A Distributed Cooperative Coevolutionary Algorithm for Multiobjective Optimization

K. C. Tan, Y. J. Yang, and C. K. Goh

***Abstract*—Recent advances in evolutionary algorithms show that coevolutionary architectures are effective ways to broaden the use of traditional evolutionary algorithms. This paper presents a cooperative coevolutionary algorithm (CCEA) for multiobjective optimization, which applies the divide-and-conquer approach to decompose decision vectors into smaller components and evolves multiple solutions in the form of cooperative subpopulations. Incorporated with various features like archiving, dynamic sharing, and extending operator, the CCEA is capable of maintaining archive diversity in the evolution and distributing the solutions uniformly along the Pareto front. Exploiting the inherent parallelism of cooperative coevolution, the CCEA can be formulated into a distributed cooperative coevolutionary algorithm (DCCEA) suitable for concurrent processing that allows inter-communication of subpopulations residing in networked computers, and hence expedites the computational speed by sharing the workload among multiple computers. Simulation results show that the CCEA is competitive in finding the tradeoff solutions, and the DCCEA can effectively reduce the simulation runtime without sacrificing the performance of CCEA as the number of peers is increased.**

***Index Terms*—Coevolution, distributed computing, evolutionary algorithms, multiobjective optimization.**

I. Introduction

MANY real-world optimization applications involve the optimization of multiple noncommensurable and often competing objectives simultaneously [6], [42], [45], [50]. The solutions for these multiobjective (MO) optimization problems often result from both the optimization and decision making process, which can be defined as follows [18].

1) Priori preference articulation where multiple objectives are aggregated into a scalar function with adequate weights. It transforms the MO problem into a single-objective problem prior to optimization.
2) Progressive preference articulation, i.e., decision and optimization are intertwined where partial preference information is provided, upon which the optimization occurs.
3) Posteriori preference articulation where a family of tradeoff solutions is found before a decision is made to determine the best solution.

The use of evolutionary algorithms in the posteriori preference articulation of multiobjective optimization is gaining significant attention from researchers in various fields [2], [13], [17], [24], [38], [54]. Generally, there are two issues in the optimization process of posteriori preference articulation [9]: 1) to find a set of solutions as close as possible to the Pareto optimal front and 2) to find a set of solutions as diverse as possible. As stated by Zitzler *et al.* [52], it is also important to maximize the spread of the Pareto front in MO optimization.

Schaffer [37] first treated multiple objectives separately in the evolution to generate a set of nondominated solutions in a single run. Knowles and Corne [25] proposed the Pareto archived evolution strategy (PAES) with a reference archive of nondominated solutions found in the evolution. The simplest form of PAES employs (1+1) evolution strategy (ES) and uses a mutation operator for local search. The archive approximates the Pareto front and identifies the fitness of current and potential solutions. A map of grid is applied in the algorithm to maintain diversity of the archive. Because of the simplicity of PAES, it serves as a baseline approach against other more complicated multiobjective evolutionary algorithms (MOEAs).

The Pareto envelope-based selection algorithm (PESA) [4] is motivated from the strength Pareto evolutionary algorithm (SPEA) [51] and PAES. It uses an external population to store the evolved Pareto front and an internal population to generate new candidate solutions. The PESA maintains a hyper grid based scheme to keep track of the degree of crowding in different regions of the archive, which is applied to maintain the diversity of external population and to select the internal population from the external population. The nondominated sorting genetic algorithm II (NSGA II) [10] is an improved version of nondominated sorting genetic algorithm (NSGA) [39]. It employs a fast nondominated approach to assign rank to individuals and a crowding distance assignment approach for density estimation. In the case of a tie in rank during the selection process, the individual with a lower density count will be chosen.

The strength Pareto evolutionary algorithm 2 (SPEA2) [53] is an improved version of SPEA [51]. Both the archive and population in the algorithm are assigned a fitness based upon the strength and density estimation. The strength of an individual is denoted as the number of individuals that dominates it, while the density estimation function is an adaptation of the kth nearest neighbor where the density at any point is a decreasing function of the distance to the kth nearest data point. A truncation method based upon the density is applied to keep the archive at a fixed size. The incrementing multiobjective evolutionary algorithm (IMOEA) [41] incorporates a dynamic population size that is computed according to the discovered Pareto front and desired population density. It employs the method of fuzzy boundary local perturbation with interactive local fine-tunings for broader neighborhood explorations. A preserved strategy is included in the algorithm to ensure good stability and diversity in the evolution.

Manuscript received November 27, 2003; revised November 8, 2004.

The authors are with the Department of Electrical and Computer Engineering, National University of Singapore, 117576 Singapore (e-mail: eletankc @nus.du.sg).

Digital Object Identifier 10.1109/TEVC.2005.860762

Although many successful MOEAs have been proposed over the years [4], [10], [19], [25], [41], [50], [52], the computational cost involved in terms of time and hardware for evolving a complete set of tradeoff solutions in MO optimization often becomes insurmountable as the size or complexity of the problem to be solved increases. Besides the increase in runtimes, Khare *et al.* [22] also showed that the performances of MOEAs do not scale well with respect to the number of objectives. Meanwhile, studies have shown that coevolutionary mechanisms can increase the efficiency of the optimization process significantly [27], [30], [33], [34]. Therefore, one promising approach to overcome the limitation of MOEAs is to incorporate the mechanism of coevolution by decomposing a complex MO optimization problem into smaller problems via a number of subpopulations coevolving for the set of Pareto optimal solutions in a cooperative way.

Neef *et al.* [31] introduced the concept of coevolutionary sharing and niching into multiobjective genetic algorithms by adapting the niche radius through competitive coevolution. Parmee and Watson [32] used multiple populations where each population optimizes one objective that is related to the problem. The individual fitness of each population is then adjusted by comparing the variable values of identified solutions related to a single-objective with the solutions of other populations. Lohn *et al.* [28] embodied the model of competitive coevolution in MO optimization, which contains population of candidate solutions and target population with the target objective vectors. Keerativuttiumrong *et al.* [21] extended the approach of cooperative coevolutionary genetic algorithm [33], [34] for MO optimization by evolving each species with multiobjective genetic algorithm [14] in a rather elementary way.

Inspired by the basic principle of divide-and-conquer in cooperative coevolution, this paper 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 of cooperative coevolution, the CCEA is further formulated into a computing structure suitable for concurrent processing that allows intercommunications of subpopulations residing in multiple computers over the Internet.

The remainder of this paper is organized as follows: Section II describes the principle of CCEA for MO optimization. The implementation of software infrastructure necessary to support a distributed CCEA using the resource of networked computers is presented in Section III. Section IV examines the different features of CCEA and provides a comprehensive comparison of CCEA with other MOEAs based upon various benchmark problems. The performance improvement of the distributed CCEA running on multiple networked computers is also studied in Section IV. Conclusions are drawn in Section V.

II. A Cooperative Coevolutionary Algorithm

A. Coevolution Mechanism

Recent advances in evolutionary algorithms (EAs) show that the introduction of ecological models and coevolutionary architectures are effective methods to broaden the use of traditional evolutionary algorithms [34], [36]. In particular, coevolutionary techniques provide an effective means of handling large and complex problems via a divide-and-conquer strategy. According to Khare *et al.* [23], coevolutionary techniques can be implemented at two basic levels depending on the type of modules that are evolved simultaneously. In the case of single-level coevolution [33], [34], each evolving subpopulation represents a subcomponent of the problem to be solved. On the other hand, a two-level coevolutionary process involves simultaneous optimization of the system and modules in separate subpopulations [23], [30]. Darwen and Yao [7] demonstrated that coevolution can also be achieved between evolving individuals within a single population by means of a niching mechanism.

Coevolution can be classified into competitive coevolution and cooperative coevolution. While the former tries to make individuals more competitive through evolution, the latter aims to find individuals from which better systems can be constructed. Many studies [1], [36] show that competitive coevolution leads to an “arms race” where two populations reciprocally drive each 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, e.g., increased fitness for one of the species implies a diminution in the fitness for the other species. Such an evolutionary pressure tends to produce new strategies in the populations involved in order to maintain their chances of survival.

The basic approach of cooperative coevolution is to divide a large system into many modules and evolve the modules separately [34]. These modules are then combined again to form the whole system. The cooperative coevolutionary algorithms involve a number of independently evolving species that together form a complex structure 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 difficulty of the problem favors the development of cooperative strategies and individuals. Potter and De Jong [33] present a cooperative coevolutionary algorithm with improved performance on many benchmark functions. The approach was also successfully applied to applications of string matching and neural network design [34]. Moriarty [30] applied a cooperative coevolutionary approach to evolve neural networks, where each individual in one species corresponds to a single hidden neuron of a neural network as well as its connections with input and output layers. This population coevolves alongside with a second population that encodes sets of hidden neurons in its individuals (i.e., individuals from the first population) to form a neural network. Liu *et al.* [29] used cooperative coevolution to speed up the convergence of a fast evolutionary programming for solving large-scale problems with dimensions ranging from 100 to 1000.

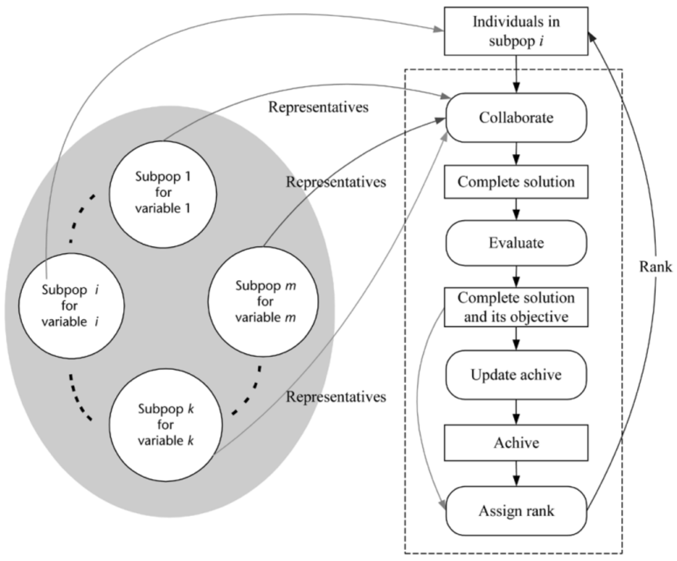

Individuals in subpop i
Representatives
Collaborate
Representatives
Complete solution
Subpop 1 for variable 1
Evaluate
Rank
Subpop i for variable i
Subpop m for variable m
Complete solution and its objective
Update achive
Representatives
Subpop k for variable k
Achive
Assign rank

Fig. 1. Cooperation and rank assignment in CCEA.

B. Adaptation of Cooperative Coevolution for Multiobjective Optimization

1) Cooperation and Rank Assignment: Given a single-objective optimization problem with m parameters, each parameter can be assigned a subpopulation, and these m subpopulations can coevolve individuals contained in each of them [27], [33], [34]. The proposed CCEA extends the idea of assigning one subpopulation to each parameter to evolve multiple nondominated individuals along the Pareto front for MO optimization. Fig. 1 depicts the principle of cooperation and rank assignment in CCEA, where individuals in a subpopulation i are cooperated with representatives from other subpopulations to form the complete candidate solutions.

As shown in Fig. 1, each subpopulation optimizes only one parameter in the evolution. An individual in a subpopulation represents only one component of a candidate solution. The extent of an individual to cooperate with other subpopulations is used to determine the fitness of the individual. Ideally, an individual needs to cooperate with all individuals from other subpopulations in order to determine the extent of cooperation and fitness. Such an exhaustive approach however requires extensive computational effort and high complexity in the algorithm. In practical cooperative coevolutionary algorithms [33], [34], an individual often cooperates with only certain members of other subpopulations to estimate its cooperation extent and fitness.

In CCEA, the best two individuals in a subpopulation are defined as the representative set of the subpopulation. To evaluate an individual in a subpopulation, two types of cooperation are applied to produce two candidate solutions. The first cooperation combines the individual under evaluation with the best representative of every other subpopulation, while the second cooperation combines the individual with a random representative of every other subpopulation. The better of the two candidate solutions is retained, i.e., solution of the first cooperation is selected unless it is dominated by the solution of second cooperation. Intuitively, it is often too greedy to select only the best individual as the representative, which may result in the search being trapped in local optima. The second cooperation in CCEA thus serves to assist the alleviation of possible premature convergence in the evolution. The candidate solution is then mapped onto an objective vector by the objective functions, which is used to evaluate how well the selected individual cooperates with other subpopulations for producing good solutions. A simple Pareto ranking scheme [15] is applied to assign each individual a scalar rank value for use in the operation of reproduction. It ranks an individual according to how many members in the archive dominating it, e.g., the rank of an individual i can be given as

$$\mathrm{rank}(i) = 1 + n_i \tag{1}$$

where n_i is the number of archive members dominating the individual i in the objective domain.

The cooperative coevolution changes only one component at a time and provides a fine-grained search capability desirable in many applications [46]. It also increases diversity of the solutions with the approach of cooperation among multiple subpopulations. The fine-grained search strategy allows extensive exploration in one dimension without heavy disruption to the structure of the candidate solutions, which increases its ability to escape from harmful local optima. These features of cooperative coevolution are incorporated in CCEA to achieve a better performance for MO optimization.

2) Archive: The set of Pareto optimal solutions in CCEA is obtained by incorporating an archive to store the nondominated individuals discovered along the evolution. The archive is updated by the candidate solutions generated by the cooperation, and is regarded as the optimal set of solutions at the end of the evolution. It also works as an elitism mechanism that ensures a good convergence in CCEA and serves as a comparison set for rank assignment of individuals from the subpopulations. The size of archive can be adjusted according to the desired number of individuals to be distributed on the tradeoffs in the objective domain. As illustrated in Fig. 2, the archive will be updated once a candidate solution is formed. If the candidate solution is not dominated by any members in the archive, it will be added to the archive, and any archive members dominated by this solution will be discarded from the archive. When the maximum archive size is reached, a truncation method based on niche count (nc) is used to eliminate the most crowded archive members in order to maintain diversity of individuals in the archive. To distribute the nondominated individuals equally along the Pareto front, the dynamic sharing scheme by Tan *et al.* [44] is incorporated in CCEA. A partial order is used to compare the strengths of individuals such that when two individuals are compared in the reproduction, ranks will be considered first followed by niche count in order to break the tie of ranks, as adopted in [10], [25], [41]. For any two individuals i and j, $i \geq_n j$ if $rank(i) < rank(j)$ or $\{rank(i) == rank(j)$ and $nc(i) < nc(j)\}$.

3) Extending Operator: An extending operator is used in CCEA to improve the smoothness and spread of nondominated individuals in covering the entire Pareto front uniformly. Usually, the less populated regions are the gaps or boundaries in the archive. The extending operator is thus designed to guide the evolutionary search to concentrate on these underrepresented

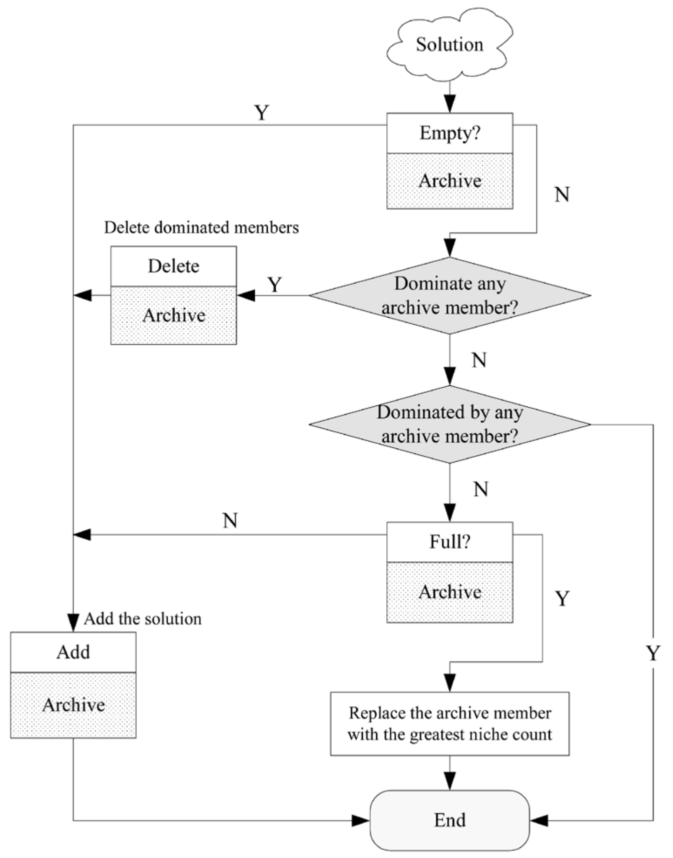

Solution
Empty?
Archive
Y
N
Delete dominated members
Delete
Archive
Y
Dominate any archive member?
N
Dominated by any archive member?
N
Full?
Archive
Y
Add the solution
Add
Archive
Y
Replace the archive member with the greatest niche count
End

Fig. 2. The process of archive updating.

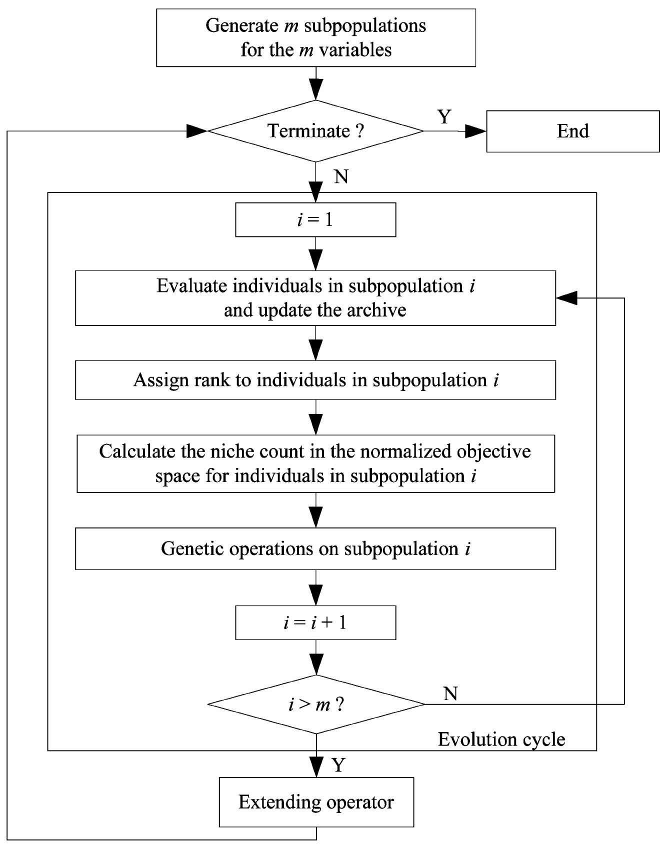

Generate m subpopulations for the m variables
Terminate ?
Y
End
N
i = 1
Evaluate individuals in subpopulation i and update the archive
Assign rank to individuals in subpopulation i
Calculate the niche count in the normalized objective space for individuals in subpopulation i
Genetic operations on subpopulation i
i = i + 1
i > m ?
N
Evolution cycle
Y
Extending operator

Fig. 3. The program flowchart of CCEA.

regions. At the initial stage of evolution, the CCEA focuses on finding the nondominated individuals to fill up the archive and to achieve a good preliminary approximation to the Pareto front. When the archive size is full, the archive member that resides in the least populated region based upon the niche count is discovered, which is subsequently cloned and added to the evolving subpopulation in order to explore the underrepresented regions thoroughly. Detailed description of the extending operator is given as follows:

```
The extending operator for CCEA
Let $c$ 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 and find the
        member with the smallest niche
        count, e.g., the member that
        resides in the least populated
        region.
Step 3) Clone $c$ copies of this archive
        member to the subpopulations, where
        each part of this member is cloned
        into its corresponding subpopula-
        tion.
```

4) Overview of CCEA: As depicted in the flowchart of CCEA in Fig. 3, m subpopulations are randomly initialized for an m-parameter problem, each of which optimizes only one parameter. In the evolution cycle, the m subpopulations are evolved in a sequential way. To evaluate an individual in the evolving subpopulation, a complete candidate solution is formed by combining the evaluated individual with representatives from other subpopulations. The archive is updated based upon the evaluation results and the range of the objective space is estimated from the updated archive. According to the information, each individual will be assigned a rank and its respective niche count in the normalized objective space will be obtained. As shown in Fig. 4, the genetic operations consist of tournament selection, uniform crossover, and bit-flip mutation. Once an evolution cycle is completed, the extending operator finds the archive member residing in the least populated region, which is subsequently cloned and added to the evolving subpopulations in order to explore the underrepresented regions thoroughly and to distribute the nondominated individuals uniformly along the Pareto front.

III. A Distributed Cooperative Coevolutionary Algorithm

A. Distributed Evolutionary Computing

It is known that the computational cost for evolutionary optimization in terms of time and hardware increases as the size and complexity of the problem increase. One approach to overcome such a 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

```
Let $sp(i, g)$ denotes the subpopulation $i$ at generation $g$,
Step (1) Perform tournament selection based on the rank and niche count on $sp(i, g)$. The
resultant subpopulation is called $slctsp(i, g)$.
Step (2) Perform uniform crossover to $slctsp(i, g)$ with probability $Pc$. The resultant
individual list is $xsp(i, g)$.
Step (3) Perform bit-flip mutation to $xsp(i, g)$ with probability $Pm$. The resultant
individual list is $sp(i, g+1)$.
```

Fig. 4. Genetic operations in a subpopulation i.

subtasks and to solve the subtasks simultaneously using multiple processors. This divide-and-conquer approach has been applied in different ways, and many parallel EA implementations have been reported in literature [3], [12], [16], [20], [35], [49].

As categorized by Rivera [35], 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 often complex since it consists of multiple subpopulations and different subpopulations may exchange individuals occasionally (migration). In hybrid parallelization, several parallelization approaches are combined. The complexity of these hybrid-parallel EAs depends on the level of hybridization.

B. A Distributed CCEA (DCCEA)

The availability of powerful networked computers presents a wealth of computing resources to solve problems with large computational effort. As the communication level in coarse-grained parallelization is low as compared to other parallelization strategies, it is a suitable computing model for distributed networks with limited communication speeds. This parallelization approach is considered here, where large problems are decomposed into smaller subtasks that are mapped into the computers available in a distributed system.

To design the CCEA suitable for distributed computing, several issues are considered, such as variant communication overhead, different computation speeds, and network restrictions. A prototype model of DCCEA with six subpopulations over three peer computers is illustrated in Fig. 5. As shown in the figure, each parameter is assigned a subpopulation, and these subpopulations are further partitioned into groups depending on the number of available peers. Without loss of generality, the six subpopulations are divided into three groups, each of which is assigned to a peer computer. Note that each peer contains its own archive and representatives, which evolves its subpopulations sequentially in an approach similar to CCEA.

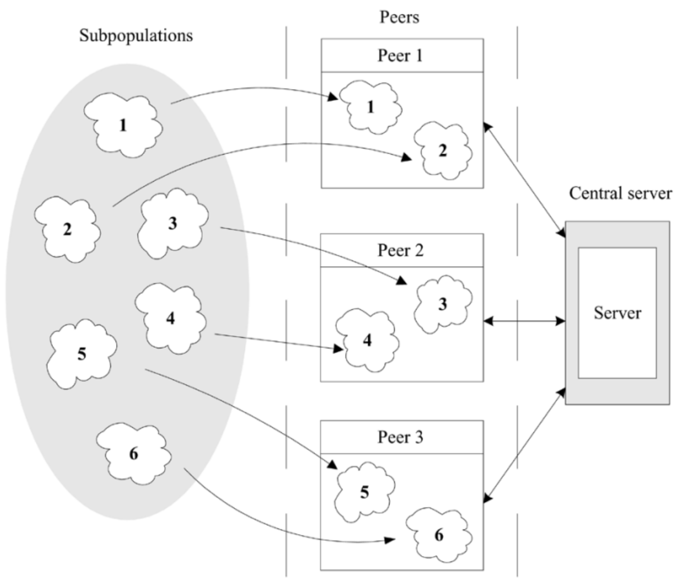


Fig. 5. The model of DCCEA.

Inside a peer, the candidate solutions generated through the collaboration consistently update the archive in the peer. The subpopulations in the peer then update their corresponding peer representatives once in every cycle. The cooperation among peers is indirectly achieved through the exchange of archive and representatives between the peers and a central server. In DCCEA, the communication time among peers is a conspicuous part of the whole execution time. To reduce the communication overhead, the exchange of archive and representatives between a peer and the central server only occurs every several generations as determined by the exchange interval. Since the peers are often not identical, the cooperation among peers will become ineffective if there is a big difference for the evolution progress among the peers, e.g., the poor cooperation among peers may deteriorate the performance of DCCEA. To make the peers cooperate well in the evolution, the peers are synchronized every several generations as determined by the synchronization interval. Note that the exchange and synchronization intervals in DCCEA can be fixed or computed adaptively along the evolution.

In DCCEA, there is no direct communication among the subpopulations. All communications are performed between the subpopulation and the central server. Such communications take

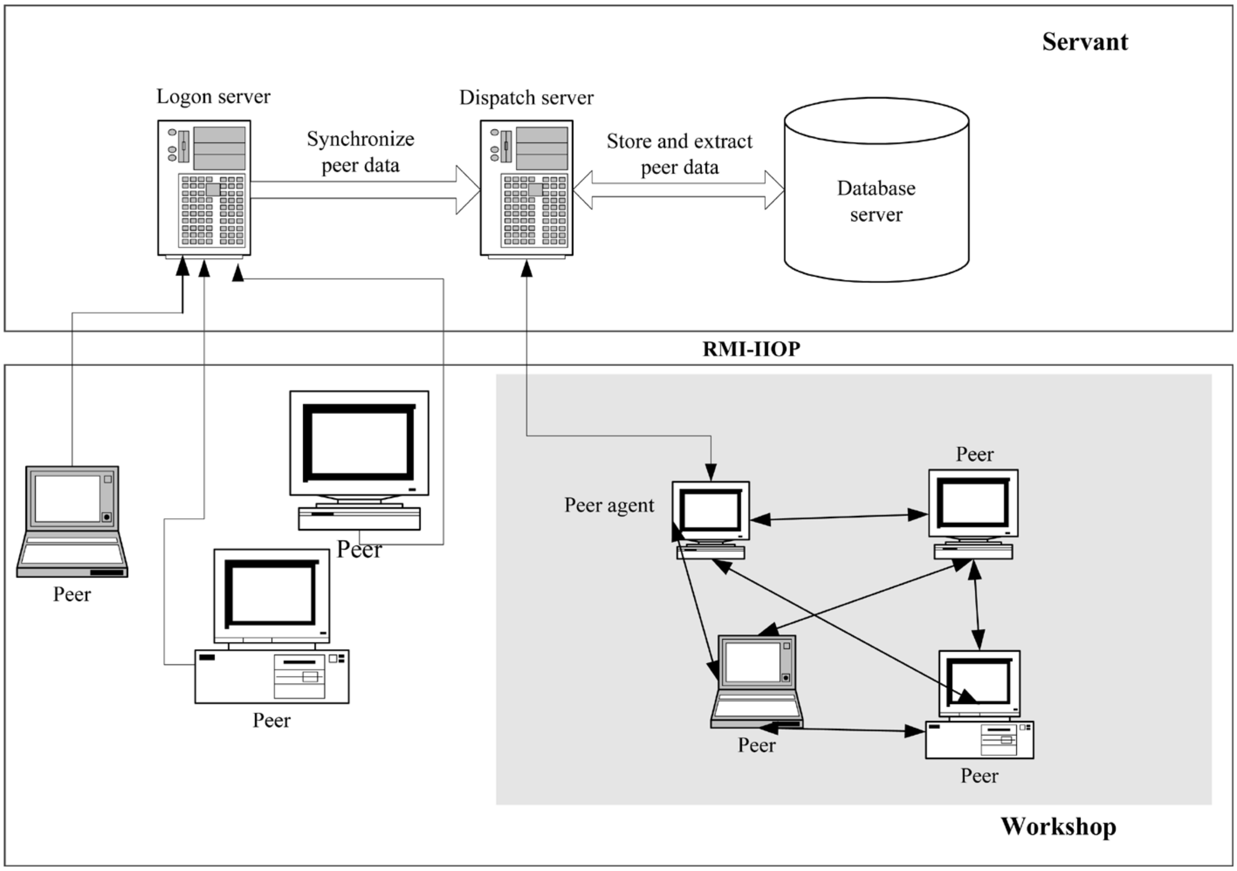


Fig. 6. Schematic framework of Paladin-DEC software.

place every k generations for a subpopulation. For the subpopulation, the communication overhead does not change. For the central server, the communication overhead increases linearly with respect to the number of computers, which is not a big issue because the central server is dedicated to communication and generally can deal with the communication amount in DCCEA. Several subpopulations can reside in one computer so that the number of computers and the communication overhead can be limited. Alternatively, more than one central server can be used to divide the computers into different groups. The communication overhead can scale well, even if there are a thousand variables.

C. Implementation of DCCEA

The DCCEA has been designed and embedded in a distributed computing framework named Paladin-DEC [43] that was built upon the foundation of Java technology offered by Sun Microsystems and was 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 [40]. 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 advantage of Java programming language, such as platform independence and reusability, makes the approach attractive.

As shown in Fig. 6, the Paladin-DEC software consists of two main blocks, i.e., the servant block and workshop block that are connected by remote method invocation over the Internet inter-ORB protocol (RMI-IIOP). The servant functions as an information center and backup station through which peers can check their identifications or restore their working status. The workshop is a place where peers (free or occupied) work together in groups, e.g., the working peers are grouped together to perform a specified task, while the free ones wait for new jobs to be assigned. The servant contains three different servers, i.e., logon 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 and the group of peers to perform the execution. It also handles the transfer of peers' information to/from the database server. Besides, the dispatcher server also synchronizes the information, updates the peer's list, and informs the database server for any modification. Whenever a task is available, the dispatcher server will transfer the task to a group of selected peers.

The working process of a peer begins when it logs onto the server by sending a valid email address to the server. The peer computers are then pooled and wait for the tasks to be assigned by the server. Once a peer detects that a task has been assigned,

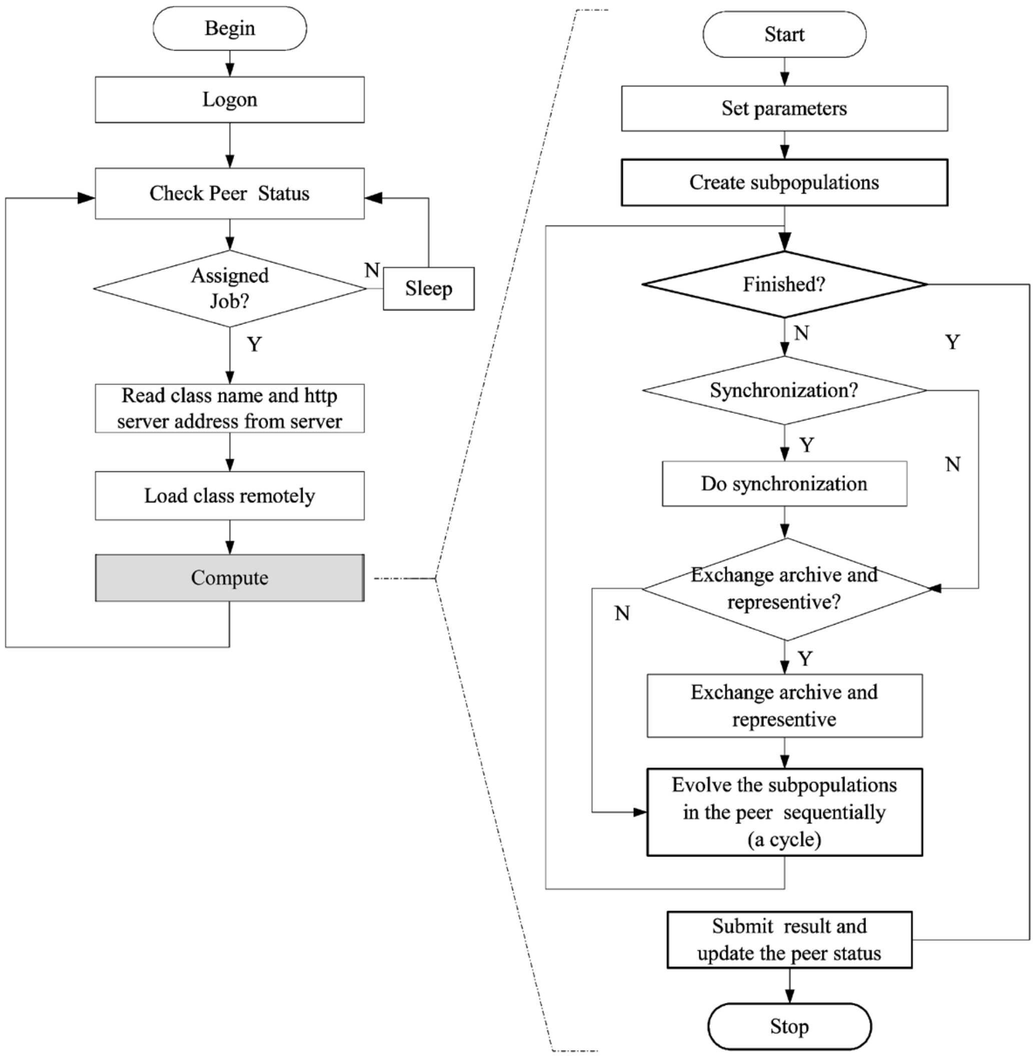

Begin
Logon
Check Peer Status
Assigned Job?
N
Sleep
Y
Read class name and http server address from server
Load class remotely
Compute
Start
Set parameters
Create subpopulations
Finished?
N
Y
Synchronization?
Y
N
Do synchronization
Exchange archive and representive?
N
Y
Exchange archive and representive
Evolve the subpopulations in the peer sequentially (a cycle)
Submit result and update the peer status
Stop

Fig. 7. The workflow of a peer.

it will extract relevant 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. 7 depicts the working process of a peer, where the description of the module "Compute" is shown in the right part of the figure.

Once a peer starts the computation procedure, it first initializes the parameters, such as generation number, number and size of subpopulations, crossover rate, and mutation rate. The peer then creates the subpopulations assigned to it. When it 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 checks whether it should exchange the archive and representatives between the peer and server. If the conditions of exchange are satisfied, the peer initiates a session in the server and exchanges/updates the archive and representatives between the peer and server. The peer then evolves its subpopulations sequentially for one generation after these procedures. If the peer meets the termination conditions, it will initiate a session to submit the results and restores itself to the ready status. If the user cancels a running job, those peers involved in the job will stop the computation and set themselves to the ready status.

D. Workload Balancing

As the processing power and specification for various computers in a network may be different, the feature of workload balancing that ensures the peers are processed in a similar pace is important in DCCEA. Besides having a long simulation time, the poor cooperation among computers will also deteriorate the performance of DCCEA if a heavy workload is assigned to the peer that has a relatively low processing power. Intuitively, workload balancing for a distributed system could be difficult because the working environment in a network is often complex and uncertain. The DCCEA resorts to a simple workload balancing strategy by assigning workload to peers according to their respective computational capabilities. As stated in Section III-C, when a peer is first launched, it uploads its configuration information to the server, including hardware configuration of the peer such as CPU speed, RAM size, etc. Based on the information, the dispatch server performs a simple task scheduling and assigns different tasks to the respective peers according to their processing powers.

E. Computation Time Analysis of CCEA and DCCEA

Assume that all mathematics, comparison, and assignment statements in the computer are executed in (nearly) identical time [47]. This assumption is reasonable as long as the necessary operations execute in about the same time. For a specific MO optimization problem, the parameter number, objective number, objective function computation time, subpopulation size, archive size, and generation number in CCEA can be denoted by m, n, t_f, N_s, N_a, G, respectively. Then it requires n comparison of individuals. The Euclidean distance computation in the objective domain requires n subtractions, n multiplications, $(n-1)$ additions, and one extraction of root. Its computation time is $3n$. The computation of crossover and mutation on an individual once is denoted by t_e. The analysis is first conducted on the computation time of CCEA, which can be divided into three parts: a single generation of one subpopulation, a cycle of CCEA, and the entire execution of CCEA.

For a single generation of one subpopulation, the major steps performed are a) evaluating individuals and updating archive and b) genetic operations. The computation time of these steps are as follows: For each individual, step a) computes the objective vector, rank, and niche count, replaces the archive member with the greatest niche count, and recomputes niche count of archive members. The computational cost of an objective vector is t_f. The individual is compared with all the archive members to compute the rank and the computation is $n \cdot N_a$. To calculate the niche count of an individual, distances between the individual and archive members are computed and compared with the niche radius. The calculation of these distances requires $n \cdot N_a$ and the comparison of distances with the niche radius needs N_a. The computation to sum contributions of members in the neighborhood for niche count can be neglected, since only a small portion of archive members falls into the neighborhood of an individual. Hence the computation of a niche count takes approximately $(3n+1) \cdot N_a$. For the worst case of archive updating, the individual is indifferent with respect to the archive and replaces the most crowded archive member. It takes N_a comparisons of individual and N_a comparisons of niche count. After updating, re-computing niche count of all the archive members takes $(3n+1) \cdot N_a \cdot N_a$. Thus the worst computation of an individual in step a) is $t_f + n \cdot N_a + (3n+1) \cdot N_a + (n+1) \cdot N_a + (3n+1) \cdot N_a \cdot N_a$. For a subpopulation, this step takes $[t_f + (5n+2) \cdot N_a + (3n+1) \cdot N_a \cdot N_a] \cdot N_s$.

Step (b) conducts tournament selection, crossover, and mutation of individuals. The selection criterion for the tournament is based on Pareto dominance relationship. In the event of a tie, individual rank and, if necessary, niche count will be used. Therefore, the tournament requires one individual comparison, one comparison of ranks and one comparison of niche count in the worst case. Its computation time is $(n+2)$. For a subpopulation, step b) requires $(n+2+t_e) \cdot N_s$. Adding up the computation time derived in steps a) and b), a single generation of one subpopulation takes

$$[t_f + (5n+2) \cdot N_a + (3n+1) \cdot N_a \cdot N_a + n + 2 + t_e] \cdot N_s. \quad (2)$$

In order to derive the computation for a cycle of CCEA, the previous result is multiplied by the subpopulation number, which is equal to the parameter number m. Thus a cycle of CCEA takes $[t_f + (5n+2) \cdot N_a + (3n+1) \cdot N_a \cdot N_a + n + 2 + t_e] \cdot N_s \cdot m$. After a cycle, the computation in extending operator takes N_a comparisons to find the most crowded archive member. To derive the computation time of CCEA, $[t_f + (5n+2) \cdot N_a + (3n+1) \cdot N_a \cdot N_a + n + 2 + t_e] \cdot N_s \cdot m$ and N_a are multiplied by the generation number G. Therefore the computation of CCEA as a whole is

$$[t_f + (5n+2) \cdot N_a + (3n+1) \cdot N_a \cdot N_a + n + 2 + t_e] \cdot N_s \cdot m \cdot G + N_a \cdot G. \quad (3)$$

Generally, this computation time should be greater than the actual amount because the archive size is smaller than N_a and the extending operator is not used when the archive is not full at the initial stage of evolution.

For DCCEA, the subpopulations are partitioned into groups and each group will be assigned to a node. Assume there are p nodes and the subpopulations are partitioned evenly. Each node has m/p subpopulations and the computation is

$$[t_f + (5n+2) \cdot N_a + (3n+1) \cdot N_a \cdot N_a + n + 2 + t_e] \cdot N_s \cdot m \cdot \frac{G}{p} + N_a \cdot G. \quad (4)$$

The communication time among nodes for cooperation cannot be neglected. Assume the network speed is b byte per second and the representation of a real number consists of 4 bytes. The archive member has m parameters and n objectives. Each subpopulation has r representatives and the total number of representatives of m subpopulations is $r \cdot m$. Therefore, the volume of archive and representative set is $4 \cdot (m+n) \cdot N_a$ bytes and $4 \cdot r \cdot m$ bytes, respectively. In the exchange of archive and representative set between a node and the server, the overall data volume of two transfers is $4 \cdot [(m+n) \cdot N_a + r \cdot m] \cdot 2 = [(m+n) \cdot N_a + r \cdot m] \cdot 8$. The communication time of all the exchanges is

$$[(m+n) \cdot N_a + r \cdot m] \cdot 8 \cdot \left(\frac{1}{b}\right) \cdot \left(\frac{G}{k}\right) \quad (5)$$

where k is the exchange interval. In the server side, the node and server archives should be combined at each exchange. The combination can be regarded as the updating of the server archive with every member in the node archive. In the worst case, it takes $n \cdot N_a$ to compare the member i in the node archive with the server archive, $(3n+1) \cdot N_a$ to compute the niche count of i in the server archive, N_a to find the most crowded member in the server archive, and $(3n+1) \cdot N_a \cdot N_a$ to recompute the niche count of the server archive. The combination of the node and server archives thus takes $[(4n+2) \cdot N_a + (3n+1) \cdot N_a \cdot N_a] \cdot N_a$. The total computation on the server side is

$$[(4n+2) \cdot N_a + (3n+1) \cdot N_a \cdot N_a] \cdot N_a \cdot m \cdot \frac{G}{k}. \quad (6)$$

Note that (6) reflects the computation time of DCCEA, which does not represent the upper bound of the computation required in finding the optimal Pareto front. When the communication time and the computation in server is relatively low in the whole computation, e.g., in the case of heavy computation of objective

TABLE I
DEFINITION OF FON, KUR, TLK, DTLZ2, AND DTLZ3

Problem	Definition	Variable	Feature
FON	$f_1(\mathbf{x}) = 1-\exp[-\sum_{i=1}^{8}(x_i - 1/\sqrt{8})^2]$ $f_2(\mathbf{x}) = 1-\exp[-\sum_{i=1}^{8}(x_i + 1/\sqrt{8})^2]$	$x_i \in [-2,2)$ $i = 1,\ldots,8$	Non-convex
KUR	$f_1(\mathbf{x}) = \sum_{i=1}^{2}[-10\exp(-0.2\sqrt{x_i^2 + x_{i+1}^2})]$ $f_2(\mathbf{x}) = \sum_{i=1}^{3}[\lvert x_i \rvert^{0.8} + 5\sin(x_i^3)]$	$x_i \in [-5,5]$ $i = 1,2,3$	Non-continuous
TLK	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = \frac{1}{x_1}\{1+(x_2'^2 + x_3'^2)^{0.25}\cdot$ $[\sin^2(50\cdot(x_2'^2 + x_3'^2)^{0.1})+1]\}$ $x_i' = x_i + N(0,0.1)$	$x_1 \in [0.1,1]$ $x_i \in [-100,100]$ $i = 2,3$	Noisy landscape
DTLZ2	$f_1(\mathbf{x}) = (1+g(\mathbf{x}_M))\cos(x_1\pi/2)\cdots\cos(x_{M-1}\pi/2)$ $f_2(\mathbf{x}) = (1+g(\mathbf{x}_M))\cos(x_1\pi/2)\cdots\sin(x_{M-1}\pi/2)$ $\vdots$ $f_M(\mathbf{x}) = (1+g(\mathbf{x}_M))\sin(x_1\pi/2)$ $g(\mathbf{x}_M) = \sum_{x_i\in\mathbf{x}_M}(x_i - 0.5)^2$ $\mathbf{x}_{\mathrm{M}} = \{x_M,\ldots,x_{M+9}\}$	$M = 5$ $x_i \in [0,1]$ $i = 1,2,\ldots,M+9$	High dimensional objective space
DTLZ3	$f_1(\mathbf{x}) = (1+g(\mathbf{x}_M))\cos(x_1\pi/2)\cdots\cos(x_{M-1}\pi/2)$ $f_2(\mathbf{x}) = (1+g(\mathbf{x}_M))\cos(x_1\pi/2)\cdots\sin(x_{M-1}\pi/2)$ $\vdots$ $f_M(\mathbf{x}) = (1+g(\mathbf{x}_M))\sin(x_1\pi/2)$ $g(\mathbf{x}_M) = 100\{\lvert \mathbf{x}_M \rvert +$ $\sum_{x_i\in\mathbf{x}_M}[(x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5))]\}$ $\mathbf{x}_M = \{x_M,\ldots,x_{M+9}\}$	$M = 5$ $x_i \in [0,1]$ $i = 1,2,\ldots,M+9$	High dimensional objective space; Many local Pareto fronts

functions, DCCEA can greatly reduce the runtime with respect to CCEA and the speedup is given as

$$\frac{\text{Eqn.3}}{\text{Eqn.4} + \text{Eqn.5} + \text{Eqn.6}} \approx p. \tag{7}$$

IV. SIMULATION STUDIES

This section presents a few benchmark problems and four performance metrics for MO optimization, which are used to compare the performance of CCEA with other popular MOEAs, such as PAES [25], PESA [4], NSGAII [10], SPEA2 [53], and IMOEA [41]. The section also shows simulation studies of DCCEA for examining its performance and efficiency of parallel implementation.

A. The Test Problems

Eleven test problems are used to examine the performance of CCEA. The test problems of ZDT1, ZDT2, ZDT3, ZDT4, ZDT5, and ZDT6 were designed by Zitzler *et al.* [52] using Deb's scheme [8], as can be found in [10] and [52]. Other test problems include FON [15], KUR [26], TLK [41], DTLZ2 [11], and DTLZ3 [11], as listed in Table I. These problems include characteristics that are suitable for examining the effectiveness of MO approaches in maintaining the population diversity as well as converging to the Pareto front. The problems of DTLZ2 and DTLZ3 also challenge the algorithms' ability to handle MO problems with high dimensional objective space. Many researchers, such as Knowles and Corne [25], Corne *et al.* [4], Deb *et al.* [10], Tan *et al.* [41], and Zitzler *et al.* [51]–[53], have used these test problems in their studies of MOEAs.

B. Performance Metrics

Four different quantitative performance metrics for MO optimization are used. These metrics are capable of evaluating nondominated individuals in several nontrivial aspects and have been widely used in the studies of MOEAs, such as in Van Veldhuizen and Lamont [48], Deb [9], and Zitzler *et al.* [52].

1) *Generational Distance (GD):* The metric of generational distance represents how "far" the known Pareto front (PF_{known}) is away from the true Pareto front (PF_{known}), which is defined as

$$\text{GD} = \left(\frac{1}{n_{PF}}\sum_{i=1}^{n_{PF}} d_i^2\right)^{1/2} \tag{8}$$

where n_{PF} is the number of members in PF_{known}, d_i is the Euclidean distance (in the objective domain) between the member i in PF_{known} and its nearest member in PF_{true}.

2) *Spacing (S):* The metric of spacing measures how "evenly" members in PF_{known} are distributed. It is defined as

$$S = \frac{\left[\frac{1}{n_{PF}}\sum_{i=1}^{n_{PF}}(d_i' - \overline{d}')^2\right]^{1/2}}{\overline{d}'}, \quad \overline{d}' = \frac{1}{n_{PF}}\sum_{i=1}^{n_{PF}} d_i' \tag{9}$$

where n_{PF} is the number of members in PF_{known} and d_i' is the Euclidean distance (in the objective domain) between the member i in PF_{known} and its nearest member in PF_{known}.

TABLE II
CONFIGURATION OF CCEA, PAES, PESA, NSGAII, SPEA2, AND IMOEA

Populations	Subpopulation size of 20 for CCEA; population size of 100 for PESA, NSGAII, and SPEA2; population size of 1 for PAES; initial population size of 20 and maximum population size of 100 for IMOEA. Archive (or secondary population) size of 100 for all the algorithms
Chromosome length	30 bits per decision variable
Selection	Binary tournament selection
Crossover rate	0.8
Crossover method	Uniform crossover
Mutation rate	2/*L* for ZDT1, ZDT2, ZDT3, ZDT4, ZDT6, TLK, DTLZ2, and DTLZ3, where *L* is the chromosome length; 1/30 for FON and KUR, where 30 is the bit number per variable
Mutation method	Bit-flip mutation
Sharing radius	Dynamic sharing except 0.1 for ZDT5
Hypergrid size	2^3 per dimension for DTLZ2 and DTLZ3; 2^5 per dimension for other problems
Number of evaluations	120, 000

3) *Maximum Spread (MS):* The metric of maximum spread measures how "well" the PF_{true} is covered by the PF_{known} through hyperboxes formed by the extreme function values observed in the PF_{true} and PF_{known} [52]. In order to normalize the metric, it is modified as

$$\text{MS} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left\{\frac{[\min(f_i^{\max}, F_i^{\max}) - \max(f_i^{\min}, F_i^{\min})]}{(F_i^{\max} - F_i^{\min})}\right\}^2} \quad (10)$$

where n is the number of objectives, $f_i^{\max}$ and $f_i^{\min}$ are the maximum and minimum of the ith objective in PF_{known}, respectively; and $F_i^{\max}$ and $F_i^{\min}$ are the maximum and minimum of the ith objective in PF_{true}, respectively.

4) *Hypervolume (HV) and Hypervolume Ratio (HVR):* The metric of hypervolume calculates the "volume" in the objective domain covered by the set of nondominated individuals for an MO minimization problem [48], [51]. It is defined as

$$\text{HV} = \text{volume}\left(\cup_{i=1}^{n_{PF}} v_i\right) \quad (11)$$

where n_{PF} is the number of members in the nondominated set. For each member i in the nondominated 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 is found by constructing a vector of the worst objective function values. In geometry, the volume of an object is the amount of space occupied by the object. Then HV is the amount of space occupied by the union of hypercubes constructed by the nondominated solutions and the reference point.

To reduce the bias and to normalize the metric, Van Veldhuizen and Lamont [48] defined the metric of hypervolume ratio as a ratio of the hypervolume of PF_{known} and the hypervolume of PF_{true}, which is given as

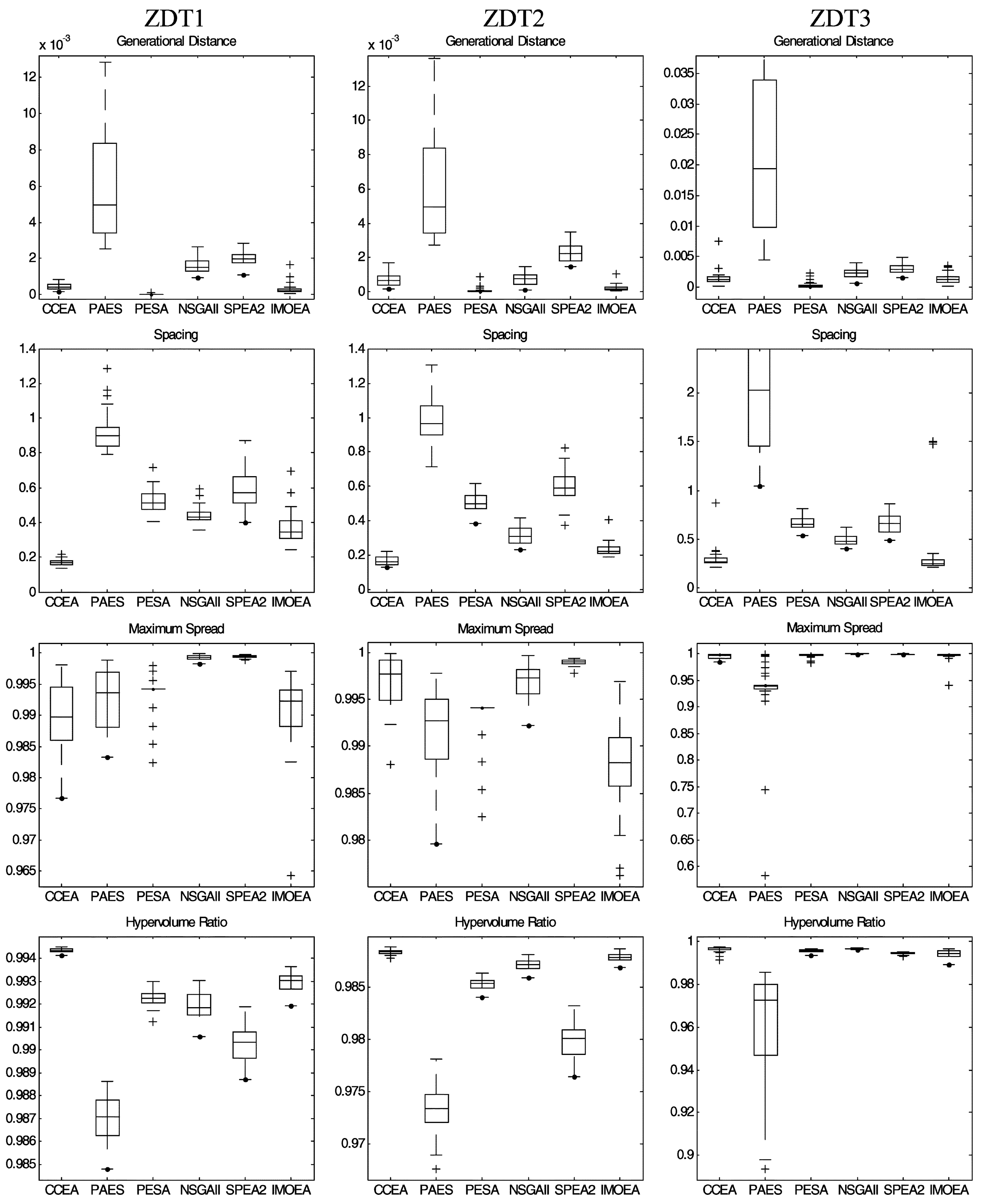
$$\text{HVR} = \frac{\text{HV}(\text{PF}_{\text{known}})}{\text{HV}(\text{PF}_{\text{true}})}. \quad (12)$$

C. Simulation Results of CCEA

In this section, simulations are carried out to examine the performance of CCEA from several aspects. These include the discovery and distribution of nondominated individuals along the entire Pareto front uniformly, the ability to escape from local optima, and the minimization of the effect of noise induced from the environment (robustness). The performance is compared between CCEA and various MOEAs based upon the test problems described in Section IV-A. Although such comparisons are not exhaustive, they provide a good basis to assess the performance of CCEA. In order to provide a fair comparison, all the algorithms considered are implemented with the same binary coding scheme of 30-bit per decision variable, tournament selection, uniform crossover, and bit-flip mutation. An exception is made for the discrete problem of ZDT5, where the first variable is a 30-bit string and the rest of the variables are of 5-bit string according to the definition of ZDT5. In the simulation, 30 independent runs (with random initial populations) of CCEA, PAES, PESA, NSGAII, SPEA2, and IMOEA are performed on each of the test functions in order to study the statistical performance, such as consistency and robustness of the algorithms. The number of function evaluations for each simulation run is fixed and the configuration of the algorithms is shown in Table II.

1) Performance Comparisons: Fig. 8(a)–(d) summarizes the simulation results of the algorithms for each test problem with respect to each performance metric. The distribution of simulation data for the 30 independent runs is represented in the box plot format [5], which has been applied by Zitzler and Thiele [51] and Tan *et al.* [42] to visualize the distribution of simulation data in their studies of MOEAs. Each box plot represents the distribution of a sample set where a thick horizontal line within the box encodes the median, while the upper and lower ends of the box are the upper and lower quartiles. Dashed appendages illustrate the spread and shape of distribution and dots represent the outside values.

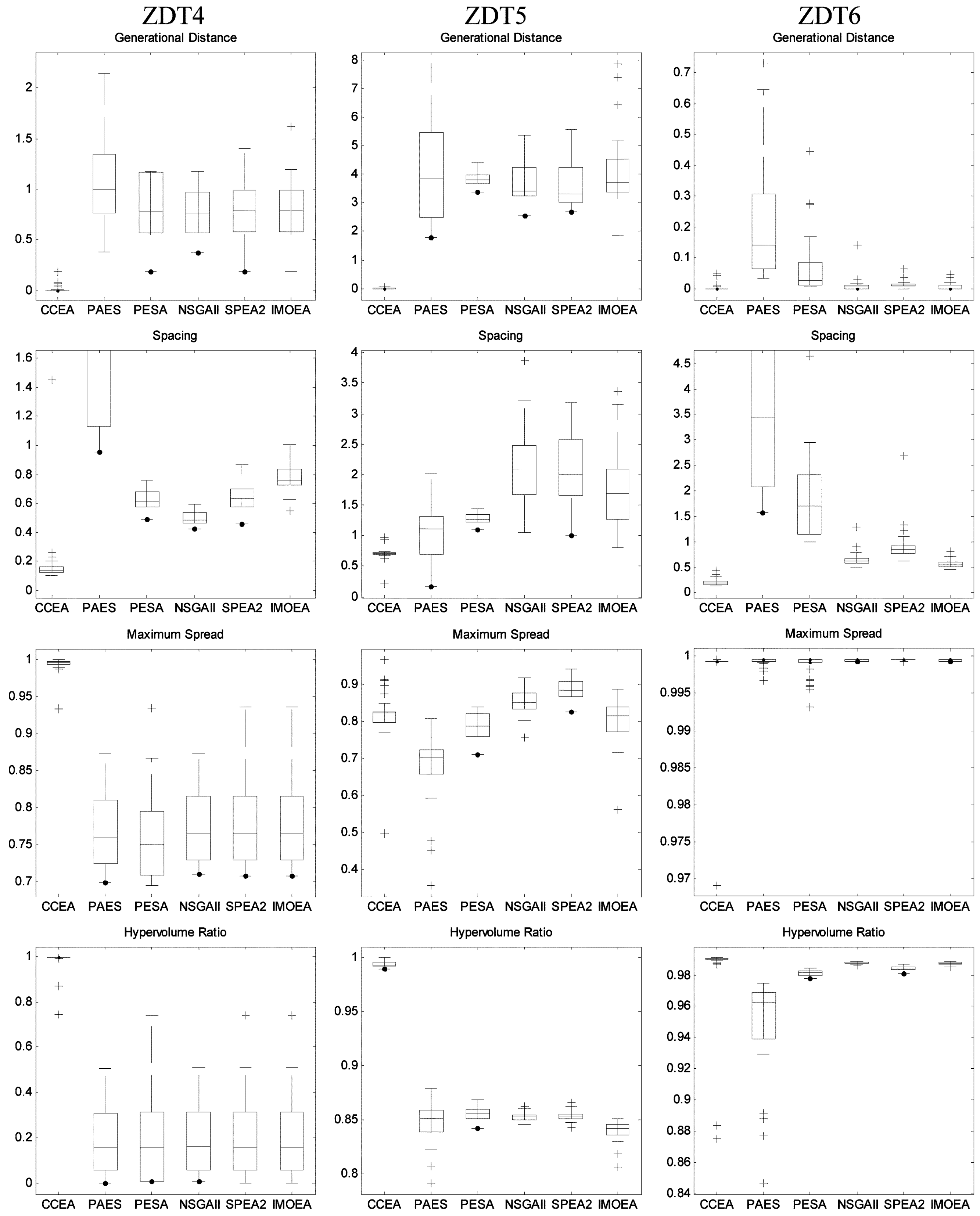
The problems of ZDT1, ZDT2, and ZDT3 are relatively easy to solve according to our simulations. As shown in Fig. 8(a), all algorithms except for PAES perform well for the metric of GD. In terms of solution distribution, CCEA has the best per-



(a)

Fig. 8. Box plots for the metrics of generational distance *(GD)*, spacing *(S)*, maximum spread *(MS)*, and hypervolume ratio *(HVR)*.

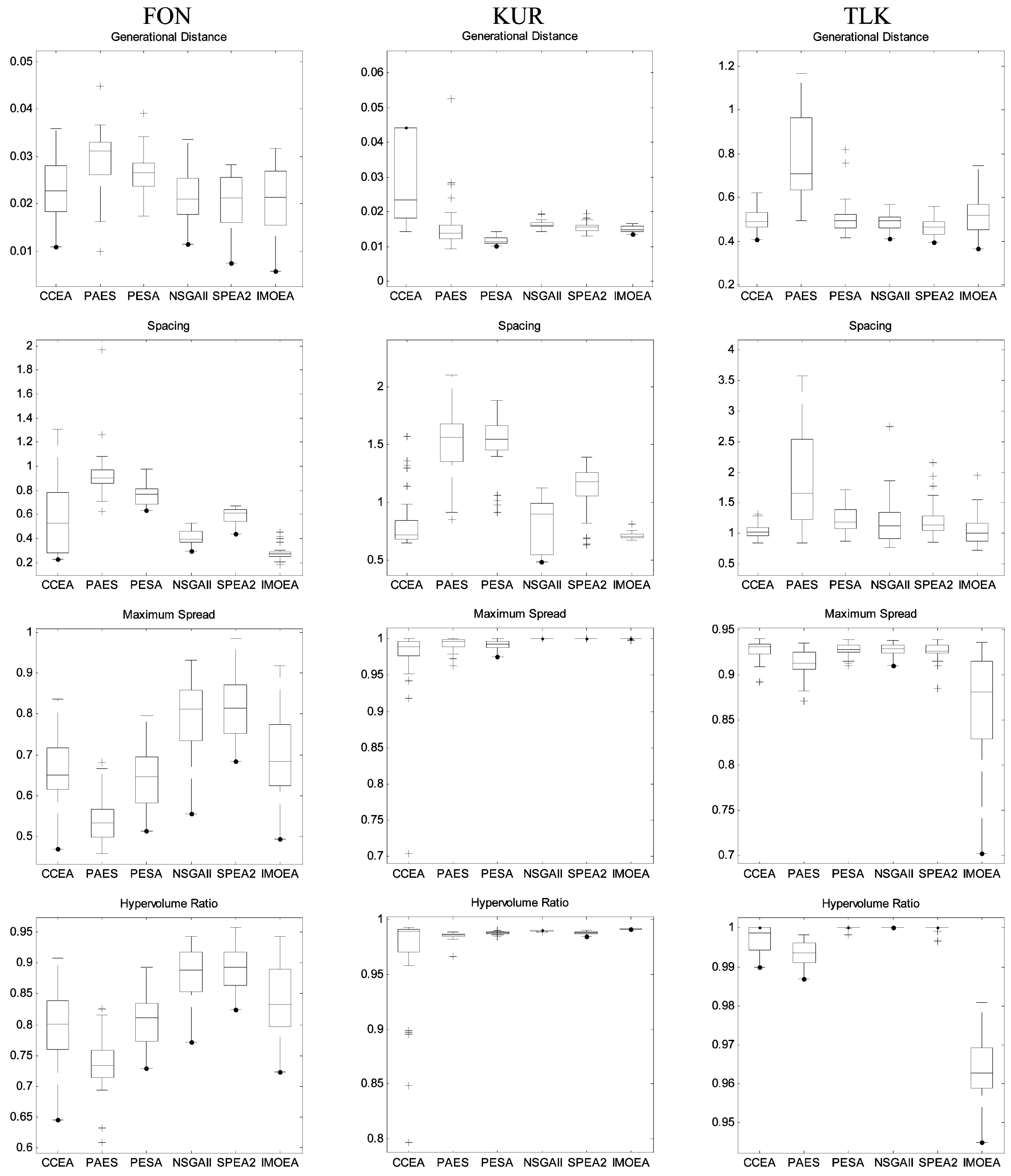
formance on spacing, due to its features of dynamic sharing and extending operator. The results show that PAES performs poorly for the metrics of GD and spacing. Without a population and crossover operator, the exploitation ability of PAES is generally weak as compared to other population-based algorithms. It is observed that the CCEA does not perform the best on MS al-



(b)

Fig. 8. *(Continued.)* Box plots for the metrics of generational distance *(GD)*, spacing *(S)*, maximum spread *(MS)*, and hypervolume ratio *(HVR)*.

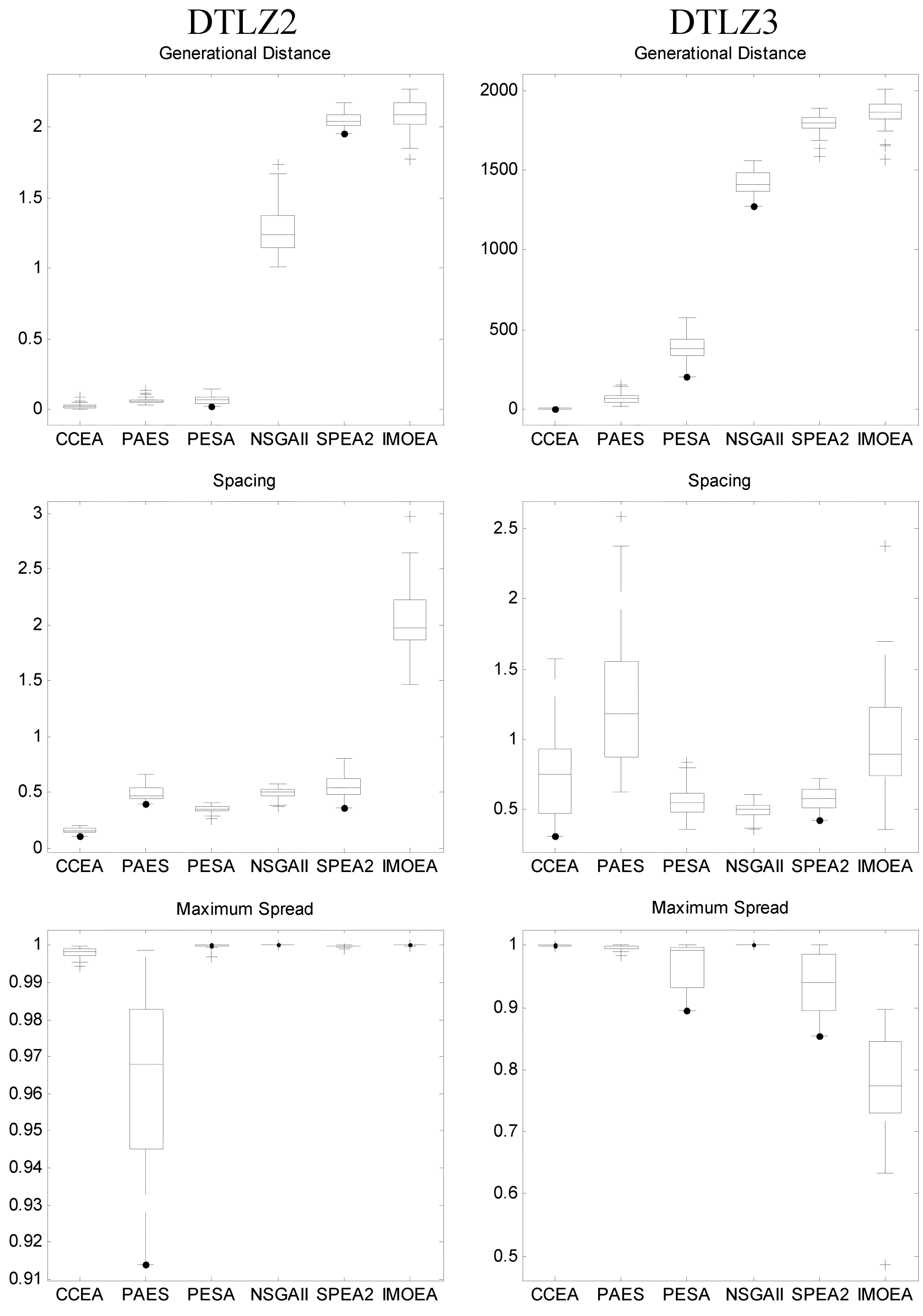
though it gives the best value of HVR. From the definition of the four metrics, there is no correlation among the metrics of GD, S, and MS, while HVR generally correlates to all the other three metrics. The best performance on spacing and the relatively good value of GD and MS resulted in the best value of HVR for CCEA.



(c)

Fig. 8. *(Continued.)* Box plots for the metrics of generational distance *(GD)*, spacing *(S)*, maximum spread *(MS)*, and hypervolume ratio *(HVR)*.

The problems of ZDT4, ZDT5, and ZDT6 are more difficult than ZDT1, ZDT2, and ZDT3. The problem of ZDT4 has 21^9 local Pareto fronts, which challenge the algorithms' ability to escape from local optima. Fig. 8(b) shows that CCEA manages to find the global Pareto front of ZDT4, while other MOEAs have been trapped at local optima, e.g., the solutions of PAES, PESA, NSGA II, SPEA 2, and IMOEA do not cover the entire global Pareto front well, resulting in the poor performance on MS and HVR. ZDT5 is the only discrete problem in the test suite. It is deceptive, i.e., most search space leads to a local Pareto front while the global Pareto front is isolated. As shown in Fig. 8(b), the CCEA manages to escape from the local Pareto front of ZDT5. The nonuniform distribution of solutions makes ZDT6 a difficult problem to tackle by most MOEAs. The CCEA again performs better than other MOEAs for the problem of ZDT6, as shown in Fig. 8(b). These results illustrate that the cooperative



(d)

Fig. 8. *(Continued.)* Box plots for the metrics of generational distance *(GD)*, spacing *(S)*, maximum spread *(MS)*, and hypervolume ratio *(HVR)*.

coevolution approach is capable of evolving individuals toward the global Pareto front effectively.

There is a high interaction among the variables in FON and KUR, and every variable has a family of optima. For the problem of FON, CCEA has similar performance to other algorithms in terms of GD, as shown in Fig. 8(c), e.g., its spacing is good for the median but large in the variance. For the problem of KUR, CCEA performs poorly on GD but excellently on spacing. For

TABLE III
MEDIAN GENERATIONAL DISTANCE FOR 30 RUNS BY CCEA WITH DIFFERENT CLONE NUMBERS IN THE EXTENDING OPERATOR

Problem	***c* = 0**	***c* = 1**	***c* = 2**
ZDT1	7.43E-04	4.04E-04	4.82E-04
ZDT2	1.01E-03	6.35E-04	6.04E-04
ZDT3	2.42E-03	1.27E-03	1.28E-03
ZDT4	6.26E-04	1.48E-04	2.87E-05
ZDT5	1.26E-06	1.26E-06	1.26E-06
ZDT6	1.14E-02	4.86E-07	4.85E-07
FON	2.85E-02	2.28E-02	2.20E-02
KUR	2.92E-02	2.35E-02	3.48E-02
TLK	4.86E-01	4.90E-01	4.79E-01
DTLZ2	4.48E-02	2.08E-02	1.22E-02
DTLZ3	3.07E+00	1.65E+00	1.81E+00

TABLE IV
MEDIAN SPACING FOR 30 RUNS BY CCEA WITH DIFFERENT CLONE NUMBERS IN THE EXTENDING OPERATOR

Problem	***c* = 0**	***c* = 1**	***c* = 2**
ZDT1	0.1902	0.1667	0.1617
ZDT2	0.1827	0.1600	0.1658
ZDT3	0.6137	0.2733	0.2841
ZDT4	0.1793	0.1409	0.1383
ZDT5	0.7125	0.7125	0.7125
ZDT6	0.3089	0.1881	0.1572
FON	0.9750	0.5291	0.4598
KUR	0.7707	0.7154	0.7451
TLK	1.0285	1.0135	1.0261
DTLZ2	0.1856	0.1604	0.1604
DTLZ3	1.5017	0.7525	0.5451

the problem of TLK with noise on parameters, all the algorithms except for PAES achieve a similar performance. There are five objectives to optimize for the problems of DTLZ2 and DTLZ3, which are used to evaluate the performance of MOEAs in producing adequate pressure for driving the evolution of individuals toward the large Pareto front in the high-dimensional objective domain. Fig. 8(d) shows that the CCEA scales well with PAES and PESA, while NSGAII, SPEA2, and IMOEA suffer in convergence toward the optimal Pareto front.

Generally, CCEA produces excellent performance in escaping from local optima in MO optimization. For the metric of spacing, CCEA shows a distinct advantage over other MOEAs, which demonstrates the capability of dynamic sharing and extending operator in maintaining diversity along the evolution. For the metric of maximum spread and hypervolume ratio, CCEA performs competitively in exploring the spread of nondominated individuals in most test problems, as shown in Fig. 8(a)–(c).

2) Effect of Extending Operator: To examine the effectiveness of extending operator, simulation of CCEA without extending operator (clone number $c = 0$), CCEA with extending operator ($c = 1$), and CCEA with extending operator ($c = 2$) was run for 30 times, respectively, for all the test problems. Table III lists the median generational distance for the 30 simulation runs. It shows that the extending operator reduces the generational distance effectively, except that no clear improvement is observed for the problem of TLK. This illustrates that the extending operator is able to improve solution diversity, which is essential in helping the algorithm to escape from local optima for MO optimization. It is also observed that one clone is sufficient to achieve good performance for generational distance, i.e., adding more clones does not necessary offer more improvements in this case.

Table IV lists the median spacing for the 30 simulation runs. In all cases, the extending operator improves the performance metric of spacing, as desired. For the problem of ZDT3 with discontinuous Pareto front, there is a significantly drop of median spacing (from 0.6137 to 0.2733) for CCEA with one clone in the extending operator. The extending operator also improves the spacing greatly for the problems of ZDT6, FON, and DTLZ3, which illustrates its effectiveness in improving the smoothness of nondominated individuals. Similarly, one clone is sufficient here for achieving good performance on spacing. In this paper, the subpopulation size is set to 20, which is relatively small for a population-based algorithm. As a result, one clone is sufficient in guiding the search toward less populated regions, while more clones may reduce the diversity of the evolution in this case.

3) Investigation of Cooperation Methods: The model of cooperation among subpopulations is an important issue in cooperative coevolution. Potter and De Jong [33] proposed two cooperation methods in their studies of cooperative coevolution for single-objective optimization. In the first approach, cooperation is performed to build a candidate solution by combining an individual with the best individual of every other subpopulation. However, this cooperation approach is often less competitive for problems with high interaction among variables. In the second approach, an additional cooperation is performed by combining the individual with a random individual of every other subpopulation. The better of the two candidate solutions is retained, i.e., solution of the first cooperation is selected unless it is dominated by the solution of second cooperation.

TABLE V
MEDIAN GD FOR CCEA WITH DIFFERENT COOPERATION METHODS IN 30 RUNS

Problem	C1	C2	C3	C4
ZDT1	4.13E-04	4.24E-04	4.04E-04	3.14E-01
ZDT2	4.05E-04	4.81E-04	6.35E-04	5.09E-01
ZDT3	1.45E-03	1.41E-03	1.27E-03	3.64E-01
ZDT4	6.98E-05	7.08E-05	1.48E-04	1.26E+00
ZDT5	1.27E-06	1.27E-06	1.26E-06	2.90E+00
ZDT6	4.89E-07	5.00E-07	4.86E-07	1.43E+00
FON	2.35E-02	2.32E-02	2.28E-02	2.13E-02
KUR	3.55E-02	2.31E-02	2.35E-02	1.76E-02
TLK	4.80E-01	5.23E-01	4.90E-01	5.34E-01
DTLZ2	8.57E-04	4.07E-01	2.08E-02	5.44E-01
DTLZ3	1.82E+00	2.24E+00	1.65E+00	5.82E+02

TABLE VI
MEDIAN SPACING FOR CCEA WITH DIFFERENT COOPERATION METHODS IN 30 RUNS

Problem	C1	C2	C3	C4
ZDT1	0.1757	0.1771	0.1667	0.8240
ZDT2	0.1613	0.1553	0.1600	0.9328
ZDT3	0.2815	0.2867	0.2733	0.9952
ZDT4	0.1333	0.1436	0.1409	1.9624
ZDT5	0.7114	0.7125	0.7125	1.1676
ZDT6	0.1754	0.1822	0.1881	0.8174
FON	0.2687	0.3497	0.5291	0.2676
KUR	0.7919	0.7145	0.7154	0.6817
TLK	1.0905	1.0284	1.0135	1.0772
DTLZ2	0.1245	0.2658	0.1604	0.2929
DTLZ3	0.6638	0.8103	0.7525	0.7201

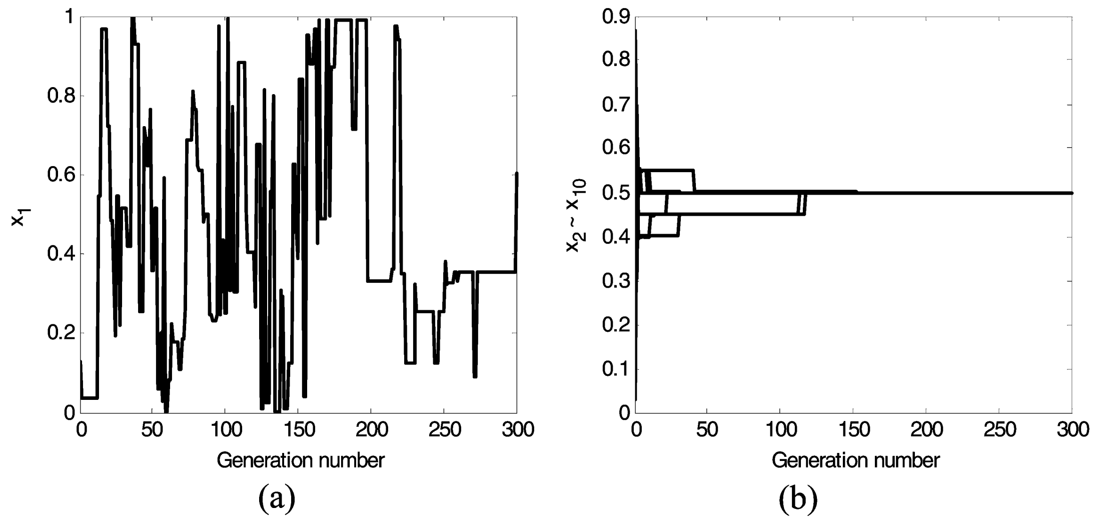


Fig. 9. Evolution of the best individual in every subpopulation for ZDT4. (a) x_1 and (b) x_2 to x_{10}.

In this section, four cooperation methods are implemented in CCEA to study their effectiveness for MO optimization, which are described as follows.

- C1: Each subpopulation selects its best individual as the representative. The evaluation of an individual involves one cooperation, which combines the individual under evaluation with the representative of every other subpopulation.
- C2: Similar to C1, each subpopulation has one representative. Two cooperations are involved in the evaluation of an individual. The first cooperation combines the individual with the representative of every other subpopulation, while the second cooperation combines the individual with a random individual of every other subpopulation. The better of the two candidate solutions is retained, i.e., solution of the first cooperation is selected unless it is dominated by the solution of second cooperation.
- C3: Each subpopulation selects its best two individuals as the representatives. Two cooperations are involved in the evaluation of an individual. The first cooperation combines the individual under evaluation with the best representative of every other subpopulation, while the second cooperation combines the individual with a random representative of every other subpopulation. The better of the two candidate solutions is retained, i.e., solution of the first cooperation is selected unless it is dominated by the solution of second cooperation.
- C4: The best two individuals in each subpopulation are selected as the representatives. The evaluation of an indi-

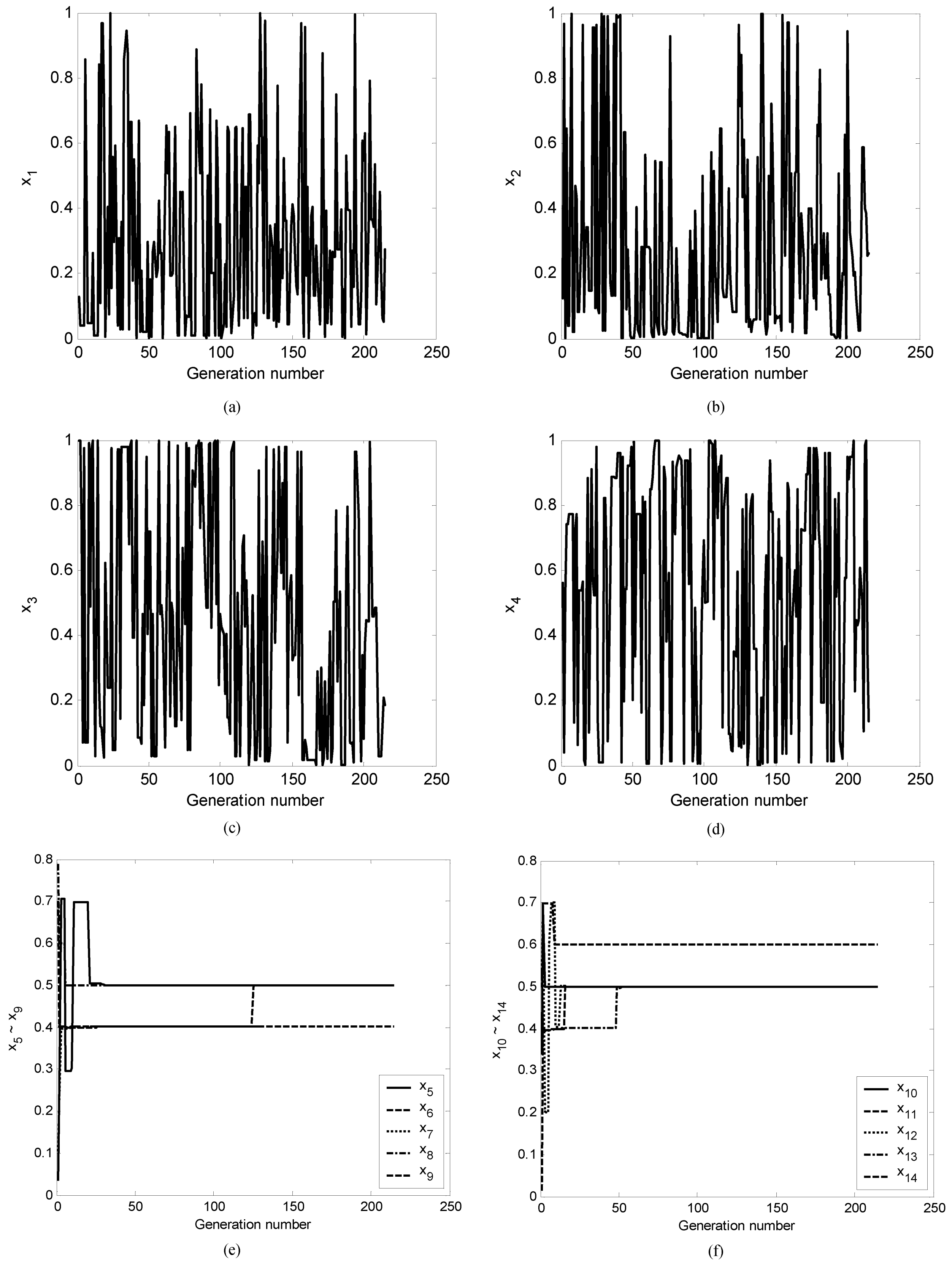
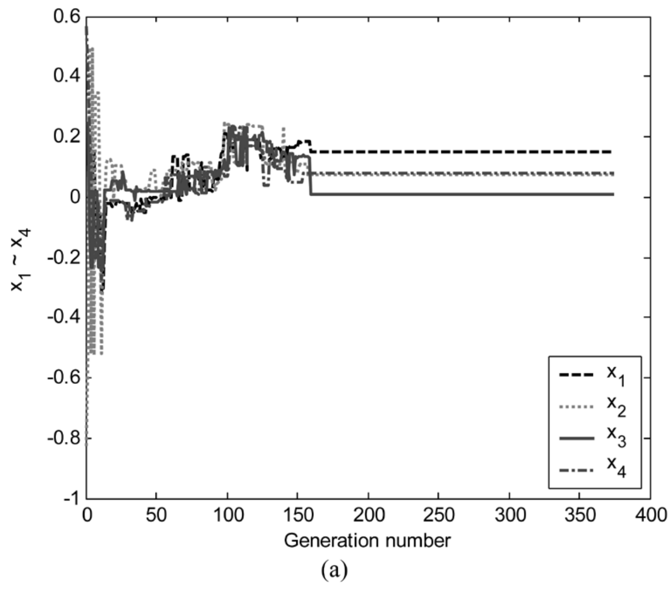


Fig. 10. Evolution of the best individual in every subpopulation for DTLZ3. (a) x_1, (b) x_2, (c) x_3, (d) x_4, (e) x_5 to x_9, and (f) $x_1 0$ to $x_1 4$.

vidual involves one cooperation, which combines the individual under evaluation with a random representative of every other subpopulation.

Tables V and VI list the median generational distance and spacing, respectively, for CCEA with different cooperation methods in 30 runs. It is observed from Table V that C1 per-



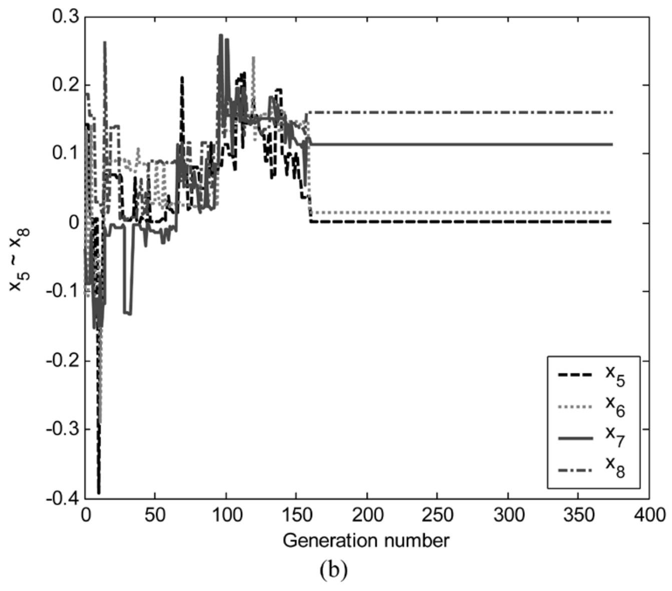


Fig. 11. Evolution of the best individual in every subpopulation for FON. (a) x_1 to $x_4$and (b) x_5 to x_8.

forms the best for ZDT2, ZDT4, TLK, and DTLZ2, while C3 performs the best for ZDT1, ZDT3, ZDT5, ZDT6, and DTLZ3. Although C4 has a slight edge over C1, C2, and C3 for FON and KUR in terms of generational distance, its performance for some of other benchmark problems can be of several orders of magnitude worst. In contrast, C1, C2, and C3 exhibit similar performances for most of the benchmark problems. It can also be observed that C4 performs worse than other methods for ZDT1 to ZDT6 in terms of spacing. With the exception of FON, DTLZ2, and DTLZ3, the distribution of solutions evolved by C1, C2, and C3 is rather similar.

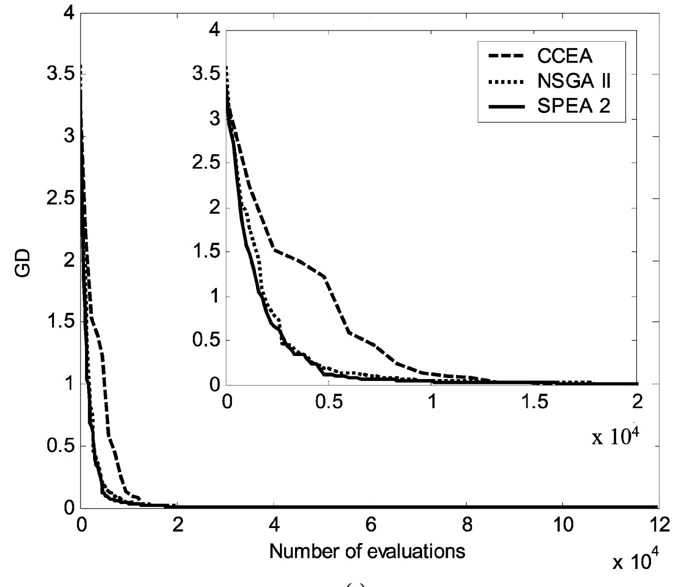
Among the four cooperation methods, C1 is the greediest method that may result in premature convergence in the evolution. It performs well for problems with low variable interactions but provides poor results for problems with high variable interactions. In contrast, C4 is the least greedy approach that is likely to maintain diverse solutions in the evolution. It performs well for problems with high variable interactions but gives poor results for problems with low variable interactions. C2 and C3 provide a balance between greedy search and diversity maintenance, which achieve generally good and robust performances according to our simulations.

4) Dynamic Behavior of CCEA: This section examines the various dynamic behaviors of CCEA in order to better understand its evolution of individuals. Figs. 9–11 show the evolution of the best individual in every subpopulation. For the problem of ZDT4, x_1 has many oscillations along the evolution, while x_2 to x_{10} converge to the value of zero gradually. In particular, x_1 experiences several oscillations even after x_2 to x_{10} have converged in the evolution. Such oscillations of x_1 help to sample the solutions of ZDT4 along the Pareto front uniformly, such that the final solution set covers the entire spectrum of ZDT4, i.e., $x_1 \in [0,1]$, $x_2 = \cdots = x_{10} = 0$. For the problem of DTLZ3, x_1 to x_4 oscillate continuously notwithstanding if x_5 to x_{14} converge in the evolution. Although CCEA produces excellent results for DTLZ3, as shown in Fig. 8(d), not all variables in x_5 to x_{14} converge to their optimal values due to the high-dimensional objective space in DTLZ3. Unlike ZDT4 and DTLZ3, all the variables in x_1 to x_8 oscillate and converge in a similar pace for the problem of FON, as shown in Fig. 11. This behavior is related to the high interaction among variables in FON, where the Pareto optimal set consists of points in the range of $(-1/\sqrt{8}) \leq x_1 = x_2 = \cdots = x_8 \leq (1/\sqrt{8})$. Compared to other MOEAs, CCEA has shown good capability in escaping from local optima when the interdependency among variables is low, as can be seen in Figs. 9(b), 10(e), and 10(f).

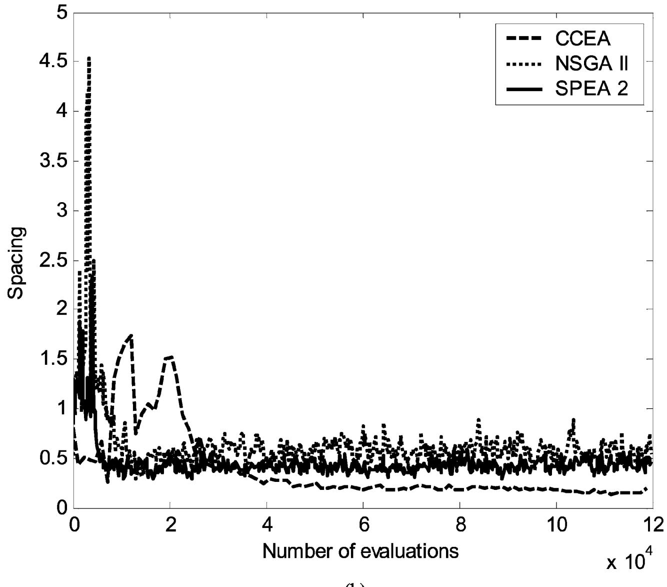
Figs. 12–14 show the evolutionary process of CCEA (with cooperation method of C3), SPEA 2, and NSGA II based upon the number of function evaluations for various test problems. On the metric of GD, CCEA converges slower than NSGA II and SPEA 2 for ZDT1, but faster for FON. It has similar convergence rate to that of NSGA II and SPEA 2 for ZDT4. In general, the convergence rate of CCEA is similar to that of NSGA II and SPEA 2 with respect to the number of function evaluations. The cooperation method of C1 and C4 will accelerate and slow down the convergence of CCEA, respectively. As listed in Table II, SPEA2 and NSGAII perform 100 function evaluations at each generation, while CCEA with C3 performs 1200, 400, and 320 function evaluations at each generation for ZDT1, ZDT4, and FON, respectively. If the comparison is based upon the number of generations, CCEA achieves the fastest convergence among the three algorithms. It can be observed from Figs. 12–14 that the variation of spacing is relevant to GD, i.e., the turning points of GD correspond to the peaks of spacing. Generally, a turning point means that the algorithm escapes or jumps from one local optima to another. In the intermediate state of jumping, there is a transition from one relatively uniform distribution of individuals along a local Pareto front to another, which results in an impulse on the metric of spacing.

D. Simulation Results of DCCEA

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

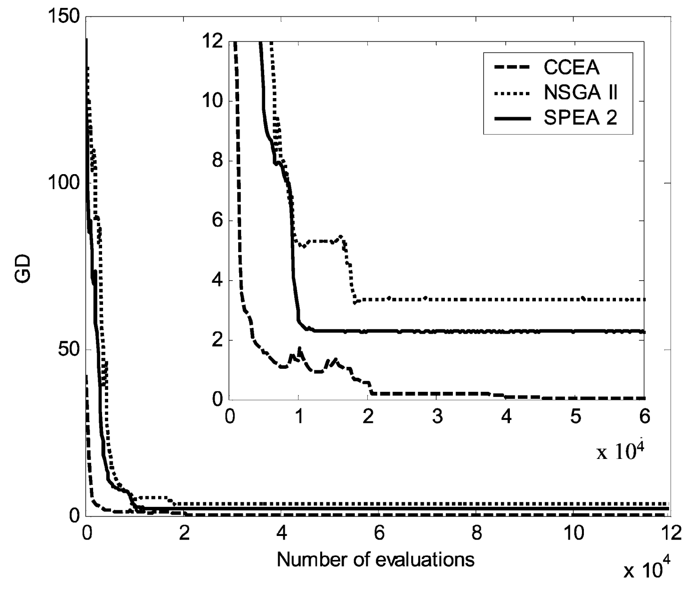


(a)

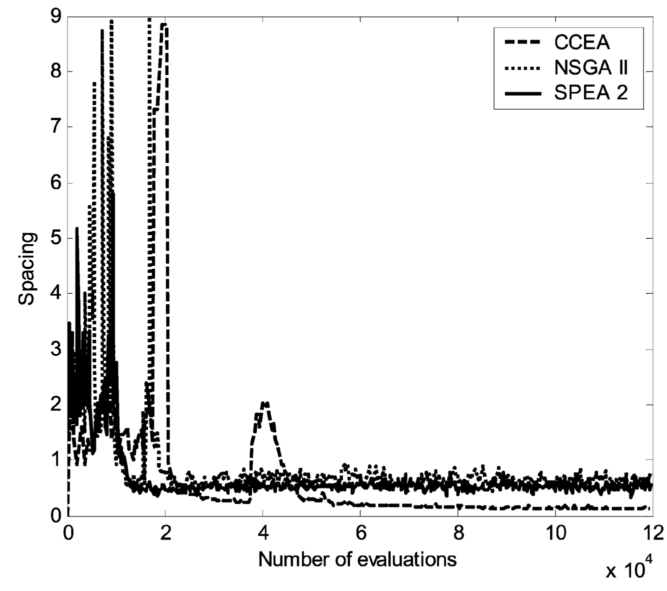


(b)

Fig. 12. The evolution of CCEA, NSGA II, and SPEA2 for ZDT1.



(a)

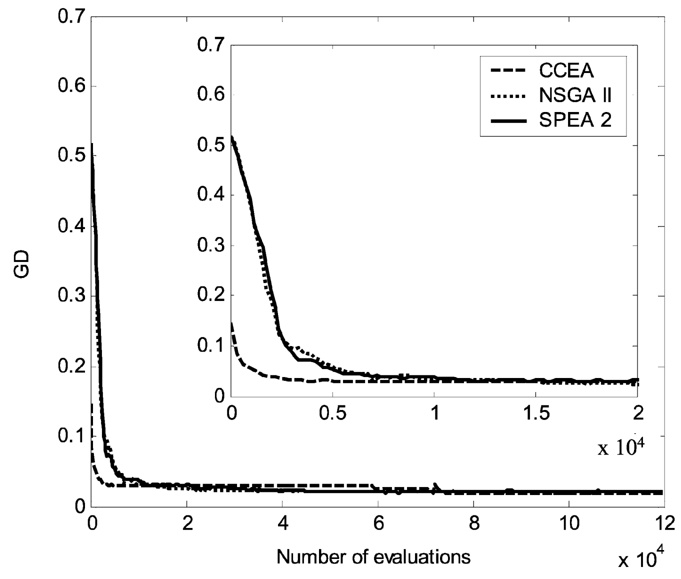


(b)

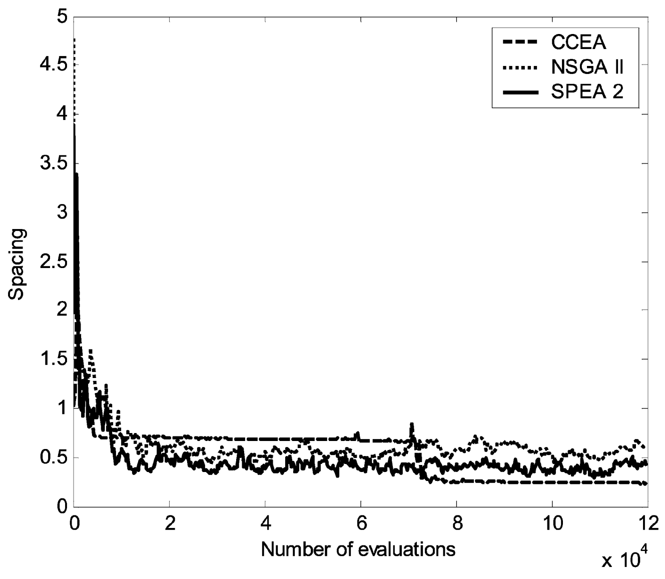
Fig. 13. The evolution of CCEA, NSGA II, and SPEA2 for ZDT4.

configuration of DCCEA consists of 11 PCs connected in a campus local area network environment. Table VII lists the configuration of the 11 PCs, where the server runs on the PIV 1600/512 PC and the peers are run on other PCs. The configuration of parameters for DCCEA is shown in Table VIII.

1) DCCEA Performance: To minimize the bias in the simulation results, 30 independent runs with randomly generated initial populations were performed. The median speedup of the 30 runs for different test problems is listed in Table IX, while the actual median runtime for each benchmark problem is plotted in Fig. 15. The performances of DCCEA (with cooperation method of C4) in GD, spacing, MS, and HVR are summarized in Fig. 16. It is observed from Fig. 15 that the median runtime is reduced as the number of peers is increased. Table IX illustrates that the speedup achievable by DCCEA depends on both the number of peers and the benchmark problem. For instance, the speedup achievable is more significant for large problems according to our simulations. In particular, DCCEA achieves a speedup of 3.46 for ZDT1, but only a speedup of 1.29 is obtained for ZDT6. When the number of peers is more than 6, the increased communication cost counteracts the reduction of computation cost, as shown in Fig. 15, i.e., the saturation for the speedup of DCCEA is reached for the test problems. It can be observed from Fig. 16 that the median metrics for the five test problems have no distinct changes (in spite of some minor fluctuations on the curves) as the number of peers is increased. This illustrates that the DCCEA is effective in reducing the simulation runtime without sacrificing the performance of CCEA as the number of peers is increased.



(a)



(b)

Fig. 14. The evolution of CCEA, NSGA II, and SPEA2 for FON.

TABLE VII
SIMULATION 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 VIII
CONFIGURATION OF PARAMETERS FOR DCCEA

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

TABLE IX
MEDIAN SPEEDUP OF 30 RUNS WITH RESPECT TO THE NUMBER OF PEERS

Number of peers	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
2	1.52	1.70	1.47	1.23	1.01
3	2.01	1.99	1.88	1.47	1.11
4	2.25	2.21	1.95	1.50	1.14
5	2.48	2.69	2.15	1.56	1.14
6	2.81	3.03	2.83	1.70	1.29
7	2.87	3.32	2.77	1.88	1.25
8	3.38	3.27	2.92	1.82	1.26
9	3.46	3.36	2.96	1.83	1.26
10	3.46	3.18	2.79	1.82	1.25

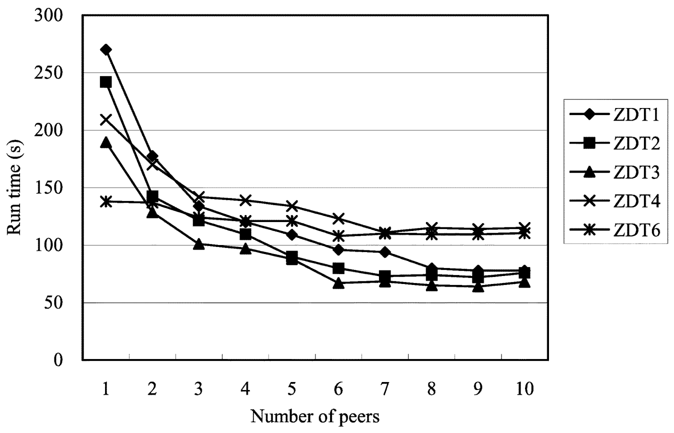


Fig. 15. Median runtime of 30 runs with respect to the number of peers.

2) *Effect of Exchange and Synchronization Interval:* In order to examine the effect of exchange and synchronization intervals, simulations with different interval settings, as shown in Table X, were carried out for 30 times, respectively, on ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6. The simulation environment here consists of five PIV 2800/512 PCs connected in a campus network. The runtime for different settings of synchronization and exchange intervals is plotted in Figs. 17 and 18, respectively. Figs. 19 and 20 show the performances of DCCEA (with cooperation method of C4) of the 30 simulation runs for different settings of synchronization and exchange intervals. When the exchange or synchronization interval is reduced, the increased communication overhead will inevitably lead to higher runtime. It is observed from Figs. 17 and 18 that there is no significant impact on the runtime when the respective interval is increased beyond 10 generations. Apart from some small fluctuations, the median metrics plotted in Fig. 19 have no distinct changes for the five test problems as the synchronization interval is varied. This behavior can be attributed to the fact that the computers used in this experimental

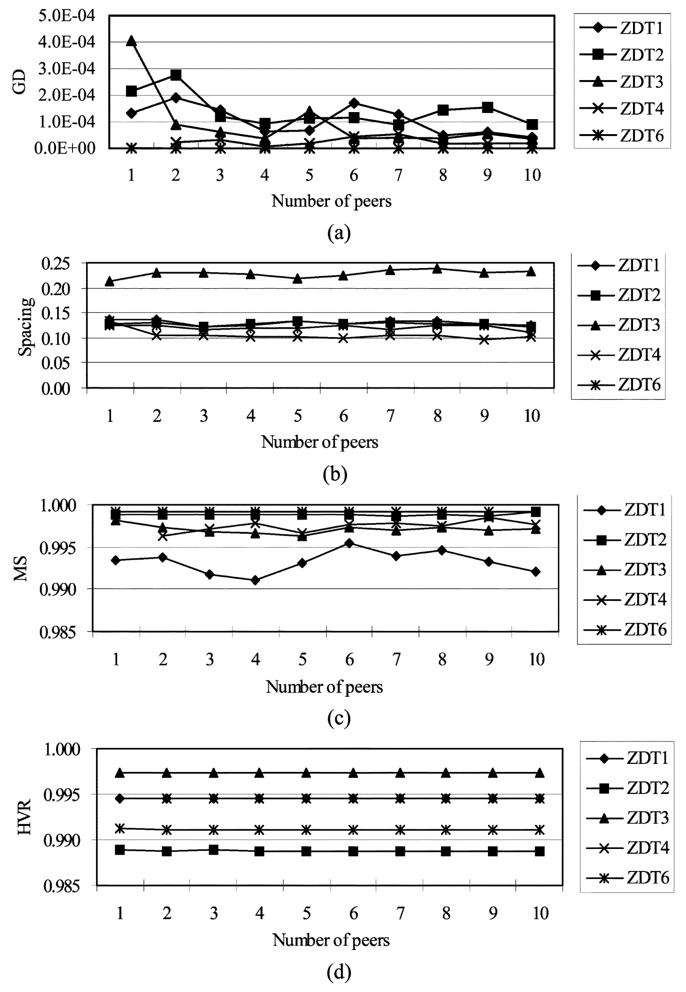


Fig. 16. Median metric of 30 runs with respect to the number of peers.

TABLE X
DIFFERENT SETTINGS OF SYNCHRONIZATION AND EXCHANGE INTERVALS

	Exchange interval = 10 generations			
Synchronization interval	1	5	10	15
	Synchronization interval = 5 generations			
Exchange interval	1	10	20	30

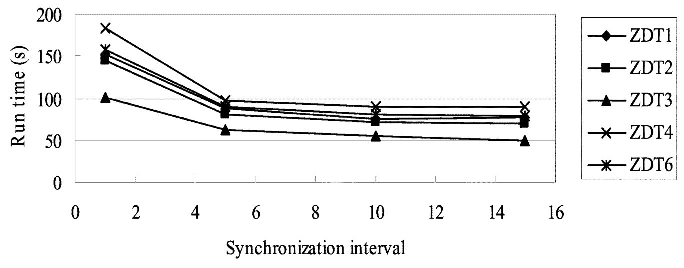


Fig. 17. Median runtime of 30 runs with respect to synchronization interval.

setup have similar capabilities and processing speeds. Although different exchange intervals also exhibit little performance variation, Fig. 20 shows a slight relation between the exchange interval and DCCEA performance. In particular, Fig. 20(a) and (b) depicts a minor deterioration in performance for ZDT1, ZDT2, and ZDT3 when the exchange interval is increased. The exchange interval represents the frequency in which the archive

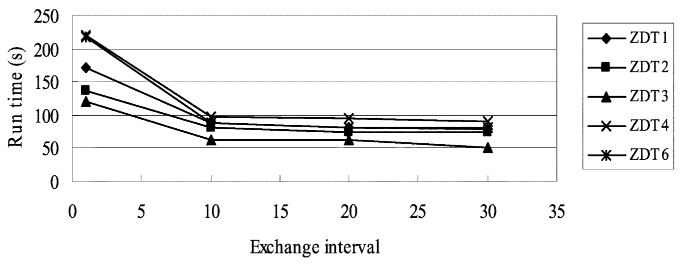


Fig. 18. Median runtime of 30 runs with respect to exchange interval.

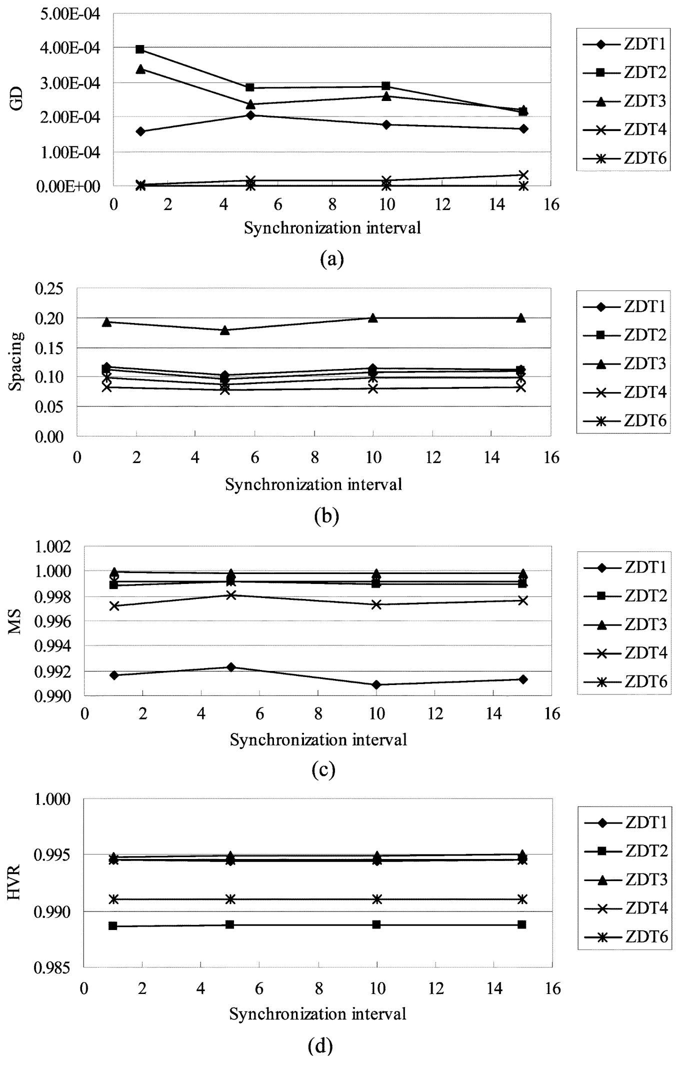


Fig. 19. Median metrics of 30 runs with respect to synchronization interval.

is updated, thus it has a direct impact on the convergence of the algorithm.

V. CONCLUSION

This paper has presented a cooperative coevolutionary algorithm for MO optimization. It applied the divide-and-conquer mechanism to decompose decision vectors into small components and evolved multiple solutions in the form of cooperative subpopulations. Incorporated with various features like archiving, dynamic sharing, and extending operator in cooperative coevolution, the CCEA is capable of maintaining

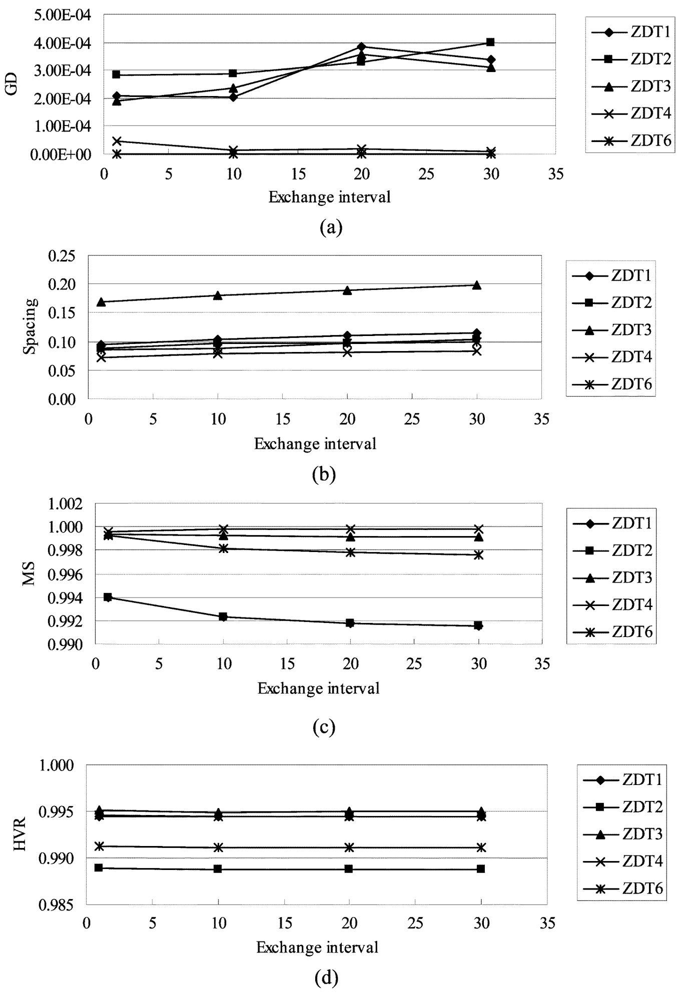


Fig. 20. Median metrics of 30 runs with respect to exchange interval.

archive diversity in the evolution and distributing the solutions uniformly along the Pareto front. The extensive quantitative and qualitative comparisons with various MOEAs on 11 benchmark problems show that CCEA produced competitive and robust results in finding the tradeoff solutions. Exploiting the inherent parallelism of cooperative coevolution, a distributed CCEA paradigm has been implemented for concurrent processing to expedite the computational speed by sharing the workload among multiple networked computers. The computational results show that DCCEA can dramatically reduce the simulation runtime without sacrificing the performance of CCEA as the number of peers is increased.

REFERENCES

[1] P. J. Angeline and J. B. Pollack, "Competitive environments evolve better solutions for complex tasks," in *Proc. 5th Int. Conf. Genetic Algorithms*, 1993, pp. 264–270.

[2] G. Ascia, V. Catania, and M. Palesi, "A GA-based design space exploration framework for parameterized system-on-a-chip platforms," *IEEE Trans. Evol. Comput.*, vol. 8, pp. 329–346, 2004.

[3] E. Cantú-Paz, "A survey of parallel genetic algorithms," *Calculateurs Paralleles, Reseaux et Systems Repartis*, vol. 10, no. 2, pp. 141–171, 1998.

[4] D. W. Corne, J. D. Knowles, and M. J. Oates, "The Pareto envelope-based selection algorithm for multiobjective optimization," in *Proc. Parallel Problem Solving Nature VI Conf.*, 2000, pp. 839–848.

[5] J. M. Chambers, W. S. Cleveland, B. Kleiner, and P. A. Turkey, *Graphical Methods for Data Analysis*. Pacific Grover, CA: Wadsworth Brooks/Cole, 1983.

[6] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*. Norwell, MA: Kluwer, 2002.

[7] P. J. Darwen and X. Yao, "Speciation as automatic categorical modularization," *IEEE Trans. Evol. Comput.*, vol. 1, pp. 100–108, 1997.

[8] K. Deb, "Multiobjective genetic algorithms: Problem difficulties and construction of test problem," *Evol. Comput.*, vol. 7, no. 3, pp. 205–230, 1999.

[9] ——, *Multiobjective Optimization using Evolutionary Algorithms*. New York: Wiley, 2001.

[10] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, pp. 182–197, 2002.

[11] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multiobjective optimization test problems," in *Proc. Congr. Evol. Comput.*, 2002, pp. 825–830.

[12] K. Deb, P. Zope, and A. Jain, "Distributed computing of Pareto-optimal solutions with evolutionary algorithms," in *Proc. 2nd Int. Conf. Evolutionary Multi-Criterion Optimization*, 2003, pp. 534–549.

[13] M. Farina, K. Deb, and P. Amato, "Dynamic multiobjective optimization problems: Test cases, approximations, and applications," *IEEE Trans. Evol. Comput.*, vol. 8, pp. 425–442, 2004.

[14] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," in *Proc. 5th Int. Conf. Genetic Algorithms*, 1993, pp. 416–423.

[15] ——, "Multiobjective genetic algorithms made easy: selection, sharing and mating restriction," in *Proc. 1st Int. Conf. Genetic Algorithms Engineering Systems: Innovations Applications*, 1995, pp. 42–52.

[16] D. E. Goldberg, "Sizing populations for serial and parallel genetic algorithms," in *Proc. 3rd Int. Conf. Genetic Algorithms*, San Mateo, CA, 1989, pp. 70–79.

[17] S. Y. Ho, L. S. Shu, and J. H. Chen, "Intelligent evolutionary algorithms for large parameter optimization problems," *IEEE Trans. Evol. Comput.*, vol. 8, pp. 522–541, 2004.

[18] C. L. Hwang and A. S. Masud, *Multiple Objective Decision Making—Methods and Applications*. Berlin, Germany: Springer-Verlag, 1979.

[19] A. Jaszkiewicz, "Do multiple-objective metaheuristics deliver on their promises? A computational experiment on the set-covering problem," *IEEE Trans. Evol. Comput.*, vol. 7, pp. 133–143, 2003.

[20] M. T. Jensen, "Reducing the run-time complexity of multiobjective EAs: The NSGA-II and other algorithms," *IEEE Trans. Evol. Comput.*, vol. 7, pp. 503–515, 2003.

[21] N. Keerativuttiumrong, N. Chaiyaratana, and V. Varavithya, "Multiobjective cooperative coevolutionary genetic algorithm," in *Proc. Parallel Problem Solving from Nature VII Conf.*, 2002, pp. 288–297.

[22] V. Khare, X. Yao, and K. Deb, "Performance scaling of multi-objective evolutionary algorithms," in *Proc. 2nd Int. Conf. Evolutionary Multi-Criterion Optimization*, 2003, pp. 376–390.

[23] V. Khare, X. Yao, and B. Sendhoff, "Credit assignment among neurons in co-evolving populations," in *Proc. Parallel Problem Solving from Nature VIII Conf.*, 2004, pp. 882–891.

[24] T. Khoshgoftaar, Y. Liu, and N. Seliya, "A multiobjective module-order model for software quality enhancement," *IEEE Trans. Evol. Comput.*, vol. 8, pp. 593–608, 2004.

[25] J. D. Knowles and D. W. Corne, "Approximating the nondominated front using the Pareto archived evolution strategy," *Evol. Comput.*, vol. 8, no. 2, pp. 149–172, 2000.

[26] F. Kursawe, "A variant of evolution strategies for vector optimization," in *Proc. Parallel Problem Solving from Nature 1st Workshop*, vol. 496, 1991, pp. 193–197.

[27] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi, "Scaling up fast evolutionary programming with cooperative coevolution," in *Proc. Congr. Evol. Comput.*, 2001, pp. 1101–1108.

[28] J. D. Lohn, W. F. Kraus, and G. L. Haith, "Comparing a coevolutionary genetic algorithm for multiobjective optimization," in *Proc. Congr. Evol. Comput.*, 2002, pp. 1157–1162.

[29] H. Lu and G. G. Yen, "Rank-density-based multiobjective genetic algorithm and benchmark test function study," *IEEE Trans. Evol. Comput.*, vol. 7, pp. 325–343, 2003.

[30] D. E. Moriarty, "Symbiotic evolution of neural networks in sequential decision tasks," Ph.D. dissertation, Univ. of Texas at Austin, 1997.

[31] M. Neef, D. Thierens, and H. Arciszewski, "A case study of a multiobjective recombinative genetic algorithm with coevolutionary sharing," in *Proc. Congr. Evol. Comput.*, 1999, pp. 796–803.

[32] I. C. Parmee and A. H. Watson, "Preliminary airframe design using co-evolutionary multiobjective genetic algorithms," in *Proc. Genetic Evol. Comput. Conf.*, 1999, pp. 1657–1665.
[33] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *Proc. Parallel Problem Solving from Nature III Conf.*, 1994, pp. 249–257.
[34] ——, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evol. Comput.*, vol. 8, no. 1, pp. 1–29, 2000.
[35] W. Rivera, "Scalable parallel genetic algorithms," *Artif. Intell. Rev.*, vol. 16, pp. 153–168, 2001.
[36] C. D. Rosin and R. K. Belew, "New methods for competitive coevolution," *Evol. Comput.*, vol. 5, no. 1, pp. 1–29, 1997.
[37] J. D. Schaffer, "Multiple-objective optimization using genetic algorithms," in *Proc. 1st Int. Conf. Genetic Algorithms*, 1985, pp. 93–100.
[38] S. Y. Shin, I. H. Lee, D. Kim, and B. T. Zhang, "Multiobjective evolutionary optimization of DNA sequences for reliable DNA computing," *IEEE Trans. Evol. Comput.*, vol. 9, pp. 143–158, 2005.
[39] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evol. Comput.*, vol. 2, no. 3, pp. 221–248, 1994.
[40] *J2EE tutorial*: Sun Microsystems, 2001.
[41] K. C. Tan, T. H. Lee, and E. F. Khor, "Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 5, pp. 565–588, 2001.
[42] ——, "Evolutionary algorithms for multiobjective optimization: Performance assessments and comparisons," *Artif. Intell. Rev.*, vol. 17, no. 4, pp. 251–290, 2002.
[43] K. C. Tan, A. Tay, and J. Cai, "Design and implementation of a distributed evolutionary computing software," *IEEE Trans. Syst., Man, Cybern. C*, vol. 33, no. 3, pp. 325–338, 2003.
[44] K. C. Tan, E. F. Khor, T. H. Lee, and R. Sathikannan, "An evolutionary algorithm with advanced goal and priority specification for multiobjective optimization," *J. Artif. Intell. Res.*, vol. 18, pp. 183–215, 2003.
[45] K. C. Tan, E. F. Khor, and T. H. Lee, *Multiobjective Evolutionary Algorithms and Applications*. Berlin, Germany: Springer-Verlag, 2005.
[46] F. Van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, pp. 225–239, 2004.
[47] D. A. Van Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithm research: A history and analysis," Dept. Elec. Comput. Eng., Air Force Inst. Technology, OH, TR-98-03, 1998.
[48] ——, "Multiobjective evolutionary algorithm test suites," in *Symp. Applied Computing*, San Antonio, TX, 1999, pp. 351–357.
[49] D. A. Van Veldhuizen, J. B. Zydallis, and G. B. Lamont, "Considerations in engineering parallel multiobjective evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 7, pp. 144–173, 2003.
[50] G. G. Yen and H. Lu, "Dynamic multiobjective evolutionary algorithm: adaptive cell-based rank and density estimation," *IEEE Trans. Evol. Comput.*, vol. 7, pp. 253–274, 2003.
[51] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, pp. 257–271, 1999.
[52] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evol. Comput.*, vol. 8, no. 2, pp. 173–195, 2000.
[53] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," Computer Engineering and Networks Laboratory (TIK), Swiss Federal Inst. Technology (ETH), Zurich, Switzerland, 103, 2001.
[54] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Trans. Evol. Comput.*, vol. 7, pp. 117–132, 2003.

K. C. Tan received the B.Eng. degree with first class honors in electronics and electrical engineering and the Ph.D. degree from the University of Glasgow, Glasgow, Scotland, U.K., in 1994 and 1997, respectively.

He is currently an Associate Professor in the Department of Electrical and Computer Engineering, National University of Singapore. He has authored or coauthored more than 130 journal and conference publications and has served as a Program Committee or Organizing Member for many international conferences. His current research interests include computational intelligence, evolutionary computing, and engineering design optimization.

Dr. Tan is an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION.

Y. J. Yang received the B.Eng. degree from Tsinghua University, Beijing, China, in 1998 and the M.Eng. degree in electrical and computer engineering from the National University of Singapore in 2004.

His research interests include evolutionary computation, multiobjective optimization, and production scheduling.

C. K. Goh received the B.Eng. degree (Hon.) in electrical engineering from the National University of Singapore in 2003. He is currently working towards the Ph.D. degree at the Centre for Intelligent Control, University of Singapore.

His current research interests include evolutionary computation and neural networks, specifically in the application of evolutionary techniques for multiobjective optimization.