This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2017.2676000, IEEE Transactions on Industrial Informatics

A Distributed Parallel Cooperative Coevolutionary Multi-Objective Evolutionary Algorithm for Large-Scale Optimization

Bin Cao, *Member, IEEE,* Jianwei Zhao, Zhihan Lv, *Member, IEEE,* and Xin Liu

***Abstract*—A considerable amount of research has been devoted to multi-objective optimization problems (MOPs). However, few studies have aimed at multiobjective large-scale optimization problems (MOLSOPs). To address MOLSOPs, which may involve big data, this paper proposes a message passing interface (MPI)-based distributed parallel cooperative coevolutionary multi-objective evolutionary algorithm (DPCCMOEA). DPCCMOEA tackles MOLSOPs based on decomposition. First, based on a modified variable analysis method, we separate decision variables into several groups, each of which is optimized by a subpopulation (species). Then, the individuals in each subpopulation are further separated to several sets. DPCCMOEA is implemented with MPI distributed parallelism and a two-layer parallel structure is constructed. We examine the proposed algorithm using the multi-objective test suites DTLZ and WFG. In comparison with cooperative coevolutionary generalized differential evolution 3 (CCGDE3) and multi-objective evolutionary algorithm based on decision variable analyses (MOEA/DVA), which are state-of-the-art cooperative coevolutionary multi-objective evolutionary algorithms (CCMOEAs), experimental results show that the novel algorithm has better performance in both optimization results and time consumption.**

***Index Terms*—decomposition, variable grouping, cooperative coevolution, large-scale optimization, message passing interface (MPI), distributed parallelism.**

I. Introduction

In real world situations, many objectives have to be fulfilled [1]. For example, in the case that an online store recommends merchandise to customers, both the accuracy and diversity should be considered to satisfy multiple consumer demands [2]. Similar kinds of big data problems can be treated as multi-objective optimization problems (MOPs), and there are always conflicts among these objectives.

Many researchers have tried to solve MOPs through mathematical models [3]. However, it is difficult to find accurate mathematical models. Moreover, these methods can only produce a single solution in one run, and the majority of them are sensitive to the shape of the Pareto front (PF) (e.g., concave or discontinuous). Nevertheless, multi-objective evolutionary algorithms (MOEAs) [4] are able to generate a set of possible solutions in a single run, and they can cope with the complicated shapes of the PF. In addition, MOEA is easy to implement. MOEA has been widely used in many fields, such as path planning of unmanned air vehicles (UAVs) [5], data mining [6], machine learning [7], software engineering [8], etc.

There are many kinds of MOEAs [9], [10], among which nondominated sorting genetic algorithm II (NSGA-II) [11] and multiobjective evolutionary algorithm based on decomposition (MOEA/D) [12] are well known. NSGA-II is based on Pareto dominance, while MOEA/D is based on decomposition. Based upon these two algorithms, many algorithms have been developed [10], [13].

To solve MOPs, MOEAs have to conduct a large number of evolutions, which is very time-consuming. When the complexity of an MOP increases, more time will be taken. Tan et al. [14] puts forward a distributed parallel cooperative coevolutionary algorithm (DCCEA) for multiobjective optimization. In the work of [15], the competitive and cooperative co-evolutionary multi-objective particle swarm optimization algorithm (CCPSO) is proposed. Due to the utilization of distributed parallelism, the optimization time is reduced. With the advent of big data, more and more information is generated. When the amount of data is large, the MOPs can be called big data optimization problems, which can be also called multiobjective large-scale optimization problems (MOLSOPs). However, in [14] and [15], the number of variables is few (no more than 30), and the effectiveness for MOLSOPs is unclear. For large-scale optimization, only using serial algorithms and a single personal computer is very time-consuming and very likely to result in overflow, etc. At this time, distributed evolutionary algorithms (dEAs) [16] will be more suitable. For high-dimensional problems, the strategies of variable grouping and cooperative coevolution (CC) [17]–[19] contribute to better optimization performance. The available grouping methods include fixed grouping [18], [20], random grouping [21], [22], the Delta method [23], dynamic grouping [17], differential grouping [24], global differential grouping [25], graph-based differential grouping (gDG) [26], etc. Cooperative coevolutionary generalized differential evolution 3 (CCGDE3) [20] employs fixed grouping and effectively solves MOPs with up to 5000 variables. Through decision variable analyses (DVA),

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant No. 61303001, in part by Special Program for Applied Research on Super Computation of the NSFC-Guangdong Joint Fund (the second phase), and in part by Foundation of Key Laboratory of Machine Intelligence and Advanced Computing of the Ministry of Education under Grant No. MSC-201602A. (Corresponding authors: Zhihan Lv, Bin Cao)

Bin Cao, and Jianwei Zhao are with the School of Computer Science and Engineering, Hebei University of Technology, Tianjin, 300401, China; Key Laboratory of Machine Intelligence and Advanced Computing (Sun Yat-sen University), Ministry of Education; Hebei Provincial Key Laboratory of Big Data Calculation, China. Xin Liu is with Hebei University of Technology, Tianjin, 300401, China (email: caobin@scse.hebut.edu.cn)

Zhihan Lv is with the Department of Computer Science, University College London, 66-72 Gower Street, London WC1E 6EA, United Kingdom. (email: z.lu@ucl.ac.uk)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2017.2676000, IEEE Transactions on Industrial Informatics

the multi-objective evolutionary algorithm based on decision variable analyses (MOEA/DVA) [27] decomposes an MOP into a set of simpler and low-dimensional sub-problems, and it outperforms several state-of-the-art algorithms. However, CCGDE3 and MOEA/DVA are executed in serial, which is too time-consuming.

The message passing interface (MPI) [28], [29] is a useful tool for high performance computation (HPC). MPI is suitable for coarse-grained parallelism and complex objective functions. In this paper, we put forward a novel MPI-based distributed parallel cooperative coevolutionary multiobjective evolutionary algorithm (DPCCMOEA) for large-scale optimization. The characteristics of the proposed algorithm are summarized as follows:

1) Based on the DVA strategy in MOEA/DVA, we propose a modified variable analysis strategy and decompose the large number of variables to several groups. Each variable group is optimized by a subpopulation (species). These species cooperate to better optimize the MOLSOP.
2) With the help of MPI, we implement the algorithm in a distributed platform. To increase the parallelism degree and to realize the adaptive implementation to different number of CPUs, each species is further decomposed to multiple subspecies (sets), the number of which is determined by the available number of CPUs. Therefore, the computation time is greatly reduced.

The remainder of this paper is organized as follows. In Section II, we give some preliminary knowledge to better understand this paper. Section III describes the proposed algorithm: DPCCMOEA. The parallelism implementation is in Section IV. Section V is the experimental results and analyses. Finally, we conclude this paper in Section VI.

II. Preliminaries

A. Variable Analysis and Grouping

In CCGDE3, fixed grouping is utilized. The large number of variables are randomly separated into several groups, which are fixed during the evolution. In MOEA/DVA, the variables are classified as follows:

- *distance variables*: affecting convergence only
- *position variables*: affecting diversity only
- *mixed variables*: affecting convergence and diversity both

In addition, variable dependencies are also obtained, which can be described as follows. For a solution vector $\boldsymbol{x} = (x_1, x_2, ..., x_i, ..., x_j, ..., x_n)$, where n is the number of variables, if

$$\Delta_1 \times \Delta_2 < 0 \tag{1}$$

can be true for a set of values a_1, a_2, b_1, b_2 in

$$\begin{cases} \Delta_1 = f(\boldsymbol{x})|_{x_i=a_1,x_j=b_1} - f(\boldsymbol{x})|_{x_i=a_2,x_j=b_1} \\ \Delta_2 = f(\boldsymbol{x})|_{x_i=a_1,x_j=b_2} - f(\boldsymbol{x})|_{x_i=a_2,x_j=b_2} \end{cases} \tag{2}$$

variables x_i and x_j are considered to be interacting. Then, only convergence variables are grouped according to interactions among them.

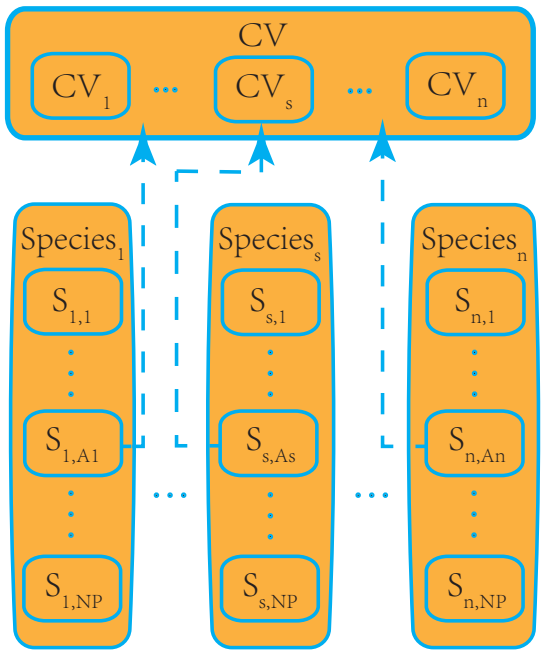

Fig. 1. CC framework.

B. CC

In large-scale optimization, there are a large number of variables. Separating variables into multiple groups and optimizing them under the CC framework [18], [19] results in good performance. As illustrated in Fig. 1, variables in different groups (species) are optimized separately and combined with each other to form a complete solution vector (CV) that can be evaluated.

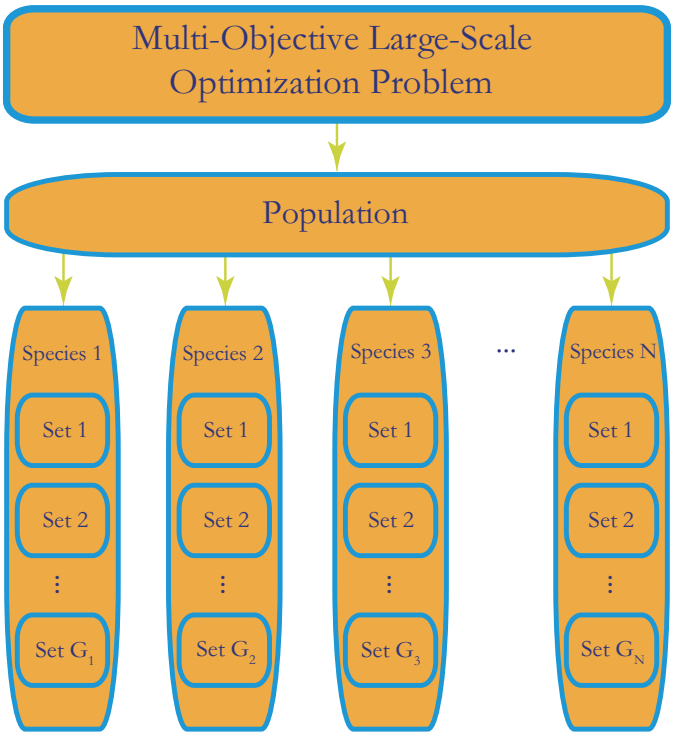

Fig. 2. Algorithm structure.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2017.2676000, IEEE Transactions on Industrial Informatics

Algorithm 1: DPCCMOEA

```
/* Initialization                                   */
Set generation number G = 0, set parameters, and
allocate memory;
/* Grouping variables                               */
Analyze and group decision variables as discussed in
Sections III-B;
/* Construct MPI parallel structure                 */
According to the grouping information, construct the
MPI parallel structure as discussed in Section IV-A;
/* Species Initialization                           */
Initialize individuals of the sets in each species;
while G < G_MAX do
    /* Information Exchange                         */
    Exchange information among CPUs as discussed in
    Section IV-B;
    /* Population Evolution                         */
    Evolve each sets as discussed in Section III-C;
    /* Population Updating                          */
    Update the sets as discussed in Section III-C, which
    is similar to MOEA/D;
    G = G + 1;
end
/* Population Reduction                             */
Perform population reduction according to reference
weights, and output the final population;
/* Finalization                                     */
Destroy variables and free memory;
```

III. DPCCMOEA

In this section, we describe the main idea of DPCCMOEA. First, the overall structure is given. Then, the variable analysis and grouping strategy is described. Finally, the optimization of the population is detailed.

A. Algorithm Structure

To solve MOLSOPs, we propose a decomposition based algorithm. To fully exploit the parallelism, we integrate a variety of decomposition strategies, which are variable (population) decomposition and species decomposition. Therefore, the population is decomposed into several species; and individuals in each species are further separated into many sets (subspecies). The decomposition structure can be illustrated in Fig 2.

The overall process is in Algorithm 1.

B. Variable Property Analysis and Grouping

In MOEA/DVA, the classification of variables is related to its effect on the convergence and diversity of solutions. For example, see the following problem, which is partly derived from the Rosenbrock function:

$$\begin{cases} f_1(\boldsymbol{x}) = & x_1 - \cos(2\pi x_2) + (x_3 - x_4^2)^2 + (x_4 - \frac{1}{2})^2 \\ f_2(\boldsymbol{x}) = & 1 - x_1 + \sin(2\pi x_2) + (x_3 - x_5^2)^2 + (x_5 - \frac{1}{2})^2 \end{cases} \tag{3}$$
$$s.t.\ x_i \in [0,1], i = 1,2,3,4,5$$

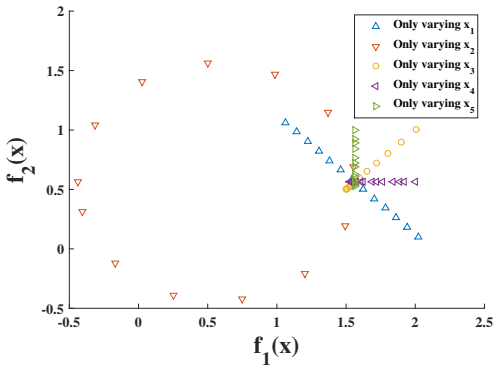

Fig. 3. Illustration of sampled points of the MOP formulated in Eq. 3 by varying one variable and fixing the others to 0.5.

TABLE I
RESULTS OF PROPERTY ANALYSIS

	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7		
# Div. Vars.	2	2	2	2	2	2	2		
MOEA/DVA	✓	✓	✓	✓	✓	✓	✓		
DPCCMOEA	✓	✓	✓	✓	✓	✓	✓		
	WFG1	**WFG2**	**WFG3**	**WFG4**	**WFG5**	**WFG6**	**WFG7**	**WFG8**	**WFG9**
# Div. Vars.	4	4	4	4	4	4	4	4	4
MOEA/DVA	✓	✓	✗	✓	✓	✓	✗	✓	✗
DPCCMOEA	✓	✓	✓	✓	✓	✓	✓	✓	✓

By varying the variables, we can obtain Fig. 3. From DVA, we know x_1 is a position variable; x_3, x_4 and x_5 are distance variables; and x_2 is a mixed variable.

The *interdependence analysis* in MOEA/DVA is not so precise. Detections are conducted more than one time and several sets of values a_1, a_2, b_1, b_2 are sampled. Thus, we use graph-based differential grouping (gDG) [26] and extend it to MOLSOPs. gDG is performed with respect to each objective, and the difference values ($|\Delta_1 - \Delta_2|$) are stored in a matrix, based on which we summarize the difference values and obtain a $nDim$-tuple $\boldsymbol{V_d}$ for all variables; here, $nDim$ is the number of variables.

We suppose that $\boldsymbol{S^{Pos}}$, $\boldsymbol{S^{Mix}}$ and $\boldsymbol{S^{Dis}}$ are the sets of position variables, mixed variables and distance variables [27], respectively. $|\boldsymbol{S}|$ denotes the cardinality of set $\boldsymbol{S}$. For the diversity category, there are three cases:

- **If** $\boldsymbol{0 < |S^{Pos}| + |S^{Mix}| < nDim}$, we classify all position variables and mixed variables into the diversity category;
- **Else if** $\boldsymbol{0 < |S^{Pos}| < nDim}$, we classify all position variables into the diversity category;
- **Else**, find the maximum value $V_{d,max}$ in $\boldsymbol{V_d}$, for each variable i, if $V_{d,i} > 0.2 \times V_{d,max}$, it is classified into the diversity category.

The remaining variables will be in the convergence category.

The results of property analysis of MOEA/DVA and DPCCMOEA for all instances (3 objectives, 1000 variables) in test suites DTLZ (Deb-Thiele-Laumanns-Zitzler) [30] and WFG (Walking-Fish-Group) [31] after 20 runs are listed in Table I. And we can see DPCCMOEA can correctly recognize all variables in the diversity category; while for MOEA/DVA, the results are not so good, as for WFG3, WFG7 and WFG9, more

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2017.2676000, IEEE Transactions on Industrial Informatics

variables (21.25, 999.9 and 985.35 on average, respectively, which are much more than 4) are incorrectly recognized and the recognition is a little unstable.

After the *interdependence analysis*, we group variables according to their properties and the interactions among them [26], [27]. First, all variables in the diversity category are allocated to a single group. As for the MOP formulated in Eq. 3, x_1 and x_2 will be in the same group. Variables in the convergence category will be separated to several groups according to interactions among them.

In addition, we collect all independent variables in a separate group, and if its size is too large, we split it with respect to a predefined threshold N_{th}^{Group}. For other groups, as long as the size of two smallest groups is below the threshold, we combine them so that the groups will not be too small. Thus, there will not be too many groups with few variables, which will result in the requirement of a large number of MPI processes (CPUs). Also, as long as the number of variables in the largest group is above the threshold, we split it. Therefore, each group will have a reasonable number of variables.

C. Optimization of the Population

The optimizer used is differential evolution (DE) [32], [33].

The optimization is actually conducted in the sets (Fig. 2). The optimization process can be described as follows.

$$child_{i,j} = \ p_{i,j} + F \times (p_{a_1,j} - p_{a_2,j}) \quad \textbf{if } j \in \boldsymbol{Index} \tag{4}$$

where $\boldsymbol{p_i}$ is the parent vector; i is the subscript of the newly generated vector $\boldsymbol{child_i}$, which is selected by binary tournament; j is the subscript of the decision variable; a_1 and a_2 denote the randomly selected solutions other than i; and $\boldsymbol{Index}$ is the set of variables to be optimized for the current set in the current generation.

Each set optimizes the diversity category and its allocated convergence group in a mixed way, and the switch between them is determined by the average improvement of individuals. To perform the fitness assessment, we should combine the remaining variables with the generated offspring to form a complete solution. As we also store the complete variable information about the individuals in each set, newly generated vectors will be integrated with this information to obtain a complete solution.

To enhance the exploration of the solution space and improve diversity, two vectors are used, which can be other stored complete solutions.

$$trail_{i,j} = \begin{cases} child_{i,j} & \textbf{if } j \in \boldsymbol{Index} \\ p_{i,j} & \textbf{if } j \notin \boldsymbol{Index} \bigwedge r_1 \leq 0.5 \\ p_{b_1,j} & \textbf{if } j \notin \boldsymbol{Index} \bigwedge r_1 > 0.5 \bigwedge r_2 \leq 0.5 \\ p_{b_2,j} & \textbf{otherwise} \end{cases} \tag{5}$$

where $\boldsymbol{trail_i}$ is the trail vector; r_1 and r_2 are uniform random numbers within the range of $(0, 1)$; and b_1 and b_2 denote the randomly selected solutions other than i. Then *polynomial mutation* is performed on $\boldsymbol{trail_i}$. After the fitness evaluation, the population updating is similar to that in MOEA/D.

IV. Parallel Implementation and Communication Topology

As described in the previous section, the decomposition structure is two-fold (Fig. 2). Based on this, we build the parallel structure.

MPI is a powerful tool for distributed parallelism. By sending and receiving messages, different CPUs communicate with each other and cooperatively complete the task. The heavy computational burden of tackling MOLSOPs can be allocated to large numbers of CPUs and the operation time will be greatly reduced.

The communication among CPUs is also time-consuming, too much information exchange will cancel out the speedup from computation allocation. Therefore, we should properly design the communication topology and choose the exchanged information.

A. Parallel Implementation

In our experimentation, the maximum Fitness Evaluations (FEs) are set to $1e4 \times nDim$, while the FEs used for *interdependence analysis* are $nDim \times (nDim + 1)$. The percent of FEs for *interdependence analysis* P_d^{FEs} is about

$$P_d^{FEs} \ = \ \frac{nDim \times (nDim + 1)}{1e4 \times nDim} \times 100\% \tag{6}$$

As $nDim = 1000$, $P_d^{FEs} \approx 10\%$. This seems not large. However, if we only parallelize the other part of the algorithm, the serial *interdependence analysis* process will be the major consumer of time. For example, if 100 CPUs are used for parallelization, the percent of computation time *interdependence analysis* takes will be

$$P_d^{TIME} \ = \ \frac{P_d^{FEs}}{P_d^{FEs} + (1 - P_d^{FEs})/100} \times 100\% \tag{7}$$

If $nDim = 1000$, $P_d^{TIME} \approx 97.56\%$, which greatly hinders the computation time reduction of the parallelism. Thus, we parallelize *interdependence analysis* with the help of MPI.

In the following, we will describe the parallel implementation in a top-down fashion according to Fig. 2.

Assuming there are N_{MPI} CPUs available for parallelism, for the population decomposition, CPUs are allocated to each species.

$$N_i^{MPI} = N^{MPI}/N^{Species}, \ i \in 1, 2, ..., N^{Species} \tag{8}$$

where N_i^{MPI} is the number of CPUs available for species i.

Then, in each species, the individuals are divided into multiple sets (subspecies).

$$NS_{i,j} = NP/N_i^{MPI}, \ j \in 1, 2, ..., N_i^{MPI} \tag{9}$$

where $NS_{i,j}$ is the size of set (subspecies) j of species i and NP is the population (species) size.

Through the above allocation of CPUs in a top-down manner, each CPU is in charge of a set. All CPUs can optimize the MOP in parallel, and the operation time can be greatly reduced.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2017.2676000, IEEE Transactions on Industrial Informatics

B. Communication Topology

CPUs exchange information according to the von Neumann topology, where the CPU nodes can be seen as a two-dimensional grid and each node is connected to four nodes (Fig. 4). To identify the neighbors, the *step* has to be computed, and according to the *step*, we decide to which process the current process should send information and from which process it should receive information.

$$step = \sqrt{N_{CPU}} \tag{10}$$

$$\begin{cases} R_{i,a} = (R_i - step + N_{CPU})\%N_{CPU} \\ R_{i,b} = (R_i + step)\%N_{CPU} \\ R_{i,l} = (R_i - 1 + N_{CPU})\%N_{CPU} \\ R_{i,r} = (R_i + 1)\%N_{CPU} \end{cases} \tag{11}$$

where $step$ is the step size, which is the length of each side of the two-dimensional square grid; $R_i, i \in \{1, 2, ..., N_{CPU}\}$ is the *rank* of MPI process i, which is the ID for CPU i; $R_{i,a}$, $R_{i,b}$, $R_{i,l}$ and $R_{i,r}$ are the *rank*s of the four neighbors: the above one, the below one, the left one and the right one.

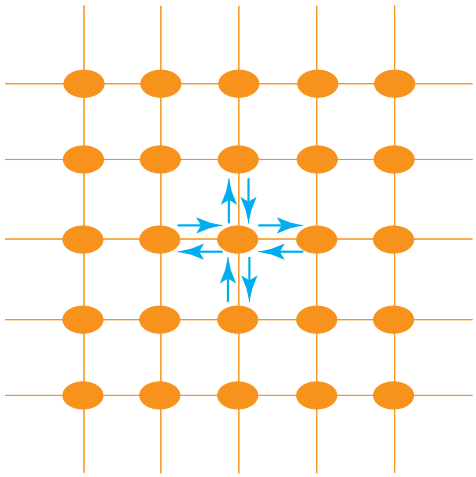

Fig. 4. Von Neumann topology.

Information exchange among CPUs can be summarized as follows:

- Within species
  As individuals refer to their neighbors for evolution and the division of species to sets will separate adjacent individuals, each set (CPU) transmit the individual information to its adjacent sets (CPUs).
- Beyond species
  All species exchange information with each other and the von Neumann topology is used. The whole individual information is transferred, according to which the receiver species updates its individuals.
- Global
  We form a communication topology among all CPUs. During the evolution, all CPUs will exchange information according to the von Neumann topology, which includes their several best individuals according to the fitness value and the most potential individual according to the utility.

V. Experimental Study

A. Parameter Setting

In the comparison of the algorithms, we contrast the proposed DPCCMOEA with two state-of-the-art cooperative coevolutionary MOEAs for tackling MOLSOPs: CCGDE3 [20] and MOEA/DVA [27].

For fair comparison, we set the population sizes of all three algorithms to $NP = 120$. CCGDE3 is based on fixed grouping, we simply set the number of groups to 2 and each group has $NP/2$ individuals. In MOEA/DVA, the number of control variable analyses and the number of interdependence analyses are set to $NCA = 20$ and $NIA = 6$, respectively. In DPCCMOEA, we set $NCA = 20$ and $NIA = 1$; and the group size threshold is $N_{th}^{Group} = 111$.

The number of variables in each test problem is $nDim = 1000$, the objective number is $nObj = 3$ and the maximum number of FEs is $N_{FE}^{max} = 1e4 \times nDim$.

In MOEA/DVA and DPCCMOEA, the neighborhood size and replace limit of each new solution are set to $0.1 \times NP$ and $0.01 \times NP$, respectively.

For DE, in CCGDE3 and DPCCMOEA, F and CR are set to 0.5 and 1.0, respectively. *SBX* and *polynomial mutation* are used in MOEA/DVA and *polynomial mutation* is used in DPCCMOEA. The distribution indices are both set to 20, and the *SBX* probability is set to 1.0. The *polynomial mutation* probability is set to $1.0/nDim$.

The proposed algorithm is based on the distributed MPI framework. The testing platform is the Tianhe-2 supercomputer. There are 24 cores in each node. In the experimentation, we make use of 15 nodes; that is, we use 360 cores. Thus, there are 360 CPUs in operation.

All three algorithms in the comparison run 20 times for each test instance.

B. Multi-objective Test Suite

The multi-objective test suites used in our experiment are: DTLZ [30] and WFG [31]. For more information about the test suites, please refer to the supplementary material.

C. Performance Metric

To compare the performances of these algorithms, we adopt the inverted generational distance (IGD) metric and the hypervolume (HV) indicator [34] (denoted as I_{IGD} and I_{HV}, respectively), which can comprehensively assess the convergence and diversity of the solutions. IGD has the following definition:

$$I_{IGD}(\boldsymbol{P}^*, \boldsymbol{A}) = \frac{\sum_{\boldsymbol{p} \in \boldsymbol{P}^*} d(\boldsymbol{p}, \boldsymbol{A})}{|\boldsymbol{P}^*|} \tag{12}$$

where $\boldsymbol{P}^*$ is a subset of points uniformly sampled from PF. $\boldsymbol{A}$, generated by the optimization algorithm, is the approximation of the PF. $|\boldsymbol{P}^*|$ is the cardinality of $\boldsymbol{P}^*$; $\boldsymbol{p}$ is an element of $\boldsymbol{P}^*$; $d(\boldsymbol{p}, \boldsymbol{A})$ is the minimal Euclidean distance between $\boldsymbol{p}$ and the elements in $\boldsymbol{A}$. The smaller the I_{IGD} value, the better the performance; while for the I_{HV} value, the opposite situation is true.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2017.2676000, IEEE Transactions on Industrial Informatics

D. Statistical Comparison

To systematically measure the results of the algorithms, we utilize statistical analysis. For the characteristics of the stochastic algorithms, we adopt the nonparametric test to evaluate the algorithms [35], [36].

E. Comparison with CCGDE3 and MOEA/DVA

The results of IGD and HV indicators are listed in Tables II and IV, respectively, and the corresponding average rankings of the nonparametric Friedman test are listed in Tables III and V.

TABLE II
AVERAGE IGD VALUES OF THE APPROXIMATED PF FOUND BY CCGDE3, MOEA/DVA AND DPCCMOEA

	CCGDE3		MOEA/DVA		DPCCMOEA	
DTLZ1	2.07E+03	(1.83E+03)	**2.20E-01**	(1.07E-02)	1.30E+03	(1.71E+02)
DTLZ2	5.19E+00	(6.14E-01)	4.87E-02	(2.07E-08)	**4.80E-02**	(3.10E-04)
DTLZ3	4.55E+03	(4.85E+03)	**8.15E-01**	(3.51E-02)	2.04E+03	(4.79E+02)
DTLZ4	4.25E+00	(9.00E-01)	1.26E-01	(5.89E-08)	**2.77E-02**	(7.38E-04)
DTLZ5	5.40E+00	(7.74E-01)	**3.30E-03**	(1.40E-08)	1.66E-02	(8.20E-05)
DTLZ6	4.60E+02	(1.68E+01)	3.43E+02	(2.06E+00)	**1.51E-02**	(1.25E-04)
DTLZ7	2.66E+00	(9.02E-01)	1.04E-01	(4.84E-06)	**7.05E-02**	(1.25E-03)
WFG1	1.59E+00	(1.40E-01)	2.64E+00	(6.97E-04)	**1.11E+00**	(8.58E-03)
WFG2	5.73E-01	(1.26E-01)	**2.05E-01**	(1.51E-04)	2.43E-01	(1.05E-02)
WFG3	5.11E-01	(6.49E-02)	**6.22E-02**	(1.20E-02)	1.88E-01	(6.09E-03)
WFG4	8.18E-01	(1.90E-01)	2.27E-01	(8.59E-07)	**1.57E-01**	(1.31E-03)
WFG5	3.33E-01	(3.49E-02)	2.76E-01	(1.24E-05)	**2.10E-01**	(1.07E-03)
WFG6	1.76E+00	(1.64E+00)	2.77E-01	(3.76E-04)	**1.82E-01**	(6.58E-04)
WFG7	8.13E-01	(7.37E-02)	1.76E-01	(9.09E-04)	**1.74E-01**	(1.03E-03)
WFG8	8.53E-01	(9.96E-02)	2.13E-01	(1.14E-03)	**2.09E-01**	(2.01E-03)
WFG9	3.49E-01	(1.55E-01)	2.07E-01	(3.61E-03)	**1.93E-01**	(2.80E-03)

Note1: The numbers in parentheses represent the standard deviations.
Note2: The numbers in bold are the best average IGD values for the corresponding test instances.

TABLE III
AVERAGE RANKING OF THE ALGORITHMS (FRIEDMAN) WITH RESPECT TO IGD VALUES

Algorithm	Ranking	Final Ranking
DPCCMOEA	1.3125	1
MOEA/DVA	1.7500	2
CCGDE3	2.9375	3

From the IGD values in Table II, we can see that CCGDE3 performs the worst in all test instances, MOEA/DVA is the second best, and DPCCMOEA is the best, which is also illustrated by the average ranking of the Friedman test in Table III.

In the DTLZ test suite, DTLZ1 has a very simple structure. However, the function $g(\boldsymbol{x}_M)$ has the following form:

$$g(\boldsymbol{x}_M) = 100\left[|\boldsymbol{x}_M| + \sum_{x_i \in \boldsymbol{x}_M} (x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5))\right] \tag{13}$$

similar to Rastrigin's Function:

$$f(x) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10) \tag{14}$$

which has a large number of local optima. Thus, CCGDE3 and DPCCMOEA can be easily trapped in the local optima, but MOEA/DVA performs much better. In DTLZ2, trigonometric functions are introduced, but $g(\boldsymbol{x}_M) = \sum_{x_i \in \boldsymbol{x}_M} (x_i - 0.5)^2$, which is quite simple, and all algorithms perform much better and DPCCMOEA is slightly better than MOEA/DVA. Based on DTLZ2, DTLZ3 replaces $g(\boldsymbol{x}_M)$ by that in DTLZ1, so CCGDE3 and DPCCMOEA perform as bad as DTLZ1 and MOEA/DVA is still much better. DTLZ4 maps the variables nonlinearly based on DTLZ2. Compared to DTLZ2, both CCGDE3 and DPCCMOEA have better performances, but MOEA/DVA performs much worse. For DTLZ5, there is a degeneration mapping compared to DTLZ2 and the PF is a curve. Based on DTLZ5, DTLZ6 replaces $g(\boldsymbol{x}_M) = \sum_{x_i \in \boldsymbol{x}_M} (x_i - 0.5)^2$ by $g(\boldsymbol{x}_M) = \sum_{x_i \in \boldsymbol{x}_M} x_i^{0.1}$, so a non-linear mapping is performed to the variables. From the optimization results of the three algorithms, DPCCMOEA is stable for DTLZ5 and DTLZ6. In DTLZ7, the PF is disconnected, for which, DPCCMOEA performs the best. In summary, we can know CCGDE3 and DPCCMOEA are sensitive to local optima (DTLZ1 and DTLZ3), while MOEA/DVA cannot be easily trapped in inferior local optima; for non-linear transformation (DTLZ4 and DTLZ6), MOEA/DVA does not perform equally well, CCGDE3 is not stable, while DPCCMOEA is the best; for the degeneration (DTLZ5) of PF, all algorithms perform well; and for disconnections (DTLZ7) in PF, CCGDE3 and MOEA/DVA perform worse than DPCCMOEA.

For the WFG test suite, there are complex transformations on both the variables and the shape of the PF, and the differences among algorithms are not obvious. Overall, DPCCMOEA performs best, MOEA/DVA is slightly worse and CCGDE3 is the worst.

TABLE IV
AVERAGE HV VALUES OF THE APPROXIMATED PF FOUND BY CCGDE3, MOEA/DVA AND DPCCMOEA

	CCGDE3		MOEA/DVA		DPCCMOEA	
DTLZ1	4.73E-01	(4.16E-01)	**1.00E+00**	(4.92E-10)	7.13E-01	(9.64E-02)
DTLZ2	9.24E-01	(1.62E-02)	9.99E-01	(2.25E-09)	**1.00E+00**	(2.80E-08)
DTLZ3	5.70E-01	(4.07E-01)	**1.00E+00**	(8.91E-09)	9.48E-01	(3.00E-02)
DTLZ4	8.59E-01	(9.65E-02)	9.92E-01	(1.31E-08)	**9.99E-01**	(3.07E-07)
DTLZ5	7.68E-01	(6.41E-02)	**9.89E-01**	(1.61E-08)	9.89E-01	(1.21E-06)
DTLZ6	7.68E-01	(1.54E-02)	8.98E-01	(1.20E-03)	**1.00E+00**	(4.37E-10)
DTLZ7	5.47E-01	(6.29E-02)	8.21E-01	(7.25E-06)	**8.29E-01**	(1.64E-05)
WFG1	7.30E-01	(2.12E-02)	6.34E-01	(2.43E-04)	**7.91E-01**	(1.02E-03)
WFG2	8.87E-01	(1.09E-02)	8.70E-01	(1.02E-04)	**9.71E-01**	(1.19E-03)
WFG3	8.37E-01	(1.36E-02)	8.78E-01	(1.28E-02)	**8.96E-01**	(1.02E-03)
WFG4	8.55E-01	(1.49E-02)	7.22E-01	(8.60E-07)	**9.69E-01**	(6.77E-04)
WFG5	8.73E-01	(1.41E-02)	8.59E-01	(7.24E-06)	**9.50E-01**	(8.57E-04)
WFG6	9.28E-01	(4.05E-02)	8.90E-01	(2.21E-04)	**9.71E-01**	(7.51E-05)
WFG7	8.06E-01	(2.22E-02)	9.67E-01	(2.68E-04)	**9.72E-01**	(8.24E-05)
WFG8	8.14E-01	(1.60E-02)	9.57E-01	(3.97E-04)	**9.61E-01**	(8.13E-04)
WFG9	8.77E-01	(3.54E-02)	9.37E-01	(2.71E-03)	**9.47E-01**	(3.72E-03)

Note1: The numbers in parentheses represent the standard deviations.
Note2: The numbers in bold are the best average IGD values for the corresponding test instances.

TABLE V
AVERAGE RANKING OF THE ALGORITHMS (FRIEDMAN) WITH RESPECT TO HV VALUES

Algorithm	Ranking	Final Ranking
DPCCMOEA	1.1875	1
MOEA/DVA	2.1250	2
CCGDE3	2.6875	3

The results of the HV indicator are similar to that of IGD except WFG2 and WFG3. The reason is that, from the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2017.2676000, IEEE Transactions on Industrial Informatics

visualizations of WFG2 in the supplementary material, the distribution of solutions obtained by MOEA/DVA is quite worse compared to DPCCMOEA; for WFG3, the PF is a line, and solutions of MOEA/DVA are distributed on the line while there are some solutions of DPCCMOEA scattered near the line.

TABLE VI
COMPUTATIONAL TIME TAKEN BY CCGDE3, MOEA/DVA AND DPCCMOEA

	CCGDE3		MOEA/DVA		DPCCMOEA
DTLZ1	7.93E+03	(3.54E+01)	2.13E+03	(9.51E+00)	**2.24E+02**
DTLZ2	6.89E+03	(3.04E+01)	7.41E+02	(3.27E+00)	**2.27E+02**
DTLZ3	7.51E+03	(3.50E+01)	2.15E+03	(1.00E+01)	**2.15E+02**
DTLZ4	6.62E+03	(2.97E+01)	8.13E+02	(3.65E+00)	**2.23E+02**
DTLZ5	6.78E+03	(1.92E+01)	7.64E+02	(2.16E+00)	**3.53E+02**
DTLZ6	1.43E+04	(5.99E+01)	4.99E+03	(2.09E+01)	**2.39E+02**
DTLZ7	6.89E+03	(3.13E+01)	7.05E+02	(3.20E+00)	**2.20E+02**
WFG1	2.54E+04	(9.24E+01)	1.67E+04	(6.08E+01)	**2.74E+02**
WFG2	1.22E+04	(5.13E+01)	7.36E+03	(3.10E+01)	**2.38E+02**
WFG3	1.31E+04	(5.40E+01)	7.32E+03	(3.01E+01)	**2.43E+02**
WFG4	1.72E+04	(6.48E+01)	1.13E+04	(4.25E+01)	**2.65E+02**
WFG5	1.52E+04	(5.84E+01)	8.36E+03	(3.21E+01)	**2.61E+02**
WFG6	7.39E+05	(3.08E+02)	7.34E+05	(3.06E+02)	**2.40E+03**
WFG7	1.37E+04	(5.39E+01)	1.01E+04	(3.96E+01)	**2.55E+02**
WFG8	2.83E+05	(2.69E+02)	2.80E+05	(2.66E+02)	**1.05E+03**
WFG9	1.02E+06	(3.15E+02)	1.01E+06	(3.13E+02)	**3.24E+03**
DTLZ	5.69E+04	(3.35E+01)	1.23E+04	(7.23E+00)	**1.70E+03**
WFG	2.14E+06	(2.60E+02)	2.09E+06	(2.54E+02)	**8.22E+03**
SUM	2.20E+06	(2.21E+02)	2.10E+06	(2.12E+02)	**9.92E+03**

Note: the numbers in parentheses represent the speedup ratio.

For the operation time and speedup in Table VI, because DPCCMOEA is conducted in an MPI distributed environment, it takes much less time than CCGDE3 and MOEA/DVA. The speedups of DPCCMOEA compared to CCGDE3 and MOEA/DVA are 221 and 212, respectively, which are about 61.4% and 58.9% of the ideal speedup (360). By comprehensively considering the time consumed in serial implementation of CCGDE3 and MOEA/DVA, we save a large amount of time.

VI. CONCLUSION AND PROSPECT

In this paper, we have proposed an MPI-based distributed parallel cooperative coevolutionary multi-objective evolutionary algorithm for large-scale optimization, denoted DPCCMOEA. Through a series of decompositions, DPCCMOEA decomposes complex MOLSOPs into simpler low-dimensional subproblems. Additionally, using a 2-layer MPI parallel structure, we can evolve subproblems in parallel, greatly reducing operation time. We compare DPCCMOEA with CCGDE3 and MOEA/DVA in two aspects: optimization performance and operation time. Experimental results showed that the proposed algorithm not only obtains better optimization results but also significantly reduces the time consumption. For future work, we will test the effectiveness of DPCCMOEA on wireless sensor network deployment problems, etc.

REFERENCES

[1] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 32 – 49, 2011. [Online]. Available: //www.sciencedirect.com/science/article/pii/S2210650211000058

[2] Y. Zuo, M. Gong, J. Zeng, L. Ma, and L. Jiao, "Personalized recommendation based on evolutionary multi-objective optimization [research frontier]," *IEEE Computational Intelligence Magazine*, vol. 10, no. 1, pp. 52–62, Feb 2015.

[3] S. Ruzika and M. M. Wiecek, "Approximation methods in multiobjective programming," *Journal of Optimization Theory and Applications*, vol. 126, no. 3, pp. 473–501, 2005. [Online]. Available: http://dx.doi.org/10.1007/s10957-005-5494-4

[4] A. Gaspar-Cunha and J. A. Covas, "Robustness in multiobjective optimization using evolutionary algorithms," *Computational Optimization and Applications*, vol. 39, no. 1, pp. 75–96, 2008. [Online]. Available: http://dx.doi.org/10.1007/s10589-007-9053-9

[5] E. Besada-Portas, L. de la Torre, J. M. de la Cruz, and B. de Andrs-Toro, "Evolutionary trajectory planner for multiple UAVs in realistic scenarios," *IEEE Transactions on Robotics*, vol. 26, no. 4, pp. 619–634, Aug 2010.

[6] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay, and C. A. C. Coello, "A survey of multiobjective evolutionary algorithms for data mining: Part I," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 1, pp. 4–19, Feb 2014.

[7] Y. Jin and B. Sendhoff, "Pareto-based multiobjective machine learning: An overview and case studies," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 3, pp. 397–415, May 2008.

[8] K. Praditwong, M. Harman, and X. Yao, "Software module clustering as a multi-objective search problem," *IEEE Transactions on Software Engineering*, vol. 37, no. 2, pp. 264–282, March 2011.

[9] V. L. Vachhani, V. K. Dabhi, and H. B. Prajapati, "Survey of multi objective evolutionary algorithms," in *2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015]*, March 2015, pp. 1–9.

[10] A. Trivedi, D. Srinivasan, K. Sanyal, and A. Ghosh, "A survey of multiobjective evolutionary algorithms based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. PP, no. 99, pp. 1–1, 2016.

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

[12] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, Dec 2007.

[13] M. Gong, L. Jiao, H. Du, and L. Bo, "Multiobjective immune algorithm with nondominated neighbor-based selection," *Evolutionary Computation*, vol. 16, no. 2, pp. 225–255, Jun 2008.

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

[15] C. Goh, K. Tan, D. Liu, and S. Chiam, "A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design," *European Journal of Operational Research*, vol. 202, no. 1, pp. 42 – 54, 2010. [Online]. Available: //www.sciencedirect.com/science/article/pii/S0377221709003166

[16] Y.-J. Gong, W.-N. Chen, Z.-H. Zhan, J. Zhang, Y. Li, Q. Zhang, and J.-J. Li, "Distributed evolutionary algorithms and their models: A survey of the state-of-the-art," *Applied Soft Computing*, vol. 34, pp. 286 – 300, 2015. [Online]. Available: //www.sciencedirect.com/science/article/pii/S1568494615002987

[17] X. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 2, pp. 210–224, April 2012.

[18] M. A. Potter and K. A. D. Jong, "A cooperative coevolutionary approach to function optimization," in *Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature: Parallel Problem Solving from Nature*, ser. PPSN III. London, UK, UK: Springer-Verlag, 1994, pp. 249–257. [Online]. Available: http://dl.acm.org/citation.cfm?id=645822.670374

[19] ——, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evolutionary Computation*, vol. 8, no. 1, pp. 1–29, March 2000.

[20] L. M. Antonio and C. A. C. Coello, "Use of cooperative coevolution for solving large scale multiobjective optimization problems," in *2013 IEEE Congress on Evolutionary Computation*, June 2013, pp. 2758–2765.

[21] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Information Sciences*, vol. 178, no. 15, pp. 2985 – 2999, 2008, nature Inspired Problem-

Solving. [Online]. Available: //www.sciencedirect.com/science/article/pii/S002002550800073X

[22] M. N. Omidvar, X. Li, Z. Yang, and X. Yao, "Cooperative co-evolution for large scale optimization through more frequent random grouping," in *IEEE Congress on Evolutionary Computation*, July 2010, pp. 1–8.

[23] M. N. Omidvar, X. Li, and X. Yao, "Cooperative co-evolution with Delta grouping for large scale non-separable function optimization," in *IEEE Congress on Evolutionary Computation*, July 2010, pp. 1–8.

[24] M. N. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 378–393, June 2014.

[25] Y. Mei, M. N. Omidvar, X. Li, and X. Yao, "A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization," *ACM Trans. Math. Softw.*, vol. 42, no. 2, pp. 13:1–13:24, Jun. 2016. [Online]. Available: http://doi.acm.org/10.1145/2791291

[26] Y. Ling, H. Li, and B. Cao, "Cooperative co-evolution with graph-based differential grouping for large scale global optimization," in *Proc. 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNCFSKD)*, Aug. 13-Aug. 15 2016, pp. 1097–1104.

[27] X. Ma, F. Liu, Y. Qi, X. Wang, L. Li, L. Jiao, M. Yin, and M. Gong, "A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 2, pp. 275–298, April 2016.

[28] W. Gropp, E. Lusk, and A. Skjellum, *Using MPI: Portable Parallel Programming with the Message-passing Interface*. Cambridge, MA, USA: MIT Press, 1994.

[29] W. Gropp, E. Lusk, and R. Thakur, *Using MPI-2: Advanced Features of the Message-Passing Interface*. Cambridge, MA, USA: MIT Press, 1999.

[30] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multiobjective optimization," *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*, pp. 105–145, 2005. [Online]. Available: http://dx.doi.org/10.1007/1-84628-137-7_6

[31] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 5, pp. 477–506, Oct 2006.

[32] K. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

[33] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997. [Online]. Available: http://dx.doi.org/10.1023/A:1008202821328

[34] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, April 2003.

[35] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3 – 18, 2011. [Online]. Available: //www.sciencedirect.com/science/article/pii/S2210650211000034

[36] J. Alcalá-Fdez and et al., "KEEL: A software tool to assess evolutionary algorithms to data mining problems," *Soft Comput.*, vol. 13, no. 3, pp. 307–318, 2008.

Bin Cao received the Ph.D. degree in 2012 and he is currently with the School of Computer Science and Engineering of Hebei University of Technology, Tianjin, China. From 2012 to 2014, he was a Postdoc in the Department of Computer Science and Technology of Tsinghua University, Beijing, China. His research interests include intelligent computation with its applications to Big Data, Cyber-Physical System, Graphics and Visual Media; High Performance Computing and Cloud Computing.

Jianwei Zhao received the bachelor degree from Tianjin University of Technology in 2014. He is currently a master degree candidate in the School of Computer Science and Engineering of Hebei University of Technology, Tianjin, China. His main research interests include intelligent computation with its applications to Big Data, Cyber-Physical System, Graphics and Visual Media; High Performance Computing and Cloud Computing.

Zhihan Lv is an engineer and researcher in virtual/augmented reality and multimedia, with a major in mathematics and computer science, having plenty of work experience in virtual reality and augmented reality projects, and is engaged in the application of computer visualization and computer vision. His research application fields range widely, from everyday life to traditional research fields (geography, biology, medicine). During the past few years, he has successfully completed several projects on PCs, Websites, Smartphones, and Smart glasses.

Xin Liu received the master degree from the Jilin University in 2012. She is currently with Hebei University of Technology, Tianjin, China. Her main research interests include intelligent computation techniques with applications to Big Data, Cyber-Physical System, Graphics and Visual Media; High Performance Computing and Cloud Computing.