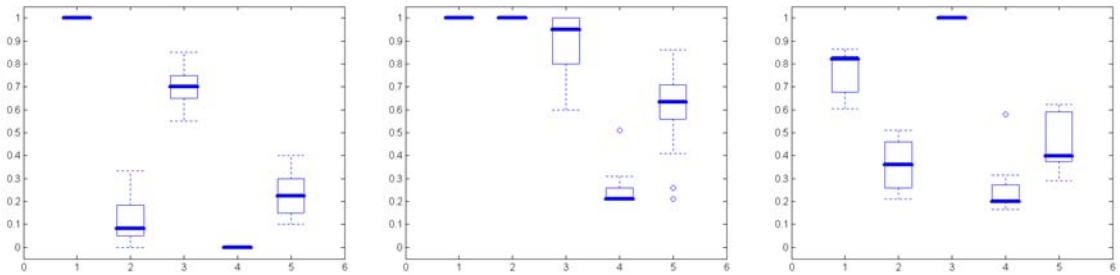
Figure 6.14 True Pareto front and Pareto fronts resulted by MOGA, NSGA-II, PAES, RDGA and SPEA II on Function *F3*

From Figures 6.14 – 6.16, it is obvious that MOGA has great difficulty in finding the true Pareto front of this MOP. On the other hand, NSGA-II, SPEA and RDGA always identify some points on the global Pareto front. Moreover, comparing to NSGA-II and SPEA II, RDGA has the lowest density value, which means RDGA tends to produce a more homogenously distributed Pareto front by minimizing individual's density value independently.

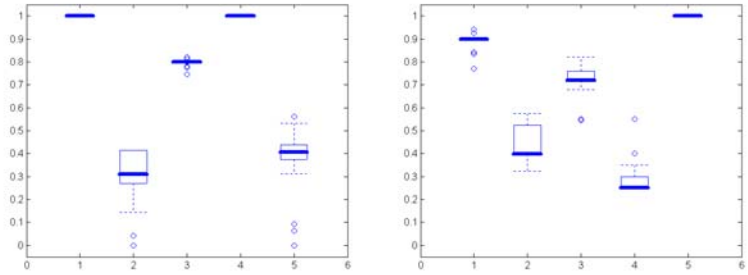


(a) average rank value **(b) average density value** **(c) average distance value**

Figure 6.15 Box plots of average individual rank, density and distance values on Function *F3*



$C(X_1, X_{1-5})$ $C(X_2, X_{1-5})$ $C(X_3, X_{1-5})$



$C(X_4, X_{1-5})$ $C(X_5, X_{1-5})$

Figure 6.16 Box plots using C measure on Function *F3*

6.3.4 F4—*MOP with high-dimensional objective space*

Minimize $f_1(x, y)$, $f_2(x, y)$, and $f_3(x, y)$, where

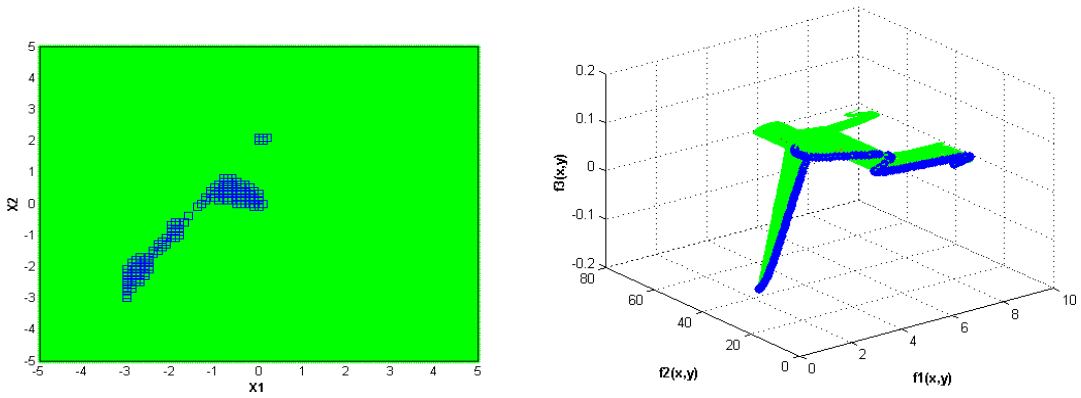
$$f_1(x, y) = 0.5(x^2 + y^2) + \sin(x^2 + y^2)$$

$$f_2(x, y) = \frac{(3x - 2y + 4)^2}{8} + \frac{(x - y + 1)^2}{27} + 15 \qquad (6.7)$$

$$f_3(x, y) = \frac{1}{(x^2 + y^2 + 1)} - 1.1e^{(-x^2 - y^2)}$$

subject to $-30 \le x, y \le 30$.

Originally designed by Viennet [62], this test function has been adopted by many researchers in that it provides three partial-contradict objective functions as shown in Figure 6.17. Figure 6.18(b) – (f) show resulting Pareto fronts by five MOEAs for a randomly generated initial population, which is shown in Figure 6.18(a). The Box plots for the average values of three indicators over 50 runs are depicted in Figures 6.19(a), (b) and (c), respectively. The performance measures of $C(X_i, X_j)$ for the comparison sets between algorithms i and j are shown in Figure 6.20, where algorithms 1– 5 represent MOGA, NSGA-II, PAES, RDGA and SPEA II in alphabetical order, respectively.

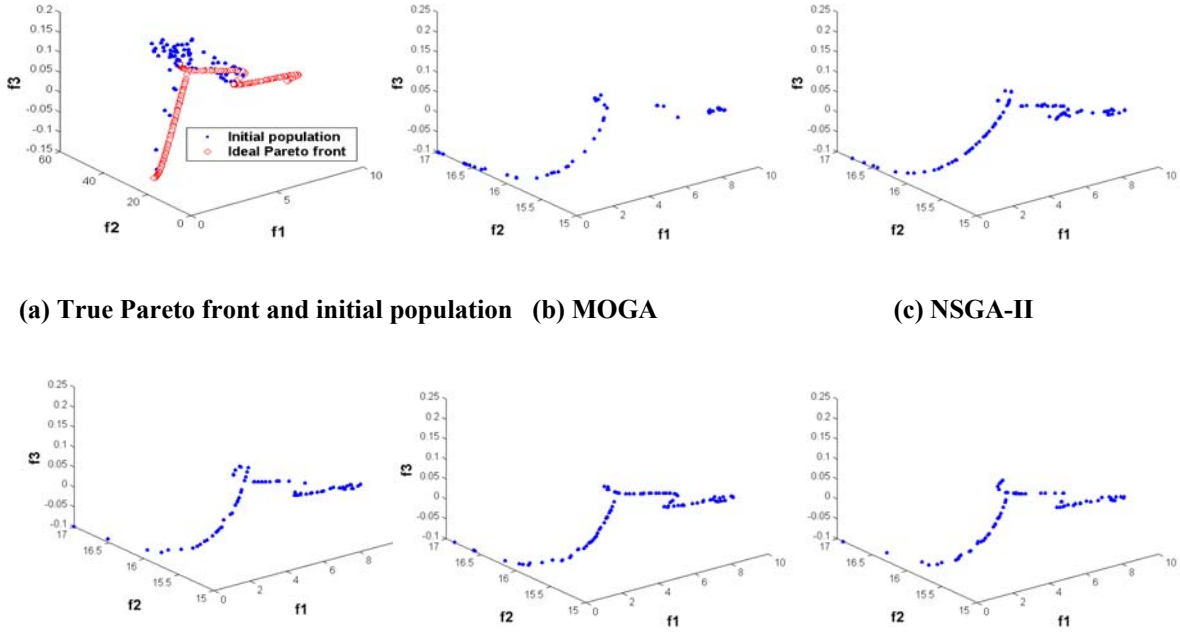


(a) Decision space and Pareto optimal set **(b) Objective space and Pareto front**

Figure 6.17 Decision space, objective space and Pareto front on Function *F4*

Indeed, test function ***F4*** possesses several challenging characteristics such as: high-dimensional objective space, discontinuous Pareto optimal set and several local minima in objective functions. From the resulting Pareto fronts and Box plots of the performance indicators in Figure 6.18 – 6.20, RDGA, NSGA-II, PAES, and SPEA II all show the ability to approximate the true Pareto front and the population-based MOEAs (i.e., RDGA, SPEA II and NSGA-II) provide higher C value as shown in Figure 6.20. Furthermore, we can see that RDGA produces smallest average individual density value and distance value comparing to NSGA-II and SPEA II. Because RDGA converts

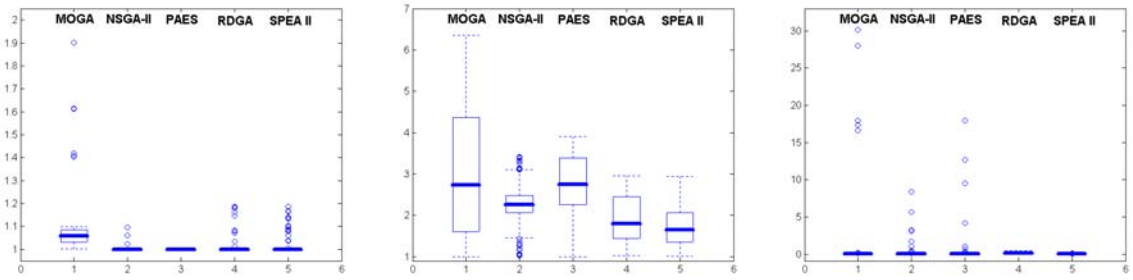
original objective space into a bi-objective rank-density domain, it is not so sensitive to the complexity of high-dimensional objective spaces. Therefore, RDGA holds the potential promise in solving these types of MOPs.



(a) True Pareto front and initial population **(b) MOGA** **(c) NSGA-II**

(d) PAES **(e) RDGA** **(f) SPEA II**

Figure 6.18 True Pareto front and Pareto fronts resulted by MOGA, NSGA-II, PAES, RDGA and SPEA II on Function *F4*



(a) average rank value **(b) average density value** **(c) average distance value**

Figure 6.19 Box plots of average individual rank, density and distance values on Function *F4*

As shown in Figure 6.21(b) and (c), Although NSGA-II performs worse than RDGA and SPEA II in terms of density preservation and distance minimization, it converges relatively fast in the rank domain (Figure 6.21(a)). This phenomenon can be

partially credited from the pure Pareto ranking scheme used by NSGA-II, which will not be affected by the density information during the evolutionary process. However, fast convergence of rank value does not imply density and distance values will converge fast as well, and ves versa. As shown in Figure 6.21(a) – (c), although RDGA converges much slower than the other three population-based MOEAs in terms of rank indicator, it has the fastest convergence speed in terms of distance indicator comparing to all the other selected MOEAs. This effect can be explained by the restricted mating method and "forbidden region" scheme applied by RDGA. On one hand, instead of using roulette wheel or tournament selection scheme, RDGA randomly selects an individual as one of the parent to mate with the best individuals located in the neighboring cells, which ensures those worst individuals have the same probabilities with the elitists to be selected and updated by their better fitted offspring, Although this strategy may sacrifice the convergence speed of an elitist in finding a *single* true non-dominated point, it yet offers those ill performed individuals a fair chance to catch up the better ones and draws the entire population to the true Pareto front. On the other hand, the "forbidden region" concept prevents an individual leading to a wrong direction when the density subpopulation is evolved. In this case, whether a new generated offspring can survive is not only because it has lower density value than its corresponding parent, also because it has equal or higher rank value comparing to the selected parent. For this reason, as an extra constraint of RDGA, "forbidden region" concept also helps compress the entire population and push it closer to the true Pareto front. Therefore, both "restricted mating" and "forbidden region" techniques contribute low variance and fast convergence of average individual distance value as shown in Figure 6.19(c) and Figure 6.21(c) (note:

these two consequences are particularly significant for function ***F4***, which may easily result in an extremely high variance of distance value during the evolutionary process if an ill performed individual has never been updated since the beginning). In addition, it is worthy to note that PAES is not a population-based algorithm and only non-dominated individuals are stored in the archive at each generation. These characteristics distinct PAES from other MOEAs mainly in two aspects: its initial rank and density values are always equal to one and the average individual rank value will remain to be one during the entire evolutionary process. From the simulation study, although PAES outperforms MOGA for all the test functions, it cannot provide competitive results comparing to the other two most advanced MOEAs (i.e., NSGA-II and SPEA II) and the proposed RDGA in terms of rank, density, distance indicators and C measure.

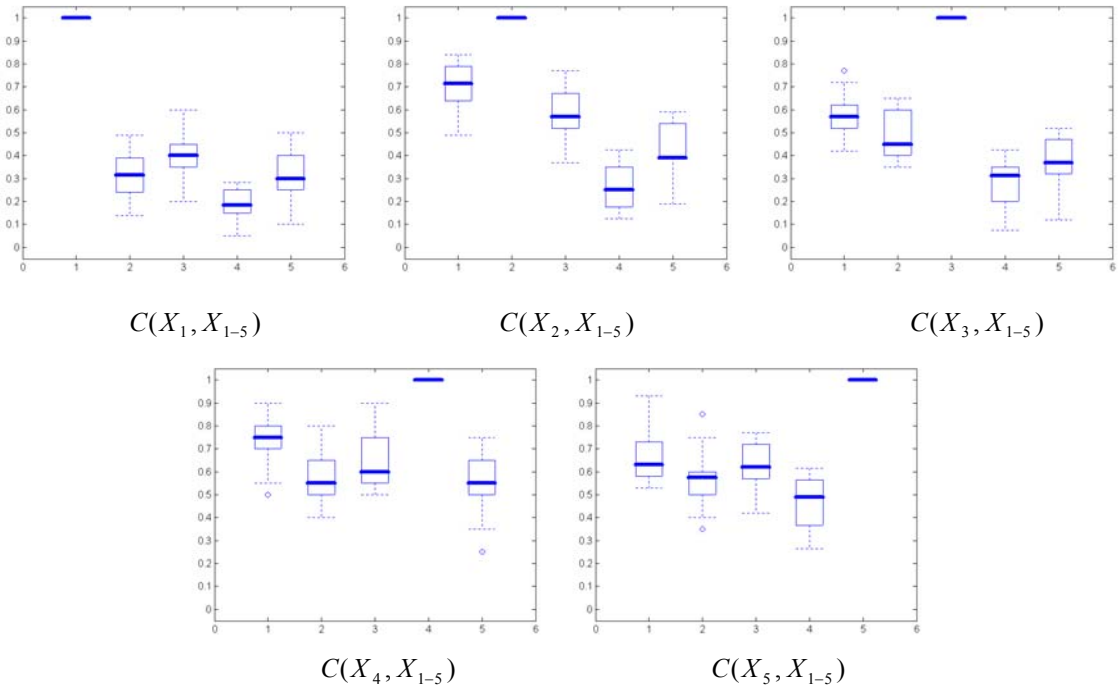
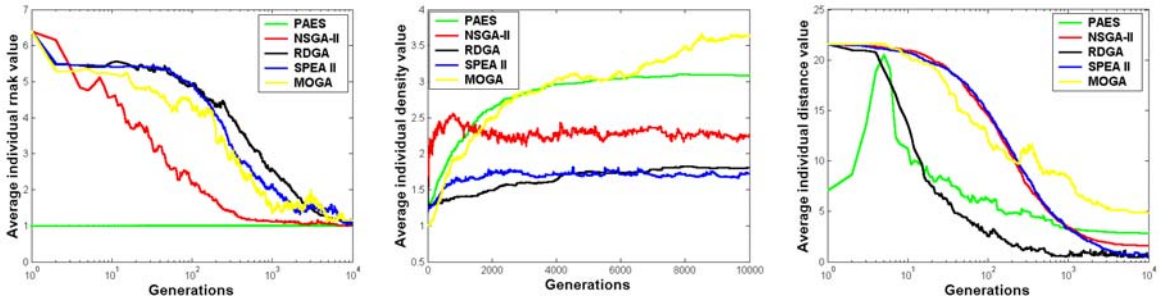


Figure 6.20 Box plots using C measure on Function *F4*



(a) average rank value **(b) average density value** **(c) average distance value**

Figure 6.21 Evolutionary trajectories of rank, density and distance values on Function *F4*

6.4 Neural Network Design by RDGA

Since the original emergence of Artificial Neural Network (ANN) in 1940's, there has been an extensive qualitative and quantitative analysis on different classes of neural networks possessing various architectures and training algorithms. Without a proven guideline, the design of an optimal neural network for a given problem is often regarded as an *ad hoc* process. Given a sufficient number of neurons, more than one neural network structure (i.e., with different weighting coefficients and numbers of neurons) can be trained to solve a given problem within an error bound if given enough training time. The decision of "which network is the best" is often decided by which network will better meet the user's needs for a given problem. It is known that the performance of neural networks is sensitive to the number of hidden neurons. Too few neurons can result in underfitting problems (poor approximation), while too many neurons may contribute to overfitting problems. Obviously, achieving a better network performance and simplifying the network topology are two conflicting objectives. This has promoted research on how to identify an optimal and efficient neural network structure. AIC (Akaike Information Criterion) [63] and PMDL (Predictive Minimum Description Length) [64] are two well-

adopted approaches. However, AIC can be inconsistent and has a tendency to overfit a model, while PMDL only succeeds in relatively simple neural network structures and seemed very difficult to extend to a complex NN structure optimization problem. Moreover, all of these approaches tend to produce a single neural network by each run, which does not offer the designers with alternative choices.

Over the past decade, evolutionary algorithms have been successfully applied to the design of network topologies and the choice of learning parameters [65]. They reported some encouraging results that are comparable with conventional neural network design approaches. However, the multiobjective trade-off characteristic of the neural network design has not been well studied and applied in the real world applications. Therefore, in the similar spirit of RDGA, a Hierarchical Rank Density Genetic Algorithm (HRDGA) is devised for neural network design in order to evolve a set of near-optimal neural networks. Without loss of generality, the type of the evolved neural networks is restricted to the Radial Basis Function (RBF) neural network.

6.4.1 Neural network design dilemma

To generate a neural network that possesses the practical applicability, several essential conditions need to be considered.

1) A training algorithm that can search for the optimal parameters (i.e., weights and biases) for the specified network structure and training task.
2) A rule or algorithm that can regulate the network complexity and ensure it to be sufficient for solving the given training problem.

3) A metric or measure to evaluate the reliability and generalization of the produced neural network.

The design of an optimal neural network involves all of these three problems. As given in [66], the ultimate goal of the construction of a neural network with the input-output relation $\mathbf{y} = f_{NS}(\mathbf{x}, \boldsymbol{\omega})$ is the minimization of the expectation of a cost function $g_T(f_{NS}(\mathbf{X}, \omega), \mathbf{Y})$ as

$$E[g_T(f_{NS}(\mathbf{X}, \boldsymbol{\omega}), \mathbf{Y})] = \int\int g_T(f_{NS}(\mathbf{x}, \boldsymbol{\omega}), \mathbf{y}) f_{\mathbf{x},\mathbf{y}}(\mathbf{x}, \mathbf{y}) d\mathbf{x} d\mathbf{y} \tag{6.8}$$

where $f_{\mathbf{x},\mathbf{y}}(\mathbf{x}, \mathbf{y})$ denotes the joint *pdf* that depends on the input vector $\mathbf{x}$ and the target output vector $\mathbf{y}$. Given a network structure NS, a family of input-output relations $F_{NS} = \{f_{NS}(\mathbf{x}, \boldsymbol{\omega})\}$, parameterized by $\boldsymbol{\omega}$, consisting of all network functions that may be formed with different choices of the weights can be assigned. The structure NS' is said to be *dominated* by NS'' if $F_{NS'} \subset F_{NS''}$. In order to choose the optimal neural network, we need to find the determination of the network function $f_{NS}^*(\mathbf{x})$ (i.e., the determination of the respective weights $\boldsymbol{\omega}^*$) that gives the minimal cost value within the family F_{NS}

$$f_{NS}^*(\mathbf{x}) = f_{NS}(\mathbf{x}, \boldsymbol{\omega}^*) = \arg\min_{\boldsymbol{\omega}} \mathbf{E}[g_L(f_{NS}(\mathbf{X}, \boldsymbol{\omega}), \mathbf{Y})], \tag{6.9}$$

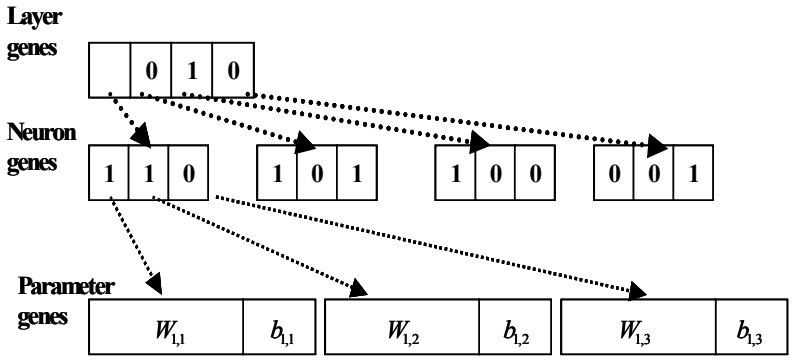
and the determination of the network structure NS' that realizes the minimal cost value within a set of structures $\{NS\}$

$$NS^* = \arg\min_{NS \in F_{NS}} \mathbf{E}[g_T(f_{NS}^*(\mathbf{X}), \mathbf{Y})]. \tag{6.10}$$

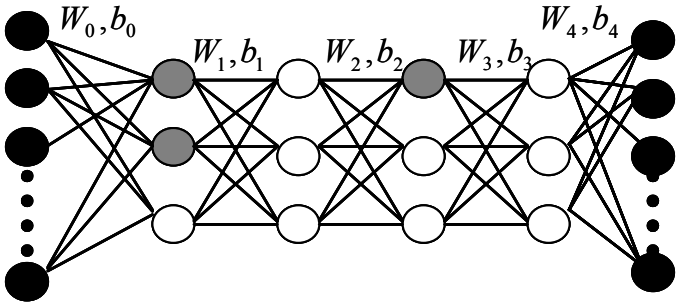
Obviously, the solutions of this task need not result into a unique network. In [67], if several structures $NS_1^*, NS_2^*, \cdots$ meet the criterion as shown in Equation (6.10), the one with the minimal number of hidden neurons is defined as an *optimal*. However, as a neural network can only tune the weights by the given training data sets, and these data sets are always finite, there will be a trade-off between NN learning capability and the variation of the hidden neuron numbers. A network with insufficient neurons might not be able to approximate well enough the functional relationship between input and target output. On the other hand, if the number of neurons is excessive, the realized network function will depend greatly on the resulting realization of the limited training set. This trade-off characteristic implies that a single *optimal* neural network is very difficult to find as extracting $f_{NS}^*(\mathbf{x})$ from F_{NS} by using a finite training data set is a difficult task, if not impossible [67]. Therefore, instead of trying to obtain a single *optimal* neural network, finding a set of *near-optimal* networks with different network structures seems more feasible. Each individual in this set of neural networks may provide different training and testing performances for different training and testing data sets. Moreover, the idea of providing "a set of" candidate networks to the decision makers can offer more flexibilities in selecting an appropriate network judged by their own preferences. For this reason, genetic algorithms and multiobjective optimization techniques can be introduced in neural network design problems to evolve network topology along with parameters and present a set of alternative candidates networks.

6.4.2 Hierarchical genetic algorithm in neural network design

In the literature of using genetic algorithms to assist neural networks design, several approaches have been proposed for evolving NN structure together with weights and biases [65,68-69]. Among all these methods, a hierarchical genotype representation is incorporated into an RBF neural network design.



(a) Genotype structure of an MLP neural network



(b) Phenotype of the neural network

Figure 6.22 Genotype and phenotype of HGA based MLP neural network

Hierarchical Genetic Algorithm (HGA) was first proposed by Ke, *et. al*., [70] for fuzzy controller design using two layer genes to evolve membership. In the proposed HGA-NN [69], a three-layer HGA is used to evolve a Multi-layer Perceptron (MLP) neural network. The chromosome structure (genotype) is shown in Figure 6.22(a). As shown in Figure 6.22(a), each candidate chromosome corresponding to a neural network

is assumed to have four hidden layers (shown in the high-level *layer genes*), where the first and the third hidden layers are activated and the second and the fourth hidden layers are deactivated.

The mid-level *neuron genes* indicate that two out of three neurons in the first hidden layer are activated, while only one neuron in the third hidden layer is activated. The low-level *parameter genes* are then used to represent the weighting parameters of each corresponding neuron activated. The active status of one control gene determines whether the parameters of the next level controlled by this gene will be activated or not. As an example, a genetic chromosome (genotype) shown in Figure 6.22(a) corresponds to an individual neural network (phenotype) with two hidden layers and two neurons in the first hidden layer and one neuron in the second layer as shown in Figure 6.22(b). By using this hierarchical genotype design, a problem, so called "one phenotype mapping different genotypes" can be prevented [69].

In a similar spirit, HGA is tailored to evolve an RBF (Radial-Basis Function) neural network. A radial-basic function can be formed as

$$f(\mathbf{x}) = \sum_{i=1}^{m} \omega_i \exp(-\| \mathbf{x} - \mathbf{c}_i \|^2) \tag{6.11}$$

where $\mathbf{c}_i$ denotes the center of the *i*th localized function, ω_i is the weighting coefficient connecting the *i*th Gaussian neuron to the output neuron, and *m* is the number of Gaussian neurons in the hidden layer. Without loss of generality, we choose the variance as unity for each Gaussian neuron.

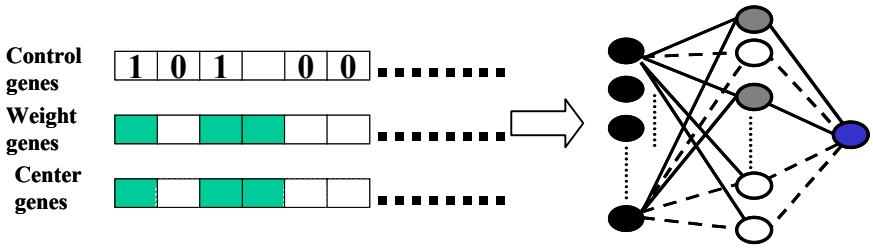


Figure 6.23 Genotype and Phenotype of HGA based RBF neural network

In HGA based RBF neural network design, genes in the genotype are classified into three categories: control genes, weight genes and center genes. The lengths of these three kinds of genes are the same. The value of each control gene (0 or 1) determines the activation status (off or on) of the corresponding weight gene and center gene. The weight genes and center genes are represented by real values. Control genes and weight genes are randomly initialized and the center genes are randomly selected from given training data samples. Figure 6.23 shows the genotype and phenotype of HGA based RBF neural network.

6.4.3 HRDGA for neural network design

To assist RBF network design, RDGA and HGA are combined as a Hierarchical Rank-Density based Genetic Algorithm to carry out the fitness evaluation and mating selection schemes [71]. The HRDGA operators are designed as followed. Figure 6.24 shows the flow chart of HRDGA for NN design procedure.

1) In HRDGA, each individual (chromosome) represents a candidate neural network. The control genes are binary bits (0 or 1). For the weight and center genes, real values are adopted as the gene representation to reduce the length of the chromosome. The population size is fixed and chosen *ad hoc* according to the complexity of the problem to be solved.

2) One-point crossover is used in the control gene segment and two-point crossover in the other two gene segments. The crossover points were randomly selected, and the crossover rates were chosen to be 0.8, 0.7 and 0.7 for the control, weight and center genes, respectively. One-point mutation was applied in each segment. In the control gene segment, common binary value mutation was adopted. In the weight and center gene segments, real value mutation was performed by adding a Gaussian noise with zero mean and unit variance. The mutation rates were set to be 0.1, 0.05 and 0.05 for the control, weight and center genes, respectively.

3) Since HRDGA is applied to optimize the neural network topology along with its performance, we need to convert them into the rank-density domain. Therefore, the original fitness—network performance and number of neurons—of each individual in a generation is evaluated and ranked, and the density value is calculated. Then the new rank and density fitness values of each individual will be evaluated, and the individuals with higher fitness measures will reproduce and crossover with other high fitness individuals with a certain probability. Their offspring replaces the low fitness parents forming a new generation. Mating is then iteratively processed.

4) When the desired number of generations is met, the evolutionary process stops.

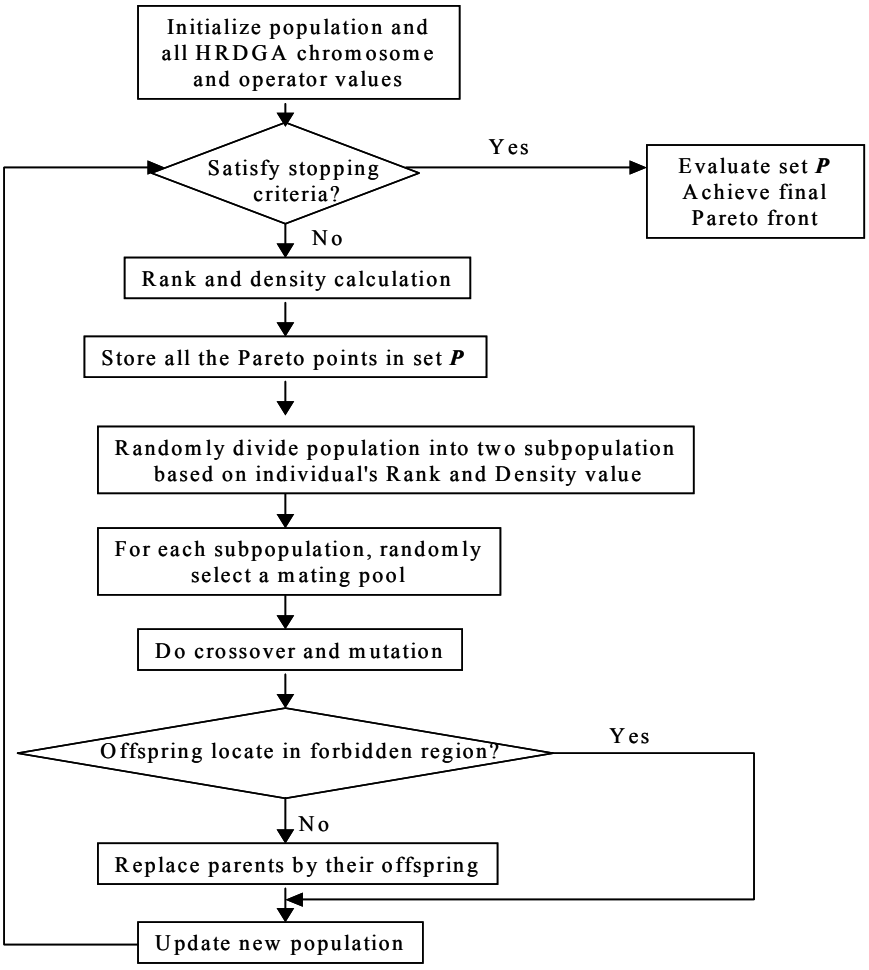


Figure 6.24 Flowchart of the main procedure of HRDGA based neural network design

6.4.4 Experimental study—Mackey-Glassy chaotic time series prediction

Since the proposed HRDGA is designed to evolve the neural network topology together with its best performance, it proves useful in solving complex problems such as time series prediction or pattern classification. For a feasibility check, we use the HRDGA assisted NN design to predict the Mackey-Glass chaotic time series.

The Mackey-Glass time series is a continuous time-delay data series. The time-delay differential equation is:

$$\frac{d(x(t))}{d(t)} = \frac{a \times x(t-\tau)}{(1+x^c(t-\tau))} - b \times x(t). \tag{6.12}$$

The chaotic behavior of the Mackey-Glass time series is determined by the delay parameter τ. Some examples are listed in Table 6.5. Larger values of τ produce more chaotic dynamics which are much more difficult to predict. Here $a = 0.2$, $b = 0.1$ and $c = 10$ are assigned for Equation (6.12). In this experimental study, HRDGA is used to evolve neural networks to predict a chaotic Mackey-Glass time series with $\tau = 150$. The network is set to predict $x(t+6)$ based on $x(t), x(t-6), x(t-12)$ and $x(t-18)$.

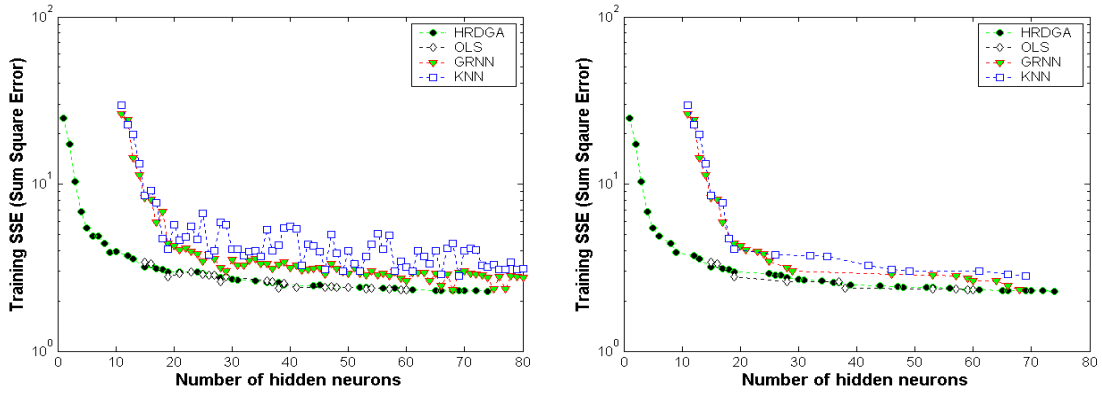
Table 6.5 Characteristics of Mackey-Glass time series

Delay parameter τ	**Chaotic characteristics**
$\tau < 4.53$	A stable fixed point attractor
$4.53 < \tau < 13.3$	A stable limit cycle attractor
$13.3 < \tau < 16.8$	Period limit cycle doubles
$\tau > 16.8$	Chaotic attractor characterized by τ

In the proposed HRDGA, 150 initial center genes are selected, 150 control genes and 150 weight genes are initially generated as well. Population size was set to be 400.

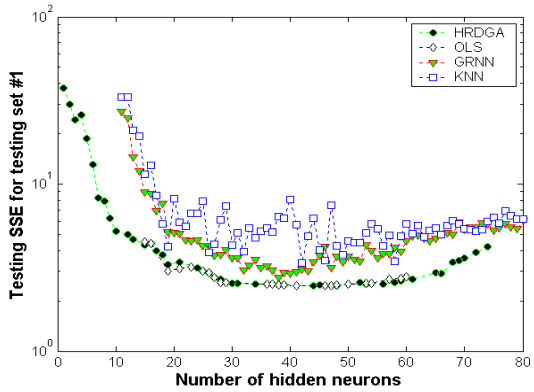
For comparison, three well-known center selection methods—KNN (K-Nearest Neighbour) [72], GRNN (Generalized Regression Neural Network) [73] and OLS (Orthogonal Least Square Error) [74] methods are applied on the same time series

prediction problem. For KNN and GRNN types of networks, 70 networks are generated with the neuron numbers increasing from 11 to 80 with the step size of one. Each of these networks will be trained by KNN and GRNN methods. For the OLS method, the selection of the tolerance parameter ρ determines the trade-off between the performance and complexity of the network. Ideally, ρ should be larger than, but very close to, the ratio $\sigma_\varepsilon^2 / \sigma_d^2$, where σ_ε^2 is the variance of the residuals, and σ_d^2 is the variance of the desired output. A smaller ρ value will produce a neural network with more neuron numbers, whereas a larger ρ value generally results in a network with less number of neurons. Therefore, by using different ρ values, we generated a group of neural networks with various training performances and numbers of hidden neurons. For the given Mackey-Glass time series prediction problem, we selected 100 different ρ values, which range from 0.01 to 0.4 with the step size of 0.01. The stop criteria for KNN, GRNN and OLS algorithms is either the epochs exceeds 5,000, or the training Sum Square Error (SSE) between two sequential generations is smaller than 0.01. For HRDGA, the stopping generation is set to be 5,000.

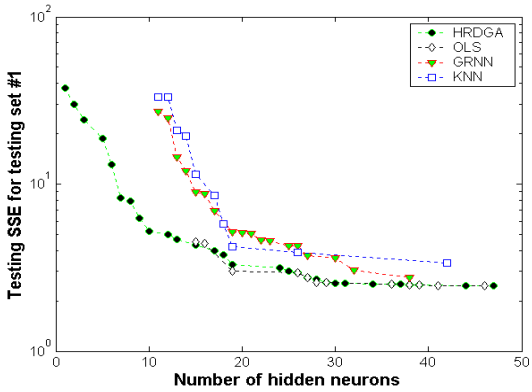


(a) Training performances **(b) Resulting Pareto front**

Figure 6.25 Training performances and Pareto fronts for the resulting neural networks with different number of hidden neurons

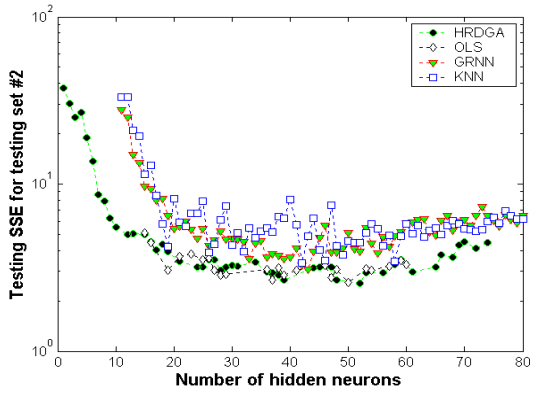


(a) Testing performances

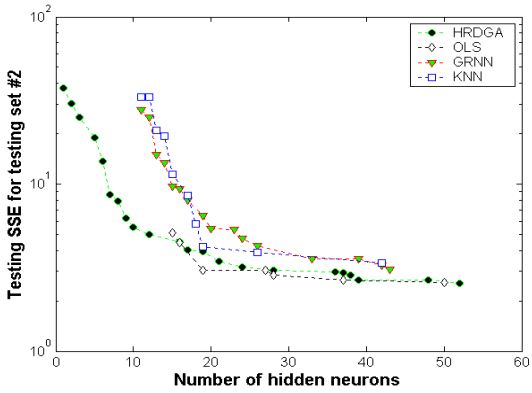


(b) Resulting Pareto front

Figure 6.26 Testing performances and Pareto fronts for the resulting neural networks with different number of hidden neurons for testing set #1

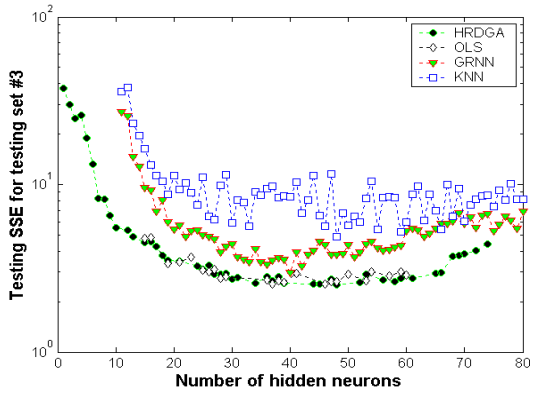


(a) Testing performances

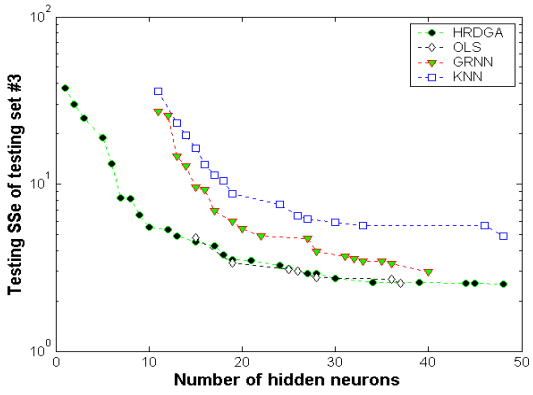


(b) Resulting Pareto front

Figure 6.27 Testing performances and Pareto fronts for the resulting neural networks with different number of hidden neurons for testing set #2



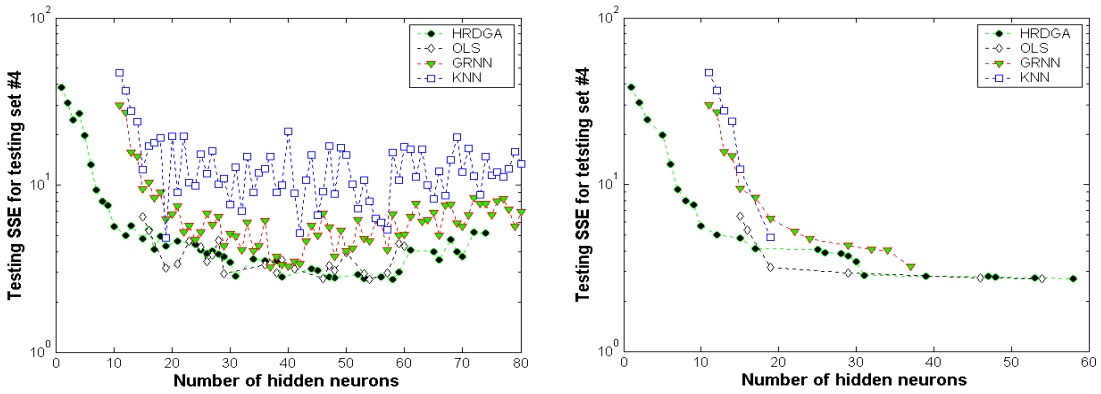
(a) Testing performances



(b) Resulting Pareto front

Figure 6.28 Training performances and Pareto fronts for the resulting neural networks with different number of hidden neurons for testing set #3

For the given time series, first 250 seconds of the data is used as the training data set, and then the data from 250 – 499, 500 – 749, 750 – 999 and 1,000 – 1,249 seconds are used as the corresponding testing data sets to be predicted by four different approaches. Each approach runs 30 times with different parameter initializations to obtain the statistical average. Figure 6.25(a) shows the resulting average training SSE of neural networks with different number of hidden neurons by four training approaches. Figure 6.25(b) shows the approximated Pareto fronts (i.e., non-dominated sets) by the selected four approaches. Figure 6.26(a) shows the average testing SSEs of the resulting networks by using the first testing data set for each approach, and Figure 6.26(b) shows their corresponding Pareto fronts. Furthermore, Figures 6.27(a) and (b), Figures 6.28(a) and (b) and Figures 6.29(a) and (b) show the same types of results by using the second, third and fourth testing data sets, respectively.



(a) Testing performances **(b) Resulting Pareto front**

Figure 6.29 Training performances and Pareto fronts for the resulting neural networks with different number of hidden neurons for testing set #4

For each algorithm, the resulting network that provides the best training result is selected from the final Pareto front as the best network for the training set. Meanwhile, each non-dominated individual network will be evaluated by each of the testing set, and

the one which provides best testing performance is extracted as the best network for the corresponding testing set.

Table 6.6 shows the performances and their corresponding numbers of hidden neurons of the best networks for the training and testing sets.

Table 6.6 Structure and performance comparison between KNN, OLS, GRNN and HRDGA

	Best performance for Training set		**Best performance for Testing set #1**		**Best performance for Testing set #2**		**Best performance for Testing set #3**		**Best performance for Testing set #4**	
	Training SSE	Neuron number	Testing SSE	Neuron number	Testing SSE	Neuron number	Testing SSE	Neuron number	Testing SSE	Neuron number
KNN	2.8339	69	3.3693	42	3.4520	**42**	4.8586	48	4.8074	**19**
GRNN	2.3382	68	2.7720	**38**	3.0711	43	2.9644	40	3.2348	37
OLS	2.3329	**60**	**2.4601**	46	2.5856	50	2.5369	**37**	**2.7199**	54
HRDGA	**2.2901**	74	2.4633	47	**2.5534**	52	**2.5226**	48	2.7216	58

From Figures 6.25 – 6.29, comparing to KNN and GRNN, HRDGA and OLS algorithms have much smaller training and testing errors for the same network structures. KNN trained networks produce the worst performances, because the RBF centers of the KNN algorithm are randomly selected, which make KNN to achieve at best a "local optimum" solution. Since GA always seeks "global optimum", and the orthogonal result is near optimal, the performances of OLS are comparable to HRDGA.

Moreover, from Figure 6.25, it is found that when the network complexity increases, the training error decreases. This phenomenon can be observed from the results by all of the selected training approaches. However, this phenomenon is only partially

maintained for the relationship between the testing performances and the network complexity. Before the number of hidden neurons reaches a certain threshold, the testing error still decreases as the network complexity increases. After that, the testing error has the tendency to fluctuate even when the number of hidden neurons continuously increases. This occurrence can be considered as that the resulting networks are overfitted. The network with the best testing performance before overfitting occurs is called the *optimal* network and judged as the final solution by conventional NN design algorithms [66]. However, from Figures 6.25 – 6.29 and Table 6.6, it is very difficult to identify a single *optimal* network that can offer the best performances for all the testing data sets, since these data sets possess different traits. Therefore, instead of searching for a single *optimal* neural network, an algorithm that can result in a near-complete set of near-optimal networks can be a more reasonable and applicable option. This is the essential reason that MOEAs can be justified for this type of neural network design problems.

From the simulation results, although KNN and GRNN approaches did not provide better training and testing results comparing to the other two approaches, they have share the advantage that the designer can control the network complexity by increasing or decreasing the neuron numbers at will. On the other hand, although the OLS algorithm always provides near-optimal network solutions with good training and testing performance, it also has serious problem to generate a set of network solutions in that the designers cannot manage the network structure directly. The trade-off characteristic between network performance and complexity totally depends on the value of tolerance parameter ρ. Same ρ value means completely different trade-off features for different

NN design problems. In addition, as shown in Figure 6.30, the relationship between ρ value and network topology is a nonlinear, many-to-one mapping, which may cause a redundant computation effort in order to generate a near-complete neural network solution set. Compared with the other three training approaches, HRDGA does not have problems in designing trade-off parameters, because it treats each objective equally and independently, and its population diversity preserving techniques help it to build a near-uniformly distributed non-dominated solution set.

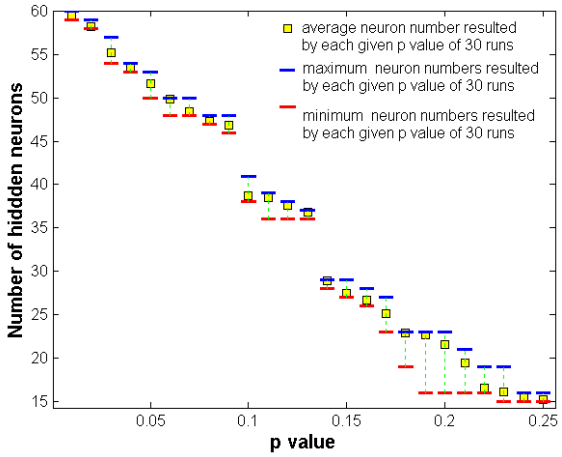


Figure 6.30 Relationship between ρ values and network complexities

Therefore, comparing to the other three traditional training approaches, the proposed HRDGA algorithm offers several benefits for the neural network design problems in terms of:

a) Providing a set of candidate solutions, which is evolved by GA's population-based optimization capability and the definition of Pareto optimality;

b) Presenting competitive or even superior individuals with high training and testing performances. This is resulted from GA's feature of seeking "global optimum" and HRDGAs' Pareto ranking technique; and

c) Offering a near-complete, non-dominated set and long-extended Pareto front, which is credited from HRDGA's population diversity keeping design that can be found in AARS, density preserving technique and the concept of "forbidden region."

VII. DYNAMIC POPULATION SIZE IN MOEA DESIGN

7.1 Introduction

In the previous three chapters, several existing MOEAs were reviewed and examined by a set of MOP test functions. From the design procedures of these MOEAs, we know that all of these algorithms share the same purpose—searching for a uniformly distributed, *near-optimal* and *near-complete* Pareto front for a given MOP. However, this ultimate goal is far from being accomplished by the existing MOEAs in terms of dealing with some of MOPs with special challenging characteristics as discussed in Chapter VI. In one respect, most of the MOPs are very complicated and require the computational resources to be homogenously distributed in a high dimensional search space. On the other hand, those better-fit individuals generally have strong tendencies to restrict searching efforts within local areas because of the "genetic drift" phenomenon, which results into the loss of diversity due to stochastic sampling. This phenomenon is a well-known trade-off decision pertaining to the *efficiency* and *efficacy* dilemma [75]. Additionally, most of the existing MOEAs adopt a heuristically chosen population size to initialize the evolutionary process. However, as addressed in [76], evolutionary algorithm may suffer from *premature* convergence if the population size if too small, whereas a over estimated population size will result in a heavy burden of undesired computation and a long waiting time for fitness improvement.

In the case of Single Objective (SO) optimization, several methods of determining an optimal population size from different perspectives have been proposed [76-79]. Since

the purpose of solving an SO problem is to search for a single optimal solution at the final generation, the distribution characteristics of the final population is not an issue to be concerned. However, in order to solve MOPs, an MOEA needs to uniformly distribute its computation effort in all the explored and unexplored areas and locate reasonable number of possible non-dominated points to sketch a *near-complete* Pareto front. In general, the size of final Pareto set yielded by most MOEAs remains to be equivalent to the size of initial population. As indicated in [6], the exact trade-off surface of an MOP is often unknown in *a priori*, it is difficult to estimate an optimal number of individuals necessary for effective exploration of the solution space as well as a good representation of the trade-off surface. This difficulty implies that a "guessed" size of the initial population is not appropriate in a real world application. Therefore, a dynamic population size autonomously adjusted by the on-line characteristics of population trade-off and density distribution information will be more efficient and effective than a constant population size in terms of avoiding *premature* convergence and unnecessary computational complexity.

As pointed out in [80], the issue of dynamic population in MOEAs has not been well attended yet. Although in some elitism based MOEAs, main population and elitist archive are separated and updated by exchanging elitists between them, the size of the main population or the sum of the main population and the archive is still fixed [10-12]. Therefore, either a "guessed" size of initial population is needed in some of these algorithms or a maximum size of archive is predetermined [52]. Tan, Lee and Khor proposed an Incrementing Multiobjective Evolutionary Algorithm (IMOEA) [80], which

devises a fuzzy boundary local perturbation technique and a dynamic local fine-tuning method in order to achieve broader neighborhood explorations and eliminate gaps and discontinuities along the Pareto front. However, this algorithm adopts a heuristic method to estimate the desired population size $dps(n)$ for next generation according to the approximated trade-off hyperareas of current generation, but not based on the dominance and density information of the entire objective space. Therefore, the computation load may be wrongly determined if the approximation of $dps(n)$ value is inaccurate, which may force IMOEA adjust grid density to reach the incorrect "optimal" population size. Moreover, IMOEA is relatively complicated and not compared with those most recently designed MOEAs (i.e., PAES, SPEA II, NSGA-II and RDGA). Its robustness needs to be further examined by different initial populations.

In this Chapter, based on RDGA, a Dynamic population-size Multiobjective Evolutionary Algorithm (DMOEA) is proposed. In DMOEA, a cell-based rank and density calculation strategy is devised and an MOP will be converted into a bi-objective optimization problem in terms of individual's rank and density values [54]. Meanwhile, a population growing strategy is designed based on the converted fitness and three types of qualitative indicators—age, health and crowd— are associated with each individual in order to determine the likelihood of eliminating an individual. In addition, an objective space compression strategy is devised and the resulting Pareto front is continuously refined based on different steady states. Three recently designed complex test functions are used to examine the efficiency and effectiveness of the proposed DMOEA. For a fair comparison, five representative MOEAs (PAES [52], SPEA II [12], NSGA-II [14],

RDGA [54] and IMOEA [80]) are also tested by the chosen benchmark problems. By examining four performance measures and the resulting Pareto fronts, DMOEA is found to be competitive with, or even superior to, the five selected MOEAs in terms of keeping the diversity of the individuals along the trade-off surface, tending to extend the Pareto front to new areas and finding a well-approximated Pareto optimal front. Moreover, from simulation results, DMOEA shows the potential to autonomously converging to the optimal population size, which is found insensitive to the initial population size chosen.

7.2 Incrementing Multiobjective Evolutionary Algorithm

Although Pareto Archive Evolutionary Strategy (PAES) implements a population incrementing scheme by keeping adding new non-dominated individuals to the archive, the first MOEA that applies dynamic population strategy is Incrementing Multiobjective Evolutionary Algorithm (IMOEA) proposed by Tan, Lee and Khor [80]. In IMOEA, the method of fuzzy boundary local perturbation was incorporated with interactive local fine-tuning for boarder neighborhood exploration to increase population size with competent offspring. Considering an *m*-dimension objective space, the desired population size $dps(n)$, with the desired population size per unit volume, ppv, and the approximated tradeoff hyperarea of $A_{to}(n)$ discovered by the population at generation n is defined as

$$lowbps \le dps(n) = ppv \times A_{to}(n) \le upbps, \tag{7.1}$$

where $lowbps$ and $upbps$ are the lower and upper bound for the desired population size $dps(n)$, respectively. In addition, IMOEA applied the method used in [81] to estimate the approximated number of hyperareas by

$$A_{to}(n) \approx \frac{\pi^{(m-1)/2}}{(\frac{m-1}{2})!} \times \left(\frac{d(n)}{2} \right)^{m-1} \tag{7.2}$$

where $d(n)$ is the diameter of the hypersphere at generation n. Therefore, based on the difference between resulting population size and estimated desired population size $dps(n)$, IMOEA adaptively filled in or filtered out individuals according to their rank and density status. In the simulation results, NSGA and SPEA are compared with IMOEA on three test functions and IMOEA shown better performance than the other two in terms of several selected indicators. However, none of the advanced MOEAs (i.e., PAES, SPAE II, NSGA-II and RDGA) was used and compared with IMOEA and the robustness of IMOEA on different initial population size is not carefully examined.

7.3 Dynamic Multiobjective Evolutionary Algorithm

Generally, the approximation of the Pareto-optimal set involves two objectives: the distance to the true Pareto front is to be minimized while the diversity of the generated solutions is to be maximized [54]. For the first objective, a Pareto-based fitness assignment (ranking scheme) is usually designed in many existing MOEAs [12] in order to guide the search towards the ideal Pareto optimal front. For the second objective, some MOEAs provide a density estimation method to preserve the population diversity. Unfortunately, these two objectives are conflicting since the diversity preservation process will slow down the convergence speed, or even degrade the quality of the resulting Pareto front. In one respect, as a general GA, MOEA exploits the "genetic drift" characteristic to converge the solution to each of the optimal point. On the other hand, the

"genetic drift" phenomenon must be avoided in order to sketch a uniformly sampled trade-off surface for the final Pareto front. This contradicted issue is very difficult to be solved by MOEAs with fixed population size, since they have to homogenously distribute the predetermined computation resource to all the possible directions in the objective space. Therefore, to cope with this contradiction, a Dynamic Multiobjective Evolutionary Algorithm (DMOEA) is proposed in this chapter.

Similar to the other advanced MOEAs [12-14,54], DMOEA also converts the original MOP into a bi-objective optimization problem—minimizing individual rank value and maintaining individual density value [84]. However, as adding or removing an individual will affect the rank and density values of other individuals, the rank and density values of each individual need to be recalculated after the population has been updated. This recalculation will cost more computation time as the population size increases to a larger number. Therefore, to solve this problem, we design a novel cell-based rank and density calculation scheme.

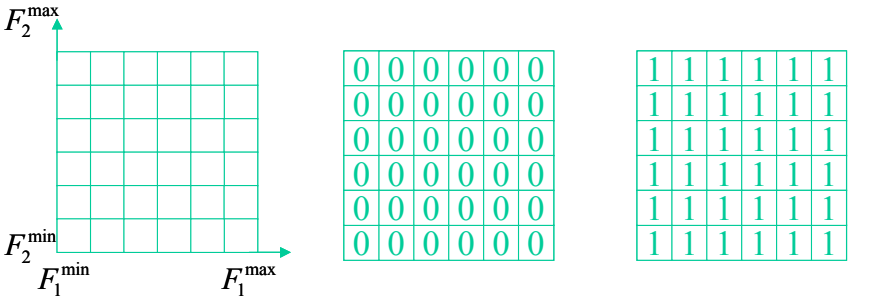
7.3.1 Cell-based Rank and Density Calculation Scheme

In DMOEA, the original *n*-dimensional objective space is divided into $K_1 \times K_2 \times \ldots \times K_n$ cells (i.e. grids), thus the cell width in the *ith* objective dimension d_i can be formed as

$$d_i = \frac{F_i^{\max} - F_i^{\min}}{K_i}, i = 1,...,n, \tag{7.3}$$

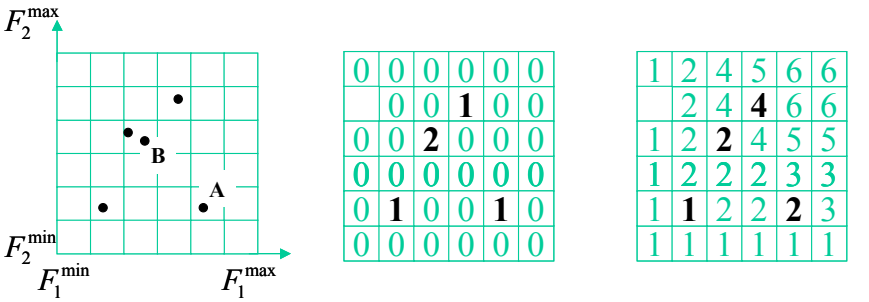
where $F_i^{\max}$ and $F_i^{\min}$ are the estimated high and low boundaries for the *ith* objective dimension. After the objective space has been determined and divided, as shown in Figure 7.1(a), the center position of each cell will be obtained and two matrixes are set up to store the rank and density values of each cell, which initially are 1 and 0, respectively (shown in Figure 7.1(b) – (c)). Second, each individual of initial population will search for its nearest cell center and identify this cell as its "home address" and consider the other individuals who share the same "home address" as its "family members". Then as shown in Figure 7.2(a) – (c), for each of these "homes", the number of "family members" who dwell in it will be counted and saved as the density value of the "home". In addition, the rank values of the cells that dominated by any of these "homes" will be increased by the density values of those "home". Third, when an offspring is generated and accepted (individual **C** in Figure 7.3(a)), its "home address" can be easily located by following the second step and the density value of its "home" will increase by one and the rank values of the cells dominated by its "home" will be increased by one. Meanwhile, if an old individual (individual **B** in Figure 7.2(a)) is removed, its "home" will be notified and the density value of its "home" will decrease by one and the rank values of the cells dominated by its "home" will be decreased by one, correspondingly. Therefore, at each generation, an individual can access its "home address" and then obtain the corresponding rank and density values. The "home address" is merely a "pointer" to inform an individual where to find its rank and density values. For instance, as shown in Figure 7.3, the "home address", rank and density values of individual **A** are (5,2), 2 and 1, respectively. Therefore, if the estimated objective space is large enough that a newly generated or a removed individual does not change the boundary of the range of current

objective space, the size of each cell will not change, which means an individual's "home address" will never change if this individual is not removed. By this means, the original objective of searching for a *near-complete*, *near-optimal* and *uniformly distributed* Pareto front has been converted to locate as many optimal "home addresses" as possible, each of which contains *ppv* number of these individuals.



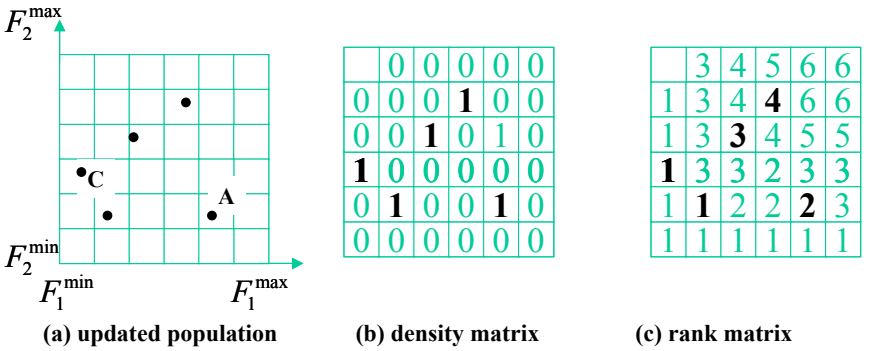
(a) divided objective space **(b) initial density matrix** **(c) initial rank matrix**

Figure 7.1 Estimated objective space, initial density matrix and initial rank matrix



(a) initial population **(b) density matrix** **(c) rank matrix**

Figure 7.2 Initial population and its corresponding density and rank matrices



(a) updated population **(b) density matrix** **(c) rank matrix**

Figure 7.3 (a) Updated population and its corresponding density matrix and rank matrix

Although the genetic operations (i.e., crossover and mutation) are still performed by genotype chromosomes, the fitness evaluation of whether the resulting offspring is good or not is based on its location on the rank and density matrices. By this method, the procedure of updating rank and density matrixes is totally irrelevant to the procedure of fitness evaluation on an individual. On one respect, as each crossover or mutation operation can only produce at most two new individuals, the computation load on updating the rank and density will be trivial for each generation. On the other hand, when two individuals are compared, they just need to provide their "home addresses" and the current rank or density status of their "home addresses" can be evaluated to determine which individual is better fitted. Therefore, no matter how large the population size is, the computation effort of both matrixes updating and fitness evaluation will not be affected, which provide an efficient way in applying dynamic population size in evolutionary process.

7.3.2 Cell Rank and Health Indicator

Once the rank and density values of each cell have been obtained by using the method described as Subsection 7.3.1, two indicators that are associated with rank and density values are designed to determine if a cell is "comfortable" enough for an individual to inhabit. They are *health* and *crowd* indicators.

In DOMGA, we convert the rank value of a cell into a *health* indicator in order to measure the dominance status of the concerned cell comparing to the other cells. Assume

at generation n, a cell c has a rank value $rank(c,n)$, the *health* value of cell c at generation n is given as

$$H(c,n) = \frac{1}{rank(c,n)}. \tag{7.4}$$

From Equation (7.4), a cell with rank value 1, which is the healthiest, will have an H value equal to 1 and a cell with higher rank value will have a lower H value that is more closer to 0 (Figure 7.4). Therefore, an H value indicates the Pareto rank information of a cell and the relationship between a cell's rank value and H value is not linear. In one aspect, H values drop very fast if rank values are greater than 1, which results in a significant difference between dominated and non-dominated cells in terms of *health* condition. On the other hand, H values also saturate very fast, which assigns all the dominated cells very low *H* values (near zero) if their rank values are very high. This characteristic can be used by the individual elimination scheme that will be discussed later.

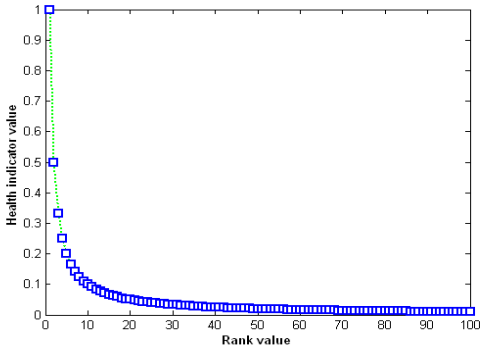


Figure 7.4 Relationship between rank value and health value.

7.3.3 Cell Density and Crowd Indicators

Referring to [80], consider an *m*-dimension objective space, the desired population size, $dps(n)$, with the desired population size per unit volume, ppv, and the

explored trade-off hyper-area, $A_{to}(n)$, discovered by the population at generation n can be defined as Equation (7.1). Therefore, with given population size per unit volume, ppv, the optimal population size can be obtained if an MOEA can correctly discover all the trade-off hyper-areas for an MOP. In DMOEA, instead of estimate the trade-off hyper-area $A_{to}(n)$ for each generation [80], we concentrate on searching for a *near-complete* final set of trade-off hyper-areas and ensure each of these areas contains ppv number of non-dominated individuals. Therefore, by using DMOEA, the optimal population size and final Pareto front will be found simultaneously at the final generation.

As discussed in Subsection 7.3.1, the density value of a cell is defined as the number of the individuals located in it. The finer the resolution of the cell is, the better performance DMOEA can provide. A crowdness is associated with each cell to show current density information of the concerned cell. Assume at generation n, the density value of cell c is $density(c, n)$, the crowdness indicator of cell c is defined as

$$D(c, n) = \frac{density(c, n)}{ppv}. \tag{7.5}$$

Therefore, by using crowdness indicator, we can obtain the information about how congested each cell is, comparing to the desired ppv value.

7.3.4 Population growing strategy

In general, if an MOEA has a fixed population size, a "replacement" scheme is always applied. In this scheme, in order to keep the population size unchanged, a newborn offspring will replace one of its parents if its fitness value is higher than that of

the parent. However, this scheme brings up a problem that some of the replaced parents may still be very valuable and have not been well exploited yet before they are replaced. Although some MOEAs (i.e. NSGA-II and SPEA II) adopt an elitist archive in addition to the main population in order to store some of the non-dominated individuals generated during the evolutionary process, this problem is still not completely resolved. Therefore, DMOEA applies two independent strategies—population growing strategy and population decline strategy. The first strategy only focus on pure population increment and ensures each of the individual survives enough generations so that it can contribute its valuable schemas. Meanwhile a population declining strategy is also designed to prevent the population size growing excessively. The second strategy will be discussed in the next Subsection.

Because exploring the cells with minimum rank values and maintaining these cells densities to a desired value are two converted objectives of DMOEA, crossover and mutation operations need to be devised to fit both of the purposes. For crossover, a reproduction pool with a fixed number of selected parents is setup; a Cellular GA [53] is then applied to explore the new search area by "diffusion"— each selected parent performs crossover with a randomly selected individual located in the nearest cell that dominates the concerned cell. If a resulting offspring is located in a cell with a better fitness (a lower rank value or a lower density value) than its selected parents, it will be kept to the next generation; otherwise, it will not survive. The mutation operation is analogous. As a result, this strategy will guarantee that a newborn individual will have a better fitness value than at least one of its parents, which helps DMOEA to cover all the

unexplored cells in the objective space. To prevent "harmful" offspring from surviving and affecting the evolutionary direction and speed, *forbidden region* concept is applied in the offspring-generating scheme for the density subpopulation.

7.3.5 Population declining strategy

As discussed in Subsection 7.3.4, a population declining strategy is necessary to prevent the population size growing unbounded. In DMOEA, whether an individual will be removed or not depends on its *health* and *crowdness* indicators we mentioned in Subsection 7.3.2 and 7.3.3. Moreover, to ensure that each appeared individual has enough lifespan to contribute its valuable schemas, an *age* indictor is also designed in DMOEA. For an individual in the initial population, its *age* value is assigned to be one, and its *age* will increase by one if the individual survives at the next generation. Similarly, the *age* of a newborn offspring is one and grows generation by generation as long as it lives. Assume at generation n, an individual y has an *age* value $age(y,n)$, its *age* indicator $A(y,n)$ is given by

$$A(y,n) = \frac{age(y,n) - A_{th}}{n}, \tag{7.6}$$

where A_{th} is a pre-specified *age* threshold, which means that an individual will not be eliminated if its *age* is less than A_{th}.

To ensure that an eliminated individual has a low fitness value, DMOEA moderately removes three types of individuals with different livelihoods:

1) Likelihood of removing the most unhealthy individuals

At generation n, find a set Y_r that contains all the individuals with the highest rank value $r_{\max}$. Therefore, if $r_{\max}$ is larger than 1, the likelihood of individual $y_i \in Y_r$ to be eliminated is given by

$$l_1^i = (1 - H(c_i, n))^2 \times A(y_i, n), \tag{7.7}$$

where $H(c_i, n) = \frac{1}{r_{\max}}$ denotes the *health* indicator value of the cell c_i that contains individual y_i at generation n.

2) Likelihood of removing the unhealthy individuals in the most crowded cells

At generation n, find a set Y_d that contains all the individuals with the highest density value, and then find a set $Y_{dr} \subseteq Y_d$ that includes all the individuals with the highest cell rank value $r_{d\max}$. In addition, denote the *pure* Pareto rank of individual $y_i \in Y_{dr}$ to be r_{di}. Therefore, if $r_{d\max}$ is greater than 1, the likelihood of individual y_i to be eliminated is given by

$$l_2^i = (1 - H(c_i, n))^2 \times (1 - \frac{1}{r_{di}})^2 \times (D(c_i, n) - 1) \times A(y_i, n), \tag{7.8}$$

where $H(c_i, n) = \frac{1}{r_{d\max}}$ and $D(c_i, n)$ represent the *health* and *crowdness* values of the cell c_i that contains individual y_i at generation n. It is noted that $R_{dr} = \{r_{di}\}$ represents the local rank value of the individuals of set Y_{dr} and is calculated by *pure* Pareto ranking scheme proposed by Goldberg [25]. Although all the individuals located in the same cell

share the same rank value, they may still have local dominance relationship as shown in Figure 7.5, where individuals **A** and **B** have the highest and lowest local (pure Pareto) rank values, respectively. Therefore, by measuring local rank values among all the individuals in one cell, DMOEA can determine the likelihood of eliminating an individual more precisely.

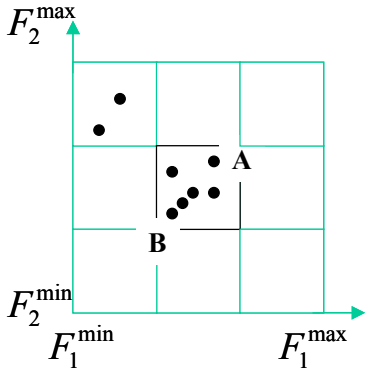


Figure 7.5 Illustration of the pure Pareto ranking for the individuals located in the same cell

3) Likelihood of removing non-dominated individuals from the most crowded cells after convergence

At generation n, if $r_{\max}$ is equal to 1, find a set Y_{rc} that contains all the individuals with the highest density value, and their local pure Pareto rank values of individual $y_i \in Y_{rc}$ to be r_{di}. Therefore, the likelihood of individual y_i to be eliminated is given by

$$l_3^i = (D(c_i, n) - 1) \times (1 - \frac{1}{r_{di}})^2 \times A(y_i, n), \tag{7.9}$$

where $D(c_i, n)$ represents the *crowdness* value of the cell c_i that contains individual y_i at generation *n*.

To determine if an individual y_i will be eliminated, three random numbers between [0, 1] are generated to compare with the concerned likelihood, l_1^i, l_2^i and l_3^i, according to the situation of the given individual. If the likelihood is larger than the corresponding random number, the selected individual will be removed from the population. Otherwise, the selected individual will survive to the next generation. Therefore, from Equations (7.7) – (7.9), we can draw some observations as follows.

1) Because the age indicator $A(y_i, n)$ influences all of three likelihood, l_1, l_2 and l_3 will be negative number if the age of the concerned individual is smaller than the age threshold A_{th}. This implies that if an individual is not old enough, it will not be eliminated from the population no matter how high its rank and density value is.

2) At each generation, DMOEA will remove those *most unhealthy* individuals according to likelihood l_1, based on their rank values and ages. Assume the age indicator of an individual y is $A(y, n) \approx 1$, the relationship between its rank value and l_1 value is illustrated in Figure 7.6. Without considering the effects of other indicators, when an *unhealthy* individual in the set Y_r has a very high $r_{\max}$ value, it will have a very high likelihood (l_1) to be eliminated, since it is too far away from the current Pareto front. Moreover, as $r_{\max}$ drops and gets closer to 1, l_1 will decrease very fast, and the concerned individual will not be removed easily because it is very likely to be evolved into an elitist in the future. Therefore, this "shell removing" strategy will keep eliminating the individuals located on the outside layer with an adaptive probability until the entire population converges into a non-dominated set.

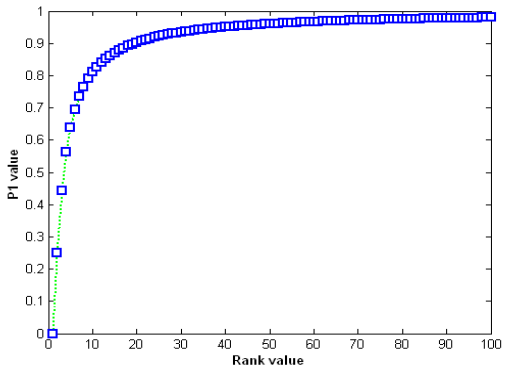


Figure 6 Relationship between rank values and l_1 values

3) Because all the individuals in the same cell share the fixed computation resource (or "living resource"), the individuals located in a crowded cell have to compete much harder for the limited resource than those located in a sparse cell. Therefore, another elimination scheme based on *crowdness* indicator values is designed in DMOEA in order to remove some *unhealthy* individuals that stay in the most crowded areas. From Equation (7.8), at each generation, if an individual belongs to the set Y_{dr}, it will have the likelihood of l_2 to be eliminated based on its *age*, *health*, and local rank value and density condition. From this scheme, the population tends to be homogeneously distributed by eliminating the redundant individuals.

4) After every individual has converged into a Pareto point, another elimination scheme is implemented based on l_3 values. Therefore, the resulting trade-off hyperareas $A_{to}(n)$ are counted, and the final population is truncated to ensure each cell contains ppv number of individuals; thus the optimal population size can be calculated by Equation (7.1).

7.4 Objective Space Compression Strategy

Although the cell-based rank and density calculation scheme discussed in Subsection 7.3.1 can significantly improve the efficiency of DMOEA during its evolutionary process, it cannot guarantee the accuracy of the resulting Pareto front since an individual's rank value is represented by the rank value of its "home address", not by its own dominance status. Because the size of true Pareto front is generally unknown, the boundaries of the objective are usually selected to be very large, which may be far away from the true Pareto front, to ensure entire true Pareto front is covered by the estimated objective space. In this case, if the predetermined cell scale $K_1,\ldots,K_n$ are not chosen to be correspondingly large enough, the size of a cell will be too spacious comparing to the true Pareto front, which may result in a set of inaccurate Pareto optimal set. This phenomenon can be illustrated as Figure 7.7 (a), where the rank value of both cell **A** and **B** is 1 since they contain true Pareto front. In this case, all the resulting individuals located in cells **A** and **B** are non-dominated solutions according to proposed cell-based rank calculation scheme. However, if we examine these individuals by using pure Pareto ranking method, we will find that most of these individuals are dominated points. To address this problem, we can either increase the cell scale $K_1,\ldots,K_n$ to a very large number or adaptively compress objective space. Nevertheless, the first method will increase the computation time because it leaves too many redundant empty cells when the resulting Pareto front approaches true Pareto front. Therefore, an objective space compression strategy is designed to adjust the size of objective space and make it suitable to search for the true Pareto front with a high precision. Assume at generation n, the high

and low boundaries of the *ith* dimension of the objective space and current population are $F_i^{\max}$, $F_i^{\min}$, $P_i^{\max}$ and $P_i^{\min}$. Then three criteria are evaluated:

1) maximum cell rank value of all the individuals is 1;

2) $(F_i^{\max} - P_i^{\max}) > 0.1(F_i^{\max} - F_i^{\min})$ and $(P_i^{\min} - F_i^{\min}) > 0.1(F_i^{\max} - F_i^{\min})$ (7.10)

3) minimum age value of all the individuals is greater than predefined age threshold A_{th}.

The ratio, 0.1 in Equation (7.10) is chosen heuristically. Therefore, if all of above three criteria are satisfied, a new-generated high boundary of the objective space is defined as:

$$F_i^{\max} = (P_i^{\max} + F_i^{\max})/2, \tag{7.11}$$

which means the distance between the updated high boundary of the objective space and the high boundary of the current population has decreased to half of its original value. Similarly,

$$F_i^{\min} = (P_i^{\min} + F_i^{\min})/2, \tag{7.12}$$

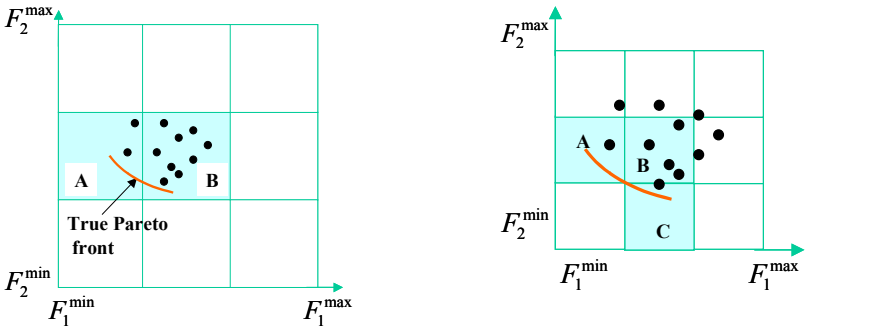
which means the distance between the updated low boundary of the objective space and the low boundary of the current population has decreased to half of its original value. The rationale of introducing the first criterion is to ensure the approximated area of the true Pareto front has been discovered before the objective space is compressed, which can avoid incorrect truncation of potential non-dominated cells. Moreover, after a compression strategy is performed, the cell rank and density value will not remain the same as before, and the "home address" of each individual may change correspondingly. As a consequence, the cell rank and density values need to be recalculated, which may

cost tremendous amounts of computing time. For these reasons, the objective space is not compressed at each generation..

Comparing Figure 7.7(a) with 7.7(b), we can see that the proposed objective space compression strategy results in three effects:

1) Some individuals that are originally considered as Pareto points are pushed out of the updated non-dominated cells, which implies the resulting Pareto front are refined.
2) The density values of the updated non-dominated cells are reduced.
3) Some new non-dominated cells may be created (cell C in Figure 7.7(b)).

Therefore, by continuously compressing the objective space, the resulting non-dominated set can be tuned and a more extended and homogenously distributed Pareto front can be obtained.



(a) Original objective space **(b) Compressed objective sapce**

Figure 7.7 Illustration of objective space compression strategy

7.5 Convergence Properties and Final Refinement Method

Based on all the techniques introduced from Subsection 7.3—7.4, we can determine if DMOEA has converged by examining the following criteria:

a. The rank values of all cells are 1s.

b. The objective space cannot be compressed anymore.

c. Each resulting non-dominated cell contains ppv individuals.

When all three criteria are met, the resulting non-dominated set can hardly be refined by DMOEA any further. At this stage, DMOEA will keep running and the cell-based rank calculation scheme will be replaced by pure Pareto ranking scheme [25], whereas cell-based density calculation scheme remains unchanged. The reason of this step is because another criterion "the Pareto ranks of all resulting individuals are equal to 1" should be satisfied as well to guarantee there is no dominance relationship among resulting Pareto solutions at the final generation. Figure 7.8 shows the flowchart of proposed DMOEA.

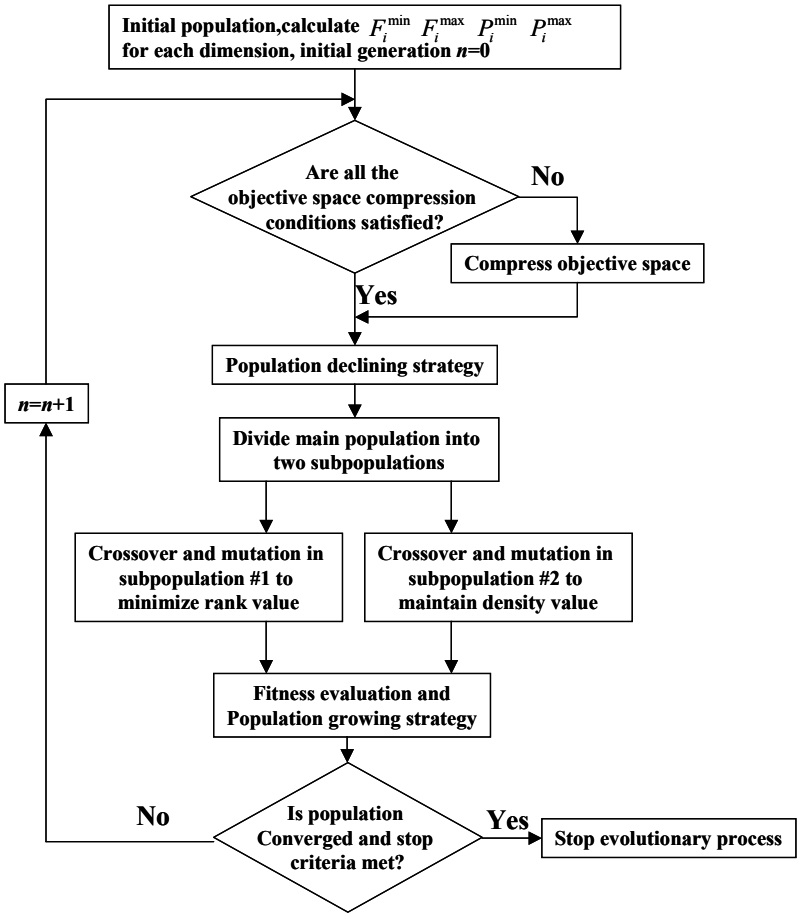


Figure 7.8 Flow chart of DMOEA

7.6 Simulation I—Testing Study on DMOEA

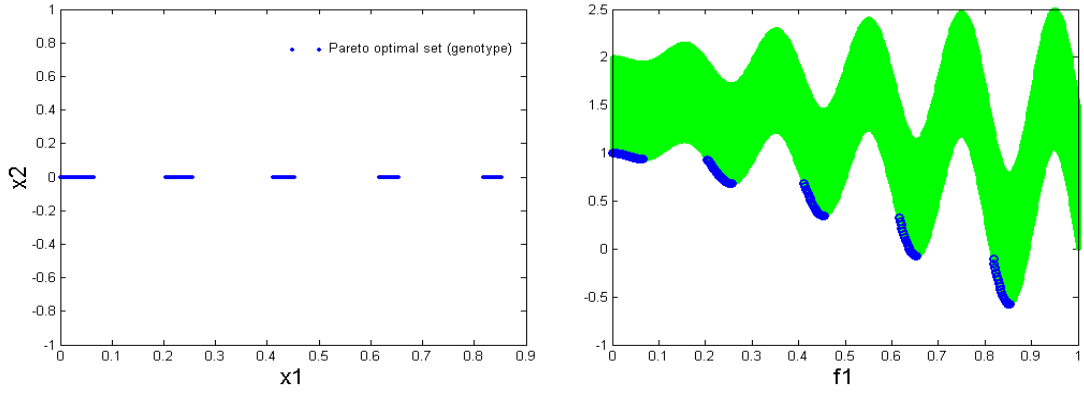
Here a modified MOP designed by Deb [82] is used as the test function ***F5*** that has a discontinuous Pareto front to examine the performance of DMOEA. Figures 7.9(a) and (b) show the Pareto optimal set (i.e., in terms of decision variables, x_1, x_2) and true Pareto front (i.e., in terms of objective variables, f_1, f_2) for this problem.

Minimize $f_1(x_1, x_2)$ and $f_2(x_1, x_2)$, where

$$f_1(x_1, x_2) = x_1,$$

$$f_2(x_1, x_2) = (1 + x_2) \times (1 - \frac{x_1}{1 + x_2})^2 - \frac{x_1}{1 + x_2} \times \sin(10\pi x_1) \tag{7.13}$$

with $0 \le x_1, x_2 \le 1$.

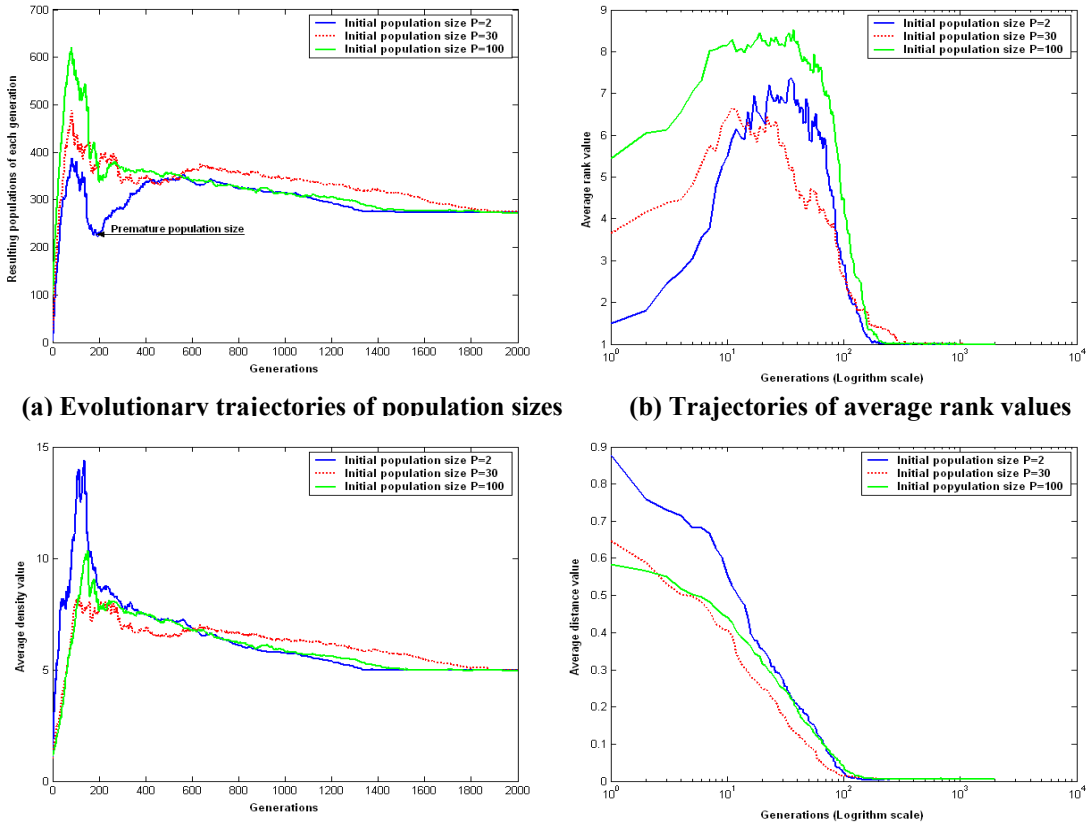


(a) Pareto optimal set of function *F5* **(b) Pareto front of function *F5***

Figure 7.9 Illustration of Pareto optimal set and Pareto front of function *F5*

For the given test function, we select the boundary of the feasible objective space to be [0, 1] and [-1, 2.5] and the number of cells of each dimension to be $K_1 = 50$ and $K_2 = 100$. The desired population size per cell is predefined as $ppv = 5$. Three specified initial population sizes—2, 30 and 100—are chosen to test the robustness of DMOEA. The age threshold, the stopping generation, the chromosome length of each decision variable, the crossover rate and the mutation rate are chosen to be 10, 2000, 15, 0.7, and

0.1, respectively in the simulation. DMOEA is run for 50 times for each selected population size to obtain the average results and for each run, a new initial population with the specified number of individuals is randomly generated and evolved by DMOEA. Moreover, three indicators derived from each generation to quantitatively measure the performance: average population rank value, average population density value and average generational distance value. The final average population rank value, final average population density value and final average generational distance are derived from the last generation and illustrated via Box plots for the test function considered.



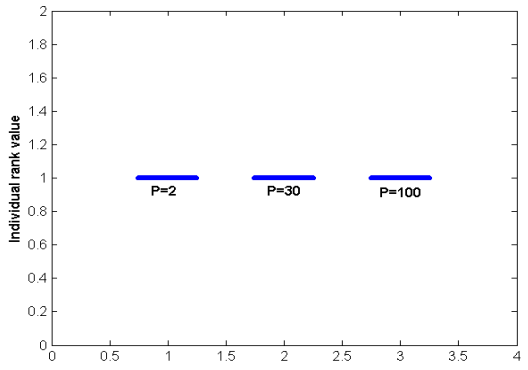
(a) Evolutionary trajectories of population sizes **(b) Trajectories of average rank values**

(c) Trajectories of average density values **(d) Trajectories of average distance values**

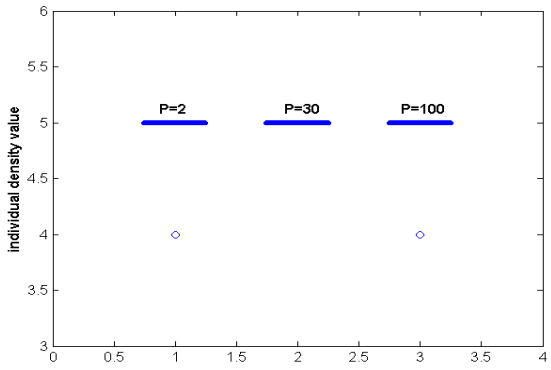
Figure 7.10 Evolutionary trajectories for the population size and the values of three indicators resulting by DMOEA with three different initial population sizes ($A_{th} = 10$) on Function *F5*

The evolutionary trajectories for the average sizes of populations and the values of three indicators over 50 runs are illustrated in Figure 7.10(a), (b), (c) and (d)

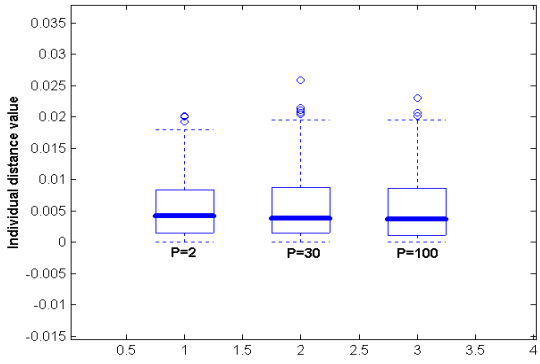
respectively. The corresponding Box plots of the average final indicator values are shown in Figure 7.11. Figure 7.12(a) shows the objective space and true Pareto front of the given test function and Figure 7.12(b) shows the final Pareto front resulted by DMOEA with initial population size $P = 2$. From Figures 7.10 and 7.11, we can observe that for the given MOP test function, chosen grid of cells and predefined ppv value, 275 individuals are determined as the final optimal population size (Pareto set). This implies that there are 55 trade-off cells (hyper-areas $A_{to}(n)$) that contain non-dominated individuals discovered by DMOEA at the final generation.



(a) Box plots for population rank values

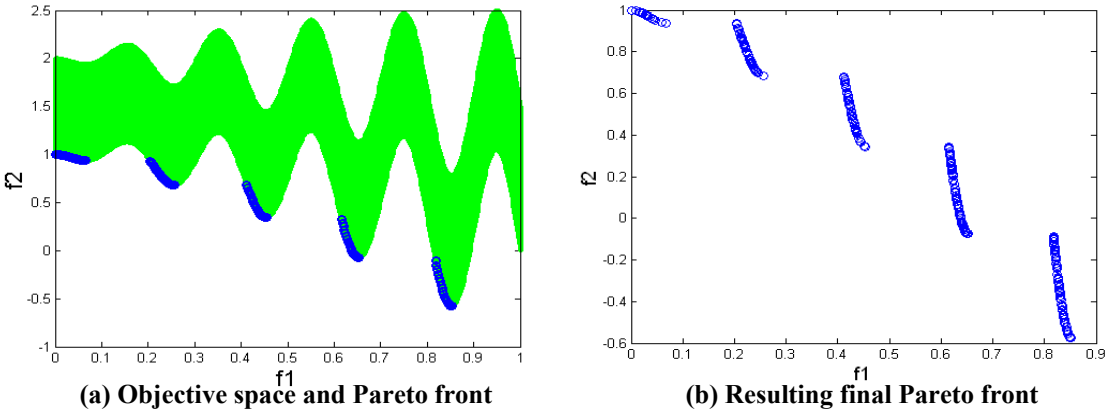


(b) Box plots for population density values



(c) Box plots for population distance values

Figure 7.11 Box plots of three indicators with three different initial population sizes ($A_{th} = 10$) on Function *F5*



(a) Objective space and Pareto front **(b) Resulting final Pareto front**

Figure 7.12 Comparison of the true Pareto front and the final Pareto front resulted by DMOEA ($P = 2$) on Function *F5*

7.7 Simulation II—Comparison Study on DMOEA with Other MOEAs

In order to compare the performance of DMOEA with other advanced MOEAs, three more complex benchmark problems are tested by six MOEAs— PAES, SPEA II, NSGA-II, RDGA, IMOEA and the proposed DMOEA in the simulation, and each of the algorithms runs for 50 times to obtain the statistical results. For each test function, DMOEA will run with the initial population size equal to 2 and achieve an approximated desired population size *dps* . Afterwards, for each of fifty runs, an initial population with *dps* individuals is randomly generated and used by each of three population-based MOEAs (i.e., NSGA-II, SPEA II and RDGA), while only one initial individual is generated for PAES according to its design procedure [52]. The archive size is set to be 100 for all these MOEAs that involve the elitism scheme. For IMOEA, its initial population size is also set to be 2 for a fair comparison. We use three indicators derived from the final generations of 50 runs to benchmark the comparison results via statistical Box plots. They are: average individual rank value, average individual density value and average individual distance. As discussed in Subsections 4.2.3, for an individual,

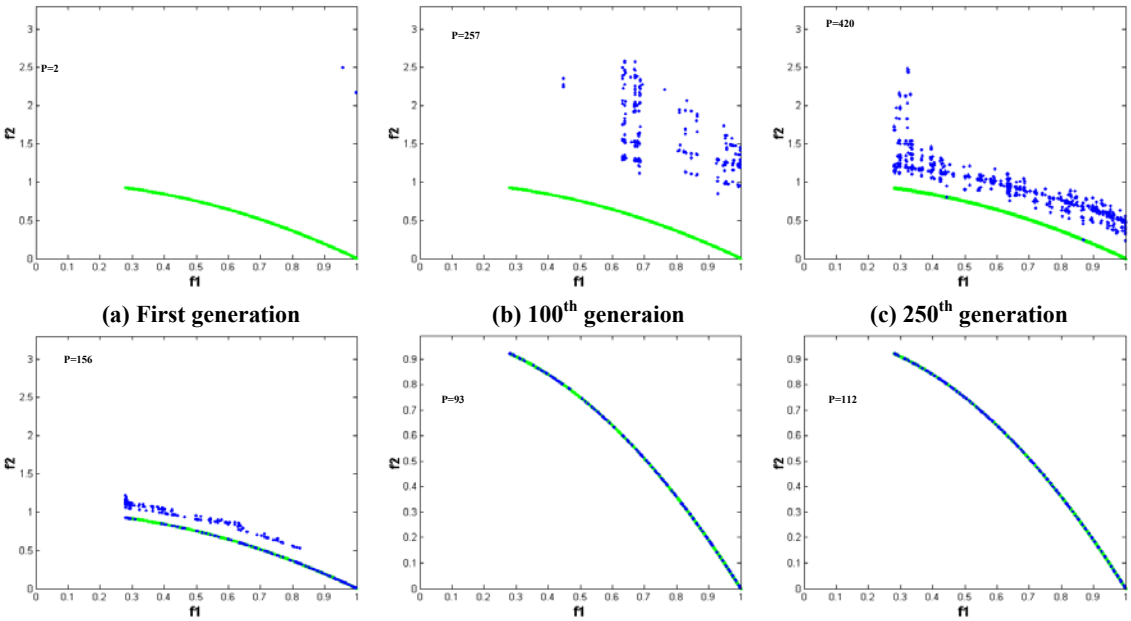
different ranking schemes will produce different rank values, which will be used in respective fitness evaluations and selections. Therefore, for a fair comparison in terms of ranking indicators among different MOEAs, we use Goldberg's pure Pareto ranking method [25] to recalculate the rank value for each individual resulted by each applied MOEAs. Meanwhile, the average individual density value is calculated as the mean value of all the individual density values. Furthermore, the "final average individual distance" is also used as the third indicator to show how far the non-dominated points on the resulting final Pareto front PF_{final} are away from the true Pareto front PF_{true}, where PF_{true} is known in *a priori* for the given test functions. Moreover, in order to compare the dominance relationship between two final populations resulted by two different MOEAs, the coverage of two sets (C value) [13] is also measured to show how the final population of one algorithm dominate the final population of another algorithm.

To examine the performances of the selected MOEAs and the proposed DMOEA on the test functions with different Pareto front features, three numerical benchmark problems are used in the simulation study. Function ***F3*** has been used in Chapter VI, which has a high-dimensional decision space and local and global Pareto fronts in objective space [82]. Function ***F6*** has a high-dimensional decision and a high-dimensional objective space [83]; and its true Pareto front is a surface instead of a curve. Function ***F7*** is advanced from function ***F6***, which also involves high-dimensional decision and objective spaces and the true Pareto front is 1/8 of a unit sphere. For a fair comparison, the stopping generation, the chromosome length of each decision variable, the crossover rate and the mutation rate are chosen to be 10,000, 15, 0.7 and 0.1, respectively for all population-based MOEAs considered. One point crossover is used for

all the population based MOEAs. In addition, we select (1+10)-PAES, and a bit flip mutation rate $1/k$ is used for a chromosome of k genes. The tournament size t_{dom} is chosen to be 2.

7.7.1 F3—MOP with high-dimensional decision space

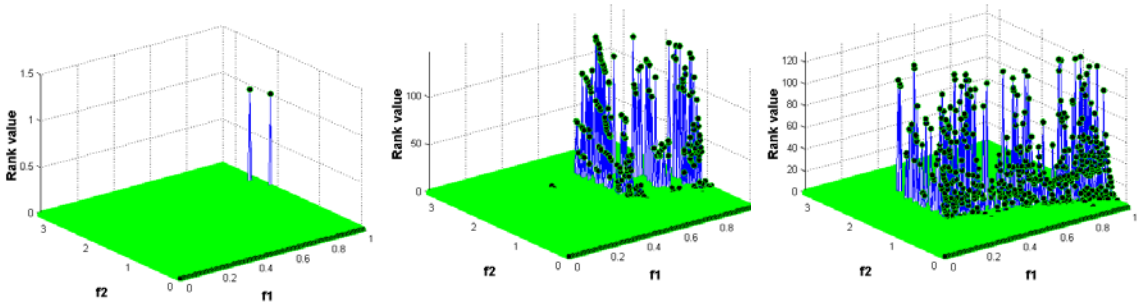
As an MOP with a high-dimensional decision space and local Pareto fronts in objective space, this test function is described as Equation (6.6) and its objective space is illustrated as Figure 6.11. For DMOEA, the initial population, the age threshold, the population size per unit volume, *ppv* and the cell scales K_1 and K_2 are selected as 2, 10, 3, 50 and 50, respectively. Figures 7.13(a) – (f) show the snapshots of the objective space and individuals resulted from DMOEA at generations 1, 100, 250, 750, 1,300 and 10,000, respectively. Similarly, Figures 14(a) – (f) and Figures 15(a) – (f) show the corresponding rank and density values of these individuals resulted from DMOEA at those generations, respectively.



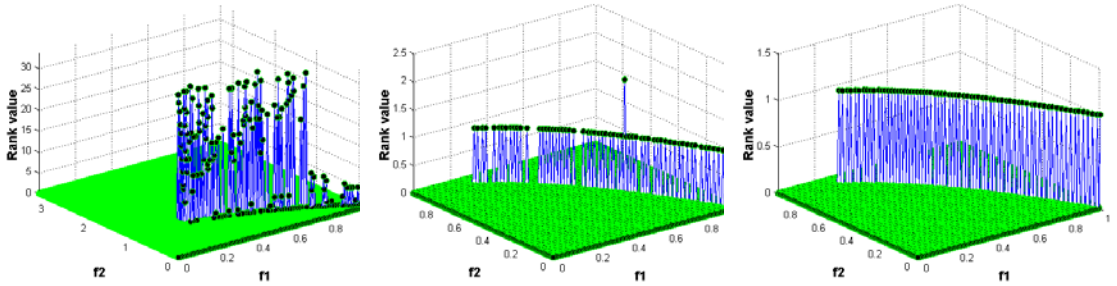
(a) First generation **(b) 100th generaion** **(c) 250th generation**

(d) 700th generation **(e) 1,300th generation** **(f) 10,000th generation**

Figure 7.13 Snapshots of objective spaces and populations resulted from DMOEA on Function *F3*

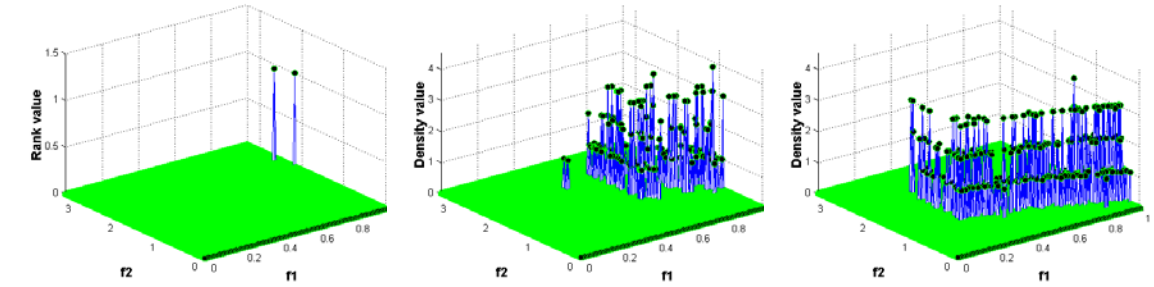


(a) First generation (b) 100th generation (c) 250th generation

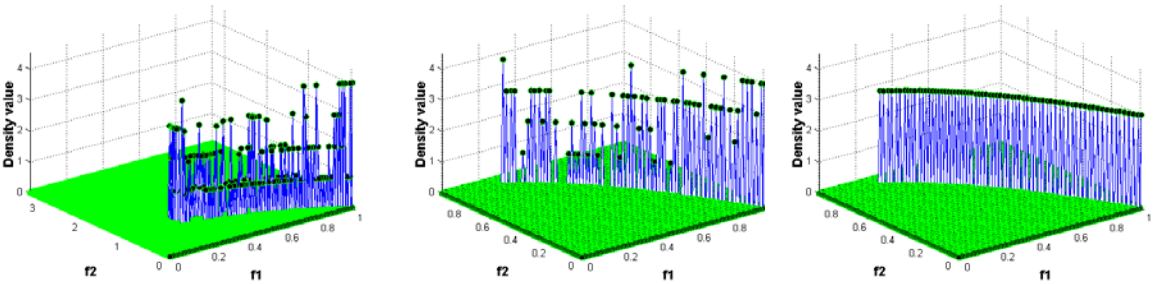


(d) 700th generation (e) 1300th generation (f) 10,000th generation

Figure 7.14 Snapshots of objective spaces and rank values resulted from DMOEA on Function *F3*



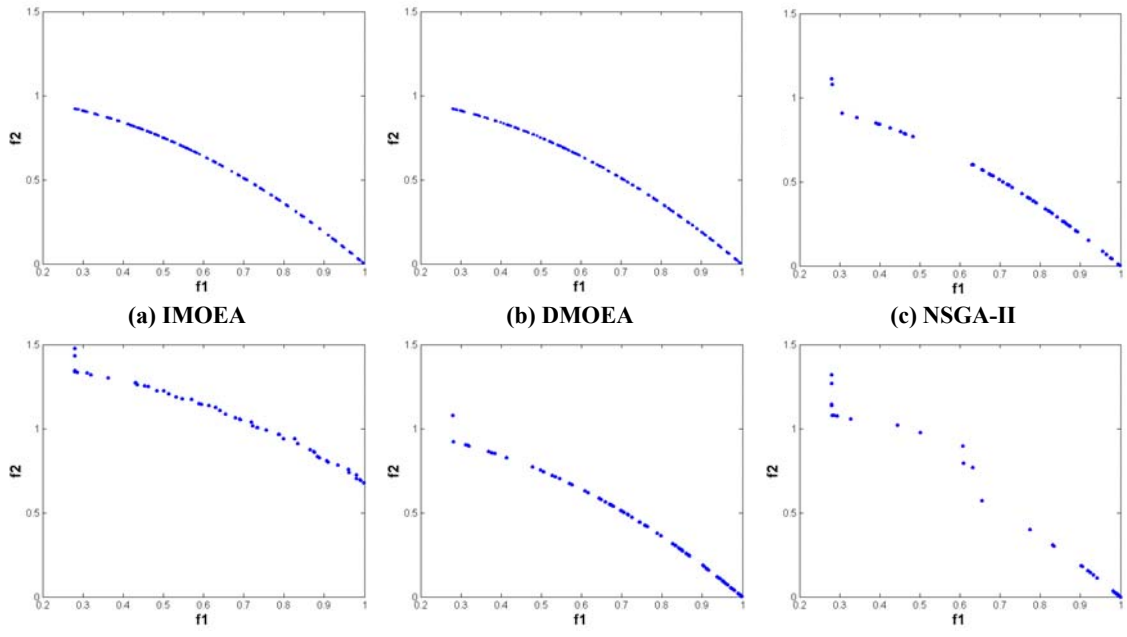
(a) First generation (b) 100th generation (c) 250th generation



(d) 700th generation (e) 1300th generation (f) 10,000th generation

Figure 7.15 Snapshots of objective spaces and density values resulted from DMOEA on Function *F3*

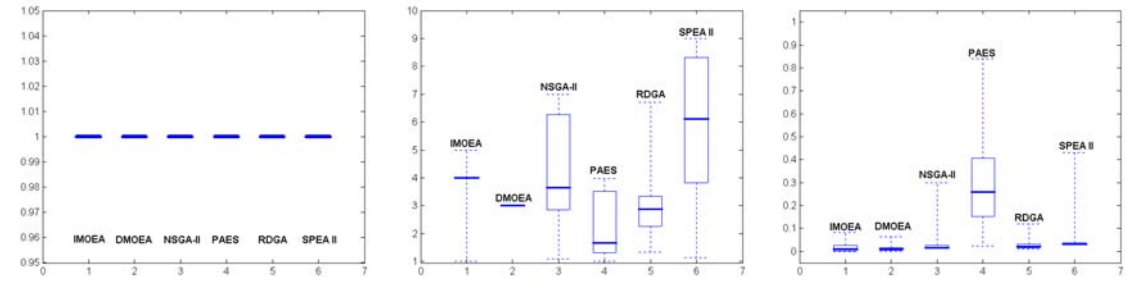
From Figures 7.13 – 7.15, we can observe that although the initial population size is selected to be a very small number, DMOEA can find the true Pareto front easily as shown in Figure 7.13(f). In the beginning, two parents are randomly generated (i.e., $P = 2$) and perform genetic operations (i.e., crossover and mutation). As these two individuals do not dominate each other, and they are located in different "home addresses", their rank and density values are all 1 (Figure 14(a) and 15(a)). At the following generations, because the initial population is far away from the true Pareto front, and the population size is much fewer than the optimal one, the proposed population growing strategy affects the evolutionary process more than the population declining strategy. For this reason, both the population size and rank values of the dominated cells increase very fast to ensure those newly generated individuals disperse to the true Pareto front (Figures 7.13(b) and (c) and Figures 7.14(b) and (c)). Meanwhile, as cell density is preserved by DMOEA, the density values of all the individuals does not change very much as shown in Figures 7.15(b) and (c). When the population moves closer to the true Pareto front, it will be more difficult for the parents to generate better-fitted offspring, which means the population growing strategy has difficulty in balancing the population declining strategy, and both the population size and the cell ranks will decrease as shown in Figures 7.13(d) and 7.14(d). When all the cells rank values drops to 1 and density values are 3 (ppv value), the objective space will be compressed, and the new structure of cells will be created based on the new objective space and the original K_1 and K_2. This procedure will continue until all the individuals are non-dominated points, and the density value of each cell is equal to ppv as shown in Figures 7.13(f), 7.14(f) and 7.15(f).



(a) IMOEA **(b) DMOEA** **(c) NSGA-II**

(d) PAES **(e) RDGA** **(f) SPEA II**

Figure 7.16 Pareto fronts resulted from IMOEA, DMOEA, NSGA-II, PAES, RDGA and SPEA II on Function *F3*



(a) Average individual rank value (b) Average individual density value (c) Averge individual distance value

Figure 7.17 Box plots of average individual rank, density and distance values on Function *F3*

As obtained from the result of DMOEA, the optimal population size for the given grid scale K_1 and K_2 is around 110. Therefore, we run NSGA-II, RDGA and SPEA II with a fixed initial population size of 100 for a fair comparison. In addition, PAES with one initial individual and IMOEA with two initial individuals are also run for 10,000 generations. The *lowbps*, *ppv* and *upbps* in IMOEA are chosen to be 1, 3 and 5, respectively. Figure 7.16(a) – (f) show the resulting Pareto fronts by six chosen MOEAs, while the Box plots for the average values of three indicators over 50 runs are illustrated

in Figures 7.17(a), (b) and (c), respectively. The performance measures of $C(X_i, X_j)$ for the comparison sets between algorithms i and j are shown in Figure 7.18, where algorithms 1 – 6 represent IMOEA, DMOEA, NSGA-II, PAES, RDGA and SPEA II, respectively.

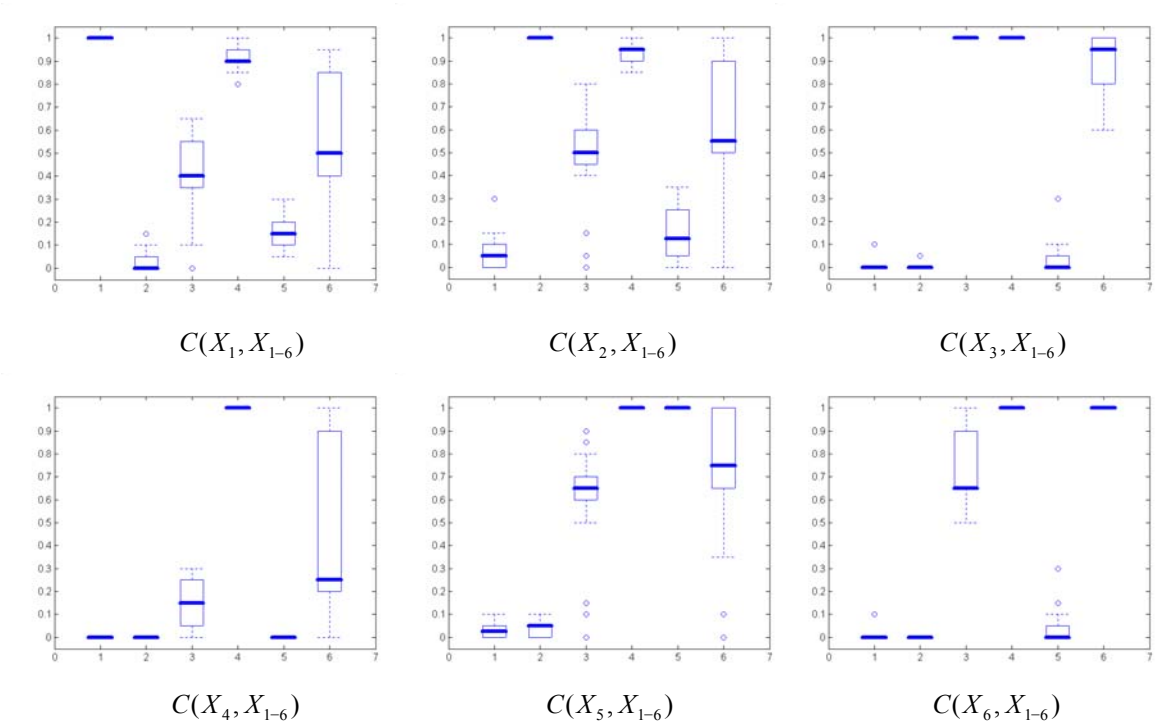


Figure 7.18 Box plots using C measure on Function *F3*

From Figures 7.16 – 7.18, it is obvious that PAES has great difficulty in finding the true Pareto front of this MOP. On the other hand, NSGA-II, SPEA II and RDGA can always identify some points on the global Pareto front. IMOEA and the proposed DMOEA can always find a *near-complete*, *near-optimal* Pareto front. In addition, PAES and IMOEA also result in about 100 individuals at the final generation, which is similar to the optimal population size found from DMOEA. Nevertheless, many individuals resulted from PAES are not located on the global Pareto front and thus PAES produces very low $C(X_4, X_{1-6})$ values as shown in Figure 7.18. Moreover, as shown in Figure

7.17(b), the average individual density value generated by IMOEA is 4 instead of 3 (pre-defined ppv value). Since IMOEA's goal is to meet the desired population size $dps(n)$ at generation n as estimated by Equations (7.1) and (7.2), the cell density value has to be higher than ppv to keep the population size close to its optimal value if some of the hyperareas are not explored. However, for DMOEA, finding $dps(n)$ of each generation is not its primary concern since the final optimal population size can be easily calculated as the cell density is preserved and a complete set of hyperareas are discovered. In this case, DMOEA produces a more complete Pareto front than those by the other five MOEAs, and it also provides the highest $C(X_2, X_{1-6})$ values, which means the solution set that was resulted from DMOEA most likely going to dominate the rest of the solution sets resulted from the other chosen MOEAs.

7.7.2 F6—MOP with high-dimensional objective space

Minimize $f_1(x)$, $f_2(x)$ and $f_3(x)$, where

$$f_1(x) = (1 + g(\mathbf{x})) \cos(\frac{\pi x_1}{2}) \cos(\frac{\pi x_2}{2}), \quad (7.14)$$

$$f_2(x) = (1 + g(\mathbf{x})) \cos(\frac{\pi x_1}{2}) \sin(\frac{\pi x_2}{2}),$$

$$f_3(x) = (1 + g(\mathbf{x})) \sin(\frac{\pi x_1}{2}),$$

$$g(\mathbf{x}) = \sum_{i=1}^{12} (x_i - 0.5)^2,$$

subject to $0 \le x_i \le 1, \ i = 1, \ldots, 12.$

This test function is proposed in [84] as an MOP with high-dimensional decision and objective spaces. Meanwhile, the true Pareto front of ***F6*** is exact the first quadrant of a unit sphere. As the mathematical expression of the true Pareto front is clearly defined,

the distance between the final and true Pareto front can be precisely calculated. The desired population size can be determined based on the ppv value and the grid scales $K_1 — K_3$. According to [84], although NSGA-II can locate most of the population at its final generation on the true Pareto front, the resulting non-dominated individuals are not homogeneously distributed, which implies that this test function produces a great challenge for MOEAs in searching for a good representation of the true Pareto front when it is a surface instead of a curve.

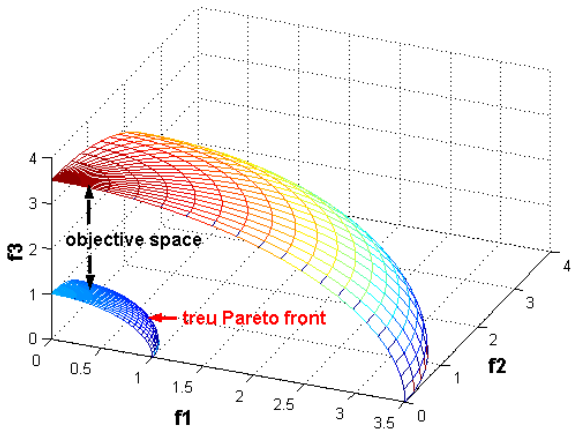
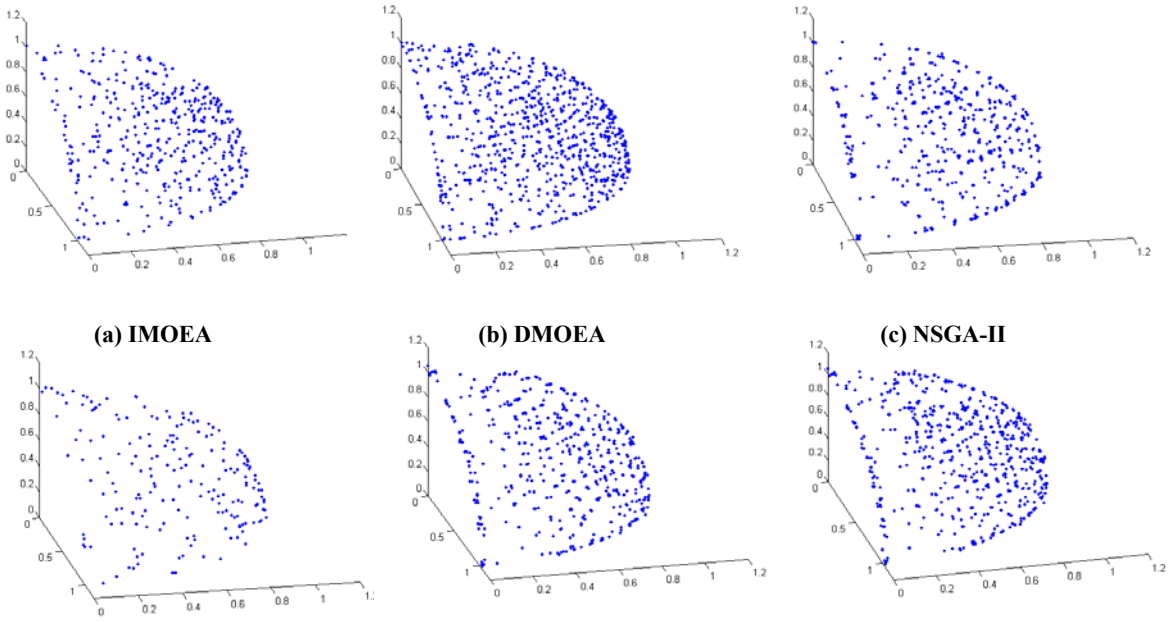


Figure 7.19 Objective space and Pareto front of Function *F6*

The objective space (space between two spheres) and the true Pareto front are shown in Figure 7.19. For DMOEA and IMOEA, the initial population, the population size per unit volume, ppv and the cell scales K_1, K_2 and K_3. are selected as 2, 3, 20, 20 and 20, respectively. The age threshold is chosen to be 10 in DMOEA. At the final generation, DMOEA results in about 1,800 individuals as the approximated optimal population size. Based on this estimation, the initial population size for NSGA-II, RDGA and SPEA II is chosen to be 1,800. Figures 7.20(a) – (f) show the resulting Pareto fronts by six chosen MOEAs and the Box plots for the average values of three indicators over 50 runs are illustrated in Figures 7.21(a), (b) and (c), respectively. The performance

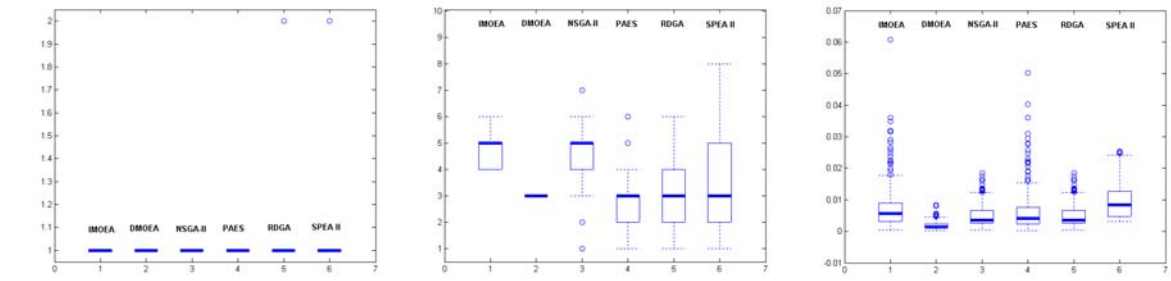
measures of $C(X_i, X_j)$ for the comparison sets between algorithms i and j are shown in Figure 7.22, where algorithms 1 – 6 represent IMOEA, DMOEA, NSGA-II, PAES, RDGA and SPEA II, respectively. Moreover, the evolutionary trajectories of the population size and average individual rank, density and distance values over 50 runs by six selected MOEAs are shown in Figures 7.23 (a) – (d), respectively.



(a) IMOEA **(b) DMOEA** **(c) NSGA-II**

(d) PAES **(e) RDGA** **(f) SPEA II**

Figure 7.20 Pareto fronts resulted from IMOEA, DMOEA, NSGA-II, PAES, RDGA and SPEA II on Function *F6*



(a) Average individual rank value (b) Average individual density value (c) Averge individual distance value

Figure 7.21 Box plots of average individual rank, density and distance values on Function *F6*

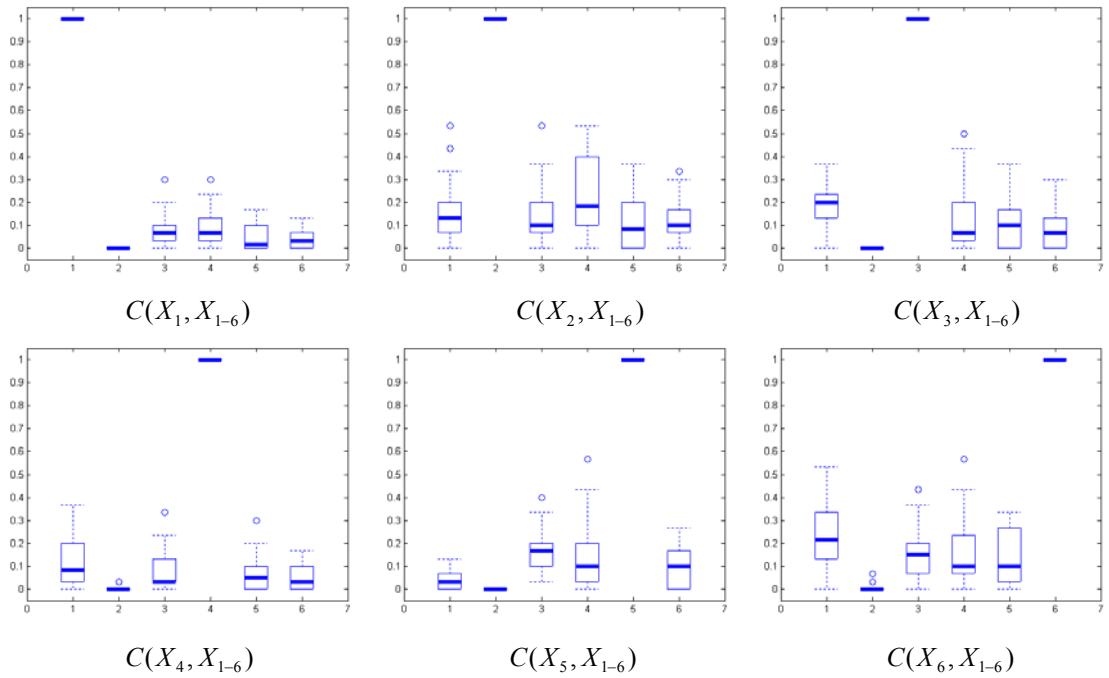
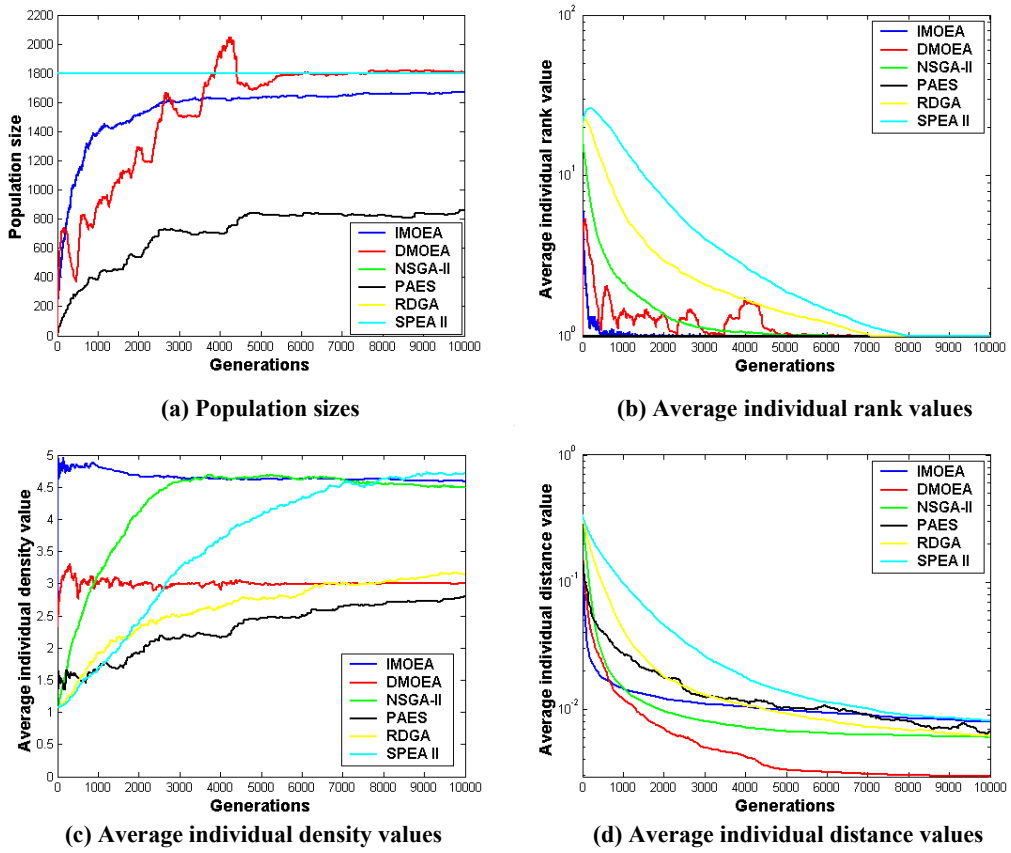


Figure 22 Box plots using C measure on Function *F6*

From Figures 7.20– 7.22, it is obvious that DMOEA produces a more accurate and homogenously distributed Pareto front comparing to the other advanced MOEAs. Indeed, if the initial population size is correctly chosen, the MOEAs with the fixed population size (i.e., NSGA-II, RDGA and SPEA II) also yield to a competent Pareto front in terms of rank, density and distance values as shown in Figures 7.21 (a) – (c) and Figures 7.23(c) and (d). In addition, as the true Pareto front is a surface instead of a curve, it is difficult for the resulting non-dominated sets from any two MOEAs to cover each other. As the result, the C values are relatively low as seen in Figure 7.22. In particular, because the Pareto points resulting from DMOEA have the lowest average individual distance values and a converged average individual density value (as shown in Figures 7.23(d) and 7.23(c)), they are very competitive, which makes the resulting Pareto

front of all the other five MOEAs have great difficulty to cover, and $C(X_1, X_2)$, $C(X_3, X_2), \cdots, C(X_6, X_2)$ values are all near zero.



(a) Population sizes **(b) Average individual rank values**

(c) Average individual density values **(d) Average individual distance values**

Figure 7.23 Evolutionary trajectories of population sizes and average individual rank, density and distance values from six selected MOEAs over 50 runs on Function *F6*

Furthermore, Figures 7.23(a) – (d) also show the convergence speeds of chosen MOEAs. Generally, IMOEA converges very fast since the Fuzzy Boundary Local Perturbation method is used to assist EA in discovering better-fitted individuals at each generation. However, as discussed in Subsection 7.2, IMOEA's primary goal is to estimate $dps(n)$ by Equation (7.1), however, the cell density value is not carefully preserved. As a result, the final population size produced by IMOEA is not very accurate, and the average density value shows an appreciable deviation from ppv value as shown in Figures 7.23(a) and (c). Indeed, for an MOP whose true Pareto front is known, the

optimal population size can be computed if the population per unit volume ppv and cell scales $K_1 \times K_2 \times \ldots \times K_m$ are given. For instance, for the test function ***F6***, assume $ppv = 3$ and $r = K_1 = K_2 = K_3 = 20$, the desired population size dps can be calculated as

$$dps = \frac{1}{8} \times ppv \times 4\pi r^2 \approx \frac{3}{8} \times 4 \times 3.14 \times 400 = 1885, \tag{7.15}$$

which is very close to the final population size discovered by DMOEA. However, according to Equation (7.1), for the same setting, the number of hyperareas is approximated by IMOEA as:

$$A_{to}(n) \approx \frac{\pi^{(m-1)/2}}{(\frac{m-1}{2})!} \times \left(\frac{d(n)}{2} \right)^{m-1} = \pi \times (\frac{\sqrt{3} \times 20}{2})^2 = 942. \tag{7.16}$$

Thus the desired population size $dps(n)$ for IMOEA at generation *n* is calculated as

$$dps(n) = ppv \times A_{to}(n) = 3 \times 942 = 2826, \tag{7.17}$$

which is much larger than the correct value from Equation (7.15). For this reason, to reach the infeasible high value of the desired population size, IMOEA is forced to increase the population size by encouraging more individuals to dwell in the same cell, which explains the high average individual density values shown in Figures 7.21(b) and 7.23(c). Nevertheless, because the lower and upper bound for $dps(n)$— $lowbps$ and $upbps$ are hard constraints, the cell density value cannot be larger than $upbps$. Therefore, the final population size resulting from IMOEA is still held at a reasonable level as shown in Figure 7.23(a).

It is also interesting to observe that some fluctuations occur on the population, rank and density trajectories resulting from DMOEA in Figures 7.23 (a) – (c). This effect is credited to the proposed objective space compression strategy if the original objective space is greater than the surface of the true Pareto front. Each time when all three criteria described in Subsection 7.4 are satisfied, the objective space will be compressed to an extent. As a result, the size of each cell will decrease, and some of individuals originally located in a non-dominated cell will be pushed into a dominated cell, which implies that some cells will have higher rank or lower density values comparing to their previous status. Therefore, the steady state is disturbed, and the population growing and declining strategies start their process simultaneously to fill those sparse areas and remove dominated individuals, and then reach a new steady population size. Because the increase of the rank values is not significant, the likelihood of eliminating those dominated individuals l_1 is low as shown in Figure 7.4, which makes the population growing strategy dominates the population declining strategy and the population size will rise from this stage. When all the sparse cells are filled with certain numbers of new individuals, DMOEA will experience difficulty in finding a better-fitted offspring. Therefore, from this stage, the population declining strategy affects the population more than the population growing strategy, and the population size will decrease until a new steady state is reached. This process keeps refining the population as well as the cell size until the objective space does not have any room to be compressed at the final steady state. In addition, at the final steady state, all the non-dominated cells discovered by DMOEA should have a ppv number of individuals. It is also worthy to note that the individual distance value continuous to drop without any oscillation during the objective

space compression process (Figure 7.23(d)), which helps DMOEA search for *near-optimal* Pareto points.

7.7.3 F7—*MOP with high-dimensional objective space and local Pareto fronts*

Minimize $f_1(x)$, $f_2(x)$ and $f_3(x)$, where

$$f_1(x) = (1 + g(\mathbf{x}))\cos(\frac{\pi x_1}{2})\cos(\frac{\pi x_2}{2}), \tag{7.18}$$

$$f_2(x) = (1 + g(\mathbf{x}))\cos(\frac{\pi x_1}{2})\sin(\frac{\pi x_2}{2}),$$

$$f_3(x) = (1 + g(\mathbf{x}))\sin(\frac{\pi x_1}{2}),$$

$$g(\mathbf{x}) = 12 + \sum_{i=1}^{12}(x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5)),$$

subject to $0 \le x_i \le 1,\ i = 1,\ldots,12.$

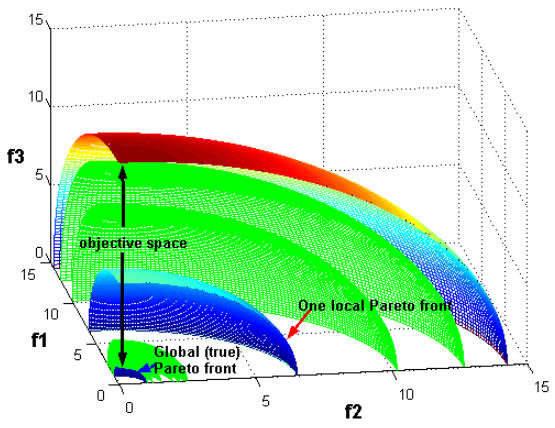
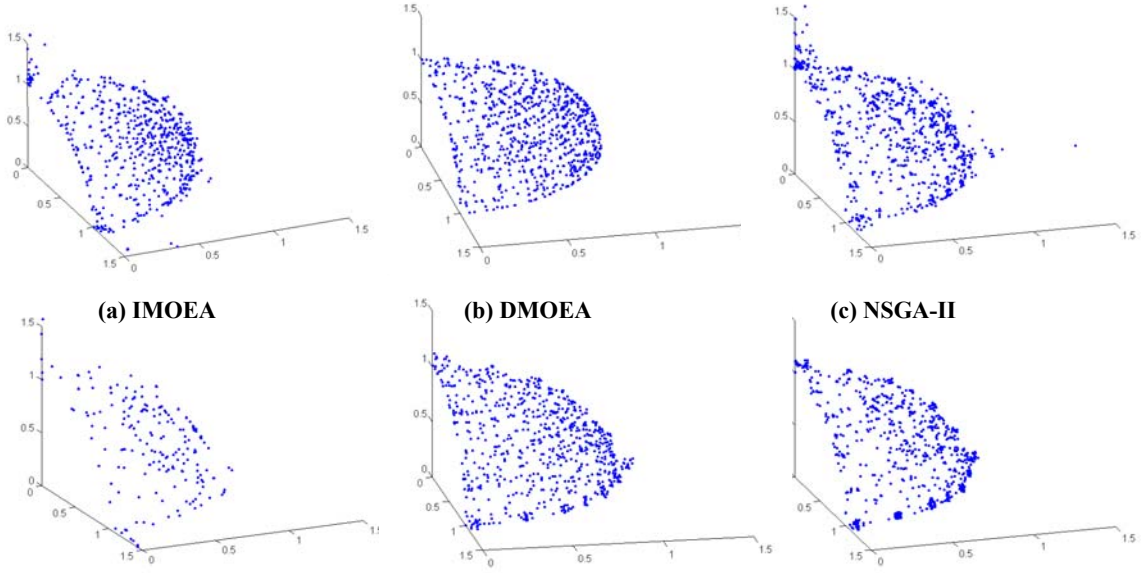


Figure 7.24 Objective space and Pareto front of Function *F7*

This test function is proposed in [84] as an MOP with high-dimensional decision and objective spaces. In addition, function $g(\mathbf{x})$ introduces $(3^{12} - 1)$ local Pareto optimal fronts and one global (true) Pareto front as shown in Figure 7.24. For DMOEA and IMOEA, the initial population, the population size per unit volume, *ppv* and the cell scales K_1, K_2 and K_3 are selected as 2, 3, 20, 20 and 20, respectively. The age threshold

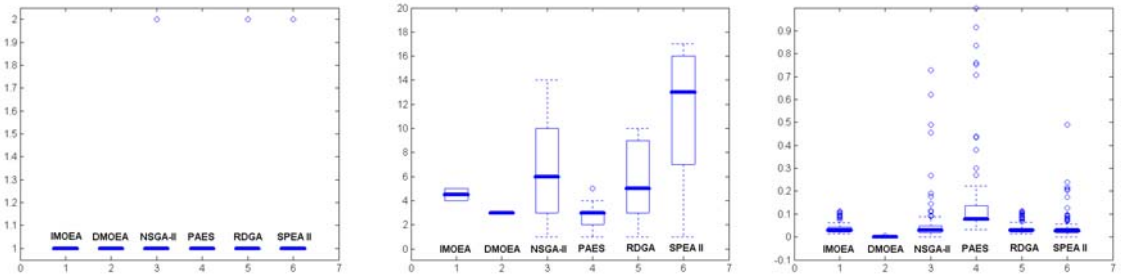
is chosen to be 10 in DMOEA. At final generation, DMOEA results in about 1,700 individuals as the approximated optimal population size. Based on this value, the initial population size for NSGA-II, RDGA, SPEA II is chosen to be 1,700. Figures 7.25(a) – (f) show the resulting Pareto fronts by six chosen MOEAs, while the Box plots for the average values of three indicators over 50 runs are illustrated in Figures 7.26 (a), (b) and (c), respectively. The performance measures of $C(X_i, X_j)$ for the comparison sets between algorithms i and j are shown in Figure 27, where algorithms 1 – 6 represent IMOEA, DMOEA, NSGA-II, PAES, RDGA and SPEA II, respectively.



(a) IMOEA **(b) DMOEA** **(c) NSGA-II**

(d) PAES **(e) RDGA** **(f) SPEA-II**

Figure 7.25 Pareto fronts resulted from IMOEA, DMOEA, NSGA-II, PAES RDGA and SPEA II on Function *F7*



(a) Average individual rank value (b) Average individual density value (c) Average individual distance value

Figure 7.26 Box plots of average individual rank, density and distance values on Function *F7*

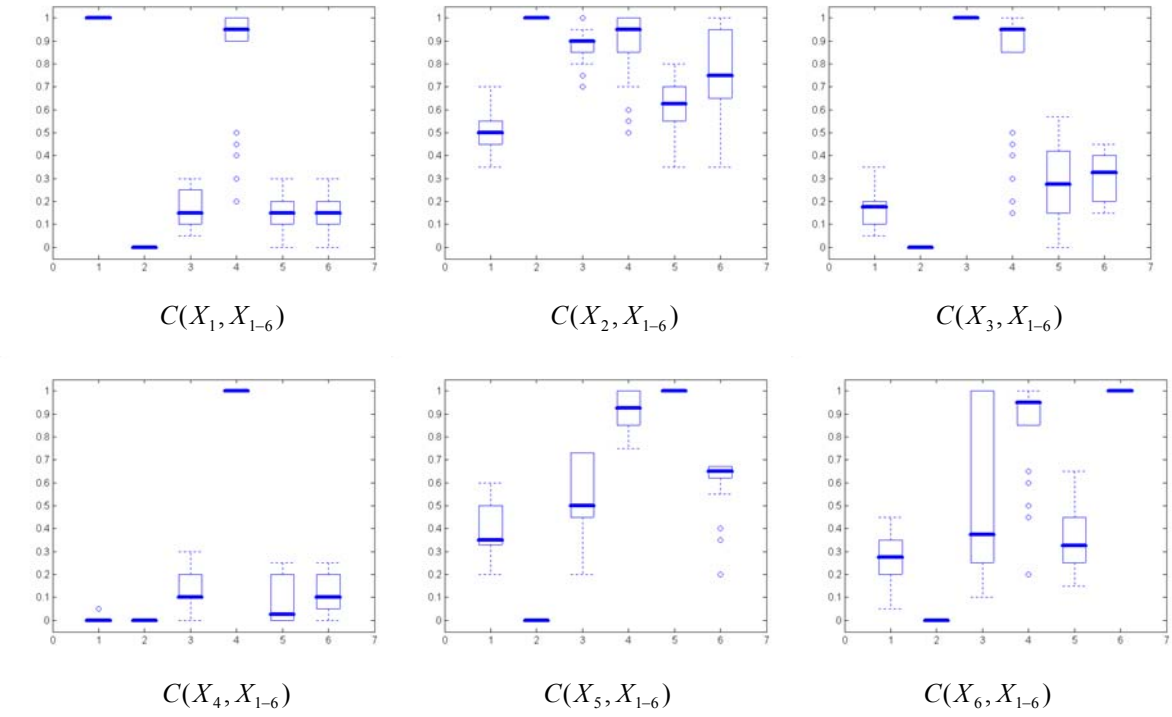


Figure 7.27 Box plots using C measure on Function *F7*

Apparently, from Figures 7.25– 7.27, test function ***F7*** produces great challenges for an MOEA to locate the true Pareto front. As shown in Figure 7.24, many local Pareto fronts exist near the true Pareto front, which means even the rank values of all the individuals are 1, the resulting population may not represent a true Pareto front (Figures 7.25 (a) – (f)). However, comparing to the other five selected MOEAs, DMOEA yields the lowest average individual distance value and a constant individual density value, which implies that DMOEA provides a better performance than the selected MOEAs in terms of discovering a *uniformly distributed*, *near-optimal* and *near-complete* Pareto front. At the final generation, the population sizes resulting from PAES and IMOEA are about 450 and 1,300, respectively, and as shown in Figures 7.25 (a) and (d), many of these individuals stay on the local Pareto fronts. In addition, DMOEA generates higher *C* values than the other chosen MOEAs, and none of the solutions by the other five MOEAs

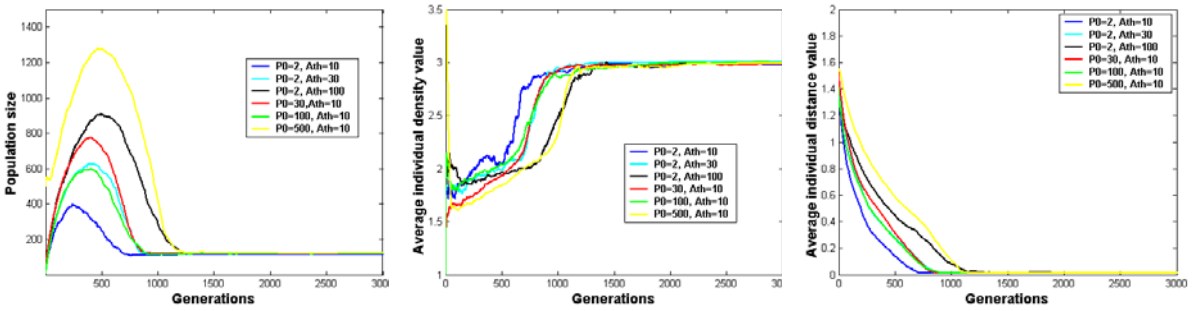
covers the final population of DMOEA since $C(X_1, X_2), C(X_3, X_2), \ldots, C(X_6, X_2)$ values are all near zero.

7.8 ROBUSTNESS STUDY

From the description in Subsection 7.1, the performance of DMOEA may be affected by several parameters such as the initial population size P_0, age threshold A_{th}, the population size per unit volume, ppv and the grid scale $K_1, \ldots, K_m$. Among these parameters, the initial population size and age threshold are the most important ones since the other two parameters are mostly determined by users based on their preferences and requirements in the resolution of the resulting Pareto front. In general, a user may not clearly understand the design mechanism of DMOEA and just randomly select an initial population size and age threshold. Therefore, the relationship between these two parameters and the performances of the final Pareto front needs to be characterized in order to study the robustness of DMOEA based on these two parameters. In Subsection 7.6, DMOEA with different initial population size has been examined by test function ***F5***. The results imply that DMOEA is not sensitive to the setting of initial population size. To further investigate the robustness of DMOEA on different parameter settings, three other test functions—***F3***, ***F6*** ad ***F7*** are used and DMOEA is run for six settings of P_0 and A_{th} on all of three test functions described in Section IV. These settings are: $P_0 = 2, A_{th} = 10$; $P_0 = 2, A_{th} = 30$; $P_0 = 2, A_{th} = 100$; $P_0 = 30, A_{th} = 10$; $P_0 = 100, A_{th} = 10$ and $P_0 = 500, A_{th} = 10$. Figures 7.28(a) – (c), Figures 7.29(a) – (c) and Figures 7.30(a) – (c) show the evolutionary trajectories of the population size, average individual density value and average individual distance value resulted from DMOEA for the given six

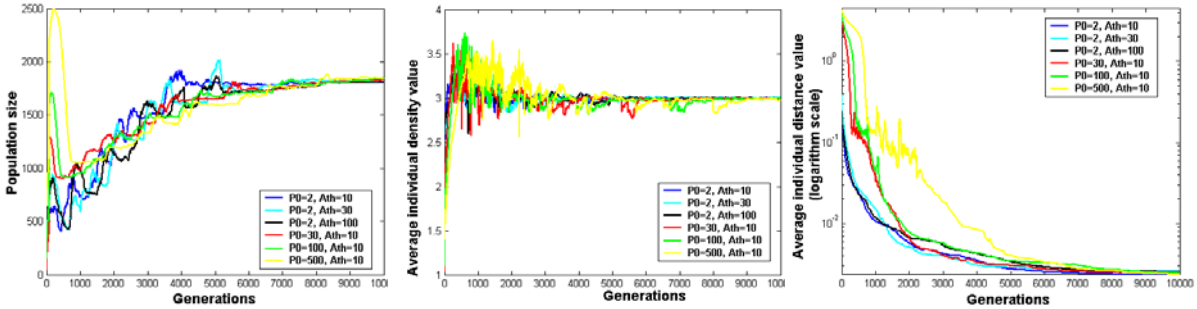
settings over fifty runs on each of three test functions. Note that average individual rank value is not shown in these figures since the rank values are almost always 1 for all the individuals at the final generations. In addition, because test function ***F3*** is relatively simple and DMOEA converges faster on this problem, only the first 3,000 generations are illustrated in Figure 7.28, whereas 10,000 generations are exemplified on functions ***F6*** and ***F7*** as shown in Figures 7.29 and 7.30.

From Figures 7.28– 7.30, it is apparent that no matter which setting we select on DMOEA, the population size, average individual density and average individual distance all converge to a constant value at the final generation, which implies different combinations of initial population size and age threshold may not change the resulting optimal population size and qualities of final Pareto front. However, convergence speed may vary according to different settings. In particular, when initial population size or age threshold values are chosen to be relatively high, the convergence speed will be slow due to the high population size generated in the middle of evolutionary process. Nevertheless, based on the objective compression strategy, this significant high-population size only occurs at the first lobe when the compression action has not started yet. Meanwhile, according to the cell-based rank and density fitness assignment scheme described in Subsection 3.1, the computational complexity will not increase remarkably when the population size increases, thus the computation time will not alter very much even the population size is extraordinary high. Table 7.1 shows the average computation time for test function ***F7*** with 10,000 generations from IMOEA, PAES, NSGA-II, RDGA, SPEA II and DMOEA with six settings over 50 runs.



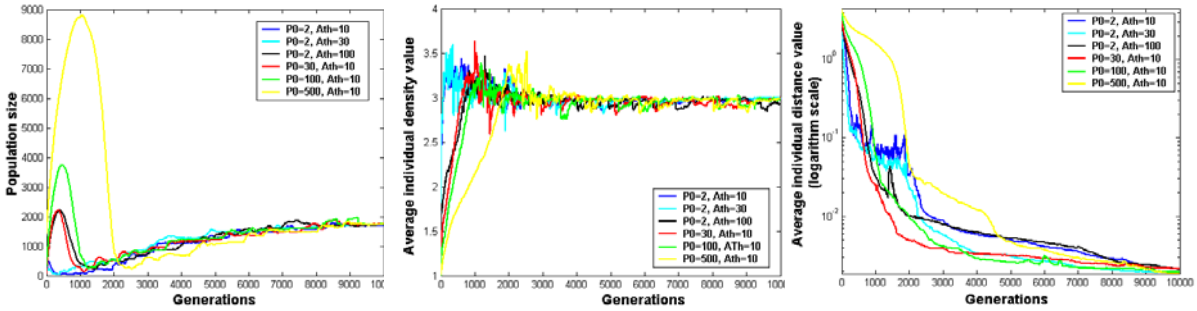
(a) Population sizes **(b) Average individual density values** **c) Average individual distance values**

Figure 7.28 Evolutionary trajectories of population sizes and average individual density and distance values from six settings of DMOEA over 50 runs on Function *F3*



(a) Population sizes **(b) Average individual density values** **c) Average individual distance values**

Figure 7.28 Evolutionary trajectories of population sizes and average individual density and distance values from six settings of DMOEA over 50 runs on Function *F6*



(a) Population sizes **(b) Average individual density values** **c) Average individual distance values**

Figure 7.28 Evolutionary trajectories of population sizes and average individual density and distance values from six settings of DMOEA over 50 runs on Function *F7*

Table 7.1 Comparison results of computation time of *F3* from selected MOEAs and DMOEA with different settings

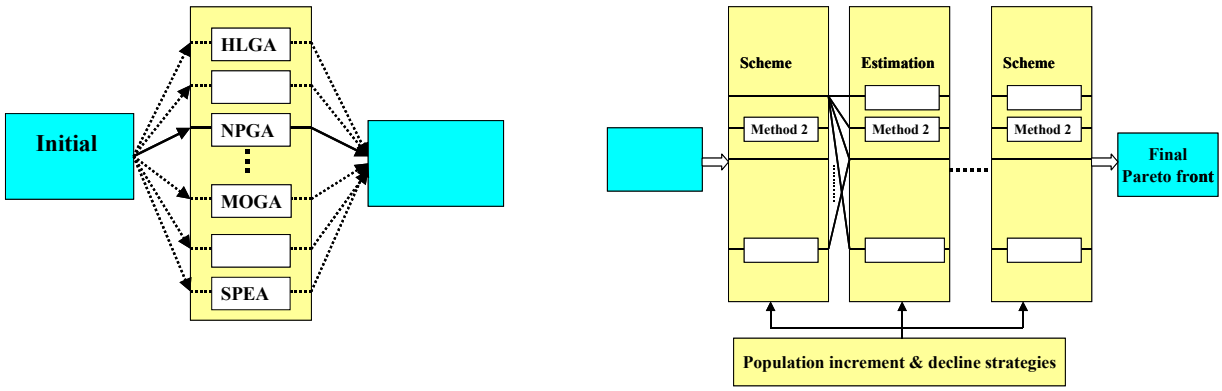
	IMOEA	PAES	NSGA-II	RDGA	SPEA II	DMOEA (2,10)	DMOEA (2,30)	DMOEA (2,100)	DMOEA (30,10)	DMOEA (100,10)	DMOEA (500,10)
Time (min)	**106**	**133**	**251**	**684**	**407**	**25**	**25**	**25**	**26**	**26**	**27**

The "CPUTIME" command from MATLAB (version 6.1) is used to measure the time elapsed for each MOEA implemented in MATLAB. Each MOEA is running in a HP computer with dual 2-GHz processors and 1-GByte RAM. It is worthy of noting the time shown in Table 7.1 provides only a relative measure among chosen MOEAs based on the complexity of the algorithms.

From Table 7.1, we can observe that among all chosen MOEAs, DMOEA demands the shortest running time and the improvement is significant comparing to the other state-of-the-art MOEAs. In addition, different settings will not change the computation efforts of DMOEA and makes the final result of DMOEA robust in terms of both efficien*cy* and effectiveness.

VIII. EMO TOOLBOX DESIGN

As discussed in previous chapters, there are many existing MOEAs in literature and being used by researchers and designers in different research or application fields. Although most of these algorithms were well designed and the algorithms or pseudo codes are optimized, they still require the users (designers) equipped with certain computer programming expertise and an extensive understanding of all the techniques devised. Since most of MOEAs are quite sophisticated due to the complexity of MOPs, the programming effort can be tedious and time consuming and needs to be completed before users can start their design task for which they should really be engaged in [85]. Therefore, a simple solution is to design a user-friendly computer-aided toolbox that includes certain MOEA modules to assist the designers in dealing with particular MOP. The designers merely select a series of build-in modules according to their basic knowledge of MOEAs or help files of the toolbox and input the specific decision variables, objective functions and constraints for the given problem to be solved.



(a) Skeleton of MOEA Toolbox designed by Tan (b) Skeleton of envisioned MOEA Toolbox

Figure 8.1 Comparison of skeletons of two MOEA Toolboxes

By now, an MOEA Toolbox built on MATLAB platform has been designed by Tan *et al* [85]. However, this toolbox does not incorporate those most advanced MOEAs (i.e. NSGA-II, SPEA II and RDGA) and a fixed population size needs to be chosen heuristically by users before the running of a specified MOEA. Furthermore, this toolbox follows the exact design procedure specified by each MOEA to build a fixed object as shown in Figure 8.1(a). However, as mentioned in Chapters IV and V, an MOEA can be divided into several crucial building blocks, such as ranking methods, density estimation approaches, fitness assignment strategies, elitism schemes and some other supplementary routines. Different combinations of these building blocks can result in different types of MOEAs existed, or even lead into some novel MOEAs. For instance, a new MOEA can be configured as: AARS (RDGA) + Crowding distance estimation method (NSGA-II) + elitism + mating restriction (RDGA) + archive truncation (SPEA II), which may provide high performances for some kinds of MOPs. Therefore, by using this building block strategy and dynamic population size, a new Evolutionary Multiobjective Optimization (EMO) Toolbox is designed. This toolbox offers users more flexibilities in choosing their favorite method for each building block; and the population growing and declining strategy can help the resulting algorithms produce a *near-optimal* and *near-complete* Pareto front with an optimal number of individuals. The skeleton of the proposed toolbox is shown in Figure 8.1(b).

The main Graphical User Interface (GUI) of EMO toolbox is shown in Figure 8.2, which includes eight functions. We will describe each of them in this Chapter.

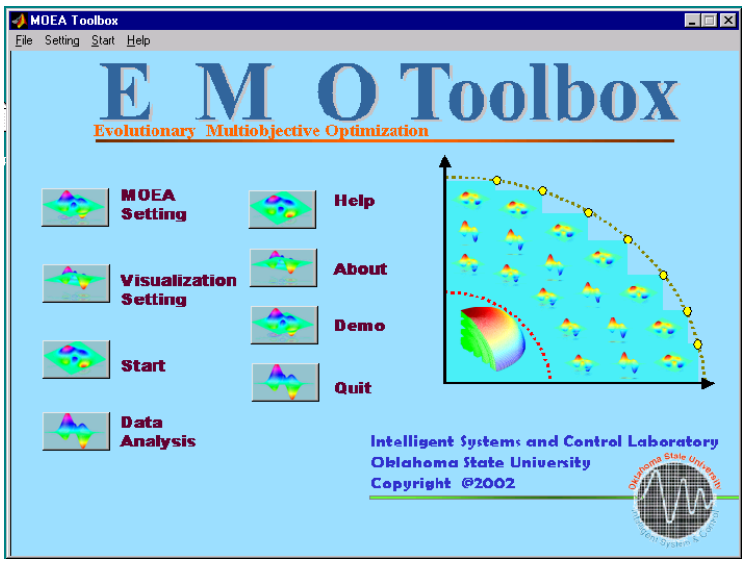


Figure 8.2 Main graphical user interface of EMO Toolbox

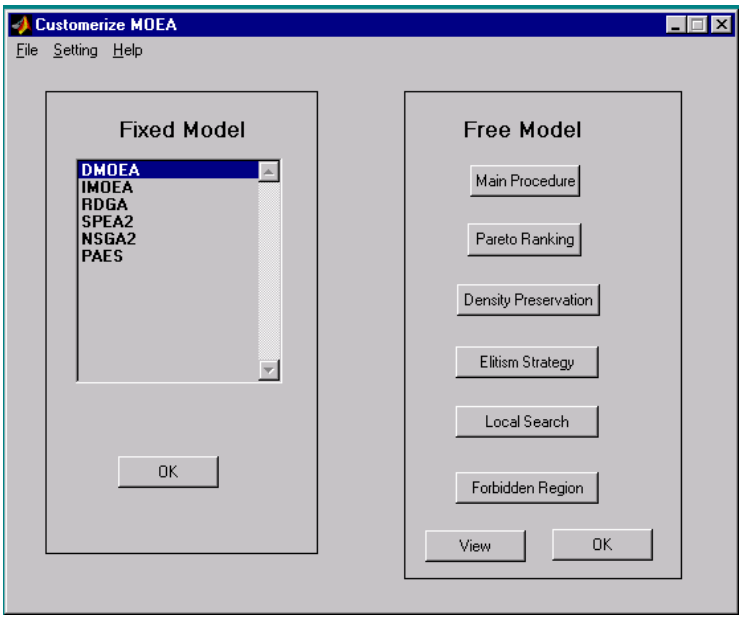
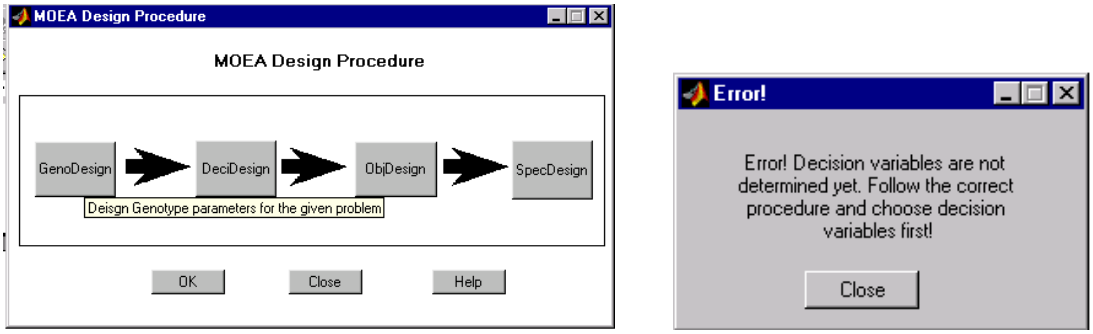


Figure 8.3 GUI of model selection

8.1 MOEA Setting

This function is the main function of toolbox. It provides two alternative choices to the users. The first choice lists six advanced MOEAs (i.e., DMOEA, IMOEA, RDGA, NSGA2, SPEA II and PAES) as discussed in previous chapters, a user can choose any one of them as the algorithm used for the optimization. The design scheme of each of these algorithms is fixed as a build-in function, whereas the design parameters are

specified by users. The second choice offers users more flexible choices when they prefer to design an MOEA by selecting and combining their favorite modules. The GUI for selecting a model is shown as Figure 8.3.



(a) main design procedure **(b) an error message sample**

Figure 8.4 GUI of main design procedure and error message

8.1.1 Main procedure of fixed MOEA model design

As shown in Figure 8.4(a), the main design procedure includes four steps with a predefined sequence: genotype design (GenoDesign), decision variable design (DecDesign), objective function design (ObjDesign) and special parameter design (SpecDesign). The later design procedure cannot be fulfilled until its previous procedure is finished, otherwise an error message will appear (Figure 8.4(b)).

8.1.1.a Genotype Design

As shown in Figure 8.5, genotype parameters (crossover rate, mutation rate, selection method, population size, stopping generations) can be chosen and inputted directly. In addition, current parameter setting can be saved as a MATLAB data file and a previously saved setting can also be loaded to the MATLAB workspace from an existing data file and read by the sliders and editors of the GUI.

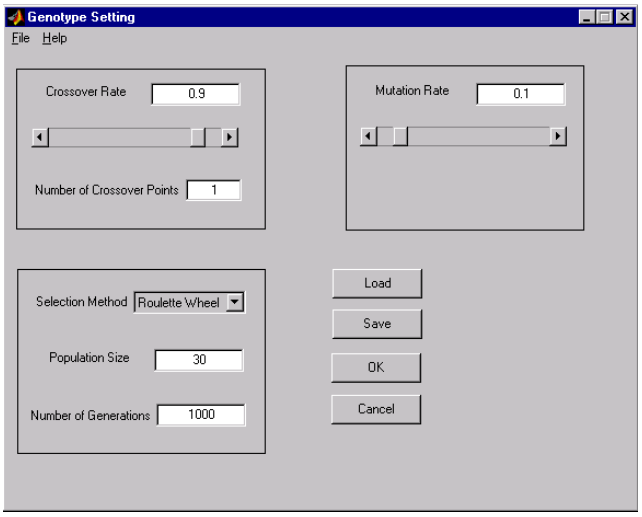
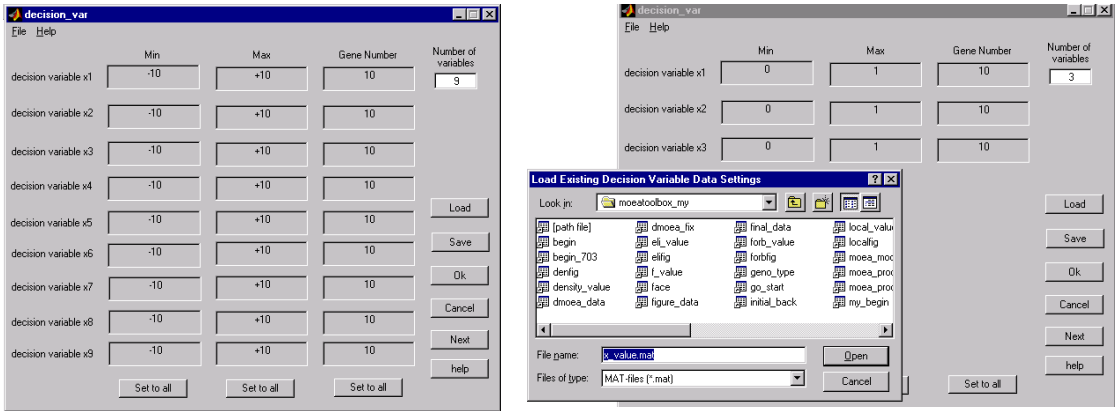


Figure 8.5 GUI of genotype parameter design

8.1.1.b Decision variable setting

As shown in Figure 8.6(a), for each decision variable, there are three parameters need to be determined, maximum value, minimum value and chromosome length (gene number). For each design page, at most 9 variables can be set and if the number of variables are larger than 9, the next design page will appear automatically. Similar to genotype design, the decision variable setting can be saved and loaded (Figure 8.6(b) as well.



(a) GUI of starting a new setting **(b) GUI of loading an existing setting**

Figure 8.6 GUI of decision variable setting

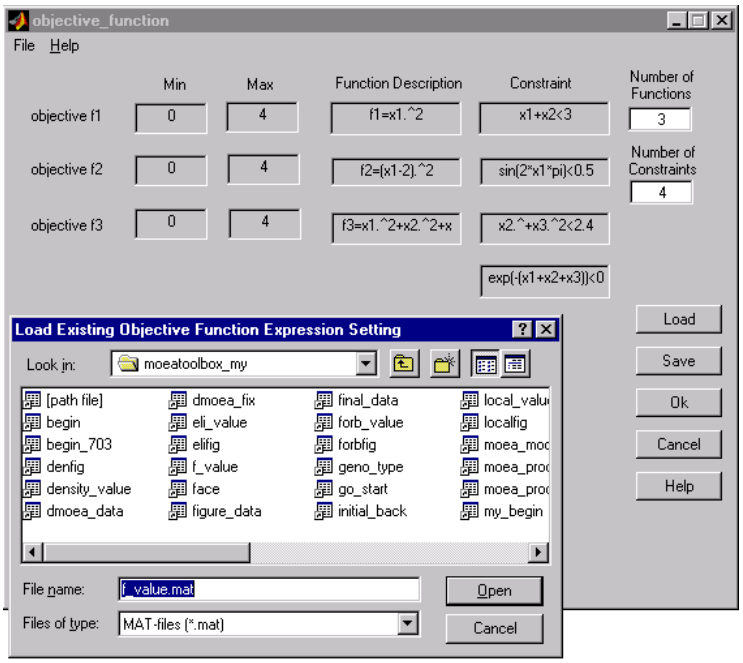


Figure 8.7 GUI of objective function and constraint setting

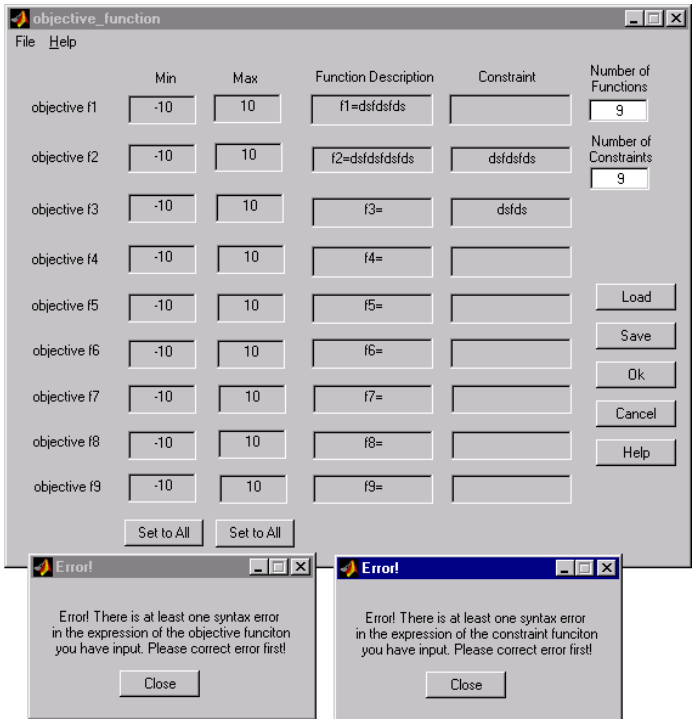


Figure 8.8 Error message for input syntax error

8.1.1.c Objective function and constraint setting

As shown in Figure 8.7, for each objective function, there are three parameters need to be determined, maximum value, minimum value and expression of each objective

function. The mathematical function expression should be compatible to MATLAB format. Moreover, the number of constraints and the constraint expression can also be determined through this GUI (Figure 8.7). When "OK" button is clicked, the function expression will be crosschecked and error messages will appear if there is any syntax error in the expression (Figure 8.8). The error must be corrected before next design procedure starts.

8.1.1.d Special parameter setting

For each specified MOEA model, there are several key parameters need to be determined. For example, Dynamic Multiobjective Evolutionary Algorithm (DMOEA) needs "age threshold", "density grid scale" and "population per cell to be set before the algorithm can properly run (Figure 8.9). After all four steps of main design procedure have been completed, the setting of MOEA parameters is over.

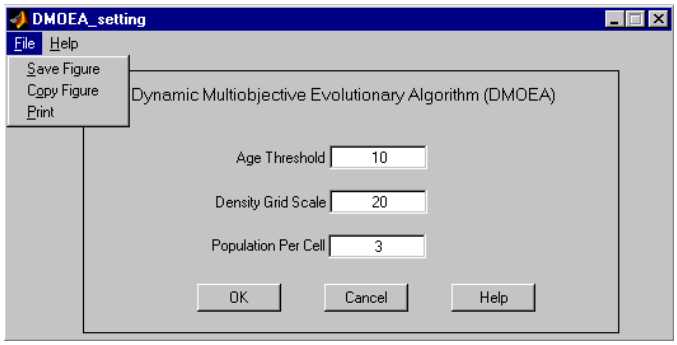


Figure 8.9 GUI of special parameter setting

8.1.2 Main procedure of free MOEA model design

The first three steps of free model design (i.e., genotype, decision variable and objective function) are the same as those in fixed model design. However, in free model design, designers need to choose a particular method for those key schemes (i.e., ranking, density preservation and elitism) in MOEA design.

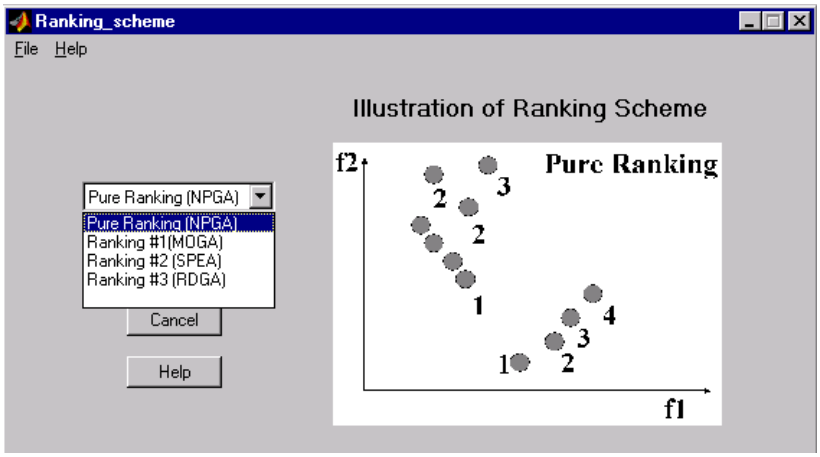


Figure 8.10 GUI of ranking method setting

8.1.2.a Ranking scheme setting

As shown in Figure 8.10, there are four types of ranking methods can be selected by the designers. For each method, a figure is illustrated in order to visualize how the selected method will work. If none of the method is selected, Pure Ranking method used in (NPGA) will be considered as the defaul.

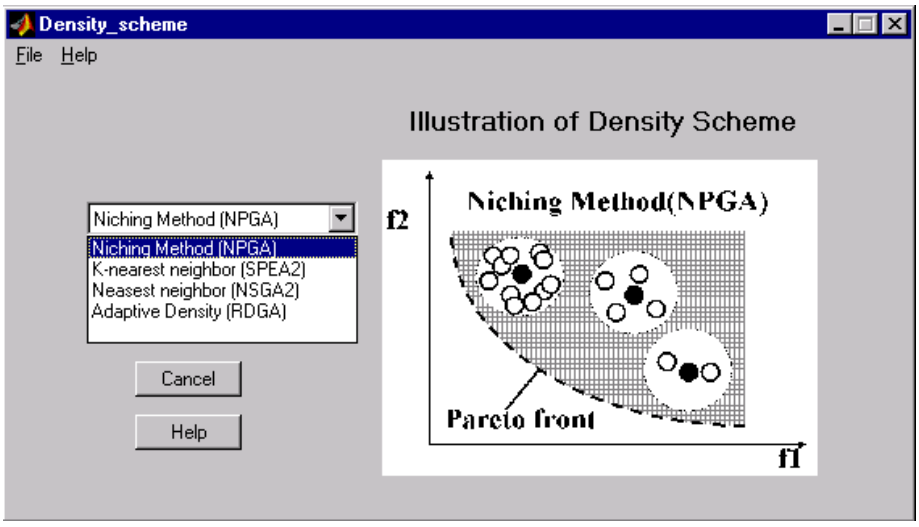


Figure 8.11 GUI of density preservation method setting

8.1.2.b Density scheme setting

As shown in Figure 8.11, there are four types of density preservation methods can be selected by the designers. For each method, a figure is illustrated in order to visualize

how the selected method will work. If none of the method is chosen, the Niching method will be considered as the default and the niche radius needs to be determined as shown in Figure 8.12

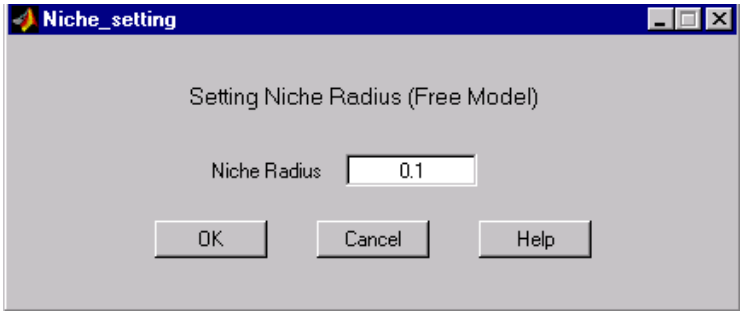


Figure 8.12 GUI of determining niche radius

8.1.2.c Elitism scheme setting

As shown in Figure 8.13, there are three ratios need to be input when designing an elitism scheme. The figure on the right side illustrates the meaning of each ratio. If all the ratios are 0%, there will be no elitism scheme used in the algorithm.

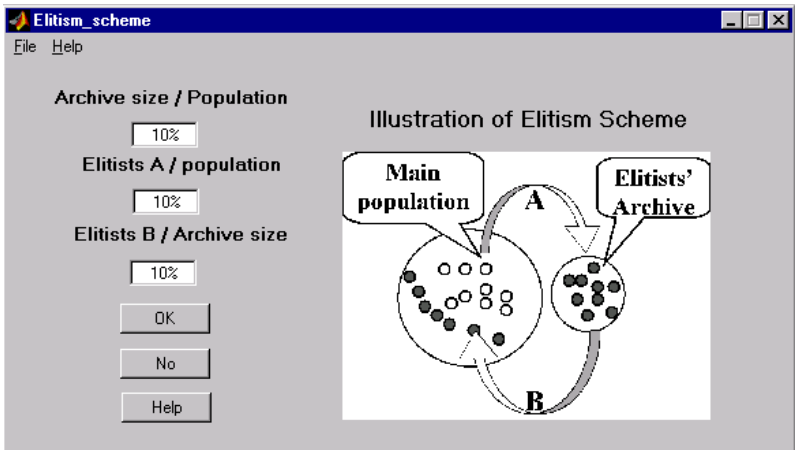


Figure 8.13 GUI of elitism scheme setting

8.1.2.d Local search setting

As shown in Figure 8.14, the local search computation can be restricted within a single cell or the neighboring cells around the concerned cell. The figure on the right side

illustrates how these tow settings will work. If none of the methods are chosen, the designed algorithm will not include local search scheme.

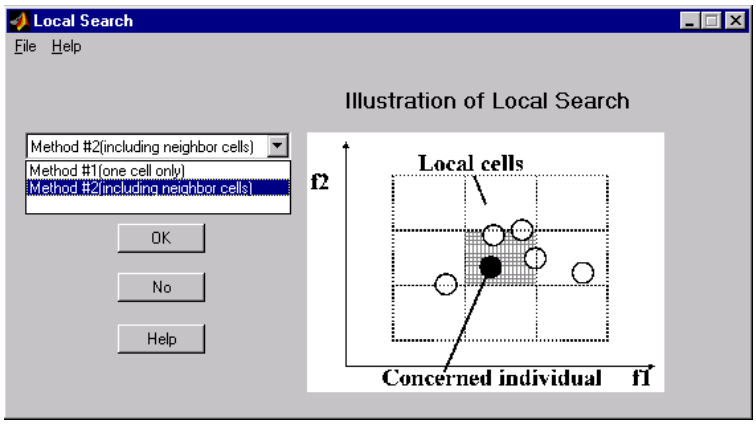


Figure 8.14 GUI of local search setting

8.1.2.e Forbidden region setting

As shown in Figure 8.15, the forbidden region concept will be applied in the designed algorithm if "Yes" button is clicked. Otherwise, the free model MOEA will not apply forbidden region during its evolutionary process.

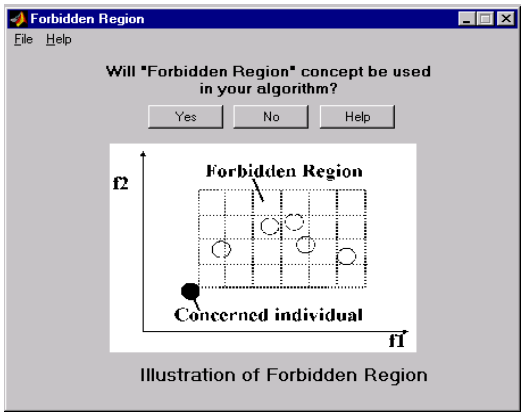


Figure 8.15 GUI of forbidden region setting

After all the schemes have been set, the setting information can be viewed as shown in Figure 8.16. Moreover, if the designers are not satisfied with the current setting, they can change any of them by click the "Reset" button.

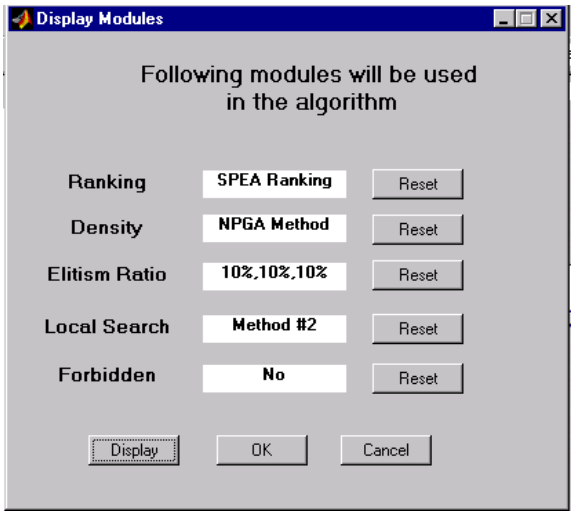


Figure 8.16 GUI of viewing all parts of free model setting

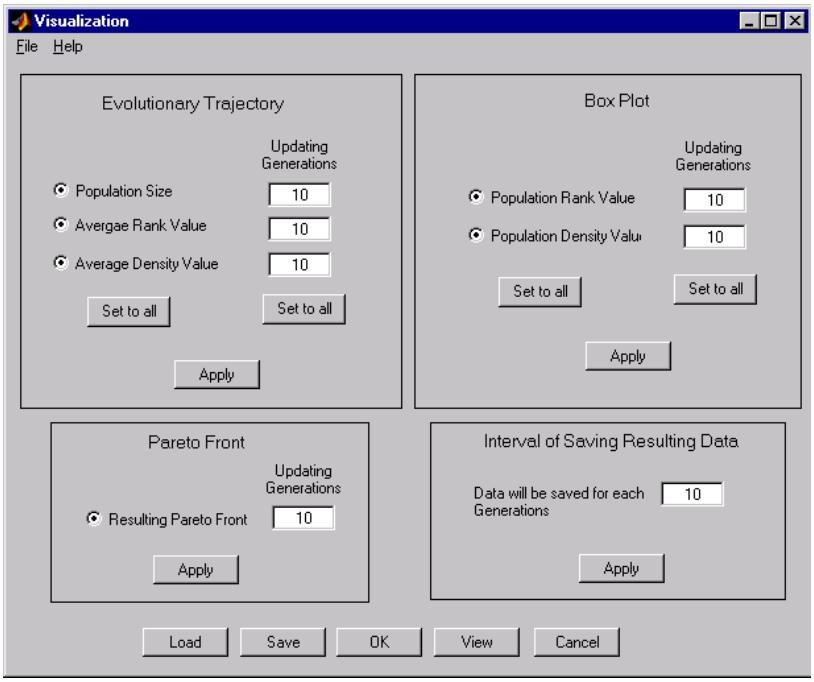


Figure 8.17 GUI of visualization setting

8.2 Visualization Setting

In order to help the users to view the quality of the results during the evolutionary process, visualization parameters need to be set before the algorithm starts running. For example, as shown in Figure 8.17, for each 10 generations, the evolutionary trajectory of the population size, average rank value and average density value will be displayed.

Meanwhile, the resulting Pareto front and the statistical box plots of current population rank and density values will be shown as well. The resulting data will be saved to a user specified data file for each 10 generations according to the setting in Figure 8.17.

8.3 Start Running

When the "Start" button is clicked, Figure 8.18 will show the chosen settings for all the parameters. If the users are satisfied with current setting, the specified MOEA will start running. Otherwise, the parameters will be reset.

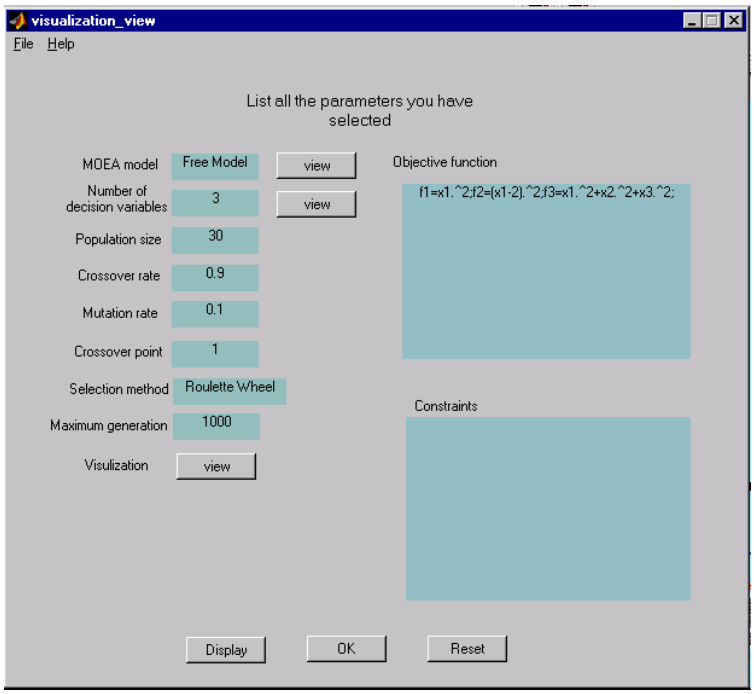


Figure 8.18 GUI of listing of all the chosen parameters.

When the MOEA starts running, it will not stop till the pre-determined visualization interval is met. When the evolutionary result is shown in Figure 8.19, the user can choose "Save Figure" to save the illustrated figure, "Save Data" to save the resulting data, "Continue" to continue running or "Stop" to stop program running.

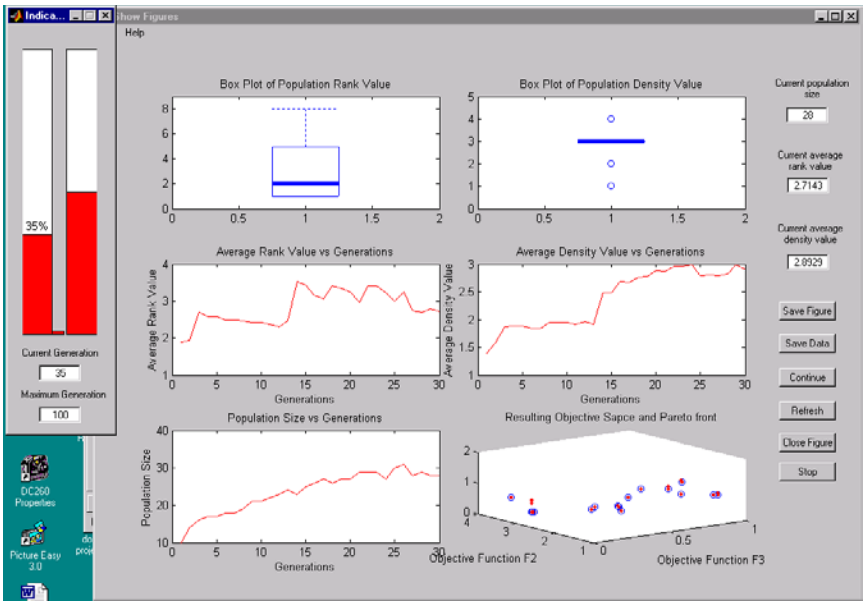


Figure 8.19 GUI of visualizing the evolutionary result for certain intervals

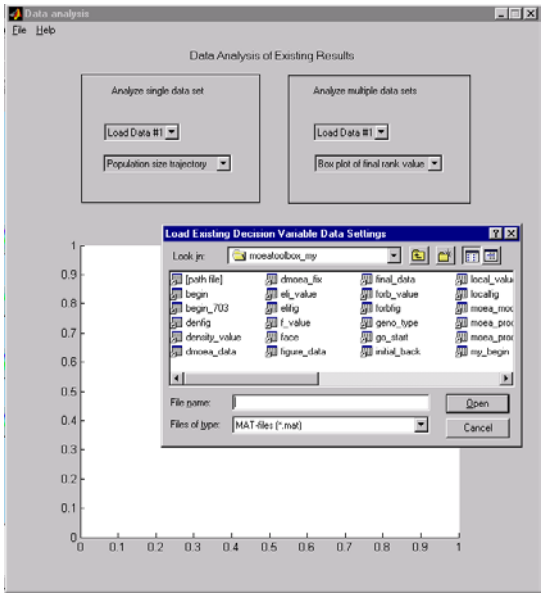
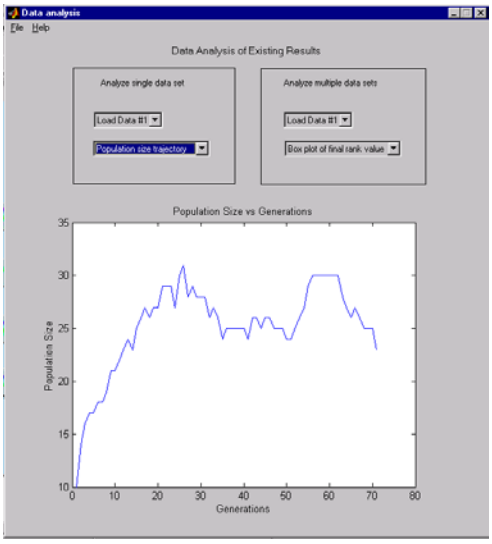


Figure 8.20 GUI of loading data files for analysis

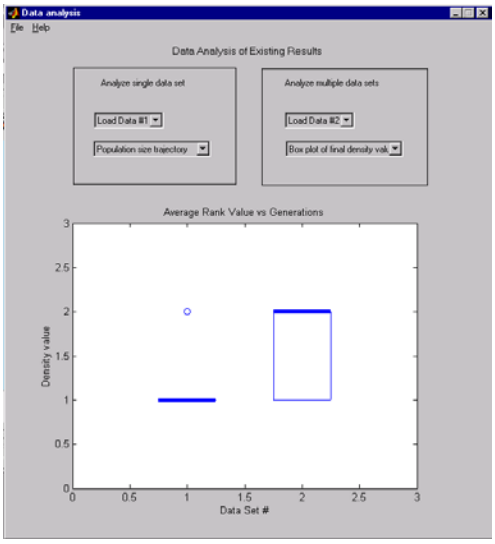
8.4 Data Analysis

EMO toolbox can also help users to analyze existing data as shown in Figure 8.20. By loading an MOEA resulting data file, the history record of evolutionary trajectories, statistical Box plots and final Pareto front can be visualized (Figure 8.21(a)).

Moreover, the toolbox allows users to perform comparisons of more than one data files resulting from different MOEAs. The Box plots of final rank, density and *C* values can be compared (Figure 8.21(b)).



(a) GUI of single data set analysis;



(b) multiple data sets comparison

Figure 8.21 Data analysis for resulting data

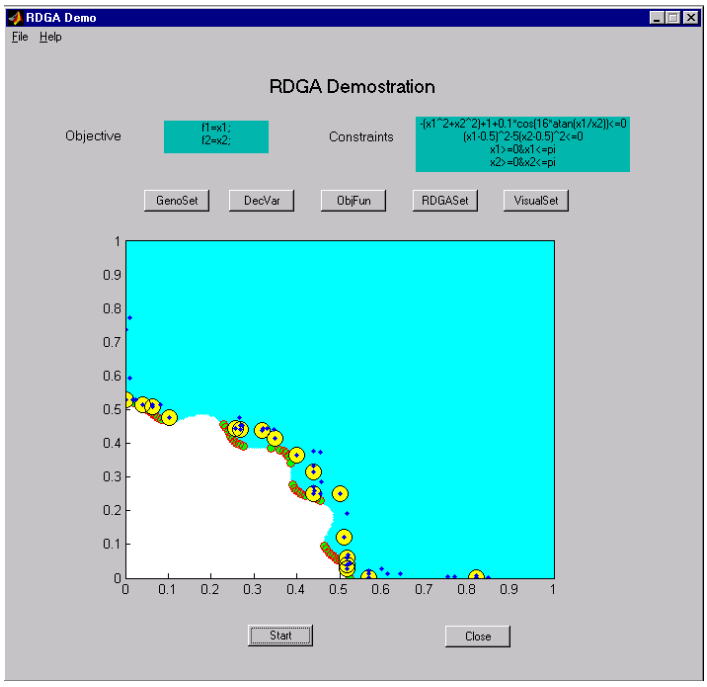


Figure 8.22 GUI of toolbox demonstration

8.5 Demonstrations

To guide the complete procedure of designing an MOEA by using EMO toolbox, a RDGA is designed as the demonstration for the MOP test function described as Equation (6.2). Each design step can be illustrated by clicking one of five buttons above the figure. The figure will show the movie file of how the population and resulting Pareto front will evolve as RDGA runs with given parameter settings (Figure 8.22).

8.6 Help Files

A complete help file is created and associated with each "Help" button in all the GUIs. Figure 8.23 illustrates the "Help Contents" of EMO toolbox.

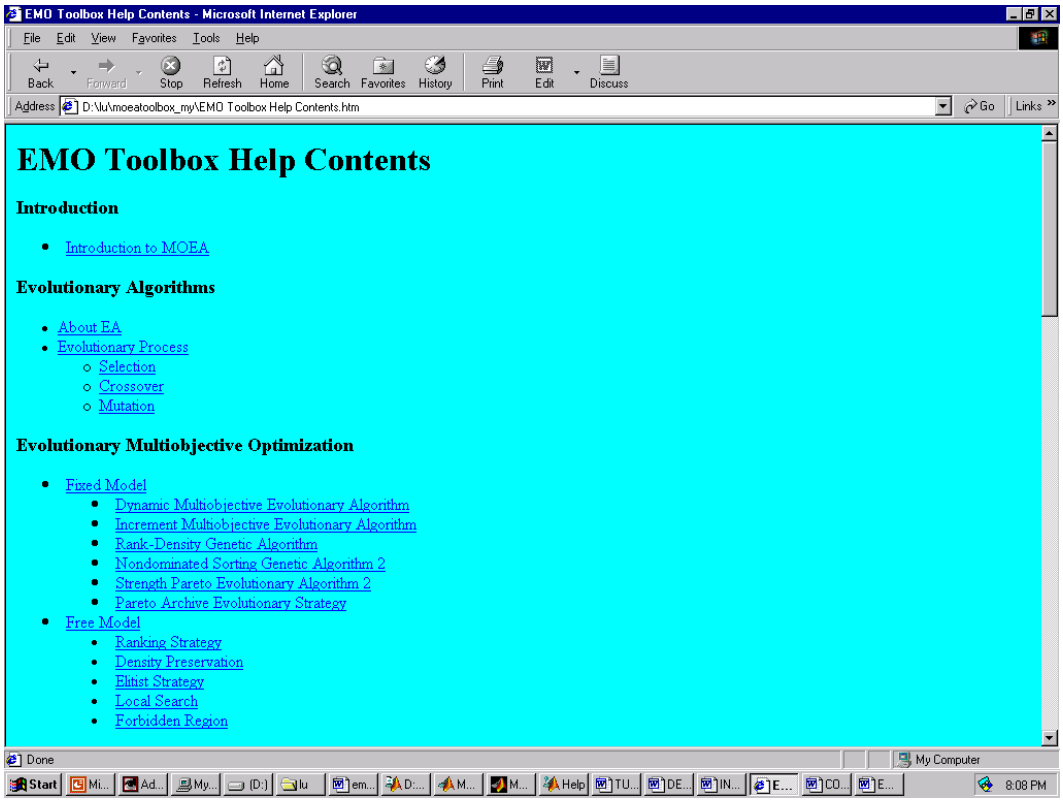


Figure 8.23 GUI of help contents of EMO toolbox

IX. PARTICL SWARM OPTIMIZATION IN MOEA

Although evolutionary algorithms have shown their unique advantages in solving multiobjective optimization problems, their drawback is also obvious—need relatively longer time in producing a high quality Pareto front comparing to the traditional optimization methods (i.e., linear weighting method). This low-efficiency problem is resulted from EA's population-based information sharing and random variation characteristics, which cannot be overcome by evolutionary algorithm itself. Although in Chapter VII, Dynamic Multiobjective Evolutionary Algorithm (DMOEA) proposed a promising way to improve the computational efficiency of MOEA by applying dynamic population strategies, it is still restricted by EA's intrinsic properties. Therefore, in order to aim at improving efficiency of MOEA, we need to search for a clever technique to assist MOEA to achieve a *near-complete*, *near-optimal* and *uniformly distributed* Pareto front with a faster convergence speed. Particle Swarm Optimization (PSO) is considered to be such a candidate.

9.1 Particle Swarm Optimization

Particle Swarm Optimization (PSO) was first proposed by Kennedy and Eberhart [86] in 1995, which was inspired by the choreography of a bird flock. This technique can be seen as a distributed behavior algorithm that performs multidimensional search [87]. According to PSO, the behavior of each individual is affected by either the best local or the best global individual to help it fly through a hyperspace. Moreover, an individual can learn from its past experiences to adjust its flying speed and direction. Therefore, by

observing the behavior of the flock and memorizing their flying histories, all the individuals in the swarm can quickly converge to *near-optimal* geographical positions with well-preserved population density distribution.

Normally, PSO is considered as an evolutionary computation approach in that it uses the common evolutionary computation techniques such as:

1. It is initialized with a population of random solutions.
2. It searches for the optimum by updating generations.
3. The adjustments of individuals are analogous to real value crossover operation in evolutionary algorithms.
4. Fitness evaluation is evaluated by objective functions.

However, the updates of the individuals are not accomplished by random crossover or mutation of genes, an equation can compute the new velocity of each individual i at the jth dimension based on its current location $x(i,j)$, previous velocity $V_p(i,j)$, previous location $p_{best}(i,j)$ at which the highest fitness value this individual has been achieved, and the population global location ($g_{best}(j)$) at which the highest fitness value the population has achieved. Therefore, the velocity updating equation is

$$V_p(i,j) = \omega V_p(i,j) + R_1(p_{best}(i,j) - x(i,j)) + R_2(g_{best}(j) - x(i,j)), \tag{9.1}$$

where ω is an inertia weight value [88] and R_1 and R_2 are two random numbers between 0 and 1. After the velocity is updated, the new location of ith individual at the jth dimension can be calculated as

$$x(i, j) = x(i, j) + V_p(i, j) \tag{9.2}$$

Comparing to evolutionary algorithms, the information sharing mechanism in PSO is significantly different. In EAs, individuals share their information with each other by crossover and the whole population moves like one group towards an optimal point. In PSO, only $g_{best}(j)$ provides the information to other individuals to adjust their speeds. It is a one-way information sharing mechanism [89]. The entire population focuses on the best individual and converges to the best solution quickly.

Due to PSO's single-point-centered characteristic, it is unable to locate the Pareto front since there are more than one best individuals exist in the population. However, with certain modifications (i.e., Pareto ranking [43]+ niche sharing [42], neighborhood method [54]), PSO can become suitable to solve MOPs. By now, there are very few papers [89-92] found to extend PSO in solving MOPs, this research area is still in its beginning stage.

9.2 Dynamic Particle Swarm Multiobjective Optimization (DPSMO)

In this research, to tackle multiobjective optimization problems, PSO is devised with dynamic population size proposed in Chapter VI. In another word, DMOEA's crossover and mutation scheme is replaced by PSO's information sharing method in order to improve convergence speed. To prevent the degradation of the effectiveness and efficiency of the algorithm, the following strategies are applied in the new algorithm:

1. The genotype of each individual will be a real number instead of binary genes.
2. For each individual, its genotype will includes two types of velocity parameters—rank velocity and density velocity. On each dimension of the decision vector, an individual will be assigned with a rank and a density velocity.
3. Cell rank value of each individual is still calculated, all the individuals with rank value equal to 1 are the global best (rank) individuals. However, for any individual **A**, only those best (rank) individuals that dominate it will be considered as the candidates of **A**'s g_{best_rank}. If more than one candidates of g_{best_rank} exists, the one with lowest density value will be selected as the g_{best_rank} of individual **A**.
4. For any individual **A**, its local best (rank) individual p_{best_rank} is randomly selected from the individuals that are located in the same cell and dominate **A**. If there is no such kind of individual exists, p_{best_rank} will be individual **A** itself.
5. Cell density value of each individual is calculated. For any individual **A**, its best (density) global individual g_{best_den} is the individual that has the lowest cell density value (except those reside in "forbidden region").
6. For an individual **A**, its local best (density) individual p_{best_den} is randomly selected from the individuals that are located in the same cell or neighboring cells (except those reside in "forbidden region") and has the lowest cell density value.
7. The entire population is equally and randomly divided into two subpopulations that responsible for minimizing rank value and maintaining density value, respectively. All the individuals will be cloned and the location of its copy will be

updated based on its new rank or density velocities according to the subpopulation it belongs to.

8. Both offspring and its parent will survive to the next generation.
9. Population declining strategy performs the same task as described in DMOEA.
10. Objective compression strategy performs same as described in DMOEA.

From the procedures of Particle Swarm Multiobjective Optimization with Dynamic population size (DPSMO), we can see:

1. As final result will be a set of solutions instead of a single solution, the geography restriction described as step 3 or 5 has to be applied to assign an individual a global best target to follow. Otherwise, any non-dominated individuals may affect an individual's new location at each generation, thus we may see all the individuals jump around and converge slowly.
2. To obtain optimal solutions with uniformly distribution, the population density value needs to be preserved as well as the minimization of population rank value. Therefore, each individual has two types of velocities, rank velocity and density velocity, which will guarantee both Pareto optimality and uniform distribution of the final results will be achieved.
3. Dynamic population strategy is applied. For an individual, Equations (9.1) and (9.2) update its velocities and locations on each dimension of the decision space. Indeed, this action implies a crossover operation among an individual, its local best and its global best. The newly updated individual can be considered as an offspring. For this reason, "population growing strategy" in DMOEA is not

applied in DPSMO since an individual is supposed to know "where to go" before it moves in particle swarm. Moreover, instead of applying "population growing strategy", using simple offspring updating method based on Equation (9,1) and (9.2) will save significant running time spent in DMOEA on evaluating an offspring's fitness value and comparing with its parents.

4. As there may be more than one particles affect an individual's moving speed and direction, and most importantly, there are two types of velocities associated with each individual, the "cloning" method in step 7 implements an elitism scheme to keep the newly explored better-fitted individuals. This method is crucial for DPSMO because it guarantees the population converges to the correct direction.

5. Ill-fitted individuals will be removed based on "population declining strategy", thus the population size can be controlled and the population quality will be increased.

9.3 Simulation Study on DPSMO

To validate proposed DPSMO, we selected Function ***F3*** as the benchmark problem in the simulation. Equation (6.6) and Figure 6.13 show the mathematic formula and true Pareto front of this problem respectively. For a fair comparison, the initial population, the age threshold, the population size per unit volume, ppv and the cell scales K_1 and K_2 are selected as 2, 10, 3, 50 and 50, respectively, which are same with those for DMOEA in Subsection 7.7.1. Both algorithms run 50 times and the stopping generation is set to be 2,000. Figure 9.1 shows the true Pareto front, resulting Pareto front by DMOEA and DPSMO. The evolutionary trajectories of the population size and

average individual rank, density and distance values over 50 runs by DMOEA and DPSMO are shown in Figures 9.2 (a) – (d), respectively.

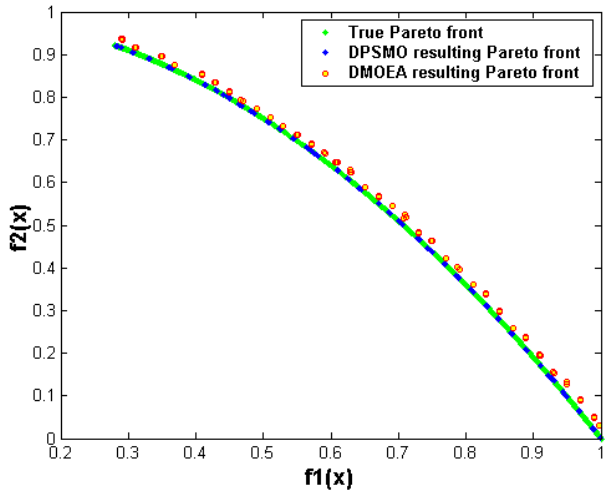
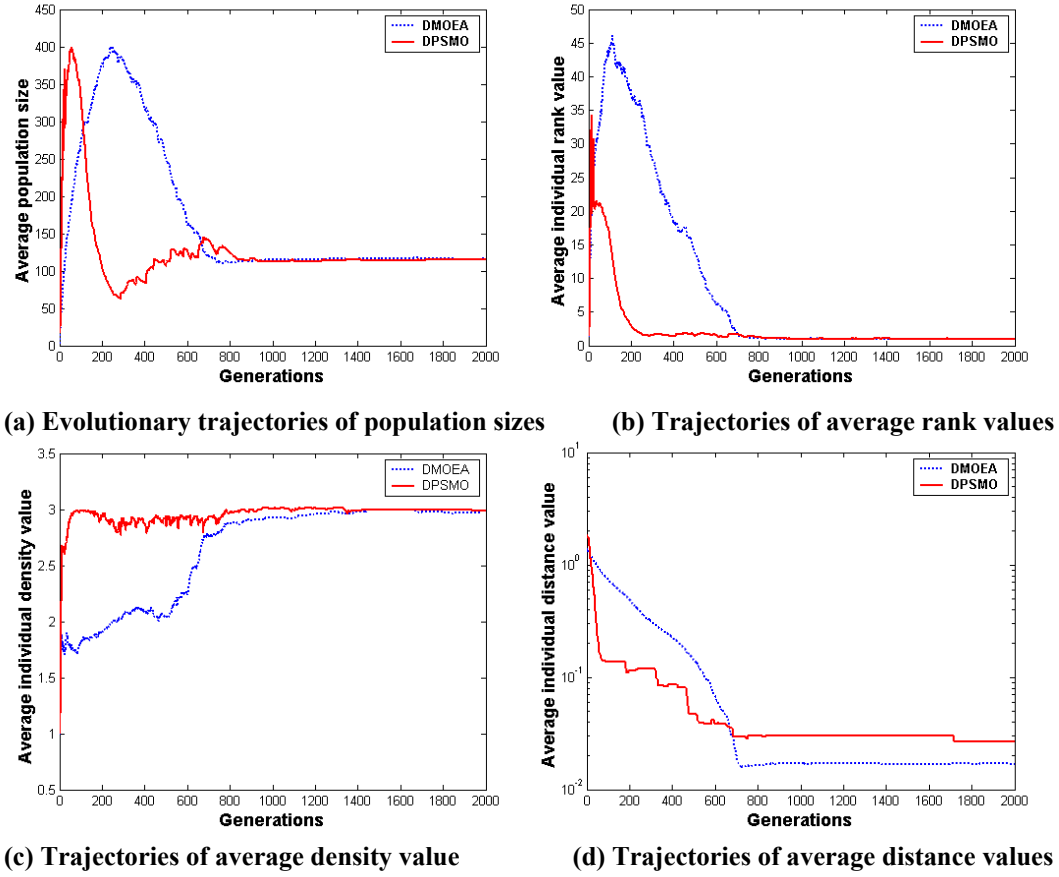


Figure 9.1 Resulting Pareto fronts by DMOEA and DPSMO on Function *F3*

Form Figure 9.1, apparently, there are many final solutions resulting from DPSMO are dominated by those from DMOEA. This result can also be verified by Figure 9.2(d), which shows the final Pareto front of DMOEA is closer to the true Pareto front than that of DPSMO. This effect can partly explained by the intrinsic characteristics of Function ***F3***'s local and global optimality—when the resulting Pareto front is getting closer to the true (global) Pareto front, both algorithms have more difficulty to yield better-fitted offspring. Moreover, for DPSMO, since only the global best individuals and local best individuals can provide the moving information to the entire population, it is more possible for DPSMO to stuck on a middle stage if all the current individuals are Pareto optimal and there is no even better-fitted g_{best} is generated. This problem will hold DPSMO from locating true Pareto front, especially when the given MOP has more than one local Pareto fronts.



(a) Evolutionary trajectories of population sizes **(b) Trajectories of average rank values**

(c) Trajectories of average density value **(d) Trajectories of average distance values**

Figure 9.2 Evolutionary trajectories for the population size and the values of three indicators resulting by DMOEA and DPSMO on Function *F3*

Comparing to DPSMO, DMOEA does not have this problem because evolutionary algorithm applies a population-based information sharing mechanism. A better-fitted offspring can be generated by a crossover operation between any two individuals, no matter how good these parents are. However, from Figure 9.2 (a) – (c), we can also see the advantage of DPSMO since it produces much faster convergence speed. In DPSMO, each particle knows its moving direction and how fast it should go if there exists another individual with better performance. Therefore, before it is trapped by a local Pareto front, the probability that an individual generates a better-fitted offspring by DPSMO is much higher than that of DMOEA. This characteristic will result in both less

evaluation time and less generation numbers, which are the major reasons that DPSMO is almost much faster than DMOEA on Function ***F3*** in terms of converging entire population to a uniformly distributed Pareto front.

9.4 Dynamic Particle Swarm Evolutionary Algorithm (DPSEA)

Since both DMOEA and DPSMO have significant benefit and drawback, we can integrate particle swarm and evolutionary algorithm together in order to take advantages of both algorithms and improve the quality of the evolved solutions. In one aspect, evolutionary algorithm can help each individual share its information with any other individuals instead of only focusing on the best individuals. On the other hands, particle swarm can inform an individual which direction will be the best way to go and how fast its velocities should be. Therefore, inspired by both algorithms, a Dynamic Particle Swarm Evolutionary Algorithm (DPSEA) is designed to improve *efficiency* and *efficacy* of evolutionary process.

The main skeleton of DPSEA is constructed based on DPSMO. Nevertheless, in addition to the location updating strategy of particle swarm, the individuals will perform crossover operation as well. At each generation, an offspring may be generated through two mechanisms—updating the location of a cloned individual or performing crossover between two selected parents. Population growing strategy will be used to determine if an offspring generated through crossover can survive to the next generation and population declining strategy is applied to remove an existing ill-fitted individual. Therefore, comparing to DPSMO, the only change in DPSEA is adding a crossover operation and a

population growing strategy borrowed from DMOEA in both of rank and density subpopulations. By adding these two operations, the running interval for each generation may increase comparing to DMOEA and DPSMO because there are two information-sharing actions performed in DPSEA. However, this sacrifice will be worthy if these two actions can prompt each other and find more valuable individuals than using only one information sharing action.

9.5 Comparison Study on DMOEA, DPSMO and DPSEA

To compare the performance of DPSEA with DMOEA and DPSMO, two benchmark problems— Function ***F3*** and ***F6*** are tested. For Function ***F3***, the initial population, the age threshold, the population size per unit volume, ppv, the cell scales K_1 and K_2 and stopping generations are selected as 2, 10, 3, 50, 50 and 2,000, respectively. For Function ***F6***, the initial population, the age threshold, the population size per unit volume, ppv, the cell scales K_1, K_2 and K_3 and stopping generations are selected as 2, 10, 3, 20, 20, 20 and 10,000, respectively. For each test function, final Pareto front, trajectories of population size, average rank, density and distance values, Box plots of final rank, density and distance values and *C* values resulting from all three algorithms are illustrated.

9.5.1 Simulation on Function F3

Figure 9.3 shows the zoomed sample of the true Pareto front and the resulting Pareto fronts by DMOEA, DPMO and DPSEA. . Figures 9.4(a) – (c) show the Box plots for the final rank, density and distance indicators over 50 runs, respectively. The

performance measures of $C(X_i, X_j)$ for the comparison sets between algorithms i and j are shown in Figure 9.5, where algorithms 1 – 3 represent DPSMO, DMOEA and DPSEA, respectively. Moreover, the evolutionary trajectories of the population size and average individual rank, density and distance values over 50 runs by three algorithms are shown in Figures 9.6 (a) – (d), respectively.

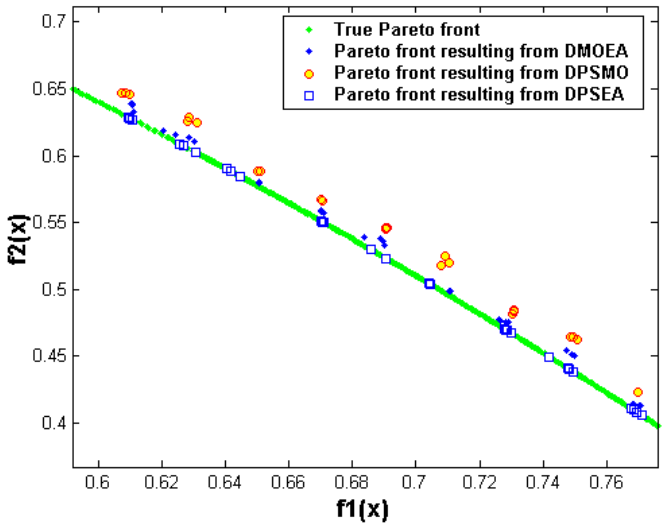
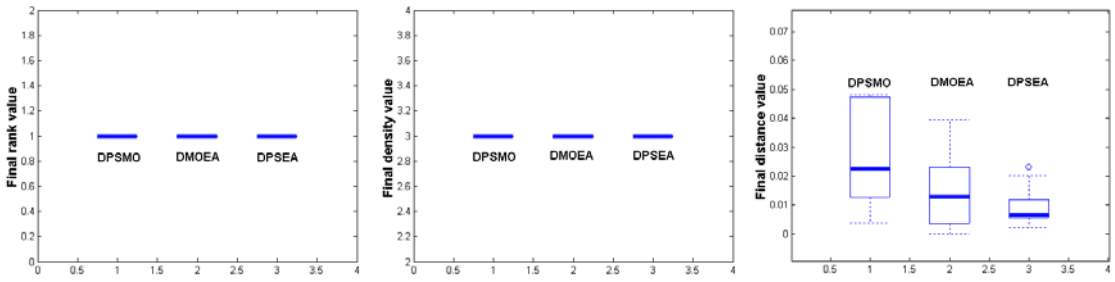


Figure 9.3 Resulting Pareto fronts by DMOEA, DPSMO and DPSEA on Function *F3*



(a) Final rank value **(b) Final density value** **(c) Final distance value**

Figure 9.4 Box plots of three indicators on Function *F3*

From Figures 9.3 – 9.6, we can see that all three algorithms have the capability to converge to a Pareto front with rank value and density value equal to 1 and 3, respectively. However, from Figures 9.3 and 9.4(c), it is obvious that DPSEA's resulting Pareto front is closer to the true Pareto front than those produced by the other two

algorithms. In addition, Figure 9.5 (c) shows that about 70% and 45% of final populations resulting from DPSMO and DMOEA are covered by DPSEA and 0% and 10% of population resulting from DPSEA are covered by DPSMO and DMOEA, respectively. This result proves that DPSEA produce better Pareto fronts than the other two algorithms in terms of finding *near-optimal*, *near-complete* and *uniformly distributed* Pareto front.

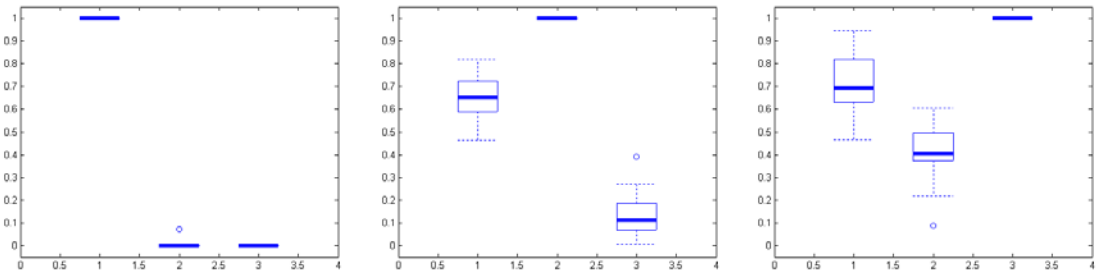
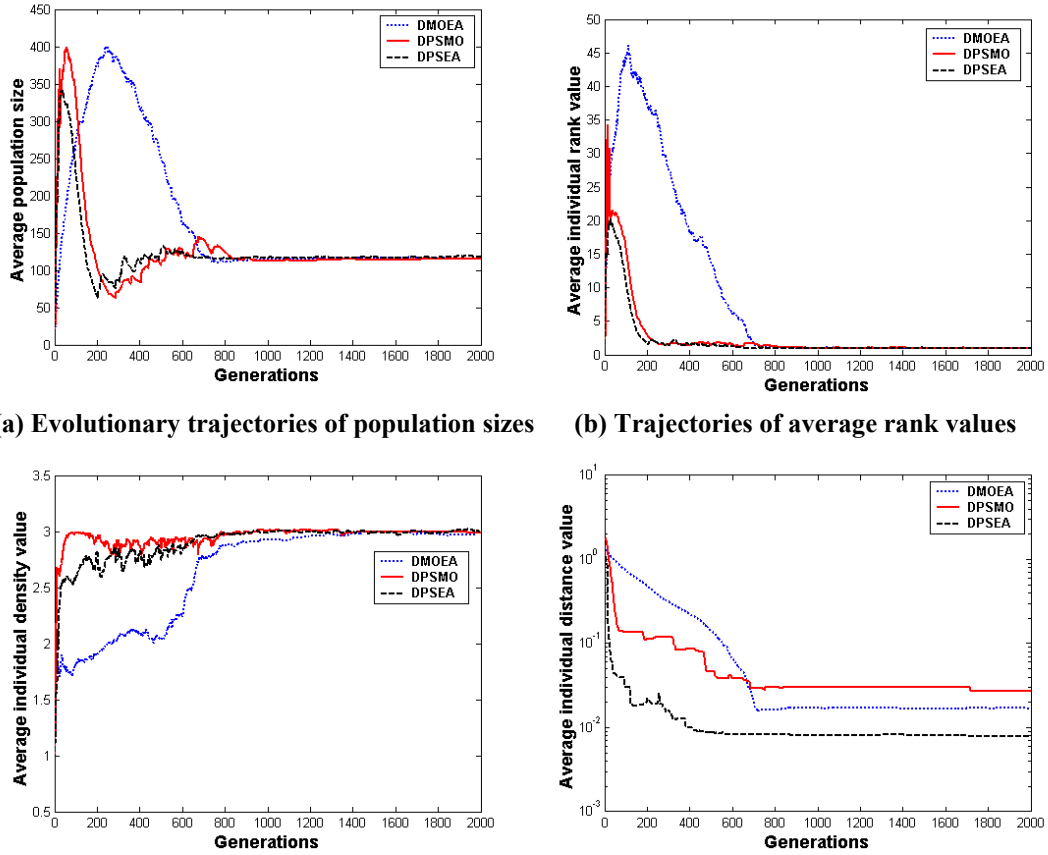


Figure 9.5 Box plots using C measure on Function *F3*

Form Figures 9.6 (a) – (d), it is observed that DPESA is even faster than DPSMO in terms of generation numbers to converge. This phenomenon shows that two information-sharing techniques can promote each other and help entire population converges relatively faster than any one of them. When both of the techniques assist evolutionary process, it will be much easier for an individual to find a better-fitted offspring. These newly generated offspring will keep approaching true Pareto front and push previously non-dominated individual into a dominated one, which will be eliminated by population declining strategy. This mechanism explains why DPSEA produces lowest distance value within smallest number of generations as shown in Figure 9.6(d).



(a) Evolutionary trajectories of population sizes **(b) Trajectories of average rank values**

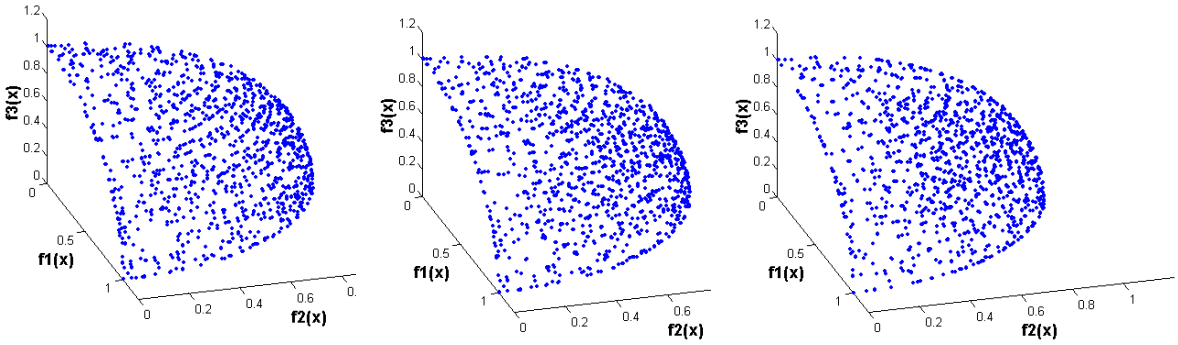
(c) Trajectories of average density value **(d) Trajectories of average distance values**

Figure 9.6 Evolutionary trajectories for the population size and the values of three indicators resulting by DMOEA and DPSMO and DPSEA on Function *F3*

9.5.2 Simulation on Function F6

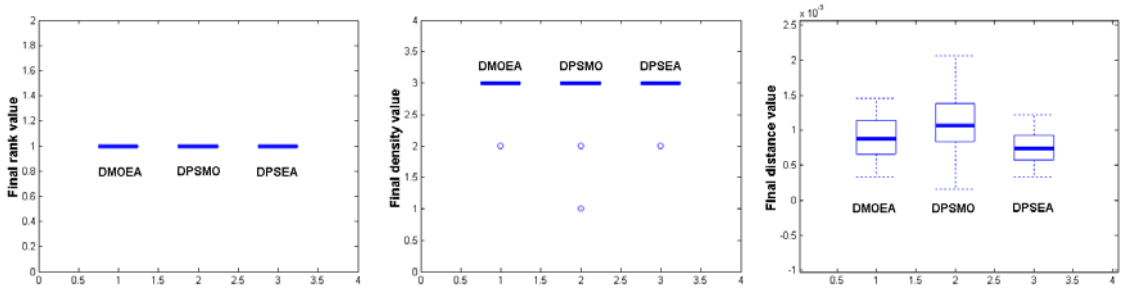
The mathematical formula and true Pareto front for Function ***F6*** are given in Equation (7.14) and Figure 7.19. The first quadrant of a unit sphere is exactly the true Pareto front. Figure 9.7(a) – (c) shows the resulting Pareto fronts by DMOEA, DPMO and DPSEA, respectively. Figures 9.8(a) – (c) show the Box plots for the final rank, density and distance indicators over 50 runs, respectively. The performance measures of $C(X_i, X_j)$ for the comparison sets between algorithms i and j are shown in Figure 9.9, where algorithms 1 – 3 represent DPSMO, DMOEA and DPSEA, respectively. Moreover, the evolutionary trajectories of the population size and average individual

rank, density and distance values over 50 runs by three algorithms are shown in Figures 9.10(a) – (d), respectively.



(a) DMOEA **(b) DPSMO** **(c) DPSEA**

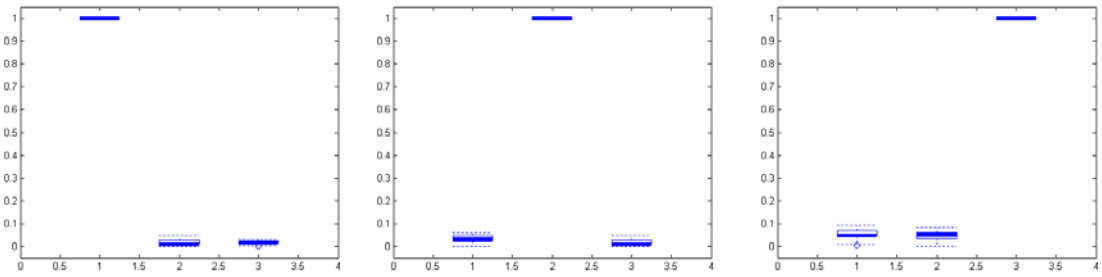
Figure 9.7 Resulting Pareto fronts from DMOEA, DPSMO and DPSEA on Function *F6*



(a) Final rank value **(b) Final density value** **(c) Final distance value**

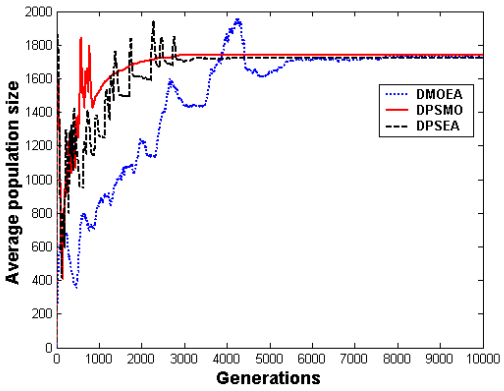
Figure 9.8 Box plots of three indicators on Function *F6*

From Figures 9.7(a) – (c), we can see that all three algorithms result in completive Pareto fronts from their appearances. Meanwhile, from Figures 9.8 (a) – (c) and 9.9 (a) – (c), we cannot find significant differences from the indicators of final results from all three algorithms as well. Since Function ***F6*** does not generate any local Pareto front, there will be no hindrance for DPSMO to locate true Pareto front. However, by applying two information-sharing techniques, DPSEA still shows its ability to approximate more accurate Pareto front than the other two algorithms as shown in Figure 9.8(c) and 9.10(d).

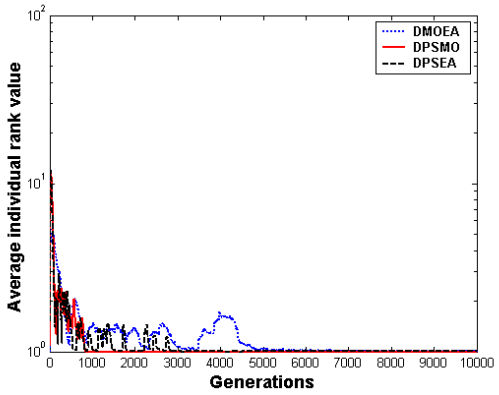


$C(X_1, X_{1-3})$ $C(X_2, X_{1-3})$ $C(X_3, X_{1-3})$

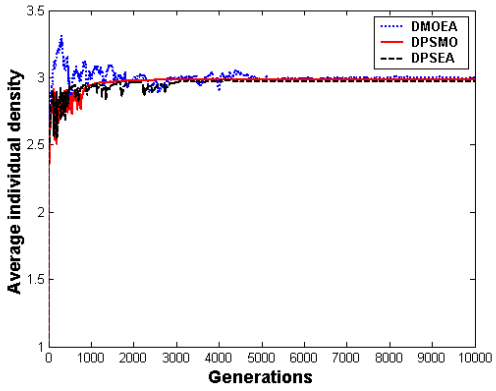
Figure 9.9 Box plots using C measure on Function *F6*



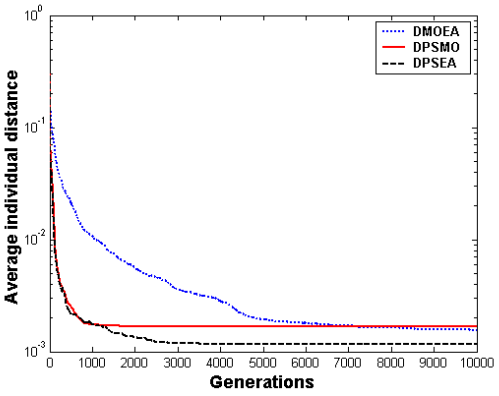
(a) Evolutionary trajectories of population sizes



(b) Trajectories of average rank values



(c) Trajectories of average density value



(d) Trajectories of average distance values

Figure 9.10 Evolutionary trajectories for the population size and the values of three indicators resulting by DMOEA and DPSMO and DPSEA on Function *F6*

However, by examining the evolutionary trajectories as shown in Figures 9.10 (a) – (d), we can see remarkable difference among three algorithms in terms of convergence

property. It only takes DPSMO less than 1,000 generations to converge and DMOEA needs about 7,000 and 35,00 generations, respectively. Table 9.1 shows the average running time per generation for each of three algorithms. The "CPUTIME" command from MATLAB (version 6.1) is used to measure the time elapsed for each algorithm implemented in MATLAB and a HP computer with dual 2-GHz processors and 1-GByte RAM is used for simulation. From Table 9.1, we can see that DPSMO runs than DMOEA and DPSEA faster at each generation. This is contributed by DPSMO's population growing method, which does not evaluate newly generated offspring and filter the incompetence ones as described in Subscetion 9.2. DPSEA is a bit slower than DMOEA since it applies two information-sharing techniques at each generation. However, because the most time consuming parts are population declining strategy and objective compression strategy, which are used by all three algorithms, the difference of time consuming per generation for these algorithms is not remarkable. Therefore, from the above observations, it is clear that DPSMO has the fastest convergence speed since it spends least time on each generation and takes smallest number of generations to converge. Although DPSEA will spend a bit longer time on each generation than DMOEA, the total time consuming of convergence for DPSEA is still significantly shorter than DMOEA since DPSEA takes much smaller generations to converge. In addition, we need to keep in mind that DPSEA will produce more accurately approximated Pareto front than the other two algorithms in terms of distance values and DPSMO may generate less competitive Pareto front, especially there are local Pareto fronts exist for the given MOP. Nevertheless, combining particle swarm optimization

with evolutionary algorithms provides a potential way to design an MOEA in solving real world MOPs that need fast processing time to generate qualified Pareto fronts.

Table 9.1 Comparison results of computation time of *F6* from DMOEA, DPSMO and DPSEA

	DMOEA	DPSMO	DPSEA
Time (sec)	**0.18**	**0.15**	**0.20**

X. CONCLUSIONS AND FUTURE WORKS

Although the conventional algorithms, such as linear weighting, goal programming and min-max optimization are still widely used to solve MOPs, multiobjective evolutionary algorithms have drawn growing attentions from more and more researchers in that they are designed to deal simultaneously with a set of candidate solutions. This characteristic allows MOEAs to find an entire set of Pareto optimal solutions in a single run of the algorithms, instead of having to perform a series of separate runs as in the cases of the conventional mathematical programming techniques. In addition, evolutionary algorithms are less susceptible to the concavity, discontinuity and local optimality of the Pareto front, whereas these issues are critical concerns for those conventional approaches.

According to the No Free Lunch (NFL) theorem [51], no formal assurances of an algorithm's general effectiveness exists if insufficient knowledge of the problem domain is incorporated into the algorithm domain. Therefore, some of the studies on the MOP test suite are included in this research and seven benchmark MOP test functions are examined by some state-of-the-art MOEAs (i.e. NSGA-II, SPEA II). From the comparison and analysis of the simulation results, although some of the difficulties cannot be thoroughly addressed by these MOEAs, it is clear that three techniques are the crucial building blocks in a successful MOEA design procedure. These techniques include: a Pareto ranking scheme, a density estimation and preservation method and an elitism scheme. A Pareto ranking scheme helps the initial population converges to a

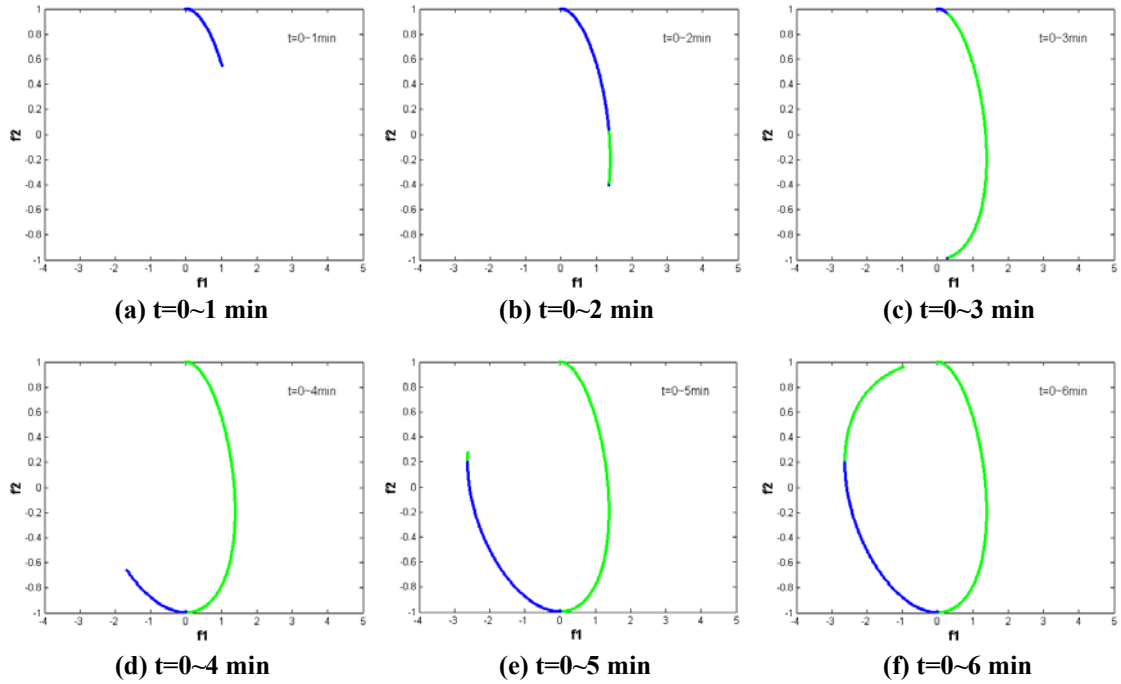
Pareto front at the final generation, a density estimation and preservation method can prevent the emergence of the "too crowded" areas, and an elitism scheme stores those non-dominated individuals to avoid losing any the Pareto points generated throughout the entire evolutionary process. By synergistically integrating these techniques and other schemes (i.e., mating restriction, forbidden region), a Rank-Density based Genetic Algorithm (RDGA) [54] is designed and investigated by the given MOP test suite. By examining the selected performance indicators, RDGA is found to be competitive with, or even superior to, the other advanced MOEAs in terms of keeping the diversity of the individuals along the trade-off surface, tending to extend the Pareto front to new areas, and finding a well-approximated Pareto optimal front. Moreover, RDGA is manipulated by using a hierarchical gene representation to solve a real multiobjective optimization problem—a radial basis neural network design problem.

Although RDGA shows its capability in coping with several types of challenging MOPs, it still cannot tackle the confliction between avoiding and exploiting "genetic drift" phenomenon. In fact, if an MOEA has fixed population size, it will be difficult, if not impossible, to solve this problem since the limited computation resource cannot be congregated and homogeneously distributed simultaneously. Therefore, based on the principal ideas of RDGA, a Dynamic Muleiobjective Evolutionary Algorithm (DMOEA) [84] is proposed in this research. In DMOEA, in one aspect, an offspring will be added into the population if its fitness value is higher than one of its parents while the corresponding parent is still maintained. This intention constructs a pure population growing strategy in order to excite the population covering those unexplored areas. On

the other hand, three kinds of probabilities of "eliminating" individuals are computed based on the dynamic situations of the individuals' rank and density values. By adaptively removing those incompetent individuals in terms of their rank and density values, DMOEA can control the population size within a reasonable number. In addition, the cell-based rank and density calculation technique and objective compression strategy offers DMOEA less computation effort on fitness evaluation even a large population size if involved. From the experiment result, DMOEA can effectively exploit an optimal population size by locating all the trade-off hyper-areas and approximate a *near-optimal*, *near-complete* Pareto front. Meanwhile, DMOEA shows its potential in solving complicated MOPs with different characteristics (i.e., local optimality, non-uniformly distributed and high dimensional decision and objective spaces).

Based on the extensive study of MOEAs, a module-based EMO Toolbox is designed on MATLAB platform. This toolbox most of the advanced MOEAs (i.e. NSGA-II, SPEA II, RDGA, IMOEA, PAES and DMOEA) are provided to the users. Moreover, as an MOEA can be considered as a hybrid of several key techniques (ranking scheme, density estimation, elitism, etc.) and for each technique, there are several variations exist in literature, this toolbox provides users a free model design function. A designer or user can have more flexibility in choosing their favorite method for each building block; and the population growing and declining strategy can help the resulting algorithms produce a near-optimal and near-complete Pareto front with optimal number of individuals. The user-friendly visualization Graphical User Interface (GUI), Data Analysis GUI and on-line Help link and Demonstrations also help users effectively

design a MOEA and efficiently apply it to solve real world multiobjective optimization problems.



(a) t=0~1 min **(b) t=0~2 min** **(c) t=0~3 min**

(d) t=0~4 min **(e) t=0~5 min** **(f) t=0~6 min**

Figure 10.1 An example of MOP with time varying objective function and Pareto front

Although evolutionary algorithms have been successfully applied in solving many multiobjective optimization problems, it is also worthy to note that MOEA is not an efficient approach in dealing with MOPs with time-varying decision variables, objective functions and Pareto fronts. For example, Equation (10.1) represents an MOP with such characteristics.

Minimize $f_1(x)$ and $f_2(x)$, where

$$f_1(x) = x_1(t) \times x_2(t), \quad (10.1)$$
$$f_2(x) = x_3(t),$$
$$x_1(t) = \sin(100\ t),$$
$$x_2(t) = e^{20t},$$
$$x_1(t) = \cos(100\ t)$$

subject to $0 \le t \le 6\,\text{min}$

The objective space and Pareto fronts for different time interval are illustrated in Figures 10.1(a) – (f). It is apparent that MOEAs will not satisfy the time constraint and response fast enough to cope with this type of problem. Therefore, a Dynamic Particle Multiobjective Optimization (DPSMO) algorithm is designed by combining Particle Swarm Optimization (PSO) technique with dynamic population strategy. According to PSO, as an individual will know where to fly and how fast its speed should be, it can quickly move to an optimal position based on its historical trajectories and the knowledge of the location of the best individual in the swarm. However, as PSO only performs a point-centered, one-way information sharing mechanism, it may have difficulty in approximating true Pareto fronts on MOPs with local Pareto optimality. For this reason, a hybrid Dynamic Particle Swarm Evolutionary Algorithm (DPSEA) is devised to take advantages of PSO's fast convergence characteristic and EA's population-based information sharing capability. From the simulation results, DPSMO dramatically improves the convergence speed comparing to DMOEA and DPSEA produces better Pareto fronts than DMOEA and DPSMO. Although DPSMO and DPSEA provide a novel solution for MOEA in dealing with MOPs that need fast convergence speed, further study and investigation are still needed to test the abilities of these two algorithms and improve their performances.

Some other interesting issues may also be studied in future work. These issues include: convergence characteristics of MOPs, dynamic or noisy fitness evaluation in MOEA, on-line and real time MOEAs, mathematical representation of true Pareto front and the existence and uniqueness quantification of Pareto front. In summary, these issues

can be categorized into three types: theoretical study, algorithm development and the investigation of the real applications. Especially, a suitable MOP in real world environment needs to be developed and studied to examine all kinds of state-of-the-art multiobjective evolutionary algorithms.

BIBLIOGRAPHY

[1] H. Eschenauer, J. Koski and A. Osyczka, *Multicriteria Design Optimization* : *Procedures and Applications*, Springer-Verlag, New York. 1986.

[2] C. M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evol. Comput.*, vol. 3, pp. 1-16, 1995.

[3] A. Charnes, and W. Cooper, *Management Models and Industrial Applications of Linear Programming*, John Wiley & Sons, New York. 1961.

[4] S. Russell and P. Norvig, *Artificial Intelligence*: *A Modern Approach*. Prentice Hall, Upper Saddle River, NJ. 1995.

[5] H. Schwefel, *Evolution and Optimum Seeking*, John Wiley & Sons, New York. 1995.

[6] K. Iwata, Y. Murotsu, T. Iwatsubo, and S. Fuji, "A probabilistic approach to the determination of the optimum cutting condition," *ASME*, *Journal of Engineering for* Industry, vol. 4, pp. 1099-1107. 1972

[7] S. Fenster and A. Ugural, *Advanced Strength and Applied Elasticity*, Elsevier, New York. 1987.

[8] Y. Ijiri, *Management Goals and Accounting for Control*, North Holland, Amsterdam. 1965.

[9] R. Philipson, and A. Ravindran, "Application of goal programming to machinability data optimization," *ASME*, *Journal of Mechanical Design*, vol. 3, pp. 286-291. 1978.

[10] A. Osyczka, *Multicriterion Optimization in Engineering with Fortran Programs*. John Wiley & Sons, New York. 1984.

[11] J. Holland, *Adaption in Natural and Artificial Systems*, MIT press, 2nd Ed., Cambridge, 1992.

[12] E. Zitzler, M. Laumanns and L. Thiele, *SPEA2: Improving the Strength Pareto Evolutionary Algorithm*, Technical Report TIK-Report 103, Swiss Federal Institute of Technology, 2001.

[13] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, pp. 257-271, 1999.

[14] K. Deb, S. Agrawal, A. Pratap and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *Proc. Parallel Problem Solving from Nature—PPSN VI*, pp. 849-858, 2000.

[15] D. A. Van Veldhuizen, Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. PhD thesis, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, May 1999.

[16] G. Brassard and P. Brately, *Algorithms*: *Theory and Practice*, Prentice Hall, NJ, 1988.

[17] P. Husbands, "Genetic algorithms in optimization and adaptation," *Advances in Parallel Algorithms*, pp. 227-276, 1992.

[18] J. Pearl, *Heuristics*, Addison-Wesley, MA, 1989.

[19] D. E. Goldberg, *From Genetic and Evolutionary Optimization to the Design of Conceptual Machines*, Technical Report 98008, University of Illinois at Urbana-Champaign, 1998.

[20] R. Neapolitan and N. Kumarss, *Foundation of Algorithms*, D. C. Heath and Company, MA, 1996.

[21] G. B. Lamont (ed.), *Compendium of Parallel Parograms for the Intel iPSC Computers*. Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson AFB, 1993.

[22] A. Vicini and D. Quagliarella, Multipoint Transonic Airfoil Design by Means of a Multiobjective Genetic Algorithm, Technical Report AIAA-97-0082, Washington, D.C., AIAA, 1997.

[23] L. Eshelman, (ed.). *Proc. Of the Sixthe Int'l Conf. On Genetic Algorithms.* Morgan Kaufmann, San Mateo, Cambridge, 1995.

[24] J. McDonnell, R. Reynolds and D. Fogel (ed.). *Evolutionary Programming IV: Proceedings of the Fourth Annual Conference on Evolutionary Programming.* MIT Press, 1995.

[25] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, 1989.

[26] L. Fogel, A. Owens and M. Walsh, *Artificial Intelligence Through Simulated Evolution.* John Wiely & Sons, New York, 1996.

[27] D. Fogel, *System Identification Through Simulated Evolution: A Machine Learning Approach to Modeling*, Ginn Press, Needham Heights, 1991.

[28] H. Schwefel, *Numerical Optimization of Computer Models*, John Wiley & Sons, Chichester, 1981.

[29] J. Holland, *Adaption in Natural and Artificial Systems.* MIT Press, 1st Ed., Cambridge, 1975.

[30] L. Davis, *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, Amsterdam, 1991.

[31] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Program.* 2nd Ed., Springer-Verlag, New York, 1993.

[32] D. Beasley, D. Bull and R. Martin, "An overview of genetic algorithms: Part 1, Fundamentals," *University Computing,* vol. 2, pp. 58-69, 1993.

[33] C. Janikow and Z. Michalewicz, "An experimental comparison of binary and floating point representations in genetic algorithms," in *Proc. 4th Int. Conf. Genetic Algorithms,* pp. 31-36, 1991.

[34] A. Gillies, *Machine Learning Procedures for Generating Image Domain Feature Detectors.* PhD thesis, University of Michigan, Ann Arbor, 1985.

[35] G. Baker, "Adaptive selection methods for genetic algorithms", *Proceedings of the 2nd Int. Conf. Genetic Algorithms*, pp. 14-21, 1987.

[36] K. De Jong, *The Analysis and Behavior of A Class of genetic Adaptive Systems.* PhD thesis, University of Michigan, Ann Arbor, 1985.

[37] G. Syswerda, "Uniform crossover in genetic algorithms," in *Proc 3rd Int. Conf. Genetic Algorithms,* pp. 2-9, 1989.

[38] L. Davis, "Job shop scheduling with genetic algorithms," in *Proc 1st Int. Conf. Genetic Algorithms,* pp. 136-140, 1985.

[39] G. Syswerda, "Schedule optimization using genetic algorithms," *Handbook of Genetic Algorithms,* pp. 2-9, 1989.

[40] J. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proc. 1st Int. Conf. Genetic Algorithms*, pp. 93-100, 1985.

[41] Fourman, M., "Compaction of symbolic layout using genetic algorithms," in *Proc. 1st Int. Conf. Genetic Algorithms*, pp. 141-153, 1985.

[42] J. Horn, N. Nafpliotis and D. E. Goldberg, "A niched pareto genetic algorithm for multiobjetcive optimization," in *Proc. 1st IEEE Cong. Evolutionary Computation,* pp. 82-87, 1994.

[43] N. Srinivas and K. Deb, "Multi-Objective function optimization using non-dominated sorting genetic algorithms," *Evol. Comput.*, vol. 2, pp. 221-248, 1994.

[44] C. Fonseca and P. Fleming, "Genetic algorithms for multiobjective optimization: formulation, discussion and generalization", in *Proc. 5th Int. Conf. Genetic Algorithms*, pp. 416-423, 1993.

[45] S. W. Mahfoud, "Genetic drift in sharing methods," in *Proc. 1st IEEE Cong. Evolutionary Computation,* vol. 1, pp. 67-72, 1994.

[46] G. W. Greenwood, X. S. Hu and J. G. D'Ambrosio, "Fitness functions for multiple objective optimization problems: Combining preferences with Pareto rankings," in *Foundations of Genetic Algorithms 4* (*FOGA96*), pp. 437-455, 1996.

[47] D. S. Todd and P. Sen, "A multiple criteria genetic algorithm for containership loading," in *Proc. 7th Int. Conf. Genetic Algorithms*, pp. 674-681, 1997.

[48] A. G. Cunha, P. Olivera and J. Covas, "Use of genetic algorithms in multicriteria optimization to solve industrial problems," in *Proc. 7th Int. Conf. Genetic Algorithms*, pp. 682-688, 1997.

[49] H. Meunier, E. G. Talbi and P. Reininger, "A multiobjective genetic algorithm for radio network optimization," in *Proc. 7th IEEE Cong. Evolutionary Computation,* pp. 317-324, 2000.

[50] D. H. Loughlin and S. Ranjithan, "The neighborhood constraint-method: A genetic algorithm-based multiobjective optimization technique," in *Proc. 7th Int. Conf. Genetic Algorithms*, pp. 666-673, 1997.

[51] D. H. Wolpert and W. G. Macready, "No free lunch theorem for optimization," *IEEE Trans. Evol. Comput.*, vol.1, pp. 67-82, 1997.

[52] J. D. Knowles and D. W. Corne, "Approximating the non-dominated front using the Pareto archived evolutionary strategy," *Evol. Comput.*, vol. 8, pp.149-172, 2000.

[53] T. Krink and R. K. Ursem, "Parameter control using agent based patchwork model," in *Proc. 7th IEEE Cong. Evolutionary Computation,* pp. 77-83, 2000.

[54] H. Lu and G. G. Yen, "Rank-density based multiobjective genetic algorithm," in *Proc. 9th IEEE Cong. Evolutionary Computation,* pp.944-949, 2002.

[55] H. Lu and G. G. Yen, "Multiobjective optimization design via genetic algorithm," in *Proc. 2001 IEEE Conf. Control Applications,* pp.1190-1195, 2001.

[56] C. M. Fonseca and P. J. Fleming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms—part I: A unified formulation," *IEEE Trans. System, Man, and Cybernetics*, vol. 28, pp. 26-37, Jan. 1998.

[57] C. C. H. Borges and H. J. C. Barbosa, "A non-generational genetic algorithm for multiobjective optimization," in *Proc. 7th IEEE Cong. Evolutionary Computation,* pp. 172-179, 2000.

[58] M. Laumanns, E. Zitzler, and L. Thiele, " A unified model for multi-objective evolutionary algorithms with elitism," in *Proc. 7th IEEE Cong. Evolutionary Computation,* pp. 46-53, 2000.

[59] Z. Michalewicz, "Genetic algorithms, numerical optimization, and constraints," in *Proc. 6th Int. Conf. Genetic Algorithms*, pp. 151-158, 1995.

[60] D. A. Van Veldhuizen and G. B. Lamont, "On measuring multiobjective evolutionary algorithm performance," in *Proc. 7th IEEE Cong. Evolutionary Computation,* pp. 204-211, 2000.

[61] M. Tanaka, "GA-based decision support system for multicriteria optimization," in *Proc. Int. Conf. Systems, Man, and Cybernetics*, pp. 1556-1561*,* 1995.

[62] V. R. Vemuri and W. Cedeño, "A new genetic algorithm for multiobjective optimization in water resource management," in *Proc. 2nd IEEE Cong. Evolutionary Computation*, pp. 495-500, 1995.

[63] N. Murata, S. Yoshizawa and S. Amari, "Network information criterion—determining the number of hidden units for an artificial neural network model," *IEEE Trans. Neural Networks*, vol. 5, pp. 865-872, 1994.

[64] X. M. Gao, S. J. Ovaska and Z. O. Hartimo, "Speech signal restoration using an optimal neural network structure," in *Proc. IEEE Int. Conf. Neural Networks*, pp. 1841-1846, 1996.

[65] X. Yao, "Evolving artificial neural network," *International Journal of Neural Systems,* vol. 4, pp. 203-222, 1993.

[66] A. Doering, M. Galicki and H. Witte, "Structure optimization of neural networks with the A*-Algorithm," *IEEE Trans. Neural Networks*, vol. 8, pp. 307-317, 1997.

[67] S. Geman, E. Bienenstock and R. Dousat, "Neural networks and the bias/variance dilemma," *Neural Comput.*, vol. 2, pp. 303-314, 1989.

[68] B. Zhang and D. Cho, "Evolving neural trees for time series prediction using Bayesian evolutionary algorithms," in *Proc. 1st IEEE Symp. Combination of Evolutionary Computation and Neural Networks*, pp. 17-23, 2000.

[69] G. G. Yen and H. Lu, "Hierarchical genetic algorithm based feed-forward neural network design," *International Journal of Neural Systems*, pp. 31-45, 2002.

[70] T. Y. Ke, K. S. Tang, K. F. Man and P. C. Luk, "Hierarchical genetic fuzzy controller for a solar power plant," in *Proc. IEEE Int. Symp. Industrial Electronics*, pp. 584-588, 1998.

[71] G. G. Yen and H. Lu, "Hierarchical rank density genetic algorithm for radial-basis function design," in *Proc. 9th IEEE Cong. Evolutionary Computation,* pp.25-30, 2002.

[72] T. Kaylani and S. Dasgupta, "A new method for initializing radial basis function classifiers," in *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics*, pp. 2584-2587, 1994.

[73] P. D. Wasserman, *Advanced Method in Neural Computing*, New York: Van Nostrand Reinhold, 1993

[74] S. Chen, C. F. Cowan and P. M. Grant, "Orthogonal least square learning algorithm for radial basis function networks," *IEEE Trans. Neural Networks*, vol. 2, pp. 302-309, 1991.

[75] S. Ricardo and B. Amnon, "Evolutionary strategies for a parallel multi-objective genetic algorithm," in *Proc. 9th Int. Conf. Genetic Algorithms*, pp. 227-234, 2000.

[76] J. Arabas, Z. Michalewicz and J. Mulawka, " GAVaPS-A genetic algorithm with varying population size," in *Proc. 1st Cong. Evolutionary Computation,* pp. 73-74, 1994.

[77] N. Zhuang, M. Benten and P. Cheung, "Improved variable ordering of BDDS with novel genetic algorithm," in *Proc. IEEE Symp. Circuits and Systems*, pp. 414-417, 1996.

[78] J. Grefenstte, "Optimization of control parameters for genetic algorithms," *IEEE Trans. Systems, Man and Cybernetics*, vol. 16, pp. 122-128, 1986.

[79] J. Alander, "On optimal population size of genetic algoriths," in *Proc*. *IEEE Conference on Computer Systems and Software Engineering*, pp. 65-70, 1992.

[80] K. Tan, T. Lee and E. Khor, "Incrementing multi-objective evolutionary algorithms: performance studies and comparisons," in *Proc. 1st Evolutionary Multi-Criterion Optimization* (EMO'2001), pp. 111-125, 2001.

[81] K. Tan, T Lee and E. Khor, "Evolutionary algorithms with goal and priority information for multi-objective optimization," in *Proc. 6th IEEE Cong. Evol. Comput.*, pp.106-113, 1999.

[82] K. Deb, "Multiobjective genetic algorithms: problem difficulties and construction of test problems," *Evol. Comput.*, vol. 7, pp. 205-230, 1999.

[83] K. Deb, L. Thiele, M. Laumanns and E. Zitzler, "Scalable Multi-objectvie optimization test problems," in *Proc. 9th IEEE Cong. Evol. Comput.*, pp. 825-830, 2002.

[84] H. Lu and G. G. Yen, "Dynamic population size in multiobjective evolutionary algorithm," in *Proc. 9th IEEE Cong. Evol. Comput.*, pp. 1648-1653, 2002.

[85] K. Tan, T. Lee, D. Khoo and E. Khor, "MOEA toolbox for computer aided multi-objective optimization," *IEEE Trans. Systems, Man and Cybernetics*, vol. 31, pp. 537-556, 2001.

[86] J. Kennedy and R.C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Networks*, pp. 1942-1948, 1995.

[87] J. Kennedy, "The particle swarm optimization: social adaptation of knowledge," in *Proc. 4th IEEE Cong. Evolutionary Computation,* pp. 303-308, 1997.

[88] Y, Shi and R.C. Eberhart, "A modified particle swarm optimizer," in *Proc. 5th IEEE Cong. Evolutionary Computation,* pp. 69-73, 1998.

[89] X, Hu and R.C. Eberhart, "Multiobjective optimization using dynamic neighborhood particle swarm optimization," in *Proc. 9th IEEE Cong. Evol. Comput.*, pp. 1677-1681, 2002.

[90] J. Moore and R. Chapman, *Application of particle swarm to multiobjetcive optimization*, Department of Computer Science and Software Engineering, Auburn University, 1999.

[91] T. Ray, T. Kang and S. K. Chye, "Multiobjective design optimization by evolutionary algorithm," *Engineering Optimization*, 2002 (in Press).

[92] C.A.Coello and M. S. Lechuga, "MOPSO: A proposal for multiple objective particle swarm optimization," in *Proc. 9th IEEE Cong. Evol. Comput.*, pp. 1051-1056, 2002.