EVOLUTIONARY ALGORITHM FOR MULTIOBJECTIVE OPTIMIZATION: COOPERATIVE COEVOLUTION AND NEW FEATURES

YANG YINGJIE
(B. Eng, Tsinghua University)

A THESIS SUBMITTED
FOR THE DEGREE OF MASTER OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
NATIONAL UNIVERSITY OF SINGAPORE

2004

Acknowledgements

I would like to express my most sincere appreciation to my supervisor, Dr. Tan Kay Chen, for his good guidance, support and encouragement. His stimulating advice is of great benefit to me in overcoming obstacles on my research path.

Deep thanks go to my friends and fellows Khor Eik Fun, Cai Ji, Goh Chi Keong, who have made contributions in various ways to my research work.

I am also grateful to all the individuals in the Center for Intelligent Control (CIC), as well as the Control and Simulation Lab, Department of Electrical and Computer Engineering, National University of Singapore, which provides the research facilities to conduct the research work.

Finally, I wish to acknowledge National University of Singapore (NUS) for the financial support provided throughout my research work.

Table of Contents

Summary

This work seeks to explore and improve the evolutionary techniques for multi-objective optimization. First, an introduction of multiobjective optimization is given and key concepts of multiobjective evolutionary optimization are discussed. Then a cooperative coevolution mechanism is applied in the multiobjective optimization. Exploiting the inherent parallelism in cooperative coevolution, the algorithm is formulated into a distributed computing structure to reduce the runtime. To improve the performance of multiobjective evolutionary algorithms, an adaptive mutation operator and an enhanced exploration strategy are proposed. Finally, the direction of future research is pointed out.

The cooperative coevolutionary algorithm (CCEA) evolves multiple solutions in the form of cooperative subpopulations and uses an archive to store non-dominated solutions and evaluate individuals in the subpopulations based on Pareto dominance. The dynamic sharing is applied to maintain the diversity of solutions in the archive. Moreover, an extending operator is designed to mine information on solution distribution from the archive and guide the search to regions that are not well explored so that CCEA can distribute the non-dominated solutions in the archive evenly and endow the solution set with a wide spread. The extensive quantitative comparisons show that CCEA has excellent performance in finding the non-dominated solution set with good convergence and uniform distribution.

Exploiting the inherent parallelism in cooperative coevolution, a distributed CCEA (DCCEA) is developed by formulating the algorithm into a computing structure suitable for parallel processing where computers over the network share the computational workload. The computational results show that DCCEA can dramatically reduce the runtime without sacrificing the performance as the number of peer computers increases.

The adaptive mutation operator (AMO) adapts the mutation rate to maintain a balance between the introduction of diversity and local fine-tuning. It uses a new approach to strike a compromise between the preservation and disruption of genetic information. The enhanced exploration strategy (EES) maintains diversity and non-dominated solutions in the evolving population while encouraging the exploration towards less populated areas. It achieves better discovery of gaps in the discovered Pareto front as well as better convergence. Simulations are carried out to examine the effects of AMO and EES with respect to selected mutation and diversity operators respectively. AMO and EES have shown to be competitive if not better than their counterparts and have their own specific contribution. Simulation results also show that the algorithm incorporated with AMO and EES is capable of discovering and distributing non-dominated solutions along the Pareto front.

List of Abbreviations

AMO	Adaptive mutation operator
CCEA	Cooperative coevolutionary algorithm
DCCEA	Distributed cooperative coevolutionary algorithm
DEC	Distributed evolutionary computing
EA	Evolutionary algorithm
EES	Enhanced exploration strategy
GA	Genetic algorithm
GD	Generational distance
HLGA	Hajela and Lin's genetic algorithm
HV	Hyper-Volume
HVR	Hyper-Volume ratio
IMOEA	Incrementing multiobjective evolutionary algorithm
MIMOGA	Murata and Ishibuchi's multiobjective genetic algorithm
MO	Multiobjective
MOEA	Multiobjective evolutionary algorithm
MOGA	Multiobjective genetic algorithm
MS	Maximum spread
NPGA	Niched Pareto genetic algorithm
NSGA II	Non-Dominated sorting genetic algorithm II
PAES	Pareto archived evolutionary strategy
PESA	Pareto envelope based selection algorithm

S	Spacing
SPEA	Strength Pareto evolutionary algorithm
SPEA 2	Strength Pareto evolutionary algorithm 2
VEGA	Vector evaluated genetic algorithm

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Statement of the Multiobjective Optimization Problem

Many real-world optimization problems inherently involve optimizing multiple non-commensurable and often competing criteria that reflect various design specifications and constraints. For such a multiobjective optimization problem, it is highly improbable that all the conflicting criteria would be optimized by a single design, and hence trade-off among the conflicting design objectives is often inevitable.

The phrase "multiobjective (MO) optimization" is synonymous with "multivector optimization", "multicriteria optimization" or "multiperformance optimization" (Coello Coello 1998). Osyczka (1985) defined multiobjective optimization as a problem of finding:

> "a vector of decision variables which satisfies constraints and optimizes a vector function whose elements represent the objective functions. These functions form a mathematical description of performance criteria which are usually in conflict with each other. Hence, the term 'optimize' means finding such a solution which would give the values of all the objective functions acceptable to the designer."

In mathematical notation, considering the minimization problem, it tends to find a parameter set $\boldsymbol{P}$ for

$$\underset{\boldsymbol{P}\in\Phi}{\text{Min}}\ \boldsymbol{F}(\boldsymbol{P}), P \in \boldsymbol{R}^n, \tag{1.1}$$

where $\boldsymbol{P} = \{p_1, p_2,\ldots, p_n\}$ is a n-dimensional individual vector having n decision variables or parameters while Φ defines a feasible set of $\boldsymbol{P}$. $\boldsymbol{F} = \{f_1, f_2,\ldots, f_m\}$ is an objective vector with m objective components to be minimized, which may be competing and non-commensurable to each other.

The contradiction and possible incommensurability of the objective functions make it impossible to find a single solution that would be optimal for all the objectives simultaneously. For the above multiobjective optimization problem, there exist a family of solutions known as Pareto-optimal set, where each objective component of any solution can only be improved by degrading at least one of its other objective components (Goldberg and Richardson 1987; Horn and Nafpliotis 1993; Srinivas and Deb 1994). Following are some useful terms in multiobjective optimization:

Pareto Dominance

When there is no information for preferences of the objectives, Pareto dominance is an appropriate approach to compare the relative strength between two solutions in MO optimization (Steuer 1986; Fonseca and Fleming 1993). It was initially formulated by Pareto (1896) and constituted by itself the origin of research in multiobjective optimization. Without loss of generality, an objective vector $\boldsymbol{F}_a$ in a minimization problem is said to dominate another objective vector $\boldsymbol{F}_b$, denoted by $\boldsymbol{F}_a \prec \boldsymbol{F}_b$, *iff*

$$f_{a,i} \le f_{b,i} \quad \forall \quad i \in \{1,2,...,\ m\} \text{ and } f_{a,j} < f_{b,j} \quad \exists\, j \in \{1,2,...,\ m\} \tag{1.2}$$

Local Pareto-optimal Set

If no solution in a set ψ dominates any member in a set Ω, where $\Omega \subseteq \psi \subseteq \Phi$, then Ω denotes local Pareto-optimal set. The Ω usually refers to a Pareto-optimal set found in each iteration of the optimization or at the end of optimization in a single run. "Pareto-optimal" solutions are also termed "non-inferior", "admissible", or "efficient" solutions (Van Veldhuizen and Lamont 1999).

Global Pareto-optimal Set

If no solution in the feasible set Φ dominates any member in a set Γ, where $\Gamma \subseteq \Phi$, then Γ denotes the global Pareto-optimal set. It is always true that there is no solution in local Pareto-optimal set Ω dominating any solution in Γ. The Γ usually refers to actual Pareto-optimal set in a MO optimization problem, which can be obtained via the solutions of objective functions concerning the space of Φ or approximated through many repeated optimization runs.

Pareto Front

Given the MO optimization function $\boldsymbol{F}(\boldsymbol{P})$ and Pareto optimal set Ω, Van Veldhuizen and Lamont (2000) defined the Pareto front $\boldsymbol{PF}^*$ as:

$$\boldsymbol{PF}^* = \{\boldsymbol{F}(\boldsymbol{P}) = (f_1(\boldsymbol{P}), f_2(\boldsymbol{P}), \cdots, f_m(\boldsymbol{P})) \mid \boldsymbol{P} \in \Omega\} \tag{1.3}$$

Horn and Nafpliotis (1993) stated that the Pareto front is a (m-1) dimensional surface in a m-objective optimization problem. Van Veldhuizen and Lamont (1999) later

pointed out that the Pareto front of MO optimization with $m = 2$ objectives is at most a (restricted) curve, and is at most a (restricted) (m-1) dimensional surface when $m \geq 3$.

Totally Conflicting, Non-conflicting and Partially Conflicting Objective Functions

The objective functions of a MO optimization problem can be categorized as totally conflicting, non-conflicting or partially conflicting. Given a solution set Φ, a vector of objective functions $\boldsymbol{F} = \{f_1, f_2, \ldots, f_m\}$ is said to be totally-conflicting if there exist no two solutions $\boldsymbol{P}_a$ and $\boldsymbol{P}_b$ in set Φ such that ($\boldsymbol{F}_a \prec \boldsymbol{F}_b$) or ($\boldsymbol{F}_b \prec \boldsymbol{F}_a$). MO problems with totally conflicting objective functions needs no optimization process because the whole solution set in Φ are global Pareto-optimal. On the other hand, the objective functions are said to be non-conflicting if any two selected solutions $\boldsymbol{P}_a$ and $\boldsymbol{P}_b$ in set Φ always satisfy ($\boldsymbol{F}_a \prec \boldsymbol{F}_b$) or ($\boldsymbol{F}_b \prec \boldsymbol{F}_a$). MO problems with non-conflicting objective functions can be easily transformed into single-objective problems by arbitrarily considering one of the objective components throughout the optimization process or combining the objective vector into a scalar function. This is because improving one objective component will always lead to improving the rest of the objective components, and vice versa. The size of global or local Pareto-optimal set is one for this class of MO problems. If a MO optimization problem belongs to neither the first class nor the second, it belongs to the third class of partially conflicting objective functions. Most MO optimization problems belong to the third class, where a family of Pareto-optimal solutions is desired.

Example

Consider the Fonseca and Fleming's two-objective minimization problem (Fonseca and Fleming 1993). The two objective functions, f_1 and f_2, to be minimized are given as:

$$f_1(x_1,...,x_8) = 1 - exp\left(-\sum_{i=1}^{8}\left(x_i - \frac{1}{\sqrt{8}}\right)^2\right) \tag{1.4a}$$

$$f_2(x_1,...,x_8) = 1 - exp\left(-\sum_{i=1}^{8}\left(x_i + \frac{1}{\sqrt{8}}\right)^2\right) \tag{1.4b}$$

where $-2 \le x_i < 2,\ \forall i = 1,2,...,8$. According to (1.4), there are 8 parameters (x_1,…, x_8) to be optimized so that f_1 and f_2 are minimized.

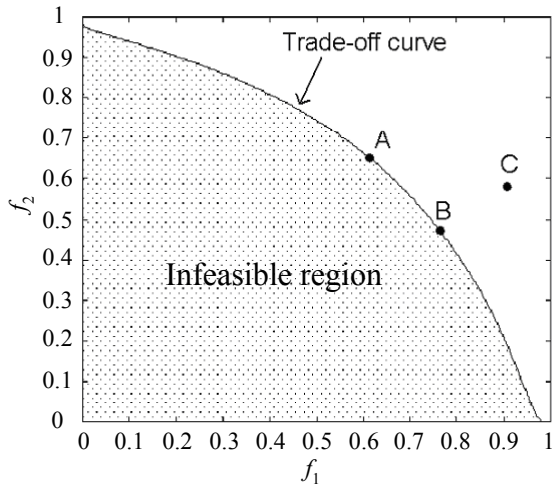


Fig. 1.1. Trade-off curve in the objective domain

The trade-off curve of Eq. (1.4) is shown by the curve in Fig. 1.1, where the shaded region represents the infeasible area in objective domains. One cannot say whether *A* is better than *B* or vice-versa because one solution is better than the other on one-objective and worse in the other. However *C* is worse than *B* because solution *B* is

better than *C* in both of the objective functions. *A*, *B* ... constitute the non-dominated solutions while *C* is a dominated solution.

1.2 Background on Multiobjective Evolutionary Algorithms

Evolutionary algorithms (EAs) are stochastic search methods that simulate the process of evolution, incorporating ideas such as reproduction, mutation and the Darwinian principle of "survival of the fittest". Since the 1970s several evolutionary methodologies have been proposed, including genetic algorithms, evolutionary programming, and evolution strategies. All of these approaches operate on a set of candidate solutions. Although the underlying principles are simple, these algorithms have proven themselves as general, robust and powerful search mechanisms. Unlike traditional gradient-guided search techniques, EAs require no derivative information of the search points, and thus require no stringent conditions on the objective function, such as to be well-behaved or differentiable.

Because the set of solutions are often conflicting in the multiple objective functions, specific compromised decision must be made from the available alternatives. The final solution results from both optimization and decision-making and this process is more formally declared as follows (Hwang and Masud 1979): (1) Priori preference articulation. This method transforms a multiobjective problem into a single objective problem prior to optimization. (2) Progressive preference articulation. Decision and optimization are intertwined where partial preference information is provided upon which optimization occurs. (3) Posteriori preference articulation. A set of efficient candidate solutions is found by some method before decision is made to choose the best solution.

The priori preference articulation transforms a multiobjective problem into a single objective problem, which is different from the original one to be solved. To employ such technique, one must have some knowledge of the problem in hand. Moreover, the optimization process is often sensitive to the importance factors of objectives.

Single objective optimization algorithms provide in the ideal case only one Pareto-optimal solution in one optimization run. A representative convex part of the Pareto front can be sampled by running a single objective optimization algorithm each time with a different vector of importance factors (Lahanas et al. 2003). However, many runs are burdensome in computation effort and are not efficient to find good approximation to the Pareto front. Moreover there is a great drawback that the single-objective optimization cannot reach the non-convex parts of the Pareto front. For two objectives, the weighted sum is given by $y = w_1 f_1(x) + w_2 f_2(x)$, i.e. $f_2(x) = -(w_1 / w_2) f_1(x) + y / w_2$ (Lahanas et al. 2003). The minimization of the weighted sum can be interpreted as finding the value of y for which the line with slope $-w_1 / w_2$ just touches the Pareto front as it proceeds outwards from the origin. It is therefore not possible to obtain solutions on non-convex parts of the Pareto front with this approach.

Making use of multiobjective evolutionary algorithms in the posteriori preference articulation is currently gaining significant attentions from researchers in various fields as more and more researchers discover the advantages of their adaptive search to find a set of trade-off solutions. Corne et al. (2003) argued that “single-objective approaches are almost invariably unwise simplifications of the real-problem”, “fast and effective techniques are now available, capable of finding a well-distributed set of diverse trade-

off solutions, with little or no more effort than sophisticated single-objective optimizers would have taken to find a single one", and "the resulting diversity of ideas available via a multiobjective approach gives the problem solver a better view of the space of possible solutions, and consequently a better final solution to the problem at hand" .

Indeed, the objective function in EAs is permitted to return a vector value, not just a scalar value and evolutionary algorithms have the ability of capturing multiple solutions in a single run (Corne et al. 2003). These reasons make evolutionary algorithms suitable for multiobjective optimization. Pareto-based multiobjective evolutionary algorithms have the highest growth rate compared to other multiobjective evolutionary algorithms since Goldberg and Richardson first proposed them in 1987 and it is believed that this trend will continue in the near future. This growing interest can be reflected by the significantly increasing number of different evolutionary-based approaches and the variations of existing techniques published in technical literatures. As a consequence, there have been many survey studies on evolutionary techniques for MO optimization (Fonseca and Fleming 1995a; Coello Coello 1996; Bentley and Wakefield 1997; Horn 1997; Coello Coello 1998; Van Veldhuizen and Lamont 2000, Tan et al. 2002a).

Deb (2001) pointed out two important issues in MO optimization: (1) to find a set of solutions as close as possible to the true Pareto front; (2) to find a set of solutions as diverse as possible. As pointed by Zitzler and Thiele (1999), to maximize the spread of the obtained front, i.e. for each objective a wide range should be covered, is also an important issue in multiobjective optimization.

1.3 Thesis Outline

This thesis tries to develop advanced and reliable evolutionary techniques for MO optimization. It introduces a cooperative coevolution mechanism into MO optimization and develops two new features for multiobjective evolutionary algorithms. The thesis consists of five chapters.

Chapter 2 presents a framework of multiobjective evolutionary algorithms, discusses the key concepts of evolutionary multiobjective optimization in decision-making, and gives a brief overview of some well-known MOEA implementations.

Chapter 3 presents a cooperative coevolutionary algorithm (CCEA) for multiobjective optimization. Exploiting the inherent parallelism in cooperative co-evolution, a distributed CCEA (DCCEA) is developed to formulate the algorithm into a computing structure suitable for parallel processing where computers over the network share the computational workload.

In Chapter 4, two features are proposed to enhance the ability of multiobjective evolutionary algorithms. The first feature is the adaptive mutation operator that adapts the mutation rate to maintain a balance between the introduction of diversity and local fine-tuning. The second feature is the enhanced exploration strategy that encourages the exploration towards less populated areas and hence distributes the generated solutions evenly along the discovered Pareto front.

Chapter 5 concludes the whole thesis and points out the direction of future research.

Chapter 2

Multiobjective Evolutionary Algorithms

2.1 Conceptual Framework

Many evolutionary techniques for MO optimization have been proposed and implemented in different ways. VEGA (Schaffer 1985), MOGA (Fonseca and Fleming 1993), HLGA (Hajela and Lin 1992), NPGA (Horn and Nafpliotis 1993), IMOEA (Tan et al. 2001) and NSGA-II (Deb et al. 2002a) work on single population. SPEA (Zitzler and Thiele 1999), SPEA2 (Zitzler et al. 2001), PAES (Knowles and Corne 2000) and PESA (Corne et al. 2000) use an external population/memory to preserve the best individuals found so far besides the main evolved population. Although each MO evolutionary technique may have its own specific features, most MO evolutionary techniques exhibit common characteristics and can be represented in a framework as shown in Fig. 2.1.

MOEAs originated from SOEAs (Goldberg 1989a) in the sense that both techniques involve the iterative updating/evolving of a set of individuals until a predefined optimization goal/stopping criterion is met. At each generation, individual assessment, genetic selection and evolution (e.g. crossover and mutation), are performed to transform the population from current generation to the next generation with the aim to improve the adaptability of the population in the given test environment. In some

evolutionary approaches, the elitism is also applied to avoid losing the best-found individuals in the mating pool to speed up the convergence. Generally speaking, MOEAs differ from SOEAs mainly in the process of individual assessment and elitism/archiving. The individual assessment and elitism will be further discussed in the following subsections.

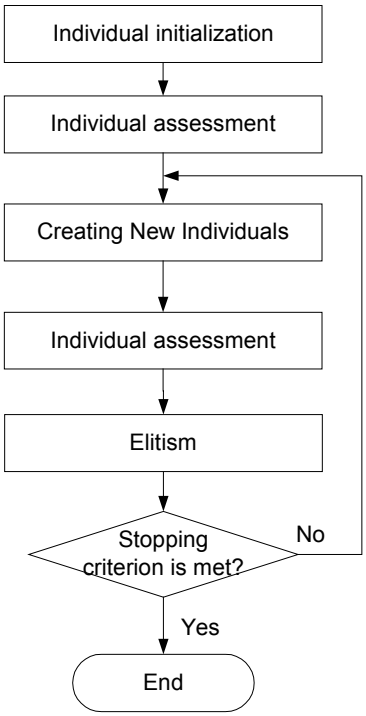


Fig. 2.1. The framework of multiobjective evolutionary algorithms

2.2 Individual Assessment for Multiobjective Optimization

In MO optimization, the individuals should be pushed toward the global Pareto front as well as be distributed uniformly along the global Pareto front. Therefore the individual assessment in MOEA should simultaneously exert a pressure (denoted as $\bar{P}_n$ in Fig. 2.2) to promote the individuals in a direction normal to the trade-off region and a pressure (denoted as $\bar{P}_t$ in Fig. 2.2) tangentially to that region. These two pressures,

which are normally orthogonal to each other, give the unified pressure (denoted as $\bar{P}_u$ in Fig. 2.2) and direct the evolutionary search in the MO optimization context.

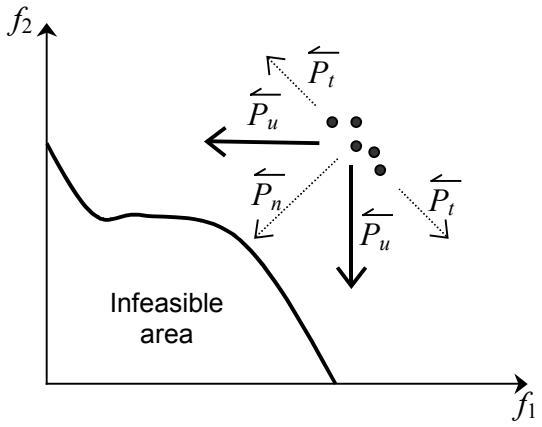


Fig. 2.2. The improvement pressures from multiobjective evaluations

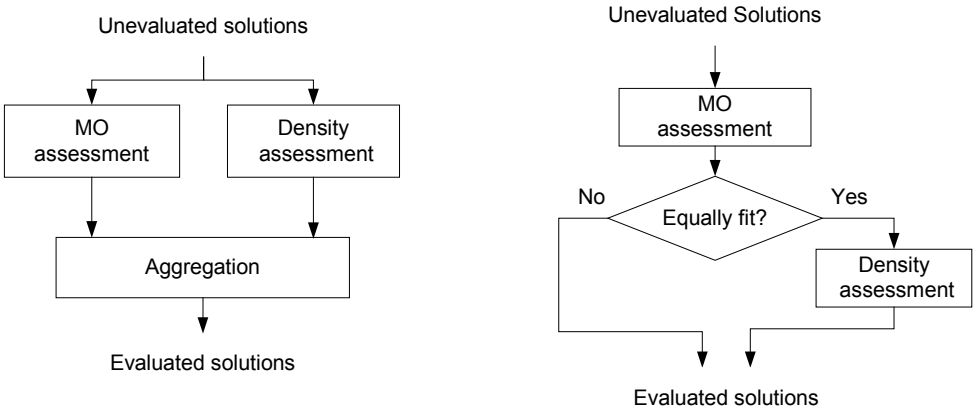
Some MOEAs, such as MIMOGA (Murata and Ishibuchi 1995), MSGA (Lis and Eiben 1997) and VEGA (Schaffer 1985), implement $\bar{P}_u$ through a single-step approach in the assessment. For example, MIMOGA applies the random assignment of weights on each individual to exert $\bar{P}_u$, where weights are not constant for each individual. However this simple technique do not have good control on the direction of the exerted $\bar{P}_u$. For other MOEAs, the $\bar{P}_n$ and $\bar{P}_t$ are implemented explicitly in different operational elements.

Pareto dominance is a widely used MO assessment technique to exert $\bar{P}_n$. It has shown its effectiveness in attainting the tradeoffs (Goldberg and Richardson 1987; Fonseca and Fleming 1993; Horn and Nafpliotis 1993; Srinivas and Deb 1994). However it is weak in diversifying the population along the tradeoff surface, which has been shown

in (Fonseca 1995b) that the individuals will converge to arbitrary portions of the discovered trade-off surface, instead of covering the whole surface. Thus the MO assessment alone is insufficient to maintain the population distribution because it does not induce $\bar{P}_t$ for tangential effect in the evolution. To address this issue, a density assessment has to be added to induce sufficient $\bar{P}_t$. The general working principle of density assessment is to assess the distribution density of solutions in the feature space and then made decision to balance up the distribution density among the sub-divisions of feature space. As MO assessment, density assessment is also considered as a fundamental element in MOEAs, which maintains individual diversity along the trade-off surface.

Many methods for individual assessment have been proposed and integrated into various MOEAs in different ways. They can be categorized into the aggregated approach and comparative approach. As shown in Fig. 2.3, the two approaches are different in the hybridization of MO and density assessment to generate the unified pressure $\bar{P}_u$. In the aggregated approach, the results from the MO and density assessment are aggregated for the individual assessment decision. The aggregation function applied can be either linear, as implemented in non-generational GA (Valenzuela-Rendón and Uresti-Charre 1997), or non-linear, as in MOGA (Fonseca and Fleming 1993) and non-generational GA (Borges and Barbosa 2000). In this case, the effect of $\bar{P}_n$ and $\bar{P}_t$ on the resulting $\bar{P}_u$ is mainly based on the aggregation function used. Thus the aggregation function must be carefully constructed so as to keep the balance between $\bar{P}_n$ and $\bar{P}_t$.

In the comparative approach, only the individuals that are equally fit in MO assessment will be further compared through the density assessment. This approach assigns a higher priority level to MO assessment than density assessment. At the initial stage of the evolution, the effect of $\bar{P}_n$ is larger than that of $\bar{P}_u$ because the candidate individuals are comparable via MO assessment when the opportunity to move closer to the global trade-offs is high. When the population begins to converge to the discovered trade-offs, most individuals are equally fit in MO assessment and the density assessment will exert the major effect to disperse the individuals. Some of the existing MO evolutionary techniques adopting the comparative approaches are (Horn and Nafpliotis 1993; Srinivas and Deb 1994; Deb et al. 2002a; Knowles and Corne 2000; Khor et al. 2001).



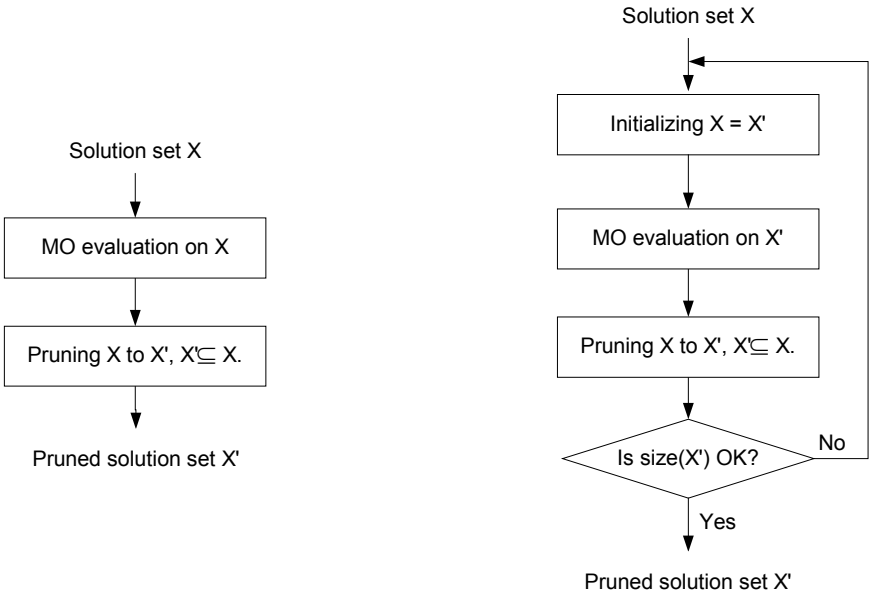
(a) Aggregated approach (b) Comparative approach

Fig. 2.3. Generalized multiobjective evaluation techniques

2.3. Elitism

The basic idea of elitism in MOEAs is to keep record of a family of the best-found non-dominated individuals (elitist individuals) that can be assessed later in the MO evolution process. Among the existing literatures that have reported the successful

work of elitism for evolutionary MO techniques are (Zitzler and Thiele 1999; Tan et al. 2001; Deb et al. 2002a; Coello Coello and Pulido 2001; Khor et al. 2001). For the sake of limited computing and memory resources in implementation, the set of elitist individuals often has a fixed size and pruning process is needed when the size of the elitist individuals exceeds the limit. Fig. 2.4 gives two different implementations of pruning process, batch and recurrence mode.



(a) Batch mode (b) Recurrence mode

Fig. 2.4. Two modes of pruning process for MO elitism

Let X denote an individual set consisting of the current elitist individuals and the promising individuals from the genetic evolution, which exceeds the allowable size (size(X')) of elitist individuals X'. In the batch mode of pruning process, all individuals from X are undergone the assessment and the results are applied to prune X to X'. While in the recurrence mode, a group of the least promising individuals is removed from a given population X to complete a cycle. This cycling process repeats to further

remove another set of the least promising individuals from the remaining individuals until a desired size is achieved.

The recurrence-mode of pruning process is likely to avoid the extinction of local individuals, which somehow leads to the discontinuity of the discovered Pareto front. But it often requires more computational effort compared to the batch-mode pruning process due to the fact that the individual assessment in recurrence mode has to be performed on the remaining individuals in each cycle of pruning.

After the elitism, the elitist set X' can be either stored externally, which is often identified as the second/external population (Zitzler and Thiele 1999; Borges and Barbosa 2000; Knowles and Corne 2000; Coello Coello and Pulido 2001), or given a surviving probability of one in the next generation. If the former case is employed, the elitist set X' can optionally take part in the mating process to increase the convergence rate. However, it should be carefully implemented to avoid too much influence from the elitist set in the mating, which may subsequently lead to pre-mature convergence.

2.4. Density Assessment

Density assessments in MOEAs encourage the divergence in the tangential direction of the currently found trade-off surface by giving high selection probability in the less crowded region. The density assessment techniques reported along the development of evolutionary techniques for multiobjective optimization include Sharing (Goldberg 1989a), Grid Mapping (Knowles and Corne 2000; Coello Coello and Pulido 2001), Density Estimation (Zitzler et al. 2001) and Crowding (Deb et al. 2002a).

i) Sharing

Sharing was originally proposed by Goldberg (1989a) to promote the population distribution and prevent genetic drift as well as to search for possible multiple peaks in single objective optimization. Fonseca and Fleming (1993) later employed it in multiobjective optimization. Sharing is achieved through a sharing function. Let *d* be the Euclidean distance between individuals *x* and *y*. The neighborhood size is defined in term of *d* and specified by the so-called niche radius σ_{share}. The sharing function is defined as follows:

$$sh(d) = \begin{cases} 1-(d/\sigma_{share})^{\alpha} & \text{if d}<\sigma_{share} \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

And the niche count function is defined with the help of sharing function:

$$nc(x) = \sum_{y} sh(dist(x,y)) \tag{2.2}$$

The niche radius σ_{share} is a key parameter in sharing.

ii) Grid Mapping

To keep track of the degree of crowding in different regions of the space, an *m*-dimensional grid is used to partition the feature space, where *m* is the dimensions of the objective space. When each individual is generated, its grid location is found and a map of the grid is maintained to indicate for each grid location how many and which individuals in the population reside there. To maintain the uniformity of the distribution, individuals with higher grid-location count should be given less sampling probability than those with lower grid-location count in the selection process. This approach has been proposed and applied in at least Pareto Archived Evolutionary strategy (PAES) (Knowles and Corne 2000), Pareto Envelope Based Selection

Algorithm (Corne et al. 2000) and Micro-Genetic Algorithm (Coello Coello and Pulido 2001).

iii) Crowding

Crowding was proposed by Deb et al. (2002a) in their Non-dominated Sorting Genetic Algorithm II (NSGA-II). The crowding distance is an estimate of the size of the largest cube enclosing a single solution without any other point in the population and indicates the density of solutions surrounding a particular individual. This measure is defined as the average distance of two points on either side of the selected solution along each of the objectives. During the selection process, the crowding distance will be used to break a tie between two solutions with the same rank.

iv) Density Estimation

Density estimation was proposed in the strength Pareto evolutionary algorithm 2 (SPEA2) (Zitzler et al. 2001). It is adapted from $k \cdot th$ nearest neighbor method and it is given by the inverse of the distance to the $k \cdot th$ nearest neighbor. The density estimation is used both in the selection and in the archive truncation process.

2.5 Overview of Some Existing MOEAs

Five well-known algorithms are selected for the comparison studies in following chapters. These algorithms have been applied or taken as references in many literatures.

2.5.1 Pareto Archived Evolution Strategy

The Pareto archived evolution strategy (PAES) (Knowles and Corne 2000) is unique from other MOEAs in that it is a non-population based local search algorithm.

However, PAES does maintain an archive to preserve non-dominated solution and utilizes the archive information in the selection process. PAES uses only the mutation operator to implement a hill climbing strategy. The grid mapping is applied to keep track of the degree of crowding. The algorithm flow of PAES is shown in Fig. 2.5.

Arc_size (Archive size)

genNum (Maximum number of generation)

Step1: Set $n = 0$

Step2: *Initialization:* Generate single initial solution $C(n)$, empty the archive *Arc*.

Step3: *Evaluation:* Evaluate the current solution $C(n)$.

Step4: *Updating archive:* Add the current solution $C(n)$ into the archive *Arc* if it is non-dominated. If the size of *Arc* is more than Arc_size, grid mapping is employed for archive truncation.

Step5: *Mutation:* Mutate the current solution $C(n)$ to create a new potential solution $M(n)$.

Step6: *Evaluation:* Evaluate the potential solution $M(n)$.

Step7: If $M(n)$ dominates $C(n)$, $C(n+1) = M(n)$. Else $C(n+1) = C(n)$.

Step8: *Termination:* $n = n + 1$. If $n = genNum$, stop. Else if $M(n-1)$ dominates $C(n-1)$, go to Step 4. Else go to Step 5.

Fig. 2.5. Algorithm flowchart of PAES

2.5.2 Pareto Envelope Based Selection Algorithm

The Pareto envelope based selection algorithm (PESA) (Corne et al. 2000) draws its motivation from the strength Pareto evolutionary algorithm (SPEA) (Zitzler and Thiele, 1999) and PAES. It uses an external population to store the current approximate Pareto front and an internal population to evolve new candidate solutions. PESA uses the grid

mapping to perform online tracking of the degree of crowding in different regions of the archive. Tournament selection in PESA is based on the grid-location count to guide the search towards the less populated areas. The algorithm flow of PESA is shown in Fig. 2.6.

Pop_size (Internal population size)

Arc_size (Archive size)

genNum (Maximum number of generation)

Step1: Set $n = 0$

Step2: *Initialization:* Generate an initial internal population *Pop*(n) and empty the archive *Arc*.

Step3: *Evaluation:* Evaluate the individuals in the internal population *Pop*(n).

Step4: *Updating archive:* Copy all non-dominated individuals in *Pop*(n) into the archive *Arc*. If the size of *Arc* is more than Arc_size, grid mapping is employed for archive truncation.

Step5: Empty the internal population *Pop*(n+1) = $\varnothing$.

Step6: *Crossover:* With p_c, select two parents from archive the *Arc* and crossover them to create a child. Add this child to the internal population *Pop*(n+1).

Step7: *Mutation:* With $1 - p_c$, select one parent from the archive *Arc* and mutate it to create a child. Add this child to the internal population *Pop*(n+1).

Step8: Go to Step 6 until the internal population *Pop*(n+1) is full.

Step9: *Evaluation:* Evaluate the individuals in the internal population *Pop*(n+1).

Step10: *Termination:* $n = n + 1$. If $n = genNum$, stop. Else go to Step 4.

Fig. 2.6. Algorithm flowchart of PESA

2.5.3 Non-dominated Sorting Genetic Algorithm II

The non-dominated sorting genetic algorithm II (NSGA II) (Deb et al. 2002a) is the improved version of its predecessor NSGA (Srinivas and Deb 1994). It employs a fast non-dominated approach to assign rank to individuals and a crowding distance assignment to estimate the crowding. In case of a tie in rank during the selection process, the individual with a smaller crowding distance wins. Together with an elitism scheme, the NSGA II claims to produce better results than NSGA. The algorithm flow of NSGAII is shown in Fig. 2.7.

Pop_size (Parent population size)

Chd_size (Child population size)

genNum (Maximum number of generation)

Step1: Set $n = 0$.

Step2: *Initialization:* Generate an initial parent population $Pop(n)$ and empty the child population $Chd(n)$.

Step3: *Evaluation:* Evaluate the initial parent population $Pop(n)$.

Step4: *Mating selection:* Select individuals from $Pop(n)$ to create the mating pool.

Step5: *Variation:* Apply the crossover and mutation operators to the mating pool to create the child population $Chd(n)$.

Step6: *Evaluation:* Evaluate the child population $Chd(n)$.

Step7: *Elitism selection:* Combine the parent and child population. Sort this combined population $Pop(n) \cup Chd(n)$ according to Pareto dominance and assign crowding distance for $Pop(n) \cup Chd(n)$. Finally Pop_size solutions are selected from $Pop(n) \cup Chd(n)$ based on the

crowded comparison operator and copied into the next population $Pop(n+1)$.

Step8: *Termination:* $n = n + 1$. If $n = genNum$, stop. Else go to Step 4.

Fig. 2.7. Algorithm flowchart of NSGA II

2.5.4 Strength Pareto Evolutionary Algorithm 2

The strength Pareto evolutionary algorithm 2 (SPEA 2) (Zitzler et al, 2001) is the improved version of its predecessor SPEA. In SPEA 2, both archive and population are assigned fitness based on strength and density estimation. The strength of an individual is defined as the number of individuals that dominates it. The density estimation mechanism has been described in Section 2.4. A truncation method based on the density estimation is employed to keep the fixed size of archive. The elitism is implemented using an internal and an external population. All The algorithm flow of SPEA 2 is shown in Fig. 2.8.

Pop_size (Internal population size)

Arc_size (Archive population size)

$genNum$ (Maximum number of generation)

Step1: Set $n = 0$

Step2: *Initialization:* Generate an initial internal population $Pop(n)$ and empty the archive $Arc(n) = \varnothing$.

Step3: *Evaluation:* Evaluate the individuals in $Pop(n)$.

Step4: *Environmental selection:* Copy the non-dominated solutions in the $Pop(n)$ and $Arc(n)$ to the new archive $Arc(n+1)$. If the size of $Arc(n+1)$ exceeds Arc_size, then truncation is performed based on density

estimation. If the size of $Arc(n+1)$ is less than Arc_size, the $Arc(n+1)$ is filled with the best dominated solutions in $Arc(n)$.

Step5: *Mating selection:* Select individuals from $Arc(n+1)$ to create the mating pool.

Step6: *Variation:* Apply the crossover and mutation operators to the mating pool to create new population $Pop(n+1)$.

Step7: *Termination:* $n = n + 1$. If $n = genNum$, stop. Else go to Step3.

Fig. 2.8. Algorithm flowchart of SPEA 2

2.5.5 Incrementing Multiobjective Evolutionary Algorithm

The incrementing multiobjective evolutionary algorithm (IMOEA) (Tan et al. 2001) is an MOEA with dynamic population size that is computed online according to the discovered approximate Pareto front and desired population density. It employs the method of fuzzy boundary local perturbation with interactive local fine-tuning to achieve broad neighbourhood exploration and create the desired number of individuals. Elitism is implemented in the form of the switching preserved strategy. The algorithm flow is shown in Fig. 2.9.

dps (Dynamic population size)

Arc_size (Archive population size)

genNum (Maximum number of generation)

Step1: Set $n = 0$.

Step2: *Initialization:* Generate an initial population $pop(n)$.

Step3: *Evaluation:* Evaluate the individuals in the population $pop(n)$.

Step4: Calculate the dynamic population size $dps(n)$, number of perturbations

np(*n*) and number of tournament selected individuals *nsi*(*n*).

Step5: *Mating selection:* Tournament select *nsi*(*n*) individuals from *pop*(*n*) according to their niche cost to create *selpop*(*n*).

Step6: *Crossover:* Perform crossover with crossover probability P_c on *selpop*(*n*) to create *crosspop*(*n*)..

Step7: *Mutation:* Perform *FBLP* with *np*(*n*) perturbations for each individuals in *crosspop*(*n*) to create *evolpop*(*n*).

Step8: *Switching preservation: pop*(*n*+1) = *pop*(*n*)$\bigcup$ *evolpop*(*n*). If the number of non-dominated solution in *pop*(*n*+1) is less than *dps*(*n*), truncate *pop*(*n*+1) based on Pareto dominance. Else truncate *pop*(*n*+1) based on niche cost.

Step9: *Termination:* $n = n + 1$. If $n = genNum$, stop. Else go to Step3.

Fig. 2.9. Algorithm flowchart of IMOEA

Chapter 3

Cooperative Coevolution for Multiobjective Optimization

3.1 Introduction

Although the MOEAs are capable of approximating the optimal Pareto front in multiobjective optimization with varying success (Knowles and Corne 2000; Corne et al. 2000; Deb et al. 2002a; Zitzler et al. 2001; Tan et al. 2001), the computational cost involved in terms of time and hardware for evolving the complete set of trade-off solutions often become insurmountable as the size or complexity of the problem increases. Meanwhile, studies have shown that coevolutionary mechanism can increase the efficiency of the optimization process significantly (Potter and De Jong 1994, 2000; Moriarty 1997; Liu et al. 2001). Therefore, one promising approach to overcome the limitation in MOEAs is to incorporate the coevolutionary mechanism by co-evolving the solution set with a number of subpopulations in a cooperative way.

Neef et al. (1999) introduced the concept of coevolutionary sharing and niching into multiobjective genetic algorithms, which adapted the niche radius through competitive coevolution. Parmee et al. (1999) used multiple populations where each population

optimized one objective related to the problem. The individual fitness in each population was adjusted by comparing the variable values of identified solutions related to a single objective with solutions of other populations. Lohn et al. (2002) embodied the model of competitive coevolution in multiobjective optimization, which contained the population of candidate solutions and the target population consisting of target objective vectors. Keerativuttiumrong et al. (2002) extended the cooperative coevolutionary genetic algorithm (Potter and De Jong 1994, 2000) to MO optimization by evolving each species with a multiobjective genetic algorithm (Fonseca and Fleming 1993) in a rather elementary way.

This chapter presents a cooperative coevolutionary algorithm (CCEA) to evolve multiple solutions in the form of cooperative subpopulations for MO optimization. Incorporated with various features like archiving, dynamic sharing and extending operator, the CCEA is capable of maintaining search diversity in the evolution and distributing the solutions uniformly along the Pareto front. Exploiting the inherent parallelism in cooperative coevolution, the CCEA is formulated into a computing structure suitable for concurrent processing that allows inter-communications among subpopulations residing in multiple computers over the Internet. This distributed CCEA (DCCEA) DCCEA can reduce the runtime effectively without sacrificing the performance of CCEA.

The remainder of this chapter is organized as follows: Section 3.2 describes the principle of the proposed CCEA for multiobjective optimization. Section 3.3 presents a distributed version of CCEA and its implementation that uses resources of networked computers. Section 3.4 examines the different features of CCEA and provides a

comprehensive comparison of CCEA with other well-known MOEAs. The performance improvement of the distributed CCEA running on multiple networked computers is also shown in Section 3.4. Conclusions are drawn in Section 3.5.

3.2 Cooperative Coevolution for Multiobjective Optimization

3.2.1 Coevolution Mechanism

Recent advances in evolutionary algorithms show that the introduction of ecological models and the use of coevolutionary architectures are effective ways to broaden the use of traditional evolutionary algorithms (Rosin and Belew 1997; Potter and De Jong 2000). Coevolution can be classified into competitive coevolution and cooperative coevolution. While competitive coevolution tries to get individuals that are more competitive through evolution, the goal of cooperative coevolution is to find individuals from which better systems can be constructed. Many studies (Angeline and Pollack 1993; Rosin and Belew 1997) show that competitive coevolution leads to an "arms race" where two populations reciprocally drive one another to increase levels of performance and complexity. The model of competitive coevolution is often compared to predator-prey or host-parasite interactions, where preys (or hosts) implement the potential solutions to the optimization problem while the predators (or parasites) implement individual "fitness-cases". In a competitive coevolutionary algorithm, the fitness of an individual is based on direct competition with individuals of other species that evolve separately in their own populations. Increased fitness of one of the species implies a diminution in the fitness of the other species. This evolutionary pressure tends to produce new strategies in the populations involved to maintain their chances of survival.

The basic idea of cooperative coevolution is to divide-and-conquer (Potter and De Jong 2000): divide a large system into many modules, evolve the modules separately, and then combine them together again to form the whole system. The cooperative coevolutionary algorithms involve a number of independently evolving species that together form complex structures for solving difficult problems. The fitness of an individual depends on its ability to collaborate with individuals from other species. In this way, the evolutionary pressure stemming from the difficulty of the problem favors the development of cooperative strategies and individuals. Potter and De Jong (1994) presented a cooperative coevolutionary genetic algorithm that improved the performance of GAs on many benchmark functions significantly. It could lead to faster convergence as compared to conventional GAs for low-level to moderate-level of variable interdependencies. This approach was discussed in more details by Potter and De Jong (2000) and applied successfully to string matching task and neural network designs.

Moriarty (1997) used a cooperative coevolutionary approach to evolve neural networks where each individual in one species corresponds to a single hidden neuron of a neural network and its connections with the input and output layers. This population coevolved alongside a second one whose individuals encode sets of hidden neurons (i.e., individuals from the first population) forming a neural network. Liu et al. (2001) used cooperative coevolution to speed up convergence rates of fast evolutionary programming on large-scale problems whose dimension ranged from 100 to 1000. This cooperative coevolutionary approach performed as good as (and sometimes better than) single population evolutionary algorithms, required less computation than single-

population evolution as the populations involved are smaller, and converged faster in term of number of generations.

3.2.2 Adaptation of Cooperative Coevolution for Multiobjective Optimization

3.2.2.1 Cooperative Cooperation and Rank Assignment

Given a single objective optimization problem with n parameters, each parameter is assigned a subpopulation, and these n subpopulations coevolve the individuals in each of them (Potter and De Jong 1994, 2000; Liu et al. 2001). The proposed CCEA for MO optimization adopts the idea of assigning one subpopulation to each parameter and applies this idea to MO optimization where multiple non-dominated solutions are targeted. Fig. 3.1 depicts the principle of cooperation and rank assignment in CCEA, which shows that individuals in subpopulation i cooperate with representatives of other subpopulations to form the complete solutions.

Each subpopulation only optimizes one parameter and an individual in a subpopulation is just a component of a complete solution. Here, the best r individuals in a subpopulation are defined as the representative set of the subpopulation. To evaluate an individual in a subpopulation, a representative is randomly selected from the representative set of every other subpopulation and these representatives are combined with the individual under evaluation to form a complete solution. Then this complete solution is mapped into an objective vector by the objective functions. The objective vector can be used to evaluate how well the selected individual cooperates with other subpopulations to produce good solutions. Since the objective vector cannot be used as fitness in the selection of evolutionary algorithms directly, a Pareto based rank

assignment scheme is applied to give each individual a scalar rank value. The rank of an individual partially reflects the distance between the objective vector of this individual and the current Pareto front.

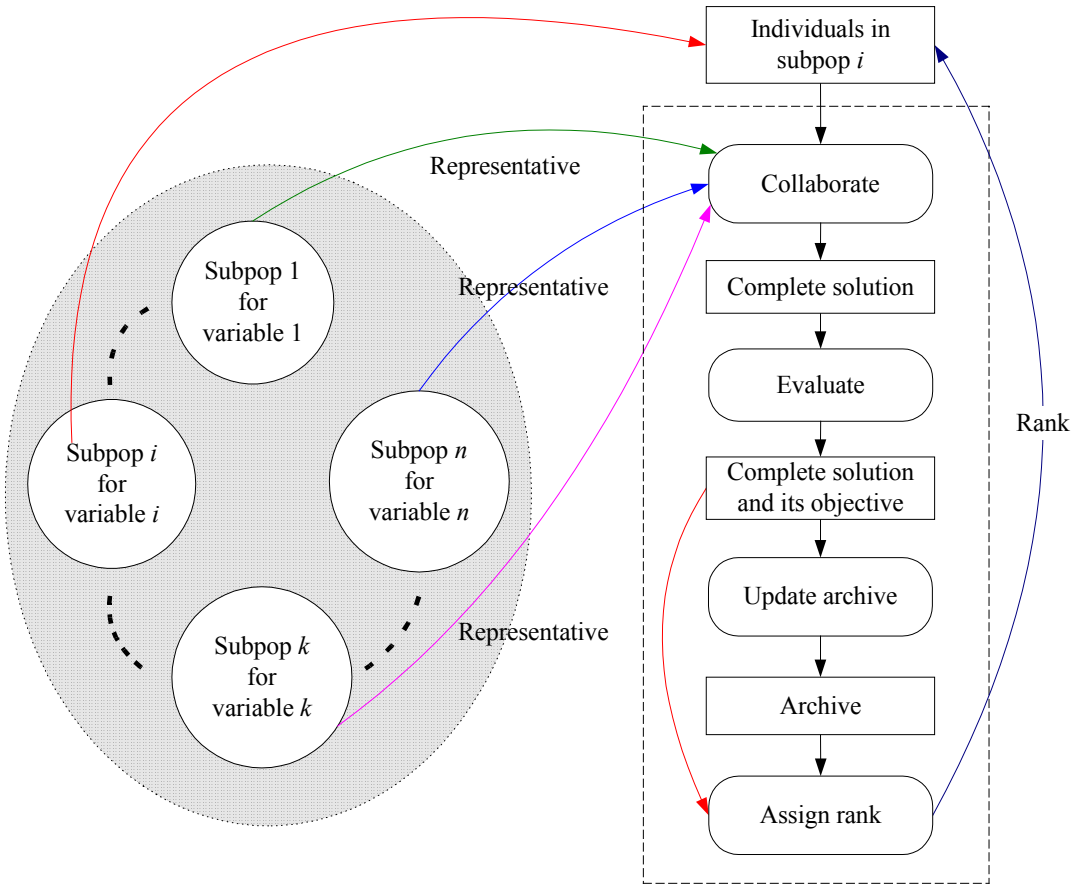


Fig. 3.1. Cooperation and rank assignment in CCEA

By incorporating an archive into the algorithm to store the set of non-dominated solutions, multiple solutions for MO optimization can be achieved by CCEA. This archive is updated in every generation and outputted as the optimal solution set when the evolutionary process is accomplished. Preserving the best solutions found so far, the archive works as an elitism mechanism, which not only results in continuous improvement for the quality of the archive but also ensures the convergence of CCEA.

Moreover, the archive is served as the comparison set in the rank assignment of individuals from subpopulations after these individuals obtain their objective vectors through collaboration. A canonical Pareto ranking scheme (Fonseca and Fleming 1995b) is applied in CCEA, which ranks individuals according to how many members in the archive dominating them.

3.2.2.2 Archive Updating

The archive size is given by a predefined number *archive_size*, which can be adjusted according to the required number of solutions. As illustrated in Fig. 3.2, once a complete solution is evaluated using the objective functions, it will update the archive according to its objective vector. If the solution is not dominated by any archive member, it will be added to the archive and the archive members dominated by it will be discarded. When the maximum archive size is reached, a truncation method based on niche count will be activated to replace the most crowded archive member with the new non-dominated solution in order to maintain the diversity of the archive. To distribute the non-dominated solutions evenly along the Pareto front, the dynamic sharing approach proposed by Tan et al. (2003b) is implemented in CCEA. While used in the archive updating, niche count is also involved in the tournament selection to generate the mating pool in CCEA. A partial order is applied to compare two individuals in the tournament selection: For two individuals i and j, $i \geq_n j$, if (i dominates j), or ($rank(i) < rank(j)$), or $\{rank(i) == rank(j)$ and $nc(i) < nc(j)\}$.

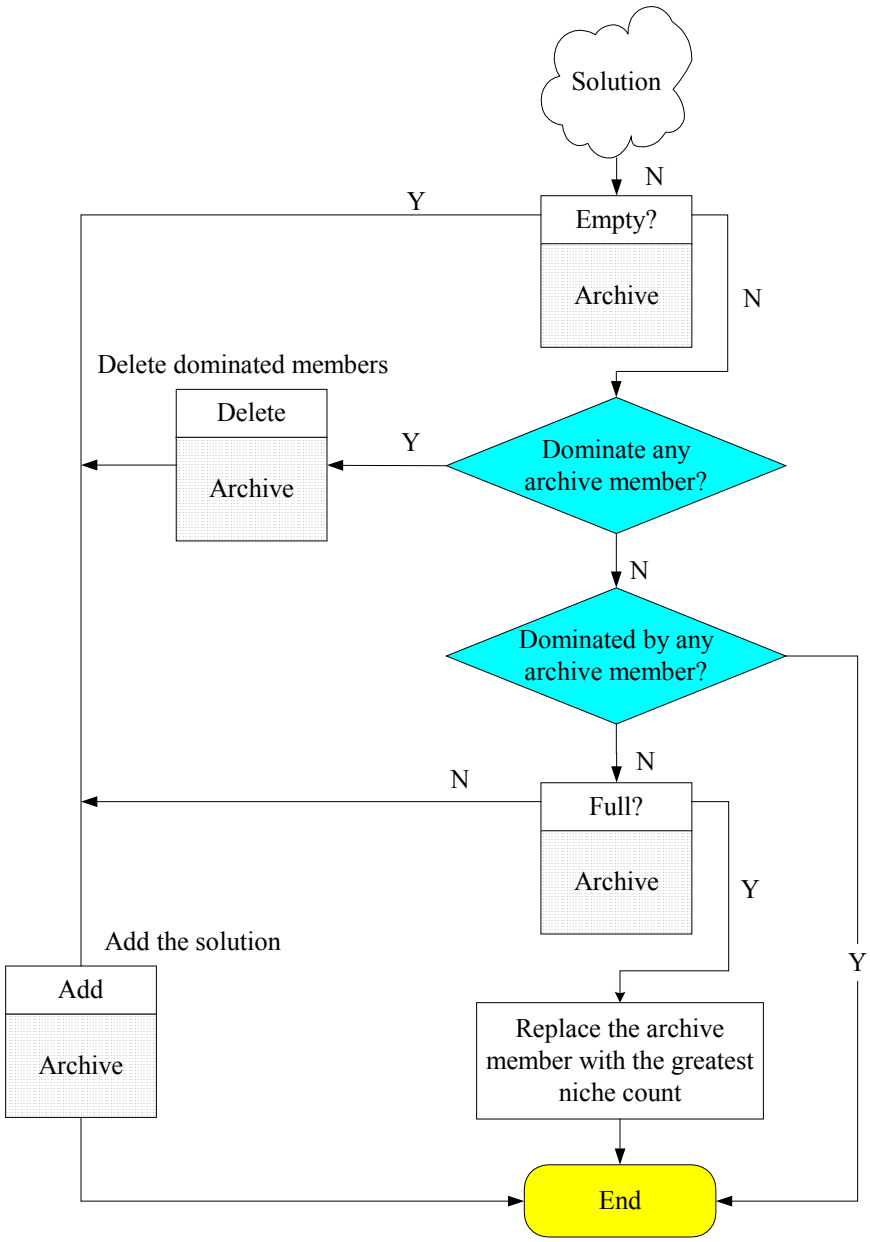


Fig. 3.2. The process of archive updating

3.2.3 Extending Operator

A new feature of extending operator is proposed in CCEA to improve the smoothness and spread of non-dominated solutions. Ordinarily, the under-populated regions are the gaps or boundaries of the archive, which should be given more attention if the Pareto front should be covered by the archive as much as possible. To make these unobvious regions outstanding, the role of extending operator in CCEA is to guide the

evolutionary search into these areas. The archiving scheme plays a critical role in the realization of the extending operation. Firstly, since complete solutions are all stored in the archive, the subpopulations have no pressure to keep the diversity of their own individuals so that they can adaptively focus their search in the regions that are not explored thoroughly. Secondly, by extracting the information of the solution distribution from the archive, archive members in the most under-populated regions will be found and copied to the subpopulations. Hence, these members will have a higher chance to be selected into the mating pool. Detailed description of the extending operator is as follows:

The Extending Operator for CCEA:

Let n be the number of clones.

Step (1) If the archive is not full, exit.

Step (2) Calculate the niche count of each member in the archive. Then find the member with the smallest niche count. This member resides in the most under-populated region.

Step (3) Clone n copies of this archive member to the subpopulations. Here, each part of this member is cloned into its corresponding subpopulation.

In the initial stage of CCEA, the algorithm should concentrate on the search of non-dominated solutions to fill up the archive and achieve a good approximation of the Pareto front. Moreover, a small number of members in the archive are not sufficient to approximate the Pareto front well and give accurate information of solution distribution. Therefore, the extending operator will be activated only when the archive is full. The solution with the smallest niche count is then selected and cloned to the

subpopulations. Such an operation forces the algorithm to pay more attention to the under-populated regions, as desired.

3.2.4 Panorama of CCEA

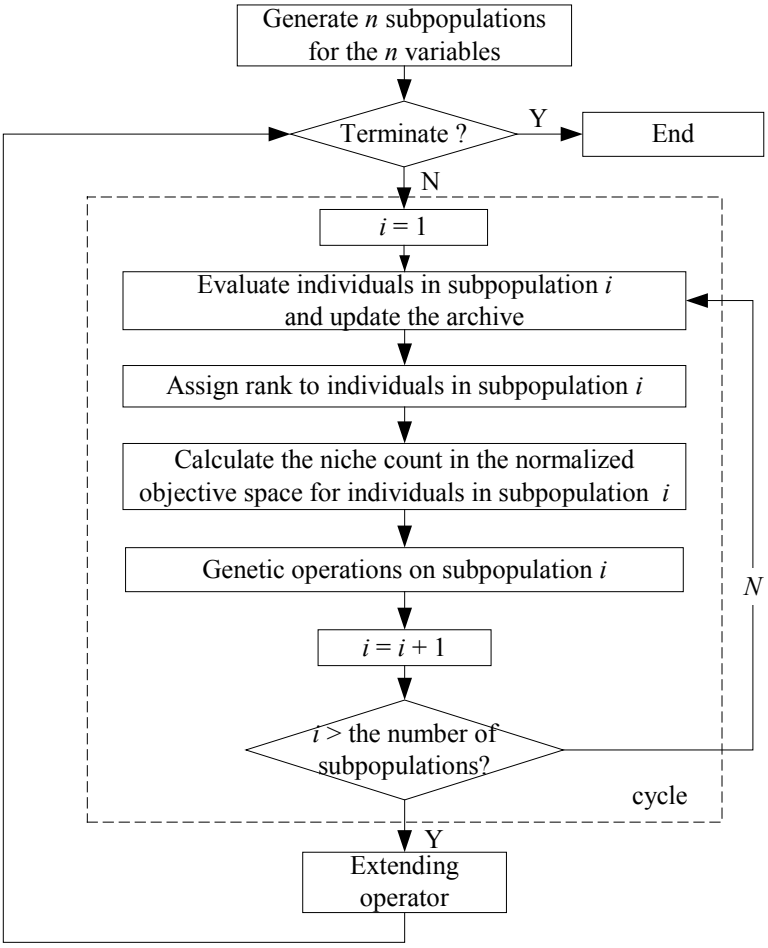


Fig. 3.3. The program flowchart of CCEA

As depicted in the flowchart of CCEA in Fig. 3.3, n subpopulations are randomly initialized and each of them optimizes one variable for a n-variable problem. In the evolution cycle, as marked by the dash box, the n subpopulations will be evolved in a sequential way. To evaluate an individual in the currently evolving subpopulation, a complete solution should be constructed by combining the currently evaluated

individual with the representatives of other subpopulations. The archive will be updated based on the evaluation result of the complete solutions and the ranges of the objective space will be estimated from the updated archive. Based on the objective vector, each individual will be assigned a rank and its respective niche count will be obtained in the dynamic objective space. The genetic operations during the evolution process consist of tournament selection, uniform crossover and bit-flip mutation. Once an evolution cycle is finished, the extending operator finds the archive member residing in the region that is not explored thoroughly, and copies it to subpopulations. With the extending operator, CCEA gives a wide spread and uniform distribution to the non-dominated solution set.

3.3 Distributed Cooperative Coevolutionary Algorithm

3.3.1 Distributed Evolutionary Computing

Although evolutionary algorithm (EA) is a powerful tool, the computational cost involved in terms of time and hardware increases as the size and complexity of the problem increases, since it often needs to perform a large number of function evaluations in the evolution process. One promising approach to overcome the limitation is to exploit the inherent parallel nature of EA by formulating the problem into a distributed computing structure suitable for parallel processing, i.e., to divide a task into subtasks and to solve the subtasks simultaneously using multiple processors. This divide-and-conquer approach has been applied to EA in different ways and many parallel EA implementations have been reported in literatures (Cantú-Paz 1998; Goldberg 1989b; Rivera 2001).

As categorized by Rivera (2001), there are four possible strategies to parallelize EAs, i.e., global parallelization, fine-grained parallelization, coarse-grained parallelization, and hybrid parallelization. In global parallelization, only the fitness evaluations of individuals are parallelized by assigning a fraction of the population to each processor. The genetic operators are often performed in the same manner as traditional EAs since these operators are not as time-consuming as the fitness evaluation. This strategy preserves the behavior of traditional EA and is particularly effective for problems with complicated fitness evaluations. The fine-grained parallelization is often implemented on massively parallel machines, which assigns one individual to each processor and the interactions between individuals are restricted into some neighborhoods. In coarse-grained parallelization, the entire population is partitioned into subpopulations. This strategy is complex since it consists of multiple subpopulations and different subpopulations may exchange individuals occasionally (migration). In hybrid parallelization, several parallelization approaches are combined, and the complexity of these hybrid parallel EAs depends on the level of hybridization.

The availability of powerful-networked computers presents a wealth of computing resources to solve problems with large computational effort. Because the communication amount in coarse-grained parallelization is small compared with other parallelization strategies, it is a suitable computing model for distributed computer network where the communication speed is limited. This parallelization approach is considered here where large problems are divided into many smaller subtasks and mapped into the computers available in a distributed system.

3.3.2 The Distributed CCEA (DCCEA)

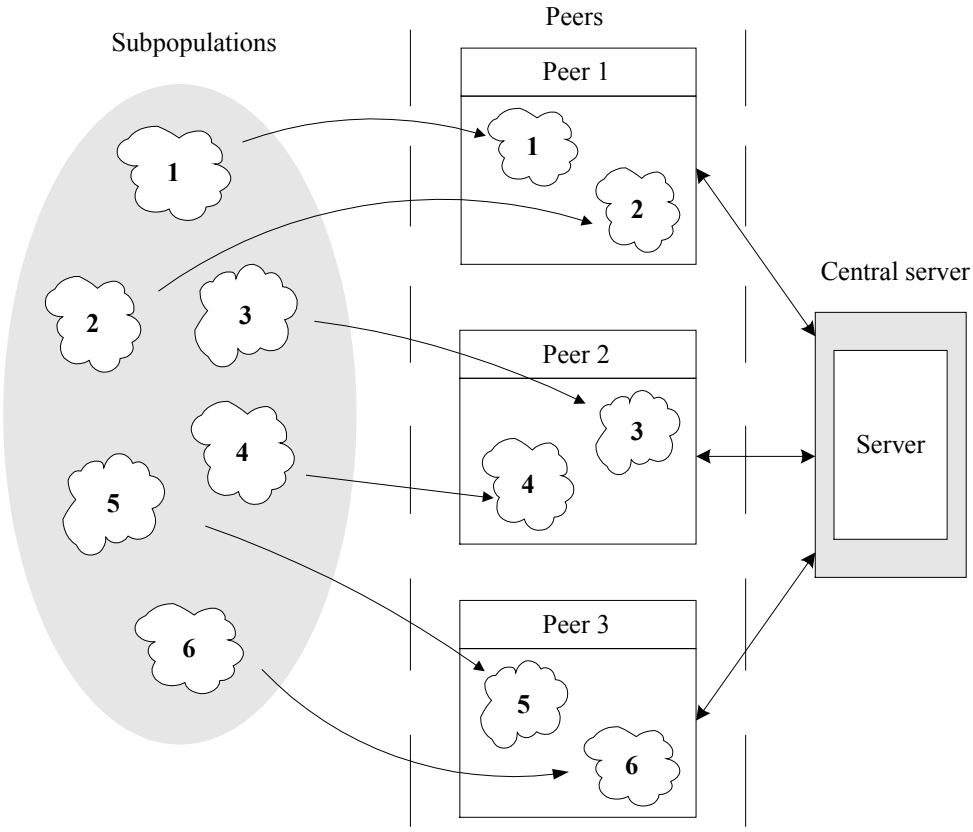


Fig. 3.4. The model of DCCEA

The proposed distributed CCEA adopts the coarse-grained parallelization strategy of EAs. To make the original CCEA fit into a distributed scenario, the design of DCCEA should consider several features of distributed computing such as variant communication overhead, different computation speed and network restrictions. A toy model with six subpopulations and three peers is given in Fig. 3.4 to illustrate the design concept of DCCEA. As shown in Fig. 3.4, each parameter of the problem is assigned a subpopulation as in CCEA. In a distributed scenario, these subpopulations are further partitioned into a number of groups, which is determined by the available number of peers. In Fig. 3.4, the 6 subpopulations are divided into 3 groups and each of them is assigned to a peer computer. Each peer has its own archive and

representatives, and evolves its subpopulations sequentially in the similar way as in CCEA.

Inside a peer computer, the complete solution generated through collaboration will continuously update the peer archive. The subpopulations in the peer update the corresponding peer representatives once every cycle. The cooperation among peers is indirectly achieved through the exchanges of archive and representatives between peers and a central server. In the distributed scenario, the communication time among peers is a conspicuous part of the whole run time. To reduce the communication overhead, the exchange of archive and representatives between one peer and the central server occurs once every several generations. The number of generations between two exchanges is called the exchange interval. Generally the peers are not identical and the cooperation among peers becomes ineffective if there are big differences in the evolution progresses of peers. In such case, the bad cooperation among peers will deteriorate the performance of DCCEA. To keep the peers cooperate well in the evolution, these peers should be synchronized every few generations. Here, the synchronization interval is defined as the number of generations between two synchronizations. The exchange and synchronization intervals can be fixed or adaptively determined along the evolution.

3.3.3 The Implementation of DCCEA

The implementation of DCCEA is embedded into the distributed computing framework named Paladin-DEC (Tan et al. 2002b, 2003a), which is built upon the foundation of Java technology offered by Sun Microsystems and is equipped with application programming interfaces (APIs) and technologies from J2EE. The J2EE is a component-based technology provided by Sun for the design, development, assembly,

and deployment of enterprise applications. Enterprise Java Bean (EJB) is the middle-tier component by which data are presented and business logics are performed. Different tiers are independent from each other and can be changed easily, e.g., such as changing the database or adding/removing some business logics. Furthermore, the unique advantages of Java programming language, such as platform independence and reusability, make this approach attractive.

As shown in Fig. 3.5, the Paladin-DEC software consists of two main blocks, i.e., the servant block and workshop block that are connected by RMI-IIOP (Remote Method Invocation over Internet Inter-ORB Protocol). 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 the specified task, while the free ones wait for the new jobs to be assigned. The servant contains three different servers, i.e., logon 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 peer's information to the dispatcher server. The dispatcher server is responsible for choosing the tasks to be executed, the group of peers to perform the execution, and to transfer the peers' information to/from the database server. The dispatcher server also synchronizes the information, updates the peer's list, and informs the database server for any modification. Whenever there is a task available, the dispatcher server will transfer the task to a group of selected peers.

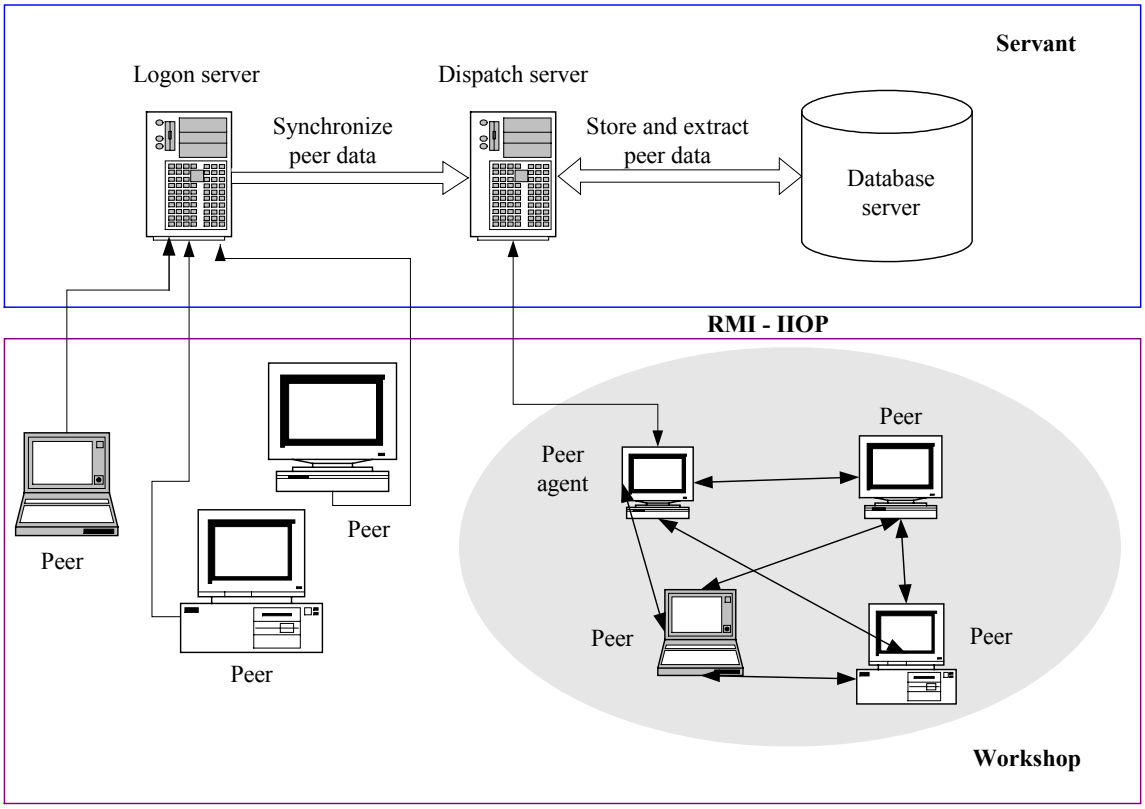


Fig. 3.5. Schematic framework of Paladin-DEC software

The working process of a peer begins once the peer (or client) is started and logons to the server, which is realized by sending a valid email address to the server. The peer computer will then be pooled and waiting for the task to be assigned by the server. Once a peer detects that a task is assigned, it will extract the information from the server, such as class name and path, as well as the http server address before loading the class remotely from the server. If the class loaded is consistent with the Paladin-DEC system, it will be allowed to initiate the computation procedure. Fig. 3.6 depicts the entire working process of a peer, where the detail description of the box "Compute" is shown in the right part of the figure.

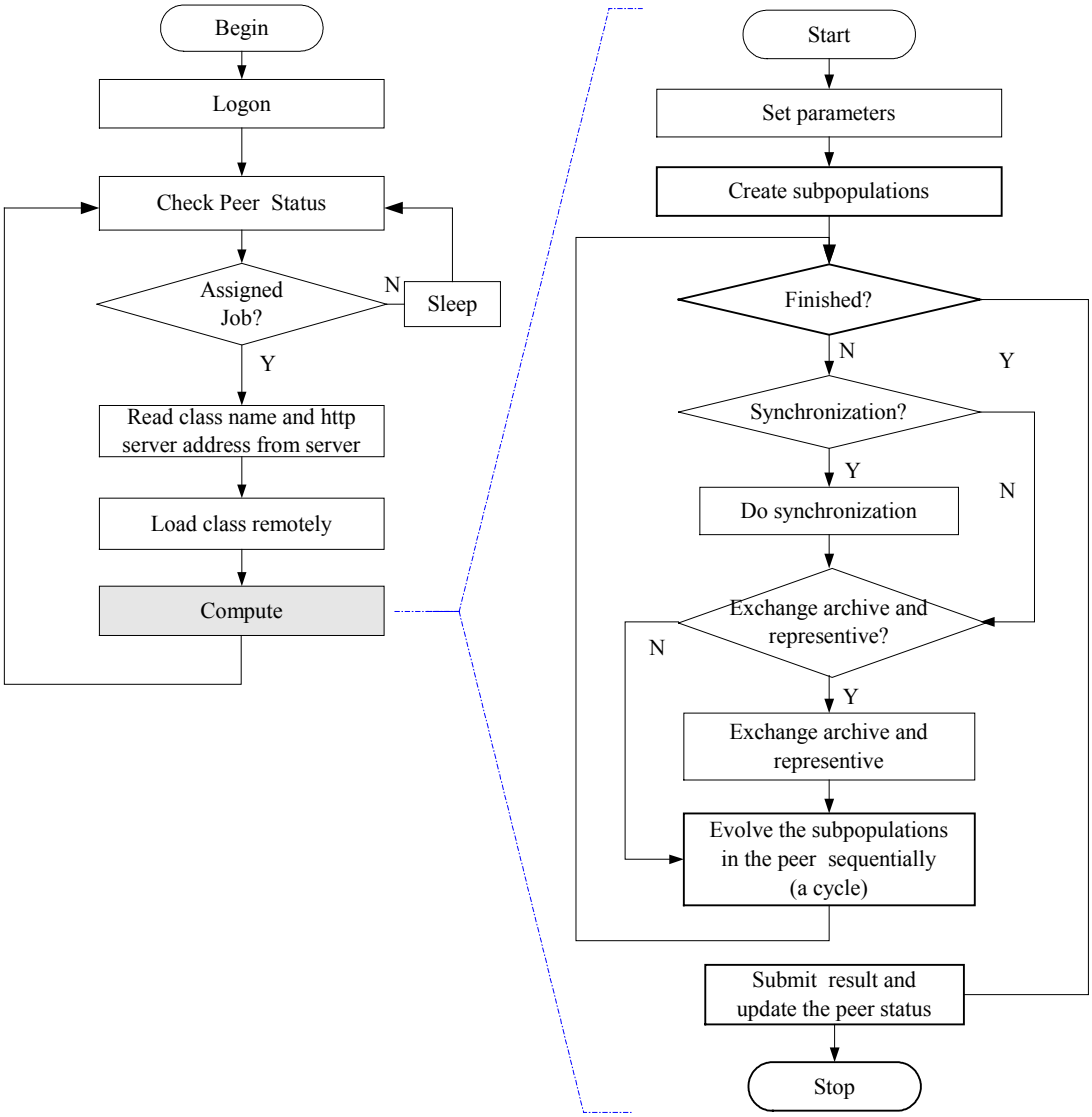


Fig. 3.6. The workflow of a peer

When a peer starts the computation procedure, it first initializes the parameters, such as generation number, subpopulation groups, subpopulation size, crossover rate, and mutation rate. Then the peer creates the subpopulations assigned to it. Synchronization is crucial to DCCEA in order to achieve a good cooperation among peers. When a peer reaches a synchronization point, it suspends its evolution until the server signals that all the peers have reached the synchronization point. At each generation, the peer will check whether it is time to exchange the archive and representatives between the peer

and the server. If the conditions of exchange are satisfied, the peer will initiate a session in the server that retrieves the archive and representatives of the peer, then updates the server archive with the peer archive and updates the server representatives corresponding to the peer. For the peer, it will obtain the new server archive and server representatives, and replaces its current archive and representatives. After these steps, the peer evolves its subpopulations sequentially for one generation. If the peer meets the termination conditions, it will initiate a session to submit the results and then restore 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.

3.3.4 Workload Balancing

As the processing power and specification for various computers in a network may be different, the feature of work balancing that ensures the peers are processed in a similar pace is required in DCCEA. This is important since the total computation time is decided by the peer that finished the work last, and if the peer with the least computational capacity is assigned with the heaviest workload, not only would longer time be required but also the bad cooperation among nodes will deteriorate the performance of DCCEA. Intuitively, work balancing for a distributed system could be difficult because the working environment in a network is often complex and uncertain. The DCCEA resorts to a simple work balancing strategy by assigning the workload to the peers according to their respective computational capabilities. As stated in Section 3.3.3, when a peer is first launched, it uploads its configuration information, which could be accessed by the servant. The hardware configuration of the peer is recorded in the information file, such as the CPU speed, RAM size, etc. After reading the information file, the dispatch server performs a simple task

scheduling and assigns different tasks to the respective peers according to their computational capabilities.

3.4 Case study

In this section, four performance metrics for multiobjective optimization are described. Then some benchmark problems are described, which will be used in the comparison of CCEA with PAES, PESA, NSGAII, SPEA2, and IMOEA. In this section, the extensive simulations of the algorithms are performed based upon the benchmark problems and simulations of DCCEA are presented to verify its performance.

3.4.1 Performance Metrics

Four different quantitative performance measures for MO optimization are used, which are referred from other studies in MO optimization, such as Van Veldhuizen and Lamont (1999), Deb (2001), and Zitzler et al. (2000). These measures are chosen here since they have been widely used for performance comparisons in MO optimization, and can evaluate the non-dominated solutions in several nontrivial aspects.

1) Generational Distance (GD)

The metric of generational distance is a value representing how "far" the PF_{known} is from PF_{true} and is defined as,

$$GD = (\frac{1}{n}\sum_{i=1}^{n} d_i^2)^{1/2} \tag{3.1}$$

where n is the number of members in PF_{known}, d_i is the Euclidean distance (in objective space) between the member i in PF_{known} and its nearest member of PF_{true}. The smaller the generational distance is, the closer the PF_{known} is to the PF_{true}.

2) Spacing (S)

The metric of spacing measures how "evenly" members in PF_{known} distribute. It is defined as,

$$S = [\frac{1}{n}\sum_{i=1}^{n}(d_i - \bar{d})^2]^{1/2} / \bar{d},\ where\ \bar{d} = \frac{1}{n}\sum_{i=1}^{n} d_i \tag{3.2}$$

where n is the number of members in PF_{known}, d_i is the Euclidean distance (in objective space) between the member i in PF_{known} and its nearest member of PF_{known}. The smaller the spacing is, the more evenly the members in PF_{known} distribute.

3) Maximum Spread (MS)

Zitzler et al. (2000) defined a metric measuring how well the PF_{true} is covered by the PF_{known} through the hyper-boxes formed by the extreme function values observed in PF_{true} and PF_{known}. In order to normalize the metric, this metric is modified as,

$$\bar{D} = \sqrt{\frac{1}{M}\sum_{m=1}^{M}\{[(\min(f_m^{\max}, F_m^{\max}) - \max(f_m^{\min}, F_m^{\min})]/(F_m^{\max} - F_m^{\min})]\}^2} \tag{3.3}$$

where n is the number of members in PF_{known}; $f_m^{\max}$, $f_m^{\min}$ are the maximum and minimum of the $m \cdot th$ objective in the PF_{known}; $F_m^{\max}$, $F_m^{\min}$ are the maximum and

minimum of the $m \cdot th$ objective in the PF_{true}. The greater the maximum spread is, the more area of PF_{true} is covered by the PF_{known}.

4) Hyper-Volume (HV) and Hyper-Volume Ratio (HVR)

The metric of hyper-volume calculates the volume (in the objective space) covered by the members of a non-dominated set for multiobjective minimization problems (Van Veldhuizen and Lamont 1999; Zitzler and Thiele 1999). It is defined as,

$$HV = volume(\cup_{i=1}^{n} v_i) \tag{3.4}$$

Mathematically, for each member i in the non-dominated set, a hypercube v_i is constructed with a reference point W and the member i as the diagonal corners of the hypercube. The reference point can simply be found by constructing a vector of the worst objective function values. To eliminate the bias to some extent and to be able to calculate a normalized value of this metric of hyper-volume, Van Veldhuizen and Lamont (1999) used the metric of hyper-volume ratio that is the ratio of the hyper-volume of PF_{known} and the hyper-volume of PF_{true},

$$HVR = HV(PF_{known}) / HV(PF_{true}) \tag{3.5}$$

It measures the evenness and range of PF_{known} with respect to PF_{true} at the same time. The greater the hyper-volume ratio is, the better the PF_{known} covers the PF_{true}.

3.4.2 The Test Problems

Nine test problems are used here to validate the performance of CCEA. Table 3.1 summarizes features of these test problems and Fig. 3.7 illustrates the respective Pareto

fronts. These problems include important characteristics that are suitable for validating the effectiveness of MO optimization methods in maintaining the population diversity as well as converging to the final Pareto front. Many researchers including Knowles and Corne (2000), Corne et al. (2000), Deb (2002a), Tan et al. (2001), and Zitzler et al. (1999, 2000, 2001), have used these problems in the validation of their algorithms.

Table 3.1. Features of the test problems

	Test problem	Features
1	ZDT1	The Pareto front is convex
2	ZDT2	The Pareto front is non-convex
3	ZDT3	The Pareto front consists of several noncontiguous convex parts
4	ZDT4	The Pareto front is highly multi-modal and there are 21^9 local Pareto fronts
5	ZDT6	The Pareto-optimal solutions are non-uniformly distributed along the global Pareto front. The density of the solutions is the lowest near the Pareto-optimal front and the highest away from the front
6	FON	The Pareto front is non-convex
7	KUR	The Pareto front consists of several noncontiguous convex parts
8	TLK	Noisy landscape
9	DTL2	High dimension of the objective space

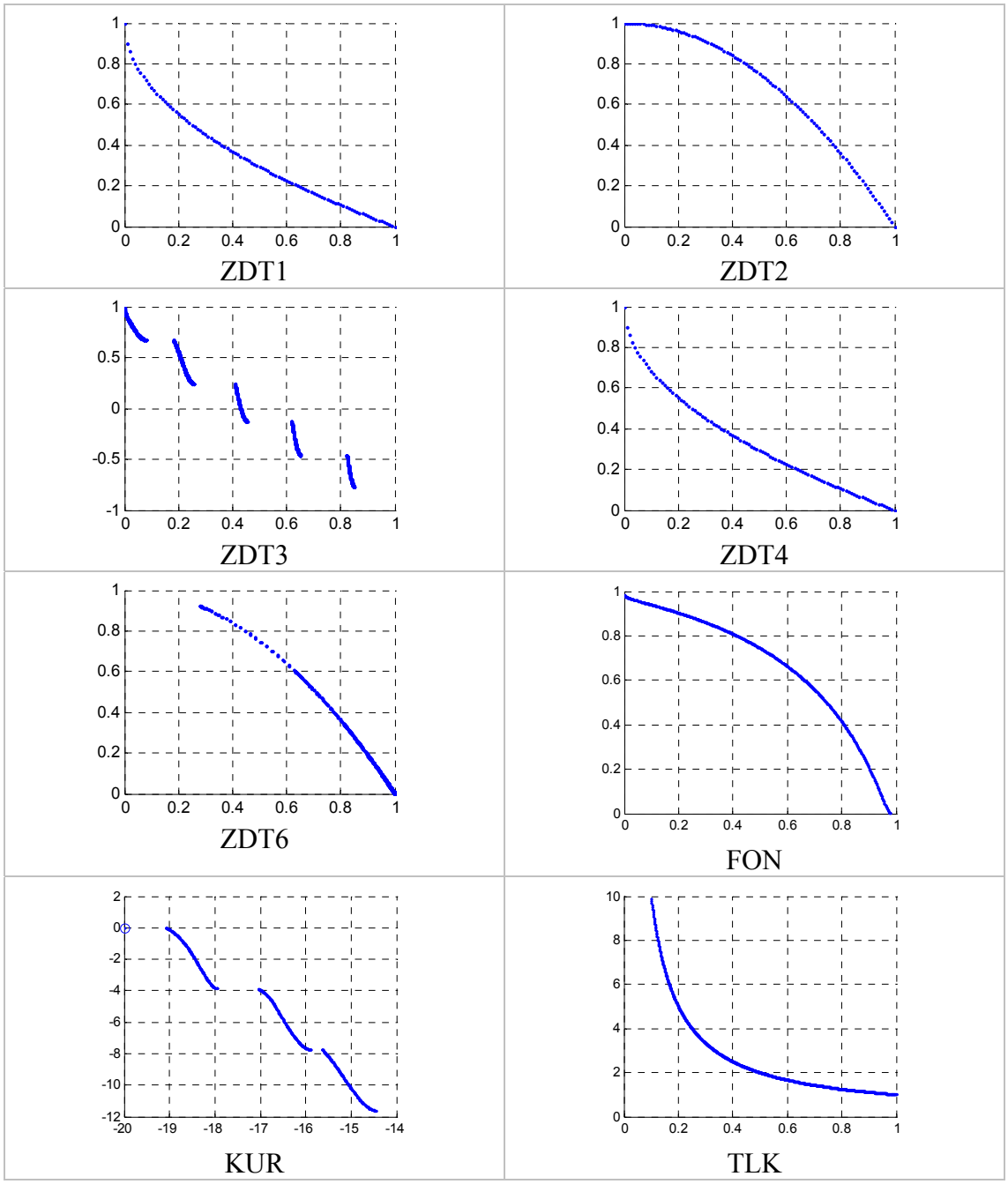


Fig. 3.7. The Pareto fronts of the test problems

1) Test Problem ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6

These problems were designed using Deb's scheme by Zitzler et al. (2000) and were used in a performance comparison of eight well-known MOEAs. Each of these test problems is structured in the same manner and is consists of three functions (Deb 1999). The definitions of the three functions f_1, g, h in ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6 are listed in Table 3.2.

$$\begin{aligned} &Minimize \;\; T(x) = (f_1(x_1), f_2(x)) \\ &subject\ to\ f_2(x) = g(x_2, \cdots, x_m)h(f_1(x_1), g(x_2, \cdots, x_m)) \\ &where\ x = (x_1, \cdots, x_m) \end{aligned} \tag{3.6}$$

Table 3.2. Definitions of f_1, g, h in ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6

ZDT1	$f_1(x_1) = x_1$ $g(x_2, \cdots, x_m) = 1 + 9 \cdot \sum_{i=2}^{m} x_i / (m-1)$ $h(f_1, g) = 1 - \sqrt{f_1 / g}$ $where\ m = 30,\ and\ x_i \in [0,1].$	(3.7)
ZDT2	$f_1(x_1) = x_1$ $g(x_2, \cdots, x_m) = 1 + 9 \cdot \sum_{i=2}^{m} x_i / (m-1)$ $h(f_1, g) = 1 - (f_1 / g)^2$ $where\ m = 30,\ and\ x_i \in [0,1].$	(3.8)
ZDT3	$f_1(x_1) = x_1$ $g(x_2, \cdots, x_m) = 1 + 9 \cdot \sum_{i=2}^{m} x_i / (m-1)$ $h(f_1, g) = 1 - \sqrt{f_1 / g} - (f_1 / g)\sin(10\pi f_1)$ $where\ m = 30,\ and\ x_i \in [0,1].$	(3.9)
ZDT4	$f_1(x_1) = x_1$ $g(x_2, \cdots, x_m) = 1 + 10(m-1) + \sum_{i=2}^{m} (x_i^2 - 10\cos(4\pi x_i))$ $h(f_1, g) = 1 - \sqrt{f_1 / g}$ $where\ m = 10,\ x_1 \in [0,1] and\ x_2, \cdots, x_m \in [-5,5].$	(3.10)
ZDT6	$f_1(x_1) = 1 - \exp(-4x_1)\sin^6(6\pi x_1)$ $g(x_2, \cdots, x_m) = 1 + 9((\sum_{i=2}^{m} x_i) / (m-1))^{0.25}$ $h(f_1, g) = 1 - (f_1 / g)^2$ $where\ m = 10,\ x_i \in [0,1]$	(3.11)

2) Test Problem FON

Test problem FON is Fonseca's two-objective minimization problem that has been widely studied (Fonseca and Fleming 1993; Tan et al. 2001, 2003b; Van Veldhuizen and Lamont 1999). Besides its non-convex Pareto front, this test problem has a large and nonlinear trade-off curve that is suitable to challenge the algorithm's ability in finding and maintaining the entire Pareto front uniformly. In addition, the performance of algorithms can easily be compared via visualization of the Pareto front for this problem. This two-objective minimization problem is given by

$$\begin{aligned} &Minimize(f_1, f_2) \\ &\begin{cases} f_1(x_1,...,x_8) = 1 - \exp[-\sum_{i=1}^{8}(x_i - 1/\sqrt{8})^2] \\ f_2(x_1,...,x_8) = 1 - \exp[-\sum_{i=1}^{8}(x_i + 1/\sqrt{8})^2] \end{cases} \\ &where \ -2 \le x_i < 2, \forall i = 1,2,\cdots,8 \end{aligned} \tag{3.12}$$

There are eight parameters $(x_1,\cdots,x_8)$ to be optimized so that f_1 and f_2 are minimal. Due to the symmetry and trade-offs of these two functions, the Pareto-optimal sets are points on the curve defined as (Fonseca and Fleming 1993),

$$x_1 = x_2 = \cdots = x_8, \frac{-1}{\sqrt{8}} \le x_1 \le \frac{1}{\sqrt{8}} \tag{3.13}$$

3) Test Problem KUR

Kursawe (1990) used a two-objective optimization problem that is very complicated. The Pareto front is non-convex as well as disconnected. There are three distinct disconnected regions in the Pareto front. The decision variable values corresponding to the Pareto front are also disconnected in the decision variable space and difficult to know as given below,

$$\begin{aligned} &Minimize(f_1, f_2) \\ &\begin{cases} f_1(x) = \sum_{i=1}^{2}[-10\exp(-0.2\sqrt{x_i^2 + x_{i+1}^2})] \\ f_2(x) = \sum_{i=1}^{3}[|x_i|^{0.8} + 5\sin(x_i^3)] \end{cases} \\ &where \ -5 \le x_i < 5, \forall i = 1,2,3 \end{aligned} \tag{3.14}$$

4) Test Problem TLK

Tan et al. (2002a) constructed this test problem to evaluate search algorithms in a noisy environment to test their robustness in the sense that the disappearance of important individuals from the population has little effect on the global evolution behavior,

$$\begin{aligned} &Minimize(f_1, f_2) \\ &f_1 = x_1 \\ &f_2 = \frac{1}{x_1}\left\{1 + \left(x_2'^2 + x_3'^2\right)^{0.25}\left[\sin^2\left(50\left(x_2'^2 + x_3'^2\right)^{0.1}\right) + 1.0\right]\right\} \end{aligned} \tag{3.15}$$

Instead of performing the optimization on the 'real' parameters, x_i, the optimization is performed on the 'corrupted' parameters with additive noise elements,

$$x_i' = x_i + N(\sigma, \mu) \tag{3.16}$$

where $0.1 \le x_1 \le 1$; $-100 \le x_i \le 100 \ \forall i = 2,3$ and $N(\sigma,\mu)$ is a white noise. The distribution density of the noise is given as normal distribution,

$$P(x \mid N(\sigma, \mu)) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{3.17}$$

where μ and σ are the mean and variance of the probability density distribution. In the normal curve, approximately 68% of the scores of the distribution lie between $\mu \pm \sigma$. On this test problem, both μ and σ are given as 0.0 and 0.1, respectively. Note that the noisy search environment is modeled with the corrupted parameters. This is to provide

noisy global optimum points in the parameter domain, while maintaining the global Pareto front in the objective domain for easy comparison or illustration.

5) Test Problem DTL2

This problem was designed by Deb et al. (2002b) to test the MOEAs' ability to solve problems with a large number of objectives. It is scalable, easy to construct and understand,

$$Minimize(f_1, f_2, \cdots, f_M) \tag{3.18}$$

$$\begin{cases} f_1(x) = (1+g(\mathbf{x}_M))\cos(x_1\pi/2)\cdots\cos(x_{M-1}\pi/2) \\ f_2(x) = (1+g(\mathbf{x}_M))\cos(x_1\pi/2)\cdots\sin(x_{M-1}\pi/2) \\ \vdots \\ f_M(x) = (1+g(\mathbf{x}_M))\sin(x_1\pi/2) \\ g(\mathbf{x}_M) = \sum_{x_i\in\mathbf{x}_M}(x_i-0.5)^2 \end{cases}$$

$$where\ M=5,\ \mathbf{x}_{\mathrm{M}} = \{x_M, \cdots, x_{M+9}\}, x_i \in [0,1], \forall i = 1,2,\cdots, M+9$$

All the points on the Pareto front satisfy the equation below,

$$\sum_{i=1}^{M} f_i^2 = 1 \tag{3.19}$$

3.4.3 Simulation Results of CCEA

In this section, simulations are carried out to validate the performance of CCEA in several aspects, which include the discovery and distribution of non-dominated solutions along the entire Pareto front uniformly, the escape from harmful local optima and the minimization of the effect of noise induced from the environment (robustness). The performance is compared between CCEA and various multiobjective optimization methods based on the nine test problems described in Section 3.4.1. Besides CCEA,

other evolutionary multiobjective optimization methods used for the study include PAES, PESA, NSGAII, SPEA2 and IMOEA. In order to guarantee a fair comparison, all the algorithms considered are implemented with the same binary coding scheme of 30-digit per decision variable, tournament selection, uniform crossover, and bit-flip mutation. The number of evaluations in each run is fixed and the configurations of the algorithms are shown in Table 3.3.

Table 3.3. The configurations of the MOEAs

Populations	Subpopulation size 20 in CCEA; population size 100 in PESA, NSGAII, SPEA2; population size 1 in PAES; initial population size 20, maximum population size 100 in IMOEA. Archive (or secondary population) size 100 in all the algorithms
Chromosome length	30 bits for each variable
Selection	Binary tournament selection
Crossover rate	0.8
Crossover method	Uniform crossover
Mutation rate	$2/L$, where L is the chromosome length, for ZDT1, ZDT2, ZDT3, ZDT4, ZDT6, TLK, and DTL2; 1/30, where 30 is the bit number of one variable, for FON, and KUR
Mutation method	Bit-flip mutation
Hyper-grid size	2^3 per dimension for DTL2; 2^5 per dimension for other problems
Representative number	2 for FON and KUR; 1 for other problems
Number of evaluations	120,000

3.4.3.1 Performance Comparisons

In the simulations, 30 independent runs (with random initial populations) of CCEA, PAES, PESA, NSGAII, SPEA2 and IMOEA are performed on each of the nine test functions in order to study the statistical performance, such as consistency and robustness of the methods. Fig. 3.8(a-d) summarizes the simulation results of the algorithms for the problems ZDT1, ZDT2, ZDT3, ZDT4, ZDT6, FON, KUR and TLK. The distribution of simulation data for 30 independent runs is represented in the box plot format (Chambers et al. 1983). Each box plot represents the distribution of a sample set where a horizontal line within the box encodes the median, while the upper and lower ends of the box are the upper and lower quartiles. The appendages illustrate the spread and shape of distribution, and dots represent the outside values.

Maybe PAES is the simplest possible multiobjective evolutionary algorithm while providing competitive results. For almost all the test problems and all the metrics, the performance of PAES is the worst and the variance is large compared to other MOEAs. A possible reason is that PAES is a non-population based local search algorithm where the mutation acts as local search method. It seems that a population of candidate solutions is helpful to improve the result consistency.

With respect to the generational distance, the results show that PESA gives the best good performance for problems of ZDT1, ZDT2, ZDT3 and KUR. CCEA is found to be very competitive for all the problems and it outperforms other MOEAs for the problems of ZDT4 and ZDT6, FON and DTL2. The problem ZDT4 has many local Pareto fronts that challenge the ability of algorithms to escape from harmful local optima. As can be seen from Fig. 3.8(b), only CCEA has the chance to find the global

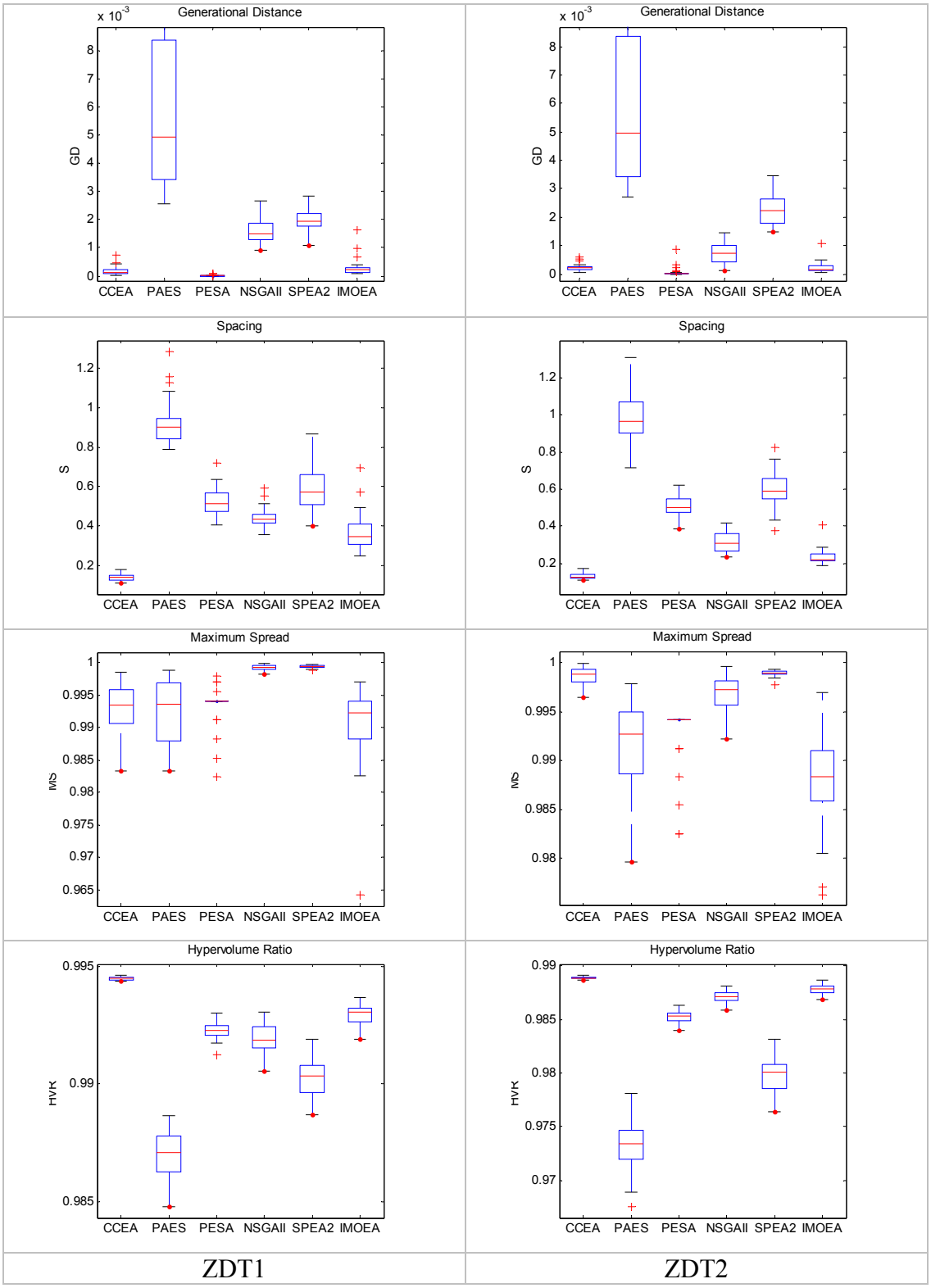
Pareto front while other MOEAs are trapped by the local Pareto fronts. It shows that CCEA has a strong ability to escape from the local optima. The non-uniform distribution of solutions makes ZDT6 difficult to be tackled by MOEAs. Once again, CCEA is clearly better than other MOEAs. All the results prove that the cooperative coevolution can work well in MO optimization and can effectively push solutions to the global Pareto front.

Concerning the metric of spacing, CCEA shows distinct advantage over other MOEAs. For all the test problems except TLK, CCEA performs the best in maintaining the diversity of solutions and distributing solutions uniformly along the discovered Pareto front. Even for the problem TLK with noise on parameters, CCEA is comparable with other MOEAs. These successes are attributed to the extending operator that guides the search to gaps and boundaries and fills the under-populated regions with new generated solutions. Such idea is general and can be used in other MOEAs.

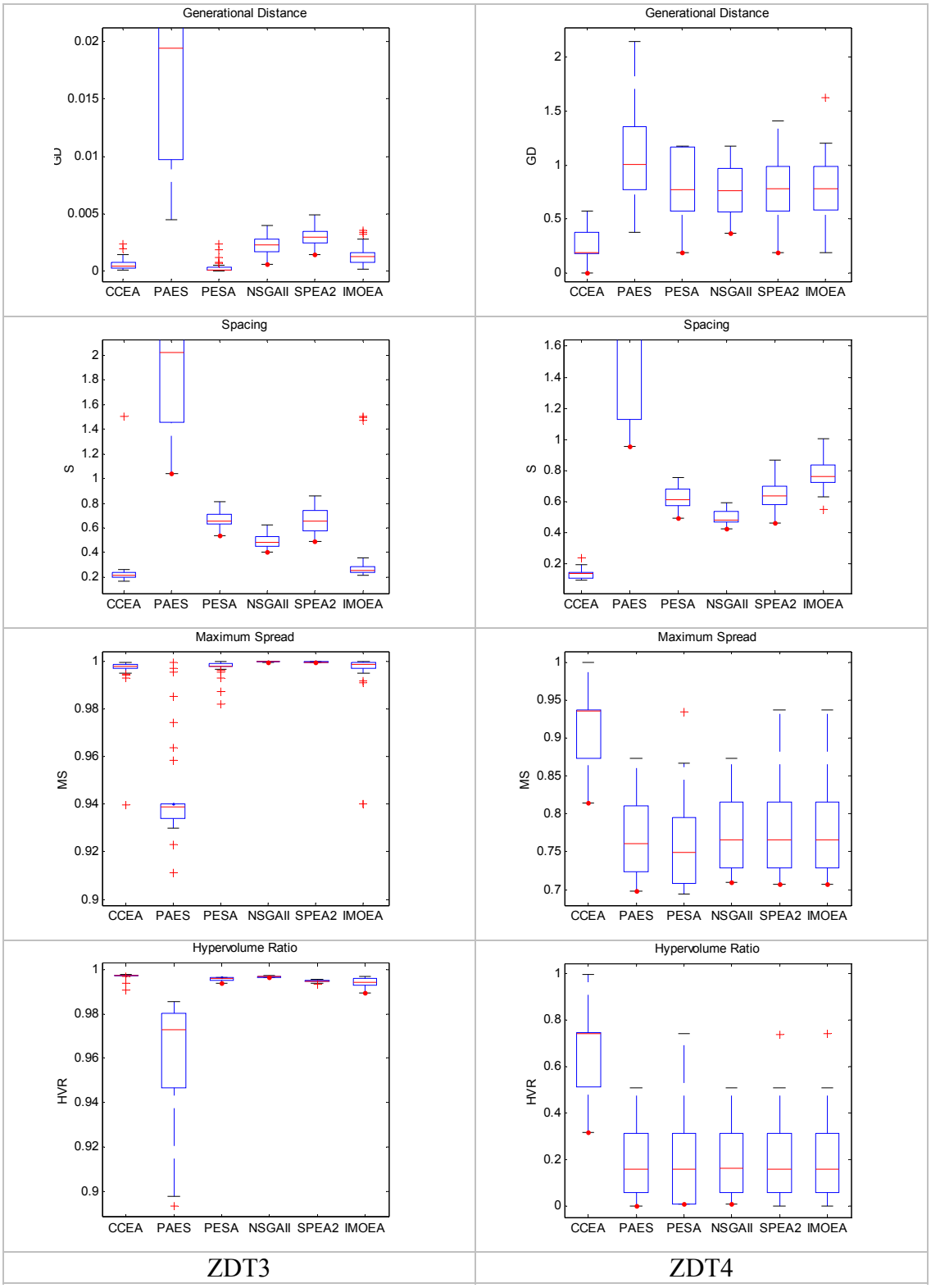
For the metrics of maximum spread and hyper-volume ratio, the CCEA is competitive in exploring the spread of non-dominated solutions for all cases. This is consistent with the excellent performance of CCEA in the metrics of generational distance and spacing. For the problem ZDT4, the maximum spread and hyper-volume ratio of CCEA are much higher than other algorithms. The reason is that the PAES, PESA, NSGA II, SPEA 2, and IMOEA stop at the local Pareto fronts and their solution set cannot approximate the true Pareto front nicely.

The problem DTL2 has a large number of objectives, which bring the difficulty for MOEAs to produce enough pressure to push solutions to the Pareto front. Fig. 3.8(e)

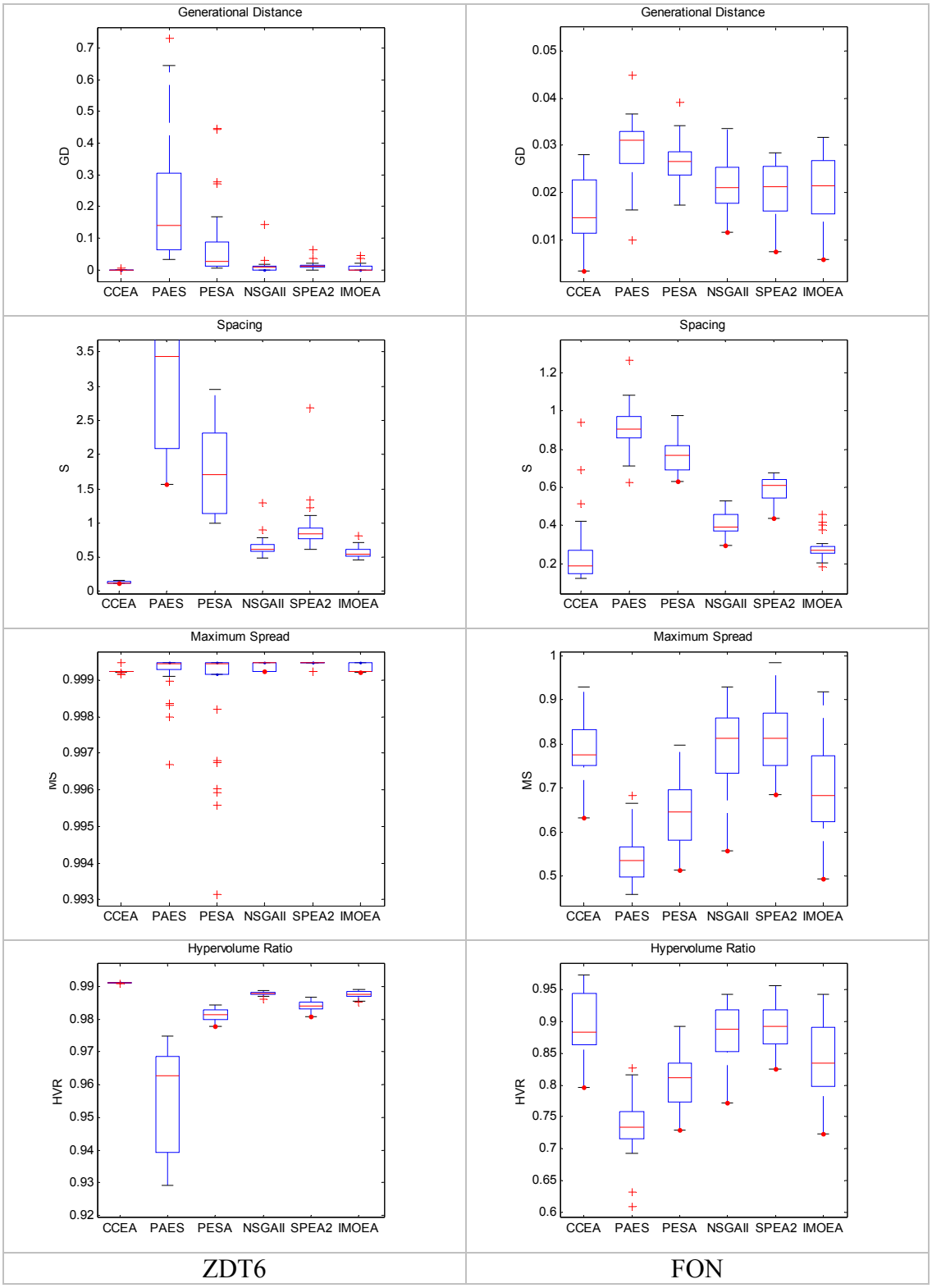
shows that CCEA scales well with PAES and PESA, while NSGAII, SPEA2 and IMOEA suffered in converging to the optimal Pareto front.



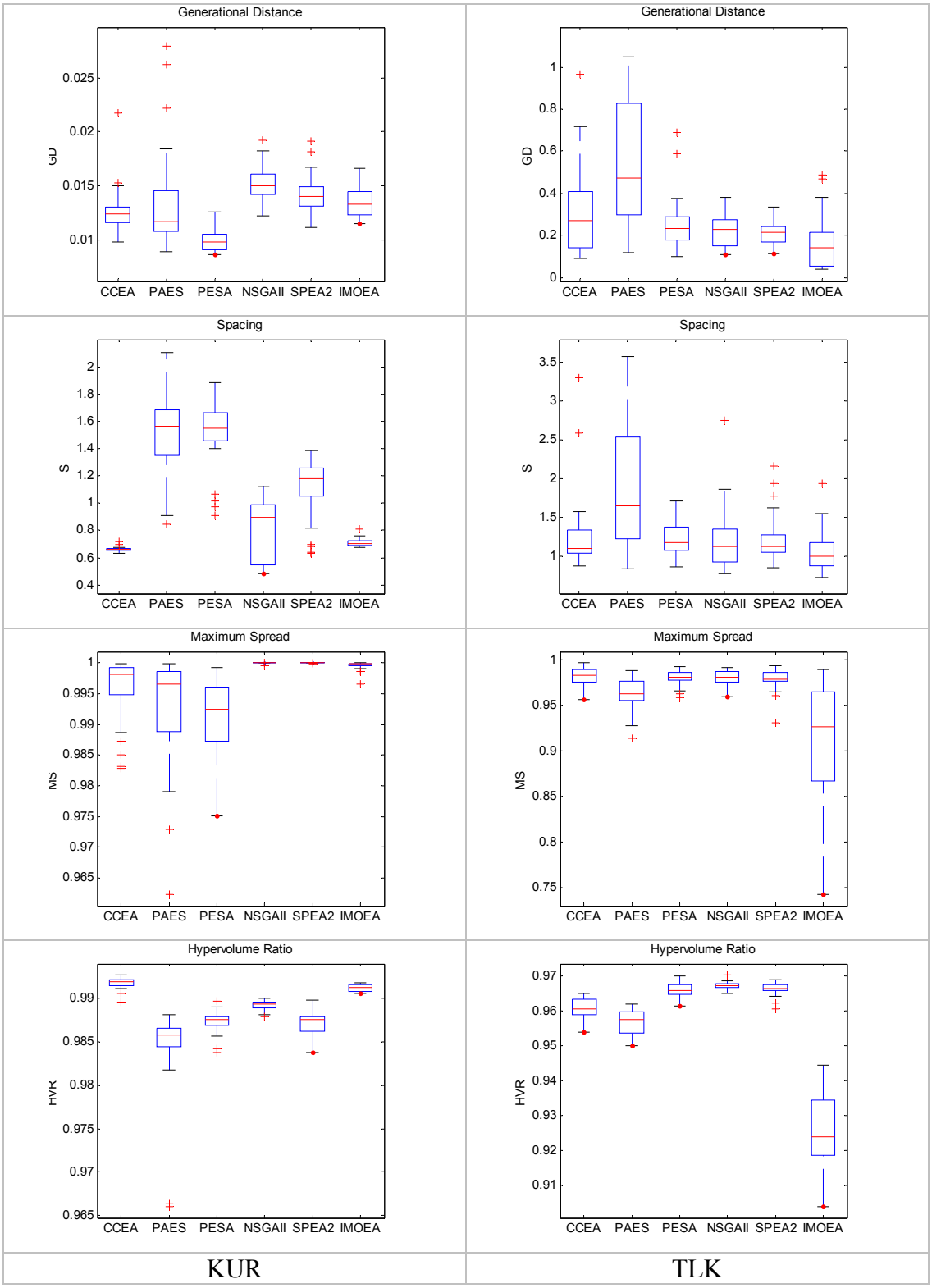
(a)


Generational Distance
GD
CCEA PAES PESA NSGAII SPEA2 IMOEA
Spacing
S
Maximum Spread
MS
Hypervolume Ratio
HVR
ZDT3
ZDT4

(b)


Generational Distance
GD
CCEA PAES PESA NSGAII SPEA2 IMOEA
Generational Distance
GD
CCEA PAES PESA NSGAII SPEA2 IMOEA
Spacing
S
CCEA PAES PESA NSGAII SPEA2 IMOEA
Spacing
S
CCEA PAES PESA NSGAII SPEA2 IMOEA
Maximum Spread
MS
CCEA PAES PESA NSGAII SPEA2 IMOEA
Maximum Spread
MS
CCEA PAES PESA NSGAII SPEA2 IMOEA
Hypervolume Ratio
HVR
CCEA PAES PESA NSGAII SPEA2 IMOEA
Hypervolume Ratio
HVR
CCEA PAES PESA NSGAII SPEA2 IMOEA
ZDT6
FON

(c)


Generational Distance
GD
CCEA PAES PESA NSGAII SPEA2 IMOEA
Generational Distance
GD
CCEA PAES PESA NSGAII SPEA2 IMOEA
Spacing
S
CCEA PAES PESA NSGAII SPEA2 IMOEA
Spacing
S
CCEA PAES PESA NSGAII SPEA2 IMOEA
Maximum Spread
MS
CCEA PAES PESA NSGAII SPEA2 IMOEA
Maximum Spread
MS
CCEA PAES PESA NSGAII SPEA2 IMOEA
Hypervolume Ratio
HVR
CCEA PAES PESA NSGAII SPEA2 IMOEA
Hypervolume Ratio
HVR
CCEA PAES PESA NSGAII SPEA2 IMOEA
KUR
TLK

(d)

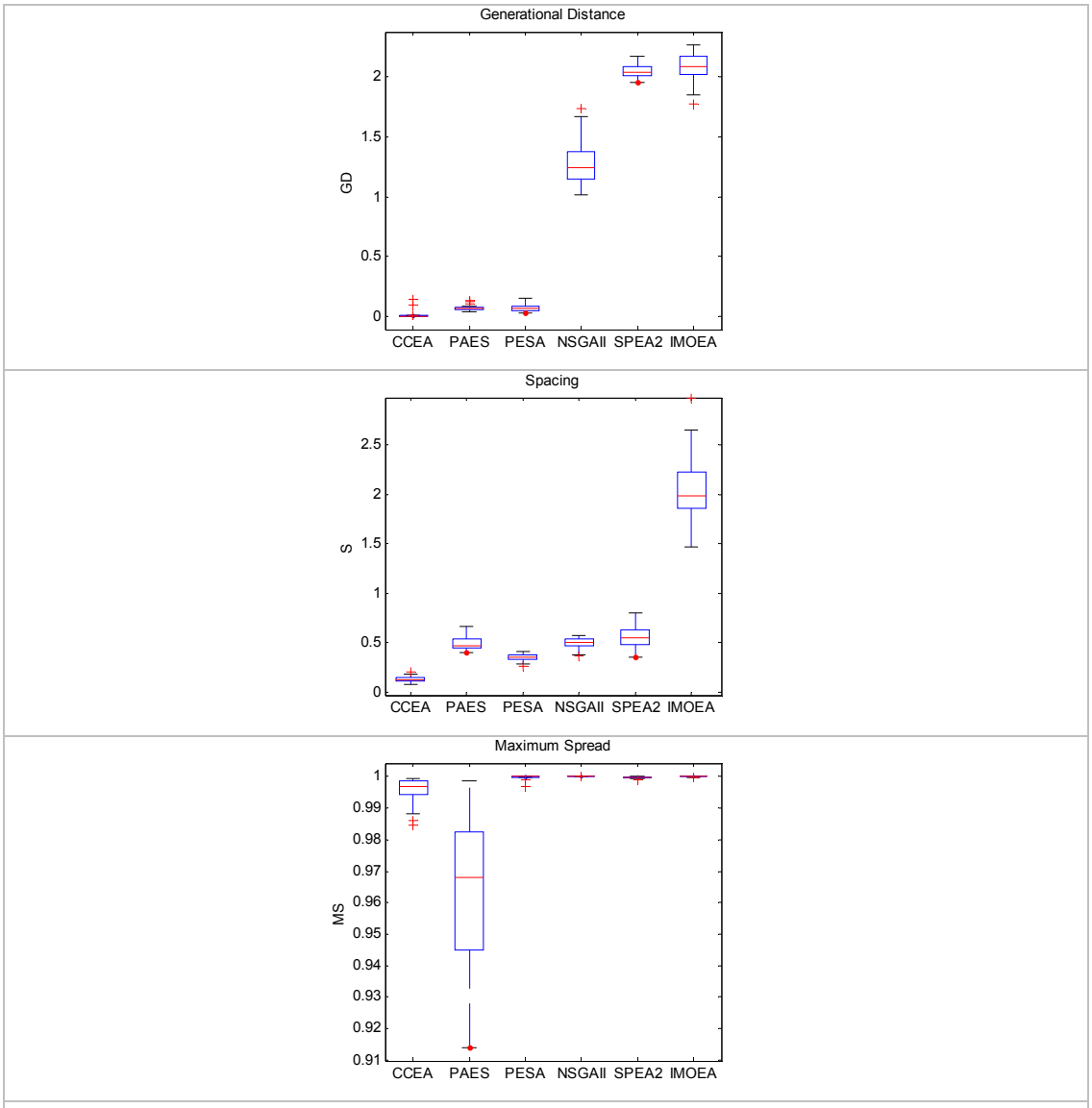


Fig. 3.8. Box plots for the metrics of GD, S, MS, and HVR

The dynamic characteristics of CCEA on four metrics for test problems ZDT4 and ZDT6 are illustrated in Fig. 3.9. These graphs describe the evolution of various metric values along the number of function evaluations. As shown in the figure of GD, there are four steps along the evolution for ZDT4. Each step means that CCEA jumps out of a local Pareto front. Through these jumps, CCEA reaches the global Pareto front at the end of the evolution. Corresponding to the jumps of GD, pulses of spacing can be

found for ZDT4. With Fig. 3.9, the evolution process of CCEA can be observed in detail, which gives us a better understanding of how CCEA works.

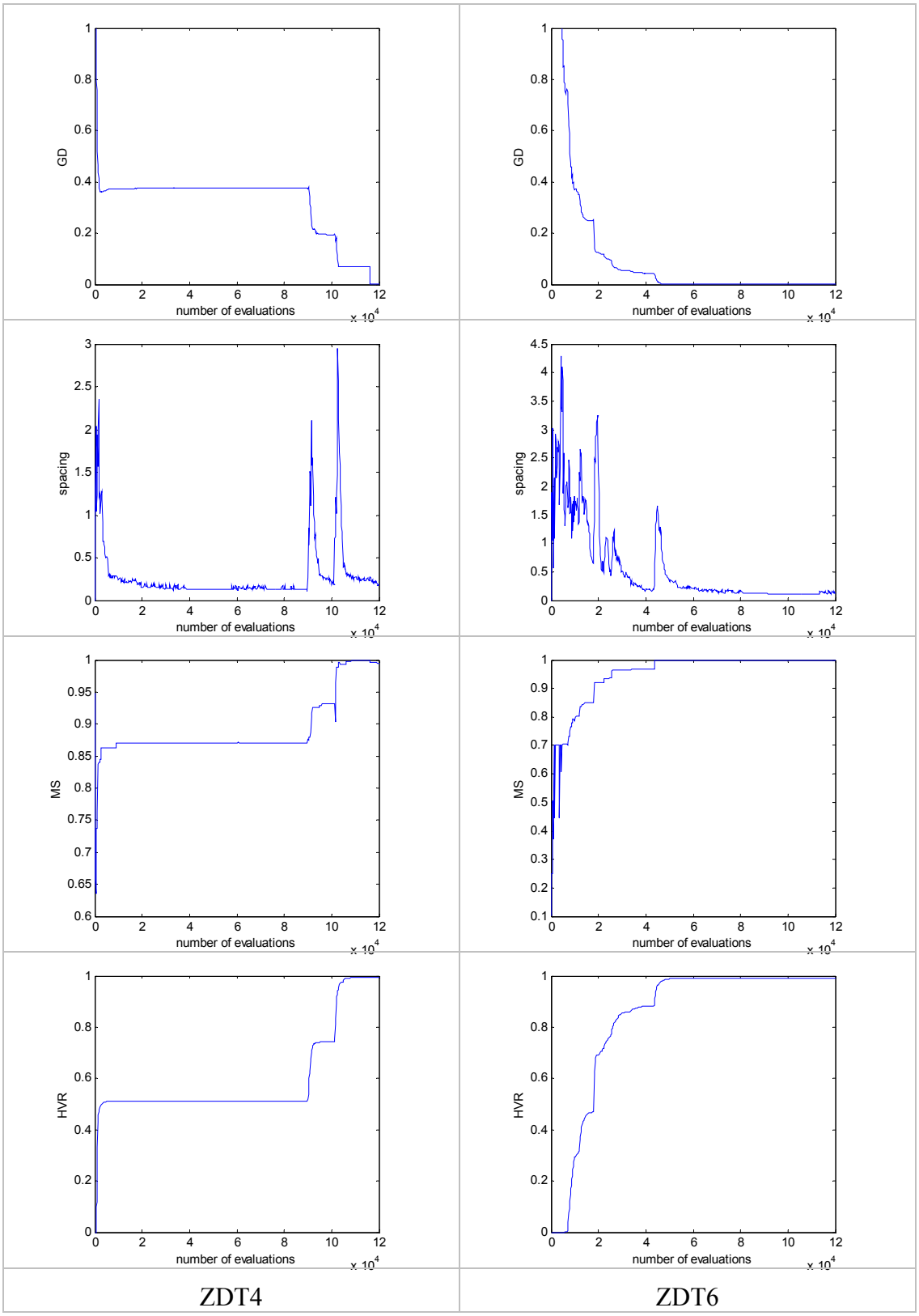


Fig. 3.9. Dynamic behaviors of the CCEA in multiobjective optimization

3.4.3.2 Effect of Extending Operator

To further verify effectiveness of the extending operator, CCEA without extending operator, CCEA with extending operator (clone number n = 1) and CCEA with extending operator (n = 2) were run for 30 times respectively for all the test problems. Table 3.4 lists the median generational distance for the 30 runs. Although the motivation of extending operator is not to reduce the generational distance, it is beneficial to the reduction of generational distance. It seems that the spacing and spread of the non-dominated solutions are correlated to the generational distance and their improvements are helpful for the convergence to the Pareto front.

Table 3.5 and Table 3.6 list the median spacing and median maximum spread respectively for 30 simulation runs. In most cases, the extending operator can improve the performance metrics of spacing and spread. Although the extending operator has resulted negative effects in some cases, such as ZDT1 and ZDT4, these effects are small. The tables show that the results for extending operator with n = 1 are better than n = 2. Here, the subpopulation size is only set at 20 and is relatively small for a population-based algorithm, which suggests that one clone is enough to guide the search as more clones may reduce the solution diversity. For test problem ZDT3 with discontinuous Pareto front, the extending operator is able to reduce the spacing greatly. Besides, the extending operator is capable of reducing the spacing and improving the maximum spread of the non-dominated solutions for the problem FON. The results for other problems also illustrate that the extending operator is effective in improving smoothness and maximum spread of the non-dominated solutions.

Table 3.4. Median generational distance of CCEA with/without the extending operator

Problem	**CCEA without extending operator**	**CCEA with extending operator (*n*=1)**	**CCEA with extending operator (*n*=2)**
ZDT1	1.80E-04	1.32E-04	1.76E-04
ZDT2	2.52E-04	2.15E-04	1.44E-04
ZDT3	7.01E-04	4.05E-04	4.29E-04
ZDT4	1.87E-01	1.85E-01	1.85E-01
ZDT6	5.28E-07	4.92E-07	4.95E-07
FON	2.66E-02	1.47E-02	1.34E-02
KUR	1.37E-02	1.24E-02	1.49E-02
TLK	2.69E-01	2.69E-01	2.68E-01
DTL2	1.15E-03	8.57E-04	1.03E-03

Table 3.5. Median spacing of CCEA with/without the extending operator

Problem	**CCEA without extending operator**	**CCEA with extending operator (*n*=1)**	**CCEA with extending operator (*n*=2)**
ZDT1	0.1299	0.1376	0.1354
ZDT2	0.1312	0.1274	0.1376
ZDT3	0.2469	0.2140	0.2129
ZDT4	0.1267	0.1339	0.1358
ZDT6	0.1373	0.1246	0.1307
FON	0.8289	0.1901	0.1544
KUR	0.6542	0.6589	0.6703
TLK	1.1074	1.1074	1.1125
DTL2	0.1255	0.1214	0.1208

Table 3.6. Median maximum spread of CCEA with/without the extending operator

Problem	**CCEA without extending operator**	**CCEA with extending operator (*n*=1)**	**CCEA with extending operator (*n*=2)**
ZDT1	0.9931	0.9935	0.9947
ZDT2	0.9989	0.9988	0.9990
ZDT3	0.9973	0.9981	0.9978
ZDT4	0.9358	0.9355	0.9352
ZDT6	0.9992	0.9992	0.9992
FON	0.7202	0.7742	0.8577
KUR	0.9975	0.9981	0.9964
TLK	0.9826	0.9830	0.9830
DTL2	0.9957	0.9971	0.9977

3.4.4 Simulation Results of DCCEA

The test environment for DCCEA consists of 11 PCs in a campus LAN. Table 3.7 gives the configuration of the 11 PCs, e.g., the server of the system runs on the PIV 1600/512 while the peers are run on other PCs. Since the test problems of ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6 have a large number of decision variables, they are used here to test the capability of DCCEA in accelerating the executions in multiobjective optimization. The parameters configuration of the DCCEA is listed in Table 3.8.

Table 3.7. The running environment of DCCEA

PC	Configuration CPU (MHz)/RAM (MB)
1	PIV 1600/512
2	PIII 800/ 512
3	PIII 800/ 512
4	PIII 800/ 256
5	PIII 933/384
6	PIII 933/128
7	PIV 1300/ 128
8	PIV 1300/ 128
9	PIII 933/ 512
10	PIII 933/ 512
11	PIII 933/256

Table 3.8. The parameters of DCCEA

Populations	Subpopulation size 20; archive size 100
Chromosome length	30 bits for each variable
Selection	Binary tournament selection
Crossover method	Uniform crossover
Crossover rate	0.8
Mutation method	Bit-flip mutation
Mutation rate	$2/L$, where L is the chromosome length
Number of evaluations	120,000
Exchange interval	5 generations
Synchronization interval	10 generations

To minimize bias in the simulations, 30 independent runs are performed with random initial populations. The median runtime of the 30 runs is listed in Table 3.9 and is visualized in Fig. 3.10. It can be seen that the median runtime goes down as the number of peers is increased. In the case of ZDT1, the median runtime for 5 peers (each peer with 6 subpopulations) is 109 seconds, which is about one third of the 270 seconds used by 1 peer (each peer with 30 subpopulations). The results also show that 5 peers are enough for the acceleration of runtime in these problems. When there are more than 5 peers, the increment of communication cost counteracts the reduction of computational cost for each peer and the saturation of acceleration is nearly achieved.

The four median metrics of the 30 simulation runs are summarized in Fig. 3.11. It can be seen that the median metrics have no distinct change in spite of some small fluctuations on the curve for the five test problems as the number of peers is increased. This shows that the DCCEA can effectively reduce the runtime while achieving similar performances as the number of peers is increased.

Table 3.9. Median runtime of DCCEA with respect to the number of peers (second)

Number of peers	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
1	270	242	189.5	209	138
2	177.5	142.5	128.5	170	137
3	134	121.5	101	142	124
4	120	109.5	97	139	121
5	109	90	88	134	121
6	96	80	67	123	108
7	94	73	68.5	111	110
8	80	74	65	115	109.5
9	78	72	64	114	109.5
10	78	76	68	115	110.5

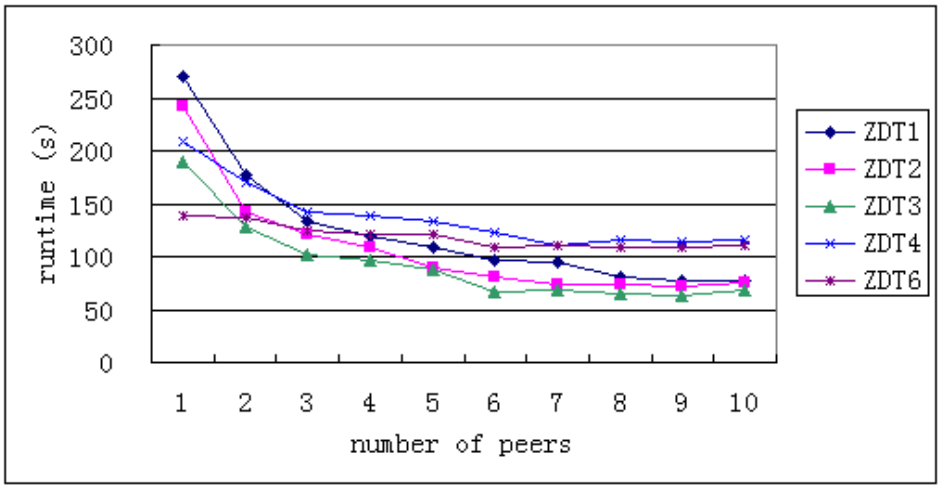
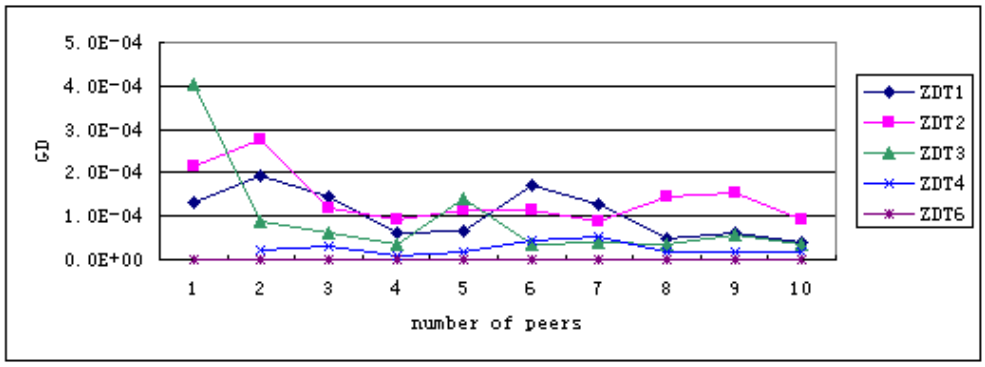
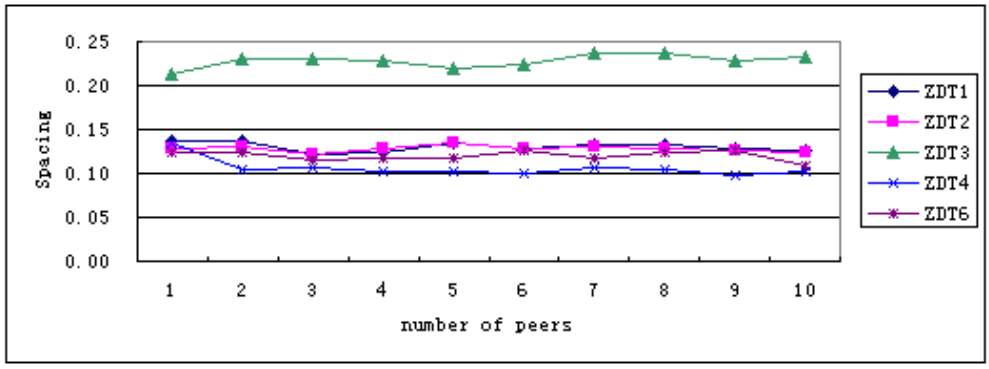


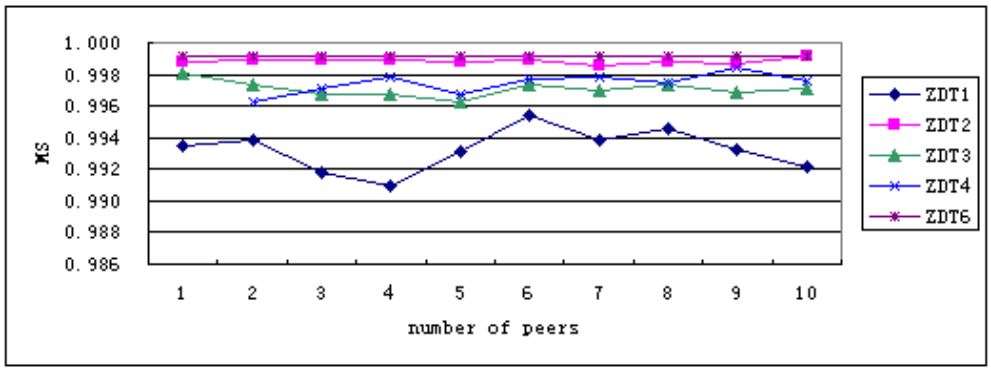
Fig. 3.10. Median runtime of DCCEA with respect to the number of peers



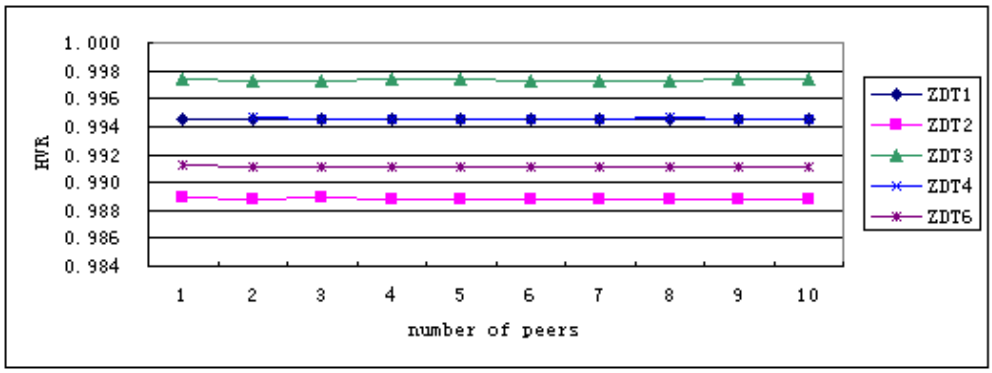
(a)



(b)



(c)



(d)

Fig. 3.11. Median metrics of DCCEA with respect to the number of peers

3.5. Conclusions

This chapter has proposed a cooperative coevolutionary algorithm that incorporates the coevolutionary mechanism by co-evolving the solution set with a number of subpopulations in a cooperative way. Incorporated with various features like archiving, dynamic sharing and extending operator, the CCEA is capable of maintaining search diversity in the evolution and uniformly distributing the solutions along the Pareto front. The extensive quantitative comparisons of various MOEAs on test problems show that CCEA has the best overall performance in endowing the non-dominated solutions with good convergence and uniform distribution. Numerous simulations have been performed to illustrate effectiveness of the proposed extending operator in improving the smoothness and maximum spread of the non-dominated solutions.

Exploiting the inherent parallelism in cooperative coevolution, a distributed CCEA paradigm has been implemented on a Java-based distributed system named Paladin-DEC to reduce the runtime by sharing the computational workload among various networked computers. The computational results show that DCCEA can dramatically reduce the runtime without sacrificing the performance of CCEA as the number of peers increases.

Chapter 4

Enhanced Distribution and Exploration for Multiobjective Optimization

4.1. Introduction

The performance of MOEAs is greatly affected by the parameters. Evolutionary algorithms are intrinsically dynamic and adaptive. The adaptation of parameters during the runtime is more consistent to the general evolutionary idea and has shown better performances over constant parameters (Bäck 1993, 1996; Fogarty 1989; Ochoa 1999; Thierens 2002). Eiben et al. (1999) classified the types of adaptation into dynamic parameter control, adaptive parameter control, and self-adaptive parameter control. The dynamic parameter control typically alters the parameters based on a deterministically rule without any feedback. Fogarty (1989) experimentally studied a dynamical mutation rate control for genetic algorithms and proposed to use a schedule that decreases exponentially over the number of generations. The adaptive parameter control modifies the parameter values when there is some form of feedback from the search that is used to determine the direction and/or magnitude of the change to the parameters. The assignment of the value of the parameters may involve credit assignment, and the action of the EA may determine whether or not the new value

persists or propagates throughout the population. The self-adaptive parameter control encodes the parameters in the chromosome and evolves these parameters during the run. The better values of these encoded parameters lead to better individuals and in turn are more likely to survive and propagate. Self-adaptation has been successfully applied in evolutionary strategy and evolutionary programming. Bäck and Schütz (1996) designed a self-adaptive scheme for binary strings following the principles from the continuous domain.

To maintain the diversity of solutions, many researchers put much effort on this issue and several approaches were proposed. The technique of niche sharing by means of a sharing function is often implemented in MOEAs (Goldberg 1989a; Fonseca and Fleming 1993, 1995b). The niche sharing sums the crowding effects of individuals in a neighborhood. Knowles and Corne (2000) used a hyper grid scheme in the Pareto archived evolution strategy (PAES). The hyper grid divides the normalized objective space into hyper boxes and every individual is given an attribute that indicates the number of solutions sharing the same box. Deb et al. (2002a) proposed the crowding distance in the non-dominated sorting genetic algorithm II (NSGA II). The crowding distance is an estimate of the size of the largest cube enclosing a single solution without any other point in the population and this is used to estimate the density of solutions surrounding a particular individual. This measure is given as the average distance of two points on either side of the selected solution along each of the objectives. Zitzler et al. (2001) used the density mechanism in the strength Pareto evolutionary algorithm 2 (SPEA2). The density estimation is adapted from $k \cdot th$ nearest neighbor method and it is given by the inverse of the $k \cdot th$ distance.

This chapter presents two features to address the objectives of MOPs, (1) minimizing the distance between the solution set and true Pareto front, (2) distributing the solutions evenly, and (3) maximizing the spread of solution set. The first feature is an adaptive mutation operator (AMO). The mutation rate of AMO is adapted with time along the entire evolution process to adjust the exploration and exploitation effects of mutation operator. The second is an enhanced exploration strategy (EES) which maintains diversity and preserves good solutions in the evolving population and extends more attention to the growth of solutions in less populated areas.

Section 4.2 describes the AMO and EES. The comparative studies are performed with some well-known mutation operators, diversity operators, and MOEAs in section 4.3. Conclusions are drawn in section 4.4.

4.2. Two New Features for Multiobjective Evolutionary Algorithms

4.2.1 Adaptive Mutation Operator (AMO)

In this section, an adaptive mutation operator (AMO) is introduced. The AMO is a variant of the simple bit-flip mutation operator and unique in two aspects. Firstly, the manner in which the mutation operation is carried out on the chromosome is different. This will be elaborated later in the section. Secondly, the mutation rate of AMO is adapted with time along the entire evolution process. In brief, the AMO is implemented for three objectives.

i. Providing the possibility of exploration to produce new structures not previously tested

ii. Providing the probability of re-introducing binary bit values lost through the selection process

iii. Performing local fine-tuning in the later stage of evolution and to achieve better convergence.

For the first objective, consider a minimization problem where m decision variables must be optimized. By using a thirty bit binary representation for potential solutions, there is a total of 2^{30m} possible binary structures or chromosomes! Hence, it is difficult if not impossible, for any MOEA with fixed population size to maintain all possible binary bit combinations at any one time. By changing the bit values according to some mutation probability, the mutation operator acts as a potential source of producing the missing structures so that the evolution process is not trapped in any local minimal. With small mutation rates, the individuals produced by mutation will not vary much from the parent in terms of the chromosome structure. Intuitively, it will be very difficult to escape local traps. However, simply increasing the mutation rate cannot solve this problem. With increased mutation rates, the probability of disrupting sub-structures within the chromosome that are responsible for good candidate solutions, is increased.

A simple and effective way to perform exploration while minimizing the disruption of good substructures within the chromosome is to mutate a specific part of the chromosome rather than the entire binary structure. More specifically, each of the decision variable encoded in the chromosome is allocated equal probability of undergoing the mutation operation. During this mutation operation, the bits of selected decision variable will be subjected to bit-flip with probability, $am\_rate(n)$. AMO

operation for a single chromosome is shown in Fig. 4.1 where *prob* is probability of the decision variable being selected and $am\_rate(n)$ is the probability of the bit-flip operation. If *prob* is set as $1/\text{var\_}num$ where var_*num* is the number of decision variables encoded in a single chromosome, on average, the AMO will perform the bit-flip operation on one decision variable for every chromosome. Thus, the AMO allows mutated individual retaining most of the substructures contributing to the chromosomes fitness.

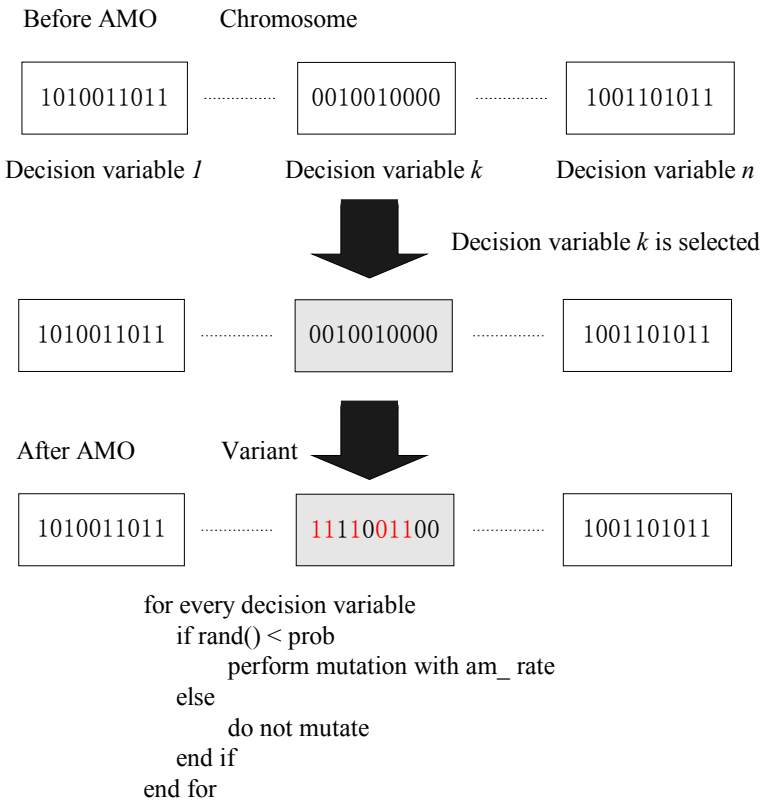


Fig. 4.1. AMO operation

Holland had presented the idea of applying the mutation operator with a time-dependent and deterministic rate schedule that reduces the mutation rate toward zero in (Holland, 1992). Some researchers had observed that by varying mutation rate, the

performance of the algorithm could be improved. Fogarty (1989) used a varying mutation rate, demonstrating that a mutation rate that decreases exponentially over generations has superior performance. Davis (1989) adapted the operator application probability based in the performance of offspring, i.e. the operators that create and cause generation of better offspring are allotted higher probabilities. Bäck and Schütz (1996) had also shown the usefulness of a time-varying mutation rate. Despite these reported success, most recent well-known MOEAs such as SPEA2, PESA, PAES, NSGAII still employ static mutation operators.

The AMO adapts the mutation rate to maintain a balance between the introduction of diversity and local fine-tuning. The mutation rate will start off with a high value to produce a diverse set of solutions for an effective genetic exploration search. This value will then decrease as a function of time or generation number to meet the exploitation requirement of local fine-tuning. The mutation rate for this operation is given by

$$am\_rate(n) = \begin{cases} a\left[1-\left(\frac{n}{genNum}\right)^2\right]+b & 0 \le n \le \alpha \\ a\left[0.1\left(\frac{n-genNum}{genNum}\right)^2\right]+b & \alpha \le n \le genNum \end{cases} \tag{4.1}$$

where n is the current generation number of the evolution process, *genNum* is the maximal generation number. Fig. 4.2 shows the adaptation of mutation rate along the evolution when a is 0.8, and b is 1/(10*30). Two distinct regions can be observed, the exploration region between 0.8~0.753 and the exploitation region between 0.048~0.003. Different from many other adaptive mutation operators where mutation rate decreases gradually along the evolution, AMO pays its attention to searching new

strings in the initial stage and then quickly to improving them in the later stage. No time is spent in exploring the immediate region between the exploration and exploitation region while AMO adapts the mutation rate according to a smooth curve inside each region.

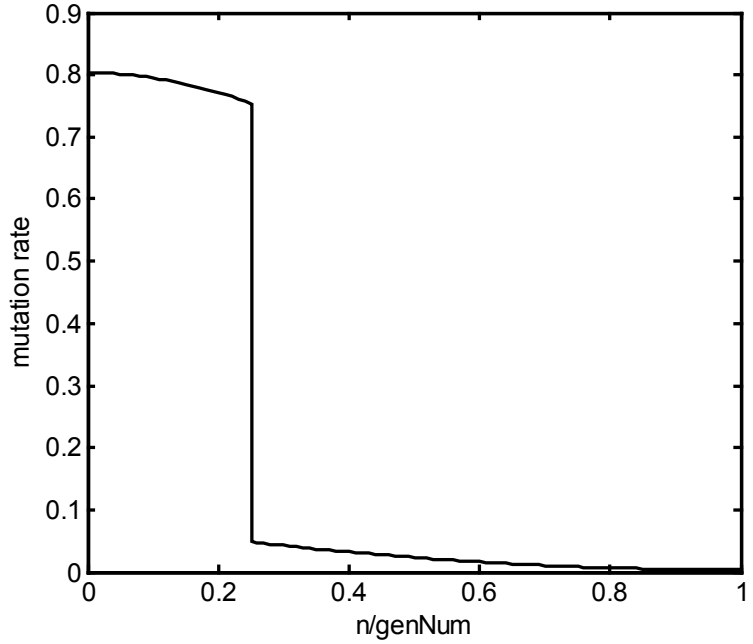


Fig. 4.2. Adaptive mutation rate in AMO

4.2.2 Enhanced Exploration Strategy (EES)

In this section, the enhanced exploration strategy (EES) is presented. The EES is an online population distribution scheme that maintains diversity and preserves non-dominated solutions together in the mating population. In addition, it improves distribution of solutions by encouraging the growth of individuals in less populated areas.

The approximation of the Pareto optimal front requires the MOEA to perform a multi-directional search simultaneously to discover multiple, widely different solutions and

this requires a substantial amount of diversity in the evolving population. According to Mahfoud (1995), simple elitist EA tends to converge towards a single solution and often loses solutions due to the effects of selections pressure, selection noise, drifting, and operator disruption. Many methods such as, sharing (Goldberg and Richardson 1987), restricted mating (Deb and Goldberg 1989) and crowding (De Jong 1975), have been proposed over the years to deal with this problem.

In this chapter, the niche sharing discussed in Section 2.4 is used to maintain the diversity where the objective space is normalized and the sharing distance is set as $\sigma_{\text{share}} = 1/archive\_size$. The niche count will be used in the selection and archive updating.

The flow chart of EES is shown in Fig. 4.3. At every generation, a certain number of individuals will be tournament selected from the archive to form the population called exp_pop and the selection criterion is based purely on the niche count. Simple bit-flip mutation is performed on exp_pop with mutation probability P_{exp} and the purpose of the entire process is to promote the growth of solutions in less populated areas. P_{exp} is set either as $1/chromosome\_length$ or $1/bit\_number\_per\_variable$ depending on the test problem. The number of individuals selected for exp_pop is dynamic and it is given by,

$$Num\_Explore = c(1 - epr^2) + d \tag{4.2}$$

where $epr(n)$ is the evolution progress rate. Evolution progress rate is developed from progress ratio, a performance metric defined as the ratio between the number of non-dominated individuals at generation n dominated any non-dominated individuals at

generation $(n-1)$ and the total number of non-dominated individuals at generation n (Tan et al. 2001). The evolution progress rate, $epr(n)$, is defined as the ratio of the number of new non-dominated solutions discovered in generation *n,* $new\_nondomSol(n)$, to the total number of non-dominated solutions in generation *n,* $total\_nondomSol(n)$.

$$epr(n) = \frac{number\ of\ new\_nondomSol(n)}{number\ of\ total\_nondomSol(n)} \tag{4.3}$$

The set of new non-dominated individuals discovered at each generation is basically composed of individuals that dominate the non-dominated individuals of the previous generation and individuals that contribute to the diversity of the solution set. The rationale behind the use an adaptive number of individuals selected for the exploration process is intuitive. When $epr(n)$ is low, it means that either the generated Pareto front is approaching the true front or the evolution process is not discovering new solutions and more resources are required to perform exploration in the less populated areas. When $epr(n)$ is high, it means that the new solutions are being discovered and requirement for resources to perform exploration can be reduced.

At the same time, individuals are being selected to a mating pool named mat_pop through the tournament selection of the combination of archive and $population(n)$ where $population(n)$ is the evolving population. The selection criterion in this case is based on Pareto based rank and the niche count will be used in the event of a tie. The population size of mat_pop is dynamic and given by Pop_size - $Num\_Explore$ where Pop_size is the population size of the evolving population. The mat_pop will then be subjected to genetic operations such as crossover and

mutation. After the genetic operations are carried out, exp_pop and mat_pop will be combined to form $population(n+1)$. The settings of c and d adopted in this chapter is 10 and 20 respectively.

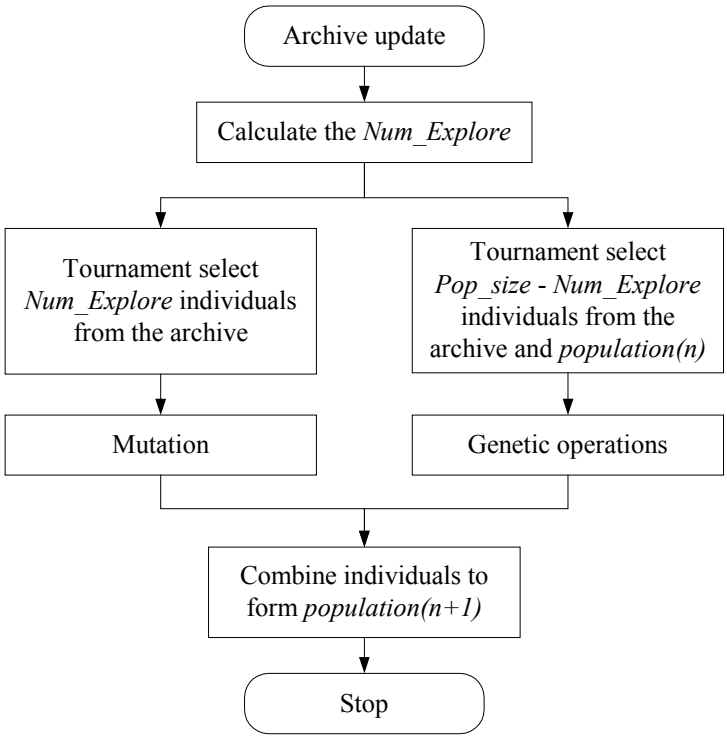


Fig. 4.3. The flow chart of EES

4.3. Comparative Study

This section will start with the Section 4.3.1 that describes three performance metrics used in the comparisons. Then the test problems are introduced in the Section 4.3.2. Three comparisons will be performed to evaluate the performance of the proposed features. The various mutation operators are surveyed and AMO is compared against the selected mutation operators in the Section 4.3.3. The diversity operators are overviewed and EES is compared against these diversity operators in the Section 4.3.4.

In Section 4.3.5, the performance comparison among a common MOEA incorporating AMO and EES and various well-known algorithms will be made.

4.3.1. Performance Metrics

Three different quantitative performance measures for MO optimization are used. The first metric is the generational distance (GD), which measures how "far" the solution set is from the true Pareto front. The metric of spacing (S) measures how "evenly" members in the solution set distribute. Zitzler (2000) defined a metric of maximum spread (MS) to measure how well the true Pareto front is covered by the solution set. For the definition of these metrics, please refer to Section 3.4.1.

4.3.2. The Test Problems

Three test problems are used in the case study. The problems, *ZDT4, ZDT6* and *FON*, can be referred to Section 3.4.2.

4.3.3. Effects of AMO

In this Section, the performance of AMO and the influence of parameter variations are investigated. This section will start with a short discussion on the bit-flip mutation and fuzzy boundary local perturbation.

4.3.3.1. Mutation operators

There are different opinions on the motivation behind its use in EA. Some researchers think that the mutation operator plays the role of ensuring that the crossover operator has a full range of genetic materials (Holland 1992), while some used it as a hill-climbing mechanism (Knowles and Corne 2000). Two mutation operators are discussed below.

1) Bit-flip mutation: Bit-flip mutation simply means the flipping of the chromosome bits. For every bit, the probability of being mutation is given by a predetermined value, the mutation rate. This mutation rate remains constant throughout the evolution process.

2) Fuzzy boundary local perturbation (FBLP): Tan et al. (2001) used the FBLP in place of simple bit-flip mutation to produce the required number of individuals in IMOEA with dynamic population sizing. Unlike bit flip mutation, the perturbation rate for FBLP varies according to the significance of the genes in the chromosome. Consider *n* genes concatenated in a chromosome to represent an optimizing parameter. A probability set $P = \{p_i \mid i = 1, \cdots n\}$ that indicates the perturbation probability for each gene, can be defined

$$p_i = \begin{cases} b\left[2\left(\frac{i-1}{n-1}\right)^2 + a\right] & , \ 1 \le i \le \beta \\ b\left[1-2\left(\frac{i-n}{n-1}\right)^2 + a\right] & , \beta < i \le n \end{cases} \tag{4.4}$$

The perturbation rate decreases with the increasing significance of the encoded bit. Hence the perturbed child is very likely to lie within the immediate neighborhood of the parent. FBLP is thus capable of local fine-tuning.

4.3.3.2. Comparison of AMO

The AMO is compared against FBLP and three bit-flip mutation operators with different settings. The parameter configurations in the different mutation operators and the different cases are shown in Table 4.1 and Table 4.2 respectively.

Table 4.1. Parameter setting for the mutation operators

Chromosome	Binary coding. 30 bits per decision variable.
Populations	Population size 100; Archive (or secondary population) size 100.
Selection	Binary tournament selection
Crossover operator	Uniform crossover
Crossover rate	0.8
Ranking scheme	Scheme of Fonseca and Fleming
Diversity operator	Niche count with radius 0.01 in the normalized objective space
Generation number	1000

Table 4.2. Different cases for the AMO evaluation

Index	Case	Description
1	AMO	AMO with $b = PM$
2	N1	Bit-flip with mutation rate $PM/2$
3	N2	Bit-flip with mutation rate PM
4	N3	Bit-flip with mutation rate $2 \cdot PM$
5	FBLP	$ab = PM/2, b = PM, \beta = bit\_num\_per\_\mathrm{var}iable/2$

PM is defined as $1/chromosome\_length$ for *ZDT4* and *ZDT6* and $1/bit\_number\_per\_\mathrm{var}iable$ for *FON*.

Table 4.3. Median values of GD, S and MS for different mutation operators

		Mutation operator				
		AMO	FBLP	N1	N2	N3
ZDT4	GD	0.7681	0.8778	0.7868	0.8142	1.4601
	S	0.6481	0.3541	0.2595	0.7463	0.7831
	MS	0.7444	0.7533	0.7572	0.7408	0.4207
ZDT6	GD	4.87e-7	0.8657	0.5471	1.5886	2.8208
	S	2.3443	1.3399	1.7457	1.1108	1.1910
	MS	0.9992	0.7042	0.7545	0.7060	0.7047
FON	GD	0.0030	0.0031	0.0031	0.0146	0.0492
	S	2.4625	1.3672	0.9318	0.8072	0.7589
	MS	0.5858	0.4845	0.4791	0.5620	0.6773

In the experiment, 30 runs are performed for each case on each test problem so as to study the statistical performance. The median of 30 runs on the three performance metrics is listed in Table 4.3. AMO displays the best generational distance for this problem. AMO is the only operator that enables the algorithm to converge upon the Pareto front of ZDT6. In addition, AMO is competitive in the spread. However, it seemed that the good performances of AMO in the spread and generation distance are achieved at the expense of spacing. This is probably due to AMO's emphasis on exploitation in the later stage of evolution. As a result, the AMO is unable to bridge the gaps between the extreme end solutions discovered during the initial exploratory phase.

4.3.3.3. Effects of Parameter *prob*

The effects of various *prob* settings are examined in Table 4.4. The purpose is to prove that the underlying idea of AMO to maintain a balance between preservation and disruption of chromosomes by selective mutation of decision variables can improve the performance of the algorithm. Similarly, 30 runs are performed for each setting on each test problem.

Table 4.4. Median values of GD, S and MS for different AMO parameter *prob*

		Parameter Settings: *prob*			
		1/ var_ *num*	0.25	0.5	0.75
ZDT4	GD	0.7681	0.7996	0.8080	0.7927
	S	0.6481	0.6627	0.7194	0.7129
	MS	0.7444	0.7158	0.7180	0.7384
ZDT6	GD	4.87e-7	4.91e-7	5.02e-7	1.0609
	S	2.3443	2.5039	3.1710	0.9033
	MS	0.9992	0.9992	0.9992	0.7047
FON	GD	0.0030	0.0034	0.0208	0.0415
	S	2.4625	2.3488	0.8112	0.7131
	MS	0.5858	0.6064	0.6638	0.6999

Note that as prob is increased, the behavior of AMO will approach that of bit-flip mutation operator albeit the changing mutation rate. It can be observed from table 6 that the metric of generation distance increases with increasing prob. This is most probably due to the fact that increasing *prob* would correspond to the disruption of more genes.

4.3.4. Effects of EES

In this section, the individual effects of EES are investigated in a fashion similar to that in Section 4.3.3. A short review of four diversity mechanisms, sharing, hyper grid, crowding and density estimation is given in this section. These diversity operators have been implemented in MOEA and together with the method of sharing. They will be references for comparing EES.

4.3.4.1. Diversity Operators

Diversity needs to be maintained in the evolving population in order for the MOEAs to discover multiple, widely different solutions. The diversity operators used in the case study include niche sharing, grid mapping, crowding, and density estimation described in Section 2.4.

4.3.4.2. Comparison of EES

The three performance measures introduced in Section 4.3.1 are used to provide a quantitative evaluation of the performance of the various operators. The three problems introduced in Section 4.3.2 are used to compare the performance of EES against the selected diversity mechanisms. The indices of the diversity operators are shown in Table 4.5. The parameters for these diversity operators are shown in Table 4.6.

Table 4.5. Description of different diversity operators

Index	Diversity operator	Description
1	ESS	Niche radius 0.01 in the normalized objective space
2	Niche sharing	Niche radius 0.01 in the normalized objective space

3	Grid mapping	Using normalized objective space
4	Crowding	Using normalized objective space
5	Density estimation	Using normalized objective space

Table 4.6. Parameter setting of different diversity operators

Chromosome	Binary coding. 30 bits per decision variable.
Populations	Population size 100; Archive (or secondary population) size 100.
Selection	Binary tournament selection
Crossover operator	Uniform crossover
Crossover rate	0.8
Mutation operator	Bit-flip mutation
Mutation rate	*PM*
Ranking scheme	Fonseca and Fleming Pareto Dominance Ranking Scheme
Hyper-grid size	2^3 per dimension for DTL2. 2^5 per dimension for other problems.
Generation number	1000

The median of 30 runs on the three metrics is listed in Table 4.7. With respect to the metric of generation distance, the algorithm incorporated with EES is clearly the best in the test problems. This is particularly evident in the test problem of ZDT6 and FON. ZDT4 proved to be the most difficult problem for all algorithms. However, EES still produces good performance in all three metrics with respect to the other diversity operators on this problem.

Table 4.7. Median values of GD, S and MS for different diversity operators

		Diversity operator				
		EES	Niche	Grid	Crowd	Density
ZDT4	GD	0.7652	0.8142	1.0008	0.7832	0.7993
	S	0.3173	0.7463	0.6567	0.2506	1.3513
	MS	0.7610	0.7408	0.7235	0.7366	0.7403
ZDT6	GD	5.05e-7	1.5886	1.5984	1.6012	1.6222
	S	0.1734	1.1108	1.1051	1.1444	1.1119
	MS	0.9992	0.7060	0.7051	0.7061	0.7043
FON	GD	0.0022	0.0146	0.0141	0.0146	0.0142
	S	0.2252	0.8072	0.9006	0.8077	0.8541
	MS	0.7732	0.7060	0.7051	0.7061	0.7043

It is also obvious that the incorporation of EES improves greatly the distribution and spread of solution along the Pareto front for all test problems. EES is particularly outstanding in the metric of spacing in test problem of *ZDT6* and *FON*. In addition, EES has the best performance in the area of maximum spread for all test problems.

Table 4.8 shows that the performance of EES with different *d* settings does not vary a lot over the test problems. This observation implies that the EES will be able to perform well against the various diversity operators despite the different settings. More importantly, it also shows that the EES is insensitive to parameter changes.

Table 4.8. Median values of GD, S and MS for different EES parameter *d*

		EES Parameter Settings: *d*			
		20	25	30	40
ZDT4	GD	0.7652	0.7712	0.7688	0.7804
	S	0.3173	0.3185	0.3224	0.3167
	MS	0.7610	0.7590	0.7590	0.7557
ZDT6	GD	5.05e-7	5.10e-7	5.13e-7	4.94e-7
	S	0.1734	1.4231	0.1660	0.1770
	MS	0.9992	0.9992	0.9992	0.9992
FON	GD	0.0022	0.0020	0.0021	0.0021
	S	0.2252	0.2273	0.2379	0.2211
	MS	0.7732	0.8053	0.7857	0.7947

4.3.5. Effects of both AMO and EES

The AMO and EES are incorporated into a general MOEA paradigm that uses binary coding, binary tournament selection, uniform crossover, and Fonseca and Fleming's ranking scheme. This algorithm is called ALG in this chapter and will be compared with five recent well-known algorithms to validate the effectiveness of AMO and EES. The five algorithms are PAES, PESA, NSGAII, SPEA2 and IMOEA that have been overviewed in Section 2.5. The indices of the different algorithms are listed in Table 4.9. The parameter settings in each algorithm are listed in Table 4.10.

Table 4.9. Indices of the different MOEAs

Index	1	2	3	4	5	6
Algorithm	ALG (AMO+EES)	PAES	PESA	NSGA II	SPEA 2	IMOEA

Table 4.10. Parameter setting of different algorithms

Chromosome length	Binary coding, 30 bits for each variable.
Populations	Population size 1 in PAES; population size 100 in ALG, PESA, NSGAII, SPEA2; initial population size 20, maximum population size 100 in IMOEA. Archive (or secondary population) size is 100 for all algorithms.
Selection	Binary tournament
Crossover operator	Uniform crossover
Crossover rate	0.8
Mutation operator	AMO in ALG; FBLP in IMOEA; bit-flip mutation in others.
Mutation rate	*PM*
Ranking	Scheme of Fonseca and Fleming
Hyper-grid size	2^5 per dimension.
Niche radius	$1/Archive\_Size$ for ALG; Dynamic sharing in IMOEA
Generation number	1000

Thirty independent runs are performed on each of the test functions so as to obtain statistical information such as consistency and robustness of the algorithms. Figs 4.4-4.6 visualize the simulation results of the algorithms with respect to the various metrics in the box plot format. Although the previous investigation of AMO and EES in Section 4.3.3 and Section 4.3.4 show that the individual effects of either feature are not enough to allow the algorithm overcome the local traps of ZDT4 and the large spread of FONs' tradeoff, each have showed their own distinct advantage over their counterpart operators. While AMO have the ability drive the evolution towards the Pareto front and to find points in unexplored regions, it lacks some form of mechanism to guide its operation. This results in the subsequent gaps observed in the discovered Pareto front. The mechanism to guide the exploration of AMO comes in the form of EES. Likewise EES may have shown the ability to locate these gaps, it is unable to escape the local optimum trap of ZDT4 or maintain a diverse solution set in FON. Thus it is not surprising that the ALG produces better performance when these two features are incorporated together.

ZDT4 proves to be the most difficult problem faced by the algorithms since no algorithm, except ALG, is able to deal with multi-modality effectively. This is reflected in the performance metric of generation distance. In addition, the ALG also chalked up outstanding results in the metric of spread and distribution. The biased search space of ZDT6 is designed to make it difficult for the algorithms to evolve a well-distributed front. In this respect, ALG is still able to give outstanding results in terms of the distribution of results. This is probably because of EES. Otherwise, ALG performance in the aspects of generation distance and spread is well matched by SPEA2 and NSGAII. The challenge of test function FON is to find and maintain the

entire Pareto front uniformly. With the exception of the ALG, the algorithms found it difficult to find a good spread and distribution.

For all test problems, ALG responds well to the challenges of the different difficulties. The ALG performs consistently well in the distribution of solutions along the Pareto front. This is even so for the test problems of ZDT6 and FON that are designed to challenge the algorithm's ability to maintain the Pareto front. The performance of ALG with respect to generational distance is also outstanding in all problems. This demonstrates the ALG's ability to converge upon the Pareto front regardless of problems such as discontinuities, convexities and non-uniformities. It also shows no problems in coping with local traps and this is reflected by its performance in the test problem ZDT4. The ALG ability to discover a diverse solution set on the Pareto frontier is demonstrated and this is most evident in the test problem of FON.

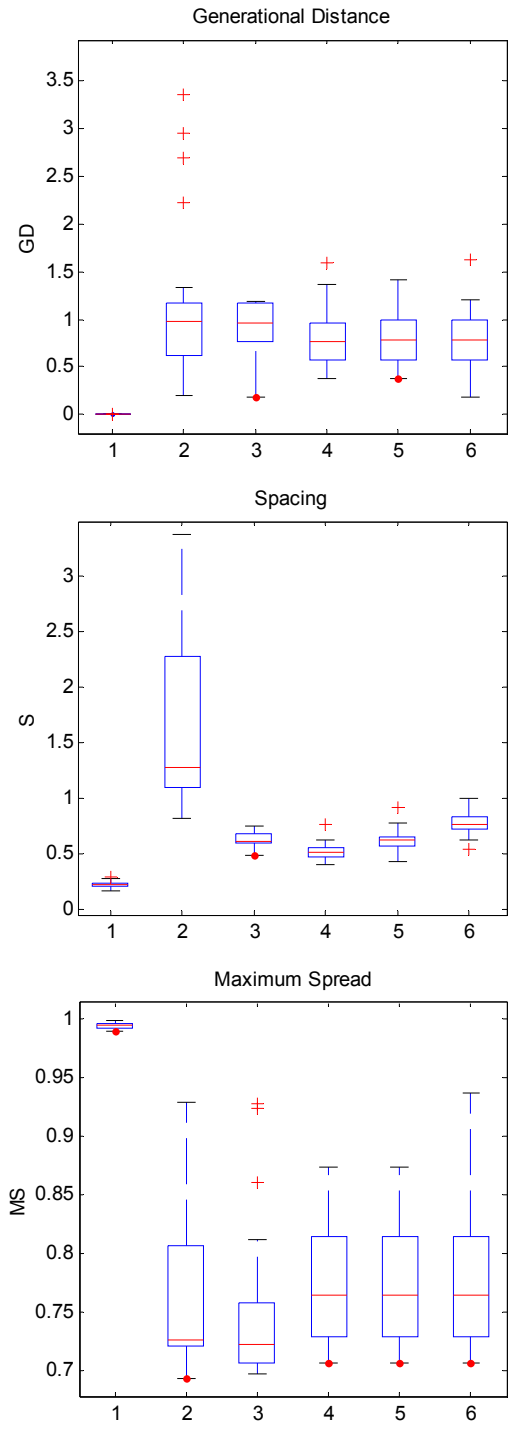
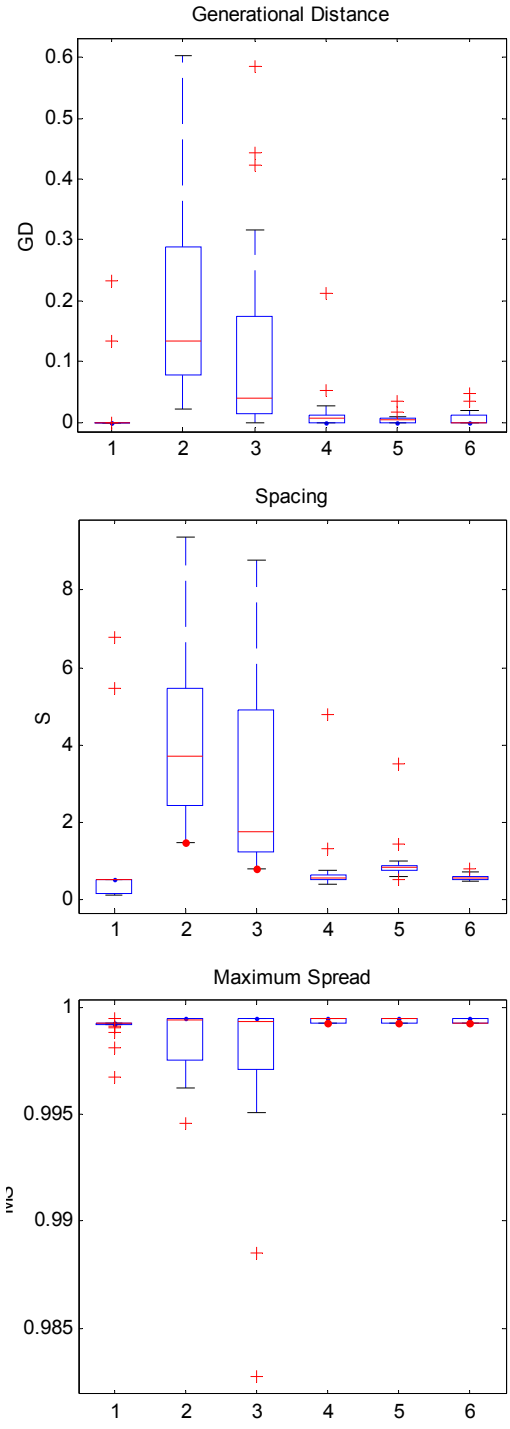
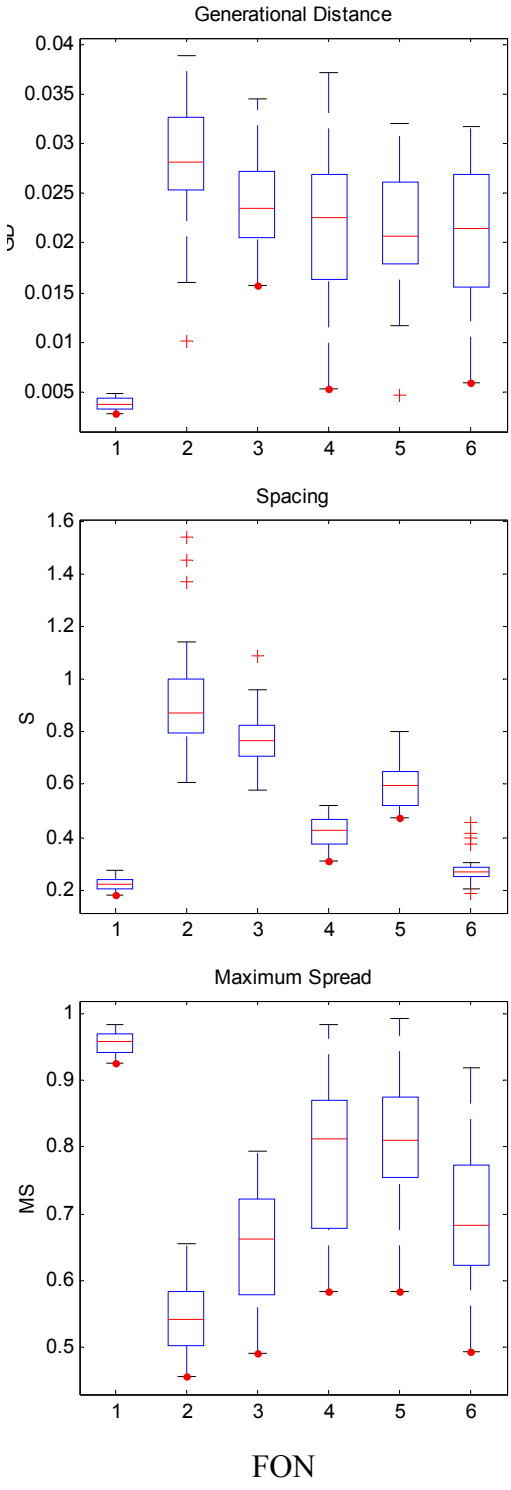


Fig. 4.4. Simulation results for ZDT4



ZDT6

Fig. 4.5. Simulation results for ZDT6



FON

Fig. 4.6. Simulation results for FON

4.4. Conclusions

This chapter presents the enhanced exploration strategy that maintains diversity and non-dominated solutions in the evolving population while encouraging the exploration towards the direction of less populated areas. This achieves better discovery of gaps in the discovered frontier as well as better convergence. An adaptive mutation operator that plays the role of producing new genetic structures is also presented. This AMO adapts the mutation rate to maintain a balance between the introduction of diversity and local fine-tuning.

A comparative study between the proposed features and various mutation operators, diversity operators, existing multiobjective evolutionary algorithms and are carried out on three test problems. Simulations are carried out to examine the effects of AMO and EES with respect to selected mutation and diversity operators respectively. AMO and EES have showed to be competitive if not better than their counterparts and have their own specific contribution. Simulations results also show that the algorithm incorporated with AMO and EES is capable of discovering and distributing non-dominated solutions along the Pareto front. The combined effects of AMO and EES enabled the algorithm to perform well in breaking out of local traps and maintaining diversity in the solution set. The combined effects of these two features allow the algorithm to find a good, well-distributed and diverse solution set along the Pareto front.

Chapter 5

Conclusions and Future Works

5.1 Conclusions

In this thesis, a cooperative co-evolution mechanism is applied in the multiobjective optimization. Exploiting the inherent parallelism in cooperative co-evolution, the algorithm is formulated into a distributed computing structure to reduce the runtime by sharing the computational workload among various networked computers. To improve the performance of multiobjective evolutionary algorithms, an adaptive mutation operator and an enhanced exploration strategy are proposed.

The cooperative co-evolutionary algorithm adopts the mechanism of coevolution by decomposing a complex MO optimization problem via a number of subpopulations co-evolving for the set of Pareto-optimal solutions in a cooperative way. Incorporated with various features like archiving, dynamic sharing and extending operator, the CCEA is capable of maintaining solution diversity and distributing the solutions uniformly along the Pareto front. The extensive quantitative comparisons of various MOEAs on nine benchmark problems show that CCEA has the best overall performance in endowing the non-dominated solution set with good convergence and uniform distribution. Many simulations have been performed to illustrate the effectiveness of the proposed extending operator in improving the smoothness and

maximum spread of the non-dominated solution set. Exploiting the inherent parallelism in cooperative co-evolution, a distributed CCEA paradigm has been implemented on a Java-based distributed system named Paladin-DEC to reduce the runtime by sharing the computational workload among various networked computers. The computational results show that DCCEA can reduce the runtime effectively without sacrificing the performance as the number of peer computers increases.

The adaptive mutation operator adapts the mutation rate to maintain a balance between the introduction of diversity and local fine-tuning. The enhanced exploration strategy maintains solution diversity and preserves non-dominated solutions in the evolving population while encouraging the exploration towards less populated areas. This achieves better discovery of gaps in the discovered Pareto front as well as better convergence. A comparative study is carried out to examine the effects of AMO and EES with respect to selected mutation and diversity operators respectively. AMO and EES have shown to be competitive if not better than their counterparts and have their own specific contribution. Simulations results also show that the algorithm incorporated with AMO and EES performs well in breaking out of local traps and finding a good, well-distributed and diverse solution set along the Pareto front.

5.2 Future works

Eiben et al. (1999) classified the types of adaptation in evolutionary algorithms into dynamic parameter control, adaptive parameter control, and self-adaptive parameter control. The dynamic parameter control has been considered to adjust the mutation rate in Chapter 4. The adaptive parameter control and self-adaptive parameter control could also be explored for the adjustment of mutation rate in MOEAs. These two types of

parameter control require less *a-prior* knowledge and could have better performance. Moreover, the adaptation mechanism may be studied for switching among several mutation and crossover operators to achieve better performance in MOEAs.

In the aspect of multiobjective search strategy, ways of identifying appropriate MO optimization methods for different problems and different types of decision making are needed. For multiobjective optimization, it is important not only to develop general methods, but also to create algorithms that work well for certain problem types or application areas. Besides, research work in theoretical aspect of MOEAs, such as convergence properties to the global Pareto front and the efficiency in reaching the acceptable optimization goals, are still insufficient. Further research in this area is essential and important.

Most existing MOEAs assume that the vector of exact objective functions can be built accurately to measure all possible solutions in the search space. However, a wide range of uncertainties has to be considered in many real-world optimization problems. Generally, uncertainties in evolutionary optimization can be categorized into three classes: the fitness function is uncertain or noisy; the design variables or the environmental parameters are subject to perturbations or deterministic changes and this issue is often known as the search for robust optimal solutions; the fitness function is time-variant where the optimum of the system is changing with time, which requires a repeated re-optimization or even continuous tracking of the optimum. Handling uncertainties in evolutionary optimization is a very important problem and receiving an increasing interest.

References

1. Angeline, P.J. 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.
2. Bäck, T. Optimal Mutation Rates in Genetic Search. In Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 2-8. Morgan Kaufmann, 1993.
3. Bäck, T. and M. Schütz. Intelligent Mutation Rate Control in Canonical Genetic Algorithms. In Proceedings of the International Symposium on Methodologies for Intelligent Systems, pp. 158-167. 1996.
4. Bentley, P.J. and J.P. Wakefield. Finding Acceptable Solutions in the Pareto-Optimal Range Using Multiobjective Genetic Algorithms. In Proceedings of 2nd On-Line World Conference on Soft Computing in Engineering Design and Manufacturing (WSC2). 1997.
5. Borges, C.C.H. and H.J.C. Barbosa. A Non-Generational Genetic Algorithm for Multiobjective Optimization. In IEEE Congress on Evolutionary Computation 2000, pp 172-179. 2000.
6. Cantú-Paz, E. A Survey of Parallel Genetic Algorithms. Calculateurs Paralleles, Reseaux et Systems Repartis 10(2), pp. 141-171. 1998.
7. Chambers, J.M., W.S. Cleveland, B. Kleiner, and P.A. Turkey. Graphical Methods for Data Analysis. Wadsworth & Brooks/Cole, Pacific CA, 1983.

8. Coello Coello, C.A. An Empirical Study of Evolutionary Techniques for Multiobjective Optimization in Engineering Design. Ph.D. Thesis, Department of Computer Science, Tulane University, USA. 1996.
9. Coello Coello, C.A. An Updated Survey of GA-Based Multiobjective Optimization Techniques. Report No. Lania-RD-98-08, Laboratorio Nacional de Informatica Avanzada (LANIA), Mexico. 1998.
10. Coello Coello, C.A. and G.T. Pulido. A Micro-Genetic Algorithm for Multiobjective Optimization. In First International Conference on Evolutionary Multi-Criterion Optimization, pp. 126-140. Springer-Verlag. 2001.
11. Corne, D.W., J.D. Knowles and M.J. Oates. The Pareto Envelope-Based Selection Algorithm for Multiobjective Optimization. In Proceedings of the Parallel Problem Solving from Nature VI Conference, pp, 839-848. Springer-Verlag. 2000.
12. Corne, D.W., K. Deb, P.J. Fleming and J.D. Knowles. The Good of the Many Outweighs the Good of the One: Evolutionary Multi-Objective Optimization. The Newsletter of the IEEE Neural Networks Society 1(1), pp. 9-13. 2003.
13. De Jong, K.A. An Analysis of the Behavior of a Class of Genetic Adaptive Systems. Ph.D thesis, University of Michigan. 1975.
14. Deb, K. Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems. Journal of Evolutionary Computation 7(3), pp. 205-230. 1999.
15. Deb, K. Multi-Objective Optimization Using Evolutionary Algorithms. New York: John Wiley & Sons. 2001.
16. Deb, K. and D. E. Goldberg. An Investigation of Niche and Species Formation in Genetic Function Optimization. In Proceedings of the Third International Conference on Genetic Algorithms, pp. 42–50. 1989.

17. Deb, K., A. Pratap, S. Agarwal and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), pp. 182-197. 2002a.
18. Deb, K., L. Thiele, M. Laumanns and E. Zitzler. Scalable Multiobjective Optimization Test Problems. In Proceedings of the 2002 Congress on Evolutionary Computation, pp. 825-830. 2002b
19. Eiben, A.E., R. Hinterding and Z. Michalewicz. Parameter Control in Evolutionary Algorithms. IEEE Transactions on Evolutionary Computation 3(2), pp. 124-141. 1999.
20. Fogarty, T. Varying the Probability of Mutation in the Genetic Algorithm. In Proceedings of the Third International Conference on Genetic Algorithms, pp. 104-109. Morgan Kaufmann. 1989.
21. Fonseca, C.M. and P.J. Fleming. Genetic Algorithm for Multiobjective Optimization: Formulation, Discussion and Generalization. In Proceeding of the Fifth International Conference on Genetic Algorithms, pp 416-423. Morgan Kaufmann. 1993.
22. Fonseca, C.M. and P.J. Fleming. An Overview of Evolutionary Algorithms in Multiobjective Optimization. Journal of Evolutionary Computation 3(1), pp. 1-16. 1995a.
23. Fonseca, C.M. and P.J. Fleming. Multiobjective Genetic Algorithm Made Easy: Selection, Sharing and Mating Restriction. In International Conference on Genetic Algorithm in Engineering Systems: Innovations and Application, pp, 12-14. 1995b.
24. Goldberg, D.E. Genetic Algorithms in Search, Optimization, and Machine Learning. Massachusetts: Addison Wesley. 1989a.

25. Goldberg, D.E. Sizing Populations for Serial and Parallel Genetic Algorithms. In Proceedings of the Third International Conference on Genetic Algorithms, pp. 70-79. Morgan Kaufmann. 1989b.
26. Goldberg, D.E. and J. Richardson. Genetic Algorithms with Sharing for Multi-Modal Function Optimization. In Proceedings of the 2nd International Conference on Genetic Algorithms, pp. 41-49. 1987.
27. Hajela, P. and C.Y. Lin. Genetic Search Strategies in Multicriterion Optimal Design. Journal of Structural Optimization 4, pp. 99-107. 1992.
28. Holland, J. H. Adaptation in Natural and Artificial Systems. The MIT Press. 1992.
29. Horn, J. Multicriterion Decision Making. In T. Bäck, D. Fogel, Z. Michalewicz (eds) Handbook of Evolutionary Computation vol. 1, pp. F1.9:1-F1.9:15. IOP Publishing Ltd and Oxford University Press. 1997.
30. Horn, J. and N. Nafpliotis. Multiobjective Optimization Using the Niched Pareto Genetic Algorithm. Technical Report IlliGAl Report 93005, University of Illinois at Urbana-Champaign, USA. 1993.
31. Hwang, C.L. and A.S.M. Masud. Multiple Objective Decision Making - Methods and Applications. Springer-Verlag. 1979.
32. Keerativuttiumrong, N., N. Chaiyaratana and V. Varavithya. Multi-objective Cooperative Coevolutionary Genetic Algorithm. In Parallel Problem Solving from Nature-PPSN VII, pp. 288-297. Springer-Verlag. 2002
33. Knowles, J.D., D.W. Corne. Approximating the Nondominated Front Using Pareto Archived Evolutionary Strategy. Evolutionary Computation 8(2), pp. 149-172. 2000.

34. Khor, E.F., K.C. Tan and T.H. Lee. Tabu-Based Exploratory Evolutionary Algorithm for Effective Multiobjective Optimization. In First Conference on Evolutionary Multi-Criterion Optimization (EMO'01), pp. 344-358. 2001.
35. Kursawe, F. A Variant of Evolution Strategies for Vector Optimization. In Proceedings of the First Conference on Parallel Problem Solving from Nature, pp. 193-197. 1990.
36. Lahanas, M., D. Baltas and N. Zamboglou. A Hybrid Evolutionary Algorithm for Multiobjective Anatomy-Based Dose Optimization in High-dose-rate Brachytherapy. Physics in Medicine and Biology 48, pp.399-415. 2003
37. Lis, J. and A.E. Eiben. A Multi-Sexual Genetic Algorithm for Multiobjective Optimization. In IEEE International Conference on Evolutionary Computation, pp 59-64. 1997.
38. Liu, Y. et al. Scaling Up Fast Evolutionary Programming with Cooperative Coevolution. In Proceedings of the 2001 Congress on Evolutionary Computation, vol. 2, pp. 1101-1108. 2001.
39. Lohn, J.D., W.F. Krausand and G.L. Haith. Comparing a Coevolutionary Genetic Algorithm for Multiobjective Optimization. In Proceedings of the 2002 Congress on Evolutionary Computation, vol. 2, pp. 1157-1162. 2002.
40. Moriarty, D.E. Symbiotic Evolution of Neural Networks in Sequential Decision Tasks. Ph.D. thesis, The University of Texas at Austin. 1997.
41. Murata, T. and H. Ishibuchi. MOGA: Multi-Objective Genetic Algorithms. In IEEE International Conference on Evolutionary Computation, vol. 1, pp. 289-294. 1995.

42. Neef, M., D. Thierens and H. Arciszewski. A Case Study of a Multiobjective Recombinative Genetic Algorithm with Coevolutionary Sharing. In Proceedings of the 1999 Congress on Evolutionary Computation, pp. 796-803. 1999.
43. Ochoa, G., I. Harvey and H. Buxton. On Recombination and Optimal Mutation Rates. In Proceedings of the Genetic and Evolutionary Computation Conference, pp. 13-17. Morgan Kaufmann. 1999.
44. Osyczka, A. Multicriteria Optimization for Engineering Design. In J.S. Gero (ed) Design Optimization, pp, 193-227. Academic Press. 1985
45. Pareto, V. Cours D'Economie Plitique, vol. 1 and 2. Lausanne: F. Rouge. 1896.
46. Parmee, I.C. and A.H. Watson. Preliminary Airframe Design Using Co-evolutionary Multiobjective Genetic Algorithms. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'99), vol. 2, pp. 1657-1665. 1999.
47. Potter, M.A. and K.A. De Jong. A Cooperative Coevolutionary Approach to Function Optimization. In Proceedings of the Parallel Problem Solving from Nature III Conference (PPSN III), pp. 249–257. 1994.
48. Potter, M.A. and K.A. De Jong. Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents. Evolutionary Computation 8(1), pp. 1-29. 2000.
49. Rivera, W. Scalable Parallel Genetic Algorithms. Artificial Intelligence Review 16, pp.153-168. 2001.
50. Rosin, C.D. and R.K. Belew. New Methods for Competitive Coevolution. Evolutionary Computation 5(1), pp.1-29. 1997.
51. Schaffer, J.D. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In Proceedings of the first International Conference on Genetic Algorithms: Genetic Algorithms and their Applications, pp. 93-100. 1985.

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

53. Steuer, J. Multi Criteria Optimization: Theory, Computation, and Application. New York: John Wiley. 1986.

54. Tan, KC, T.H. Lee and E.F. Khor. Evolutionary Algorithm with Dynamic Population Size and Local Exploration for Multiobjective Optimization. IEEE Transactions on Evolutionary Computation 5(6), pp. 565-588. 2001

55. Tan, KC, T.H. Lee and E.F. Khor. Evolutionary Algorithms for Multiobjective Optimization: Performance Assessments and Comparisons. Artificial Intelligence Review 17(4), pp. 251-290. 2002a

56. Tan, K.C., E.F. Khor, J. Cai, C.M. Heng and T.H. Lee. Automating the Drug Scheduling of Cancer Chemotherapy via Evolutionary Computation. Artificial Intelligence in Medicine 25, pp. 169-185. 2002b.

57. Tan, K.C., A. Tay and J. Cai. Design and Implementation of a Distributed Evolutionary Computing Software. IEEE Transactions on Systems, Man and Cybernetics: Part C 33(3), pp. 325-338. 2003a.

58. Tan, K.C., 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 18, pp.183-215. 2003b.

59. Thierens, D. Adaptive Mutation Rate Control Schemes in Genetic Algorithms. In IEEE Congress on Evolutionary Computation 2002, pp. 980-985. 2002.

60. Valenzuela-Rendón, M. and E. Uresti-Charre. A Non-Generational Genetic Algorithm for Multiobjective Optimization. In Proceedings of the Seventh

International Conference on Genetic Algorithms, pp. 658-665. Morgan Kauffmann. 1997.

61. Van Veldhuizen, D.A. and G.B. Lamont. Multiobjective Evolutionary Algorithm Test Suites. In Symposium on Applied Computing, pp. 351-357. Texas. 1999.
62. Van Veldhuizen, D.A. and G.B. Lamont. Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-art. Journal of Evolutionary Computation 8(2), pp. 125-147. 2000.
63. Zitzler, E. and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. IEEE Transactions on Evolutionary Computation 3(4), pp. 257-271. 1999.
64. Zitzler, E., K. Deb and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation 8(2), pp. 173-195. 2000.
65. Zitzler, E., 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. 2001.

List of Publications

The author has contributed to the following publications:

Tan, K.C., Y.J. Yang, and T.H. Lee. Designing a Distributed Cooperative Coevolutionary Algorithm for Multiobjective Optimization. IEEE Congress on Evolutionary Computation, pp. 2513-2520. Australia. 2003.

Tan, K.C., Y.J. Yang, C.K. Goh, and T.H. Lee. Enhanced Distribution and Exploration for Multiobjective Evolutionary Algorithms. IEEE Congress on Evolutionary Computation, pp. 2521-2528. Australia. 2003.

Tan, K.C., E.F. Khor, T.H. Lee, and Y.J. Yang. A Tabu-based Exploratory Evolutionary Algorithm for Multiobjective Optimization. Artificial Intelligence Review 19(3), pp. 231-260. 2003.