MULTI-OBJECTIVE PARTICLE SWARM OPTIMIZATION: ALGORITHMS AND APPLICATIONS

LIU DASHENG
(M.Eng, Tianjin University)

A THESIS SUBMITTED
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING
NATIONAL UNIVERSITY OF SINGAPORE

2008

Summary

Many real-world problems involve the simultaneous optimization of several competing objectives and constraints that are difficult, if not impossible, to solve without the aid of powerful optimization algorithms. What makes multi-objective optimization so challenging is that, in the presence of conflicting specifications, no one solution is optimal to all objectives and optimization algorithms must be capable of finding a number of alternative solutions representing the tradeoffs. However, multi-objectivity is one facet of real-world applications.

Particle swarm optimization (PSO) is a stochastic search method that has been found to be very efficient and effective in solving sophisticated multi-objective problems where conventional optimization tools fail to work well. PSO's advantage can be attributed to its swarm based approach (sampling multiple candidate solutions simultaneously) and high convergence speed. Much work has been done to the development of PSO algorithms in the past decade and it is finding increasingly application to the fields of bioinformatics, power and voltage control, spacecraft design and resource allocation.

A comprehensive treatment on the design and application of multi-objective particle swarm optimization (MOPSO) is provided in this work; and it is organized into seven chapters. The motivation and contribution of this work are presented in Chapter 1. Chapter 2 provides the necessary background information required to appreciate this work, covering key concepts and definitions of multi-objective optimization and particle swarm optimization. It also presents a general framework of MOPSO which illustrates the basic design issues of the state-of-the-arts. In Chapter 3, two mechanisms, fuzzy gbest and synchronous particle local search, are developed to improve MOPSO performance. In Chapter 4, we put forward a competitive and cooperative coevolution model to mimic the interplay of competition and cooperation among different species in nature and combine it with PSO to solve complex multiobjective function optimization problems. The coevolutionary algorithm is further formulated into a distributed MOPSO algorithm to meet the demand for large computational power in Chapter 5. Chapter 6 addresses the issue of solving bin packing problems using multi-objective particle swarm optimization. Unlike existing studies that only consider the issue of minimum bins, a multiobjective two-dimensional mathematical

model for the bin packing problem is formulated in this chapter. And a multi-objective evolutionary particle swarm optimization algorithm that incorporates the concept of Pareto optimality is implemented to evolve a family of solutions along the trade-off surface. Chapter 7 gives the conclusion and directions for future work.

Acknowledgements

First and foremost, I would like to thank my supervisor, Associate Professor Tan Kay Chen for introducing me to the wonderful field of particle swarm optimization and giving me the opportunity to pursue research in this area. His advices have kept my work on course during the past four years. Meanwhile, I am thankful to my co-supervisor, Associate Professor Ho Weng Khuen, for his strong and lasting support. In addition, I wish to acknowledge National University of Singapore (NUS) for the financial support provided throughout my research work.

I am also grateful to my labmates at the Control and Simulation laboratory: Goh Chi Keong for the numerous discussions, Ang Ji Hua Brian and Quek Han Yang for sharing the same many interests, Teoh Eu Jin, Chiam Swee Chiang, Cheong Chun Yew and Tan Chin Hiong for their invaluable services to the research group.

Last but not least, I would like to express cordial gratitude to my parents, Mr. Liu Jiahuang and Ms. Wang Lin. I own them so much for their support to my pursuing higher educational degree. They always back me as I need, especially when I was in difficulty. I would also like to send my special thanks to my wife Liu Yan, for her tenderness and encouragement that accompany me during the tough period of writing this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Optimization may be considered as a decision-making process to get the most out of available resources for the best attainable results. Simple examples include everyday decisions, such as what type of transport to take, which clothes to wear and what groceries to buy. For these routine tasks, the decision to be made can be very simple. For example, most people will choose the cheapest transport. Consider now, the situation where we are running late for a meeting due to some unforseen circumstances. Since the need for expedition is conflicting to the first consideration of minimizing cost, the selection of the right form of transportation is no longer as straightforward as before and the final solution will represent a compromise between the different objectives. This type of problems which involves the simultaneous consideration of multiple objectives are commonly termed as multi-objective (MO) problems.

Many real-world problems naturally involve the simultaneous optimization of several competing objectives. Unfortunately, these problems are characterized by objectives that are much more complex as compared to routine tasks mentioned above and the decision space are often so large that it is often difficult, if not impossible, to be solved without advanced and efficient optimization techniques. This thesis investigates the application of an efficient optimization method, known as Particle Swarm Optimization (PSO), to the field of MO optimization.

1.1 Motivation

Traditional operational research approaches to MO optimization typically entails the transformation of the original problem into a SO problem and employs point-by-point algorithms such as branch-and-bound to iteratively obtain a better solution. Such approaches have several limitations including the generation of only one solution for each simulation run, the requirement of the MO problem to be well-behaved, i.e. differentiability or satisfying the Kuhn-Tucker conditions, and the sensitivity to the shape of the Pareto front. On the other hand, metaheuristical approaches that are inspired by social, biological or physics phenomena such as cultural algorithm (CA), particle swarm optimization (PSO), evolutionary algorithm (EA), artificial immune system (AIS), differential evolution (DE), and simulated annealing (SA) have been gaining increasing acceptance as much more flexible and effective alternatives to complex optimization problems in the recent years. This is certainly a stark contrast to just two decades ago, as Reeves remarked in [156] that an eminent person in operational research circles suggested that using a heuristic was an admission of defeat!

MO optimization is a challenging research topic not only because it involves the simultaneous optimization of several complex objectives in the Pareto optimal sense, it also requires researchers to address many issues that are unique to MO problems, such as fitness assignment [45] [118], diversity preservation [96], balance between exploration and exploitation [10], and elitism [108]. Many different CA, PSO, EA, AIS, DE and SA algorithms for MO optimization have been proposed since the pioneering effort of Schaffer in [168], with the aim of advancing research in above mentioned areas. All these algorithms are different in methodology, particularly in the generation of new candidate solutions. Among these metaheuristics, multi-objective particle swarm optimization (MOPSO), which originates from the simulation of behavior of bird flocks, is one of the most promising stochastic search methodology because of its easy implementation and high convergence speed. MOPSO algorithm intelligently sieve through the large amount of information embedded within each particle representing a candidate solution and exchange information to increase the overall quality

of the particles in the swarm.

This work seeks to explore and improve particle swarm optimization techniques for MO function optimization as well as to expand its applications in real world bin packing problems. We will primarily use an experimental methodology backed up with statistical analysis to achieve the objectives of this work. The effectiveness and efficiency of the proposed algorithms are compared against other state of the art multi-objective algorithms using test cases.

It is hoped that findings obtained by this study would give a better understanding of PSO concept, and its advantages and disadvantages in application to MO problems. A fuzzy update strategy is designed to help PSO overcome difficulties in solving MO problems with lots of local minimum. Coevolutionary PSO algorithm and distributed PSO algorithm are implemented to reduce processing time in solving complex MO problems. And PSO is also applied to bin packing problem to illustrate that PSO can be used to solve real world combinatorial problems.

1.2 Contributions

1.2.1 MOPSO Algorithm Design

The design of fuzzy particle updating strategy and synchronous particle local search: The fuzzy updating strategy models the uncertainty associated with the optimality of global best, thus helping the algorithm to avoid undesirable premature convergence. The synchronous particle local search performs directed local fine-tuning, which helps to discover a well-distributed Pareto front. Experiment shows that balance between the exploration of fuzzy update and the exploitation of SPLS is the key for PSO to solve complex MO problems. Without them, PSO can not deal with multi-modal MO problems effectively.

The formation of a novel Competitive and Cooperative Coevolution Model for MOPSO: As an instance of the divide-and-conquer paradigm, the proposed competitive and cooper-

ative coevolution model helps to produce reasonable problem decompositions by exploiting any correlation or interdependency between subcomponents. This proposed method was validated through comparisons with existing state of the art multiobjective algorithms through the use of established benchmarks and metrics. The competitive and cooperative coevolution MOPSO is the only algorithm to attain the true Pareto front in all test problems, and in all cases converges faster to the true Pareto front than any other algorithm used.

The implementation of Distributed Coevolutionary MOPSO: A distributed coevolutionary particle swarm optimization algorithm (DCPSO) is implemented to exploit the inherent parallelism of coevolutionary particle swarm optimization. DCPSO is suitable for concurrent processing that allows inter-communication of subpopulations residing in networked computers, and hence expedites the computational speed by sharing the workload among multiple computers.

1.2.2 Application of MOPSO to Bin Packing Problem

The mathematical formulation of two-objective, two-dimensional bin packing problems: The bin packing problem is widely found in applications such as loading of tractor trailer trucks, cargo airplanes and ships, where a balanced load provides better fuel efficiency and a safer ride. In these applications, there are often conflicting criteria to be satisfied, i.e., to minimize the bins used and to balance the load of each bin, subject to a number of practical constraints. Existing bin packing problem studies mostly focus on the minimization of wasted space. And only Amiouny [3] has addressed the issue of balance (to make the center of gravity of packed items as close as possible to target point). However Amiouny assumed that all the items can be fitted into the bin, which will lead to bins with loosely packed items. In this thesis, a two-objective, two-dimensional bin packing model (MOBPP-2D) is formulated as no such model is available in literature. The minimum wasted space and the balancing of load are the two objectives in MOBPP-2D, which provides a good representation for the real-world bin packing problems.

The creative application of PSO concept on MOBPP-2D problem: The basic PSO concept instead of the rigid original formula has been applied to solve multiobjective bin packing problems. The best bin instead of the best solution is used to guide the search because bin level permutation may keep more information about previous solutions and help avoid random search. Multi-objective performance tests have shown that the proposed algorithm performs consistently well for the test cases used.

In conclusion, although PSO is a relatively new optimization technique, our research work has shown that PSO has great potential on solving MO problems. The successful application to discrete bin packing problem also proves that PSO's application is not limited to continuous problem as it is originally designed.

1.3 Thesis Outline

This work is organized into seven chapters. The necessary concepts and definitions of multi-objective optimization and particle swarm optimization algorithm are covered in Chapter 2. It also presented an introduction to MOPSOs, with a general framework which illustrates the basic design issues of the state-of-the-arts. Subsequently, a survey on the basic MO algorithm components of fitness assignment, diversity maintenance and elitism is presented to highlight the development trends of multi-objective problem solving techniques.

Chapter 3 addresses the issue of PSO's fast convergence to local minimum. In particular, two mechanisms, fuzzy gbest and synchronous particle local search, are developed to improve algorithmic performance. Subsequently, the proposed multi-objective particle swarm optimization algorithm incorporating these two mechanisms are validated against existing multi-objective optimization algorithms.

Chapter 4 extends the notion of coevolution to decompose the problem and track the optimal solutions in multi-objective particle swarm optimization. Most real-world multi-objective problems are too complex for us to have a clear vision on how to decompose them

by hand. Thus, it is desirable to have a method to automatically decompose a complex problem into a set of subproblems. This chapter introduces a new coevolutionary paradigm that incorporates both competitive coevolution and cooperative coevolution observed in nature to facilitate the emergence and adaptation of the problem decomposition.

Chapter 5 exploits the inherent parallelism of coevolutionary particle swarm optimization to further formulate it into a distributed algorithm suitable for concurrent processing that allows inter-communication of subpopulations residing in networked computers. The proposed distributed coevolutionary particle swarm optimization algorithm expedites the computational speed by sharing the workload among multiple computers.

Chapter 6 addresses the issue of solving bin packing problems using multi-objective particle swarm optimization. Analyzing the existing literature for solving bin packing problems reveals that the current corpus has severe limitation as they focus only on minimization of bins. In fact, some other important objectives, such as the issue of bin balance, also need to be addressed for bin packing problems. Therefore, a multi-objective bin packing problem is formulated and test problems are proposed. To accelerate the optimization process of solving multi-objective bin packing problem, a multi-objective evolutionary particle swarm optimization algorithm is implemented to explore the potential of high convergence speed of PSO on solving multi-objective bin packing problem.

Chapter 7 gives the conclusion and directions for future work.

Chapter 2

Background Materials

2.1 MO Optimization

The specification of MO criteria captures more information about the real world problem as more problem characteristics are directly taken into consideration. For instance, consider the design of a system controller that can be found in process plants, automated vehicles and household appliances. Apart from obvious tradeoff between cost and performance, the performance criteria required by some applications such as fast response time, small overshoot and good robustness, are also conflicting in nature and need to be considered directly [21] [53] [114] [201].

Without any loss of generality, a minimization problem is considered in this work and the MO problem can be formally defined as

$$\min_{\vec{x} \in \vec{X}^{n_x}} \vec{f}(\vec{x}) = \{f_1(\vec{x}), f_2(\vec{x}), ..., f_M(\vec{x})\} \tag{2.1}$$

$$\text{s.t. } \vec{g}(\vec{x}) > 0, \vec{h}(\vec{x}) = 0$$

where $\vec{x}$ is the vector of decision variables bounded by the decision space $\vec{X}^{n_x}$, and $\vec{f}$ is the set of objectives to be minimized. The terms "solution space" and "search space" are often

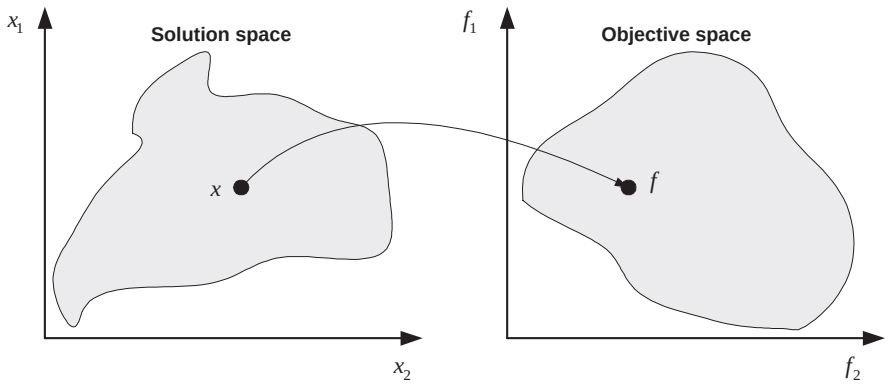

Figure 2.1: Illustration of the mapping between the solution space and the objective space.

used to denote the decision space and will be used interchangeably throughout this work. The functions $\vec{g}$ and $\vec{h}$ represents the set of inequality and equality constraints that defines the feasible region of the n_x-dimensional continuous or discrete feasible solution space. The relationship between the decision variables and the objectives are governed by the objective function $\vec{f} : \vec{X}^{n_x} \longmapsto \vec{F}^M$. Figure 2.1 illustrates the mapping between the two spaces. Depending on the actual objective function and constraints of the particular MO problem, this mapping may not be unique.

2.1.1 Totally conflicting, nonconflicting, and partially conflicting MO problems

One of the key differences between SO (single objective) and MO optimization is that MO problems constitute a multi-dimensional objective space $\vec{F}^M$. This leads to three possible instances of MO problem, depending on whether the objectives are totally conflicting, nonconflicting, or partially conflicting [193]. For MO problems of the first category, the conflicting nature of the objectives are such that no improvements can be made without violating any constraints. This result in an interesting situation where all feasible solutions are also optimal. Therefore, totally conflicting MO problems are perhaps the simplest of the

three since no optimization is required. On the other extreme, a MO problem is nonconflicting if the various objectives are correlated and the optimization of any arbitrary objective leads to the subsequent improvement of the other objectives. This class of MO problem can be treated as a SO problem by optimizing the problem along an arbitrarily selected objective or by aggregating the different objectives into a scalar function. Intuitively, a single optimal solution exist for such a MO problem.

More often than not, real world problems are instantiations of the third type of MO problems with partially conflicting objectives and this is the class of MO problems that we are interested in. One serious implication is that a set of solutions representing the tradeoffs between the different objectives is now sought rather than an unique optimal solution. Consider again the example of cost vs performance of a controller. Assuming that the two objectives are indeed partially conflicting, this presents at least two possible extreme solutions, one for lowest cost and one for highest performance. The other solutions, if any, making up this optimal set of solutions represent the varying degree of optimality with respect to these two objectives. Certainly, our conventional notion of optimality gets thrown out of the window and a new definition of optimality is required for MO problems.

2.1.2 Pareto Dominance and Optimality

Unlike SO optimization where there is a complete order exist (i.e, $f_1 \leq f_2$ or $f_1 \geq f_2$), $\vec{X}^{n_x}$ is partially-ordered when multiple objectives are involved. In fact, there are three possible relationships among the solutions defined by Pareto dominance.

Definition 2.1: Weak Dominance:$\vec{f_1} \in \vec{F}^M$ weakly dominates $\vec{f_2} \in \vec{F}^M$, denoted by $\vec{f_1} \preceq \vec{f_2}\ iff\ f_{1,i} \leq f_{2,i}\ \forall i \in \{1, 2, ..., M\}$ and $f_{1,j} < f_{2,j}\ \exists j \in \{1, 2, ..., M\}$

Definition 2.2: Strong Dominance: $\vec{f_1} \in \vec{F}^M$ strongly dominates $\vec{f_2} \in \vec{F}^M$, denoted by $\vec{f_1} \prec \vec{f_2}\ iff\ f_{1,i} < f_{2,i}\ \forall i \in \{1, 2, ..., M\}$

Definition 2.3: Incomparable: $\vec{f_1} \in \vec{F}^M$ is incomparable with $\vec{f_2} \in \vec{F}^M$, denoted by $\vec{f_1} \sim \vec{f_2}\ iff\ f_{1,i} > f_{2,i}\ \exists i \in \{1, 2, ..., M\}$ and $f_{1,j} < f_{2,j}\ \exists j \in \{1, 2, ..., M\}$

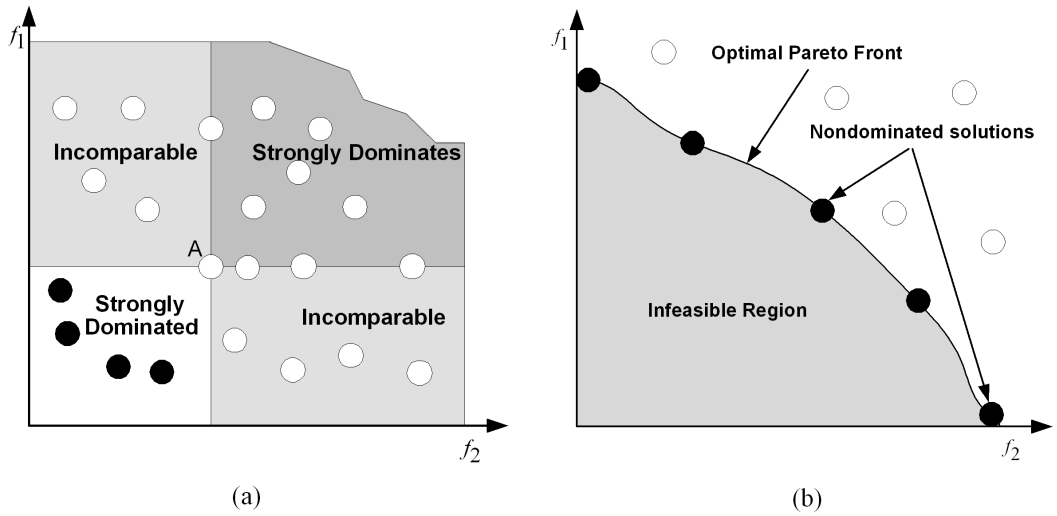

Figure 2.2: Illustration of the (a) Pareto Dominance relationship between candidate solutions relative to solution A and (b) the relationship between the Approximation Set, PF^A and the true Pareto front, PF^*.

With solution A as our point of reference, the regions highlighted in different shades of grey in Figure 2.2(a) illustrates the three different dominance relations. Solutions located in the dark grey regions are dominated by solution A because A is better in *both* objectives. For the same reason, solutions located in the white region dominates solution A. Although A has a smaller objective value as compared to the solutions located at the boundaries between the dark and light grey regions, it only weakly dominates these solutions by virtue of the fact that they share a similar objective value along either one dimension. Solutions located in the light grey regions are incomparable to solution A because it is not possible to establish any superiority of one solution over the other: solutions in the left light grey region are better only in the second objective while solutions in the right light grey region are better only in the first objective.

With the definition of Pareto dominance, we are now in the position to consider the set of solutions desirable for MO optimization.

Definition 2.4: Pareto Optimal Set (PS^*): The Pareto optimal set is the set of nondominated solutions such that $PS^* = \{\vec{x}_i^* | \nexists \vec{F}(\vec{x}_j) \prec \vec{F}(\vec{x}_i^*), \vec{F}(\vec{x}_j) \in \vec{F}^M\}$.

Definition 2.5: Pareto Optimal Front (PF*): The Pareto optimal front is the set of objective vectors of nondominated solutions such that PF$^* = \{\vec{f}_i^* | \nexists \vec{f}_j \prec \vec{f}_i^*, \vec{f}_j \in \vec{F}^M\}$.
The nondominated solutions are also termed "noninferior", "admissible" or "efficient" solutions. Each objective component of any nondominated solution in the Pareto optimal set can only be improved by degrading at least one of its other objective components [184].

2.1.3 MO Optimization Goals

An example of the PF* is illustrated in Figure 2.2(b). Most often, information regarding the PF* are either limited or not known *a priori*. It is also not easy to find a nice closed analytic expression for the tradeoff surface because real-world MO problems usually have complex objective functions and constraints. Therefore, in the absence of any clear preference on the part of the decision-maker, the ultimate goal of multi-objective optimization is to discover the entire Pareto front. However, by definition, this set of objective vectors is possibly an infinite set as in the case of numerical optimization and it is simply not achievable.

On a more practical note, the presence of too many alternatives could very well overwhelm the decision-making capabilities of the decision-maker. In this light, it would be more practical to settle for the discovery of as many nondominated solutions possible as computational resources permits. More precisely, the goal is to find a good approximation of the PF* and this approximate set, PF^A should satisfy the following optimization objectives.

- Minimize the distance between the PF^A and PF*.
- Obtain a good distribution of generated solutions along the PF^A.
- Maximize the spread of the discovered solutions.

An example of such an approximation is illustrated by the set of nondominated solutions denoted by the filled circles residing along the PF* in Figure 2.2(b). While the first optimization goal of convergence is the first and foremost consideration of all optimization

problems, the second and third optimization goal of maximizing diversity are unique to MO optimization. The rationale of finding a diverse and uniformly distributed PF^A is to provide the decision maker with sufficient information about the tradeoffs among the different solutions before the final decision is made. It should also be noted that the optimization goals of convergence and diversity are somewhat conflicting in nature, which explains why MO optimization is much more difficult than SO optimization.

2.2 Particle Swarm Optimization Principle

Particle swarm optimization (PSO) was first introduced by James Kennedy (a social psychologist) and Russell Eberhart (an electrical engineer) in 1995 [92], which originates from the simulation of behavior of bird flocks. Although a number of scientists have created computer simulations of various interpretations of the movement of organisms in a bird flock or fish school, Kennedy and Eberhart became particularly interested in the models developed by Heppner (a zoologist) [62].

In Heppner's model, birds would begin by flying around with no particular destination and in spontaneously formed flocks until one of the birds flew over the roosting area. To Eberhart and Kennedy, finding a roost is analogous to finding a good solution in the field of possible solutions. And they revised Heppner's methodology so that particles will fly over a solution space and try to find the best solution depending on their own discoveries and past experiences of their neighbors.

In the original version of PSO, each individual is treated as a volume-less particle in the D dimensional solution space. The equations for calculating velocity and position of particles are shown below:

$$v_{id}^{k+1} = v_{id}^{k} + r_1^k \times p \times sgn(p_{id}^k - x_{id}^k) + r_2^k \times g \times sgn(p_{gd}^k - x_{id}^k) \tag{2.2}$$

$$x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k} \tag{2.3}$$

where $d = 1, 2, ..., D$ and D is the dimension of search space; $i = 1, 2, ..., N$ and N is the size of swarm; $k = 1, 2, ...$ denotes the number of cycles (iterations); v_{id}^k is the d dimension velocity of particle i in cycle k; x_{id}^k is the d dimension position of particle i in cycle k; p_{id}^k is the d dimension of personal best of particle i in cycle k; p_{gd}^k is the d dimension of global best in cycle k; p and g are the increment step size; r_1^k and r_2^k are two random values uniformly distributed in the range $[0, 1]$; $sgn()$ is the sign function which returns the sign of the number.

Such a simple paradigm was proved to be able to optimize simple two-dimensional linear functions. With increment step size set relatively high, the flock will cluster in a tiny circle around the target in a few cycles. With increment step size set low, the flock will gradually approach the target, swing out sometimes, and finally landing on the target. A high value of p relative to g result in excessive wandering of some isolated particles through the solution space, while the reverse results in the flock rushing prematurely toward local minimum. Approximately equal values of the two increments seem to result in most effective search of the solution space.

2.2.1 Adjustable Step Size

Further research shows that to adjust velocity not by a fixed step size but according to the distance between current position and best position can improve performance. So the equation 2.2 is changed to:

$$v_{id}^{k+1} = v_{id}^k + c_1 \times r_1^k \times (p_{id}^k - x_{id}^k) + c_2 \times r_2^k \times (p_{gd}^k - x_{id}^k) \tag{2.4}$$

Here c_1 is called cognition weight and c_2 is called social weight. Low values of them allow particles to roam far from target regions before being tugged back; while high values results in abrupt movement toward, or past, target region. Because there is no way to guess whether c_1 or c_2 should be larger, they are usually both set to 2.

One important parameter V_{max} is also introduced, and the particle's velocity on each dimension cannot exceed V_{max}. If V_{max} is too large, particle may fly past good solutions. If V_{max} is too small, particle may not explore sufficiently beyond locally good regions. V_{max} is usually set at 10-20% of the dynamic range of each dimension. And early experiments showed that a population of 20-50 is enough for most applications.

2.2.2 Inertial Weight

The maximum velocity V_{max} serves as a constraint to control the exploration ability of a particle swarm. To better control the exploration and exploitation in particle swarm optimization, the concept of inertial weight (w) was developed. Inertial weight was first introduced by Shi and Eberhart in 1998 [176] with the motivation to eliminate the need for V_{max}. After incorporating w, equation 2.4 is changed to:

$$v_{id}^{k+1} = w \times v_{id}^{k} + c_1 \times r_1^k \times (p_{id}^k - x_{id}^k) + c_2 \times r_2^k \times (p_{gd}^k - x_{id}^k) \tag{2.5}$$

Shi and Eberhart argued that a large inertial weight facilitates the global search (discovering new places), while a small inertial weight facilitates local search (fine-tuning) [174]. A linearly decreasing inertial weight from 0.9 to 0.4 is recommended.

2.2.3 Constriction Factor

The work done by Clerc and Kennedy [23] showed that the use of a constriction factor K may be necessary to insure convergence for PSO. The introduced constriction factor influences the velocity of the particles by dampening it. And the equation for incorporating K into PSO is as follows:

$$v_{id}^{k+1} = k \times [v_{id}^{k} + c_1 \times r_1^k \times (p_{id}^k - x_{id}^k) + c_2 \times r_2^k \times (p_{gd}^k - x_{id}^k)] \tag{2.6}$$

where $k = \frac{2}{|2-\varphi-\sqrt{\varphi^2-4\varphi}|}$, $\varphi = c_1 + c_2$, and $\varphi \succ 4$. When the constriction factor is used, φ is usually set to 4.1. The constriction factor is thus 0.729. And $c_1 = c_2 = 0.729 \times 2.05 = 1.49445$.

2.2.4 Other Variations of PSO

One version of the algorithm reduces the two best positions (gbest and pbest) to one that is the midway between them in each dimension. This version is not successful because it has a bad tendency to converge to that point whether it is an optimum or not.

In another version, the momentum term is removed. The adjustment was as follows:

$$v_{id}^{k+1} = c_1 \times r_1^k \times (p_{id}^k - x_{id}^k) + c_2 \times r_2^k \times (p_{gd}^k - x_{id}^k) \tag{2.7}$$

This version, though simplified, proved to be ineffective for finding the global optimum.

Eberhart and Kennedy [41] has also tried a local version of PSO, in which particles only have information of their own bests and their neighbor's bests, rather than that of the entire swarm. This is a circular structure where each particle is connected to its K immediate neighbors. In this way, one segment of the swarm may converge on a local minimum, while another segment converges on a different minimum or keep searching. Information spreads from neighbor to neighbor, until one optimum, which is really the best one found by any part of the swarm, eventually drags all particles to it. This version of PSO is less likely to be entrapped into a local minimum, but it clearly requires more cycles on average in order to reach a criterion error level.

2.2.5 Terminology for PSO

In order to establish a common terminology, in the following we provide definitions of the technical terms commonly used in PSO.

Particle: Individual. Each particle represents a point in the solution space, which can move (fly) in the solution space to search for good solutions.

Swarm: Population or group of particles. PSO is a population based algorithm, which uses a large population of particles to search for good solutions simultaneously.

Cycle: Iteration. Each cycle represents a change of positions for all particles for one time.

Velocity: Velocity represents how far each particle move (fly) in each cycle in solution space.

Inertia Weight: Denoted by w, the inertia weight controls the impact of the previous velocity on the current velocity of a given particle.

pbest: Personal best position found by a given particle, so far.

gbest: Global best position found by the entire swarm of particles.

Cognition Weight: Denoted by c_1, cognition weight represents the attraction that a particle has toward its personal best position.

Social Weight: Denoted by c_2, social weight represents the attraction that a particle has toward the global best position found by the entire swarm.

2.3 Multi-objective Particle Swarm Optimization

Many different metaheuristical approaches, such as cultural algorithm, particle swarm optimization, evolutionary algorithm, artificial immune systems, differential evolution, and simulated annealing, have been proposed since the pioneering effort of Schaffer in [168]. All these algorithms are different in methodology, particularly in the generation of new candidate solutions. Among these metaheuristics, MOPSO is one of the most promising stochastic search methodology because of its easy implementation and high convergence speed.

```
P ← Particle Initialization
A ← Create Archive
While (Stopping criteria not satisfied)
        P ← Evaluate(P)
        A ← Update(A)
        P ← Select pbest(P)
        P ← Select gbest(P)
        P ← Update(P)
End  While
```

Figure 2.3: Framework of MOPSO

2.3.1 MOPSO Framework

The general MOPSO framework can be represented in the pseudocode shown in Figure 2.3. There are many similarities between SO particle swarm optimization algorithms (SOPSOs) and MOPSOs with both techniques involving an iterative adaptation of a set of solutions until a pre-specified optimization goal/stopping criterion is met. What sets these two techniques apart is the manner in which solution assessment and gbest selection are performed. This is actually a consequence of the three optimization goals described in Section 2.1.3. In particular, solution assessment must exert a pressure to drive the particles toward the global tradeoffs as well as to diversify the particles uniformly along the discovered PF^A. The incorporation of elitism is one distinct feature that characterizes state of the art MOPSO algorithms. Elitism in MOPSO involves two closely related process, 1) the archiving of good solutions and 2) the selection of gbest for each particle from these solutions. The archive updating and the selection of gbest for each particle must also take diversity into consideration to encourage and maintain a diverse solution set. While the general motivations may be similar, different MOPSO algorithms can be distinguished by the way in which the mechanisms of elitism and diversity preservation are implemented.

The optimization process of MOPSO starts with the initialization of the swarm. This is followed by evaluation and density assessment of candidate solutions. After which, good solutions are updated into an external archive. MOPSOs perform the archiving process

differently. Nonetheless, in most cases, a truncation process will be conducted based on some density assessment to restrict the number of archived solutions. The pbest selection process is the comparison of particle's current position and former pbest position. And the gbest selection process typically involves the set of nondominated solutions updated in the previous stage. Then the PSO updating operators are applied to explore and exploit the search space for better solutions.

2.3.2 Basic MOPSO Components

The framework presented in the previous section serves to highlight the primary components of MOPSO, elements without which the algorithm is unable to fulfill its basic function of finding PF^* satisfactorily. More elaborate works on each component with different concerns exist in the literature.

Fitness Assignment

As illustrated in Figure 2.4, solution assessment in MOPSO should be designed in such a way that a pressure $\overleftarrow{P}_n$ is exerted to promote the solutions in a direction normal to the tradeoffs region and at the same time, another pressure $\overleftarrow{P}_t$ to promote the solutions in a direction tangential to that region. These two orthogonal pressures result in the unified pressure $\overleftarrow{P}_u$ to direct the particle search in the MO optimization context. Based on the literature, it is possible to identify two different classes of fitness assignment: 1) Pareto based assignment, 2) aggregation based assignment.

Pareto based Fitness Assignment: Pareto based MOPSOs have emerged as the most popular approach [2] [26] [47] [109] [130]. On its own, Pareto dominance is unable to induce $\overleftarrow{P}_t$ and the solutions will converge to arbitrary portions of the PF^A, instead of covering to the whole surface. Thus Pareto based fitness assignments are usually applied in conjunction with density measures, which usually adopts a two stage process where comparison between

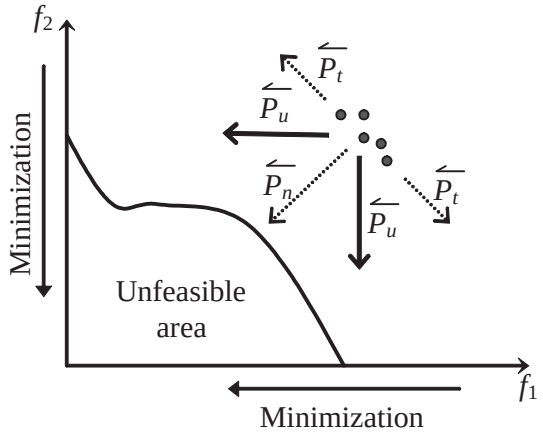

Figure 2.4: Illustration of pressure required to drive evolved solutions towards PF*

solutions is conducted based on Pareto fitness before density measure is used. Note that this indirectly assigns higher priority level to proximity. Another interesting consequence is that $\overleftarrow{P}_n$ will be higher in the initial stages of the search; and when the solutions begin to converge to the PF*, the influence of $\overleftarrow{P}_t$ becomes more dominant as most of the solutions are equally fit.

However, Fonseca and Fleming [54] highlighted that Pareto based assignment may not be able to produce sufficient guidance for search in high-dimensional problems and it has been shown empirically that Pareto based MO algorithms do not scale well with respect to the number of objectives in [95]. To understand this phenomenon, let us consider a M-objective problem where M>>2. Under the definition of Pareto dominance, as long as a solution has one objective value that is better than another solution, never mind the degree of superiority, it is still considered to be nondominated even if it is grossly inferior in the other M-1 objectives. Intuitively, the number of nondominated solutions in the searching population grows with the number of objectives resulting in the lost of sufficient pressure toward PF*.

To this end, some researchers have sought to relax the definition of Pareto optimality. Ikeda *et al* [78] proposed the α-dominance scheme which considers the contribution of all the weighted difference between the respective objectives of any two solutions under comparison to prevent the above situation from occurring. Mostaghim *et al* [130] and Reyes *et al* [157] implemented an ϵ-dominance scheme which has the interesting property of ensuring convergence and diversity. In this scheme, an individual dominates another individual only if it offers an improvement in all aspects of the problem by a pre-defined factor of ϵ. A significant difference between α-dominance and ϵ-dominance is that a solution that strongly dominates another solution also α-dominates that solution while this relationship is not always valid for the latter scheme. Another interesting alternative in the form of fuzzy Pareto optimality is presented by Farina and Amato [45] to take into account the number and size of improved objective values.

Aggregation based Fitness Assignment: The aggregation of the objectives into a single scalar is perhaps the simplest approach to generate PF^A. Unlike the Pareto based approach, aggregation based fitness induces $\overleftarrow{P}_u$ directly. However, aggregation is usually associated with several limitations such as its sensitivity to PF^* shape and the lack of control on the direction of $\overleftarrow{P}_u$. This results in the contrasting lack of interest paid by MO optimization researchers as compared to Pareto based techniques. Ironically, the failure of Pareto based MO algorithms in high-dimensional objective space may well turn the attention towards the use of aggregation based fitness assignment in MO algorithms.

Baumgartner *et al* [8] and Parsopoulos *et al* [145] implemented aggregation-based multi-objective particle swarm optimization algorithms that have been demonstrated to be capable of evolving uniformly distributed and diverse PF^A. In [8], the swarm is partitioned into n subswarms, each of which uses a different set of weights. Parsopoulos *et al* investigated two very interesting aggregation based MOPSO approaches in [145]. In one approach, the weights of each objective can be changed between 1 or 0 periodically during the optimization process (called Bang-Bang weighted aggregation); while in the other, the weights are gradu-

ally modified according to some predefined function (called dynamic weighted aggregation).

Instead of performing the aggregation of objective values, Hughes [74] suggested the aggregation of individual performance with respect to a set of predetermined target vectors. In this approach, individuals are ranked according to their relative performance in an ascending order for each target. These ranks are then sorted and stored in a matrix such that it may be used to rank the population, with the most fit being the solution that achieves the best scores most often. It has been shown to outperform many nondominated sorting algorithms for high-dimensional MO problems [74].

At this point of time, it seems that Pareto based fitness are more effective in low-dimensional MO problems while aggregation based fitness has an edge with increasing number of objectives. Naturally, some researchers have attempted to marry both methods together. For example, Turkcan and Akturk [204] proposed an hybrid MO fitness assignment method which assigns a nondominated rank that is normalized by niche count and an aggregation of weighted objective values. On the other hand, Pareto-based fitness and aggregation-based fitness are used independently in various stages of the searching process in [46, 133].

Diversity Preservation

Density Assessment: A basic component of diversity preservation strategies is density assessment. Density assessment evaluates the density at different sub-divisions in a feature space, which may be in the parameter or objective domain. Depending on the manner in which solution density is measured, the different density assessment techniques can be broadly categorized under 1) Distance-based, 2) Grid-based, and 3) Distribution-based. One of the basic issues to be examined is whether density assessment should be computed in the decision space or objective space. Horn and Nafpliotis [70] stated that density assessment should be conducted in the feature space where the decision-maker is most concerned about its distribution. Since many people are more interested in obtaining a well-distributed and

diverse PF^A, most works reported in the MO literature applied density assessment in the objective space.

Distance-based assessment is based on the relative distance between individuals in the feature space. Examples include niche sharing [109] [165] [186], *k-th* nearest neighbor [166] [224] and crowding [153] [155] [157]. Niche sharing or niching is by far the most popular distance-based approach.

Niching was proposed by Goldberg [60] to promote population distribution as well as to search for possible multiple peaks in SO optimization. The main limitation of this method is that its performance is sensitive to the setting of niche radius. Fonseca and Fleming [54] gave some bounding guidelines of appropriate niche radius values for MO problems when the number of individuals in the population and the minimum/maximum values in each objective dimension are given. However, such information are often not known *a prior* in many real-world problems. Tan *et al* [195] presented a dynamic sharing scheme where the niche radius is computed online based on the evolved tradeoffs.

The *k-th* nearest neighbor is another approach which requires the specification of an external parameter. In [166], average Euclidean distance to the two nearest solutions is used as the measure of density. Zitzler *et al* [224] adopted k as the square root of the total population size based on some rule-of-the-thumb used in statistical density estimation. Like niching, this approach is sensitive to the setting of the external parameter, which in this case is k. The *k-th* nearest neighbor can also be misleading in situations where all the nearest neighbors are located in a similar region of the feature space. In certain sense, the nearest neighbor is similar to the method of crowding. However, crowding do not have such bias since it is based on the average distance of the two points on either side of current point along each dimension of the feature space. Crowding is not influenced by any external parameters. Nonetheless, distance-based assessment schemes (nitching, crowding and *k-th* nearest neighbor) are susceptible to scaling issues and their effectiveness are limited by the presence of noncommensurable objectives.

Grid-based assessment is probably the most popular approach after niching [26] [28]. In this approach, the feature space is divided into a predetermined number of cells along each dimension and distribution density within a particular cell has direct relation to the number of individuals residing within that cell location. Contrary to distance-based assessment methods which include methods that are very different, both conceptually and in implementation, the main difference among the various implementation of this approach, if any, lies in how cells and individuals are located and referenced. For example, the cell location of an individual in [26] is found using recursive subdivision. However, in [118], the location of each cell center is stored and the cell associated with an individual is found by searching for the nearest cell center. The primary advantage of grid-based assessment is that it is not affected by the presence of noncommensurable objectives. However, this technique depends heavily on the number of cells in the feature space containing the population and it works well if knowledge of the geometry of the PF^* is known. Furthermore, its computational requirements are considerably more than distance-based assessments.

Distribution-based assessment is rather different from distance-based and grid-based methods because distribution density is based on the probability density of the individuals. The probability density is used directly in [11] to identify least crowded regions of the PF^A. It has also been used to compute the entropy as a means to quantify the information contributed by each individual to the PF^A in [31] [99] [187]. Like grid-based methods, it is not affected by noncommensurable objectives. The tradeoff is that it can be computationally intensive because it involves the estimation of probability density of the individuals. On the other hand, the computational effort is a linear function of population size which is advantageous for large population sizes. While some distribution-based methods require external parameter setting such as the window width in Parzen window estimation [58], there exist an abundance of guidelines in the literature.

An empirical investigation is conducted in [96] on the effectiveness of the various density assessment methods in dealing with convex, nonconvex and line distributions. In general,

the study shows that all techniques under investigation are able to improve distribution quality in the sense of uniformity. But the findings also suggest that it is not possible to ascertain which method is better for which type of problem because of the interactions between density assessment and searching process.

Encouraging Diversity Growth: Other means of encouraging diversity growth can also be found in the literature. For instance, in [203], Toffolo and Benini applied diversity as an objective to be optimized. Specifically, the MO problem is transformed into a two-objective problem with diversity as one of the objectives and the other objective being the ranks with respect to the objectives of the original MO problem.

Diversity can also be encouraged through the simultaneous searching of multiple isolated subswarms. In [8], each subswarm is guided towards a particular region of PF^* through using different sets of weights on objectives. Chow *et al* [22] assigned one subswarm for each objective. A communication channel is established between the neighboring subswarms for transmitting the information of the best particles, in order to provide guidance for improving their objective values.

Elitism

The use of the elitist strategy is conceptualized by De Jong in [36] to preserve the best individuals found during the searching process. When elite individuals are used to guide the search, it can improve convergence greatly, although it maybe achieved at the risk of premature convergence. Zitzler [227] is probably the first to introduce elitism into MO algorithms, sparking off the design trend of a new generation of MO algorithms [24].

Archiving: The first issue to be considered in the incorporation of elitism is the storage or archiving of elitist solutions. Archiving usually involves an external repository and this process is much more complex than in SOs since we are now dealing with a set of Pareto optimal solutions instead of a single solution. The fact that PF^* is an infinite set raises the natural question of *what should be maintained?* Without any restriction on the archive size,

the number of nondominated solutions can grow exceedingly large. Therefore, in the face of limited computing and memory resources in implementation, it is sometimes unwise to store all the nondominated solutions found. Most works enforce a bounded set of nondominated solutions which requires a truncation process when the size of the nondominated solutions exceeds a predetermined bound [26] [111]. This leads to the question of *which solution should be kept?* It is only natural to truncate the archive based on some form of density assessment discussed earlier when the number of nondominated solutions exceeds the upper bound.

For bounded archiving, two implementations of truncation can be found in the literature, i.e., batch mode and recurrence mode. The truncation criteria will be based on the density assessment process described earlier. In the batch-mode, all solutions in the archive will undergo density assessment and the worst individuals are removed in a batch. On the other hand, in the recurrence mode, an iterative process of density assessment and truncation is repeated to the least promising solution in the archive until the desired size is achieved. While the recurrence-mode of truncation has higher capability to avoid the extinction of local individuals compared to the batch-mode truncation, the recurrence-mode truncation often requires more computational effort. To reduce the computational cost involved with the pairwise comparison between a new solution and the archived solution in recurrence-mode truncation, a more efficient data-structure has been proposed in [47].

The restriction on the number of archive solutions leads to two phenomena which have a detrimental effect on the searching process. The first is the shrinking PF^A phenomenon which results from the removal of extremal solutions and the subsequent failure to rediscover them. In the second phenomenon, nondominated solutions in the archive are replaced by least crowded particles. In the subsequent cycles, new particles that would have been dominated by the removed solutions are updated into the archive. Repeated cycles of this process is known as the oscillating PF^A. The alternative and simplest approach is, of course, to store all the nondominated solutions found.

Selection of gbest: The next issue to be considered is the introduction of elitist solutions into the gbest selection process. Contrary to SO optimization, the gbest for MO optimization exist in the form of a set of nondominated solutions which inevitably leads to the issue of gbest selection for each particle. One problem faced is the "exploration-exploitation" dilemma. A higher degree of exploitation attained through the selection of gbest according to domination relationship leads to the lost of diversity. The consequences of the lack of necessary diversity to fuel the searching process is a PF^A that fails to span the entire PF^* uniformly and, in the worst case, premature convergence to local optimal solutions. While too much exploration through selection of least crowded nondominated solution as gbest may lead to slow convergence speed.

Gbest selection schemes that sought to balance the tradeoff between exploration and exploitation have been proposed. Alvarez-Benitez *et al* [2] presented a general framework for MOPSOs which allows designers to control the balance between exploration of diversity and exploitation of proximity. Tan *et al* [192] proposed an enhanced exploration strategy in which the ratio of solutions selected based on ranking and diversity is adapted based on an online performance measure. Solutions selected on the basis of rank are subjected to turbulence operators while those selected based on niche count undergo local search to improve solution distribution.

2.3.3 Benchmark Problems

Benchmark problems are used to reveal the capabilities, important characteristics and possible pitfalls of the algorithm under validation. In the context of MO optimization, these test functions must pose sufficient difficulty to impede MO algorithm's search for Pareto optimal solutions. Deb [33] has identified several characteristics that may challenge MO algorithm's ability to converge and maintain population diversity. Multi-modality is one of the characteristics that hinder convergence. Convexity, discontinuity and non-uniformity of the PF^* may prevent the MOPSO from finding a diverse set of solution.

Table 2.1: Definition of ZDT test problems

Problem	Definition
ZDT1	$f_1(x_1) = x_1$,
	$g(x_2, ...x_m) = 1 + 9 \cdot \frac{\sum_{i=2}^{m} x_i}{(m-1)}$,
	$h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}}$
	where $m = 30$, $x_i \in [0, 1]$
ZDT2	$f_1(x_1) = x_1$,
	$g(x_2, ...x_m) = 1 + 9 \cdot \frac{\sum_{i=2}^{m} x_i}{(m-1)}$,
	$h(f_1, g) = 1 - (\frac{f_1}{g})^2$
	where $m = 30$, $x_i \in [0, 1]$
ZDT3	$f_1(x_1) = x_1$,
	$g(x_2, ...x_m) = 1 + 9 \cdot \frac{\sum_{i=2}^{m} x_i}{(m-1)}$,
	$h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} - \frac{f_1}{g} \sin(10\pi f_1)$
	where $m = 30$, $x_i \in [0, 1]$
ZDT4	$f_1(x_1) = x_1$,
	$g(x_2, ...x_m) = 1 + 10(m-1) + \sum_{i=2}^{m} (x_i^2 - 10\cos(4\pi x_i))$,
	$h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}}$
	where $m = 10$, $x_1 \in [0, 1]$, $-1 \leq x_i < 1$, $\forall i = 2, ..., 10$
ZDT6	$f_1(x_1) = 1 - exp(-4x_1) \cdot \sin^6(6\pi x_1)$,
	$g(x_2, ...x_m) = 1 + 9 \cdot (\frac{\sum_{i=2}^{m} x_i}{m-1})^{0.25}$,
	$h(f_1, g) = 1 - (\frac{f_1}{g})^2$
	where $m = 10$, $x_i \in [0, 1]$

The problems of ZDT1, ZDT2, ZDT3, ZDT4, ZDT6, FON, KUR and POL are selected to validate the effectiveness of multi-objective optimization techniques in converging and maintaining a diverse Pareto optimal solution set in this work. This set of test problems are characterized by the different features mentioned above and should be a good test suite for a fair comparison of different multi-objective algorithms. Many researchers, such as [29] [35] [198] [210] [225], have used these problems in the validations of their algorithms.

The test problems of ZDT1 through ZDT6 are constructed by Zitzler *et al* [225] based

on the guideline described by Deb [33]. The ZDT test problems have the following functional structure.

$$\begin{aligned} &\min f_1(\vec{x}_{d1}) \\ &\min f_2(\vec{x}_{d2}) \quad = \quad g(\vec{x}_{d2}) \cdot h(f_1, g) \end{aligned} \tag{2.8}$$

where $\vec{x}_{d1}, \vec{x}_{d2} \in \vec{x}$, and the g and h functions control the problem difficulty and the shape of the Pareto front respectively. By having independent functions relating to convergence and diversity, this framework facilitates the incorporation of various problem features to construct different test problems. Table 2.1 summaries the definition and features of the various ZDT test problems.

FON [52] is a two-objective minimization test problem that has been widely used in the literature. Besides having a nonconvex Pareto front, there are high interactions between decision variables. FON has a large and nonlinear tradeoff curve that is suitable for challenging an algorithm's ability to find and maintain the entire Pareto front uniformly.

$$\begin{aligned} f_1(x_1, ..., x_8) &= 1 - \exp[-\textstyle\sum_{i=1}^{8}(x_i - \frac{1}{\sqrt{8}})^2], \\ f_2(x_1, ..., x_8) &= 1 + \exp[-\textstyle\sum_{i=1}^{8}(x_i - \frac{1}{\sqrt{8}})^2], \end{aligned} \tag{2.9}$$

where $-2 \leq x_i < 2$, $\forall i = 1, 2, ..., 8$

KUR [105] is characterized by an optimal Pareto front that is non-convex and disconnected, i.e., it contains three distinct disconnected regions on the final tradeoff as shown in Figure 2.5. The global tradeoff for KUR are difficult to be discovered since they are disconnected in the decision variable space. There are high interactions between the decision variables which will also pose problems to the MO algorithms.

$$\begin{aligned} f_1(x_1, x_2, x_3) &= \textstyle\sum_{i=1}^{2}[-10\exp(-0.2\sqrt{x_i^2 + x_{i+1}^2})], \\ f_2(x_1, x_2, x_3) &= \textstyle\sum_{i=1}^{3}[|x_i|^{0.8} + 5 \cdot \sin(x_i^3)], \end{aligned} \tag{2.10}$$

where $x_i \in [-5, 5]$, $\forall i = 1, 2, 3$

POL [149] challenges the algorithm's ability to cope with discontinuities as the Pareto

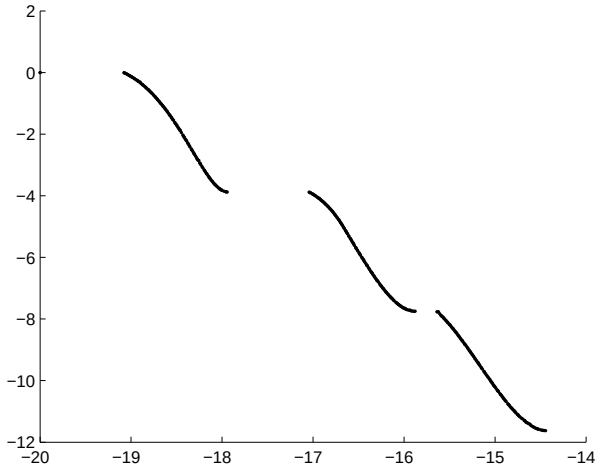

Figure 2.5: True Pareto front of KUR

optimal solutions of POL are discontinuous in the objective space as well as in the decision variable space. Most parts (the lower region) of the Pareto optimal region, as shown in Figure 2.6, are constituted by the boundary solutions of the search space.

$$\begin{aligned} f_1(x_1, x_2) &= 1 + (A_1 - B_1)^2 + (A_2 - B_2)^2 \\ f_2(x_1, x_2) &= (x_1 + 3)^2 + (x_2 + 1)^2 \\ A_1 &= 0.5sin1 - 2cos1 + sin2 - 1.5cos2 \\ A_2 &= 1.5sin1 - cos1 + 2sin2 - 0.5cos2 \\ B_1 &= 0.5sinx_1 - 2cosx_1 + 2sinx_2 - 0.5cosx_2 \\ B_2 &= 1.5sinx_1 - cosx_1 + 2sinx_2 - 0.5cosx_2 \end{aligned} \tag{2.11}$$

where$-\pi \leq x_i \leq \pi$, $\forall i = 1, 2$

2.3.4 Performance Metrics

Performance analysis of different MO algorithms essentially boils down to the evaluation of the approximate Pareto front under the constraints of some computational budget. Then performance metrics or indicators play an important role as functions that return a scalar

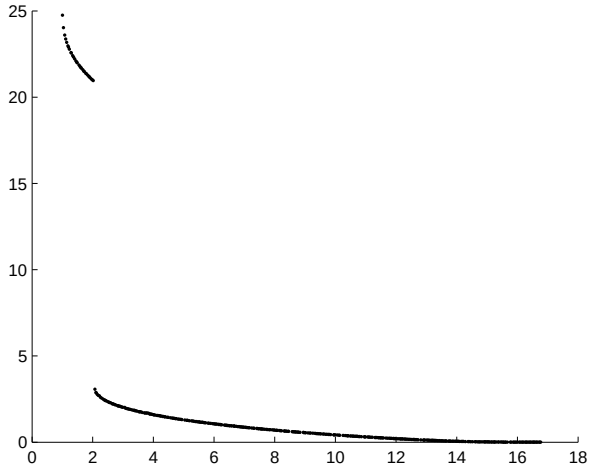

Figure 2.6: True Pareto front of POL

quantity, reflecting the quality of the scrutinized solution set. In SO optimization, this quality comes in the form of the objective function. For MO optimization, however, quality can be defined in a variety of ways, for example, the uniformity of solutions, the dominance relationship between two solution sets and the closeness to the Pareto optimal front.

There have been increasing concerns on the choice of performance metrics. To this end, Knowles and Corne [102] and Zitzler *et al* [223] have discussed at length, the suitability and limitations of various performance metrics. Comparative studies performed by researchers such as Jaszkiewicz [85], Deb *et al* [35], Tan *et al* [192], Veldhuizen and Lamont [210], made use of a suite of unary performance metrics pertinent to the optimization goals of proximity, diversity and distribution, which are used in this work and described as follows.

Proximity Indicator: The metric of generational distance (GD) gives a good indication of the gap between PF^* and PF^A. Mathematically, the metric is a function of individual distance given as,

$$\mathrm{GD} = \frac{1}{n_{PF}} \cdot \left(n_{PF} \sum_{i=1}^{n_{PF}} d_i^2 \right)^{\frac{1}{2}} \tag{2.12}$$

where $n_{PF} = |\mathrm{PF}^A|$, d_i is the Euclidean distance (in objective space) between the i-th

member of PF^A and the nearest member of PF^*. Intuitively, a low value of GD is desirable, which reflects a small deviation between the evolved and the true Pareto front.

Diversity Indicator: The matric of maximum spread (MS) measures how well the true Pareto front is covered by the discovered Pareto front. The equations for calculating MS is as follows,

$$\text{MS} = \Big\{ \frac{1}{M} \sum_{i=1}^{M} \Big(\frac{\min[\overline{f}_i^A, \overline{f}_i^*] - \max[\underline{f}_i^A, \underline{f}_i^*]}{\overline{f}_i^* - \underline{f}_i^*} \Big)^2 \Big\}^{\frac{1}{2}} \tag{2.13}$$

where $\overline{f}_i^A$ and $\underline{f}_i^A$ is the maximum and minimum of the i-th objective in PF^A respectively; $\overline{f}_i^*$ and $\underline{f}_i^*$ is the maximum and minimum of the i-th objective in PF^* respectively. The greater the MS is, the more area of PF^* is covered by the PF^A.

Distribution Indicator: The metric of spacing [172] gives an indication of how evenly the solutions are distributed along the discovered front. It is defined as,

$$\text{S} = \frac{1}{\bar{d}'} \cdot \Big(\frac{1}{n_{PF}} \cdot \sum_{i=1}^{n_{PF}} (d_i' - \bar{d}')^2 \Big)^{\frac{1}{2}} \tag{2.14}$$

$$\bar{d}' = \frac{1}{n_{PF}} \sum_{i=1}^{n_{PF}} d_i'$$

where d_i' is the Euclidean distance (in the objective domain) between the i-th member and its nearest neighbor in PF^A.

General Quality Indicator: By taking into account performance in diversity and proximity, the metric of hypervolume (HV) provides a general quality measure of the solution set. In order to calculate a normalized value and eliminate bias, Veldhuizen and Lamont [210] expressed the metric of HV as a ratio between the HV of PF^A and PF^*,

$$\text{HVR} = \frac{\text{HV}(\text{PF}^A)}{\text{HV}(\text{PF}^*)} \tag{2.15}$$

$$\text{HV} = \text{volume} \bigcup_{i=1}^{n_{PF}} v_i \tag{2.16}$$

Mathematically, for each member $\vec{f}_i^A$ in the non-dominated set, a hypercube v_i is constructed

with a reference point and $\vec{f}_i^A$ as the diagonal corners of the hypercube. The reference point can be found by constructing a vector of the worst objective function values.

2.4 Conclusion

The necessary concepts and definitions of multi-objective optimization and particle swarm optimization algorithm are covered in this chapter. It also presented the general MOPSO framework which illustrates the basic design issues of fitness assignment, diversity maintenance and elitism. Finally, state of the art MOPSO techniques, MO test problems and performance indicators used in this work are discussed.

Chapter 3

A Multiobjective Memetic Algorithm Based on Particle Swarm Optimization

The balance between exploration and exploitation is crucial to the success of search and optimization algorithms [104]. It is known that memetic algorithms (MAs) [129] hybridizing evolutionary algorithms (EAs) and local search (LS) heuristics can be implemented to maintain a balance between exploration and exploitation [17] [55] [81] [124] [139]. However such hybrids are rarely considered in particle swarm optimization algorithm design.

This chapter is concerned with the development of a memetic algorithm within the context of MOPSO. In contrast to single-objective (SO) optimization, it is essential to obtain a well-distributed and diverse solution set for finding the final tradeoff in MO optimization. However, the high speed of convergence in PSO often implies a rapid loss of diversity during the optimization process, which inevitably leads to undesirable premature convergence. Thus, the challenge in designing a multiobjective memetic algorithm based on particle swarm optimization is to deal with the premature convergence without compromising the convergence speed.

Apart from hybridizing LS heuristic and PSO, the existing particle updating strategy in PSO is extended to account for the requirements in MO optimization. Two heuristics

are proposed, including a synchronous particle local search (SPLS) and a fuzzy global-best (f-gbest) for the updating of a particle trajectory. The SPLS utilizes swarm information to perform a directed fine-tuning operation, while the second heuristic is based upon the concept of possibility [40] to deal with the problem of maintaining diversity within the swarm as well as to promote exploration in the search. The proposed multiobjective memetic particle swarm optimization algorithm incorporating f-gbest and SPLS is named as FMOPSO.

The remainder of this chapter is organized as follows: Details of the proposed features in FMOPSO are described in Section 3.1. Section 3.2 examines FMOPSO performance and individual and combined effects of the proposed features, and Section 3.3 presents a comparative study of FMOPSO with well-known MO optimization algorithms on benchmark problems. Conclusions are drawn in Section 3.4.

3.1 Multiobjective Memetic Particle Swarm Optimization

FMOPSO aims to preserve population diversity for finding the optimal Pareto front and to include the feature of local fine-tuning for good population distribution by filling any gaps or discontinuities along the Pareto front. In this section, the proposed features of SPLS and f-gbest will be described, and the implementation detail of FMOPSO will be presented.

3.1.1 Archiving

In FMOPSO, elitism is implemented in the form of a fixed-size archive to prevent the lost of good solutions due to the stochastic nature of the optimization process. The size of archive can be adjusted according to the desired number of particles to be distributed on the tradeoffs in the objective space. The archive is updated at each cycle, e.g., if the candidate solution is not dominated by any members in the archive, it will be added to the archive. Likewise, any archive members dominated by this solution will be removed from the archive. In order to maintain a set of uniformly distributed nondominated particles in

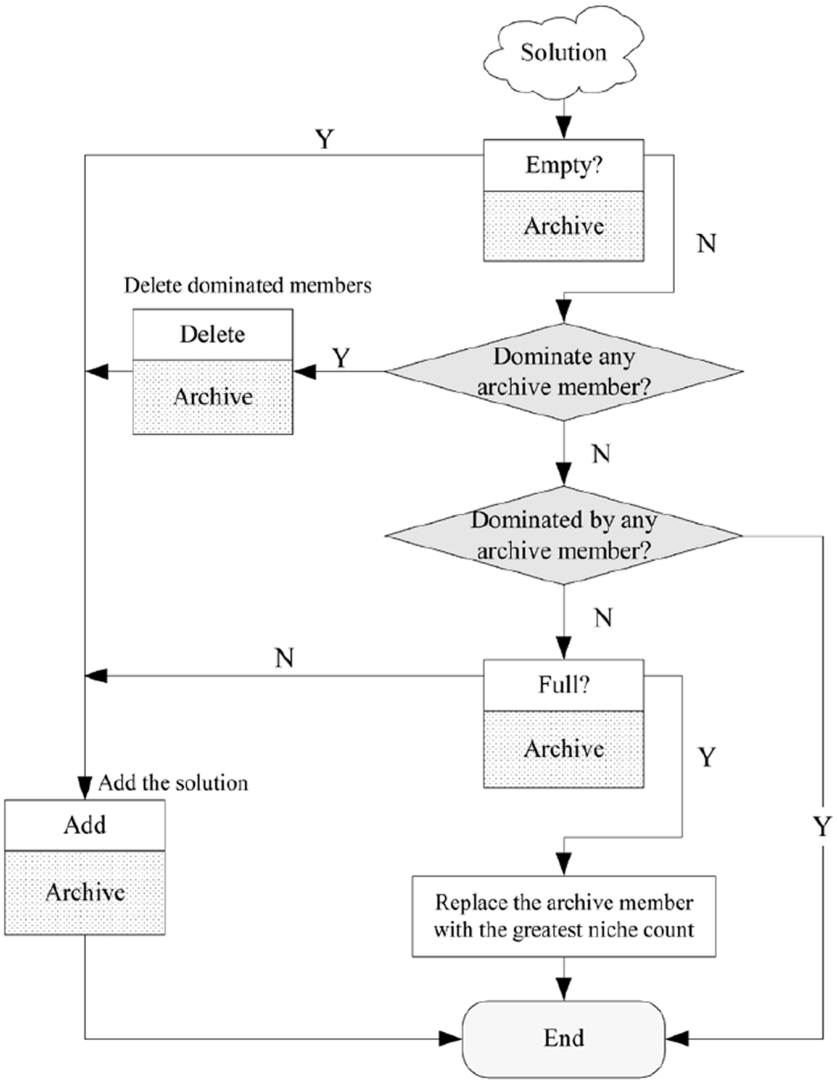

Figure 3.1: The process of archive updating

the archive, the dynamic niche sharing scheme [195] is employed. When the predetermined archive size limit is reached, a recurrent truncation process [96] based on niche count is used to eliminate the most crowded archive member. The archiving process is shown in Figure 3.1.

3.1.2 Selection of Global Best

In MOPSO, gbest plays a very important role in guiding the entire swarm towards the global Pareto front. Contrary to SO optimization, the gbest for MO optimization exist in the form of a set of nondominated solutions which inevitably leads to the issue of selecting the gbest. It is known that the selection of an appropriate gbest is critical for the search of a diverse and uniformly distributed solution set in MO optimization.

Adopting a similar approach in [26], each particle in the swarm will be assigned a nondominated solution from the archive as gbest. Specifically, binary tournament selection of nondominated solutions from the archived is carried out independently for each particle in every cycle. Therefore, each particle is likely to be assigned different archived solution as gbest. This implies that each particle will search along different direction in the decision space, thus aiding the exploration process in the optimization. In order to promote diversity as well as to encourage exploration into the least populated region in the search space, the selection is based on niche count. In the event of a tie, preference will be given to solutions lying at the extreme ends of an arbitrarily selected objective.

3.1.3 Fuzzy Global Best

In PSO, the swarm converges rapidly within the intermediate vicinity of the gbest. However, such a high convergence speed often results in: 1) the lost of diversity required to maintain a diverse Pareto front and 2) premature convergence if the gbest corresponds to a local optimum. This motivates the development of f-gbest, which is based on the concept of possibility measure to model the lack of information about the true optimality of the gbest. In contrast to conventional approaches, the gbest is denoted as "possibly at $(d_1 \times d_2 \times d_3 ... \times d_D)$" instead of a crisp location.

Consequently, the calculation of particle velocity can be rewritten as,

$$p_{cd}^k = N(p_{gd}^k, \sigma) \tag{3.1}$$

$$\sigma = f(k) \tag{3.2}$$

$$v_{id}^{k+1} = w \times v_{id}^{k} + c_1 \times r_1^k \times (p_{id}^k - x_{id}^k) + c_2 \times r_2^k \times (p_{cd}^k - x_{id}^k) \tag{3.3}$$

where p_{cd}^k is the d dimension of f-gbest in cycle k. From Equation 3.1, it can be observed that the f-gbest is characterized by a normal distribution $N(p_{gd}^k, \sigma)$, where σ represents the degree of uncertainty about the optimality of the gbest. In order to account for information received over time that reduces uncertainty about global best position, σ is modeled as some nonincreasing function of the number of cycles as given in Equation 3.2. For simplicity, $f(k)$ is defined as:

$$f(k) = \begin{cases} \sigma_{max} & cycles \prec \alpha \times max\_cycles \\ \sigma_{min} & otherwise \end{cases} \tag{3.4}$$

where σ_{max}, σ_{min}, and α are set as 0.15,0.0001, and 0.4 respectively.

The f-gbest should be distinguished from conventional turbulence or mutation operator, which applies a random perturbation to the particles. The function of f-gbest is to encourage the particles to explore a region beyond that defined by the search trajectory as illustrated in Figure 3.2. By considering the uncertainty associated with each gbest as a function of time, f-gbest also provides a simple and efficient exploration at the early stage when σ is large and encourages local fine-tuning at the latter stage when σ is small. Subsequently, this approach helps to reduce the likelihood of premature convergence and guides the search toward filling any gaps or discontinuities along the Pareto front for better tradeoff representation.

3.1.4 Synchronous Particle Local Search

The MOPSO is hybridized with a synchronous particle local search (SPLS) which performs directed local fine-tuning to improve the distribution of the solution set. The issues considered in the design of the LS operator include: 1) the selection of appropriate search direction; 2) the selection of appropriate particles for local optimization; and 3) the allocation of computational budget for LS.

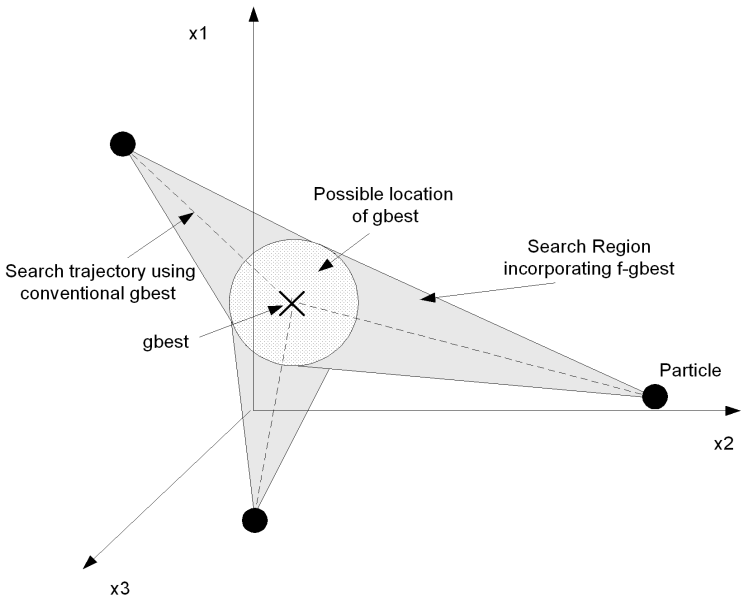

Figure 3.2: Search region of f-gbest

SPLS is performed in the vicinity of the particles, and the procedure is outlined in the following pseudo code.

Step 1) Select S_{LS} particles randomly from particle swarm.

Step 2) Select N_{LS} nondominated particles with the best niche count from the archive to a selection pool.

Step 3) Assign an arbitrary nondominated solution from the selection pool to each of the S_{LS} particles as gbest.

Step 4) Assign an arbitrary search space dimension for each of the S_{LS} particles.

Step 5) Assimilation: With the exception of the assigned dimension, update the position of S_{LS} particles in the decision space with the selected gbest position.

Step 6) Update the position of all S_{LS} assimilated particles along the pre-assigned dimension only.

The operation of SPLS is illustrated in Figure 3.3. The rationale of selecting the least crowded particles from the archive in Step 2 is to encourage the discovery of better solutions

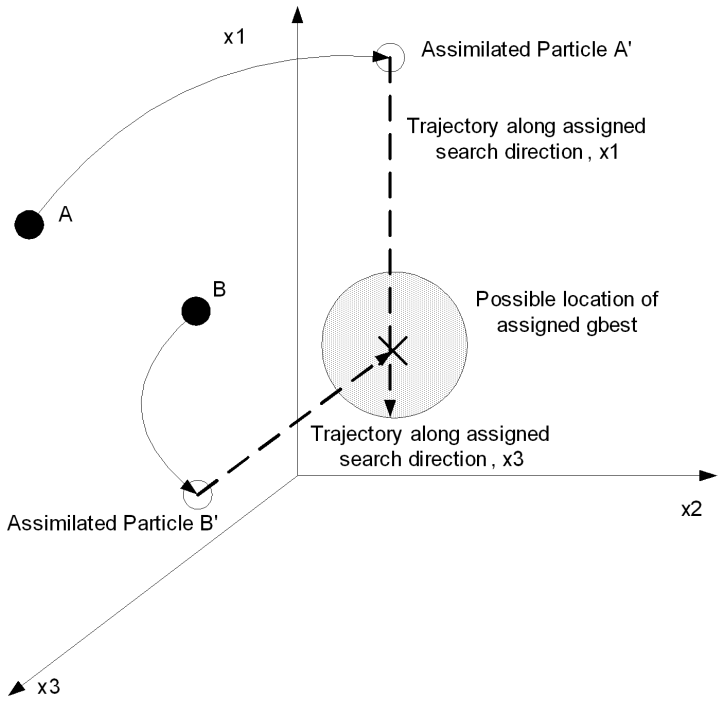

Figure 3.3: SPLS of assimilated particles along x1 and x3

that fill the gaps or discontinuities along the discovered Pareto front. Instead of selecting the particles directly from the archive for fine-tuning purposes, the assimilation process in Step 5 provide a means of integrating swarm and archive information. In contrast to random variation, Step 6 exploits knowledge about the possible location of gbest to probe the feasibility of the global best position along the assigned direction.

From the pseudo code, it is clear that the allocation of computational resource for LS is determined by S_{LS}. A high setting of S_{LS} allows the swarm to perform LS from different starting points, i.e., it enables the swarm to exploit along different directions. Likewise, a small setting of S_{LS} will restrict the LS operation. Therefore, we can control the computational effort spent on LS by appropriate settings of S_{LS}. In this work, the value of S_{LS} and N_{LS} is chosen as 20 and 2 respectively.

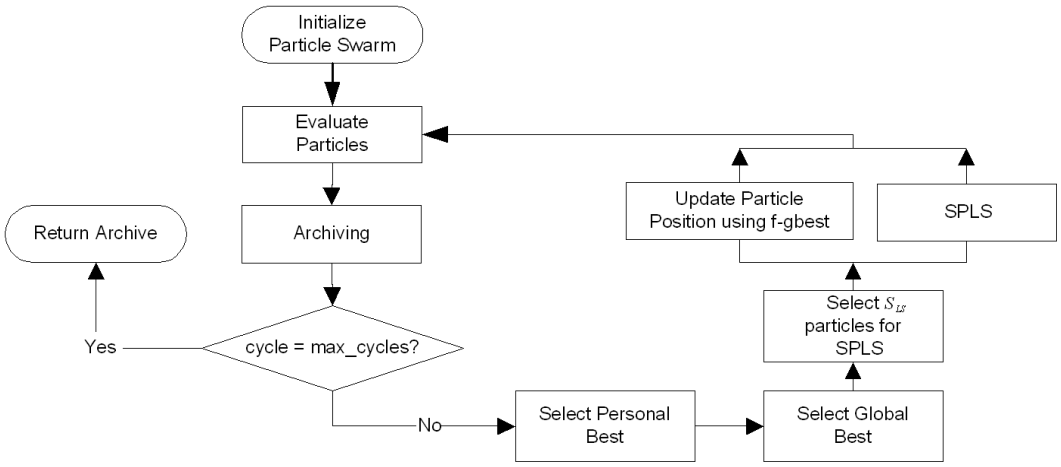

Figure 3.4: Flowchart of FMOPSO

3.1.5 Implementation

The flowchart of FMOPSO is shown in Figure 3.4. In every cycle, archiving is performed after the evaluation process of particles. Then, the pbest and gbest of each particle in the swarm are updated. Note that, the pbest of each particle will be updated only if a better pbest is found, and the f-gbest is implemented in place of conventional deterministic gbest. The update of particle position using f-gbest is performed concurrently with the SPLS. To maintain a balance between the exploration of "fly"and the exploitation of SPLS, the total number of evaluations is kept at the size of the particle swarm (N_{pop}) for every cycle. Specifically, S_{LS} particles will undergo the SPLS, while the rest of the $N_{pop} - S_{LS}$ particles in the swarm will be updated with the velocity calculated by Equation 3.3.

3.2 FMOPSO Performance and Examination of New Features

In this section, the performance of FMOPSO and the effects of new features (f-gbest and SPLS) are examined using benchmark problems. The tradeoffs generated by FMOPSO for the different benchmarks in one arbitrary run are shown in Figure 3.5 (10 000 evaluations for

ZDT1, FON, KUR and POL, 50 000 evaluations for ZDT4, 20 000 evaluations for ZDT6). It can be seen that FMOPSO can converge to the true Pareto front and evolve a diverse solution set for all benchmark problems although the solutions for ZDT1 and FON are not evenly distributed along the true Pareto front. Since FMOPSO has found the near-optimal Pareto front and covered the full extent for ZDT1 and FON, the SPLS will help find more nondominated solutions on the gaps or discontinuities along the Pareto front and the dynamic niche sharing scheme will improve spacing metric if given more cycles. Figure 3.5 shows that FMOPSO can overcome the difficulties presented by these benchmarks.

3.2.1 Examination of New Features

In this section, four versions of the algorithm (standard MOPSO, MOPSO with f-gbest only, MOPSO with SPLS only, FMOPSO) are compared to illustrate the individual and combined effects of the proposed features.

Effects of Fuzzy Update: MOPSO incorporating SPLS only is compared against FMOPSO to determine whether the fuzzy update really played an important role in FMOPSO algorithm. The evolutionary trajectories of FMOPSO (fuzzy update) and SPLS only (conventional update) on ZDT1, ZDT4 and ZDT6 are plotted in Figures 3.6-3.8. It can be observed that fuzzy update allows the discovery of a well-distributed and diverse solution set without compromising convergence speed. On the other hand, MOPSO with SPLS only tends to converge prematurely, despite its slower convergence speed.

Further information can be extracted by comparing the explored objective space versus cycle number of FMOPSO and SPLS only on ZDT1 as shown in Figure 3.9. It can be noted that the explored objective space for fuzzy update is more extensive as compared to the standard update. By taking into account the uncertainty of the information gathered about the gbest position, the fuzzy updating strategy actually prevents the evolving particles to converge upon similar regions. Therefore, in the searching process, the explored space is extended.

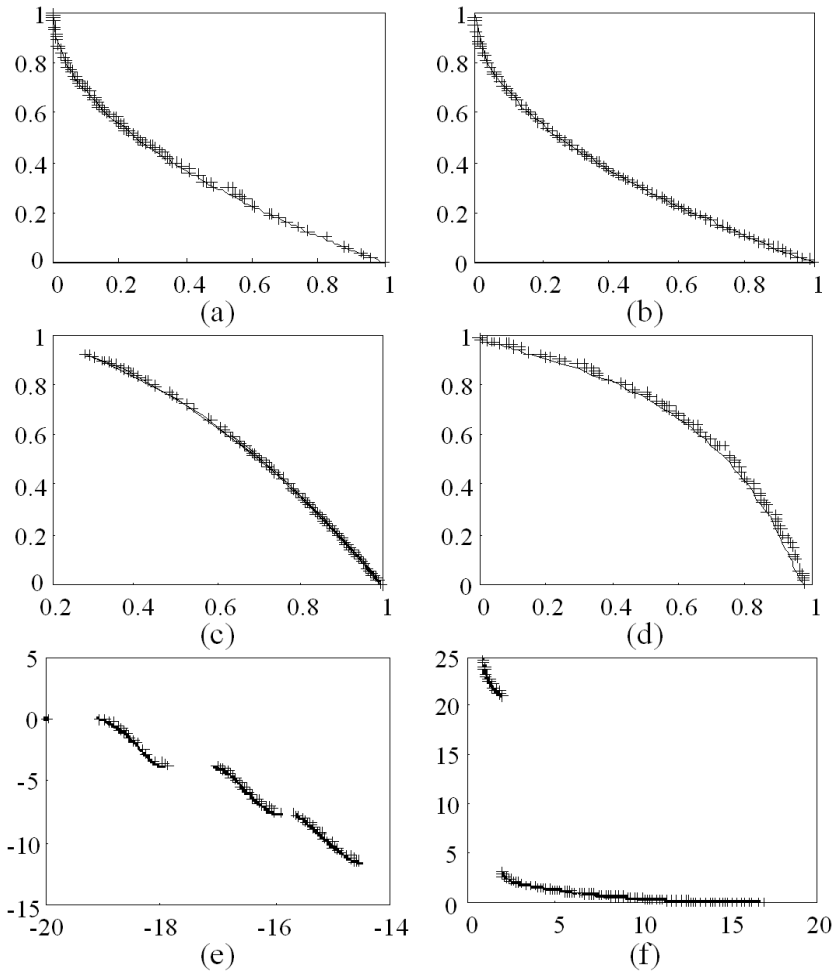

Figure 3.5: Evolved tradeoffs by FMOPSO for a) ZDT1, b) ZDT4, c) ZDT6, d) FON, e) KUR and f) POL

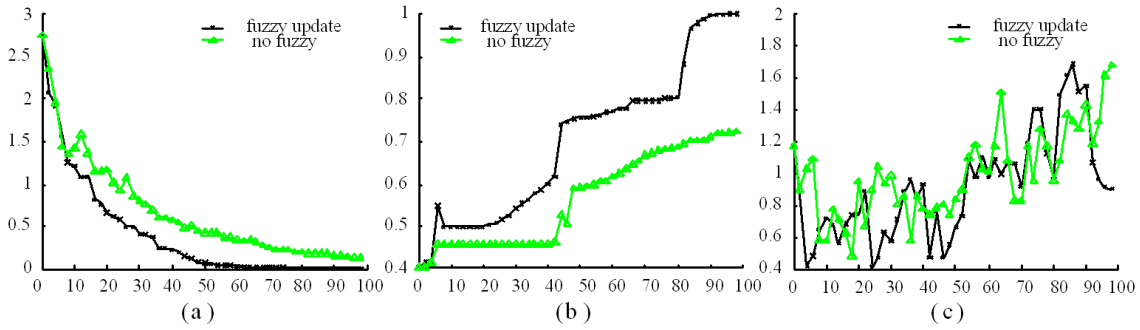

Figure 3.6: Evolutionary trajectories in a) GD, b) MS, and c) S for ZDT1

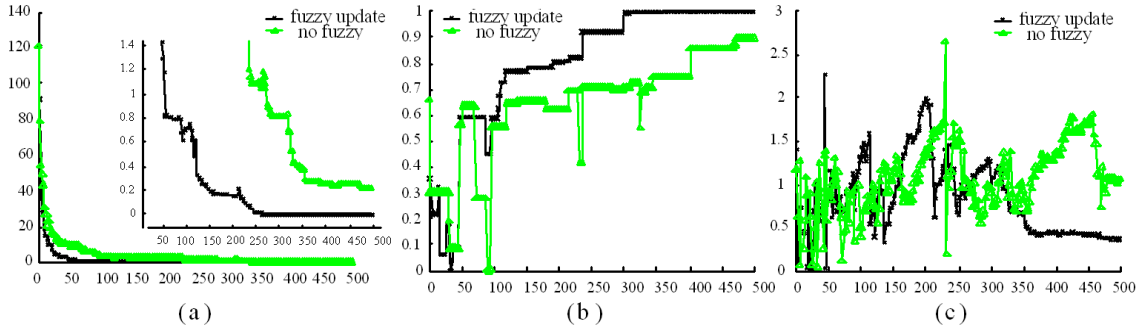

Figure 3.7: Evolutionary trajectories in a) GD, b) MS, and c) S for ZDT4

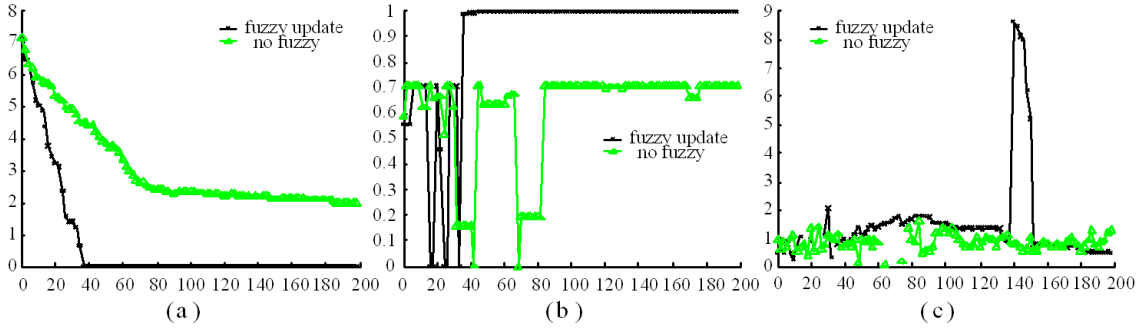

Figure 3.8: Evolutionary trajectories in a) GD, b) MS, and c) S for ZDT6

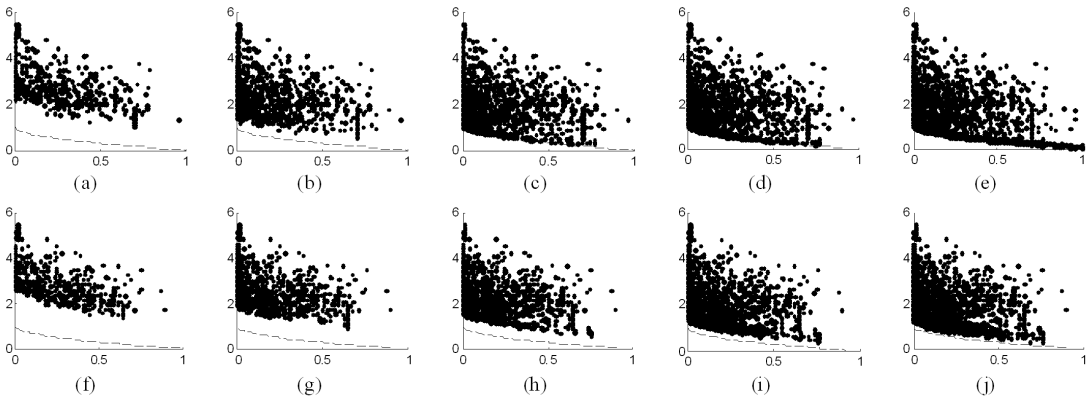

Figure 3.9: Explored objective space FMOPSO at cycle a)20, b)40, c)60, d)80, e)100 and SPLS only at cycle f)20, g)40, h)60, i)80, j)100 for ZDT1

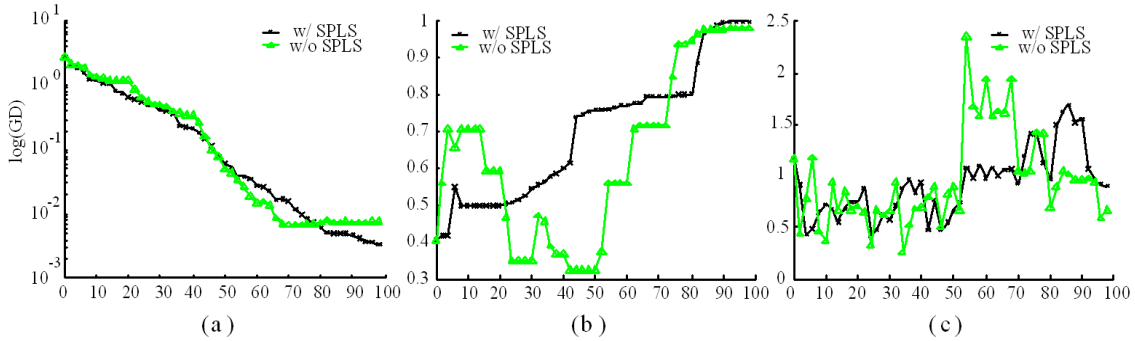

Figure 3.10: Evolutionary trajectories in a) GD, b) MS, and c) S for ZDT1

Effects of Synchronous Particle Local Search: MOPSO with f-gbest only is compared against FMOPSO to determine the influence of SPLS on algorithm performance. The evolutionary trajectories of FMOPSO and f-gbest only on ZDT1, ZDT4 and ZDT6 are plotted in Figure 3.10-3.12.

Figure 3.10(a) and Figure 3.12(a) ascertain the exploitative nature of SPLS as reflected by the fast convergence speed relative to that without local search. Figure 3.11 shows that the inclusion of f-gbest only cannot find the true Pareto front of ZDT4 in 50 000 evaluations. From the evolutionary trajectories, it can be observed that f-gbest only can also lead the search to come closer and closer to the true Pareto front as shown by the decreasing GD, but the improvement is too slow for ZDT4 without the guide of SPLS. Without the balance between the exploration of fuzzy update and the exploitation of SPLS, the inclusion of f-gbest only cannot deal with the multi-modality nature of ZDT4 effectively. The discontinuity in the spacing metric of the f-gbest only is because in those cycles there are less than three nondominated solutions in the archive. Furthermore, from the evolutionary trajectories in terms of MS and S for the different problems, it can be observed that SPLS provides a means of steadily improving diversity and distribution.

Performance Comparison of all four versions: The results of standard MOPSO, MOPSO with f-gbest only, MOPSO with SPLS only, and FMOPSO with respect to the different performance metric of ZDT1, ZDT4 and ZDT6 are summarized in Table 3.1, Table 3.2 and

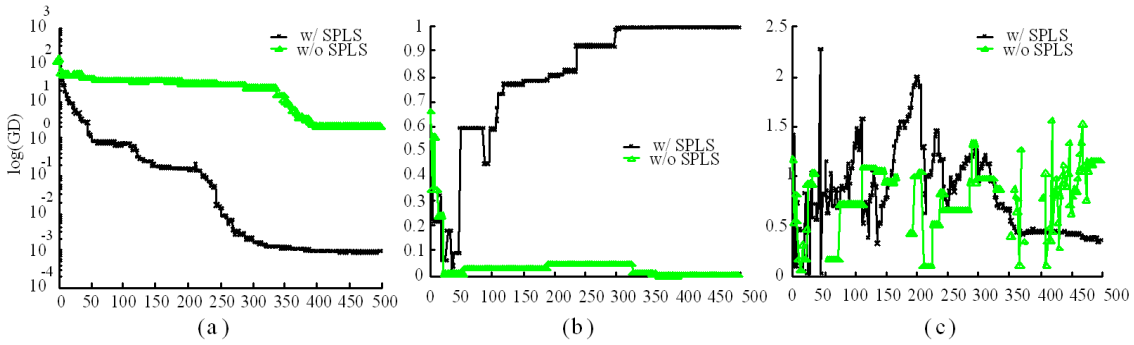

Figure 3.11: Evolutionary trajectories in a) GD, b) MS, and c) S for ZDT4

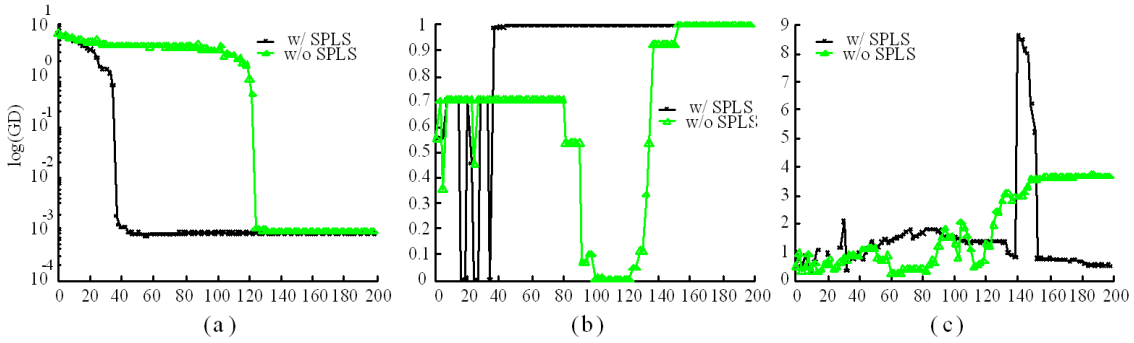

Figure 3.12: Evolutionary trajectories in a) GD, b) MS, and c) S for ZDT6

Table 3.3, respectively. When there are less than 3 solutions in the archive, NaN is shown as the computation of spacing metric requires more than 3 solutions.

It can be observed from Tables 3.2 and Table 3.3 that the incorporation of SPLS alone can greatly improve the performance of standard MOPSO in terms of proximity, diversity, and distribution for ZDT4 and ZDT6. The good performance of spacing also shows that the local search ability of SPLS helps the discovery of nondominated solutions on the gaps or discontinuities along the Pareto front. In Table 3.1, even though SPLS only cannot find the true Pareto front for ZDT1, it can still maintain a relatively good spacing metric.

From Table 3.1 and Table 3.3, it can be seen that the implementation of f-gbest alone helps MOPSO to discover the near-optimal Pareto front for ZDT1 and ZDT6. However, the f-gbest alone cannot guarantee a good performance on spacing metric. This is expected

Table 3.1: Performance of different features for ZDT1

		Standard	f-gbest-only	SPLS-only	FMOPSO
GD	Mean	0.3179	0.0144	0.1260	0.0046
	Median	0.2947	0.0107	0.1137	0.0032
	Min	0.1239	0.0031	0.0114	0.0014
	Max	0.7973	0.0428	0.2842	0.0180
	Std	0.1406	0.0106	0.0597	0.0041
S	Mean	1.0582	0.7700	0.9740	0.8032
	Median	0.9357	0.7459	0.8827	0.7018
	Min	0.6542	0.4723	0.5856	0.3543
	Max	2.4528	1.2692	1.7858	1.9060
	Std	0.3842	0.1996	0.2775	0.3237
MS	Mean	0.6703	0.9775	0.8258	0.9977
	Median	0.6677	0.9815	0.8437	1.0000
	Min	0.4465	0.8950	0.5938	0.9829
	Max	0.8961	0.9959	0.9579	1.0000
	Std	0.1278	0.0184	0.1003	0.0045

since the enhancement of global search capability provided by f-gbest is not aimed for the uniform distribution of solutions along the Pareto front. On the other hand, the combination of f-gbest and SPLS allows the discovery of a well-distributed and diverse solution set for ZDT1, ZDT4 and ZDT6 without compromising the convergence speed of the algorithm.

3.3 Comparative Study

In this section, the performance of FMOPSO is compared to five existing MO algorithms, including CMOPSO [26], SMOPSO [130], IMOEA [198], NSGAII [35], and SPEA2 [224]. All the algorithms were implemented in C++ and the simulations were performed on an Intel Pentium IV 2.8-GHz personal computer. Thirty simulation runs were performed for each algorithm on each test problem in order to study the statistical performance. A random

Table 3.2: Performance of different features for ZDT4

		Standard	f-gbest-only	SPLS-only	FMOPSO
GD	Mean	6.8211	7.8011	0.2326	0.0009
	Median	6.6020	5.3704	0.1245	0.0008
	Min	2.7099	1.2445	0.00073	0.00066
	Max	14.6854	26.7440	1.1683	0.0023
	Std	2.7790	6.8388	0.3090	0.0003
S	Mean	NaN	NaN	1.1347	0.3255
	Median	NaN	NaN	1.1103	0.3313
	Min	0.6684	0.6032	0.3012	0.2264
	Max	2.6588	3.1383	1.7716	0.4541
	Std	NaN	NaN	0.3304	0.0530
MS	Mean	0.0049	0.0038	0.7499	0.9999
	Median	0.0006	0.0000	0.9030	1.0000
	Min	0	0	0.1815	0.9989
	Max	0.1057	0.0727	1.0000	1.0000
	Std	0.0193	0.0142	0.2732	0.0002

initial population was created for each of the thirty runs on each test problem. Note that the number of evaluations is set to be relatively small to examine the convergence of the algorithms compared. The parameter settings and indices of the different algorithms are shown in Table 3.4 and Table 3.5, respectively.

ZDT1: The tradeoffs generated by the different algorithms are shown in Figure 3.13(a)-(f). Figure 3.14 summarizes statistical performance of the different algorithms. The distribution of the results is represented in box plot format [19]. The box plot format had been applied to visualize the distribution of the simulated results efficiently [226]. Each box plot represents the distribution of the 30 results, where a thick horizontal line within the box encodes the median while the upper and lower ends denote the upper and lower quartile respectively. Dashed Appendages illustrate the spread and shape of distribution and dots

Table 3.3: Performance of different features for ZDT6

		Standard	f-gbest-only	SPLS-only	FMOPSO
GD	Mean	2.0468	0.0008234	0.0671	0.0008121
	Median	2.6877	0.0008213	0.0008120	0.0008171
	Min	0.0007459	0.0006607	0.0005993	0.0006203
	Max	4.5743	0.0010	1.9997	0.0009
	Std	1.8385	0.00008347	0.3633	0.00008171
S	Mean	NaN	4.4864	NaN	0.4406
	Median	NaN	4.5363	NaN	0.4394
	Min	0.6435	1.6784	0.2692	0.2636
	Max	6.3745	7.6267	6.0548	0.6302
	Std	NaN	1.2937	NaN	0.0853
MS	Mean	0.5327	0.9138	0.9393	0.9992
	Median	0.5893	0.9995	0.9992	0.9992
	Min	0	0.0271	0	0.9992
	Max	0.9995	0.9995	0.9992	0.9992
	Std	0.3726	0.2352	0.2055	0.0000

represent outside values. The evolutionary trajectories of the different performance metrics are plotted in Figure 3.15.

From Figure 3.13(a)-(f), it can be observed that, except FMOPSO, other algorithms still have many solutions that are located far away from the true Pareto front. From Figure 3.14, it can be seen that the average performance of FMOPSO is the best among the six algorithms adopted. In addition, FMOPSO is also able to evolve a diverse solution set as evident from Figure 3.14(b) and (c). At the same time, it is noted that the solutions are not evenly distributed along the true Pareto front as illustrated by a relatively large spacing metric. Since FMOPSO has covered the full extent of the true Pareto front as illustrated by the high value of MS and the evaluation number is set to be relatively small for ZDT1 (only 10 000 evaluations), it will improve spacing metric if given more evaluations for the

Table 3.4: Parameter settings of the different algorithms

Parameter	Settings
Population	Population size 100; Archive (or secondary population) size 100.
Representation	15 bits for each variable in IMOEA, NSGAII and SPEA2.
	Real number representation in FMOPSO, CMOPSO, and SMOPSO.
Selection	Binary tournament selection
Crossover rate	0.8 in IMOEA, NSGAII, and SPEA2
Crossover method	Uniform crossover in IMOEA, NSGAII, and SPEA2
Mutation rate	$\frac{1}{chromosome\_length}$ for ZDT1, ZDT4 and ZDT6;
	$\frac{1}{bit\_number\_per\_variable}$ for FON and KUR;
Mutation method	Bit-flip mutation for IMOEA, NSGAII and SPEA2.
	Adaptive mutation for CMOPSO.
	Turbulence operator for SMOPSO.
Grid Division	30 for CMOPSO
Evaluations	10 000 for ZDT1, FON, KUR and POL; 50 000 for ZDT4; 20 000 for ZDT6.

Table 3.5: Indices of the different algorithms

Index	1	2	3	4	5	6
Algorithms	FMOPSO	CMOPSO	SMOPSO	IMOEA	NSGAII	SPEA2

incorporated SPLS and dynamic niche sharing scheme.

A look at the evolutionary trajectories can reveal more information about the performance of different algorithms. From Figure 3.15, it can be noted that FMOPSO has the faster convergence speed while the progress of other algorithms stagnates after about 50 cycles. However, it can be observed that NSGA II and SPEA2 are able to evolve a diverse solution set rather quickly. On the other hand, the incorporation of the proposed features

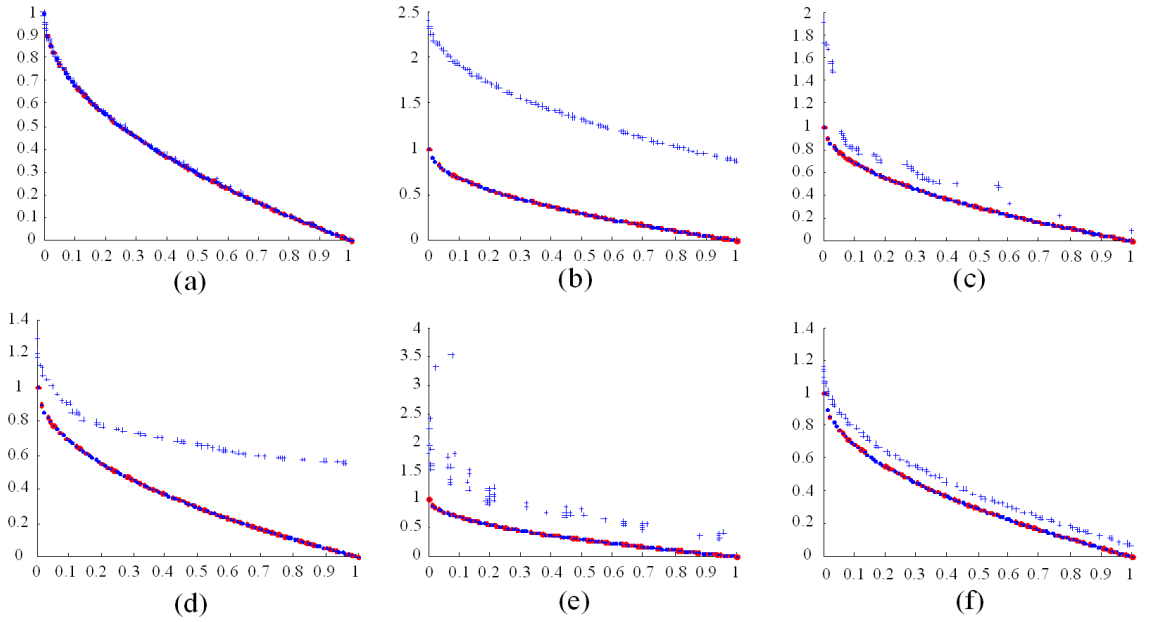

Figure 3.13: Evolved tradeoffs by a) FMOPSO, b) CMOPSO, c) SMOPSO, d) IMOEA, e) NSGA II, and f) SPEA2 for ZDT1 (PF^A + PF^* •)

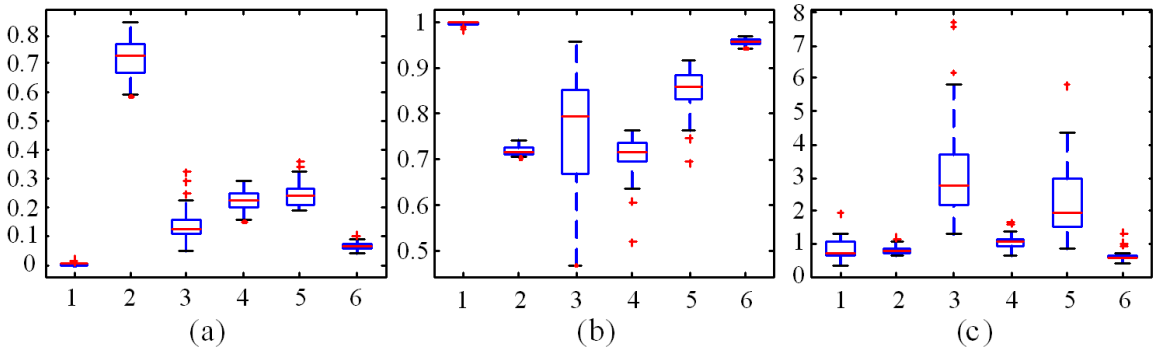

Figure 3.14: Algorithm performance in a) GD, b) MS, and c) S for ZDT1

allowed FMOPSO to improve steadily and eventually discovering a more diverse solution set.

ZDT4: The tradeoffs generated by the different algorithms are shown in Figure 3.16(a)-(f). Figure 3.17 summarizes statistical performance of the different algorithms. The evolutionary trajectories of the different performance metrics are plotted in Figure 3.18.

From Figure 3.16 and 3.17, it can be seen that SMOPSO, CMOPSO, IMOEA, NSGAII

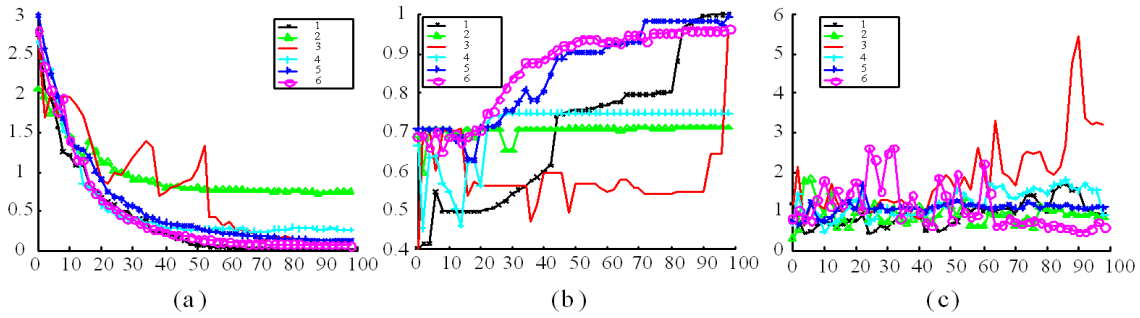

Figure 3.15: Evolutionary trajectories in a) GD, b) MS, and c) S for ZDT1

and SPEA2 are only able to locate one of the local Pareto fronts. Furthermore, it can be observed that SMOPSO and CMOPSO are unable to evolve a diverse solution set consistently. On the other hand, from Figure 3.16(a) and 3.17(a), it is noted that FMOPSO is able to escape the local optima of ZDT4 consistently as reflected by the good results obtained in GD. FMOPSO is also able to evolve a diverse and well-distributed set within 50 000 evaluations, resulting in high values of MS and low values of S. ZDT4 proved to be the most difficult problem faced by the algorithms since no algorithms adopted here, except FMOPSO, is able to deal with multi-modality effectively.

By comparing Figure 3.15 and 3.18, the multimodality nature of ZDT4 is apparent from the fact that the algorithms progress in "hops"rather than in a smooth trajectory experienced for ZDT1. The evolutionary trajectories also demonstrate the varying degree of success experienced by the different algorithms in dealing with local optimality. In general, most of the algorithms failed to escape local optima within 50 000 evaluations. The great fluctuation in spacing is an indication of the discontinuity present in the evolved Pareto front as the algorithms progress towards the true Pareto front.

ZDT6: The tradeoffs generated by the different algorithms are shown in Figure 3.19(a)-(f). Figure 3.20 summarizes statistical performance of the different algorithms. The evolutionary trajectories of the different performance metrics are plotted in Figure 3.21.

From Figure 3.19-3.20, it is clear that CMOPSO, NSGAII and SPEA2 can not find true Pareto front within 20 000 evaluations. Although SMOPSO and IMOEA are able to find the

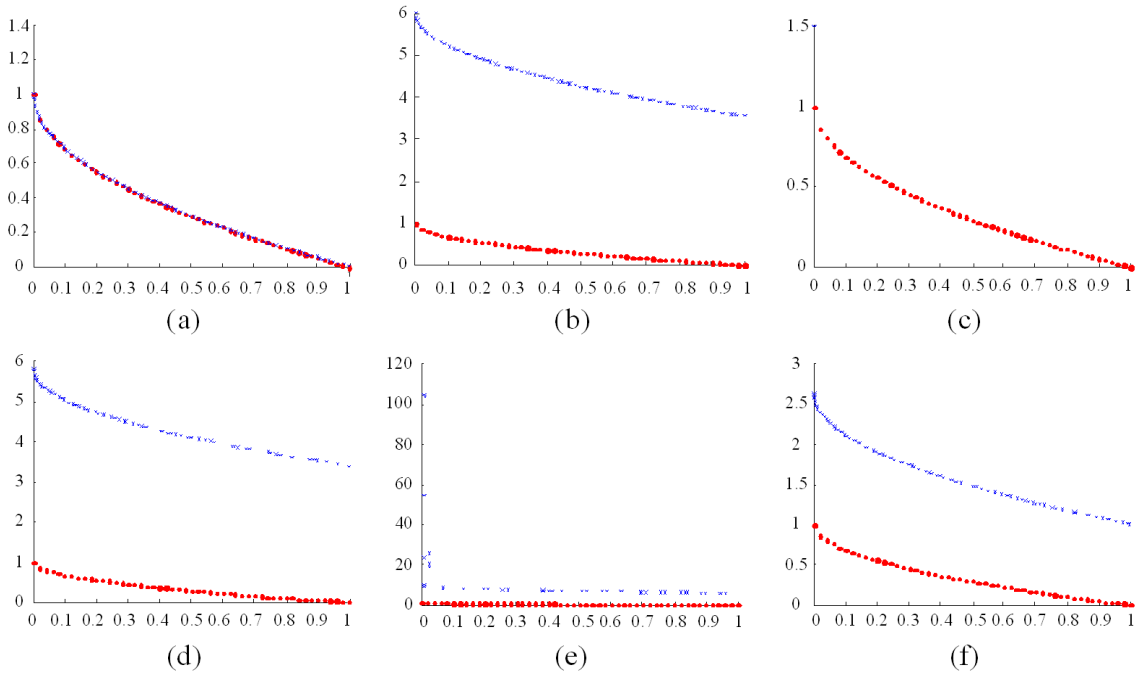

Figure 3.16: Evolved tradeoffs by a) FMOPSO, b) CMOPSO, c) SMOPSO, d) IMOEA, e) NSGA II, and f) SPEA2 for ZDT4 ($PF^A \times PF^*$ •)

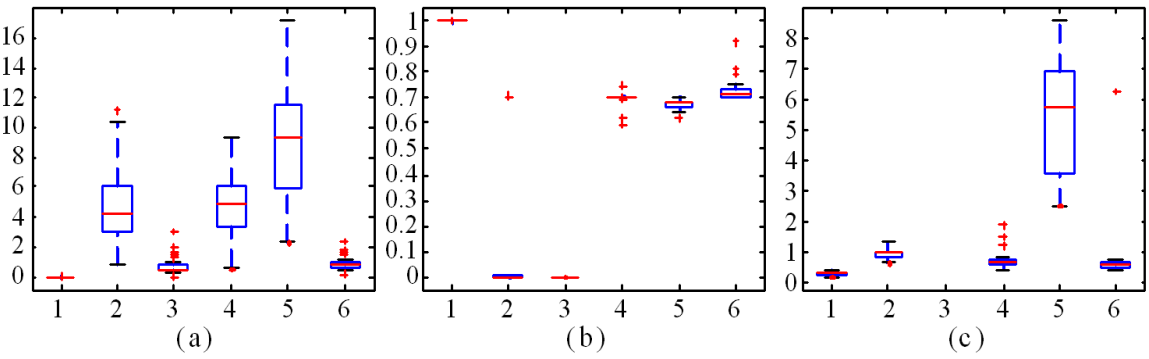

Figure 3.17: Algorithm performance in a) GD, b) MS, and c) S for ZDT4

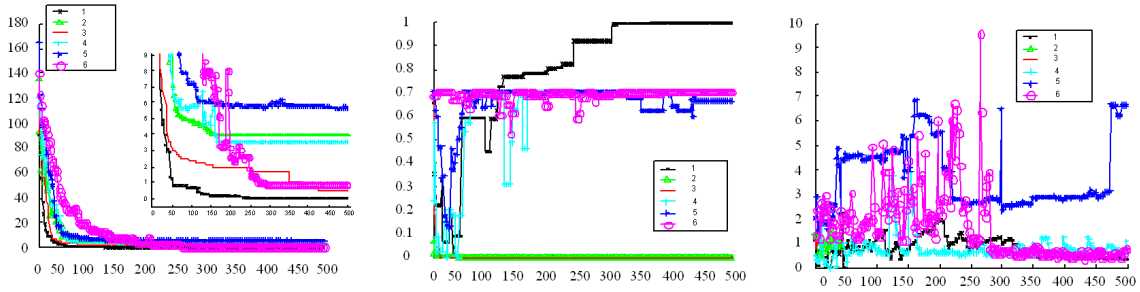

Figure 3.18: Evolutionary trajectories in a) GD, b) MS, and c) S for ZDT4

true Pareto front, they failed to evolve a diverse and well-distributed solution set as shown in Figure 3.20(b) and (c). Overall, FMOPSO is able to evolve a diverse and well-distributed near-optimal Pareto front for ZDT6 within 20 000 evaluations. As shown in Figure 3.21, at the end of 20 000 evaluations, CMOPSO, NSGAII and SPEA2 are still exploring the decision space. If given larger number of evaluations, these three algorithms may be able to evolve better solutions.

FON: The tradeoffs generated by the different algorithms are shown in Figure 3.22(a)-(f). Figure 3.23 summarizes statistical performance of the different algorithms. The evolutionary trajectories of the different performance metrics are plotted in Figure 3.24.

From Figure 3.22 and 3.23, it can be observed that most of the algorithms are able to find at least part of the true Pareto front for FON. The PSO paradigm appears to have a slight edge in dealing with the nonlinear tradeoff curve of FON, i.e., the three PSO-based algorithms outperformed other algorithms consistently on generational distance. It can be observed that FMOPSO demonstrated the best results in terms of MS and S. In addition, it also showed competitive performance in terms of GD. On the other hand, SMOPSO and CMOPSO failed to evolve a diverse set of solution as compared to FMOPSO. This could be due to the fast convergence speed of SMOPSO and CMOPSO, which may result in the loss of the required diversity to cover the entire final Pareto front.

From Figure 3.24, it can be observed that the diversity of the solution set stops improving with the convergence of the algorithm. It will be interesting to note that algorithms with

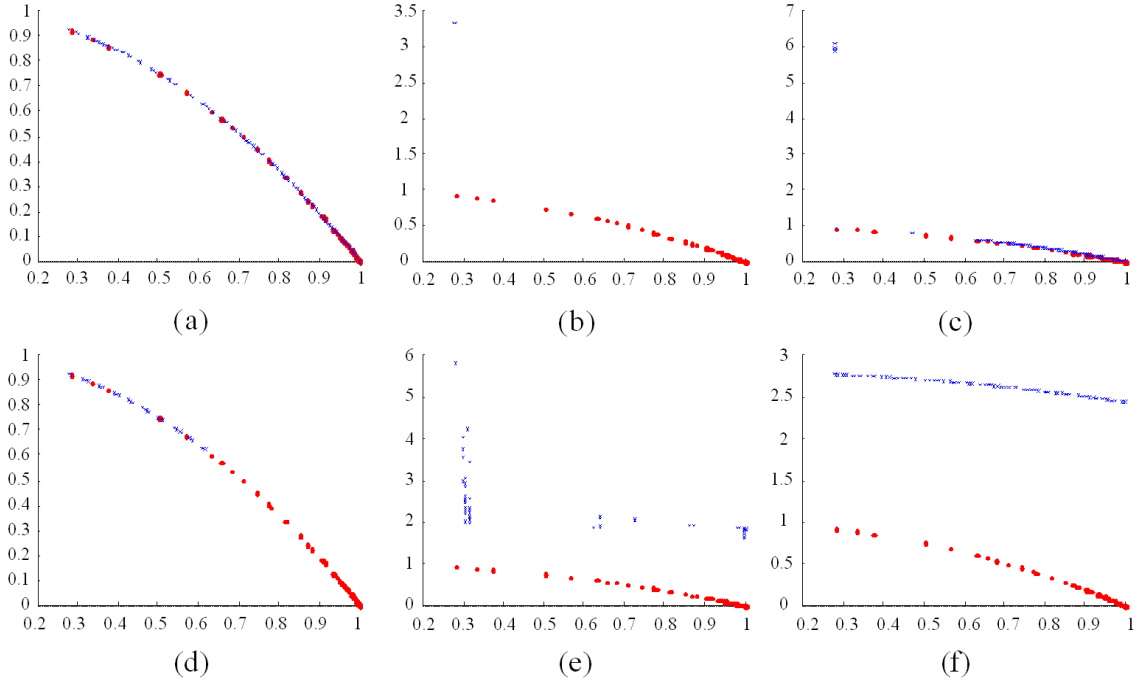

Figure 3.19: Evolved tradeoffs by a) FMOPSO, b) CMOPSO, c) SMOPSO, d) IMOEA, e) NSGA II, and f) SPEA2 for ZDT6 ($PF^A \times PF^*$ •)

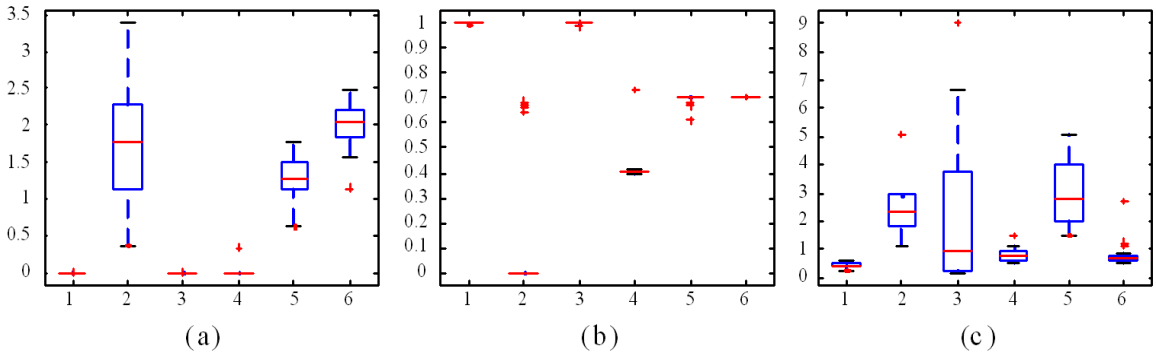

Figure 3.20: Algorithm performance in a) GD, b) MS, and c) S for ZDT6

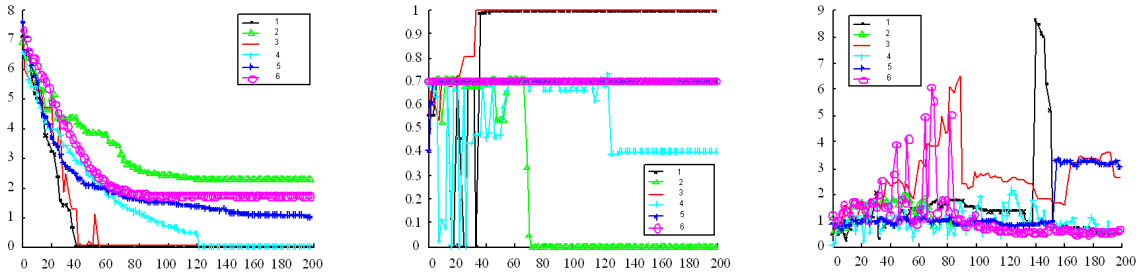

Figure 3.21: Evolutionary trajectories in a) GD, b) MS, and c) S for ZDT6

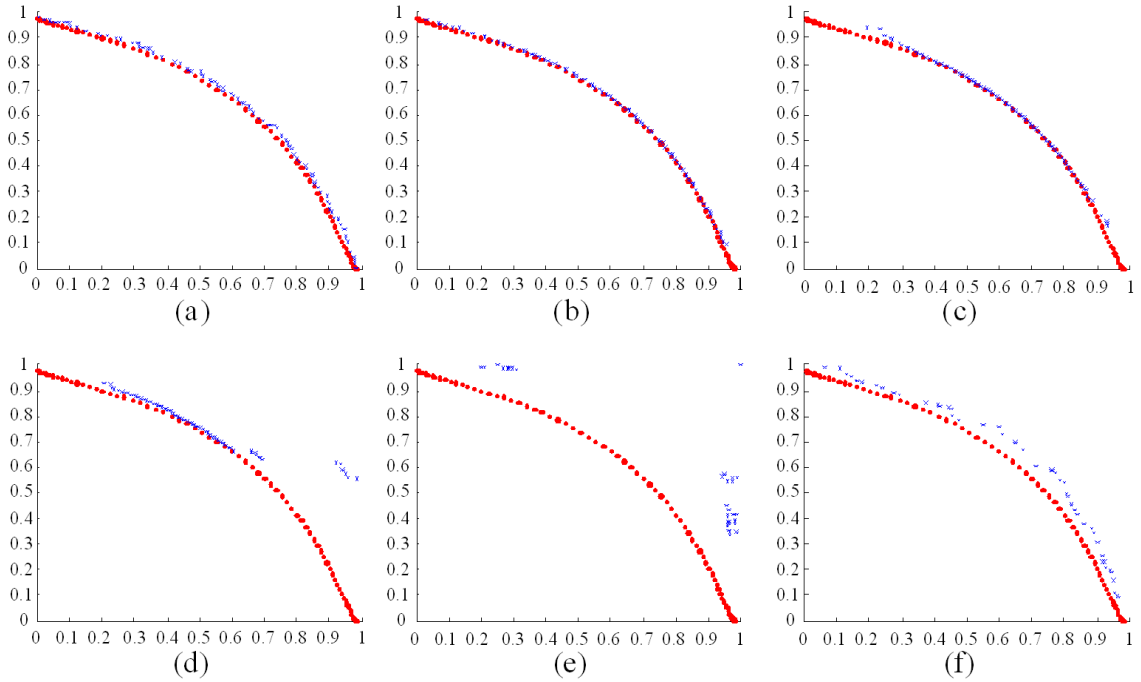

Figure 3.22: Evolved tradeoffs by a) FMOPSO, b) CMOPSO, c) SMOPSO, d) IMOEA, e) NSGA II, and f) SPEA2 for FON ($PF^A \times PF^*$ •)

slower convergence speed such as NSGA II and SPEA2 in this instance tend to evolve a more diverse solution set for FON. This is probably due to the efforts spent on diversity preservation. At the same time, the proposed features allowed FMOPSO to evolve a diverse solution set without the need to compromise convergence speed.

KUR: The tradeoffs generated by the different algorithms are shown in Figure 3.25(a)-

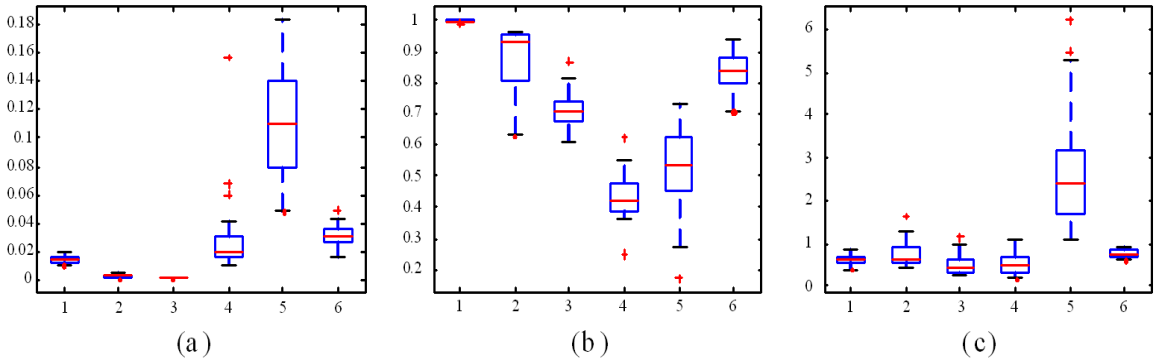

Figure 3.23: Algorithm performance in a) GD, b) MS, and c) S for FON

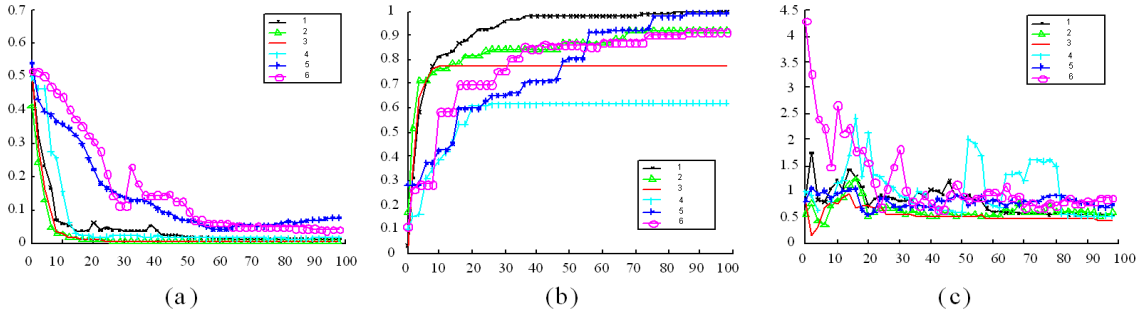

Figure 3.24: Evolutionary trajectories in a) GD, b) MS, and c) S for FON

(f). Figure 3.26 summarizes statistical performance of the different algorithms. The evolutionary trajectories of the different performance metrics are plotted in Figure 3.27.

The disconnection of the Pareto front of KUR seems not a very big problem for the algorithms adopted here as shown in Figure 3.25-3.27. All algorithms are able to find the near-optimal Pareto front. While in MS, FMOPSO is still the best. In addition, FMOPSO also showed competitive performance in terms of convergence and distribution.

It should be noted that, in Figure 3.27(a), the GD metric of NSGAII begin to increase after 30 generations instead of decreasing. One possible reason is that in the beginning NSGAII converge to some parts of true Pareto front. While it explored other regions of the true Pareto front, some new nondominated solutions are found. These new nondominated solutions are not so close to the true Pareto front although they are better distributed along the true Pareto front.

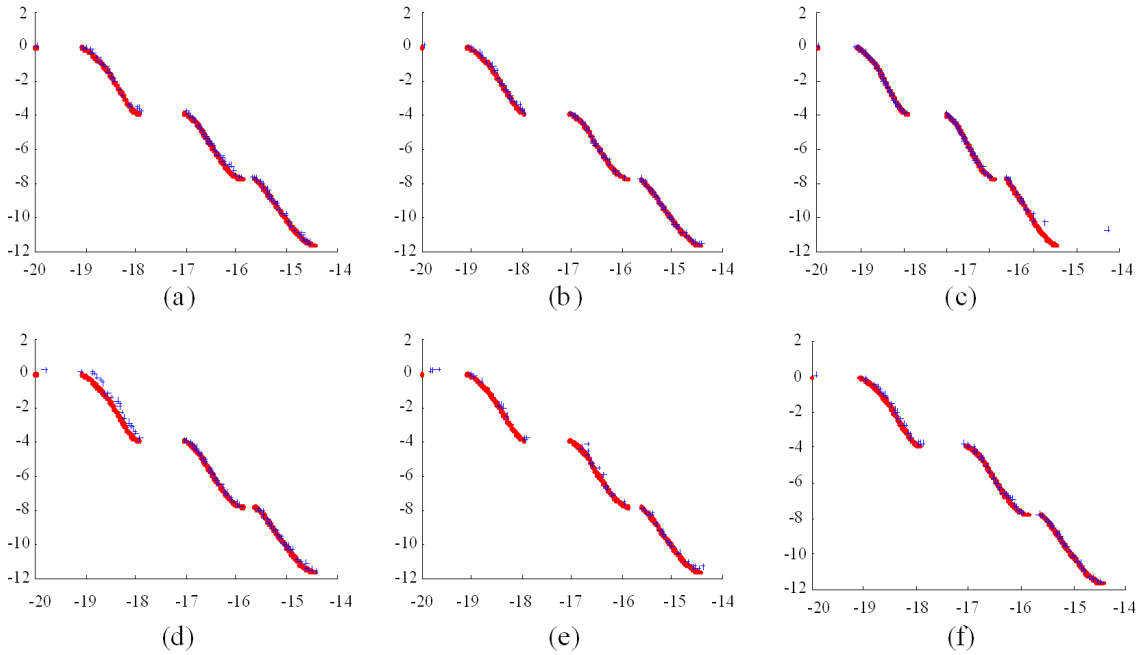

Figure 3.25: Evolved tradeoffs by a) FMOPSO, b) CMOPSO, c) SMOPSO, d) IMOEA, e) NSGA II, and f) SPEA2 for KUR (PF^A + PF^* •)

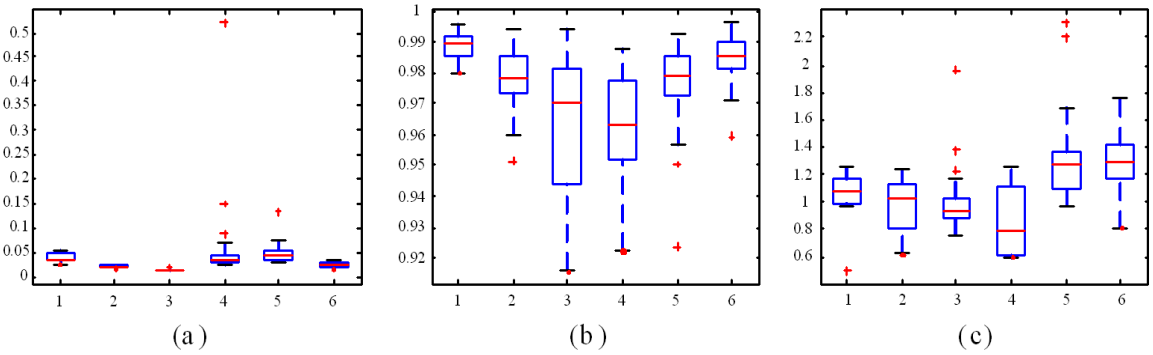

Figure 3.26: Algorithm performance in a) GD, b) MS, and c) S for KUR

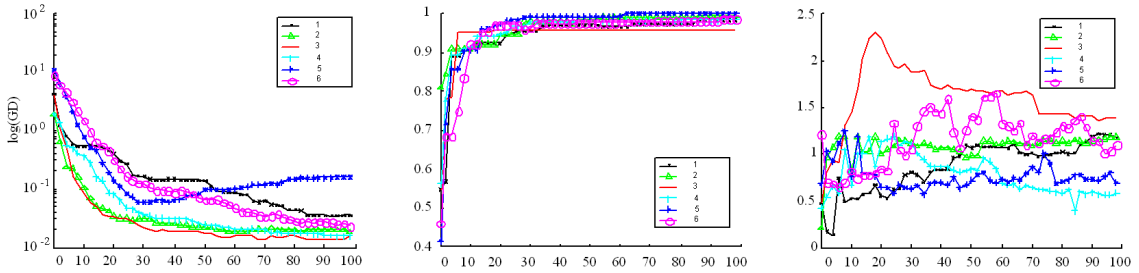

Figure 3.27: Evolutionary trajectories in a) GD, b) MS, and c) S for KUR

POL: The tradeoffs generated by the different algorithms are shown in Figure 3.28(a)-(f). Figure 3.29 summarizes statistical performance of the different algorithms. The evolutionary trajectories of the different performance metrics are plotted in Figure 3.30.

From Figure 3.28 and 3.29, it can be noted that NSGAII has the worst results in terms of convergence, diversity and distribution. On the other hand, FMOPSO, CMOPSO, SMOPSO and IMOEA showed competitive performance. PSO paradigm has a slight edge in the aspect of convergence. However, it should be noted that the performance of CMOPSO and SMOPSO in the aspect of S and MS respectively are not as good as IMOEA and FMOPSO.

It can be seen from Figure 3.30 that all algorithms show similar trend in solving POL problem. In the middle of the searching process, most algorithms fluctuate greatly in spacing metric, but they will stabilize at around 100 cycles. The stabilization of three metrics is an indication that the algorithms have converged.

In general, for all test problems, FMOPSO responded well to the challenges of the different difficulties. The FMOPSO performed consistently well in the distribution of solutions along the Pareto front. This is even so for the test problems of ZDT6 that is designed to challenge the algorithms ability to maintain the Pareto front. FMOPSO demonstrates its ability to converge upon true Pareto front regardless of difficulties such as discontinuities, convexities and non-uniformities. It also showed no problems in coping with local traps and this is reflected by its performance in the test problem ZDT4 (21^9 local Pareto front).

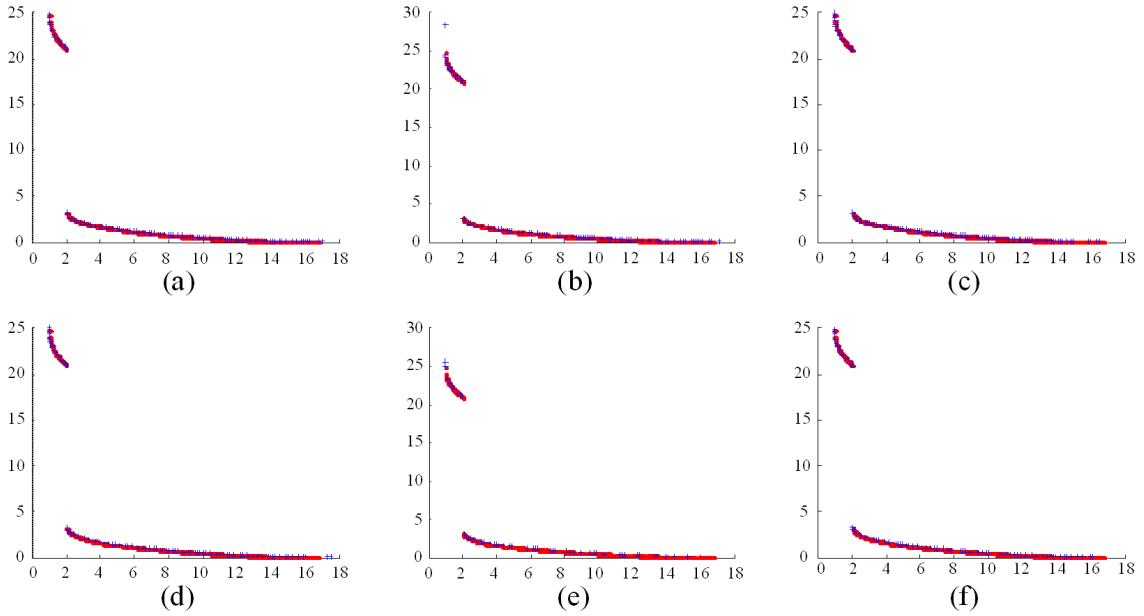

Figure 3.28: Evolved tradeoffs by a) FMOPSO, b) CMOPSO, c) SMOPSO, d) IMOEA, e) NSGA II, and f) SPEA2 for POL (PF^A + PF* •)

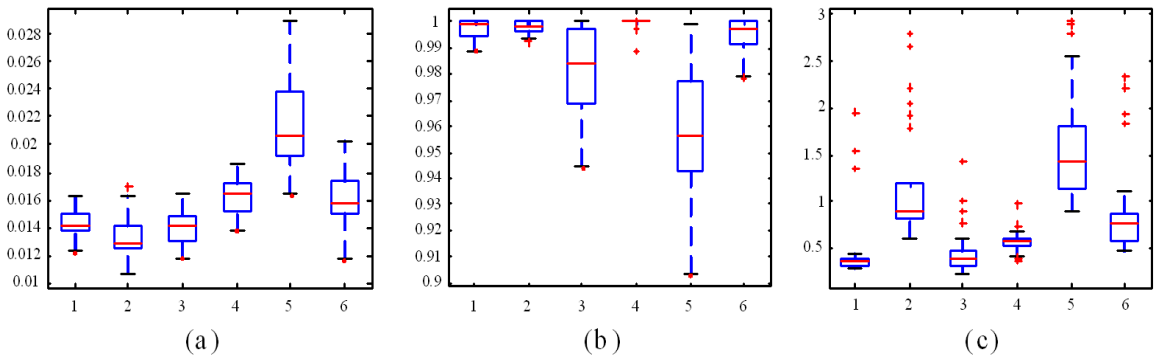

Figure 3.29: Algorithm performance in a) GD, b) MS, and c) S for POL

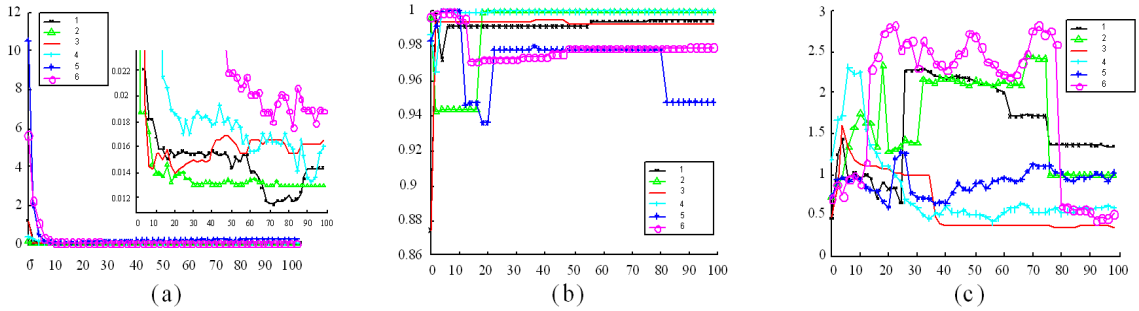

Figure 3.30: Evolutionary trajectories in a) GD, b) MS, and c) S for POL

3.4 Conclusion

A new memetic algorithm has been designed within the context of multi-objective particle swarm optimization. In particular, two new features in the form of f-gbest and SPLS have been proposed. The SPLS performs directed local fine-tuning, which helps to discover a well-distributed Pareto front. The f-gbest models the uncertainty associated with the optimality of gbest, thus helping the algorithm to avoid undesirable premature convergence. The proposed features have been examined to show their individual and combined effects in MO optimization. The comparative study showed that the proposed FMOPSO produced solution sets that are highly competitive in terms of convergence, diversity and distribution.

Chapter 4

A Competitive and Cooperative Co-evolutionary Approach to Multi-objective Particle Swarm Optimization Algorithm Design

Although PSO has proven to be successful in various fields, researchers are facing the increasing challenge of problem complexity in today's applications. The computational cost increases with the size and complexity of the MO problem and the large number of function evaluations involved in the optimization process may be cost prohibitive. The necessity to improve MOPSO's efficacy and efficiency becomes more acute especially in high-dimensional problems.

A direct method to overcome the exponential increase in problem difficulty is to segregate the search space into smaller subspaces and conduct the overall optimization process over smaller regions. In this regard, the notion of coevolution is very attractive. The coevolutionary paradigm, inspired by the reciprocal evolutionary change driven by the cooperative [152] or competitive interaction [160] between different species, has been extended to MO optimization recently [27] [79] [117] [121] [191]. The aim is to fulfill the MO optimization goals of attaining a good and diverse solution set with enhanced effectiveness and efficiency. To this end, the various implementations of coevolutionary structures in

multi-objective evolutionary optimization have been successful. As a specific instance, Tan et al [191] demonstrated that high convergence speed can be achieved while maintaining a good diversity of solutions by the incorporation of coevolutionary models.

Nevertheless, despite the different co-evolutionary models proposed for multi-objective evolutionary optimization, there have not been any genuine attempts to extend it into multi-objective particle swarm optimization. The closest attempt is a cooperative co-evolutionary model for single objective optimization proposed by van den Bergh and Engelbrecht [205]. Also, it should be highlighted that current co-evolutionary approaches usually consider cooperation and competition interaction between species separately. This is in stark contrast to nature where there is competition even in the veneer of seemingly perfect plant-pollinator co-evolution. Different species of bees will compete for nectar [169] and different species of flowers will compete to attract more bees. As such, this chapter adopts a competitive and cooperative co-evolution model to mimic the interplay of competition and cooperation among different species in nature and combine it with MOPSO to solve complex optimization problems.

The underlying idea behind the competitive and cooperative co-evolution framework is to allow the decomposition process of the optimization problem to adapt and emerge rather than being fixed at the start of the optimization process. In particular, each species will compete to represent a particular component of the MO problem while the eventual winners will cooperate to achieve better solutions. Through this iterative process of competition and cooperation, the various components are optimized by different species based on the optimization requirements of each particular time instant.

The reminder of the chapter is organized as such: In Section 4.1, the related works on co-evolutionary algorithms are reviewed, and the general framework of the proposed competitive and cooperative co-evolutionary paradigm is introduced. The detailed competitive and cooperative co-evolutionary multi-objective particle swarm optimization algorithm (CCPSO) is presented in Section 4.2. In section 4.3, the performance of the proposed algo-

rithm is measured against other leading evolutionary algorithms (EAs) on some established test functions. In section 4.4, a sensitivity analysis of parameters is given. Conclusions are drawn in section 4.5.

4.1 Competition, Cooperation and Competitive-cooperation in Coevolution

The coevolutionary paradigm can be broadly classified into two main categories namely, competitive coevolution and cooperative coevolution. For the former, the various species will always fight to gain an advantage over the others. However, for the latter, species will interact with one another within a domain model and have a cooperative relationship [152]. Regardless of the different approaches, successful implementation of coevolutionary requires the explicit consideration of several design issues [151] such as problem decomposition, parameter interactions and credit assignment, which are inherently problem dependent. Since this work incorporates the coevolutionary paradigm into multi-objective particle swarm optimization, issues that are unique to the optimization model should be considered also i.e. the interaction of the coevolutionary model with the social learning model and the balance between proximity and diversity in multiobjective optimization.

In this section, relevant works of coevolutionary model will be reviewed where their key features and limitations will be highlighted and discussed. Subsequently, the adopted competitive-cooperation model will be presented along with detailed discussions on how the different design issues are addressed.

4.1.1 Competitive Co-evolution

The model of competitive co-evolution can be viewed as an “arms race”or “predator-prey”where different species reciprocally drive each other to increase the overall level of

performance [4] [160]. Specifically, the losing species will continually adapt themselves during the evolutionary process to counter the winning species in order to become the new winner.

Lohn et al [117] incorporates the competitive co-evolutionary model into MO optimization, which contains two competing species, the species of candidate solutions and the species of target objective vectors. Empirical studies have shown the superiority of the proposed model as compared to other well-known MO algorithms such as SPEA and NSGA. However, a major limitation of this approach is the lack of explicit diversity preservation mechanism to guide the co-evolutionary optimization process. This is especially important in the context of MO optimization where the optimization goals of proximity and diversity are of equal importance. In particular, for the species of target vectors, convergence pressure must be exerted to promote individuals in a direction that is normal as well as tangential to the tradeoff region at the same time.

Compared to cooperative co-evolution, competitive convolution is less widely applied, especially in the domain of multi-objective numerical optimization. The competitive co-evolutionary model poses several difficult design issues that could probably explain the lack of work in this area. While competitive co-evolution is a natural model for evolving objects such as game playing programs for which it is difficult to write an external fitness function, the need to hand-decompose the problem into antagonistic components places severe limitation on its range of applicability. Inappropriate decomposition might easily lead to the problem of cycling, where the competitive convolution model are stuck with poor solutions that defeat each other in a cycle. Furthermore, frequent evaluations of species members are needed to determine an accurate ranking corresponding to high computational complexity for practical problems.

4.1.2 Cooperative Co-evolution

The cooperative co-evolution model is inspired by the ecological relationship of symbiosis where different species live together in a mutually beneficial relationship. The underlying idea in cooperative co-evolution is to divide and conquer [152]: divide large system into many components, evolve the components separately and then combine them together to form the whole system. The associated algorithm hence involves a number of independently evolving species that together form complex structures for solving difficult problem.

An early attempt to integrate the cooperative model for MO optimization is based on the idea of dividing the decision space of the problem into several parts and each part is optimized by a MO genetic algorithm (MOGA) [54]. In this MO cooperative coevolutionary genetic algorithm (MOCCGA) [89], each individual is evaluated twice in collaboration with either a random or the best representative from the other subpopulations and the best Pareto rank is assigned as fitness. However, MOCCGA is limited by the lack of elitism and the localized perception of Pareto optimality.

To improve on the lack of the localized perception of Pareto optimality, Maneeratana et al [121] incorporated elitism in the form of a fixed sized archive to store the set of non-dominated solutions. The same cooperative model is also successfully extended to other MO algorithms such as Niched Pareto GA [70], NSGA [184] and NSGAII [79] with significant improvements over their canonical counterparts. Like MOCCGA, these algorithms also suffer from the problem that fitness assignment conducted within each species may not be a good indicator of optimality.

In [79], Iorio and Li presented the nondominated sorting cooperative coevolutionary algorithm (NSCCGA) which is essentially the coevolutionary extension of NSGAII. NSCCGA is different from the previous two works in the sense that elitist solutions are reinserted into the subpopulations and fitness assignment takes into account the set of nondominated solutions found via the nondominated sorting. Instead of selecting nondominated individ-

uals with the best degree of crowding, representatives are selected randomly from the best nondominated front.

Tan et al [191] implemented a cooperative co-evolutionary algorithm (CCEA) that is based on the decomposition of the problem according to the number of parameters, where each species will optimize one parameter of the solution vector. A different ranking scheme was adopted where each individual is ranked against the non-dominated solutions stored in the archive instead of within the species. Furthermore, an extending operator which reinserts non-dominated solutions with the best niche count into the evolving species is implemented in CCEA to improve diversity and distribution of the Pareto front. The authors also investigated the effects of various representative selection and observed that robust performances can be better achieved by conducting cooperation with two representative from each subpopulations and retaining the better collaboration.

Similar cooperative co-evolutionary model was extended to the particle swarm optimization for SO optimization [205] where each swarm will optimize a single component of the solution vector, analogous to the decomposition used in the relaxation method [182]. The advantage of this approach over conventional particle swarm optimization is that only one component is altered at a time, resulting in the desired fine grain search. Also, diversity was significantly increased due to the different combinations formed via the different members from the different swarms.

In cooperative co-evolutionary algorithms, the fitness of an individual is depended on its ability to collaborate with individuals from other species, which favors the development of cooperative strategies and individuals. In addition, the modules that are evolved simultaneously in cooperative co-evolution can be categorized to two basic levels, component level and system level [94]. In the cases of single-level co-evolution [27] [79] [89] [121], each evolving species represents a component of the problem to be solved. On the other hand, a two-level co-evolutionary process involves simultaneous optimization of the system and components in separate species [6] [57].

However, one major issue of these co-evolutionary approaches is their dependence on the appropriate hand-decomposition of the problem into components before the optimization process. In particular, Iorio and Li [79] highlighted that co-evolutionary algorithms are susceptible to parameter interactions, as interacted parameters may end up being optimized by different species. Besides, as MO optimization is associated with a set of non-dominated solutions, appropriate measures must be undertaken to drive the various species in tandem towards the optimal Pareto front to facilitate the search of a diverse and uniformly distributed solution set. Apparently, there is an inherent tradeoff between the fine-grain search capability and the diversity in the relatively small sized searching species of co-evolutionary algorithms.

Competitive-Cooperation Co-evolution

From the earlier discussions, it is apparent that problem decomposition places severe restrictions on algorithmic design and performance for both competitive and cooperative models. Furthermore, it should be noted that cooperation and competition among the different species are adopted independently in co-evolutionary algorithms, even though the two different types of interaction are rarely exclusive within an ecological system. As plant-pollinator co-evolution in nature [169], different species of bees will compete for nectar and different species of flowers will compete to attract more bees. In this particular example, the problem decomposition arises naturally as the role at which each species play is an emergent property in nature. As such, the proposed co-evolutionary model will incorporate both elements of cooperation and competition which represent a more holistic view of the co-evolutionary forces in nature.

The proposed model involves two tightly-coupled coevolutionary processes and the relationship between them is illustrated in Figure 4.1. As in the case of conventional cooperative co-evolutionary algorithms, particles from the different species collaborate to solve the problem at hand during the cooperative process. Each species evolves in isolation and there is

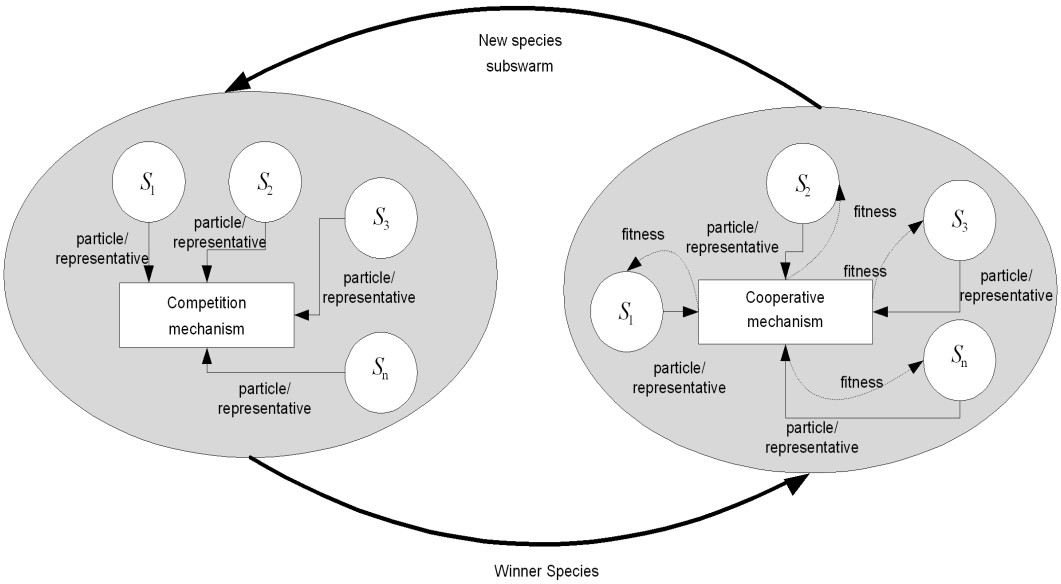

Figure 4.1: Framework of the competitive-cooperation model

no restriction on the form of representation or the underlying optimization algorithm. On the other hand, the cooperative species will enter into competition with other species for the right to represent the various components of the problem.

Credit Assignment: Credit assignment for the competitive-cooperation process is performed at the species and particle level respectively. Following the situation described earlier, the different objectives of the MO problem at the cooperative process are evaluated by assembling each particle along with the representatives of the other species to form a valid solution. Accordingly, appropriate fitness assignment such as Pareto ranking can be computed for the particular particle. In the competitive process, the fitness of particular species is computed by estimating how well it performs in a particular role relative to its competitors in the cooperation with other species to produce good solutions.

Problem Decomposition and Component Interdependency: As mentioned earlier, problem decomposition is one of the primary issues to be addressed in co-evolutionary optimization and the difficulty is that information pertinent to the number or role of components are usually not known a priori and many problems can only be decomposed into compo-

nents exhibiting complex interdependencies. To this end, the competitive and cooperative co-evolutionary model addresses this issue through emergent problem decomposition.

The competitive process in the proposed model will trigger a potential "arms race" among the various species to improve their contribution to overall fitness of the ecosystem. The benefits of this competition also include the discovery of interdependencies between the components as the species competition provides an environment in which interdependent components end up being optimized by same species. The reasonable problem decompositions emerge due to evolutionary pressure rather than being specified by the user.

Diversity: The competitive and cooperation co-evolutionary model provides a means of exploiting the complementary diversity preservation mechanisms of both competitive and cooperative models. In cooperative model, the evolution of isolated species tends to generate higher diversity across the different species, although this property does not necessarily extend to within each species. On the other hand, within a species diversity is driven by the necessity to deal with the changing competition posed by the other species in the competitive model. Furthermore, the competitive process in competitive and cooperation co-evolutionary model allows a more diversified search as the optimization of each component is no longer restricted to one species. The competing species provides another round of optimization for each component, which increases the extent of the search while maintaining low computational requirements.

4.2 Competitive-Cooperation Co-evolution for MOPSO

The competitive-cooperation co-evolutionary multi-objective particle swarm optimization algorithm (CCPSO) introduced in this chapter attempts to emulate the conflict and coexistence between cooperation and competition in nature by implementing both aspects into a MOPSO. Section 4.2.1 describes the cooperative co-evolutionary mechanism for CCPSO. Section 4.2.2 explains the competitive co-evolutionary mechanism for CCPSO. And the flowchart of CCPSO is shown in section 4.2.3.

4.2.1 Cooperative Mechanism for CCPSO

For MO problems, the original PSO uses a swarm of n-dimensional vectors. In CCPSO, these vectors are partitioned into n subswarms of 1-D vectors, each subswarm representing a dimension of the original problem. Each subswarm attempts to optimize a single variable of the solution vector, essentially a 1-D optimization problem.

One complication to this configuration is the fact that the objective function to be minimized, requires an n-dimensional vector as input. If each subswarm represents only a single dimension of the search space, it is clearly not possible to directly compute the fitness of the particles of a single subswarm considered in isolation. A context vector is required to provide a suitable context in which the particles of a subswarm can be evaluated. The simplest scheme for constructing such a context vector is to take the representative particle from each of the n subswarms and concatenating them to form such an n-dimensional vector. To calculate the fitness for all particles in subswarm i, the other $n-1$ components in the context vector are kept constant (with their values set to the representative particles from the other subswarms), while the i-th variable of the context vector is replaced in turn by each particle from the i-th subswarm. Accordingly, appropriate fitness assignment such as Pareto ranking and niche count can be computed for the particular particle. The representative of each subswarm is selected based on Pareto ranking and niche count.

The pseudocode of the cooperative mechanism is shown in Figure 4.2. At the start of the optimization process, the i-th subswarm (S_i) is initialized to represent the i-th variable. Concatenation between particles in S_i and representatives from the other subswarms form valid solutions for evaluation. As an example, consider a 3-decision variable problem where subswarms, S_1, S_2 and S_3, represent the variables x_1, x_2 and x_3 respectively. When assessing the fitness of $s_{1,j}$, it will combine with the representatives of S_2 and S_3 to form a valid solution.

Archive update is conducted after each particle evaluation. After which, Pareto ranking and niche count computation of particle, $s_{i,j}$ are conducted with respect to the archive. The

Cooperative Process

S_i : i-th subswarm
$s_{i,j}$: j-th particle of S_i
A : Archive of nondominated solutions

for $j = 1$ to $|S_i|$ **do**

- Assemble complete solution with $s_{i,j}$ and representatives from other subswarms
- Evaluate solution
- Update A

end

for $j = 1$ to $|S_i|$ **do**

- Assign Pareto ranking to $s_{i,j}$
- Calculate niche count of $s_{i,j}$

end

Update representative of S_i

Figure 4.2: Pseudocode for the adopted cooperative coevolutionary mechanism.

Pareto rank is given by

$$rank(s_{i,j}) = 1 + n_{s_{i,j}} \tag{4.1}$$

where $n_{s_{i,j}}$ is the number of archive members dominating the particle $s_{i,j}$ in the objective domain. Similar to the ranking process, the niche count (nc) of each particle is calculated with respect to the archive of nondominated solutions. The dynamic sharing proposed in [195] is employed in this work.

The cooperative process is carried out in turn for all n subswarms. Before proceeding to the evaluation of the next subswarm, the representative for variable i denoted as $s_{i,rep}$ is updated in order to improve convergence speed. This updating process is based on a partial order such that ranks will be considered first followed by niche count in order to break the tie of ranks. For any two particles, $s_{i,j}$ and $s_{i,k}$, $s_{i,j}$ is selected over $s_{i,k}$ if $rank(s_{i,j}) < rank(s_{i,k})$ or $\{rank(s_{i,j}) = rank(s_{i,k})$ and $nc(s_{i,j}) < nc(s_{i,k})\}$. The rationale of selecting a

Comptitive Process

P_i^c : Competition pool for variable i

for $j = 1$ to $|S_i|$ **do**

- Insert representative of subswarm representing variable i, $s_{i,rep}$ into P_i^c
- Select competing subswarms
- Insert competitors from competing subswarms into P_i^c
- Cooperative process
- Update subswarm representing variable i

end

Figure 4.3: Pseudocode for the adopted competitive coevolutionary mechanism.

nondominated representative with the lowest niche count is to promote the diversity of the solutions using the approach of cooperation among multiple subswarms.

4.2.2 Competitive Mechanism for CCPSO

Given the cooperative scheme of optimizing a single variable in each subswarm, one simple approach is to allow the different subswarms to take up the role of a particular problem component in a round-robin fashion. The most competitive subswarm is then determined and the component will be optimized by the winning species in the next cooperative process. Ideally, the competition depth is such that all particles from a particular subswarm compete with all other particles from the other subswarm in order to determine the extent of its suitability. However, such an exhaustive approach requires extensive computational effort and it is practically infeasible. A more practical approach is to conduct competition with only selected particles among a certain number of competitor subswarms to estimate the species fitness and suitability.

The pseudocode of the competitive mechanism is shown in Figure 4.3. For the i-th variable, the representative of the associated subswarm, i.e. $s_{i,rep}$, is selected along with

competitors from the other subswarms to form a competition pool. With regard to the issue of competitor selection, CCPSO adopts a simple scheme of selecting a random particle from each competing subswarm.

These competitors will then compete via the cooperative mechanism described before to determine the extent of cooperation achieved with the representative of the other subswarms. The winning species can be determined by simply checking the originating subswarm of the representative after the representative update. At the end of the competitive process, S_i will remain unchanged if its competition representative wins the competition. In the case where a winner emerges from other subswarms, S_i will be replaced by the particles from the winning subswarm. The rationale of replacing the losing subswarm instead of associating the winning subswarm directly with the variable is that different variables may have similar but not identical properties. Therefore, it would be more appropriate to seed the losing subswarm with the desirable information and allow it to evolve independently.

By embedding the competitive mechanism within the cooperative process, the adaptation of problem decomposition and the optimization process are conducted simultaneously. Hence, no additional computation cost is incurred by the competition. It has the further advantage of providing the chance for the different subswarms to solve for a single component, with the competitors as a source of diversity.

4.2.3 Flowchart of CCPSO

The flowchart of the proposed algorithm is shown in Figure 4.4. The optimization process starts with the initialization of the different species. After that, the cooperation mechanism described in Section 4.2.1 is conducted to evaluate the particles in each species. The algorithm employs a fixed-size archive to store non-dominated solutions along the evolution. As mentioned in the prior sections, the archive is updated after every particle evaluation within the cooperative or competitive mechanism. A complete solution formed by the species will be added to the archive if it is not dominated by any archived solutions. Likewise, any

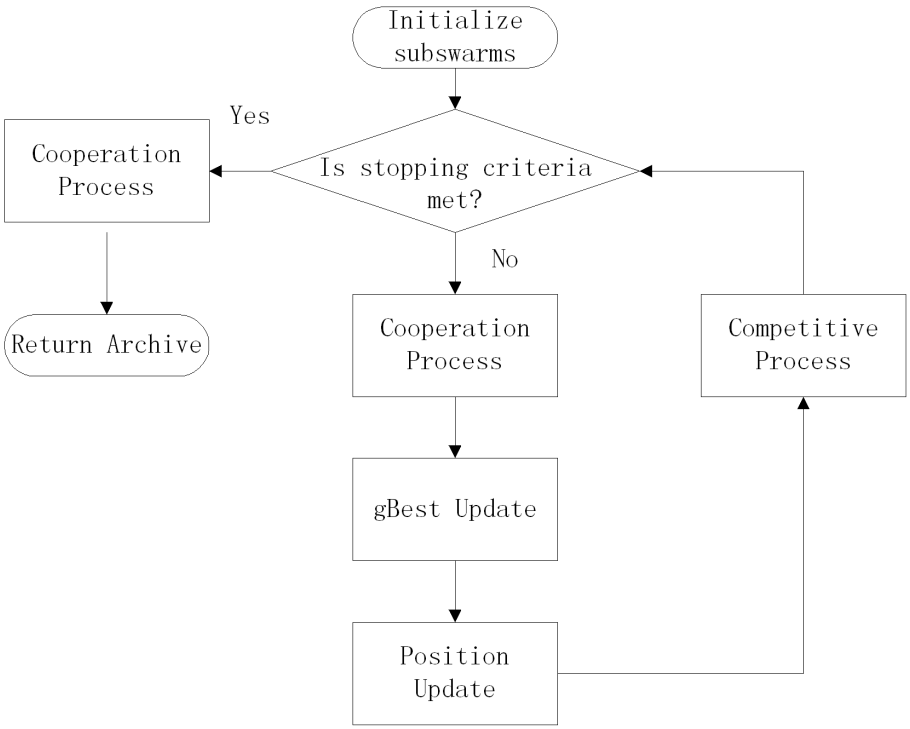

Figure 4.4: Flowchart of Competitive-Cooperative Co-evolutionary MOPSO

archive members dominated by this solution will be removed. When the predetermined archive size is reached, a recurrent truncation process [96] based on niche count is used to eliminate the most crowded archive member.

pbest is updated during the cooperation process while gbest is selected only after all the species are evaluated. The update of particle position includes the fly operator defined in equation 2.2 and 2.5. The competitive process is then activated to adapt the assignment of decision variables to the species. This process will continue until the stopping criterion is satisfied.

4.3 Performance Comparison

In this section, CCPSO is compared with the various evolutionary multi-objective optimization methods (NSGAII [35], SPEA2 [224], IMOEA [198] and PAES [103]) on four benchmark

problems (FON, KUR, ZDT4 and ZDT6). Two multi-objective particle swarm optimization algorithms (SIGMA [130] and MOPSO [26]) have also been included to reflect an accurate measure of the algorithm against other established PSO algorithms implemented rather than simply pitting the CCPSO against other MOEAs, where any inherent advantages or disadvantages that a MOPSO may have over MOEAs may skew the results. The parameter settings and indices of the different algorithms are shown in Table 4.1 and Table 4.2, respectively. Four performance metrics (GD, MS, S and N) are used to provide a quantitative assessment for the performance of different algorithms, where N represents the number of non-dominated solutions generated by the algorithm, and is restricted by the size of the archive (100). Naturally a higher N is desirable as the Pareto front is better defined.

Table 4.1: Parameter setting for different algorithms

Parameter	Settings
Populations	Population size 100 in NSGAII, SPEA2, SMOPSO, IMOEA, CMOPSO; Subpopulation size 10 in CCPSO; Population size 1 in PAES; Archive (or secondary population) size 100.
Chromosome	Binary coding; 30 bits per decision variable in IMOEA, PAES, NSGAII and SPEA2. Real number representation in CCPSO, CMOPSO, and SMOPSO.
Selection	Binary tournament selection.
Crossover operator	Uniform crossover in IMOEA, NSGAII and SPEA2.
Crossover rate	0.8 in IMOEA, NSGAII and SPEA2.
Mutation operator	Bit-flip mutation in IMOEA, PAES, NSGAII and SPEA2. Turbulence operator in CCPSO, CMOPSO, and SMOPSO.
Mutation rate	$\frac{1}{L}$ for ZDT4, ZDT6, DTLZ2 and DTLZ3 where L is the chromosome length; $\frac{1}{B}$ for FON and KUR where B is the bit size per decision variable;
Turbulence strength	0.2.
Niche Radius	Dynamic sharing [195].

Table 4.2: Indices of the different algorithms

Index	1	2	3	4	5	6	7
Algorithms	NSGA-II	SPEA2	SIGMA	CCPSO	IMOEA	MOPSO	PAES

FON: FON has a non-convex and nonlinear tradeoff curve that challenges the algorithm's ability to find and maintain the entire tradeoff curve uniformly. A stopping criterion of 8000 evaluations is used for this problem. Figure 4.5 shows the Pareto fronts generated by the algorithms across thirty independent runs. The distribution of the different performance metrics is represented by box plots in Figure 4.6, while evolutionary trajectories of GD and N are shown in Figure 4.7. Figure 4.8 plots the objective space explored by particles in one arbitrary run for CCPSO.

As shown in Figure 4.5, the Pareto fronts generated by CCPSO and MOPSO is superior to those generated by the rest of the algorithms as these two algorithms can find the extreme solutions at each end of the objective space and cover both ends of the true Pareto front. For IMOEA, it can not provide enough pressure for the search along objective 1 and ends up covering only half of the true Pareto front. From the boxplots in Figure 4.6, it is apparent that the PSO paradigm has a slight edge in dealing with the nonlinear tradeoff of FON. While SIGMA, CCPSO and MOPSO demonstrated competitive performance, the four MOEAs exhibited a varying degree of success. In particular, PAES and IMOEA performed inconsistently in thirty runs as shown by the large variance on GD, MS and S. It can be observed that MOPSO attains the best performance in MS. Nonetheless, CCPSO and MOPSO are competitive in the aspects of convergence and distribution as evident in Figure 4.6(a)-(c). Figure 4.7 reveals that SIGMA, CCPSO and MOPSO are capable of attaining higher convergence speed, and Figure 4.8 further illustrates how CCPSO converged to true Pareto front in just 10 cycles.

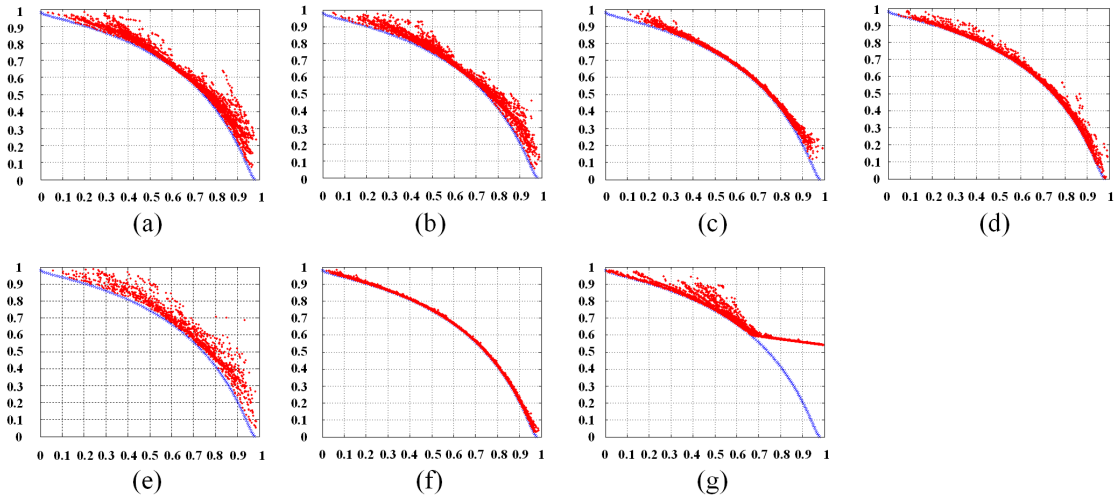

Figure 4.5: Pareto fronts generated across 30 runs by (a) NSGAII, (b) SPEA2, (c) SIGMA, (d) CCPSO, (e) IMOEA, (f) MOPSO, and (g) PAES for FON

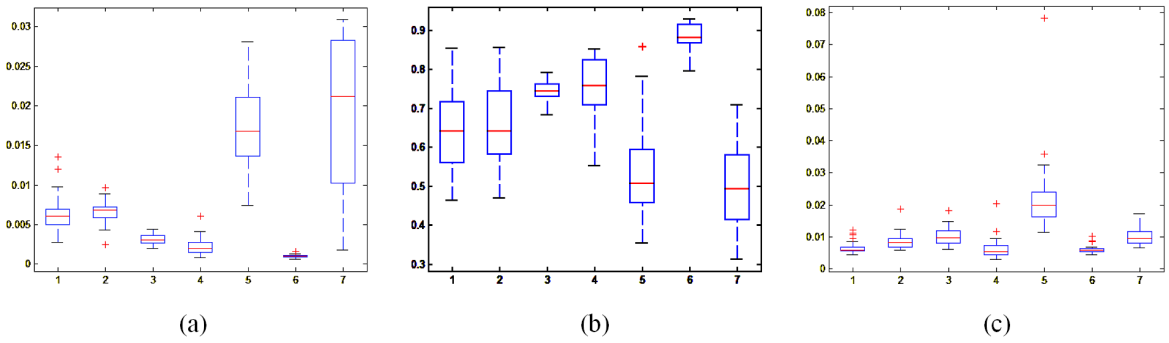

Figure 4.6: Performance metrics of (a) GD, (b) MS, and (c) S for FON

KUR: KUR is characterized by a tradeoff that is nonconvex and disconnected, i.e., it contains three distinct disconnected regions on the final tradeoff. The decision variables corresponding to the global tradeoff for KUR are difficult to be discovered, since they are disconnected in decision space as well. A stopping criterion of 4500 evaluations is used for this problem. The distribution of the different performance metrics is represented by box plots in Figure 4.9, while the evolutionary trajectories of GD and N are shown in Figure 4.10. Figure 4.11 plots the objective space explored by particles in one arbitrary run for

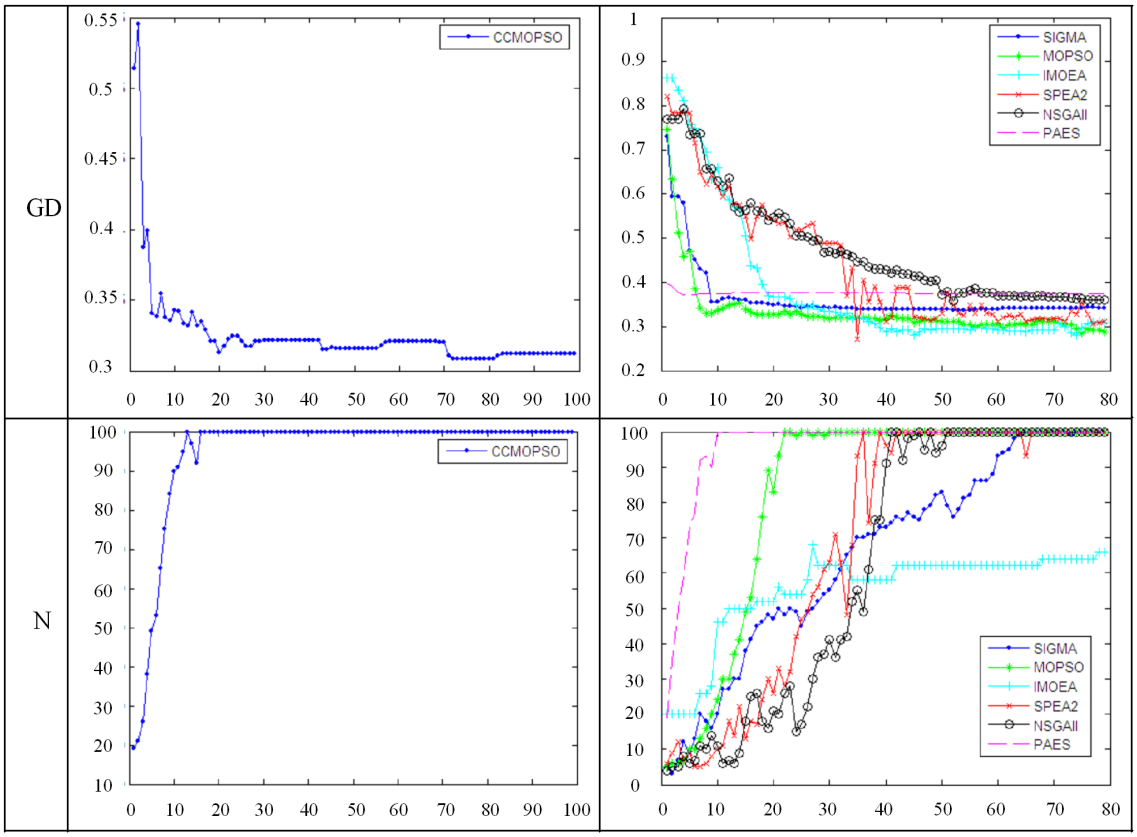

Figure 4.7: Evolutionary trajectories in GD and N for FON

CCPSO.

Most of the algorithms faced no problems finding solutions near the true Pareto front for KUR. It is evident from Figure 4.9 that CCPSO performs better than the other algorithms in GD and S while remaining strongly competitive in terms of MS. Figure 4.10 also reveals that CCPSO has a better convergence speed. Unlike the case of FON where the PSO-based algorithms are significantly faster than the MOEA counterparts, SPEA2 demonstrated competitive convergence speed. Figure 4.11 further illustrates how CCPSO converge to true Pareto front in just 20 cycles. In Figure 4.10, it should be noted that results of PAES in terms of GD is not plotted as PAES can not locate the true Pareto front within the allocated number of evaluations and its performance on GD is very poor compared to the

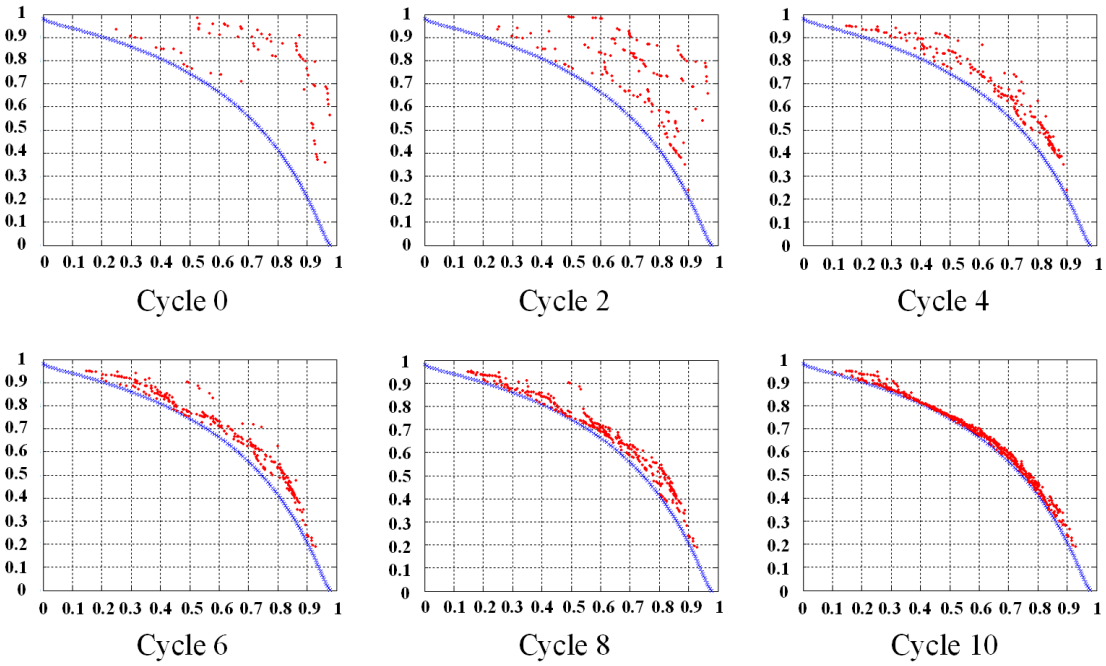

Figure 4.8: Convergence behavior of CCPSO for FON

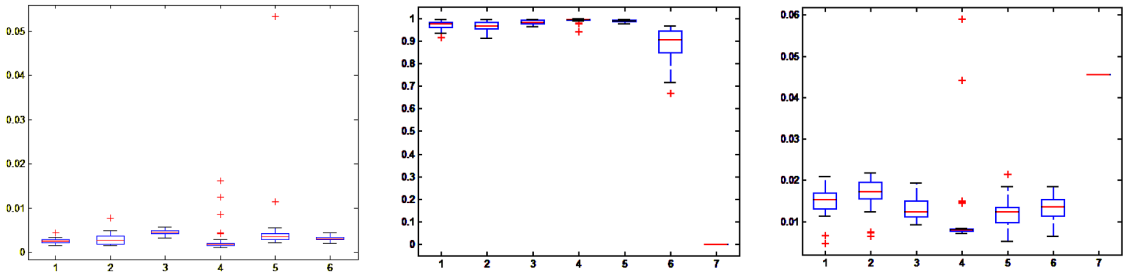

Figure 4.9: Performance metrics of (a) GD, (b) MS, and (c) S for KUR

other algorithms.

ZDT4: ZDT4 is characterized by a severe multi-modal landscape and a stopping criterion of 10 000 evaluations is used for this problem. The evolved Pareto fronts from all 30 simulation runs for the algorithms are plotted in Figure 4.12. The distribution of the different performance metrics is represented by box plots in Figure 4.13, while the evolutionary

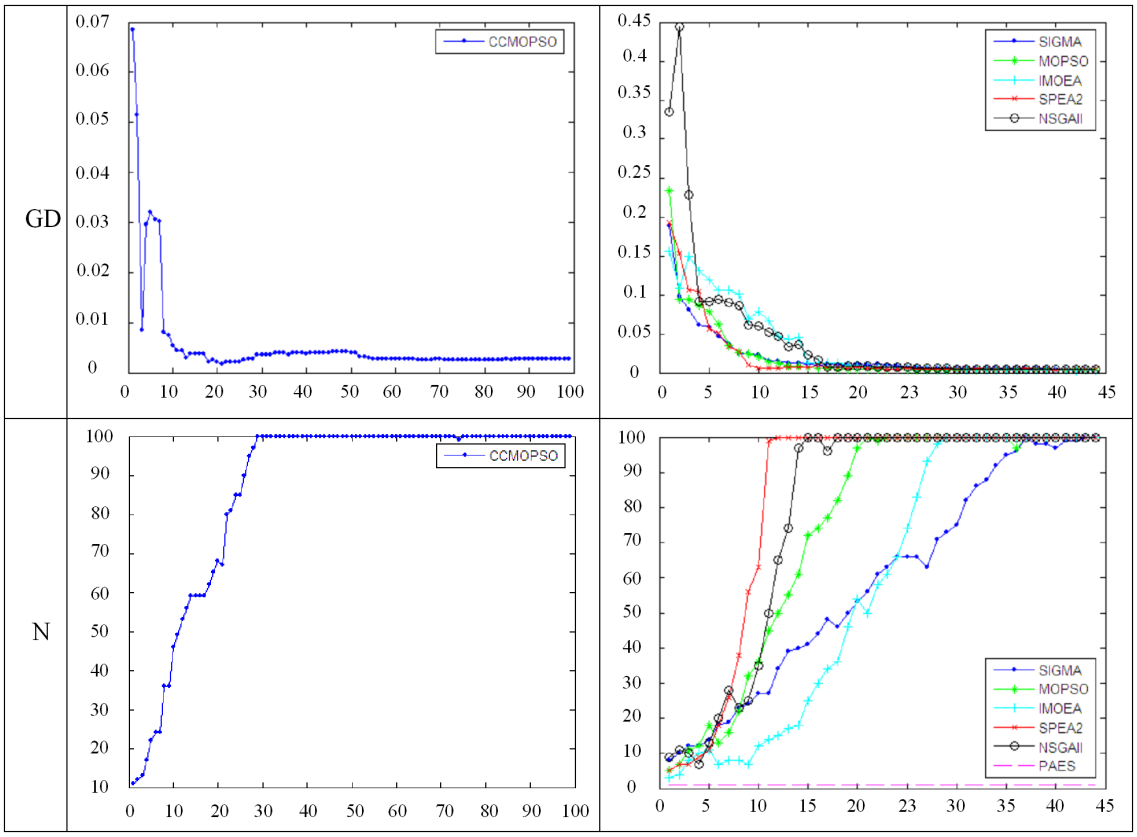

Figure 4.10: Evolutionary trajectories in GD and N for KUR

trajectories with respect to different performance metrics are shown in Figure 4.14.

ZDT4 proved to be a very difficult problem to solve, mainly due to the high number (21^9) of local fronts. As observed from Figure 4.12(d) and Figure 4.13, CCPSO is the only algorithm capable of attaining the true Pareto front. In addition, the poor performances of MOPSO and SIGMA in the aspects of GD and MS may be the consequence of high convergence speed, which leads to the algorithms getting trapped in local optima. The steps observed in the trend of MS and N for CCPSO in Figure 4.14 correspond to the effect of jumping from one local Pareto front to another during the search.

ZDT6: ZDT6 has a biased search space and non-uniformly distributed solutions along the global tradeoff, which makes it difficult for algorithms to evolve a well-distributed Pareto

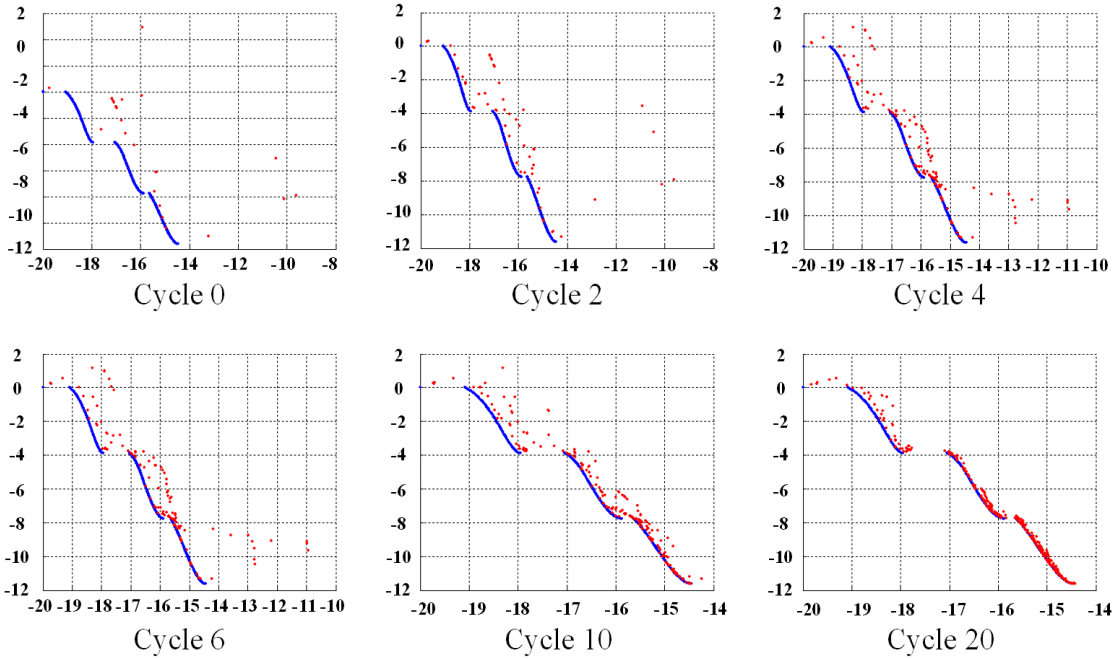

Figure 4.11: Convergence behavior of CCPSO for KUR

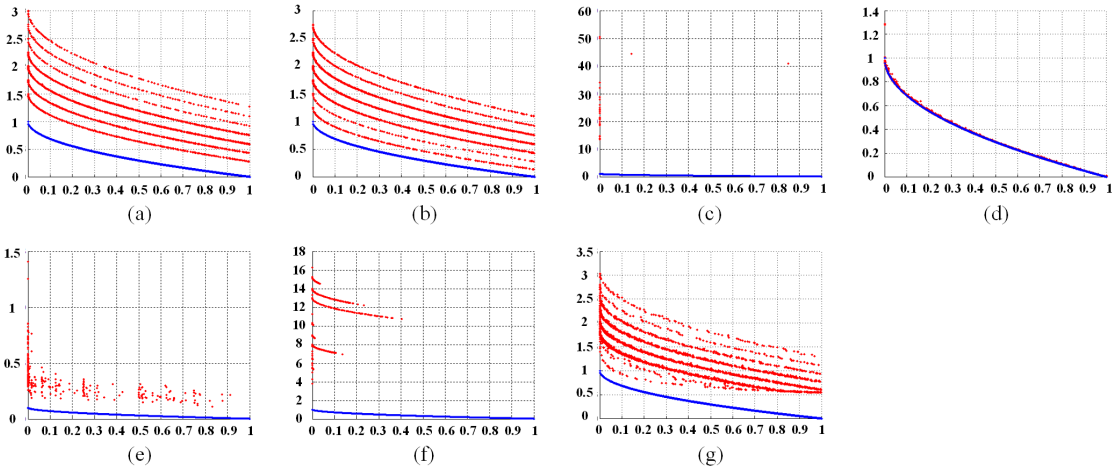

Figure 4.12: Pareto fronts generated across 30 runs by (a) NSGAII, (b) SPEA2, (c) SIGMA, (d) CCPSO, (e) IMOEA, (f) MOPSO, and (g) PAES for ZDT4

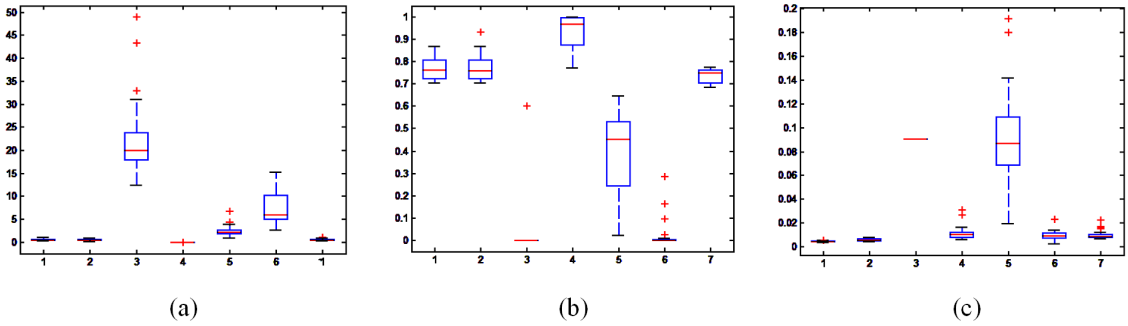

Figure 4.13: Performance metrics of (a) GD, (b) MS, and (c) S for ZDT4

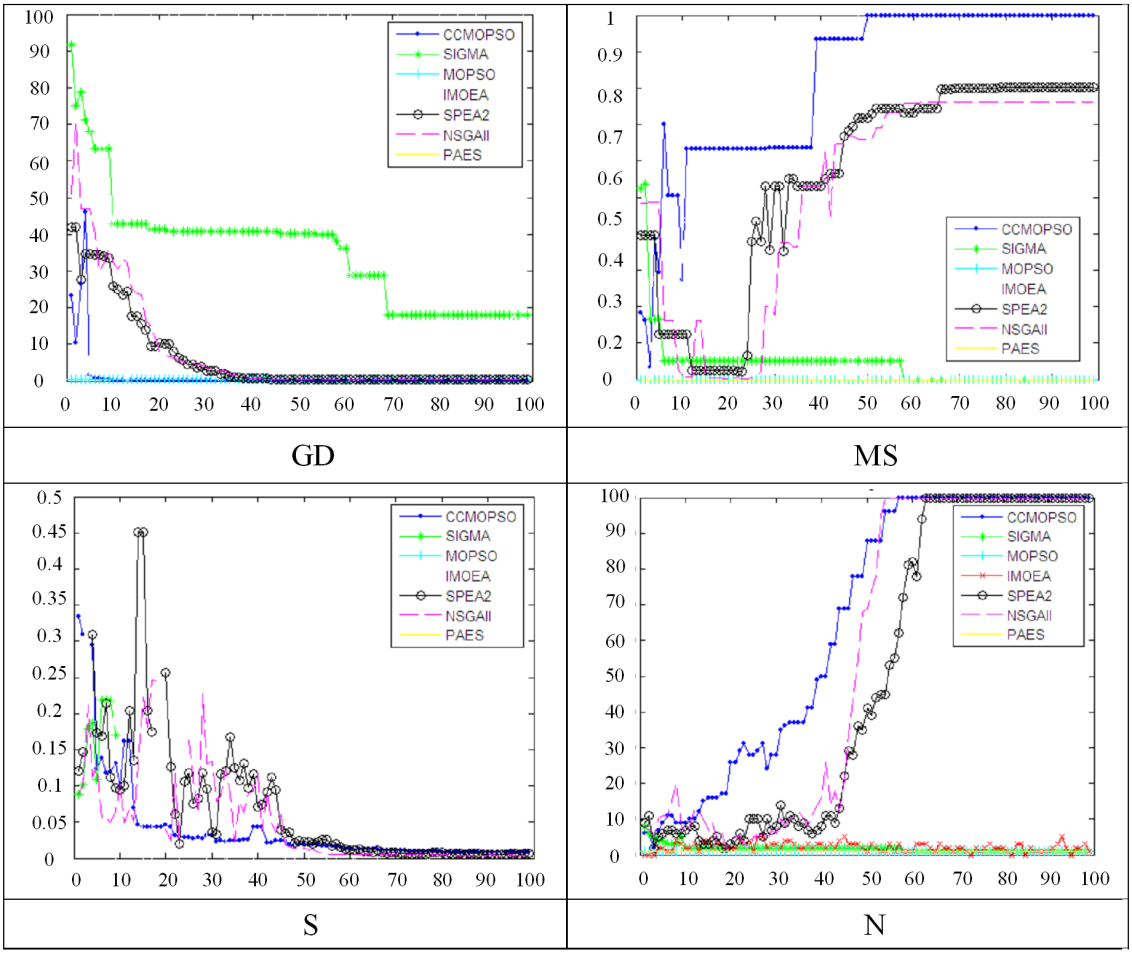

Figure 4.14: Evolutionary trajectories in GD, MS, S, and N for ZDT4

front. The evolved Pareto fronts from all 30 simulation runs for the algorithms are plotted in Figure 4.15. The distribution of the different performance metrics is represented by box plots in Figure 4.16, while the evolutionary trajectories with respect to different performance metrics are shown in Figure 4.17.

As in the case for FON, the PSO algorithms in general outperformed the rest of the algorithms, managing to cover almost the entire true Pareto front over the thirty runs. The PSO algorithms also seems having no problem escaping from local minima, with the exception of points on the extreme left of the objective space. CCPSO is the best among the algorithms with only a few solutions trapped at those points as shown in Figure 4.15.

From Figure 4.16(a) and (b), it is apparent that NSGA-II and SPEA2 are unable to reach the true Pareto front consistently and failed to cover the entire true Pareto front throughout the thirty runs. It can be noted from Figure 4.16 and Figure 4.17 that while PAES outperforms NSGA-II and SPEA2 in the aspect of GD, it performs poorly in attaining a well-distributed Pareto front. From Figure 4.16 and Figure 4.17, it can be observed that the proposed CCPSO has a better capability to attain a near-optimal, well-distributed, and well-spread Pareto front with a faster convergence speed.

4.4 Sensitivity Analysis

It can be observed from the above comparative studies that CCPSO is capable of evolving a near-optimal, diverse and uniformly distributed Pareto front for the different benchmark problems. In this section, the impact of various parameter settings of CCPSO is examined. A number of simulations are performed with different settings of $w = \{0.2, 0.1, 0.6\}$, $S_{subpop} = \{5, 10, 20, 50\}$, and $S_{arc} = \{100, 150, 200, 250\}$. An additional setup involving an adaptive inertia weight that reduces from 0.9 to 0.4 is also applied and compared against the static settings. For each experiment only one parameter was changed while maintaining all other parameters constant, and the boxplots show results of parameters being varied in increasing

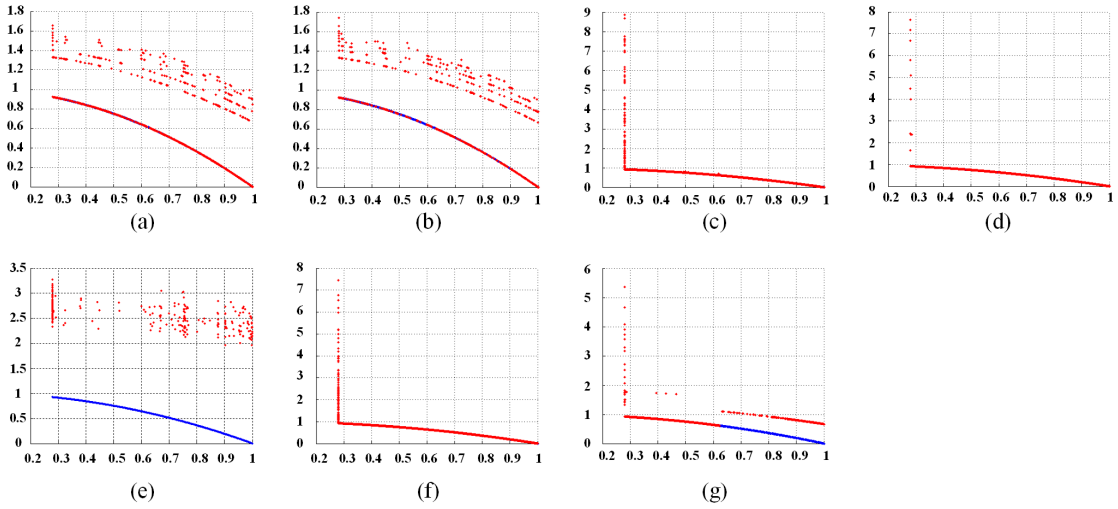

Figure 4.15: Pareto fronts generated across 30 runs by (a) NSGAII, (b) SPEA2, (c) SIGMA, (d) CCPSO, (e) IMOEA, (f) MOPSO, and (g) PAES for ZDT6

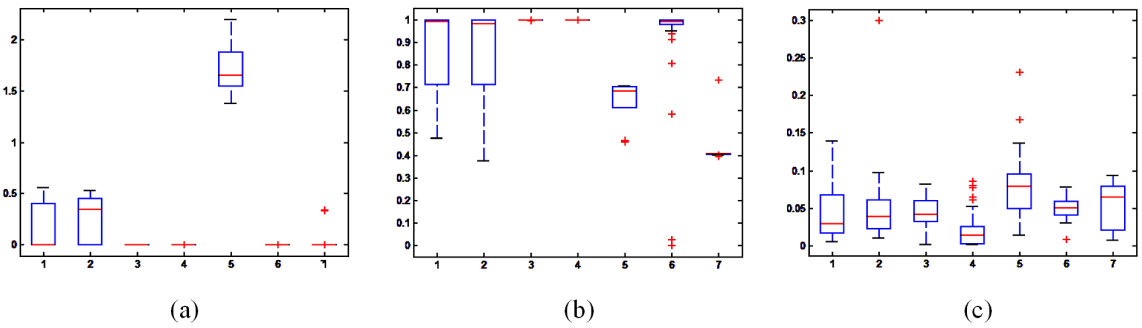

Figure 4.16: Performance metrics of (a) GD, (b) MS, and (c) S for ZDT6

order. The parameters are changed one by one in order to tell algorithm designers the individual effects of each parameter. Such information will be a good starting point for designers to fine-tune their program to real world problems.

Inertial Weight: Figure 4.18-Figure 4.20 show the performance of CCPSO over different inertia weight (w) settings. The inertia weight, which helps maintain the balance between exploration and exploitation, is varied among 0.2, 0.4, 0.6 and an adaptive weight that reduces from 0.9 to 0.4. Note that a smaller w represents a higher tendency to adopt

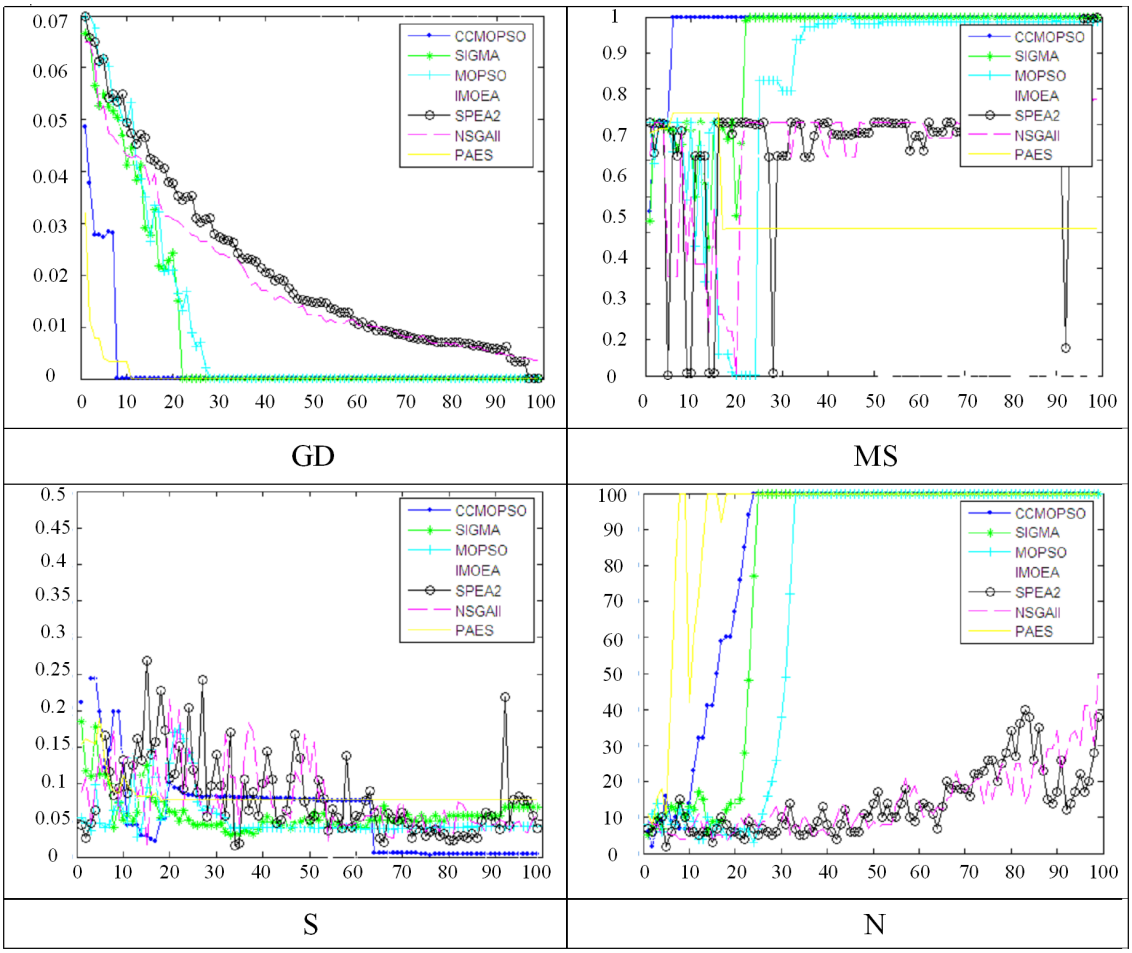

Figure 4.17: Evolutionary trajectories in GD, MS, S, and N for ZDT6

changes brought about by changes due to pbest and gbest. It can be observed that the inertia has a greater impact on MS as compared to S and GD. In particular, a higher w and the adaptive inertia scheme provide a better spread of solutions. It should also be noted that CCPSO is capable of performing consistently and effectively within a large range of w.

FON: A 0.4 weight performed the best in GD. A weight of 0.6 outperformed the rest in MS while performance in S were all virtually indistinguishable.

KUR: A 0.4 weight was the optimal setting for all metrics.

ZDT4: A weight of 0.2 performed the best in GD while a 0.6 weight obtained the best

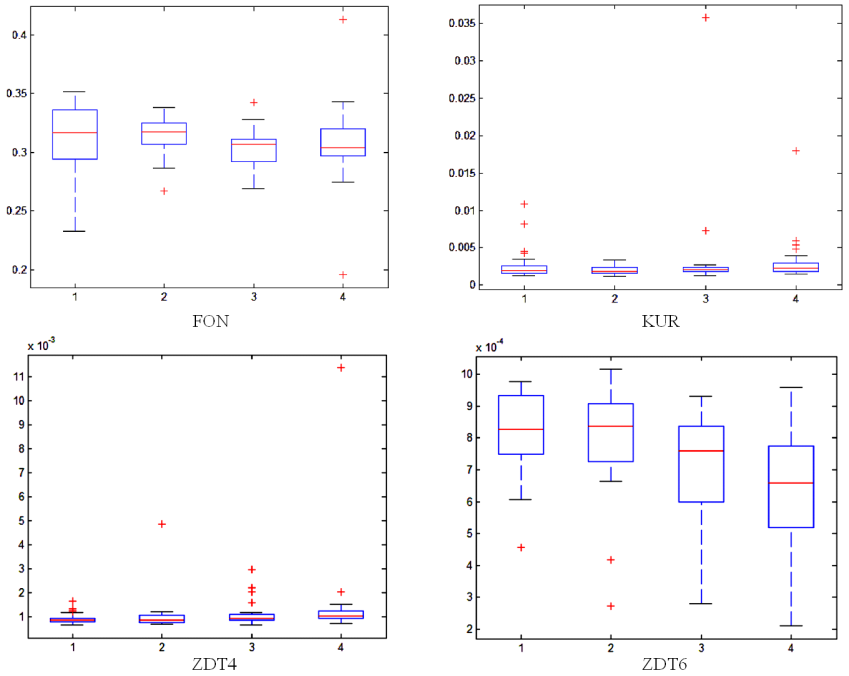

Figure 4.18: Box plots for GD by varying inertia weight

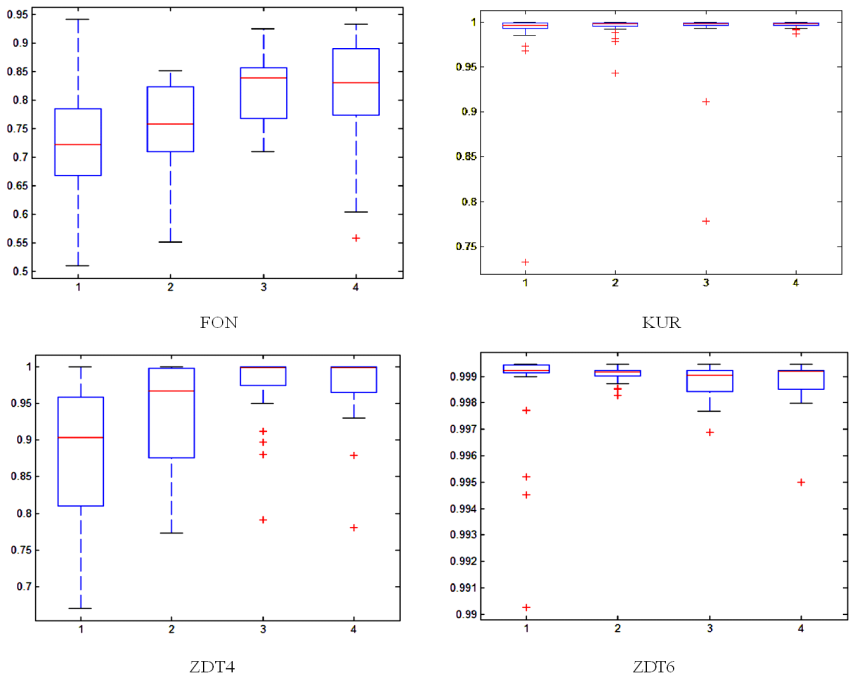

Figure 4.19: Box plots for MS by varying inertia weight

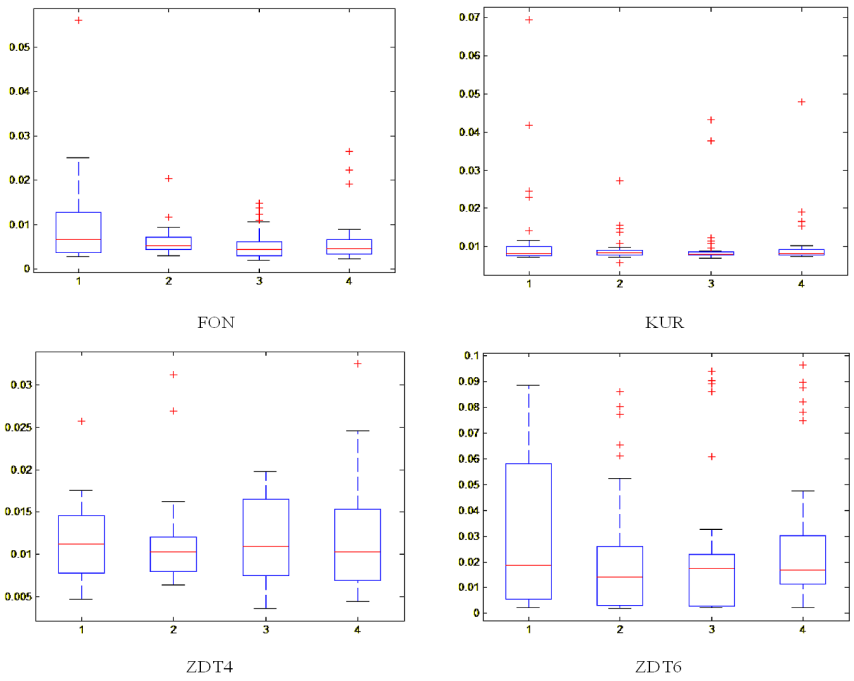

Figure 4.20: Box plots for S by varying inertia weight

maximum spread. However a weight of 0.4 was second best in these metrics while obtaining the best spacing.

ZDT6: A weight of 0.4 was again the optimal setting for all metrics.

Subswarm Size: Figure 4.21-Figure 4.23 show the performance of CCPSO over different settings $S_{subpop} = \{5, 10, 20, 50\}$. The population size of the subswarm was varied while maintaining the total number of evaluations. From the box plots, it is apparent that smaller subswarm size gives rise to better convergence to true Pareto front. Nonetheless, it is noted that better diversity are achieved at higher S_{subpop} settings. This is because, by maintaining a fixed number of evaluations, there is an inherent tradeoff between the diversity provided by a larger swarm size and the number of cycles allowed for exploitation.

FON: A subswarm size of 10 performed the best in terms of GD, but a larger subswarm size of 20 obtained better scores in S and MS.

KUR: A subswarm size of 10 performed better in nearly all metrics.

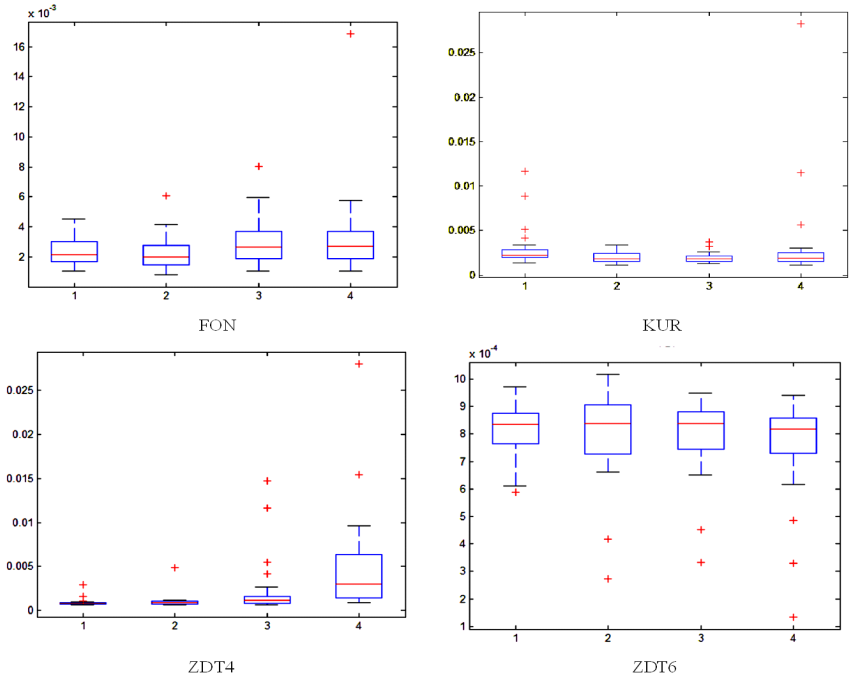

Figure 4.21: Box plots for GD by varying subswarm size

ZDT4: Smaller subswarm size performed better in all metrics. In particular, a subswarm size of 10 was the only one to achieve a maximum number of non-dominated solutions in all runs.

ZDT6: Performance in GD was virtually indistinguishable with all of them separated by a GD of less than 10^{-3}. Smaller subswarm size performed much better for S, but larger subswarm size performed better for MS.

Archive Size: Figure 4.24-Figure 4.26 show the performance of CCPSO over different settings of $S_{arc} = \{100, 150, 200, 250\}$. Apart from the problem of ZDT4 in which the archive size seems to have little effect, CCPSO tends to achieve better results in all aspects with increasing archive sizes. This is because the restriction on the number of archive solutions leads to two phenomena [48] which have a detrimental effect on the search process. The first is the shrinking PF phenomenon which results from the removal of extremal solutions and the subsequent failure to rediscover them. In the second phenomenon, nondominated solutions

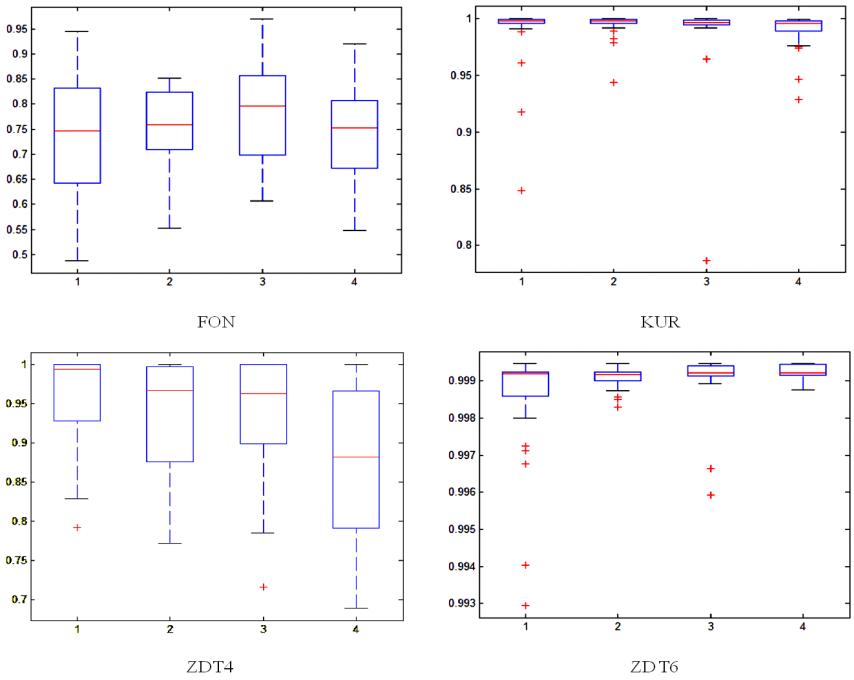

Figure 4.22: Box plots for MS by varying subswarm size

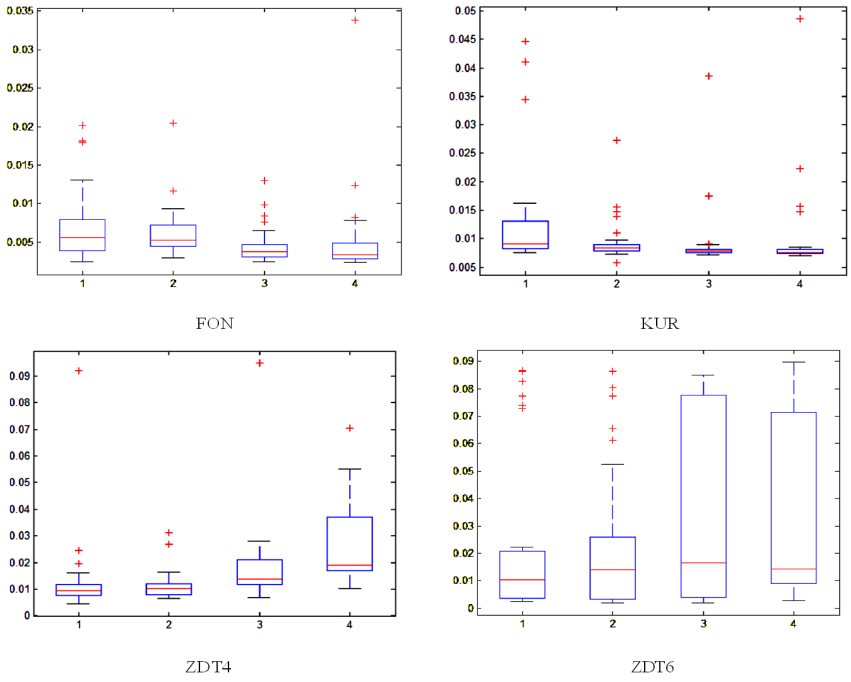

Figure 4.23: Box plots for S by varying subswarm size

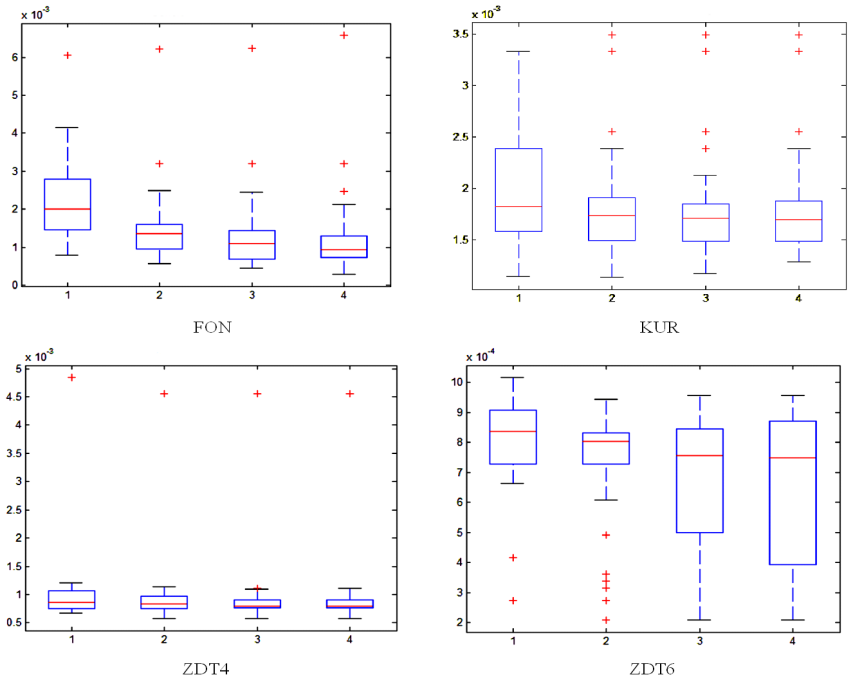

Figure 4.24: Box plots for GD by varying archive size

in the archive are replaced by least crowded individuals. In the subsequent generations, new individuals that would have been dominated by the removed solutions are updated into the archive only to be replaced solutions dominating them. Nonetheless, it can be noted that an archive size of 100 is sufficient to produce good results.

FON: Archive sizes of 150 and 200 performed better in all metrics except GD, where the larger the archive size, the better the GD performance.

KUR: Archive sizes of 150 and 200 were once again the best performers, with very close scores in almost all metrics.

ZDT4: An archive size of 200 performed the best in GD, while an archive size of 100 performed the best for S and MS.

ZDT6: An archive size of 100 was easily the best performer in terms of GD but performed poorly in S and MS.

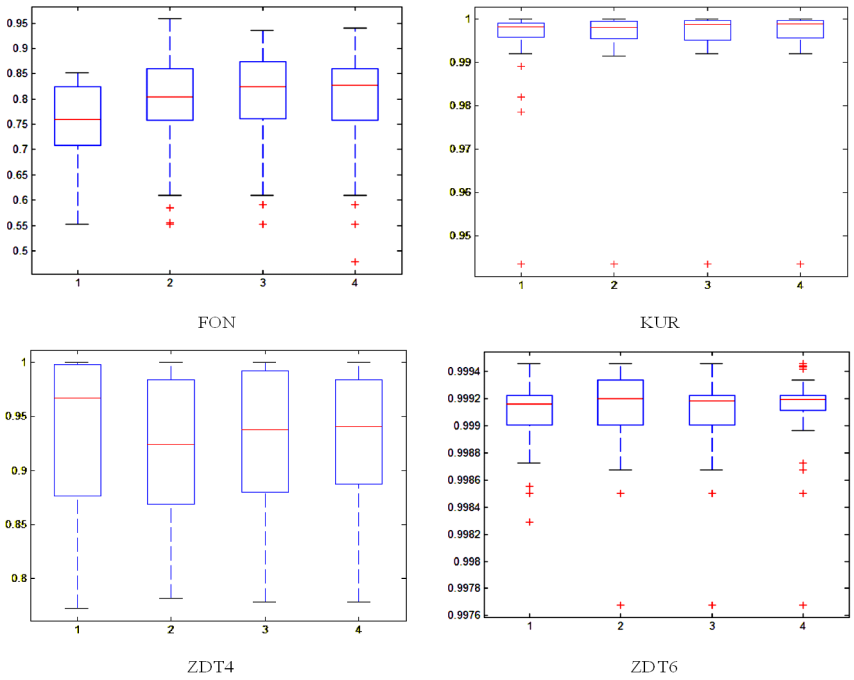

Figure 4.25: Box plots for MS by varying archive size

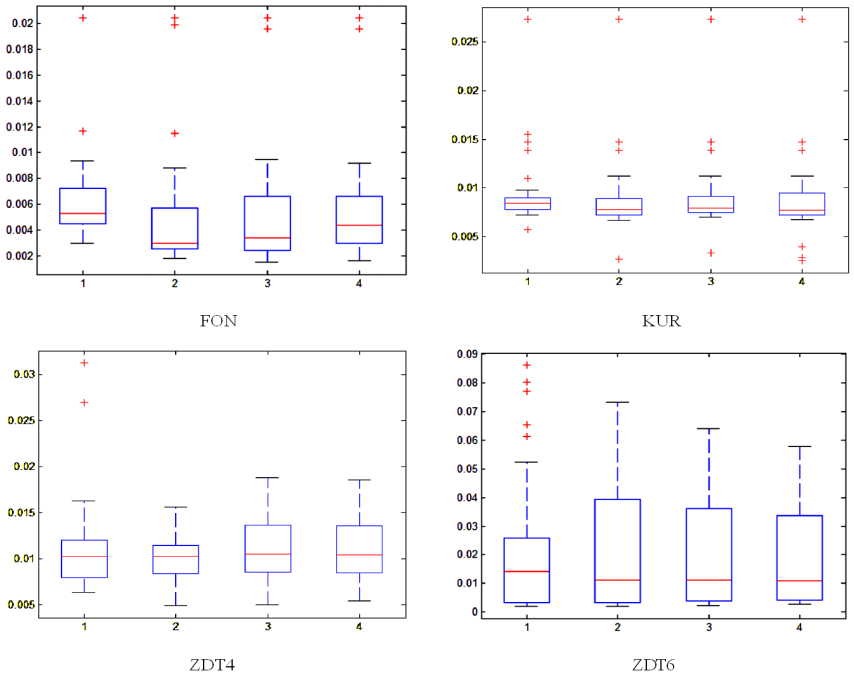

Figure 4.26: Box plots for S by varying archive size

In general, an inertia weight setting of 0.4 provided the best results for CCPSO. In the case of subswarm size setting, 10 is recommended. And as long as computational resources permit, a higher archive size is recommended.

4.5 Conclusion

In the face of increasing complexity and dimensionality of today's application, there is a need to improve the efficiency and effectiveness of MOPSO. Therefore, in order to improve the performance of MOPSO, the algorithm put forward in this chapter attempts to further emulate the conflict and coexistence between cooperation and competition in nature by implementing both aspects into co-evolutionary model. This is accomplished by having species compete amongst themselves for the right to represent different components, and the winners cooperate to solve the whole problem. This proposed algorithm is validated through comparisons with existing state-of-the-art multiobjective algorithms through the use of established benchmarks and metrics for comparison. CCPSO was the only algorithm to attain the true Pareto front in all test problems, and in all cases converged faster to the true Pareto front than any other algorithm used in this research.

Chapter 5

A Distributed Co-evolutionary Particle Swarm Optimization Algorithm

As discussed in Chapter 4, with the growing complexity of multi-objective problems, the computation cost for multi-objective particle swarm optimization algorithm (MOPSO) also increases, and this problem is addressed through the decomposition of a complex problem. In this chapter, the MOPSO is further formulated into a distributed algorithm to utilize the large computing power available in network computers.

As a decentralized and parallel computing method, distributed algorithm uses two or more computers communicating in a network to accomplish a common objective or task. Ideally distributed algorithm is drastically more fault-tolerant and more powerful than the combination of many stand-alone computers. However, if not properly designed, the performance of distributed algorithm may degrade until worse off than its non-distributed algorithm counterpart because the communication and execution cost may outweigh the benefit of distributed algorithm. Large communication and execution cost of distributed algorithm is usually caused by long idle time during synchronization, delayed communication, lost or corrupted packages, and lost connection.

In this chapter, the proposed distributed co-evolutionary particle swarm optimization

algorithm (DCPSO) will show how it becomes more fault-tolerant toward lost connection with the incorporation of a revised competitive co-evolution mechanism. DCPSO also incorporates other mechanisms, for example, request-acknowledge message to handle lost or corrupted packages and dynamic load balancing to minimize the idle time of computational nodes in synchronization.

The remainder of this chapter is organized as follows: Section 5.1 gives a review of existing distributed MO algorithms. Section 5.2 describes the principle of a revised co-evolutionary particle swarm optimization algorithm (CPSO), which is the non-distributed version of DCPSO. In Section 5.3, the proposed DCPSO is explained in detail, which includes an explanation of how DCPSO is particularly resistant and efficient against lost connection and how dynamic load balancing mechanism is implemented. Section 5.4 provides a comparison of CPSO with other MOEAs based upon various benchmark problems. Section 5.5 presents the test simulation on the effectiveness of DCPSO. Conclusions are drawn in Section 5.6.

5.1 Review of Existing Distributed MO Algorithms

In distributed MO algorithm, the task is divided into subtasks that are run concurrently to reduce total runtime. These subtasks may be run in the same computer (multiprocessor system) or different computers connected via network (multicomputer system and computer cluster).

It is a challenging task to successfully design a distributed algorithm. If not designed properly, the distributed algorithm can deteriorate in performance. Two major costs in distributed algorithm are execution time and communication time [209]. To minimize these two costs, five important issues need to be considered in the design of an effective distributed algorithm as shown in the following part.

1) Fault-tolerance of Distributed Algorithm: Distributed algorithm is required to be fault tolerant due to the nature of nondeterministic failure in distributed systems [209]. Network

errors can arise anytime in the system. However, no error should be able to interrupt, clash the system, or prevent the system from reaching its goal. To be able to handle lost connection, corrupted package, and other possible errors, every distributed algorithm must have its own error correction protocol. For example, many algorithms implement a reconnection mechanism after lost connection occurs to continue its operation because losing a computational node causes degradation of searching capability [191] [209].

2) Synchronous and Asynchronous Distributed Algorithm: There are two types of distributed algorithm, synchronous and asynchronous. The synchronous model is easier to visualize and design. Each node will send data and wait for the synchronization of all nodes. During waiting for synchronization, nodes are idle; and this idle time constitutes to resource wastage. In asynchronous model, there is no minimum waiting time for synchronization between nodes. Even though asynchronous model seems to be better in term of efficiency, it is harder to design and implement as the communications occur non-deterministically and message delivery is not totally guaranteed [50] [209].

3) Partitioning and Load Balancing: Since each computational node has different computational power, nodes take different amount of time to execute same task. In order to assign appropriate load to each node, the task is usually partitioned into several subtasks according to computational power of each node [12]. However, as the available computational power is changing all the time, it is essential as well to adjust the load of each node at runtime. Dynamic load balancing will change the workload of each node according to the available computational resource of each node [13].

4) Paradigms: There are four types of paradigms that are widely used to implement distributed algorithms, including master-slave model, island model, diffusion model, and hierarchical hybrid model. These four paradigms will be explained as follows: a) Master-slave model (global parallelization) is easiest to visualize. The master node is responsible for the maintenance of the population, such as performing PSO operation, computing Pareto front and synchronization. Fitness evaluation is distributed among slave nodes. This paradigm

is most suitable with the MO problems that have time-consuming objective function evaluation [159] [209] because objective function evaluation needs to be time-consuming enough for system to realize any speedup. b) Island model (coarse-grained parallelization) separates population into several subpopulations. Each subpopulation evolves its solutions independently. The good solutions may migrate across subpopulations. This model is often viewed as a ring of connected nodes that exchange information through migration of its solutions. The important factors to consider when choosing island model are when the solutions will migrate, the number of migrating solutions, and the communication topology. c) Diffusion model (fine-grained parallelization) is similar to master-slave model but each node only holds on to a few particles. The objective function evaluation will take place concurrently among the neighborhood. The network topology determines the effectiveness of the diffusion model distributed algorithm because the communication cost in this model is high within the neighborhood. Diffusion model is often implemented on a massively parallel machine due to large inter-process communications. d) In hierarchical hybrid model, more than one paradigm is combined to form a hybrid model. The complexity of the model depends on the level of hybridization.

In the following part, more details of some of the existing distributed MO algorithms will be introduced. Existing distributed MO algorithms generally show significant speedup from their non-distributed counterparts by exploiting the parallelism of the searching process.

Kim and Zeigler [98] proposed HDGA (Hierarchical Distributed Genetic Algorithm), which is an asynchronous distributed algorithm based on the hierarchical platform. The algorithm initially starts with high level cluster that is responsible to search for a candidate search space that is promising. When the candidate search space is found, the cluster will spawn a lower level cluster to search further into the assigned space with higher accuracy. This sequence of activities goes on until no promising region is found in the cluster, and the cluster returns its solutions to the parent cluster. HDGA only assigns a cluster to another computational node when promising regions are found and communications between

clusters occur only at clusters initialization and termination; hence, it efficiently minimizes the communication cost. Whenever lost connection occurs, HDGA is required to make reconnection and restart the search at the low level cluster. After terminating the cluster, a large package with all the solutions is sent to the higher-level cluster. If this large package is corrupted or lost, the cluster will resend another large package and that may cause network congestion.

Okuda et al. proposed DCMOGA (Distributed Co-evolutionary Multi-Objective Genetic Algorithm) [138] that adapts a synchronous island model where $n+1$ subpopulations are created (n is dimension of the objective space). The n subpopulations, called SOGAs, are each responsible for optimizing one objective. Another subpopulation (MOGA) tries to find the Pareto optimal solutions. Each SOGA passes the solutions back to MOGA after a few iterations, and it is expired once no further improvement is found. The required number of nodes in DCMOGA is restricted by the dimension of the objective space.

In 2004, Tan et al. [191] implemented DCCEA (Distributed Cooperative Co-evolutionary EA) under synchronous island model. The population is divided into subpopulations, and the objective evaluation and genetic operation are performed concurrently. After several iterations, each subpopulation will synchronize with other subpopulations through sending its non-dominated solutions. DCCEA incorporates partitioning to distribute workload to each computational node appropriately at the initialization stage.

All above mentioned distributed MO algorithms contain behaviors that may degrade their performance if network becomes unreliable. As in DCMOGA and DCCEA, whenever lost connection occurs, both algorithms are required to reconnect to the lost node. Reconnection to nodes introduces extra overhead and availability of that node is not guaranteed. Therefore in this work, the proposed DCPSO will focus on how to become more fault-tolerant toward lost connection with the incorporation of competitive co-evolution mechanism.

5.2 Co-evolutionary Particle Swarm Optimization Algorithm

The co-evolutionary particle swarm optimization algorithm (CPSO) proposed in this chapter is similar to CCPSO in Chapter 4. There are some modifications to make CPSO more suitable for distributed computation.

The main difference is that the assignment of decision variables to different species is adapted by a new competition mechanism. In particular, each species will compete to represent more components of the MO problem while the eventual winners will cooperate to achieve better solutions. In this iterative process of competition and cooperation, various components form different groups, and each group is optimized by one winning species. The formation of the groups and the winning species are changing during the optimization process, which reflects the different optimization requirements in different stages of searching. In addition, the interaction between the cooperative and competitive processes in CCPSO take place iteratively after each cycle, this frequency can be determined by the designer accordingly in CPSO.

5.2.1 Competition Mechanism for CPSO

To minimize the communication overhead among different species, the winning species is directly associated with the variable instead of replacing the losing species. The rationale of adding dimensions to winning species instead of replacing losing species here is that replacing mechanism need to transfer particles among different subswarms through network, while adding dimensions is just implemented inside a particle. The network transfer speed is the bottleneck for distributed algorithms and the network throughput should be minimized. For the ensuing discussions, the problem at hand is decomposed along the decision variables. Also, each decision variable may be assigned to a number of species populations and a species population may be optimizing more than one decision variable.

Comptitive Process

S_i : i-th subswarm
$s_{j,rep}$: representative for variable j

for each subswarm $S_i, i \in [1..n]$
  for each $s_{j,rep}, j \in [1..n], j \neq i$
    Select competing particles from S_i
    Competition
    If competitor win
      associate variable j with subswarm S_i
      update $s_{j,rep}$
    elseif variable j is associated with subswarm S_i
      dissociate variable j with S_i
    endif
  endfor
endfor

Figure 5.1: Pseudocode for the competitive coevolutionary mechanism in CPSO

The pseudocode of the competitive mechanism of CPSO is shown in Figure 5.1. To discover the most suitable problem decomposition, competitors are selected to compete with each $s_{j,rep}$. For the i-th subswarm, the representatives for other variables $s_{j,rep}, j \neq i$ is selected along with competitors from subswarm i for pair competition. With regard to the issue of competitor selection, CPSO simply selects a random particle from subswarm i for each $s_{j,rep}, j \neq i$. These competitors will then compete via the cooperative mechanism to determine which variable should also be associated with subswarm i. The variables where the particles from subswarm i win the competition will be associated with subswarm i. The $s_{j,rep}$ which lose the competition will be updated by the new winner.

As illustrated by the algorithmic flow of the proposed CPSO in Figure 5.2, the competitive mechanism is activated at a fixed frequency of $C_{freq} = 10$. Note that the subswarms are evolved in isolation for both competitive and cooperative processes. The algorithm employs a fixed-size archive to store non-dominated solutions along the evolution. The archive is updated within the cooperative or competitive process. A complete solution formed by the

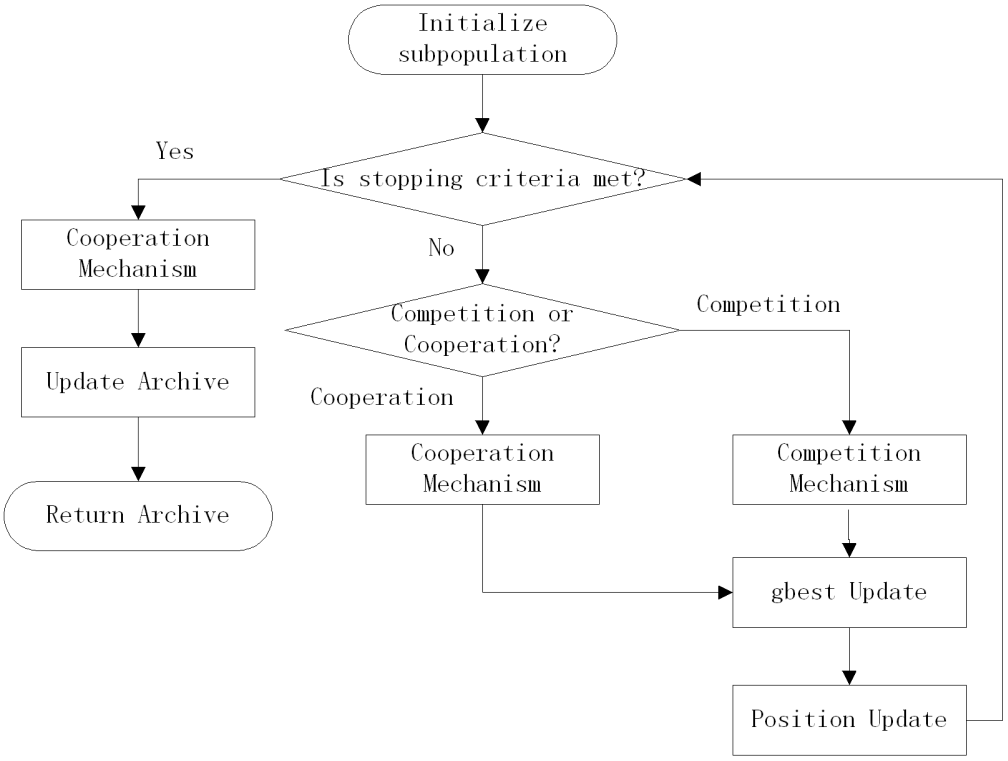

Figure 5.2: Flowchart of CPSO

subswarms will be added to the archive if it is not dominated by any archived solutions. Likewise, any archive members dominated by this solution will be removed. When the predetermined archive size is reached, a recurrent truncation process [96] based on niche count is used to eliminate the most crowded archive member.

5.3 Distributed Co-evolutionary Particle Swarm Optimization Algorithm

The availability of powerful networked computers presents a wealth of computing resources to solve problems with large computational effort. As the communication level in coarse-grained parallelization is low as compared to other parallelization strategies, it is a suitable computing model for distributed networks with limited communication speed. This paral-

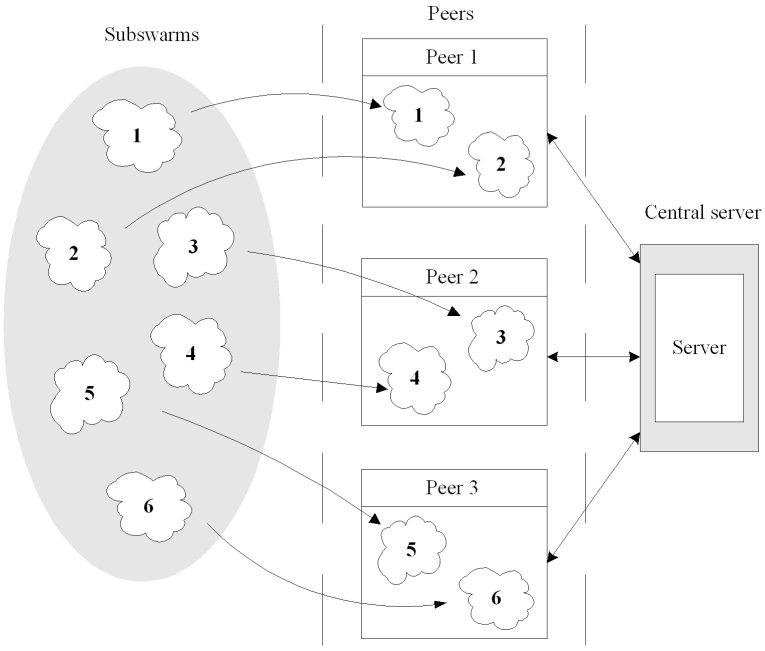

Figure 5.3: The Model of DCPSO

lelization approach is considered here, where large problems are decomposed into smaller subtasks that are mapped into the computers available in a distributed system.

A prototype model of DCPSO with six subpopulations over three peer computers is illustrated in Figure 5.3. As shown in the figure, subswarms are partitioned into groups depending on the number of available peers. Without loss of generality, the six subswarms are divided into three groups, each of which is assigned to a peer computer. Note that each peer contains its own archive and representatives, which evolves its subswarms sequentially in an approach similar to CPSO.

Inside a peer, the candidate solutions generated through the collaboration consistently update the archive in the peer. The subswarms in the peer then update their corresponding peer representatives once in every cycle. The cooperation among peers is indirectly achieved through the exchange of archive and representatives between the peers and a central server. In DCPSO, the communication time among peers is a conspicuous part of the whole execution time. To reduce the communication overhead, the exchange of archive and

representatives between a peer and the central server only occurs every several cycles as determined by the exchange interval. Since the peers are often not identical, the cooperation among peers will become ineffective if there is a big difference for the evolution progress among the peers, e.g., the poor cooperation among peers may deteriorate the performance of DCPSO. To make the peers cooperate well in the optimization process, the peers are synchronized every several cycles as determined by the synchronization interval. Note that the exchange and synchronization intervals in DCPSO can be fixed or computed adaptively along the search.

In DCPSO, there is no direct communication among the subswarms. All communications are performed between the subswarm and the central server. For the subswarm, the communication overhead does not change. For the central server, the communication overhead increases linearly with respect to the number of computers, which is not a big issue because the central server is dedicated to communication and generally can deal with the communication amount in DCPSO.

5.3.1 Implementation of DCPSO

The DCPSO has been designed and embedded in a distributed computing framework that was built upon the foundation of Java technology (J2EE). The unique advantage of Java programming language, such as platform independence and reusability, makes the approach attractive.

As shown in Figure 5.4, DCPSO framework consists of two main blocks, i.e., the servant block and workshop block that are connected by remote method invocation over the internet inter-ORB protocol (RMI-IIOP). The servant functions as an information center and backup station through which peers can check their identifications or restore their working status. The workshop is a place where peers (free or occupied) work together in groups, e.g., the working peers are grouped together to perform a specified task, while the free ones wait for new jobs to be assigned. The servant contains three different servers, i.e., logon

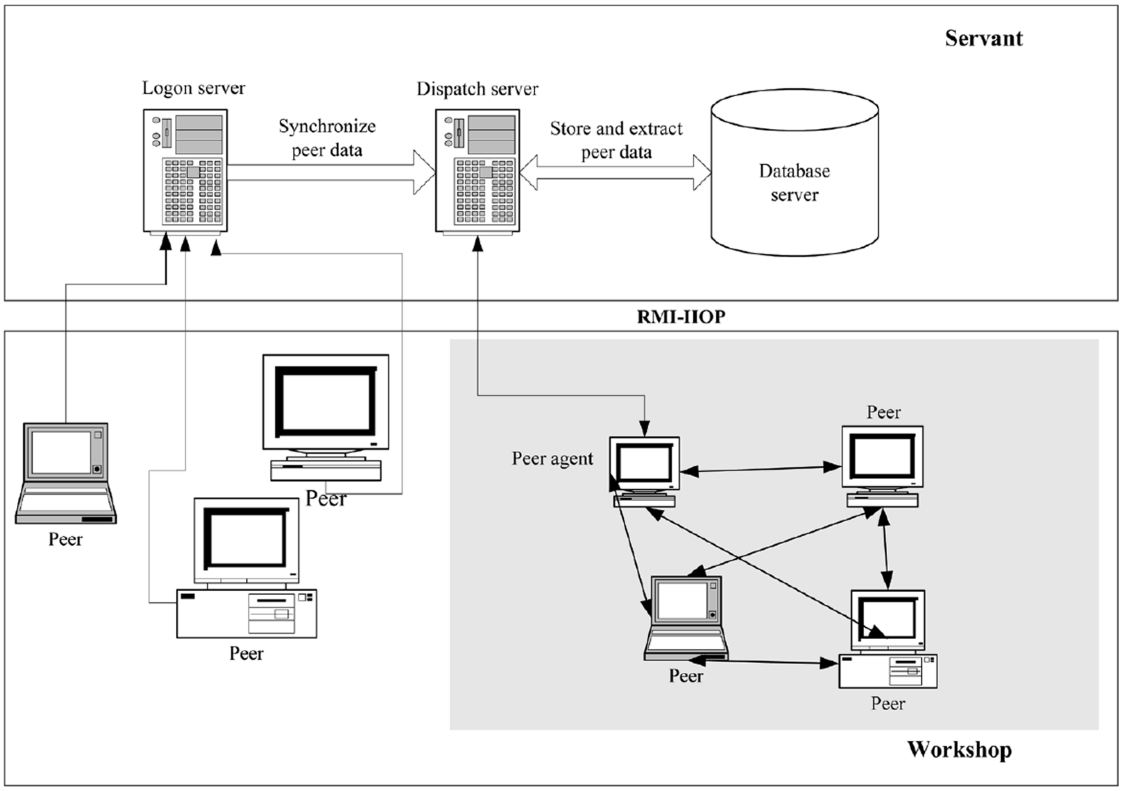

Figure 5.4: Schematic framework of DCPSO

server, dispatcher server, and database server. The logon server assigns identification to any registered peers. It also removes the information and identification of a peer when it is logged off as well as synchronizes the peers information to the dispatcher server. The dispatcher server is responsible for choosing the tasks to be executed and the group of peers to perform the execution. It also handles the transfer of peers information to/from the database server. Besides, the dispatcher server also synchronizes the information, updates the peers list, and informs the database server for any modification. Whenever a task is available, the dispatcher server will transfer the task to a group of selected peers.

The working process of a peer begins when it logs onto the server. The peer computers are then pooled and wait for the tasks to be assigned by the server. Once a peer detects that a task has been assigned, it will extract relevant information from the server, such as

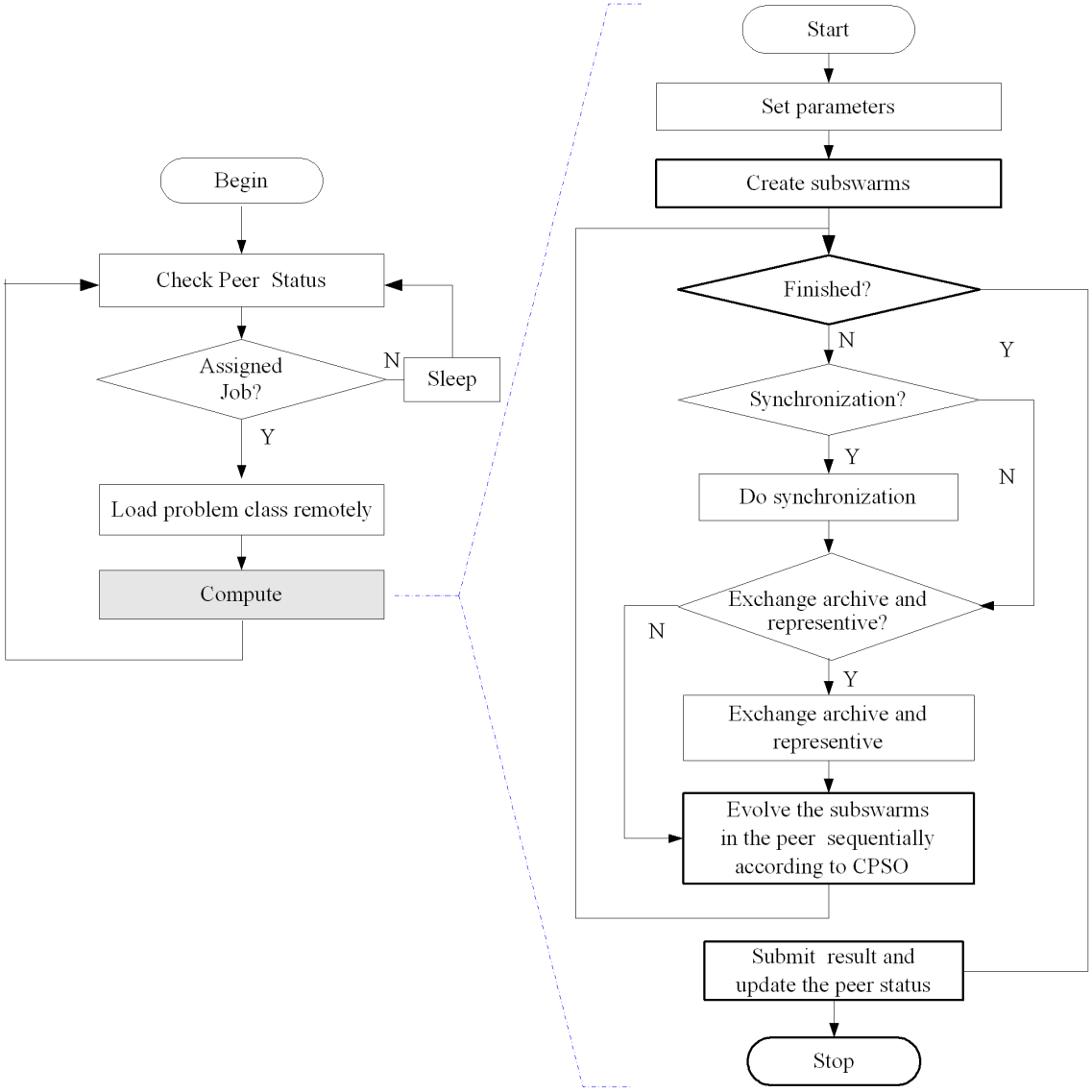

Figure 5.5: The flowchart of DCPSO

class name and path, as well as the http server address before loading the class remotely from the server. Then it will be allowed to initiate the computation procedure. Figure 5.5 depicts the working process of a peer, where the description of the module "Compute"is shown in the right part of the figure.

Once a peer starts the computation procedure, it first initializes the parameters, such as cycle number, number and size of subswarms. The peer then creates the subswarms assigned

to it. When it reaches a synchronization point, it suspends its searching process until the server signals that all the peers have reached the synchronization point. At each cycle, the peer checks whether it should exchange the archive and representatives between the peer and server. If the conditions of exchange are satisfied, the peer initiates a session in the server and exchanges/updates the archive and representatives between the peer and server. The peer then evolves its subswarms sequentially for one cycle after these procedures. If the peer meets the termination conditions, it will initiate a session to submit the results and restores itself to the ready status. If the user cancels a running job, those peers involved in the job will stop the computation and set themselves to the ready status.

5.3.2 Dynamic Load Balancing

As the processing power and specification for various computers in a network may be different, the feature of workload balancing that ensures the peers are processed in a similar pace is important in DCPSO. Besides having a long simulation time, the poor cooperation among computers will also deteriorate the performance of DCPSO if a heavy workload is assigned to the peer that has a relatively low processing power. Intuitively, workload balancing for a distributed system could be difficult because the working environment in a network is often complex and uncertain.

DCPSO assigns workload to peers according to their respective computational capabilities. When a peer is first launched, it uploads its configuration information to the server, including hardware configuration of the peer such as CPU speed, RAM size, etc. Based on the information, the dispatch server performs a simple task scheduling and assigns different tasks to the respective peers according to their processing powers. A dynamic load balancing strategy is also implemented to adjust the workload of each Peer at runtime using the response time of each peer. If a peer is found to be lagging behind a few times without improving, dynamic load balancing mechanism will signal the lagging peer to reduce its

workload accordingly. If a peer is found to be lagging severely, it will be removed from the system immediately.

5.3.3 DCPSO's Resistance towards Lost Connections

As there are various unforeseen network problems, it is important to construct a distributed algorithm that is fault-tolerant. To be able to handle unforeseen network errors, DCPSO adopts the request-and-response package transmission protocol that guarantees that packages are sent successfully. This mechanism will avoid damaged, corrupted, or lost packages. But it can not handle lost connection efficiently. With the competition mechanism, DCPSO can cope well with disconnected peer as reconnection with lost peer is not necessary when lost connection occurs. With competition mechanism, even if lost connection occurs and the subswarm representing k-th variable terminates, competition from other subswarms will continue to improve the value of the k-th variable. Test simulation on the effectiveness of the competition mechanism in Section 5.5 will illustrate that the proposed algorithm is able to derive good results even if some subswarms are missing.

5.4 Simulation Results of CPSO

In this section, five benchmark problems (ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6) are used to compare the performance of CPSO with CCEA [194] and SPEA2 [224]. 10 independent runs are executed on these three MO algorithms. The configuration of the algorithms is shown in Table 5.1, and the results of the simulation are shown in Figure 5.6-5.10.

Four performance metrics (GD, S, MS, and HVR) are used to analyze the effectiveness and efficiency of the three algorithms. CPSO is able to approximate and locate the true Pareto front effectively in all test problems, except in ZDT3. As the true Pareto front of ZDT3 is a discontinuous convex curve, it can be deduce that CPSO has some difficulty in dealing with problem that has discontinuous Pareto front.

Table 5.1: Parameter settings of the different algorithms

Parameter	Settings
Population	CPSO: subswarm size 20
	CCEA: subpopulation size 20
	SPEA2: population size 100
Archive Size	100
CPSO Configuration	$C_1 = C_2 = 1$
	Inertia weight 0.4
EA Configuration	30 bits per decision variable
	Crossover rate 0.8
	Mutation rate $2/L$, where L is chromosome length
Evaluations	120 000

Concerning generational distance, CPSO obtains very competitive results, which are far better than that of SPEA2. GD results of CPSO are also better than that of CCEA in ZDT4 and ZDT6. Strength in GD highlights that CPSO is effective in escaping local minima and efficient in converging to true Pareto front. The competition mechanism may contribute a lot to the strength of CPSO in locating true Pareto front.

The spacing results of CPSO in all test problems are poor comparing to other algorithms. Poor spacing results show that CPSO is not able to distribute the solutions evenly along Pareto front, which may be a side-effect from the competition mechanism because competition leads to arm-race that only promotes fitter solutions without considering the distribution of solutions.

The MS of CPSO are much better than that of the other two MOEAs on ZDT1 and ZDT2, while remaining competitive in other test problems. This result illustrates that CPSO is effective in covering the whole range of true Pareto front.

The simulation results from GD, S and MS show that CPSO is effective in covering the range of true Pareto front, but poor in achieving good distribution. From the definition

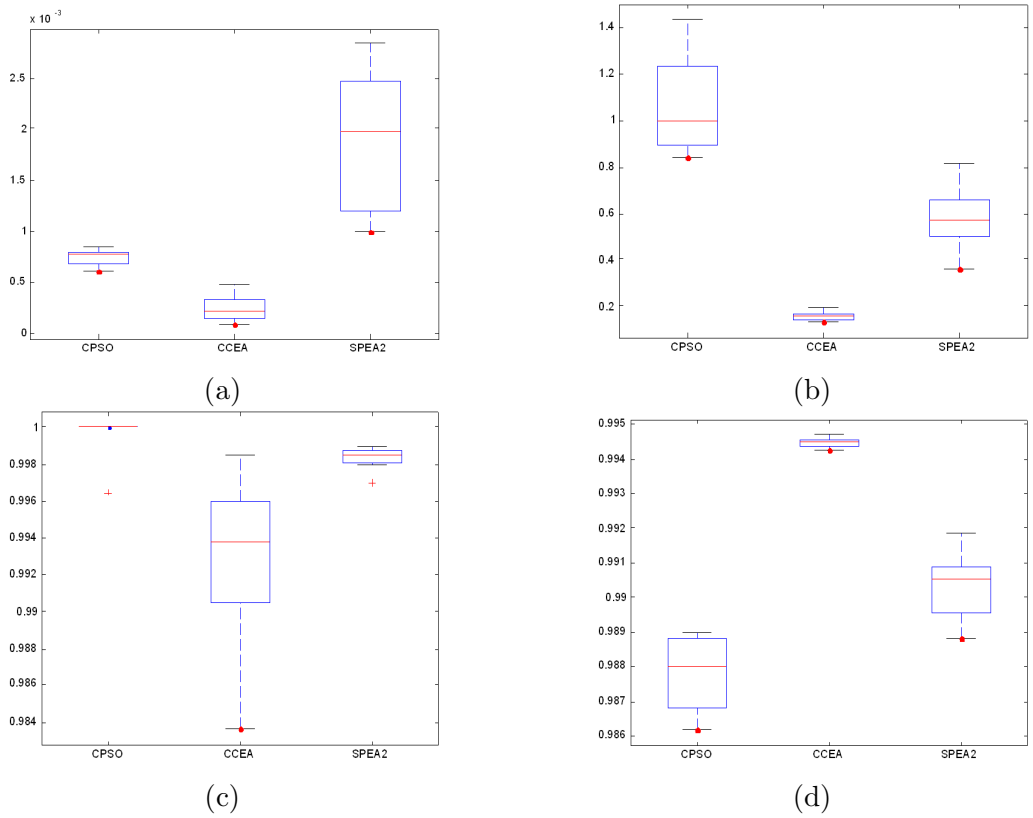

Figure 5.6: Performance comparison of CPSO, CCEA and SPEA2 on a) GD, b) S, c) MS, d) HVR for ZDT1

of the four metrics as shown in Chapter 1, there is no correlation among the metrics of GD, S, and MS, while HVR generally correlates to all the other three metrics. The best performance on GD and the relatively good value of S and MS resulted in the best value of HVR for CPSO on ZDT4. HVR is the metric to measure both evenness and range of evolved Pareto front, and the results of CPSO are relatively mediocre comparing to HVR results of the other two MOEAs for other test problems as CPSO behaves poor in spacing metric.

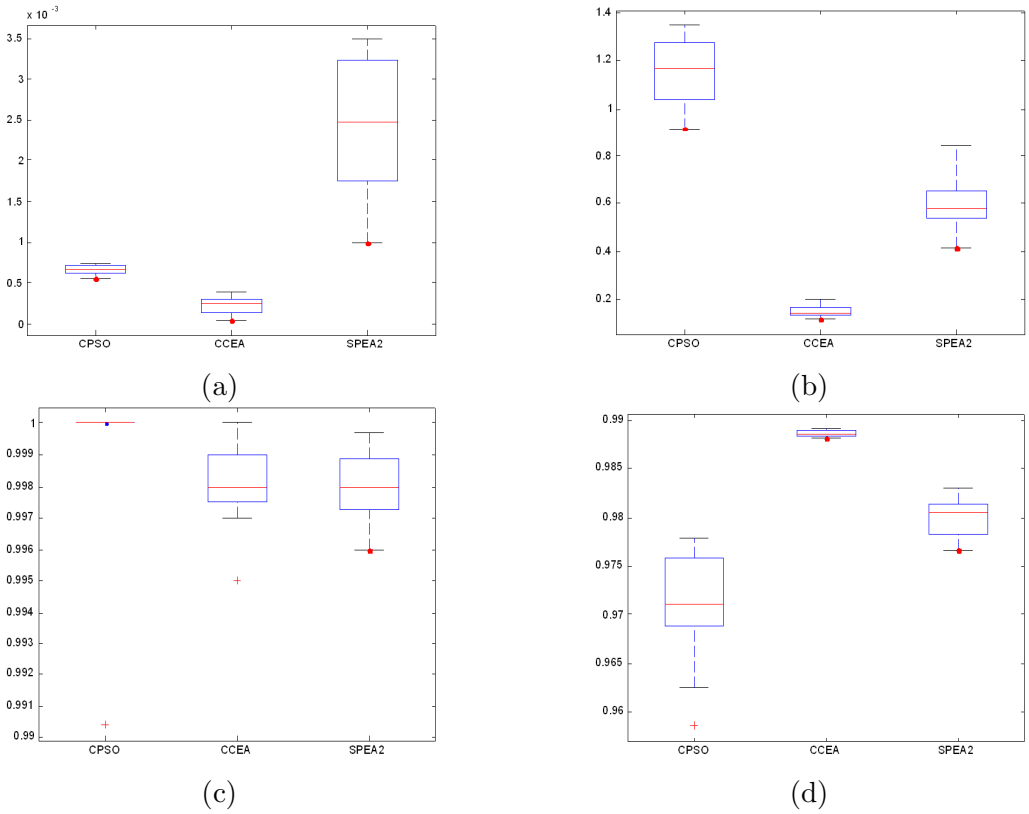

Figure 5.7: Performance comparison of CPSO, CCEA and SPEA2 on a) GD, b) S, c) MS, d) HVR for ZDT2

5.5 Simulation Studies of DCPSO

This section shows simulation studies of DCPSO to examine its efficiency of parallel implementation. The configuration of DCPSO consists of 11 PCs connected via a campus local area network. Table 5.2 represents the specifications of all PCs used in the simulation environment. One fastest PC is used as the server, and the other PCs are peers. The parameter setting of DCPSO is summarized in Table 5.3.

5.5.1 DCPSO Performance

In this section, simulations are performed to examine the speedup of DCPSO in program execution based upon the test problems of ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6. The

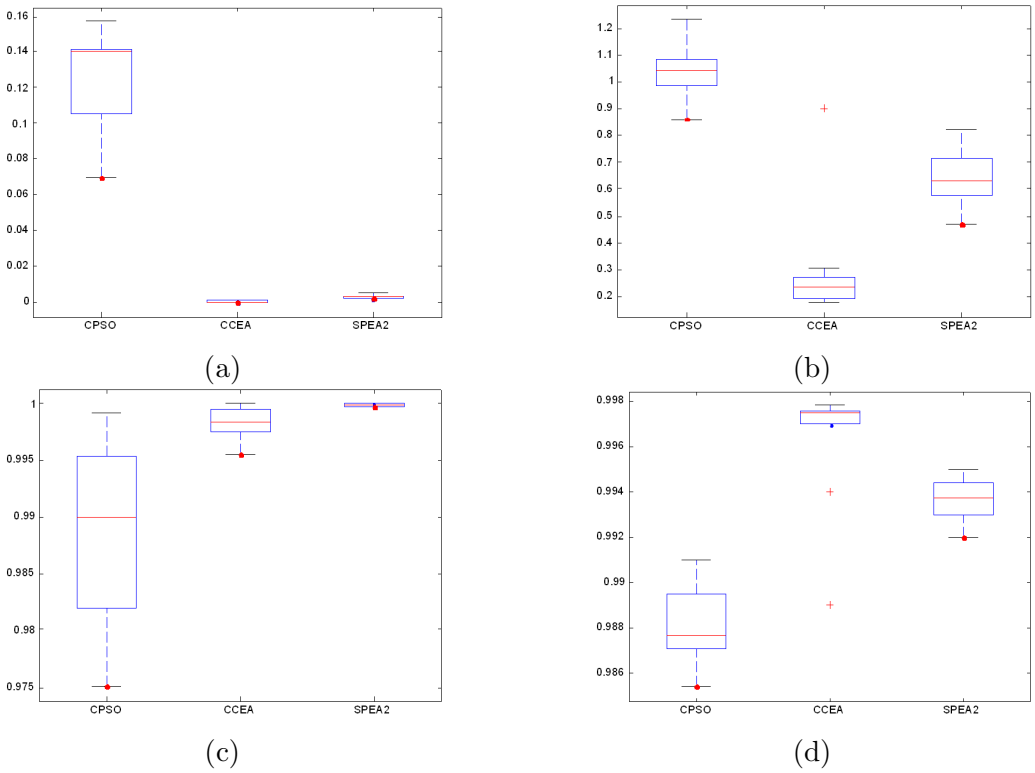

Figure 5.8: Performance comparison of CPSO, CCEA and SPEA2 on a) GD, b) S, c) MS, d) HVR for ZDT3

performance of DCPSO is assessed by comparing the total runtime of DCPSO with 1-10 Peers. To minimize the bias in the simulation results, 10 independent runs with randomly generated initial swarms were performed. The median speedup of the 10 runs for different test problems is listed in Table 5.4, while the actual median runtime for each benchmark problem is plotted in Figure 5.11.

It is observed from Figure 5.11 that the median runtime is reduced as the number of peers is increased. Table 5.4 illustrates that the speedup achievable by DCPSO depends on both the number of peers and the benchmark problem. For instance, the speedup achievable is more significant for large problems according to the simulations. In particular, DCPSO achieves a speedup of 6.83 for ZDT1, but only a speedup of 1.55 is obtained for ZDT6. When the number of peers is more than 8, the increased communication cost counteracts the

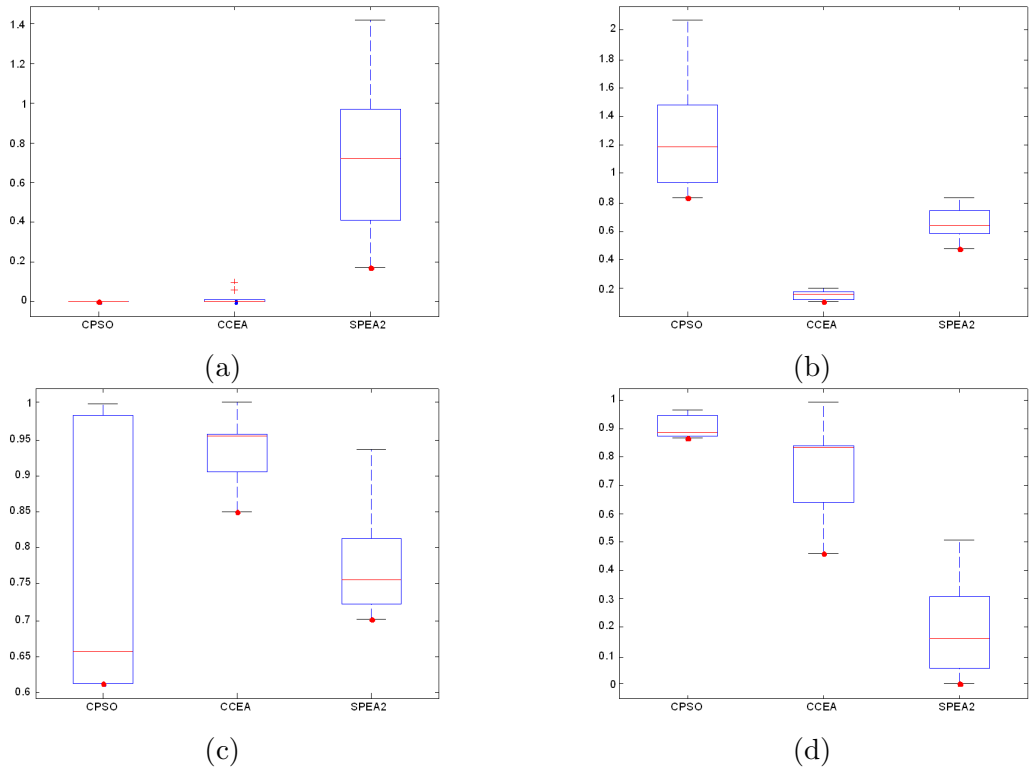

Figure 5.9: Performance comparison of CPSO, CCEA and SPEA2 on a) GD, b) S, c) MS, d) HVR for ZDT4

reduction of computation cost, as shown in Figure 5.11, i.e., the saturation for the speedup of DCPSO is reached for the test problems. The speedup is initially large (maximum 50%) and saturate as more peers are added.

Observing the result more carefully, it will be realized that the communication cost begins to outweigh the benefit of distributing the workload when there are more than 8 peers in ZDT1, ZDT2, ZDT3 and ZDT4 and more than 4 peers in ZDT6. When the communication cost begins to outweigh, the total runtime will start to saturate and there is only small decrease in runtime or even slight increase as more peers are added.

5.5.2 Effect of Dynamic Load Balancing

In this section, the DCPSO with and without dynamic load balancing is tested to see whether dynamic load balancing can increase the efficiency of the algorithm. The simulation is run

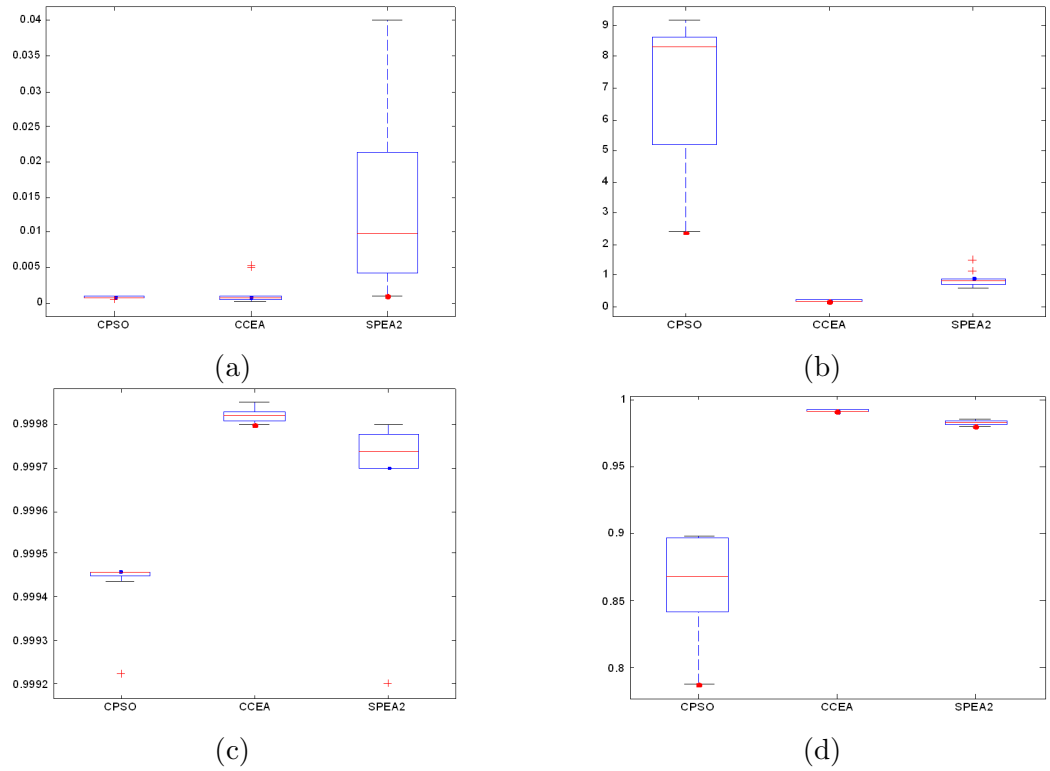

Figure 5.10: Performance comparison of CPSO, CCEA and SPEA2 on a) GD, b) S, c) MS, d) HVR for ZDT6

on the case of DCPSO with 3-10 peers for 10 runs to remove any bias. Average runtime of DCPSO with and without dynamic load balancing for ZDT1 to ZDT6 are shown in Figure 5.12-5.16. All the simulation results are also listed in Table 5.5.

As shown in Figure 5.12-5.16, DCPSO with dynamic load balancing mechanism achieves a significant speedup but saturation of speedup is still present as more peers are added. The speedup ranges from $40\% - 60\%$. Thus, unbalanced load assignment between peers lead to significantly longer runtime in synchronous distributed algorithm. In ZDT1, ZDT2, and ZDT3 cases, the runtime initially are reduced by more than 60% when dynamic load balancing is implemented. As more peers are added, the speedup becomes less because less workload is being distributed to each Peer. With small workloads there is small difference of execution time between fast and slow peers. In ZDT4 and ZDT6, the speedups are generally 40%-50% when dynamic load balancing is implemented.

Table 5.2: Specifications of the PC peers

No. of Peer	Specification
1	3GHz Pentium IV RAM 512MB
2	3GHz Pentium IV RAM 512MB
3	3GHz Pentium IV RAM 512MB
4	3GHz Pentium IV RAM 512MB
5	2.5GHz Pentium IV RAM 256MB
6	2.5GHz Pentium IV RAM 256MB
7	2.5GHz Pentium IV RAM 256MB
8	2.5GHz Pentium IV RAM 256MB
9	2.5GHz Pentium IV RAM 256MB
10	2.5GHz Pentium IV RAM 256MB
11	1.6GHz Pentium IV RAM 512MB

Table 5.3: Configuration of the DCPSO simulation

Parameter	Settings
Population	Number of subpopulations = number of parameters
	Subpopulation size is 20 particles
Archive	Archive size is 100
Collaboration interval	5 iterations
Evaluations	120,000

5.5.3 Effect of Competition Mechanism

The competition mechanism is tested on whether it can prevent degradation of effectiveness when there is a reduction of the subswarms (peers). The model of competitive-cooperation among subswarms is an important mechanism for DCPSO to handle lost peers. In this section, the number of objective function evaluations is kept constant, when the number

Table 5.4: Average speedup of DCPSO for test problems and respective no. of peers

No. of Peers	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
2	2.01	2.03	1.95	1.23	1.26
3	2.31	2.18	2.51	1.29	1.33
4	2.74	2.36	2.67	1.39	1.55
5	3.16	2.41	2.88	1.37	1.53
6	4.28	3.96	3.98	1.44	1.48
7	4.97	5.38	5.68	1.52	1.49
8	6.40	6.31	6.21	1.58	1.49
9	6.83	5.31	6.59	1.56	1.47
10	6.53	6.61	6.55	1.54	1.46

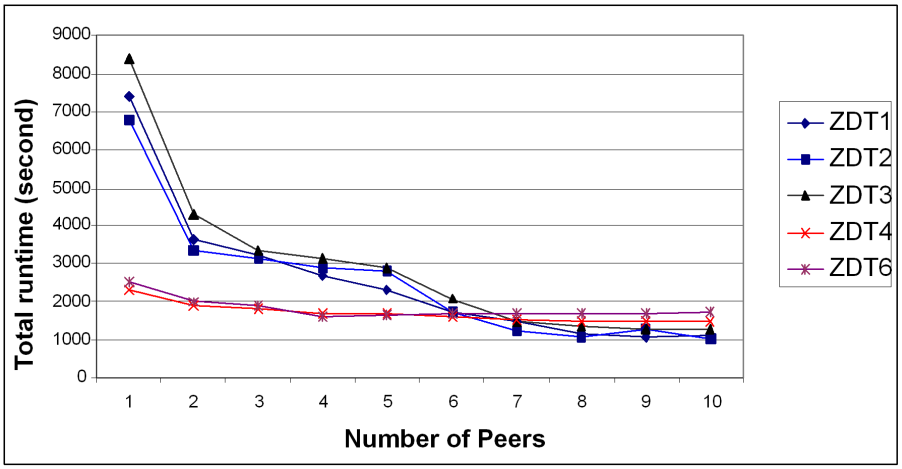

Figure 5.11: Average runtime (in seconds) of DCPSO of five test problems and respective no. of peers

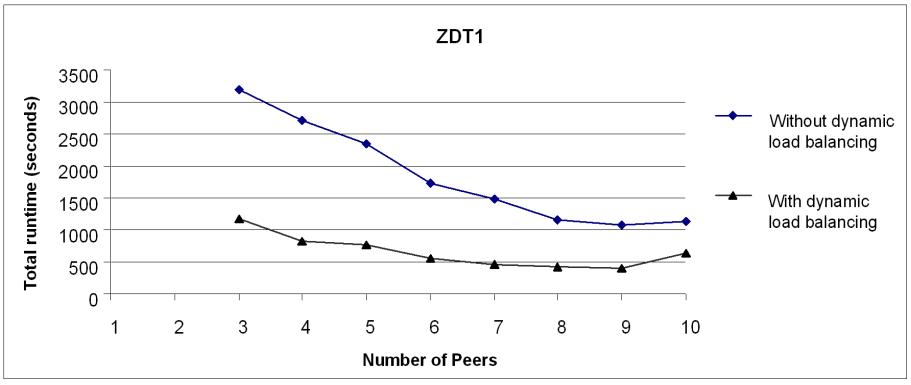

Figure 5.12: Total average runtime of DCPSO with dynamic load balancing and without dynamic load balancing for ZDT1

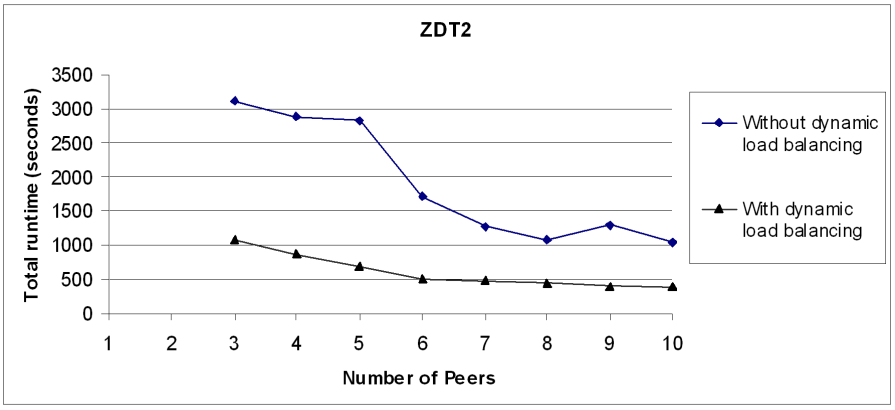

Figure 5.13: Total average runtime of DCPSO with dynamic load balancing and without dynamic load balancing for ZDT2

of subswarm is reduced. Each test problem will be run 10 times with full subswarms, 10 times with 2/3 of subswarms, and 10 times with 1/3 of subswarms. On Figure 5.17-5.20, simulation results from full subswarms are on the left, from 2/3 of subswarms are in the middle, and from 1/3 of subswarms are on the right.

For generational distance, the results from the three scenarios for all test problems have small differences, and the evolved Pareto front is still close to the true Pareto front even when there is a reduction of subswarms. The reduction of number of subswarms does not have significant influence on DCPSO's performance on GD.

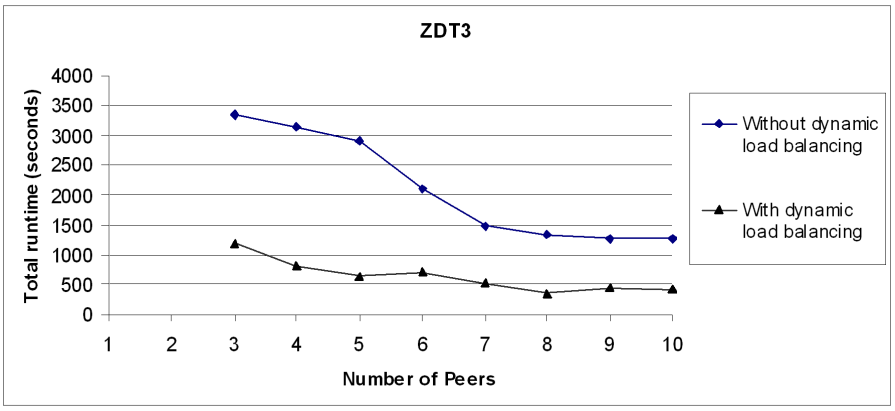

Figure 5.14: Total average runtime of DCPSO with dynamic load balancing and without dynamic load balancing for ZDT3

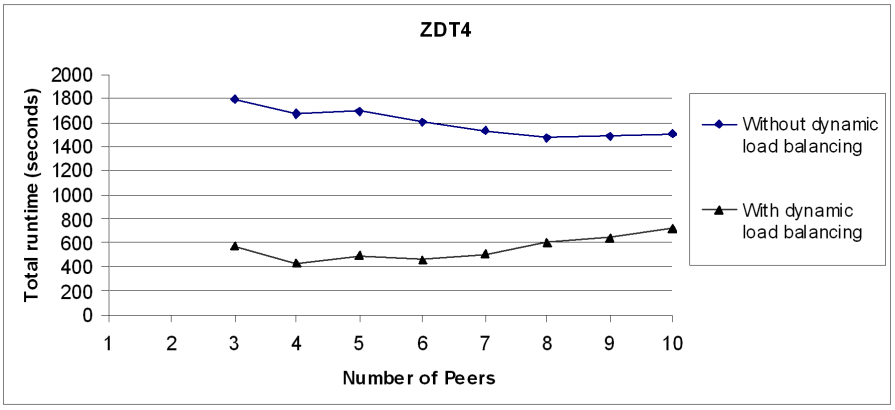

Figure 5.15: Total average runtime of DCPSO with dynamic load balancing and without dynamic load balancing for ZDT4

Concerning spacing metric, results from full subswarms, 2/3 of subswarms, and 1/3 of subswarms are approximately the same. In ZDT1, ZDT2, and ZDT3, the reduction of number of subswarms causes slight degradation of spacing metric. But, there are small improvements in ZDT4 and ZDT6, when the number of subswarms is reduced. Therefore, the reduction of subswarms does not result in poorer distribution of solutions along Pareto front.

For maximum spread, reduction of subswarms does not illustrate any significant negative effect. In all test problems except ZDT6, DCPSO is able to get similar result on MS for all

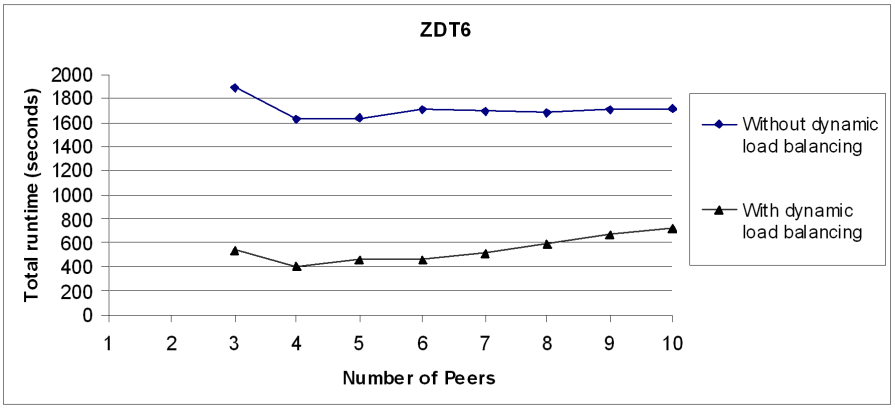

Figure 5.16: Total average runtime of DCPSO with dynamic load balancing and without dynamic load balancing for ZDT6

Table 5.5: Total average runtime of DCPSO with dynamic load balancing and without dynamic load balancing for ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6

No. of Peers		3	4	5	6	7	8	9	10
ZDT1	w/o	3199	2702	2342	1728	1489	1156	1083	1134
	w	1174.00	830.00	764.00	557.00	455.00	432.00	396.00	626.00
ZDT2	w/o	3199	2877	2824	1718	1265	1079	1282	1030
	w	1082.00	867.00	701.00	506.00	489.00	436.00	410.00	387.00
ZDT3	w/o	3339	3143	2910	2107	1479	1351	1273	1282
	w	1191.00	810.00	637.00	707.00	528.00	362.00	432.00	428.00
ZDT4	w/o	1792	1672	1689	1612	1529	1470	1491	1503
	w	573.00	424.00	497.00	463.00	505.00	600.00	648.00	725.00
ZDT6	w/o	1886	1625	1639	1703	1690	1685	1704	1720
	w	537.00	402.00	459.00	459.00	510.00	594.00	666.00	725.00

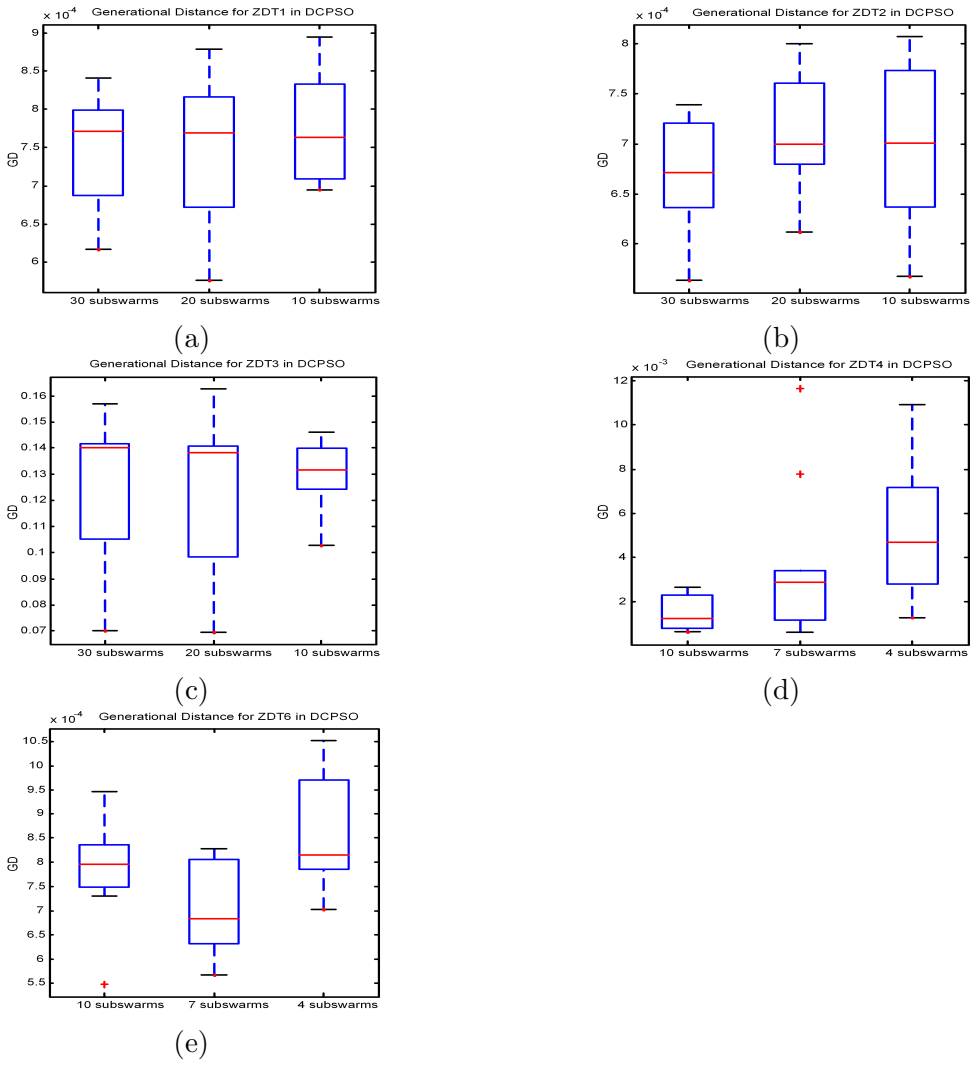

Figure 5.17: Performance comparison of DCPSO over different size of subswarms in GD for a) ZDT1, b) ZDT2, c) ZDT3, d) ZDT4, e) ZDT6

three scenarios. In ZDT6, the reduction of subswarms causes a decrease in value of maximum spread metric but the decrease is very small (observing the y-axis interval). Therefore, one can conclude that competition mechanism help DCPSO cover the true Pareto front well even when the number of subswarms is reduced.

There are slight degradation on hypervolume ratio metric in all test problems except in ZDT4. In ZDT4, the HVR in all three scenarios are equivalent. In other test problems, there are slight decreases in HVR (largest in ZDT6 because the Pareto optimal solutions of

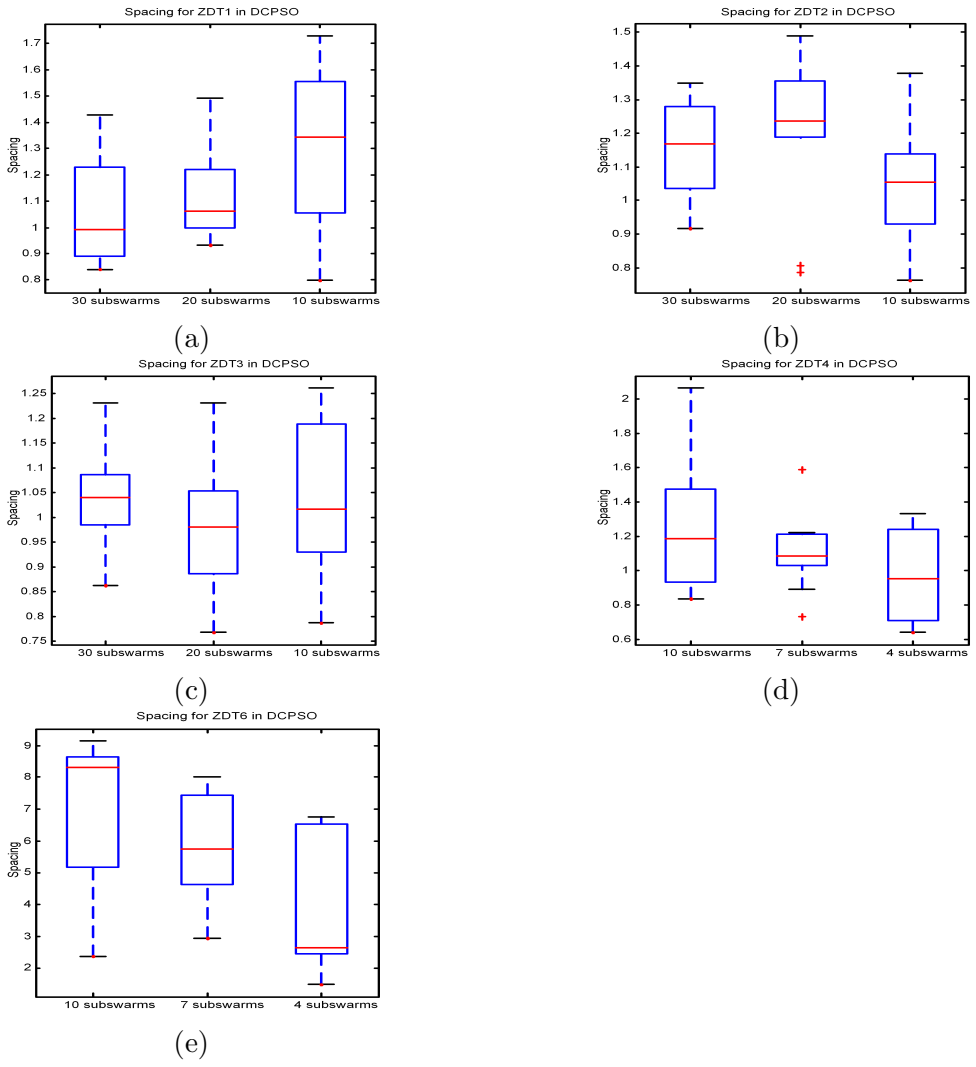

Figure 5.18: Performance comparison of DCPSO over different size of subswarms on Spacing for a) ZDT1, b) ZDT2, c) ZDT3, d) ZDT4, e) ZDT6

ZDT6 are non-uniformly distributed along the global Pareto front, and the density of the solutions is low near the Pareto front and high away from the front).

Comparison of the four metrics in all three scenarios highlights that the competition mechanism is able to prevent the degradation of performance when the number of subswarms is reduced. The results confirm that the competition mechanism is able to obtain Pareto optimal solutions even when some subswarms are missing. Therefore with competition mechanism, when there are lost peers, reconnection is not necessary. DCPSO will recalculate

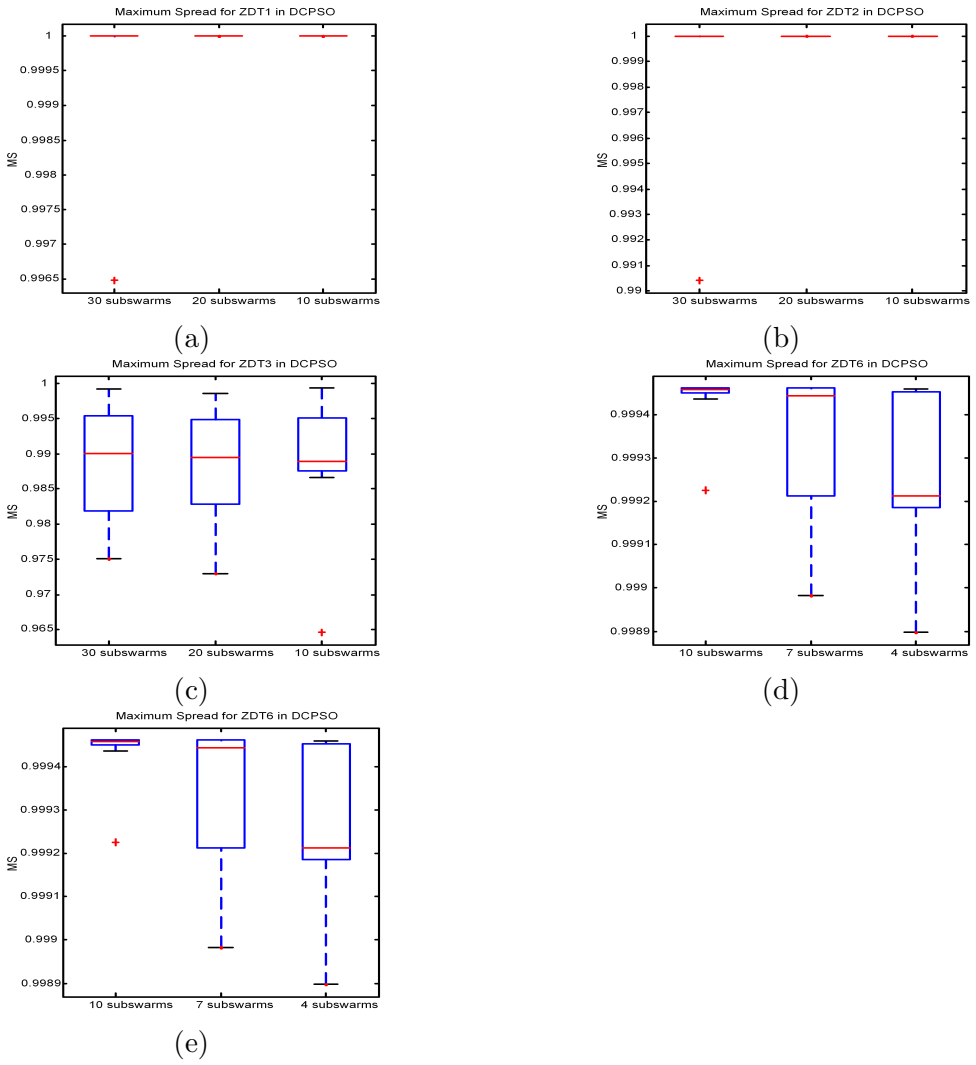

Figure 5.19: Performance comparison of DCPSO over different size of subswarms on Maximum Spread for a) ZDT1, b) ZDT2, c) ZDT3, d) ZDT4, e) ZDT6

its total number of cycles (objective function evaluation is kept constant) according to the number of active peers, and no large communication cost and complication are introduced, which leads to more efficient distributed system. DCPSO can neglect disconnected peers and still be able to achieve good performance.

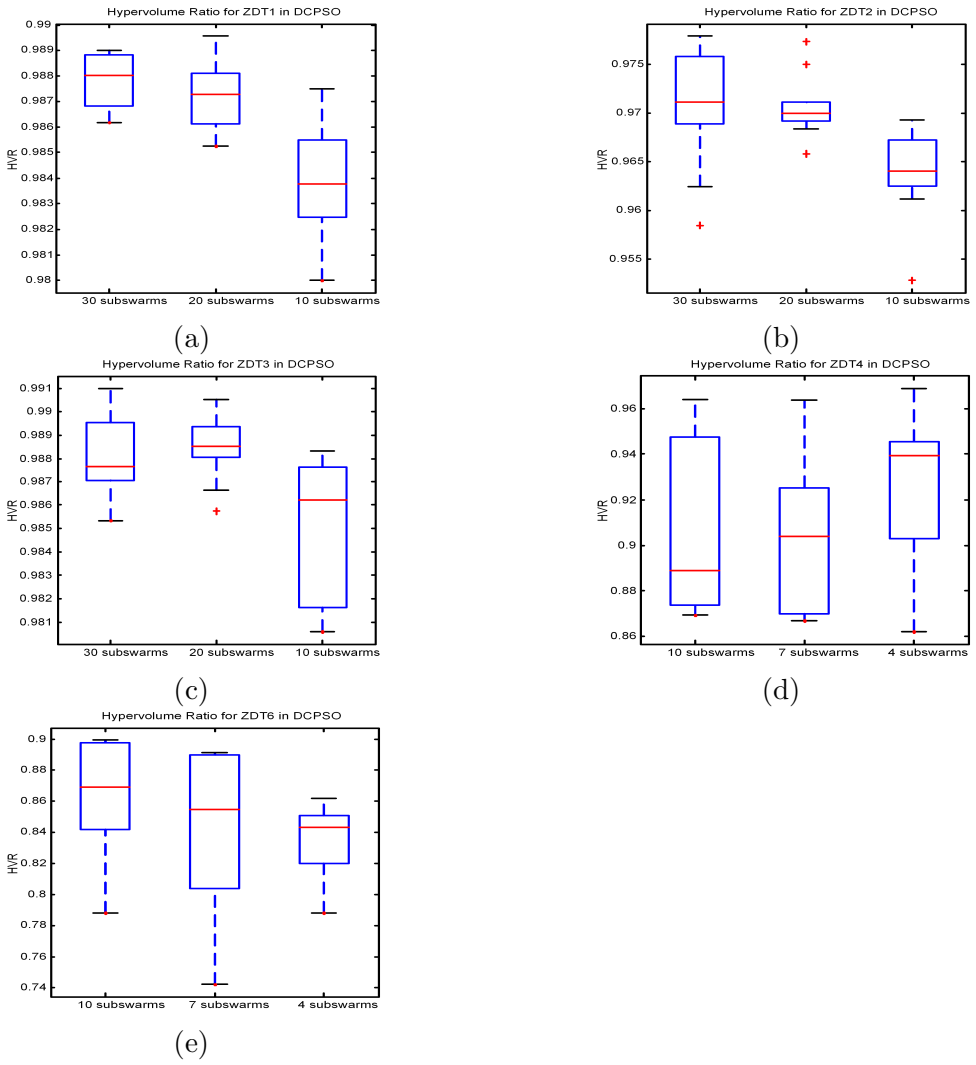

Figure 5.20: Performance comparison of DCPSO over different size of subswarms on Hypervolumn Ratio for a) ZDT1, b) ZDT2, c) ZDT3, d) ZDT4, e) ZDT6

5.6 Conclusion

MO problems often involve large computational resource and high complexity. As the availability of powerful networked computers presents a wealth of computing resources to solve problems with large computational effort, it is desirable to utilize PSO's intrinsic parallel property to formulate a distributed MOPSO algorithm. The proposed distributed co-evolutionary particle swarm optimization algorithm achieves substantial speedup from

non-distributed CPSO. The DCPSO has been designed to be fault-tolerant against network errors, such as lost package, corrupted package, and lost connection. A dynamic load balancing mechanism has also been incorporated into the DCPSO to further enhance its performance.

Chapter 6

On Solving Multiobjective Bin Packing Problems Using Evolutionary Particle Swarm Optimization

The bin packing problem is a NP-hard combinatorial optimization problem [5] where the primary aim is to pack a finite number of items using the least bins possible. It also involves a number of practical constraints and requirements to be satisfied, such as weight, centre of gravity, irregularly shaped bins and priority items. The bin packing problem has been widely studied due to its various applications in computer network design [20], job scheduling [206], industrial balancing [14], aircraft container loading [128] etc.

Many computer-assisted approaches have been developed to solve bin packing problems. Regarding waste space minimization for one to three dimensional bin packing problems, exact solution methods based on branch and bound procedure are proposed in [170] [123] [122]. However, these methods are only practical for moderate and/or small problems.

Besides the branch and bound method, heuristics such as next-fit, first-fit, best-fit and bottom-left-fill (BLF) are widely used for solving bin packing problems [3] [9] [16] [68]. Heuristic approaches are particularly useful for problems with a high complexity, for which

deterministic methods like the branch and bound approach are often unable to find the solution within a reasonable amount of time. Although heuristics are fast in generating a solution packing plan, the quality of the solution is highly dependent on the input sequence of items.

Meta-heuristics such as evolutionary algorithms [39] [44] [69] [76] [77] [88] [112] [143] [148] [164] [183] [218], simulated annealing [15] and tabu search [116] [177] which are probabilistic in nature have also been applied to solve bin packing problems. Among these methods, evolutionary algorithm (EA) is usually hybridized with a heuristic placement routine. In this two-stage approach, EA searches for optimal input sequence of items and heuristic placement routine determines how the input sequence of items is packed into bins.

The current literature on the bin packing problem mostly focuses on the minimization of wasted space. In fact, there are other important objectives that need to be addressed for bin packing problems. For example, Amiouny [3] addressed the issue of balance, to make the center of gravity of packed items as close as possible to a target point. Amiouny concentrated only on the issue of balance by assuming that the bin was large enough to hold all the items. However in most bin packing problems, both minimization of wasted space and balance of the bins needs to be achieved. For example, in the case of tractor trailer trucks loading, it is important to utilize the maximum possible volume of each trailer and minimize the per axle weight as trucks are subject to constraints that regulates the maximum weight per axle that can be carried by tractor trailer trucks. For the two-objective bin packing problems as mentioned above, there is no mathematical formulation available in literature. Therefore a multiobjective two-dimensional bin packing model (MOBPP-2D), with minimum wasted space and balancing of load as two objectives, is formulated in this work.

As the solution to MO problems like MOBPP-2D exists in the form of alternate trade-offs known as Pareto optimal set, it is important to develop an effective and efficient algorithm capable of finding the set of optimal packing solutions. Being a population based statistical search method, EA is capable of capturing multiple optimal solutions in a sin-

gle run and is very suitable for solving MO problems [195] [212]. Particle swarm optimization (PSO) is based on swarming theory where all particles in the neighborhood depend on their discoveries and past experiences of their neighbors. Social sharing of information offers an evolutionary advantage and aids towards solving complex optimization problem. Although PSO is relatively new, it has been shown to offer higher convergence speed for MO optimization as compared to canonical multiobjective evolutionary algorithms (MOEAs) [26] [47] [130] [145]. Despite the fact that EA has been widely applied to solve bin packing problems [39] [44] [69] [77], PSO has never been considered in the context of solving bin packing problems. To explore the potential of combining evolutionary computation and PSO concepts for MOBPP-2D problems, a new multiobjective evolutionary particle swarm optimization algorithm (MOEPSO) is proposed in this paper. MOEPSO can generate a variety of packing plans to satisfy different customer's different spacing and balancing requirements. The packing solutions generated will provide important information for resource planning and decision making.

This chapter is organized as follows: Section 6.1 describes the scenario and modeling of multiobjective bin packing problems. Section 6.2 describes the proposed MOEPSO algorithm and its various features including variable-length representation and specialized mutation operators. Section 6.3 gives an algorithm behavior analysis of MOEPSO on multiobjective bin packing problems. A performance comparison with other algorithms is also provided to show the effectiveness and efficiency of MOEPSO in solving bin packing problems. Conclusions are drawn in section 6.4.

6.1 Problem Formulation

As there is no mathematical model available in literature for MO bin packing problems, a new two-objective two-dimensional bin packing model (MOBPP-2D) is formulated to provide the basis for further discussion on test case generation and algorithm performance

analysis in this section. The proposed MOBPP-2D problem can be described as follows: Given an unlimited number of bins with fixed width W, height H and centre of gravity (Φ_x, Φ_y), and J number of items with $w_j \preceq W$, $h_j \preceq H$ and weight ψ_j, pack the items into the bins such that space wastage is minimized and overall centre of gravity (CG) of the bins is as close as possible to the desired CG. The second objective, minimization of CG deviation of all bins, favors packing solutions with more balanced bins. Importance of balanced load of each bin (minimum CG_{dev}) is further demonstrated in section 6.1.1 and mathematical model of MOBPP-2D is presented in section 6.1.2.

6.1.1 Importance of Balanced Load

To facilitate better understanding of the second objective (minimum CG_{dev}), a graphical representation of item and bin used in MOBPP-2D is shown in Figure 6.1. As shown in the figure, minimum CG_{dev} is achieved by minimizing d_i , the deviation of overall CG (x_i, y_i) of the loaded bin from desired CG (x_d, y_d).

In the formulation of MOBPP-2D, CG constraint is considered in order to evaluate how balanced or stabilized the bins are in a packing plan. Balanced load plays an important role in application such as loading of containers, tractor trailer trucks and cargo airplanes. In container loading problem, the toppling of container will lead to time wastage and high cost in storage and transportation. However, a balanced load can help prevent the container from toppling even if some unexpected force is applied to the side of the container. In trailers and trucks loading, a balanced load can minimize the maximum weight per axle as required by some road regulations and provide better fuel efficiency and better response of the truck as a whole to the loading. Besides, trucks have to maintain their centers of gravity low enough to withstand any tendency to topple over when driven around sharp corners. In aircraft loading, the arrangement of cargo affects the position of center of gravity of the aircraft, which in turn has an impact on aircraft drag. One therefore wants to balance the load so that the aircraft will fly more safely, fly faster and use less fuel. Since the objective

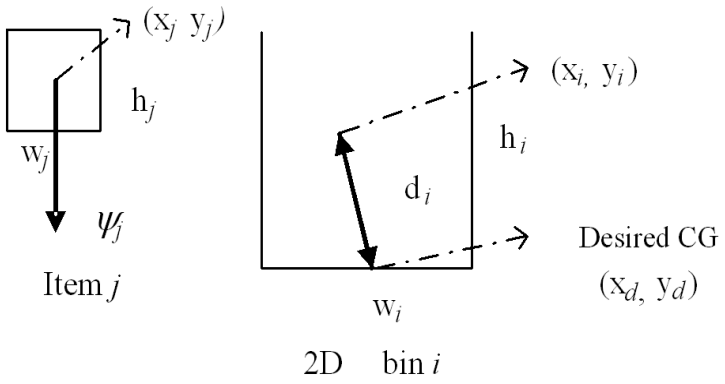

Figure 6.1: Graphical representation of item and bin

of balanced load plays such an important role in all these real world applications, it is appropriate to incorporate it as the second objective in the mathematical formulation of bin packing problems.

6.1.2 Mathematical Model

The definition of parameters, objectives and constrains for MOBPP-2D is given as follows.

Decision Variables

$X_{ij} \in \{0, 1\}$, where $i = \{1, ..., I\}$, $j = \{1, ..., J\}$. If item j is assigned to bin i, $X_{ij} = 1$, otherwise $X_{ij} = 0$;

Parameters

J = Number of items;

I = Maximum number of bins;

Symbol

$\lceil x \rceil$: The smallest integer larger or equal to x;

$\lfloor x \rfloor$: The largest integer smaller or equal to x;

Attributes of Bins

W_i: Width of bin i;

H_i: Height of bin i;

$\Phi_{x,i}$: Center of gravity of bin i in the x direction;

$\Phi_{y,i}$: Center of gravity of bin i in the y direction;

$\Phi_{d,i}$: Desired center of gravity of bin i in the x direction;

Attributes of Items

w_j: Width of item j;

h_j: Height of item j;

x_j: Center of gravity of item w.r.t. the bin i in x direction;

y_j: Center of gravity of item w.r.t. the bin i in y direction;

ψ_j: Weight of item j;

Minimization Objectives

The bin packing solutions should minimize both objectives concurrently as follows:

Number of Bins Used

$$K = \sum_{i=1}^{I} \left\lceil \sum_{j=1}^{J} \frac{X_{ij} h_j w_j}{H_i W_i} \right\rceil \tag{6.1}$$

Average Deviation of CG from Idealized CG of Bins

$$CG_{dev} = \frac{1}{K} \sum_{i=1}^{K} \sqrt{(\Phi_{x,i} - \Phi_{d,i})^2 + (\Phi_{y,i})^2} \tag{6.2}$$

CG of Bin after Loading

$$\Phi_{x,i} = \frac{\sum_{j=1}^{J} X_{ij} x_j \psi_j}{\sum_{j=1}^{J} \psi_j} \tag{6.3}$$

$$\Phi_{y,i} = \frac{\sum_{j=1}^{J} X_{ij} y_j \psi_j}{\sum_{j=1}^{J} \psi_j} \tag{6.4}$$

Packing Constraints

For every particular item $j \in \{j = 1, ..., J\}$:

$0 \prec w_j \preceq W_i$;

$0 \prec h_j \preceq H_i$;

$\psi_j \succ 0$;

For all bins $i \in \{i = 1, ..., I\}$:

$\sum_{j=1}^{J} X_{ij} w_j h_j \preceq W_i H_i$;

$0 \preceq \Phi_{x,i} \preceq W_i$;

$0 \preceq \Phi_{y,i} \preceq H_i$;

Each item must be assigned to and only to one bin:

$\sum_{i=1}^{I} \sum_{j=1}^{J} X_{ij} = J$.

6.2 Evolutionary Particle Swarm Optimization

The proposed MOEPSO is a PSO-based algorithm which incorporates EA concepts such as the use of mutation operator as a source of diversity. Furthermore, MOEPSO is characterized by the fact that particle movement is directed by either personal best or global best only in each instance. This is contrary to existing works where particle movement is influenced by both personal and global best at the same time.

In this section, detailed implementation of MOEPSO is given to show exactly how MOBPP-2D problems are tackled. The flowchart of the MOEPSO which shows various features needed to direct the movement of the particles towards feasible solutions is presented in section 6.2.1. Section 6.2.2 describes the design of particles variable length representation

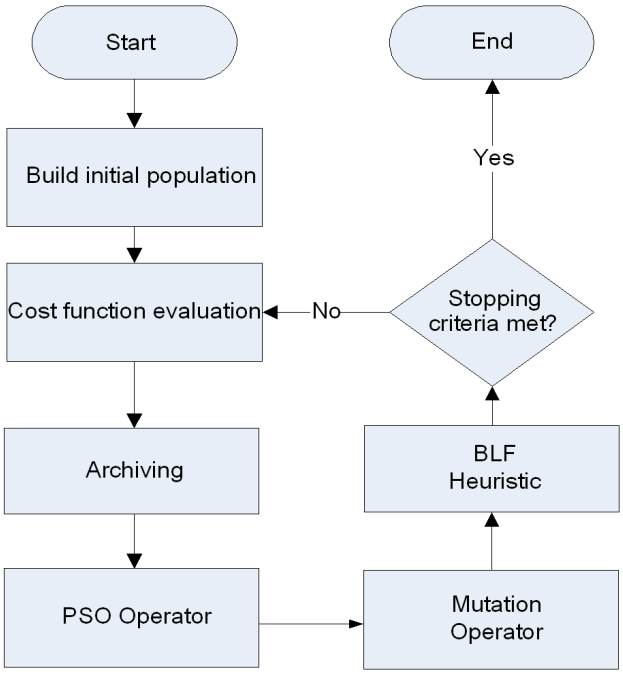

Figure 6.2: Flowchart of MOEPSO for solving the bin packing problem

and the implementation of bottom left fill (BLF) heuristic. As mentioned before, meta-heuristics like evolutionary computation techniques need to be hybridized with a heuristic placement routine to pack the sequence of items into bins. BLF heuristic is chosen in our algorithm and the advantages of using BLF heuristic are discussed. In section 6.2.3, the initialization process is explained in detail. Section 6.2.4 demonstrates how particle updating mechanism (PSO operator) works. The specialized mutation operators of MOEPSO are described in section 6.2.5. Finally, a brief introduction of archiving process is given in section 6.2.6.

6.2.1 General Overview of MOEPSO

For the success of MOEPSO, an appropriate particle representation that can be easily handled by PSO operator, mutation operator and BLF heuristic is necessary. Here we focus on the variable length representation scheme because it is considered the most intuitive

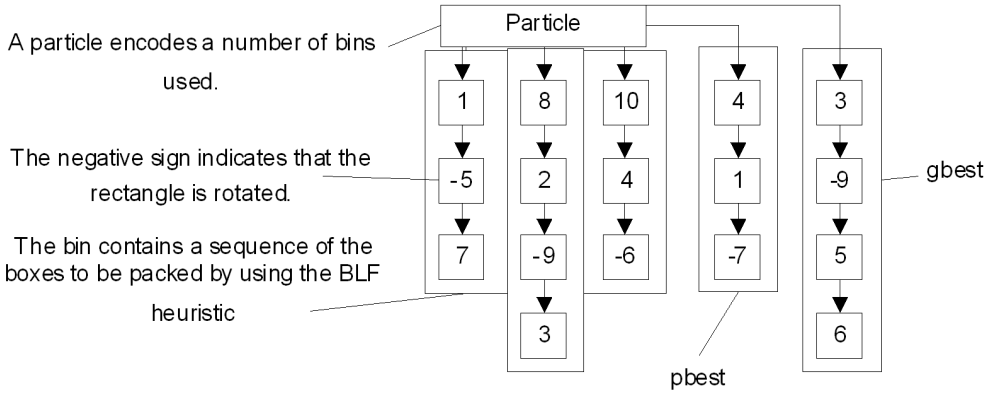

Figure 6.3: The data structure of particle representation (10 item case)

representation for a packing plan [135], and also for the considerable number of optimization operations compatible with this encoding scheme. As shown in Figure 6.2, the first step in the flowchart of MOEPSO is the building of initial solutions. In the initialization process, both random and sorted sequences are packed using BLF into the form of variable length solution particles. Once initialization has finished, the particles are evaluated against the fitness function and ranked according to the Pareto ranking scheme. After selecting either their personal best bin or global best bin as the velocity vector, particles are updated by inserting the bin represented by the velocity vector as the first bin and deleting duplicate items in other bins (PSO operation). At their new positions, the particles then undergo specialized mutation operations. If any constraint is violated during mutation, the bins violating constrains and the two least filled bins will be selected for repacking using BLF. After that, all the non-dominated solutions are inserted into the archive for storage and for the selection of global best bin in future generations.

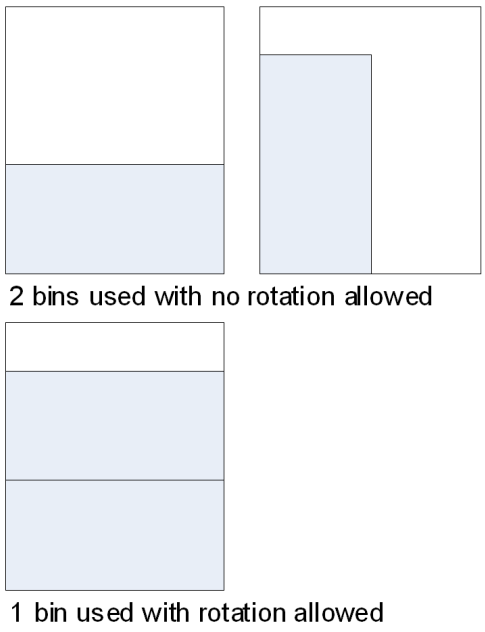

Figure 6.4: Saving of bins with the inclusion of the orientation feature into the variable length representation

6.2.2 Solution Coding and BLF

Variable-Length Representation of the Particle

To devise an effective encoding strategy on which PSO operation and mutation operation can be easily performed is important for the success of MOEPSO. In MOEPSO, order-based variable length particle structure is perhaps the most natural representation of a solution for bin packing problems. Each particle encodes the packing solution which includes number of bins used and order of which the items are packed into the bin by the BLF heuristic. In each bin, there must be at least one item and that item must not be found in any other bins. The permutations encoded are unique for every bin. After all, this is how the problem is presented and solved. Such a variable length representation is versatile as it allows the algorithm to manipulate the permutation of items in each bin without affecting other bins. And BLF check can be applied only to the changed bins.

As shown in Figure 6.3, each particle is constructed as several strings of distinct integers,

each string representing a bin and each integer representing an item. To pack 10 items, the particle length is 10 with each element in the range $[1, 10]$ representing a particular rectangular item. For the particle shown in Figure 6.3, the 10 items are packed into 3 bins, and the order of items in each bin implies the sequence of placement using BLF heuristic.

The variable length representation also facilitates the incorporation of pbest and gbest. As shown in Figure 6.3, pbest and gbest are encoded in the form of extra bins which are stored in the particle memory. pbest is defined as the best bin found by the particle; gbest is defined as the best bin found by the population.

In MOEPSO, best bin instead of best solution is chosen as pbest and gbest as the aim is to identify the smallest natural piece of a solution which is meaningful enough to convey information about solution quality [44]. In the case of MOBPP-2D: the bin points itself out as the natural "information quantum". The better each of the bins is packed, the fewer bins needed and the lower CG deviation achieved. Therefore best bin instead of best solution is defined as pbest and gbest to help the algorithm avoid falling into random search by transmitting good bin among generations.

A new state variable is attached to every index in the permutation to allow the variable length representation to be able to encode orientation of the rectangle item. In Figure 6.3, the presence of negative sign indicates that the rectangle item is rotated by 90 degrees and the absence indicates that the rectangle item is not rotated. The advantage of this new representation can be seen clearly in Figure 6.4. Without rotation, the two items in Figure 6.4 have to be packed into 2 bins. With rotation, one bin is enough.

The Implementation of the Bottom Left Fill Heuristic

To generate variable length solution particles from sequence of items, a good heuristic needs to be chosen. Generally heuristic approaches for solving two-dimensional bin packing problems are either level-oriented or bottom-left oriented. Berkey et al. [9] successfully applied

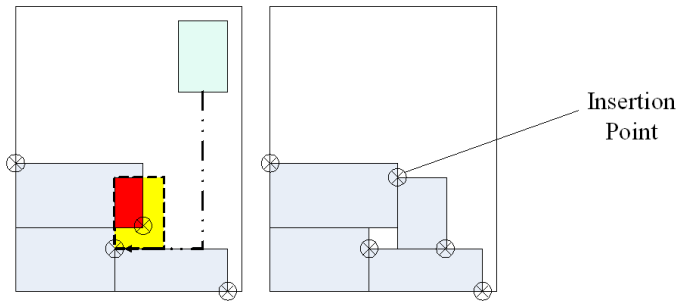

Figure 6.5: The insertion at new position when an intersection is detected at the top

heuristic such as level-oriented next-fit, level-oriented first-fit and next bottom-left on solving two-dimensional bin packing problems. In designing meta-heuristics for bin packing, bottom-left (BL) heuristic was widely used. Liu et al. [113] and Jakob [82] presented meta-heuristics that hybrid bottom-left (BL) procedure with genetic algorithm for packing rectangles and polygons respectively. Both reported that better results were achieved through hybridization of BL procedure with genetic algorithm. However, Hopper et al. [69] proposed a bottom-left-fill algorithm (BLF) that was capable of filling existing gaps in partial packing layout and showed great performance in solving bin packing problems. Since next-fit, first-fit and BL-routine tend to generate layouts with relatively large empty areas, BLF is chosen as the decoding heuristic to be used in MOEPSO.

BLF heuristic adheres strictly to BL condition, which means no rectangle items packed can be moved further to the bottom or to the left [82]. In BLF heuristic, a list of insertion points as shown in Figure 6.5-Figure 6.7 is maintained to allow the algorithm to search for available space to insert item in the bottom-left manner. Two new insertion points (at right lower and left upper corner of the rectangle item) are inserted into the list whenever a rectangle item is placed at an insertion point and the insertion point at which the rectangle item is placed is deleted from the list.

When placing a new rectangle item at an insertion point, an intersection test with those items already in the bin is carried out. As shown in Figure 6.5 and Figure 6.6, a new

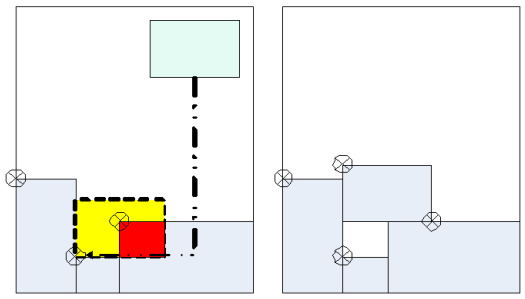

Figure 6.6: The insertion at new position when an intersection is detected at the right

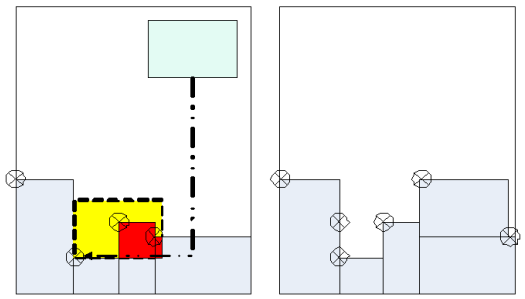

Figure 6.7: The insertion at next lower position with generation of three new insertion points

insertion point is generated at the right if there is an intersection detected at the top of the item and at the top if there is an intersection detected at the right. The empty space is still available for the insertion of items if the items can fit in. From Figure 6.7, if there is a lower available space for insertion, the rectangle is inserted there instead at the newly generated point where the intersection is detected. A new bin is generated if there is no space in previous bins to accommodate a new rectangle item. The BLF heuristic in pseudo code is presented in Figure 6.8. Figure 6.9 shows pseudo code to check whether the rectangle items in a particular bin can be accommodated in the same bin when the order of insertion of items is changed.

```
For each rectangle in list
  For each available bin
    For each insertion point in the insertion list
      check for intersection
      If there is an intersection
        insert a new insertion point
      else
        insert rectangle at current insertion point
        remove the current insertion point
        insert two new insertion point at the top left and bottom right of current rectangle
        sort the insertion list in bottom left order
      End If
    End For
  End For
  If there is no bins which can accommodate the rectangle
    create a new bin
    insert the rectangle in the newly created bin
  End If
End For
```

Figure 6.8: Pseudo code for BLF heuristic

```
Procedure Check whether all rectangles is inserted into bin
  For each rectangle in list
    For each insertion point in the insertion list
      check for intersection
      If there is an intersection
        insert a new insertion point
      else
        insert rectangle at current insertion point
        remove the current insertion point
        insert two new insertion point at the top left and bottom right of current rectangle
        sort the insertion list in bottom left order
      End If
    End For
    If there is no space to accommodate rectangle
      return false
    End if
  End For
  return true
End Procedure
```

Figure 6.9: Pseudo code to check whether all rectangles can be inserted into the bin

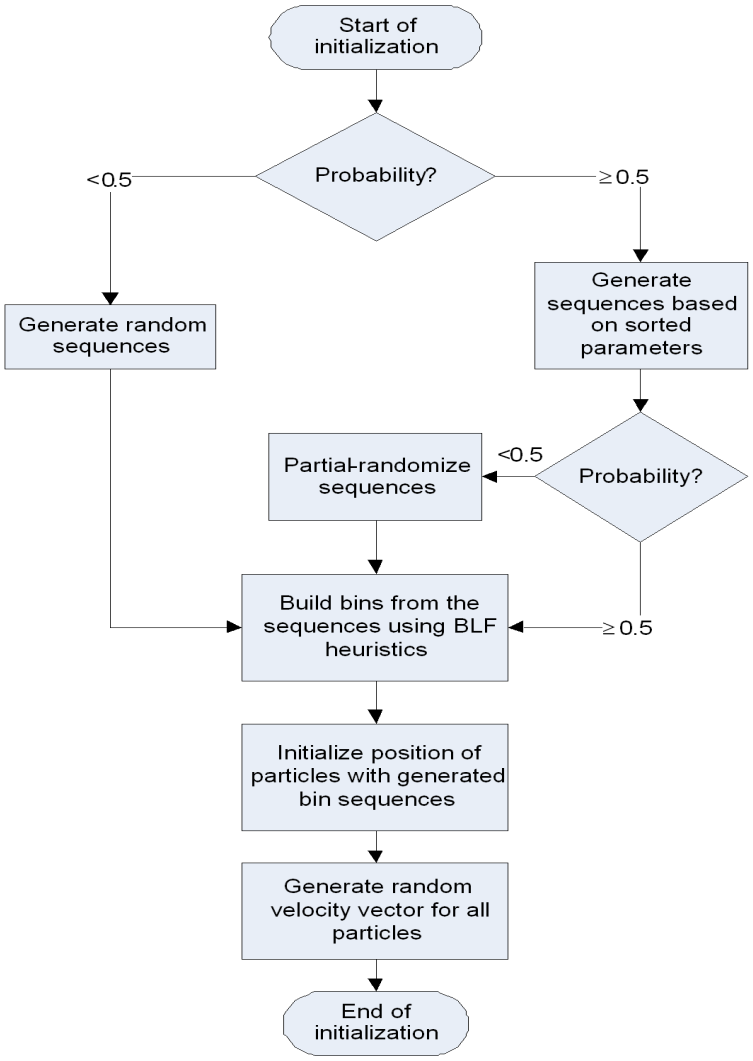

Figure 6.10: Initialization of initial solutions for the swarm of particles

6.2.3 Initialization

As shown in the flowchart of MOEPSO, the first step of the algorithm is initialization. In evolutionary computation or particle swarm optimization, initial population is usually generated randomly. However, instead of using randomly generated sequences for initial solution particle generation, seeding some sorted sequences may help to capture certain characteristic of the bin packing problem and reduce the computation time for achieving

good results. For example, if items are ordered according to density and packed using BLF, swap operation between the first bin and the last bin will help achieve lower CG deviation. If items are ordered according to size, BLF heuristic can help achieve more closely packed bins as the small items in the last part of sequence have better chance to be inserted into the empty area among large items. Therefore in MOEPSO, some of the sequences are randomly generated to provide more variety whereas some are initially sorted according to some criteria, e.g. width, height, area, perimeter and density. To avoid having repeated sequences, some sorted sequences are further partial-randomized. With the generated sequences, variable length solution particles are built by using BLF heuristic as shown in Figure 6.8. The whole procedure of initialization is described in Figure 6.10.

6.2.4 PSO Operator

Particle swarm optimization works by adjusting trajectories of a particle. However in a discrete space, the meanings of concepts such as trajectory and velocity are not so straight forward. In solving discrete space optimization problems, Kennedy defined trajectories, velocities, etc., in terms of changes of probabilities that a bit will be in one state or the other [91]. In Kennedy's method, population members were not viewed as potential solutions, instead they were probabilities. The value of v_d for each dimension determined the probability that a bit x_{id}^k would take on one or zero value, but x_{id}^k itself did not have a value until it was evaluated: the actual position of the particle in D dimensional space was ephemeral. However this definition can not be used in our MOEPSO algorithm as variable length representation instead of bit representation is chosen in MOEPSO.

In particle swarm optimization, each individual in the population will try to emulate the gbest and pbest solutions through updating by PSO equations [186]. Therefore the question is how to emulate the pbest and gbest in our MOBPP-2D problem. Since packing quality information is kept in the bins of each solution particle, our policy is to set best bin as pbest and gbest, and let other bins being repacked. In this way, best bin information is kept. The

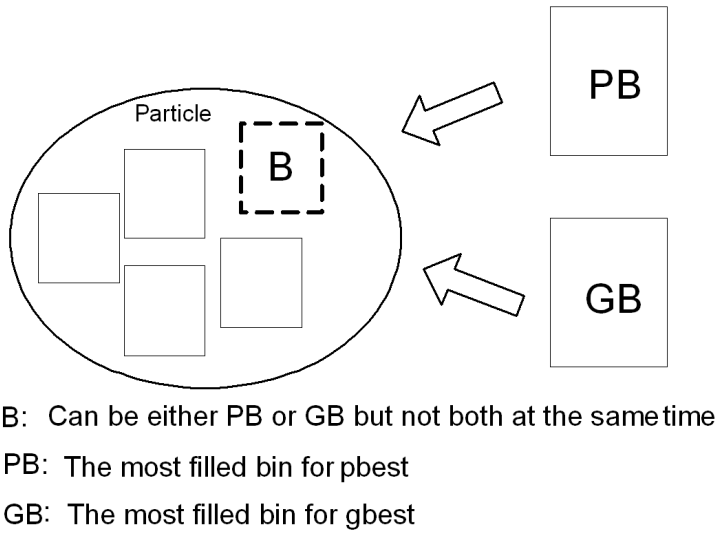

Figure 6.11: Mechanism for updating position of particle

proposed PSO operator can be shown as the following equations.

$$V_{id} = \alpha \times P_{id} \oplus (1 - \alpha) \times P_{gd}, \alpha = 0, 1 \tag{6.5}$$

$$X_{id} = X_{id} + V_{id} \tag{6.6}$$

In equation 6.5, the velocity is governed by either pbest or gbest. To choose gbest bin or pbest bin is determined by the value of α. The chosen best bin is then used in the determination of the movement of solution particle as shown in Figure 6.11. The chosen best bin will be inserted into the solution particle as the first bin and any duplicate items in other bins will be deleted.

The pbest of the particle is updated by evaluating the current solution with its previous best solution to see whether the current solution dominates the previous best solution. The pbest is replaced with the current best bin and the personal best solution is replaced by the current solution if the current solution dominates the previous best solution. If both solutions are non-dominating to each other, there is a 50% probability of the pbest being replaced by the current best bin and the previous best solution replaced by the current best

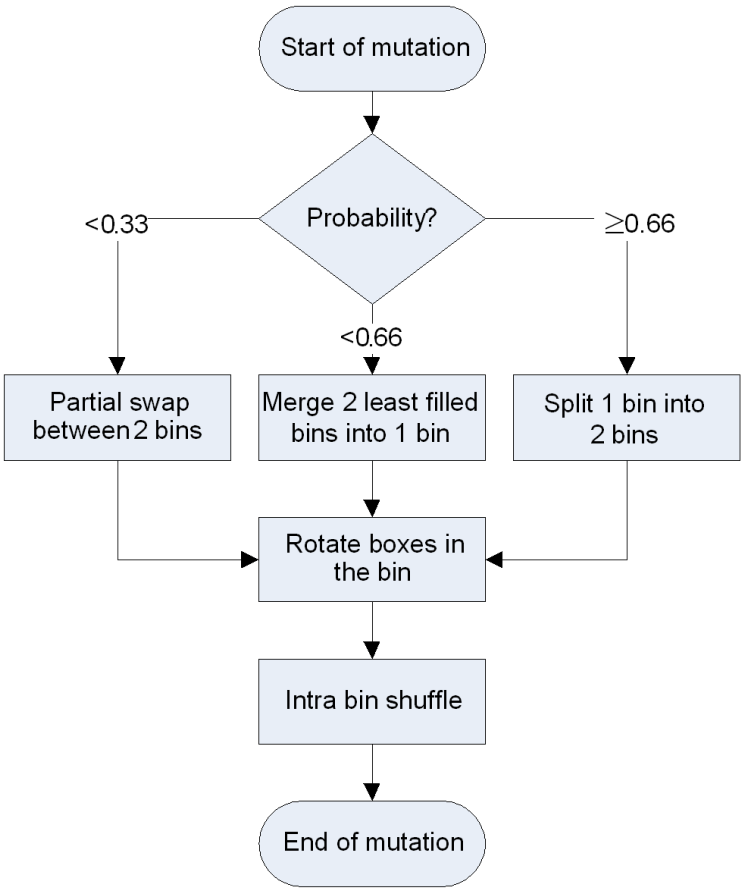

Figure 6.12: Mutation modes for a single particle

solution. There is no updating of pbest if the previous best solution dominates the current solution. The selection of gbest is by the tournament niche method where a particle with a lower niche count is selected out of two randomly selected non-dominated particles from the archive.

6.2.5 Specialized Mutation Operators

PSO operator can help MOEPSO keep best bin information, while specialized mutation operators constructed in this section consists of several mutation operations which attempt to change the internal structure of the particle to further optimize the solution.

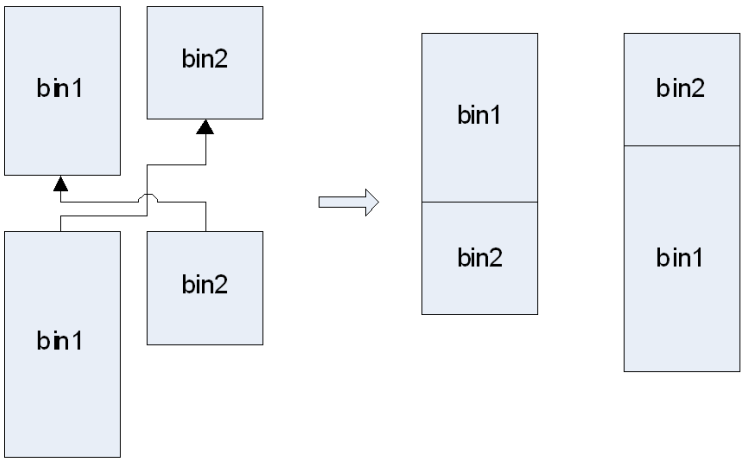

Figure 6.13: Partial swap of sequence between two bins in a particle

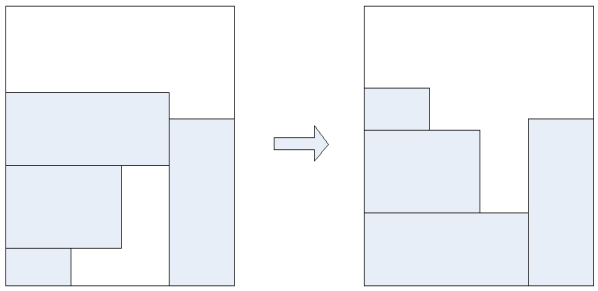

Figure 6.14: Intra bin shuffle within a bin of a particle

Figure 6.12 shows the whole mutation process of the particle. A random number is first generated to determine the mode of mutation to undergo. The modes include partial swap mechanism where sequences of items in two bins are randomly cut and exchanged, merge bins mechanism where the items in the two least filled bins are merged into one bin, and split bin mechanism where items in a randomly chosen bin are split into two bins. Partial swap operation as shown in Figure 6.13 is designed to search for better packing for bins, for example more closely packed items or smaller CG deviation. Merge bins operation help reduce the number of bins used and split bin operation can reduce the CG deviation. Thereafter, the particle undergoes another two modes of mutation where one of them randomly

rotates the rectangles in the bin, and the other shuffle the items of the bin to try to search for an arrangement which can improve the packing configuration and the stability of the bins. Figure 6.4 has shown how rotate can improve packing solution. The effectiveness of shuffle operation can be seen in Figure 6.14. A feasibility check is then applied to make sure all the constraints are satisfied. If any constraint is violated, the bins violated the constraints and the two least filled bins (emptiest bins) are selected and eliminated from the particle. The items which composed those bins are thus missing from the solution and we use the BLF to insert them back in a random order. This step help make sure that all particles are valid solutions of MOBPP-2D. As shown in Figure 6.12, the specialized mutation mechanism provides the facilities for MOEPSO to manipulate the internal bin structures of the particle.

6.2.6 Archiving

In our algorithm, elitism is implemented in the form of a fixed-size archive to prevent the lost of good particles due to the stochastic nature of the optimization process. The archive is updated at each generation, e.g., if the candidate solution is not dominated by any members in the archive, it will be added to the archive. Likewise, any archive members dominated by this solution will be removed from the archive. One solution is said to dominate another if it meets one of the three criteria below:

- A smaller number of bins used but an equal average deviation from desired CG.
- A smaller average deviation from desired CG but an equal number of bins used.
- A smaller average deviation from desired CG and a smaller number of bins used.

In order to maintain a set of uniformly distributed non-dominated solution particles in the archive, the dynamic niche sharing scheme [195] is employed. The niche radius defines how far apart two individuals should be in order for them to lower each other's fitness. When

Table 6.1: Parameter settings used by MOEPSO in the simulation

Parameters	MOEPSO
Population Size	500
Generation Size	1000 generations or no improvement in the last 5 generations
Niche Radius	0.1

the predetermined archive size is reached, a recurrent truncation process [96] based on niche count is used to eliminate the most crowded archive member.

6.3 Computational Results

MOEPSO is programmed in C++ based on a Pentium IV 2.8-GHz processor with 512MB RAM under Microsoft Windows XP. The performance of MOEPSO is examined on various test instances of the MOBPP-2D problem with varying item sizes and weights. The generation of test cases is described in detail in section 6.3.1 and the parameter settings used by MOEPSO in performance comparison is shown in Table 6.1. MOEPSO works well in niche radius range from 0.05 to 0.2. Too small or too large niche radius causes the algorithm to locate less diverse and poorly distributed solutions. For the population size, the most appropriate range for MOEPSO is from 100 to 500. Too small population results in less diversified Pareto front and too large population leads to longer computation time.

6.3.1 Test Cases Generation

MOBPP-2D models various factors affecting the packing performance, particularly on how the size and weight of items affect the stability of packed bins. In order to examine these factors thoroughly, 6 classes with 20 instances each are randomly generated following the

Table 6.2: Test cases generation

Class	1	2	3	4	5	6
w_j, h_j	$[0, 100]$	$[0, 25]$	$[0, 50]$	$[0, 75]$	$[25, 75]$	$[25, 50]$
ψ_j	[0,20] for instance 1-10 of each class [0,100] for instance 11-20 of each class					

pattern in [9] with the addition of different weight parameter. The different combinations of variables in 6 classes of test cases are shown in Table 6.2.

As shown in Table 6.2, w_j and h_j are randomly generated in different ranges for different classes. For all the instances generated, there are 500 items to be packed. The six classes of test cases generated can well examine MOEPSO performance on packing items with different sizes. An example of test case generated is given in Figure 6.15.

In the sample of the input file of Class_3_1_11.txt as shown in Figure 6.15, the first row refers to the class number of the problem. The second row refers to the number of items to be packed. The third row refers to the indexes of the filename. The fourth row refers to the width and height of the bin. Subsequent rows refer to the width, height and weight of the items.

6.3.2 Overall Algorithm Behavior

Pareto Front

To analyze the performance of MOEPSO on MOBPP-2D problems, we begin by analyzing the Pareto front generated which presents to the decision maker a set of non-dominated solutions to derive an appropriate packing plan. Figure 6.16 shows three stages of evolution progress of the Pareto front, namely initial (First), intermediate (Int) and final (Final) for 4 selected instances from Class 3. During the initial stage, there are a small number of

```
3
500
1          11
100        100
30         39         26
19         9          36
17         35         76
38         10         87
29         17         32
4          6          20
25         48         60
29         13         53
20         2          36
14         29         76
10         8          78
34         6          24
39         23         40
1          40         19
38         29         30
13         35         43
35         37         69
27         38         55
34         10         10
29         15         14
37         38         34
...
9          31         12
19         41         30
47         40         86
21         49         69
15         2          60
35         6          67
9          40         75
3          26         29
43         16         74
42         42         7
38         43         84
17         1          13
25         1          30
32         24         51
46         2          96
4          49         100
39         14         18
34         50         50
37         22         3
8          34         80
13         12         17
8          4          81
16         50         43
8          8          96
9          32         81
```

Figure 6.15: A sample input file of Class_3_1_11.txt

non-dominated solutions scattered in the upper left region of the objective space. This is due to the fact that the beginning solutions are generated using BLF heuristic which tends to produce more closely packed bins. As the evolution progresses, the number of non-dominated solutions increases and the Pareto front moves towards the final stage of the Pareto front. The final stage Pareto front also shows that the average deviation from the idealized CG of the bin decreases as the number of bins used increases, which proves that the two objectives are conflicting with each other and it is a real multiobjective problem. But it is also observed that the number of bins used is always smaller than the number of items that is available for packing. This is so because the packing has to obey the BL-condition

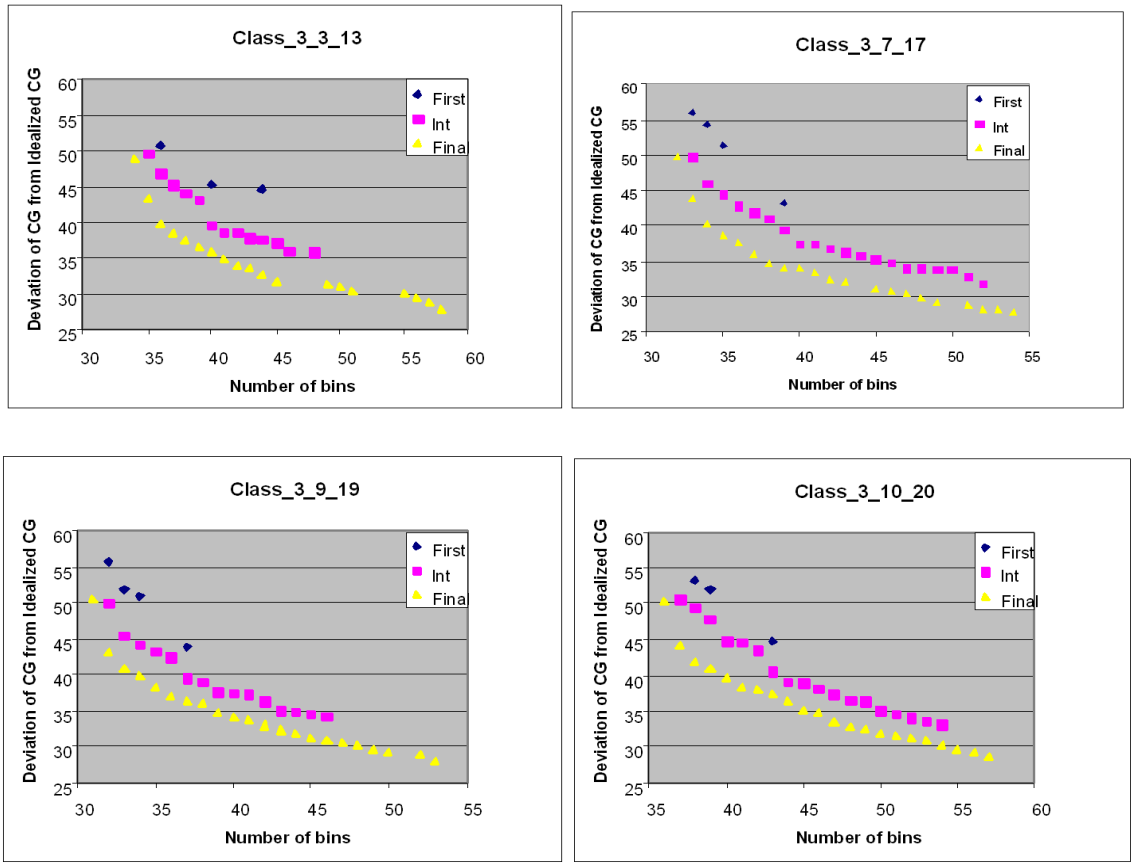

Figure 6.16: Evolution progress of the Pareto front

in MOEPSO. Only one item in each bin can not achieve a good CG deviation as the item will be shifted to the left and the CG will not be in the center. As shown in Figure 6.17, the bin on the right is favored over the one on the left although both have the same CG deviation. This is because the bin on the right has more items in it, and thus will probably give a solution which uses a smaller number of bins. The Pareto front also shows that there is a limit to the least number of bins used. This is due to the fact that there is a so-called continuous lower bound for number of bins used [123]. The continuous lower bound can be obtained by allowing each item to be cut into any number of smaller pieces.

At the end of the optimization process, improvements can be observed for both the number of bins as well as the CG deviation. There is also a good distribution of solutions

Figure 6.17: Two bins with the same deviation from idealized CG

along the discovered Pareto front, demonstrating the effectiveness of MOEPSO for MOBPP-2D problems.

Computational Result for Different Classes of Items

While the last section analyzes the general characteristics of solution Pareto front to show the effectiveness of MOEPSO, this section examines how different combinations of different sized items can affect the stability of the bins packed. As shown in section 6.3.1, class 2 consists of only small items. Class 3 consists of a combination of small and medium sized items whereas class 4 consists of a combination of small, medium and large items. Figure 6.18 shows the box plot of average deviation of CG for the non-dominated solutions in 9 randomly selected test cases from class 2, 3 and 4. Each box plot represents the distribution of a sample set where a vertical line within the box encodes the median, while the left and right ends of the box encodes the upper and lower quartiles. Dashed appendages illustrate the spread and shape of distribution, and dots represent the outside values.

In Figure 6.18, the mean deviation of CG of class 2 is consistently lower than class 3 and class 4 with class 4 test cases scoring the highest mean deviation of CG. This result shows that when the items are small, there is more room to arrange the items to achieve lower stability in the bins. When there is a mixture of small and medium size items, there appears

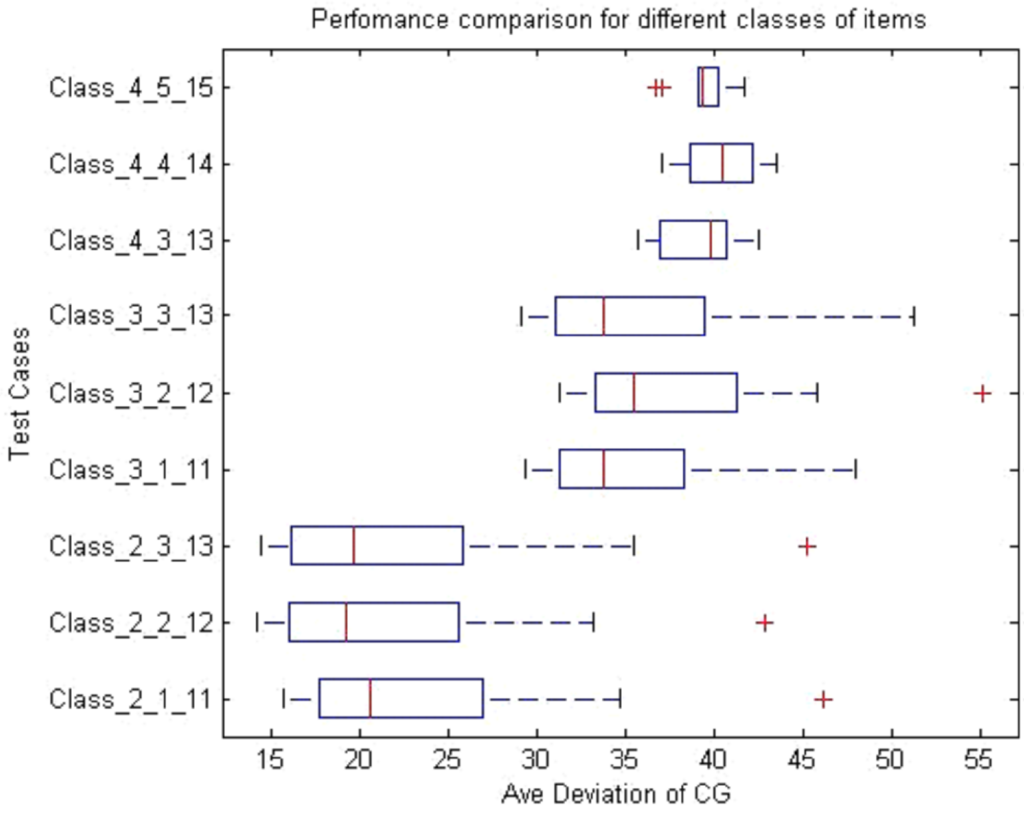

Figure 6.18: Average deviation for different classes of items

to be less permutation. The worst case is when there is a mixture of small, medium and large items to pack. However, with less permutation to arrange in each bin, the problem becomes easier where convergence is achieved faster. From the result, it could be seen that the average deviation of CG of most solutions generated are less than half the height of the bin. This further shows that MOEPSO is capable of generating solutions which uses a smaller number of stabilized bins.

Effectiveness of the PSO Operator

To understand what contributes to the performance of MOEPSO, the effects of the PSO operator is examined in this section while the impact of the mutation operator is examined in the next section.

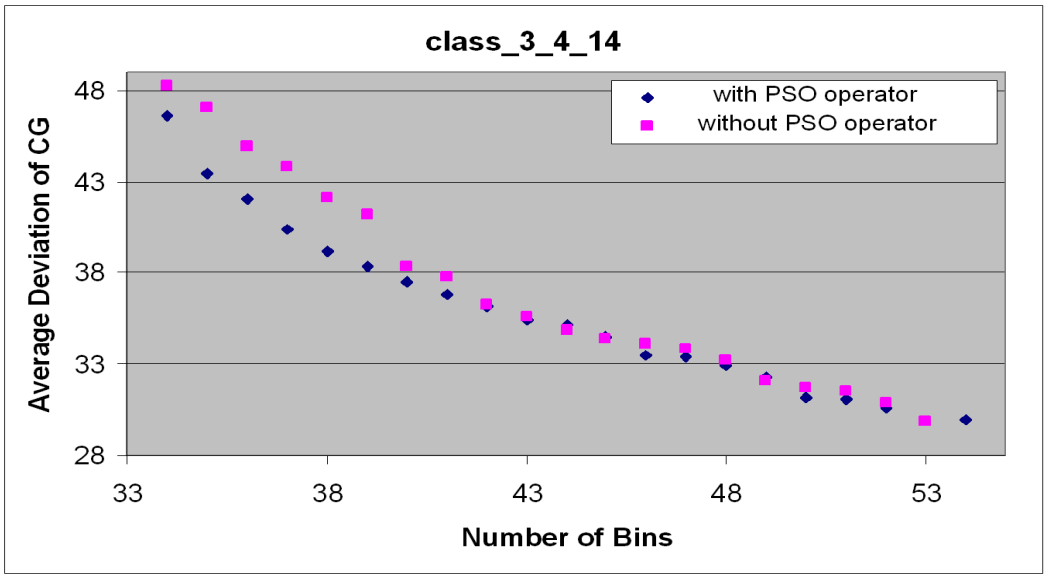

Figure 6.19: Pareto front to show the effectiveness of the PSO operator (Class_3_4_14)

Figure 6.19 and Figure 6.20 shows the effectiveness of the PSO operator in populating the non-dominated solutions along the Pareto front. From the two graphs, the PSO operator has brought about an improvement in the spread of the Pareto front and the quality of the non-dominated solutions. Without the PSO operator, there is no enough interaction between the particles. The lack of social sharing of information leads to degradation in solution quality. However, for class 3 problems, the effects of PSO operator in improving diversity is not observed. But in most cases, the PSO operator enables the algorithm to achieve a lower CG deviation. It is also observed that keeping the best bin will reduce the CG deviation considerably, especially when the number of used bins is small. For class 4 problem, the presence of the PSO operator allows the MOEPSO to discover a greater range and number of nondominated solutions. One possible reason is that the algorithm will be trapped in local optimum without the guidance of the PSO operator. As shown in Figure 6.19 and Figure 6.20, the PSO operator is an important operator for the success of MOEPSO.

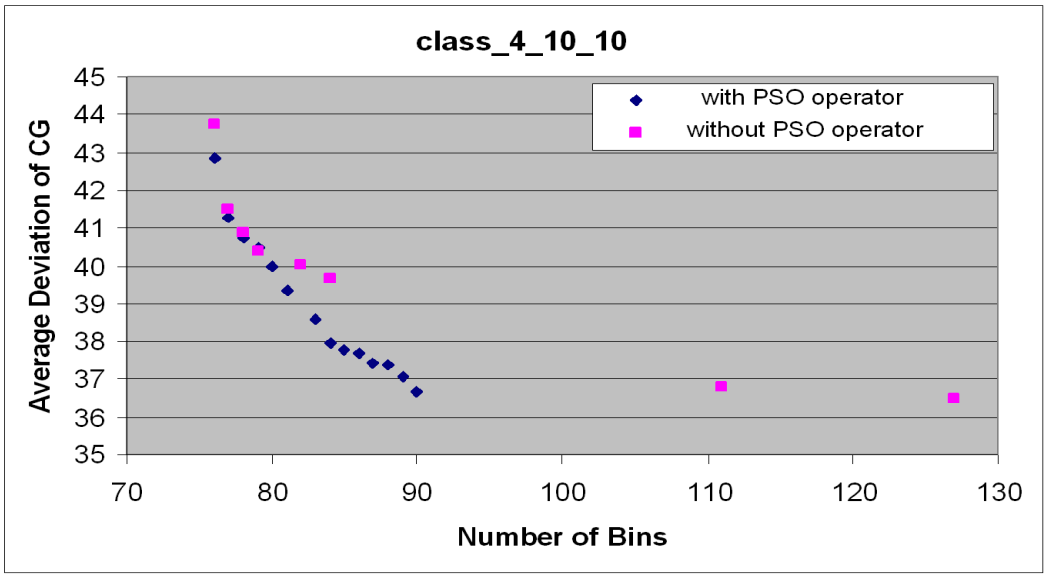

Figure 6.20: Pareto front to show the effectiveness of the PSO operator (Class_4_10_10)

Effectiveness of the Mutation Operators

Specialized mutation operators are one important feature of MOEPSO. Figure 6.21 compares the performance of the MOEPSO with and without the mutation operator for Class_3_6_16. From the graph, it is obvious that the mutation operator bring about a major improvement in the Pareto front in terms of spread and quality of the non-dominated solutions. This is because the specialized mutation operator increases MOEPSOs exploration ability in searching solution space. The above comparison is only one run of the algorithm. In order to provide a statistical performance comparison, 10 runs of the algorithms with and without mutation operator is analyzed quantitatively using three performance metrics widely used by evolutionary multiobjective optimization community (GD, MS and S). However, the computation of the GD and MS metric requires the knowledge of global Pareto front which is unknown for all the test cases analyzed here. To provide a fair comparison, a global Pareto front is obtained by selecting the non-dominated solutions from all the results of 10 runs of the algorithm with and without mutation.

Figure 6.22 shows the performance metric for test problem Class_3_6_16. From the

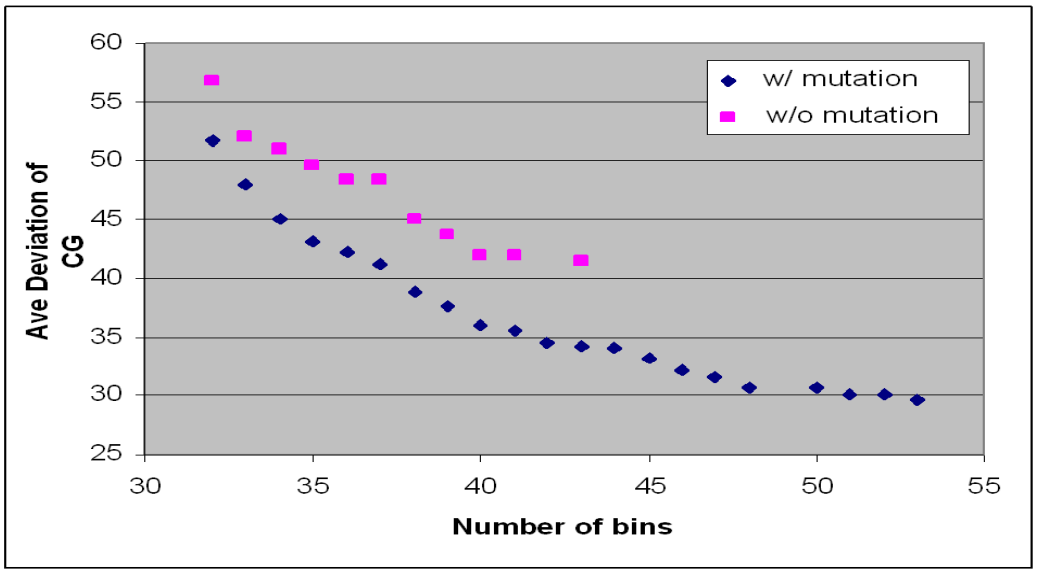

Figure 6.21: Pareto front to show the effectiveness of the mutation operator (Class_3_6_16)

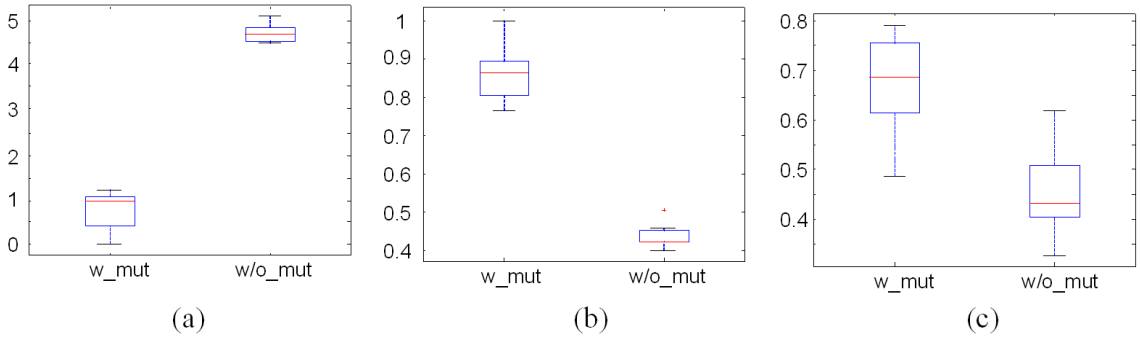

Figure 6.22: Performance for Class_3_6_16: a) GD, b) MS and c) S

performance metric of GD and MS, it is evident that the mutation operator has brought about a significant improvement in the quality of the solutions obtained. In particular, the incorporation of the mutation operator has allowed the algorithm to achieve better convergence and diversity.

6.3.3 Comparative Analysis

In this section, the performance of the proposed MOEPSO is validated against other algorithms. As MOBPP-2D is a newly formulated problem and no other algorithms have been designed for it, it is not so easy to find appropriate algorithm to compare performance. Although many EAs have been applied to solve bin packing problems, there is no MOEA available for solving MOBPP-2D. In addition, PSO has never been applied to solve BPP. In this section, a PSO algorithm revised from [214] and a MOEA algorithm revised from [189] are chosen to compare with MOEPSO as these two algorithms are easily modified to solve MOBPP-2D and they can achieve very good result on traditional single objective bin packing problem as shown in the following section.

In [214], the traveling salesman problem has been tackled by using PSO with much success. New operators for the problem has been developed based on the idea of PSO where the velocity vector of a particle is designed to be a swap sequence, and the position of a particle is designed to be a sequence of indexes. In a sense, the velocity of the particle seems to direct the particle towards an idealized permutation in combinatorial problems. This is done by applying the swap sequence to the position of the particles by swapping the position of the indexes in the sequence indicated in the swap sequence. The generation of the swap sequence is done by comparing the best position of the particle with the current position of the particle, where their difference in the position of the indexes is taken to be a series of swap factors represented in the swap sequence. This concept is used for the development of an MOPSO algorithm for comparison with MOEPSO. All above mentioned operators have been retained with the addition of BLF to pack the sequence found into bins.

The hybrid multi-objective evolutionary algorithm (HMOEA) designed for solving the truck and trailer vehicle routing problems [189] is found to be more suitable for use with bin packing problem since there is some similarities between multiobjective vehicle routing problem and multiobjective bin packing problem. The most notable features of HMOEA are

the variable length chromosome representation, specialized genetic operators, Pareto ranking and the fitness sharing features. By using a variable length chromosome representation, a chromosome can encode a complete routing plan including the number of routes and tasks served by the trucks, e.g., a route is a sequence of tasks to be served by a truck. Specialized genetic operators are also designed to manipulate the variable length representation. These genetic operators include route exchange crossover and multimode mutation. Route exchange crossover allows good sequences of route to be shared among chromosomes whereas multimode mutation complements the crossover by optimizing the route information of the chromosome by changing the task sequences within a chromosome. Pareto ranking scheme differentiates the different chromosomes from each other according to their relative fitness where the smallest rank is assigned for all non-dominated individuals, while the dominated individuals are ranked inversely according to the number of non-dominated individuals dominating over them. To avoid overcrowding in the solution space, fitness sharing is used to diversify the solutions along the Pareto front. As the number of individuals increases in a niche, the amount of competition also increases, thereby reducing the fitness of the individuals competing in that niche. During the tournament selection process, a chromosome with a lower rank in partial order is selected for reproduction. This briefly summaries the whole mechanism of HMOEA used in solving the TTVRP problem. The similarity between TTVRP and BPP make it quite easy to realize a MOEA algorithm using the ideas of HMOEA to solve the bin packing problem.

Single Objective Case

To prove that MOEPSO is not only fit for newly proposed MOBPP-2D problem, a comparison on the performance of meta-heuristic algorithms (EPSO, EA [189], PSO [214]) against the branch and bound method [123] on single objective problem (minimum bins) is done by running the algorithms against the 6 classes of test cases from [9]. EA is modified from [189] and PSO is modified from [214] and EPSO is the single objective version of MOEPSO.

Table 6.3 and Table 6.4 show the average number of optimal solutions obtained by each algorithm in 10 runs. Those results highlighted in bold shows the situation where one of the algorithms has won against the branch and bound method in terms of the number of optimal solutions generated. Indeed, EPSO has obtained the best performance by achieving 238 optimal solutions. EA ranks second by 234 optimal solutions. PSO and branch and bound method has similar performance by achieving 206 and 211 optimal solutions respectively. The higher number of optimal solutions generated shows that EPSO is more stable in performance across different runs of different test cases compared with the other three algorithms.

Bi-objective Case

In this section, the performance of MOEPSO is compared against MOPSO and MOEA on MOBPP-2D problem to illustrate their statistical behavior on test cases. As mentioned before, MOEA is modified from [189] and MOPSO is modified from [214]. The performance comparison of the three algorithms is done by providing a quantitative assessment on the algorithms using the three performance metrics GD, MS and S. 10 runs are performed for each algorithm on the two randomly selected test cases from Class 2 and 3 to study the robustness and consistency of the algorithms. The global Pareto front (GF, needed for computation of GD and MS) is obtained by selecting the non-dominated solutions from all the results of 10 runs of the 3 algorithms. The parameters settings used in this part of the analysis are shown in Table 6.5.

- *Analysis for Class_2_4_14*

Class_2_4_14 consists of all small items which are generated in the range of [0 25]. From Figure 6.23, the Pareto front of MOEPSO is better as it is much spread out and nearer to the global Pareto front (GF). Only a few nondominiated solutions are found by MOPSO. The swap sequence operation proposed in [214] is not so effective in spreading items to

Table 6.3: Number of optimal solutions obtained by branch and bound method, EA, PSO and EPSO (Class 1-3)

Class	deterministic method [123]	EA	PSO	HPSO
Class1				
n=20	10	10	10	10
n=40	10	10	9	10
n=60	7	6	5	6
n=80	3	10	5	10
n=100	1	6	3	7
Total	31	**42**	**32**	**43**
Class 2				
n=20	10	10	10	10
n=40	10	9	9	9
n=60	4	10	9	10
n=80	10	9	8	9
n=100	10	10	9	10
Total	44	**48**	**45**	**48**
Class 3				
n=20	9	10	10	10
n=40	9	6	5	6
n=60	5	5	1	6
n=80	0	4	1	4
n=100	0	5	1	5
Total	23	**30**	18	**31**

different bins to lower CG deviation. This is because MOPSO has no bin level operation to separate items in bins with high CG deviation to more bins.

Figure 6.24 summarizes the different algorithms statistical performance on 10 runs of the Class_2_4_14 test case. From Figure 6.24, MOEPSO has the lowest GD which shows that more nondominated solutions of GF are from MOEPSO. The MS metric of MOPSO is much worse than MOEPSO and MOEA which again proves that MOPSO can only find a few nondominated solutions each run. In the spacing metric, MOEPSO is the clear winner

Table 6.4: Number of optimal solutions obtained by branch and bound method, EA, PSO and EPSO (Class 4-6)

Class	deterministic method [123]	EA	PSO	HPSO
Class 4				
n=20	10	10	10	10
n=40	10	10	10	10
n=60	7	8	8	8
n=80	10	7	7	7
n=100	10	8	7	8
Total	47	43	42	43
Class 5				
n=20	10	9	10	9
n=40	10	8	9	10
n=60	8	5	5	6
n=80	0	3	3	3
n=100	1	3	0	2
Total	29	28	27	**30**
Class 6				
n=20	10	10	10	10
n=40	5	6	6	9
n=60	10	9	9	6
n=80	10	10	10	10
n=100	2	8	7	8
Total	37	**43**	**42**	**43**

since it scored the lowest average value. From the analysis, it can be seen that the average performance of MOEPSO is the best among the three algorithms.

Figure 6.25 shows the evolutionary traces for the test case Class_2_4_14. Of all the 3 algorithms, MOPSO shows the worst performance as it is stuck at the local minimum and unable to search for more non-dominated solutions in the vicinity. This is why there is no improvement in later generations where a straight line of performance metric is observed in the graphs. From the evolutionary traces of the metric, information is revealed on how

Table 6.5: Parameter settings used by MOEPSO in the simulation

Parameters	MOPSO	MOEA	MOEPSO
Crossover Rate	-	0.7	-
Mutation Rate	-	0.4	-
Population Size	500	500	500
Generation Size	200	200	200
Niche Radius	0.1	0.1	0.1

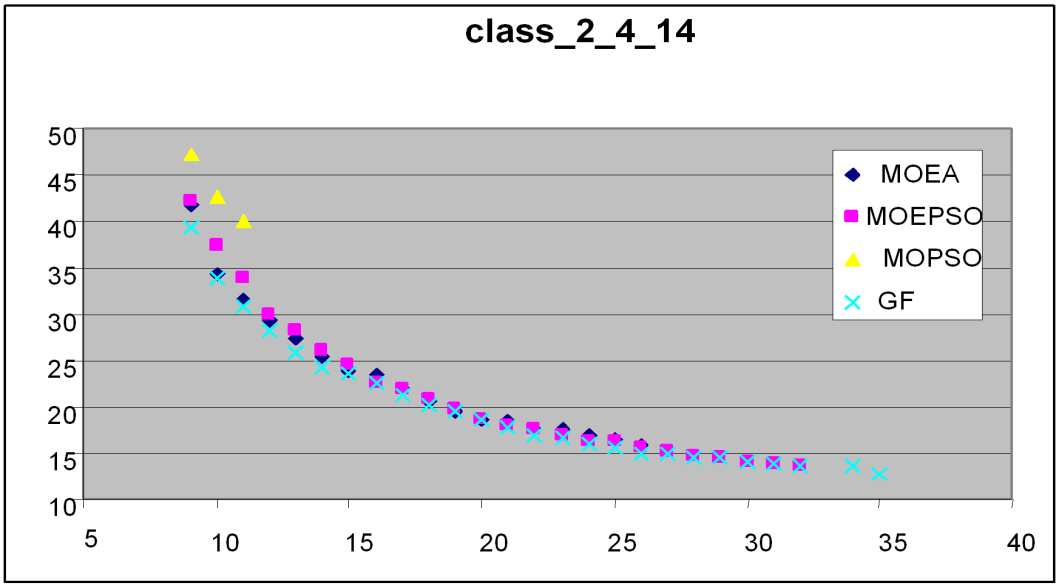

Figure 6.23: Pareto front of Class_2_4_14

MOEPSO searches for the best set of non-dominated solutions. MOEPSO has the best performance since it has the best GD in 200 generations. The MS of MOEPSO indicates that MOEPSO effectively searches the region where the global Pareto front resides. MOEPSO dominates the spacing metric as it has the lowest value throughout the whole evolution of 200 generations.

- *Analysis for Class_3_4_14*

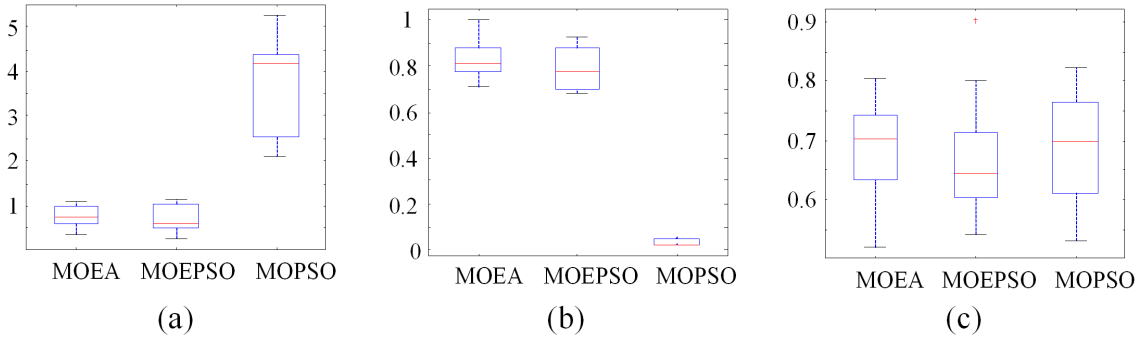

Figure 6.24: Performance for Class_2_4_14: a) GD, b) MS and c) S

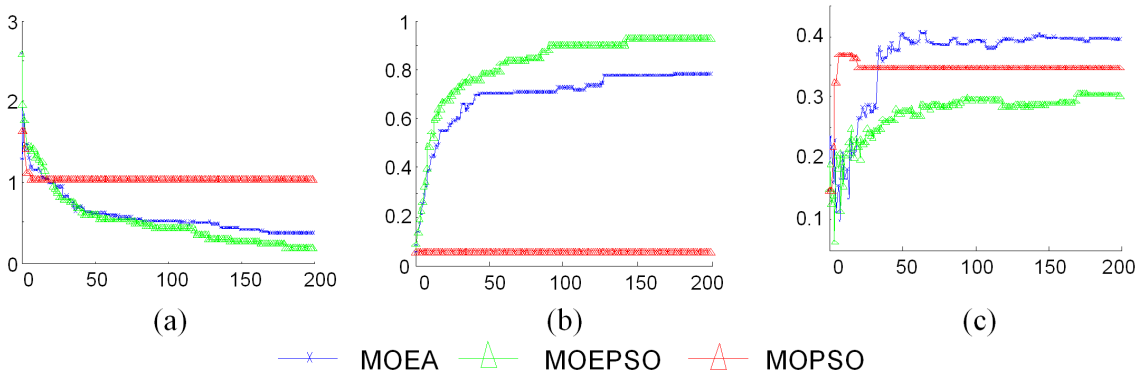

Figure 6.25: Evolutionary trajectories for Class_2_4_14: a) GD, b) MS and c) S

The test case Class_3_4_14 belongs to Class 3 and consists of mainly of small and medium size items. From Figure 6.26, the Pareto front of MOEPSO is comparable to the one of MOEA. Again, the MOPSO is stuck at the local minimum. From Figure 6.27, MOEPSO and MOEA have comparable results in the performance metric GD and MS. The average performance of MOEPSO is considered better than MOEA as MOEPSO is better in the Spacing metric.

Figure 6.28 shows the evolutionary traces for the test case Class_3_4_14. In the evolutionary traces for Class_3_4_14, MOEPSO has the best GD and MS at the end of 200 generations. In the evolutionary trace of the Spacing metric, it is observed that there are spikes in the early generations. This may be due to the discovery of non-dominated solutions

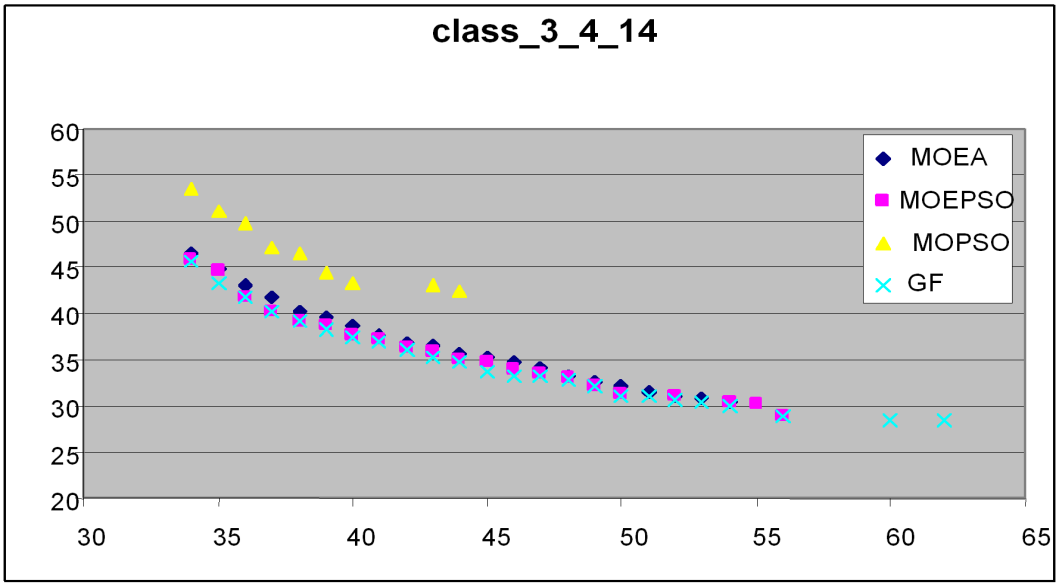

Figure 6.26: Pareto front of Class_3_4_14

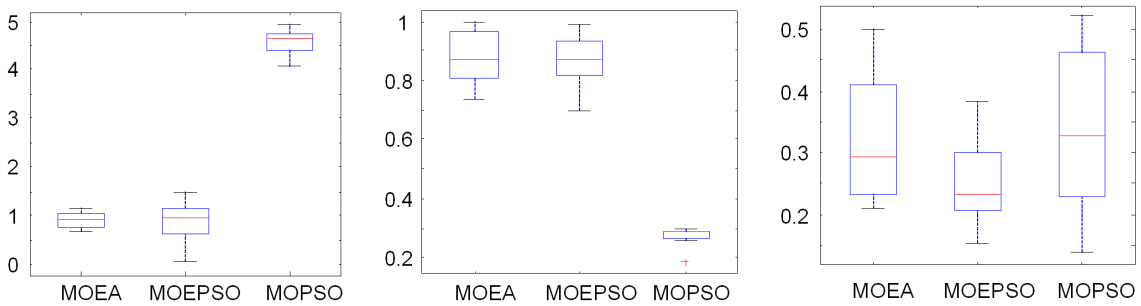

Figure 6.27: Performance for Class_3_4_14: a) GD, b) MS and c) S

which are far away from the non-dominated solutions found in the previous generations. As the generation progresses, there are fewer spikes as the non-dominated solutions are scattered more evenly on the Pareto front. As before, MOEPSO has the best trace as compared to the other two algorithms and MOPSO shows no improvement in its evolutionary trace after 100 generations.

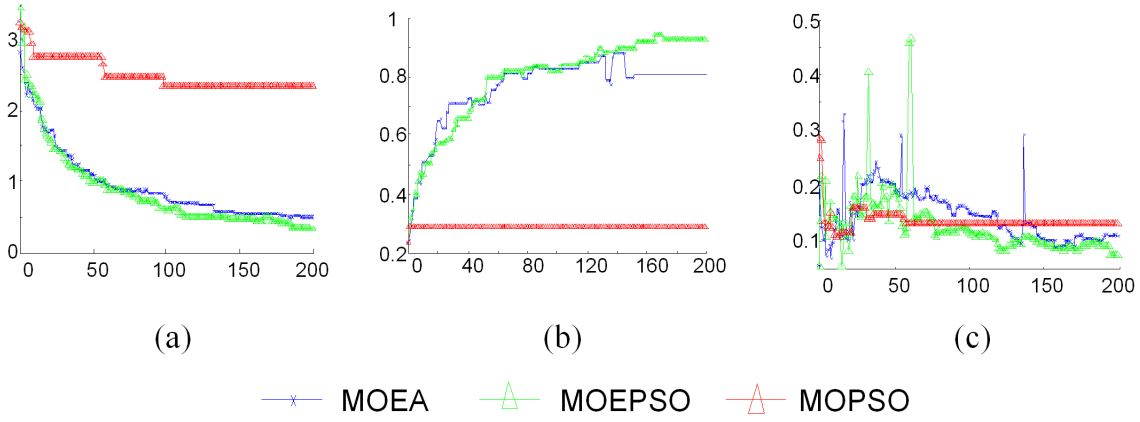

Figure 6.28: Evolutionary trajectories for Class_3_4_14: a) GD, b) MS and c) S

In general, for all test problems, MOEPSO performed consistently well on GD metric and does not converge on local Pareto front like MOPSO. MOEPSO also performed the best in terms of the spacing metric as nondominated solutions are well distributed along the Pareto front.

Algorithm Efficiency Comparison (Computational Time)

While it is important to achieve good solutions, computational cost is another issue that must be considered in the real world. In this section, the computational efficiency of the different algorithms is evaluated based on computational time. The three algorithms will stop at 1000 iterations. From Figure 6.29, the MOPSO incurs the shortest computation time among all the three randomly chosen test cases. However, the optimization result obtained by MOPSO is inferior as compared to the result obtained by MOEA and MOEPSO. One possible reason is that MOPSO is operating on sequence of items. After operation, they only pack once using BLF. However, in the cases of MOEA and MOEPSO, there is a possibility of repacking in the event of violations. Therefore MOEA and MOEPSO may take more time. From the chart, MOEPSO takes lesser time on average compared to MOEA. From the data set generated, Class 2 seems to be more difficult than the rest of the classes. This

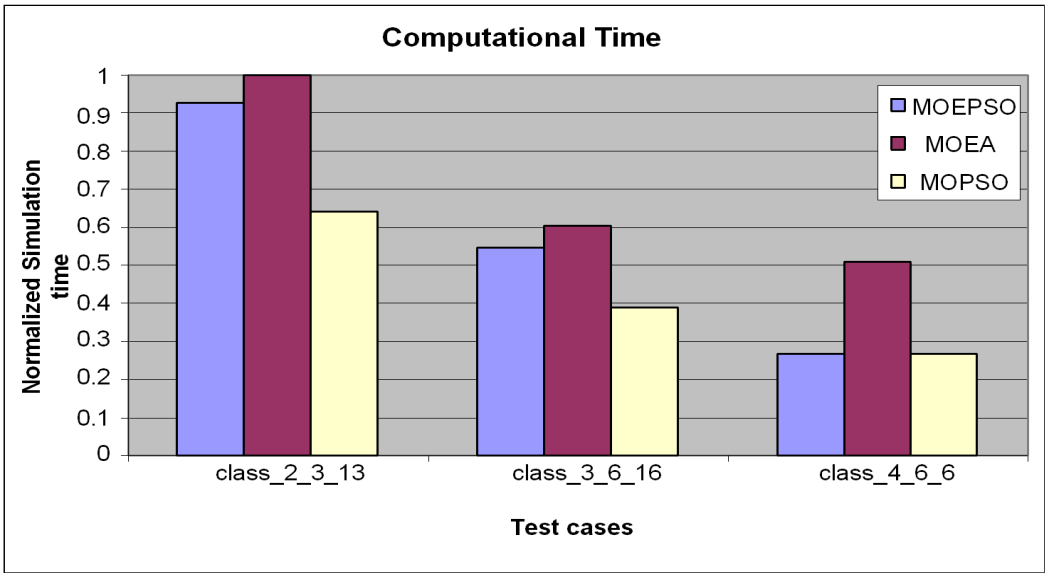

Figure 6.29: Normalized computation time for the three algorithms

is because there are all small items in Class 2. For small items, more permutation in each bin needs to be checked to achieve a more stable configuration, whereas for larger items, there are fewer items in each bin and therefore less permutation to be arranged to achieve a stable configuration.

6.4 Conclusion

In this chapter, a mathematical model for MOBPP-2D is presented and the MOEPSO is proposed to solve the problem. BLF is chosen as the decoding heuristic as it has the ability to fill in the gaps in the partial layout. The creation of variable length data structure and its specialized mutation operator make MOEPSO a flexible optimization algorithm which can manipulate permutation in bin level or particle level. Multi-objective performance tests have shown that MOEPSO performs consistently well for the test cases used in this research. Comparison study has proven that MOEPSO outperforms MOPSO [214] and MOEA [189] in most of the test cases attempted. The performance metric also shows

that MOEPSO has the best average performance. In a comparison on the performance of the meta-heuristics algorithms with the branch and bound method [123] against the single objective bin packing problem, EPSO (single objective version of MOEPSO) also stands out against all the algorithms. Overall, MOEPSO proved to be robust, producing good solutions consistently despite the difficulty of the problem. Both single objective and multiobjective bin packing problems can be easily handled using MOEPSO, which demonstrated that MOEPSO is a good candidate for solving real world bin packing problem.

Chapter 7

Conclusions and Future Works

This work has explored and improved particle swarm optimization techniques for MO function optimization as well as expanded its applications in real world bin packing problems. Three new PSO algorithms have been proposed and good performance improvement on standard MO function optimization test cases has been achieved. The successful application of key PSO concept in solving bin packing problems proves that PSO is a good candidate for solving discrete real world combinatorial optimization problems.

7.1 Conclusions

This work contributes towards the design and application of MOPSO algorithms. The impact of fuzzy gbest, local search, coevolution and distributed computation on the performance of MOPSOs has been investigated.

In Chapter 3, a new memetic algorithm of FMOPSO has been designed within the context of MO particle swarm optimization. In particular, two new features in the form of fuzzy global-best and synchronous particle local search have been proposed. The synchronous particle local search performs directed local fine-tuning, which helps to discover a well-distributed Pareto front. The fuzzy global-best models the uncertainty associated with

the optimality of global-best, thus helping the algorithm to avoid undesirable premature convergence. The proposed features have been examined to show their individual and combined effects in MO optimization. The comparative study showed that the proposed FMOPSO produced solution sets that are highly competitive in terms of convergence, diversity and distribution.

The algorithm CCPSO put forward in Chapter 4 attempts to further emulate the conflict and coexistence between cooperation and competition in nature by implementing both aspects into co-evolutionary model. This is accomplished by having species compete amongst themselves for the right to represent more components, and the winners cooperate to solve the whole problem. This proposed algorithm is validated through comparisons with existing state-of-the art multi-objective algorithms through the use of established benchmarks and metrics for comparison. CCPSO was the only algorithm to attain the true Pareto front in all test problems, and in all cases converged faster to the true Pareto front than any other algorithm used in this research.

As the availability of powerful networked computers presents a wealth of computing resources to solve problems with large computational effort, a distributed MOPSO algorithm is formulated in Chapter 5 utilizing PSO's intrinsic parallel property. The proposed distributed co-evolutionary particle swarm optimization algorithm achieves substantial speedup from non-distributed version. DCPSO has been designed to be fault-tolerant against network errors, such as lost package, corrupted package, and lost connection. A dynamic load balancing has also been incorporated into DCPSO to further enhance its performance.

In Chapter 6, a mathematical model for MOBPP-2D is presented and the MOEPSO is proposed to solve the problem. BLF is chosen as the decoding heuristic as it has the ability to fill in the gaps in the partial layout. The creation of variable length data structure and its specialized mutation operator make MOEPSO a flexible optimization algorithm which can manipulate permutation in bin level or particle level. Multi-objective performance tests have shown that MOEPSO performs consistently well for the test cases used in this research.

In a comparison on the performance of the meta-heuristics algorithms with the branch and bound method against the single objective bin packing problem, EPSO (single objective version of MOEPSO) also stands out against all the algorithms. Overall, MOEPSO proved to be robust, producing good solutions consistently despite the difficulty of the problem. Both single objective and multiobjective bin packing problems can be easily handled using MOEPSO, which demonstrated that MOEPSO is a good candidate for solving real world bin packing problem.

7.2 Future Works

Despite the fact that MOPSOs started to be developed less than ten years ago, the growth of this field has exceeded most people's expectations. Although different means of enhancing MOPSO has been studied in this work, these studies barely scratched the surface of what is left to be addressed. Although detailed analysis of issues related to the application of MOPSOs have been provided in this work, most discussions are based on empirical results on benchmark test problems. In addition, computational efficiency is considered only in terms of the number of function evaluations. Since the number of evaluation is not the best indicator of efficiency especially when evaluation cost is low, one immediate extension would be to consider the computational time complexity of the proposed mechanisms. Certainly, it would also be desirable to apply the proposed techniques to a larger number of real-world problems in the near future.

Like most existing work, this research has concentrated on MOPSO algorithm. On the other hand, it has been shown that hybrid algorithms can be very effective [18, 149]. The motivation behind hybrid methods is that a single method may not be enough for dealing with complex real world problem. Two or more methods can complement each other when carefully combined. Therefore another promising research area is to hybridize MOPSO with other computation techniques. For example, the hybridization of MOPSO and neural

network (NN) can help further utilize high convergence speed of MOPSO and modeling ability of NN on solving complex optimization problems.

Many real-world applications are characterized by a certain degree of noise, manifesting itself in the form of signal distortion or uncertain information. Despite the growing research on different MOPSO algorithms, the practical issue of noise handling has rarely been addressed by the MOPSO community. Design of noise handling techniques for MOPSO will be worth investigation. Surrogate models have been applied in the domain of SO optimization to find robust solutions, it has yet been studied for MO problems. Therefore one possible noise handling technique for MOPSO is to apply surrogate models in the optimization of noisy fitness functions where noise may be filtered out through the approximation process.

List of Publications

1. Tan, K. C., Liu, D. S., Goh, C. K. and Chiam, S. C., "A Competitive and Cooperative Co-evolutionary Approach to Multiobjective Particle Swarm Optimization Algorithm Design," *European Journal of Operational Research*, revised.
2. Liu, D. S., Tan, K. C., Huang, S. Y., Goh, C. K. and Ho, W. K., "On Solving Multiobjective Bin Packing Problems Using Evolutionary Particle Swarm Optimization," *European Journal of Operational Research*, vol. 190, no. 2, pp. 357-382, 2008.
3. Liu, D. S., Tan K. C. and Ho, W. K. "A Distributed Co-evolutionary Particle Swarm Optimization Algorithm," *IEEE Congress on Evolutionary Computation*, Singapore, September 25-28, pp. 3831-3838, 2007.
4. Liu, D. S., Tan, K. C., Goh, C. K. and Ho, W. K., "A Multiobjective Memetic Algorithm Based on Particle Swarm Optimization," *IEEE Transactions on Systems, Man and Cybernetics: Part B (Cybernetics)*, vol. 37, no. 1, pp. 42-50, 2007.
5. Liu, D. S. Tan, K. C., Goh, C. K. and Ho, W. K., "On Solving Multiobjective Bin Packing Problems Using Particle Swarm Optimization," *IEEE Congress on Evolutionary Computation*, Vancouver, BC, Canada, July 16-21, 2006.
6. Tan, K. C., Lee, T. H., Yang, Y. J. and Liu, D. S., "A Cooperative Coevolutionary Algorithm for Multiobjective Optimization," *IEEE International Conference on Systems, Man and Cybernetics*, The Hague, The Netherlands, October 10-13 2004, pp. 1926-1931, 2004.

Bibliography

[1] H. Abbass and R. Sarker and C. Newton, "PDE: A Pareto-frontier Differential Evolution Approach for Multi-objective Optimization Problems," in *Proceedings of the 2001 IEEE Congress on Evolutionary Computation*, pp. 27-30, 2001.

[2] Julio E. Alvarez-Benitez, Richard M. Everson, and Jonathan E. Fieldsend, "A MOPSO algorithm based exclusively on pareto dominance concepts," in *Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization*, pp. 459-473, 2005.

[3] S. V. Amiouny, J. J. Bartholdi III, J. H. Vande Vate, and J. X. Zhang, "Balanced Loading," *Operations Research*, vol. 40, no. 2, pp. 238-246, 1992.

[4] P. J. Angeline and J. B. Pollack, "Competitive environments evolve better solutions for complex tasks," in *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 264-270, 1993.

[5] S. Anily, J. Bramel, and D. Simchi-Levi, "Worst-case Analysis of Heuristics for the Bin Packing Problem with General Cost Structures," *Operations Research*, vol. 42, no. 2, pp. 287-298, 1994

[6] H. J. C. Barbosa and A. M. S. Barreto, "An Interactive Genetic Algorithm with Co-evolution of Weights for Multiobjective Problems," in *Proceedings of the 2001 Genetic and Evolutionary Computation Congress*, pp. 203-210, 2001.

[7] M. Basseur and E. Zitzler, "Handling Uncertainty in Indicator-Based Multiobjective Optimization, " *International Journal of Computational Intelligence Research*, vol. 2, no. 3, pp. 255-272, 2006.

[8] U. Baumgartner, Ch. Magele, and W. Renhart, "Pareto optimality and particle swarm optimization," *IEEE Transactions on Magnetics*, vol. 40, no. 2, pp. 1172-1175, 2004.

[9] J. O. Berkey and P. Y. Wang, "Two-Dimensional Finite Bin-Packing Algorithms," *Journal of the Operations Research Society*, vol. 38, pp. 423-429, 1987.

[10] P. Bosman and D. Thierens, "The balance between proximity and diversity in multi-objective evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 174-188, 2003.

[11] P. Bosman and D. Thierens, "The naive MIDEA: A baseline multi-objective EA," in *Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization*, vol. 3410, pp. 428-442, 2005.

[12] A. Bourkerche, "An Adaptive Partitioning Algorithm for Distributed Discrete Event Simulation Systems," *Journal of Parallel and Distributed Computing*, vol. 62, pp.1454-1475, 2002.

[13] A. Boukerche and S. L. Das,"Reducing Null messages overhead through load balancing in conservative distributed simulation systems," *Journal of Parallel Distributed Computing*, vol. 64, pp. 330-344, 2004.

[14] C. Boutevin, M. Gourgand, and S. Norre, "Bin Packing Extensions for Solving an Industrial Line Balancing Problem," in *Proceedings of the 5th IEEE Internatinal Symposium on Assembly and Task Planning*, pp. 115-121, 2003.

[15] M. J. Brusco, G. M. Thompson, and L. W. Jacobs, "A Morph-Based Simulated Annealing Heuristic for a Modified Bin-Packing Problem," *Journal of the Operations Research Society*, vol. 48, pp. 433-439, 1997.

[16] E. Burke, R. Hellier, G. Kendall, and G. Whitwell, "A New Bottom-Left-Fill Heuristic Algorithm for the Two-Dimensional Irregular Packing Problem," *Operations Research*, vol. 54, no. 3, pp. 587-601, 2006.

[17] E. K. Burke, P. Cowling, and P. De Causmaecker, "A memetic approach to the nurse rostering problem," *Applied Intelligence*, vol. 15, no. 3, pp. 199-214, 2001.

[18] E. K. Burke and A. J. Smith, "Hybrid evolutionary techniques for the maintenance scheduling problem," *IEEE Transactions on Power Systems*, vol. 15, no. 1, pp. 122-128, 2000.

[19] J. M. Chambers, W. S. Cleveland, B. Kleiner and P. A. Turkey. Graphical methods for data analysis. *Wadsworth and Brooks/Cole*, Pacific CA, 1983

[20] A. K. Chandra, D. S. Hirschberg, and C. K. Wong, "Bin Packing with Geometric Constraints in Computer Network Design," *Operation Research*, vol. 26, no. 5, pp. 760-772, 1978.

[21] A. J. Chipperfield and P. J. Fleming, "Multiobjective Gas Turbine Engine Controller Design Using Genetic Algorithms," *IEEE Transactions on Industrial Electronics*, vol. 43, no. 5, pp. 583-587. 1996.

[22] Chi-kin Chow and Hung-tat Tsui, "Autonomous agent response learning by a multi-species particle swarm optimization, " in *Proceedings of the 2004 Congress on Evolutionary Computation*, vol. 1, pp. 778-785, 2004.

[23] M. Clerc and J. Kennedy, "The Particle Swarm C Explosion, Stability, and Convergence in a Multidimensional Complex Space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, 2002

[24] C. A. Coello Coello, "Evolutionary Multiobjective Optimization: A Historical View of the Field," *IEEE Computational Intelligence Magazine*, vol. 1, no. 1, pp. 28-36, 2006.

[25] C. A. Coello Coello and N. Cruz Corts, "Solving Multiobjective Optimization Problems using an Artificial Immune System," *Genetic Programming and Evolvable Machines*, vol. 6, no. 2, pp. 163-190, 2005.

[26] C. A. Coello Coello, G. T. Pulido and M. S. Lechuga, "Handling Multiple Objectives With Particle Swarm Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256-279, 2004.

[27] C. A. Coello Coello and M. R. Sierra, "A Coevolutionary Multi-Objective Evolutionary Algorithm," in *Proceedings of the 2003 IEEE Congress on Evolutionary Computation*, vol. 1, pp. 482-489, 2003.

[28] C. A. Coello and M. S. Lechuga, "MOPSO: A proposal for multiple objective particle swarm optimization," in *Proceedings of the 2002 IEEE Congress on Evolutionary Computation*, pp. 1051-1056, 2002.

[29] D. W. Corne, J. D. Knowles, and M. J Oates, "The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization," in *Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature*, pp. 839-848, 2000.

[30] V. Cristea and G. Godza, "Genetic Algorithms and Intrinsic Parallel Characteristic," in *Proceedings of the Congress on Evolutionary Computation*, pp.431-436, 2000.

[31] X. Cui, M. Li and T. Fang, "Study of Population Diversity of Multiobjective Evolutionary Algorithm Based on Immune and Entropy Principles," in *Proceedings of the 2001 IEEE Congress on Evolutionary Computation*, vol. 2, pp. 1316-1321, 2001.

[32] A. G. Cunha, P. Oliviera, and J. Covas, "Use genetic algorithms in multicriteria optimization to solve industrial problems," in *Proceedings of the Seventhth International Conference on Genetic Algorithms*, pp. 682-688, 1997.

[33] K. Deb, "Multi-objective genetic algorithms: problem difficulties and construction of test problem," *Evolutionary Computation*, vol. 7, no. 3, pp. 205-230, 1999.

[34] K. Deb, P. Zope and A. Jain, "Distributed computing of Pareto-optimal solutions with evolutionary algorithms," in *Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization*, pp. 534-549, 2003.

[35] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182-197, 2002.

[36] K. A. De Jong, An analysis of the behaviour of a class genetic adaptive systems, Ph.D thesis, University of Michigan, 1975.

[37] V. Devireddy and P. Reed, "Efficient and Reliable Evolutionary Multiobjective Optimization Using ϵ-Dominance Archiving and Adaptive Population Sizing" in *Proceedings of the 2004 Genetic and Evolutionary Computation Conference*, pp. 130-131, 2004.

[38] P. Di Barba, M. Farina and A. Savini, "An improved technique for enhancing diversity in Pareto evolutionary optimization of electromagnetic devices," *The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, vol. 20, no. 2, pp. 482-496, 2001.

[39] K. A. Dowsland, "Genetic Algorithms C a Tool for OR," *Journal of the Operations Research Society*, vol. 47, pp. 550-561, 1996.

[40] D. Dubois and H. Prade, Possibility Theory: An Approach to Computerized Processing and Uncertainty, Plenum Press, New York, 1988.

[41] R. C. Eberhart and J. Kennedy, "A New Optimizer Using Particle Swarm Optimization," in *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, Nagoya, Japan, pp. 39-43, 1995.

[42] M. Emmerich, N. Beume, and B. Naujoks, "An EMO Algorithm Using the Hypervolume Measure as Selection Criterion," in *Proceedings of the Third Conference on Evolutionary Multi-Criterion Optimization*, pp. 62-76, 2005.

[43] R. M. Everson and J. E. Fieldsend, "Multiobjective Optimization of Safety Related Systems: An Application to Short-Term Conflict Alert," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 2, pp. 187-198, 2006.

[44] E. Falkenauer and A. Delchambre,"A Genetic Algorithm for Bin Packing and Line Balancing," *IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1186-1192, 1992.

[45] M. Farina and P. Amato, "A fuzzy definition of "optimality" for many-criteria optimization problems," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 34, no. 3, pp. 315-326, 2003.

[46] M. Farina, "A Minimal Cost Hybrid Strategy for Pareto Optimal Front Approximation," *Evolutionary Optimization*, vol. 3, no. 1, pp. 41-52, 2001.

[47] J. E. Fieldsend and S. Singh, "A multi-objective algorithm based upon particle swarm optimization, an efficient data structure and turbulence," in *Proceedings of the 2002 U.K. Workshop on Computational Intelligence*, Birmingham, U.K., pp:37-44, Sept. 2002

[48] J. E. Fieldsend, R. M. Everson and S. Singh, "Using Unconstrained Elite Archives for Multiobjective Optimization" *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 3, pp. 305-323, 2003.

[49] J. E. Fieldsend, and S. Singh, "Pareto evolutionary neural networks," *IEEE Transactions on Neural Networks*, vol. 16, no. 2, pp. 338-354, 2005.

[50] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of Distributed Consensus with One Faulty Process," *Journal of The Association of Computing Machinery*, vol.32, no. 2, pp. 374-382, 1985.

[51] M. Fleischer, "The Measure of Pareto Optima. Applications to Multi-objective Metaheuristics," in *Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization*, vol. 2632, pp. 519-533, 2003.

[52] C. M. Fonseca and P. J. Fleming, "Multi-objective genetic algorithm made easy: Selection, sharing and mating restriction," in *International Conference on Genetic Algorithm in Engineering Systems: Innovations and Application*, pp. 12-14. 1995.

[53] C. M. Fonseca and P. J. Fleming, "Multiobjective Optimal Controller Design with Genetic Algorithms," in *Proceedings on IEE Control*, pp. 745-749, 1994.

[54] C. M. Fonseca and P. J. Fleming, "Genetic algorithm for multiobjective optimization, formulation, discussion and generalization," in *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 416-423, 1993.

[55] P. M Franca, A. Mendes, and P. Moscato, "A memetic algorithm for the total tardiness single machine scheduling problem," *European Journal Of Operational Research*, vol. 132, no. 1, pp. 224-242, 2001.

[56] Yoshikuzu Fukuyama and Hirotata Yoshida, "A Particle Swarm Optimization for Reactive Power and Voltage Control in Electric Power Systems," in *Proceedings of the 2001 Congress on Evolutionary Computation*, Seoul, Korea, 2001

[57] N. Garcia-Pedrajas, C. Hervas-Martinez, and D. Ortiz-Boyer, "Cooperative Coevolution of Artificial Neural Network Ensembles for Pattern Classification," *IEEE Transactions on Evolutionary Computation*, vol.9, no.3, pp 271-302, 2005.

[58] C. K. Goh and K. C. Tan, "An investigation on noisy environments in evolutionary multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 3, pp. 354-381, 2007.

[59] D. E. Goldberg, Genetic Algorithms for Search, Optimization, and Machine Learning, Addison-Wesley, 1989.

[60] D. E. Goldberg, "Sizing populations for serial and parallel genetic algorithms," in *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 70-79, 1989.

[61] N. Hallam, P. Blanchfield, and G. Kendall, "Handling Diversity in Evolutionary Multiobjective Optimisation," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, pp. 2233-2240, 2005.

[62] F. Heppner and U. Grenander, "A stochastic nonlinear model for coordinated bird flocks," The Ubiquity of Chaos. Ed. S. Kranser, AAAS Publications, Washington, D. C., 1990.

[63] D. W. Hillis,"Coevolving parasites improve simulated evolution as an optimization procedure," Artificial Life 2, (eds.) C. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, pp. 313-324, 1991.

[64] T. Hiroyasu, S. Nakayama and M. Miki,"Comparison Study of SPEA2+, SPEA2, and NSGA-II in Diesel Engine Emissions and Fuel Economy Problem," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, pp. 236-242, 2005.

[65] S. L. Ho, Shiyou Yang, Guangzheng Ni, Edward W. C. Lo, and H. C. Wong,"A particle swarm optimizationbased method for multiobjective design optimizations," *IEEE Transactions on Magnetics*, vol. 41, no. 5, pp. 1756-1759, 2005.

[66] S. Y. Ho, S. S. Li, and J. H. Chen, "Intelligent evolutionary algorithms for large parameter optimization problems," *IEEE Transactions on Evolutionary Computation*, Vol. 8, No. 6, pp. 532-541, 2004.

[67] J. H. Holland, Adaptation in Natural Artificial Systems: An Introductory Analysis with Applocations to Biology, Control, and Artificial Intelligence, MIT press, 1992.

[68] E. Hopper and B.C.H.Turton, "An Empirical Investigation of Meta-heuristic and Heuristic Algorithms for a 2D packing problem," *European Journal of Operation Research*, vol. 128, pp. 34-57, 2001.

[69] E. Hopper and B. Turton, "A Genetic Algorithm for a 2D Industrial Packing Problem," *Computers & Industrial Engineering*, vol. 37, pp. 375-378, 1999.

[70] J. Horn and N. Nafpliotis, "Multiobjective optimization using the niched Pareto genetic algorithm," Technical Report No. 930005, Illinois Genetic Algorithms Laboraatory (IlliGAL), University of Illinois at Urbana-Champaign, 1993.

[71] Xiaohui Hu, Russell C. Eberhart, and Yuhui Shi, "Particle swarm with extended memory for multiobjective optimization," in *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, pp. 193-197, 2003.

[72] Xiaohui Hu and Russell Eberhart, "Multiobjective optimization using dynamic neighborhood particle swarm optimization," in *Proceedings of Congress on Evolutionary Computation*, vol. 2, pp. 1677-1681, 2002.

[73] E. J. Hughes, "Evolutionary Many-Objective Optimisation: Many Once or One Many?," in *Proceedings of 2005 IEEE Congress on Evolutionary Computation*, vol. 1, pp. 222-227, 2005.

[74] E. J. Hughes, "Multiple single objective pareto sampling, " in *Proceedings of 2003 IEEE Congress on Evolutionary Computation*, pp. 26782684, 2003.

[75] E. J. Hughes, "Evolutionary multi-objective ranking with uncertainty and noise," in *Proceedings of the First Conference on Evolutionary Multi-Criterion Optimization*, pp. 329-343, 2001.

[76] S. M. Hwang and C. Y. Kao, "On Solving Bin Packing Problems Using Genetic Algorithms," *IEEE International Conference on Systems, Man and Cybernetics (Humans, Information and Technology)*, vol. 2, pp. 1583-1590, 1994.

[77] H. Iima and Yakawa, "A New Design of Genetic Algorithm for Bin Packing," in *Proceedings of the Congress on Evolutionary Computation*, vol. 2, pp. 1044-1049, 2003.

[78] K. Ikeda, H. Kita, and S. Kobayashi, "Does Non-dominated Really Mean Near to Optimal? " in *Proceedings of the 2001 IEEE Conference on Evolutionary Computation*, vol. 2, pp. 957-962, 2001.

[79] A. W. Iorio and X. Li, "A Cooperative Coevolutionary Multiobjective Algorithm Using Non-dominated Sorting," in *Proceedings of the 2004 Genetic and Evolutionary Computation Congress*, pp. 537-548, 2004

[80] H. Ishibuchi and T. Murata, "A Multi-Objective Genetic Local Search Algorithm and Its Application to Flowshop Scheduling," *IEEE Transactions on Systems, Man, and Cybernetics - Part C*, vol. 28, no. 3, pp. 392-403, 1998.

[81] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between Genetic Search and Local Search in Memetic Algorithms for Multiobjective Permutation Flowshop," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 204-223, 2003

[82] S. Jakobs, "On Genetic Algorithms for the Packing of Polygons," *European Journal of Operational Research*, vol. 88, pp. 165-181, 1996.

[83] Stefan Janson and Daniel Merkle, "A new multiobjective particle swarm optimization algorithm using clustering applied to automated docking," in *Proceedings of the Second International Workshop on Hybrid Metaheuristics*, pp. 128-142, 2005.

[84] A. Jaszkiewicz, "On the Performance of Multiple-Objective Genetic Local Search on the 0/1 Knapsack Problem-A Comparative Experiment," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 4, pp. 402-412, 2002.

[85] A. Jaszkiewicz, "Do multi-objective metaheuristics deliver on their promises? A computational experiment on the set-covering problem," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 133-143, 2003.

[86] Y. Jin, T. Okabe and B. Sendhoff, "Adapting Weighted Aggregation for Multiobjective Evolution Strategies," in *Proceedings of the First Conference on Evolutionary Multi-Criterion Optimization*, pp. 96-110, 2001.

[87] Y. Jin, M. Olhofer and B. Sendhoff, "Dynamic Weighted Aggregation for Evolutionary Multi-Objective Optimization: Why Does It Work and How?," in *Proceedings of the 2001 Genetic and Evolutionary Computation Conference*, pp. 1042-1049, 2001.

[88] C. Y. Kao and F. T. Lin, "A Stochastic Approach for the One-Dimensional Bin-Packing Problems," *IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, pp. 1545-1551, 1992.

[89] N. Keerativuttiumrong, N. Chaiyaratana and V. Varavithya, "Multiobjective cooperative coevolutionary genetic algorithm," in *Proceedings of the Seventh International Conference on Parallel Problem Solving from Nature*, pp. 288-297, 2002.

[90] J. Kennedy and R. C. Eberhart, Swarm Intelligence, Morgan Kaufmann Publishers, 2001.

[91] J. Kennedy and R. Eberhart, "A Discrete Binary Version of the Particle Swarm Algorithm," in *Proceedings of the International Conference on Systems, Man and Cybernetics*, Piscataway, NJ, 1997.

[92] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942-1948, 1995.

[93] J. Kennedy and R. C. Eberhart, "A new optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, pp. 39-43, 1995.

[94] V. Khare, X. Yao and B. Sendhoff, "Credit assignment among neurons in co-evolving populations," in *Proceedings of the Eighth International Conference on Parallel Problem Solving from Nature*, pp. 882-891, 2004.

[95] V. Khare, X. Yao and K. Deb, "Performance scaling of multi-objective evolutionary algorithms," in *Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization*, pp. 376-390, 2003.

[96] E. F. Khor, K. C. Tan, T. H. Lee, and C. K. Goh, "A study on distribution preservation mechanism in evolutionary multi-objective optimization," *Artificial Intelligence Review*, vol. 23, no. 1, pp. 31-56, 2005.

[97] KE. F. Khor, K. C. Tan, and T. H. Lee, "Tabu-based exploratory evolutionary algorithm for effective multi-objective optimization," in *Proceedings of the First Conference on Evolutionary Multi-Criterion Optimization*, pp. 344-358, 2001.

[98] J. Kim and B. P. Zeigler, "A Framework for Multiresolution Optimization in a Parallel/Distributed Environment: Simulation of Hierarchical GAs," *Journal of Parallel and Distributed Computing*, vol. 32, pp. 90-102, 1996.

[99] H. Kita, Y. Yabumoto, N. Mori, and Y. Nishikawa, "Multi- Objective Optimization by Means of the Thermodynamical Genetic Algorithm," in *Proceedings of the Fourth Parallel Problem Solving from Nature*, pp. 504-512, 1996.

[100] J. D. Knowles, D. W. Corne and M. Fleischer, "Bounded archiving using the Lebesgue measure," in *Proceedings of the 2003 IEEE Congress on Evolutionary Computation*, vol. 4, pp. 2490-249, 2003.

[101] J. D. Knowles, and D. W. Corne, "Properties of an adaptive archiving algorithm for storing nondominated vectors," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 100-116, 2003.

[102] J. D. Knowles, and D. W. Corne, "On Metrics for Comparing Nondominated Sets," in *Proceedings of the 2002 IEEE Congress on Evolutionary Computation*, vol. 1, pp. 711-716, 2002.

[103] J. D. Knowles, and D. W. Corne, "Approximating the non-dominated front using the Pareto archived evolution strategy," *Evolutionary Computation*, vol. 8, no. 2, pp. 149-172, 2000.

[104] N. Krasnogor and J. E. Smith, "A Tutorial for Competent Memetic Algorithms: Model, Taxonomy and Design Issues," *IEEE Transactions on Evolutionary Computation*, 2005.

[105] F. Kursawe, "A Variant of Evolution Strategies for Vector Optimization," in *Proceedings of the Firsth International Conference on Parallel Problem Solving from Nature*, vol. 496, pp. 193-197, 1991.

[106] M. Laumanns, L. Thiele, E. Zitzler, and K. Deb "Archiving with Guaranteed Convergence and Diversity in Multi-Objective Optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 439-447, 2002.

[107] M. Laumanns, E. Zitzler, and L. Thiele, "On the effects of archiving, elitism, and density based selection in evolutionary multi-objective optimization," in *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization*, pp. 181-196, 2001.

[108] M. Laumanns, E. Zitzler, and L. Thiele, "A unified model for multi-objective evolutionary algorithms with elitism," in *Proceedings of the 2000 IEEE Congress on Evolutionary Computation*, vol. 1, pp. 46-53, 2000.

[109] Xiaodong Li, "A nondominated sorting particle swarm optimizer for multiobjective optimization," in *Proceedings of the 2003 Genetic and Evolutionary Computation Conference*, Berlin, Germany, pp. 37-48, July 2003.

[110] Xiaodong Li, "Better spread and convergence: Particle swarm multiobjective optimization using the maximin fitness function," in *Proceedings of the 2004 Genetic and Evolutionary Computation Conference*, pp. 117-128, 2004.

[111] Chengfei Li, Qunxiong Zhu, and Zhiqiang Geng, "Multi-objective Particle Swarm Optimization Hybrid Algorithm: An Application on Industrial Cracking Furnace," *Ind. Eng. Chem. Res.*, vol. 46, pp. 3602-3609, 2007.

Springer, in press.

[112] V. I. Litvinenko, J. A. Burgher, A. A. Tkachuk, and V. J. Gnatjuk, "The Application of the Distributed Genetic Algorithm to the Decision of the Packing in Containers Problem," *IEEE International Conference on Artificial Intelligence Systems*, pp. 386-390, 2002.

[113] D. Liu and H. Teng, "An Improved BL-algorithm for Genetic Algorithm of the Orthogonal Packing of Rectangles," *European Journal of Operational Research*, vol. 112, pp. 413-420, 1999.

[114] T. H. Liu and K. J. Mills, "Robotic Trajectory Control System Design for Multiple Simultaneous Specifications: Theory and Experimentation," in *Transactions on ASME*, vol. 120, pp. 520-523. 1998.

[115] Y. Liu, X. Yao, Q. Zhao and T. Higuchi, "Scaling up fast evolutionary programming with cooperative coevolution," in *Proceedings of the 2001 Congress on Evolutionary Computation*, pp. 1101-1108, 2001.

[116] A. Lodi, S. Martello, and D. Vigo, "Heuristic algorithms for the Three-Dimensional Bin-Packing Problem," *European Journal of Operational Research*, vol. 141, pp. 410-420, 2002.

[117] J. D. Lohn, W. F. Kraus and G. L. Haith, "Comparing a coevolutionary genetic algorithm for multiobjective optimization," in *Proceedings of the 2002 IEEE Congress on Evolutionary Computation*, pp. 1157-1162, 2002.

[118] H. Lu and G. G. Yen, "Rank-based multiobjective genetic algorithm and benchmark test function study," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 4, pp. 325-343, 2003.

[119] G. C. Luh, C. H. Chueh, and W. W. Liu, "MOIA: Multi-Objective Immune Algorithm," *Engineering Optimization*, vol. 35, no. 2, pp. 143-164, 2003.

[120] Mahdi Mahfouf, Min-You Chen, and Derek Arturh Linkens, "Adaptive weighted particle swarm optimisation for multi-objective optimal design of alloy steels," *Parallel Problem Solving from Nature - PPSN VIII*, pp. 762-771, 2004.

[121] K. Maneeratana, K. Boonlong and N. Chaiyaratana, "Multi-objective Optimisation by Co-operative Co-evolution," in *Proceedings of the Eighth International Conference on Parallel Problem Solving from Nature*, pp. 772-781, 2004.

[122] S. Martello, D. Pisinger, and D. Vigo, "The Three Dimensional Bin Packing Problem," *Operations Research*, vol. 48, pp. 256-267, 2000.

[123] S. Martello and D. Vigo, "Exact Solution of the Two-Dimensional Finite Bin Packing Problem," *Management Science*, vol. 44, pp. 388-399, 1998.

[124] P. Merz and B. Freisleben, "Fitness landscape analysis and memetic algorithms for the quadratic assignment problem," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 4, pp. 337-352, 2000.

[125] P. Merz and B. Freisleben, "A comparison of memetic algorithms, Tabu search, and ant colonies for the quadratic assignment problem," in *Proceedings of the 1999 IEEE Congress on Evolutionary Computation*, vol. 1, pp. 2063-2070, 1999.

[126] M. M. Millonas, "Swarms, phase transitions, and collective intelligence," Artificial Life III. Ed. C. G. Langton, Addison Wesley, Reading, MA, 1994.

[127] N. E. Mendoza, Y. W. Chen, Z. Nakao, T. Adachi, Y. Masuda, "A real multi-parent tri-hybrid evolutionary optimization method and its application in wind velocity estimation from wind profiler data, " *Applied Soft Computing Journal*, vol. 1, no. 3, pp. 225-235, 2001

[128] M. Mongeau and C. Bes, "Optimization of aircraft container loading," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, pp. 140-150, 2003.

[129] P. Moscato, "On evolution, search, optimization, GAs and martial arts: toward memetic algorithm," California Inst. Technol., Pasadena, CA, Tech. Rep. Caltech Concurernt Comput. Prog. Rep. 826, 1989.

[130] S. Mostaghim and J. Teich, "Strategies for finding good local guides in Multi-Objective Particle Swarm Optimization (MOPSO)," in *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, Indianapolis, IN, pp.26-33, 2003.

[131] S. Mostaghim and J. Teich, "The Role of ϵ-dominance in Multi Objective Particle Swarm Optimization Methods," in *Proceedings of the 2003 IEEE Congress on Evolutionary Computation*, vol. 3, pp. 1764-1771, 2003.

[132] S. Mostaghim and J. Teich, "Covering paretooptimal fronts by subswarms in multi-objective particle swarm optimization," in *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, vol. 2, pp. 1404-1411, 2004.

[133] C. L. Mumford, "A Hierarchical Solve-and-Merge Framework for Multi-Objective Optimization," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, pp. 2241-2247, 2005.

[134] T. Murata and H. Ishibuchi, "MOGA: Multi-objective genetic algorithms," in *Proceedings of the 1995 IEEE Congress on Evolutionary Computation*, pp. 289-294, 1995.

[135] D. Naso, B. Turchiano, and C. Meloni, "Single and multi-objective evolutionary algorithms for the coordination of serial manufacturing operations," *Journal of Intelligent Manufacturing*, vol. 17, no. 2, pp. 249-268, 2006.

[136] M. Nerome, K. Yamada, S. Endo, and H. Miyagi, "Competitive Co-evolution Based Game-Strategy Acquisition with the Packaging," in *Proceedings of the Second International Conference on Knowledge-Based Intelligent Electronic Systems*, pp 184-189, 1998.

[137] T. Okabe, Y. Jin, B. Sendhoff, and M. Olhofer, "Voronoi-based estimation of distribution algorithm for multi-objective optimization," in *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, pp. 1594-1601, 2004.

[138] T. Okuda, T. Hiroyasu, M. Miki, S. Watanabe, "DCMOGA: Distributed Cooperation model of Multi-Objective Genetic Algorithm" in *Proceedings of the Seventh International Conference on Parallel Problem Solving from Nature*, pp. 155-160, 2002.

[139] Y. S, Ong and A. J. Keane, "Meta-Lamarckian Learning in Memetic Algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 99-110, 2004.

[140] Y. S. Ong, P. B. Nair , K. Y. Lum, "Min-Max Surrogate Assisted Evolutionary Algorithm for Robust Aerodynamic Design," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 392-404, 2006.

[141] A. Osyczka and S. Krenich, "Evolutionary Algorithms for Multicriteria Optimization with Selecting a Representative Subset of Pareto Optimal Solutions," in *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization*, pp. 141-153, 2001.

[142] J. Paredis, "Coevolutionary constraint satisfaction," in in *Proceedings of the Third International Conference on Parallel Problem Solving from Nature*, pp. 46-55, 1994.

[143] R. P. Pargas and R. Jain, "A Parallel Stochastic Optimization Algorithm for Solving 2D Bin Packing Problems," in *Proceedings of the 9th Conference on Artificial Intelligence for Applications*, pp. 18-25, 1993.

[144] G. Parks, J. Li, M. Balazs and I. Miller, "An empirical investigation of elitism in multiobjective genetic algorithms," *Foundations of Computing and Decision Sciences*, vol. 26, no. 1, pp. 51-74, 2001.

[145] Konstantinos E. Parsopoulous and Michael N. Vrahatis, "Particle Swarm Optimization Method in Multimobjective Problems," in *Proceedings of the 2002 ACM Symposium on Applied Computing*, Madrid, Spain, pp.603-607, 2002.

[146] Konstantinos E. Parsopoulos, Dimitris K. Tasoulis, and Michael N. Vrahatis, "Multiobjective optimization using parallel vector evaluated particle swarm optimization," in *Proceedings of the 2004 IASTED International Conference on Artificial Intelligence and Applications*, vol. 2, pp. 823-828, 2004.

[147] E. Parzen, "On the estimation of a probability density function and mode," *Annals of Mathematical Statistics*, vol. 33, pp. 1065-1076, 1962.

[148] C. Pimpawat and N. Chaiyaratana, "Using a Co-Operative Co-Evolutionary Genetic Algorithm to Solve a Three-Dimensional Container Loading Problem," *Congress on Evolutionary Computation*, vol. 2, pp. 1197-1204, 2001.

[149] C. Poloni et al. "Hybridization of a multiobjective genetic algorithm: a neural network and a classical optimizer for a complex design problem in fluid dynamics," *Computer Methods in Applied Mechanics and Engineering*, 186(2-4):402-420, 2000.

[150] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *Proceedings of the Third International Conference on Parallel Problem Solving from Nature*, Berlin, Germany, pp. 249-257, 1994.

[151] M. A. Potter, "The Design and Analysis of a Computational Model of Cooperative Coevolution," Ph.D Thesis, George Mason University, 1997.

[152] M. A. Potter and K. A. De Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evolutionary Computation*, vol. 8, no. 1, pp. 1-29, 2000.

[153] Carlo R. Raquel and Jr. Prospero C. Naval, "An effective use of crowding distance in multiobjective particle swarm optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 257-264, 2005.

[154] F. L. W. Ratnieks, "Cooperation through coercion: policing of male production and female caste fate in honey bees and stingless bee," in Ed. C. V. Garfalo, Encontro sobre abelhas, pp. 10-14, 2002.

[155] Tapabrata Ray and K. M. Liew, "A swarm metaphor for multiobjective design optimization," *Engineering Optimization*, vol. 34, no. 2, pp. 141-153, 2002.

[156] C. R. Reeves, Modern Heuristic Techniques for Combinatorial Problems, Blackwell Scientific Publication, 1993.

[157] Margarita Reyes Sierra and Carlos A. Coello Coello, "Improving PSO-based multiobjective optimization using crowding, mutation and ϵ-dominance," in *Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization*, pp. 505-519, 2005.

[158] C. W. Reynolds, "Flocks, herds and schools: a distributed behavioral model," *Computer Graphics*, 21(4):25-34, 1987.

[159] W. Rivera, "Scalable parallel genetic algorithms," *Artificial Intelligence Review*, vol. 16, pp. 153-168, 2001.

[160] C. D. Rosin and R. K. Belew, "New methods for competitive coevolution," *Evolutionary Computation*, vol. 5, no. 1, pp. 1-29, 1997.

[161] J. Rowe, K. Vinsen and N. Marvin, "Parallel GAs for Multiobjective Functions," in *Second Nordic Workshop on Genetic Algorithms and Their Applications*, pp. 61-70, 1996.

[162] G. Rudolph and A. Agapie, "Convergence Properties of Some Multi-Objective Evolutionary Algorithms,"in *Proceedings of the 2000 Conference on Evolutionary Computation*, pp. 1010-1016, 2000.

[163] G. Rudolph, "On a Multi-Objective Evolutionary Algorithm and Its Convergence to the Pareto Set," in *Proceedings of the 1998 Conference on Evolutionary Computation*, pp. 511-516, 1998.

[164] T. P. Runarsson, M. T. Jonsson, and P. Jensson, "Dynamic Dual Bin Packing Using Fuzzy objectives," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 219-222, 1996.

[165] Maximino Salazar-Lechuga and Jonathan Rowe, "Particle swarm optimization and fitness sharing to solve multi-objective optimization problems," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, pp. 1204-1211, 2005.

[166] R. Sarker, K. Liang, and C. Newton, "A New Evolutionary Algorithm for Multiobjective Optimization," *European Journal of Operational Research*, vol. 140, no. 1, pp. 12-23, 2002.

[167] H. Sato, H. E. Aguirre and K. Tanaka, "Enhanced Multi-objective Evolutionary Algorithms Using Local Dominance," in *Proceedings of the 2004 RISP International Workshop on Nonlinear Circuits and Signal Processing*, pp. 319-322, 2004.

[168] J. D. Schaffer, "Multi-Objective Optimization with Vector Evaluated Genetic Algorithms," in *Proceedings of the First International Conference on Genetic Algorithms*, pp. 93-100, 1985.

[169] W. M. Schaffer, D. W. Zeh, S. L. Buchmann, S. Kleinhaus, M. V. Schaffer, and J. Antrim, "Competition for nectar between introduced honeybees and native North American bees and ants," *Ecology*, vol. 64, pp. 564-577, 1983.

[170] A. Scholl, R. Klein, and C. Jurgens, "Bison: A Fast Hybrid Procedure for Exactly Solving the One-Dimensional Bin Packing Problem," *Computers & Operations Research*, vol. 24, Issue 7, pp. 627-645, 1997.

[171] H. P. Schwefel, Evolution and Optimum Seeking. John Wiley & Sons, 1995.

[172] J. R. Scott, "Fault Tolerant Design Using Single and Multi-criteria Genetic Algorithms,"Master Thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, 1995.

[173] K. J. Shaw, A. L. Notcliffe, M. Thompson, J. Love, C. M. Fonseca, and P. J. Fleming, "Assessing the performance of multiobjective genetic algorithms for optimization of

batch process scheduling problem," in *Proceedings of the Conference on Evolutionary Computation*, vol. 1:37-45, 1999.

[174] Y. Shi and R. Eberhart, "Empirical Study of Particle Swarm Optimization," in *Proceedings of the 1999 Congress on Evolutionary Computation*, Washington D. C., pp. 1945-1950, 1999.

[175] Y. Shi and R. Eberhart, "Parameter Selection in Particle Swarm Optimization," in *Proceedings of the Seventh Annual Conference on Evolutionary Programming*, pp. 591-601, 1998.

[176] Y. Shi and R. Eberhart, "A Modified Particle Swarm Optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, Anchorage, Alaska, May 4-9, 1998.

[177] Y. Shigehiro, S. Koshiyama, and T. Masuda, "Stochastic Tabu Search for Rectangle Packing," *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 4, pp. 2753-2758, 2001.

[178] B. W. Silverman, Density estimation for statistics and data analysis, London: Chapman and Hall, 1986.

[179] K. B. Sim, J. Y. Kim and D. W. Lee, "Game Theory Based Coevolutionary Algorithm: A New Computational Coevolutionary Approach," *International Journal of Contol, Automation, and Systems*, vol. 2, no. 4, pp. 463-474, 2004.

[180] D. Sofge, K. A. De Jong, and A. Schultz, "A blended population approach to cooperative coevolution for decomposition of complex problems," in *Proceedings of the 2002 Congress on Evolutionary Computation*, Honolulu, Hawaii, pp. 413-418, 2002.

[181] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations Research*, vol. 35, no. 2, pp. 254-265, 1987.

[182] R. V. Southwell,"Relaxation Methods in Theoretical Physics" Clarendon Press, 1946.

[183] R. Spillman, "Solving Large Knapsack Problems with a Genetic Algorithm," *IEEE International Conference on Systems, Man and Cybernetics, 'Intelligent Systems for the 21st Century'*, vol. 1, pp. 632-637, 1995.

[184] N. Srinivas and K. Deb, "Multiobjective optimization using non-dominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221-248, 1994.

[185] D. Srinivasan, W. H. Loo and R. L. Cheu, "Traffic Incident Detection Using Particle Swarm Optimization," in *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, pp144 C 151, 2003

[186] D. Srinivasan and T. H. Seow, "Particle swarm inspired evolutionary algorithm (PS-EA) for multiobjective optimization problems," in *Proceedings of IEEE Congress on Evolutionary Computation 2003 (CEC 2003)*, pp. 2292-2297, 2003.

[187] K. C. Tan, C. K. Goh, A. A. Mamun and E. E. Zin, "An Evolutionary Artificial Immune System for Multi-Objective Optimization," *European Journal of Operational Research*, in press.

[188] K. C. Tan, C. Y. Cheong and C. K. Goh, "Solving multiobjective vehicle routing problem with stochastic demand via evolutionary computation" *European Journal of Operational Research*, vol. 177, pp. 813-839, 2007.

[189] K. C. Tan, Y. H. Chew, and L. H. Lee, "A hybrid multiobjective evolutionary algorithm for solving truck and trailer vehicle routing problems," *European Journal of Operational Research*, vol. 172, pp. 855-885, 2006.

[190] K. C. Tan, Q. Yu and J. H. Ang, "A coevolutionary algorithm for rules discovery in data mining," *International Journal of Systems Science*, vol. 37, no. 12, pp. 835-864, 2006.

[191] K. C. Tan, Y. J. Yang, and C. K. Goh, "A distributed cooperative coevolutionary algorithm for multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 5, pp. 527-549, 2006.

[192] K. C. Tan, C. K. Goh, Y. J. Yang, and T. H. Lee, "Evolving better population distribution and exploration in evolutionary multi-objective optimization," *European Journal of Operational Research*, vol. 171, no. 2, pp. 463-495, 2006.

[193] K. C. Tan, E. F. Khor and T. H. Lee, Multiobjective Evolutionary Algorithms and Applications, Springer Berlin Heidelberg, 2005

[194] K. C. Tan, T. H. Lee, Y. J. Yang, and D. S. Liu, "A Cooperative Coevolutionary Algorithm for Multiobjective Optimization," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 1926-1931, 2004.

[195] K. C. Tan, E. F. Khor, T. H. Lee and R. Sathikannan, "An evolutionary algorithm with advanced goal and priority specification for multiobjective optimization," *Journal of Artificial Intelligence Research*, vol. 18, pp. 183-215, 2003.

[196] K. C. Tan, T. H. Lee, Y. H. Chew and L. H. Lee, "A hybrid multiobjective evolutionary algorithm for solving truck and trailer vehicle routing problems," in *Proceedings of the IEEE Congress on Evolutionary Computation 2003*, vol. 3, pp. 2134-2141, 2003.

[197] K. C. Tan, T. H. Lee, and E. F. Khor, "Evolutionary algorithms for multi-objective optimization: performance assessments and comparisons," *Artificial Intelligence Review*, vol. 17, no. 4, pp. 251-290, 2002.

[198] K. C. Tan, T. H. Lee and E. F. Khor, "Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 6, pp. 565-588, 2001.

[199] D. Teodorovic and P. Lucic, "Intelligent vehicle routing system," in *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, pp. 482487, 2000.

[200] D. Teodorovic and G. Pavkovic, "The fuzzy set theory approach to the vehicle routing problem when demand at nodes is uncertain," *Fuzzy Sets and Systems*, vol. 82, no. 3, pp. 307-317, 1996.

[201] H. A. Thompson and P. J. Fleming,"An Integrated Multi-Disciplinary Optimisation Environment for Distributed Aero-engine Control System Arhitectures," in *Proceedings of the Fourteenth World Congress of International Federation of Automatic Control*, pp. 407-412. 1999.

[202] T. O. Ting, M. V. C. Rao, C. K. Loo, and Sze-San Ngu, "A New Class of Operators to Accelerate Particle Swarm Optimization," *Congress on Evolutionary Computation*, vol.4, pp. 2406 C 2410, 2003.

[203] A. Toffolo and E. Benini,"Genetic Diversity as an Objective in Multi-Objective Evolutionary Algorithms," *Evolutionary Computation*, vol. 11, no. 2, pp. 151-167, 2003.

[204] A. Turkcan and M. S. Akturk, "A problem space genetic algorithm in multiobjective optimization," *Journal of Intelligent Manufacturing*, vol. 14, pp. 363-378, 2003.

[205] F. Van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225-239, 2004.

[206] H. Van De Vel and S. J. Sun, "An application of the Bin Packing Technique to Job Scheduling on Uniform Processors," *Operation Research*, vol. 42, no. 2, pp. 169-172, 1991.

[207] G. Venter and R. T. Haftka, "A Two Species Genetic Algorithm for Designing Composite Laminates Subjected to Uncertainty," in *Proceedings of the37th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, pp. 1848-1857, 1996.

[208] F. Vavak, K. Jukes, and T. C. Fogarty, "Adaptive combustion balancing in multiple burner boiler using a genetic algorithm with variable range of local search, "in *Proceedings of the Seventh International Conference on Genetic Algorithms*, pp. 719-726, 1997.

[209] D. A. Van Veldhuizen, J. B. Zydallis and G. B. Lamont, "Considerations in engineering parallel multiobjective evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 144-173, 2003.

[210] D. A. Van. Veldhuizen and G. B. Lamont, "On measuring multiobjective evolutionary algorithm performance," in *Proceedings of the 2000 IEEE Congress on Evolutionary Computation*, vol. 1, pp. 204-211, 2000.

[211] D. A. V. Veldhuizen and G. B. Lamont, "Multiobjective Evolutionary Algorithm Test Suites, " *ACM Symposium on Applied Computing*, pp. 351-357, 1999.

[212] D. A. Van Veldhuizen and G. B. Lamont, "Multiobjective Evolutionary Algorithm Research: A History and Analysis," Technical Report TR-98-03, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Ohio, 1998.

[213] Mario Alberto Villalobos-Arias, Gregorio Toscano Pulido, and Carlos A. Coello Coello, "A proposal to use stripes to maintain diversity in a multi-objective particle swarm optimizer," in *Proceedings of the 2005 IEEE Swarm Intelligence Symposium*, pp. 22-29, 2005.

[214] K. P. Wang, L. Huang, C. G. Zhou, and W. Pang, "Particle Swarm Optimization for Traveling Salesman Problem," *International Conference on Machine Learning and Cybernetics*, vol. 3 , pp. 1583-1585, 2003.

[215] M. Y. Wu and W. Shu, "An Efficient Distributed Token-Based Mutual Exclusion Algorithm with Central Coordinator," *Journal of Parallel and Distributed Computing*, vol. 62, pp. 1602-1613, 2002.

[216] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks, " *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 694-713, 1997.

[217] X. Yao and Y. Liu, "Making use of population information in evolutionary artificial neural networks, "*IEEE Transaction on Systems, Man, and Cybernetics- Part B: Cybernetics*, vol. 28, pp. 417-425, 1998.

[218] H. W. Yeung, and K. S. Tang, "A Hybrid Genetic Approach for Container Loading in Logistics Industry, " *IEEE Transactions on Industrial Electronics*, vol. 52, no. 2, pp. 617-627, 2005.

[219] Hirotata Yoshida, Kenichi Kawata, and Yoshikuzu Fukuyama, "A Particle Swarm Optimization for Reactive Power and Voltage Control in Electric Power Systems Considering Voltage Security Assessment," *IEEE Transactions on Power Systems*, Vol. 15, No. 4, 2000

[220] L. B. Zhang, C. G. Zhou, X. H. Liu, Z. Q. Ma, and Y. C. Liang, "Solving multi objective optimization problems using particle swarm optimization," in *Proceedings of the 2003 IEEE Congress on Evolutionary Computation*, vol. 3, pp. 2400-2405, 2003.

[221] Xiao-hua Zhang , Hong-yun Meng , and Li-cheng Jiao, "Intelligent Particle Swarm Optimization in Multiobjective Optimization," in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, pp. 714-719, 2005.

[222] E. Zitzler and S. Kunzli, "Indicator-Based Selection in Multiobjective Search," in *Proceedings of the Eighth International Conference on Parallel Problem Solving from Nature*, pp. 832-842, 2004.

[223] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca and V. G. Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review,"*IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117-132, 2003.

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

[225] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173-195, 2000.

[226] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257-271, 1999.

[227] Zitzler, Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications, Ph.D Thesis, Swiss Federal Institute of Technology, Zurich, 1999.