Distributed Parallel Cooperative Coevolutionary Multi-Objective Large-Scale Immune Algorithm for Deployment of Wireless Sensor Networks

Bin Cao[a,b,c,*], Jianwei Zhao[a,b,c], Po Yang[d,*], Zhihan Lv[e], Xin Liu[f], Xinyuan Kang[a], Shan Yang[a], Kai Kang[f]

[a] *School of Computer Science and Engineering, Hebei University of Technology, 300401 Tianjin, China*
[b] *Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education, 510006 Guangzhou, China*
[c] *Hebei Provincial Key Laboratory of Big Data Calculation, 300401 Tianjin, China*
[d] *Department of Computer Science, Liverpool John Moores University, Liverpool, UK*
[e] *School of Data Science and Software Engineering, Qingdao University, 266071 Qingdao, China.*
[f] *Hebei University of Technology, 300401 Tianjin, China.*

Abstract

Using immune algorithms is generally a time-intensive process—especially for problems with a large number of variables. In this paper, we propose a distributed parallel cooperative coevolutionary multi-objective large-scale immune algorithm that is implemented using the message passing interface (MPI). The proposed algorithm is composed of three layers: objective, group and individual layers. First, for each objective in the multi-objective problem to be addressed, a subpopulation is used for optimization, and an archive population is used to optimize all the objectives. Second, the large number of variables are divided into several groups. Finally, individual evaluations are allocated across many core processing units, and calculations are performed in parallel. Consequently, the computation time is greatly reduced. The proposed algorithm integrates the idea of immune algorithms, which tend to explore sparse areas in the objective space and use simulated binary crossover for mutation. The proposed algorithm is employed to optimize the 3D terrain

*Corresponding authors.
E-mail addresses: p.yang@ljmu.ac.uk (Po Yang), caobin@scse.hebut.edu.cn (Bin Cao)

deployment of a wireless sensor network, which is a self-organization network. In experiments, compared with several state-of-the-art multi-objective evolutionary algorithms—the Cooperative Coevolutionary Generalized Differential Evolution 3, the Cooperative Multi-objective Differential Evolution and the Nondominated Sorting Genetic Algorithm III, the proposed algorithm addresses the deployment optimization problem efficiently and effectively.

Keywords: decision variable analysis (DVA), cooperative coevolution (CC), large-scale optimization, message passing interface (MPI), 3D terrain deployment, wireless sensor networks (WSNs)

1. Introduction

Self-organization [1] refers to the automatic formation of an ordered structure from an initially disordered system based on some type of rule. In the deployment optimization procedure of the wireless sensor network (WSN) [2], through self-organization, the wireless sensor nodes were optimized to maximize the *Coverage*, optimize *Connectivity Uniformity* and minimize *Deployment Cost*. With the rapid development of sensor and wireless communication technologies, WSNs have been applied to various fields. The work of [3] showed the air temperature monitoring application of the wireless sensor networks. Shen et al. [4] described the wireless sensor nodes for the medical service. Zhang et al. [5] illustrated the k-barrier coverage problem of the wireless sensor networks. Zhou et al. [6] researched on the energy issue, in which, clustering and data compression were studied; while, Zhang et al. [7] utilized the mobile sinks to alleviate the communication burden.

Also, the response of the human immune system to antigens can be viewed as a process of self-organization. Based on this concept, the clonal selection algorithm (CLONALG) [8] was proposed, which can be used for global optimization problems (GOPs) and multi-objective optimization problems (MOPs) [9]. Xue et al. [10] described the self adaptive artificial bee colony algorithm which is different from the immune algorithm and can also be a self-organizing procedure.

In the real world, many problems require several objectives (usually conflicting) to be considered simultaneously. Multi-objective evolutionary algorithms (MOEAs) [11, 12, 13] are capable of generating a plurality of solutions in a single run, which is convenient for approximating the Pareto front (PF). For NP-hard problems, evolutionary algorithms (EAs) [14, 15, 16, 17] can usually converge to a near optimal solution using limited computational resources [18] within a reasonable time compared to brute force and deterministic methods.

The first multi-objective immune algorithm (MOIA) was proposed in [19]. In this study, the immune algorithm (IA) was combined with the genetic algorithm (GA), to improve the selection of individuals for evolution. Gong et al. [20] proposed the nondominated neighbor immune algorithm (NNIA), which was prone to select a small quantity of nondominated individuals in the sparse area for cloning, recombination and mutation. In [21], simulated binary crossover (SBX) and differential evolution (DE) were combined and applied to the cloned individuals in a hybrid evolutionary framework for MOIAs called HEIA, which performed well for both unimodal and multimodal problems.

EAs are based on an iterative evolution of the population (the solutions), which is time-consuming—especially for expensive problems. Distributed evolutionary algorithms (dEAs) [22, 23] allocate the tedious computational burden across numerous computational nodes, greatly reducing the required time. Cloudde [24] used DEs with various parameters to optimize multiple populations in a distributed parallel manner, yielding a promising performance from both effect and efficiency aspects. [25] provided a comprehensive study concerning parallel/distributed MOEAs. Using the multi-objective optimization algorithm based on decomposition (MOEA/D) [13], parallel MOEA/Ds (pMOEA/Ds) [26] [27] were proposed.

Along with the arrival of "big data", many problems become complex and it will be time-consuming and storage-consuming to solve them [28, 29]. Similarly, many MOPs have a huge number of variables (more than 100 variables [30]); some examples are classification [31], clustering [32], recommendation systems [33], and so on. However, the goal of traditional MOEAs is to solve multi-objective small-scale optimization problems (MOSSOPs); consequently, the traditional algorithms may be incapable of tackling multi-objective large-scale optimization problems (MOLSOPs) because of the "curse of dimensionality". To optimize numerous variables, some promising approaches first separate the variables into groups and then optimize them in a cooperative coevolutionary (CC) [34] manner. For large-scale global optimization problems (LSGOPs), many grouping mechanisms have been applied, including fixed grouping [34], random grouping [35], the Delta method [36], dynamic grouping [37], differential grouping (DG) [38], global differential grouping (GDG) [39] and graph-based differential grouping (gDG) [40].

Antonio et al. proposed the cooperative coevolutionary generalized differential evolution 3 (CCGDE3) method [41], which used fixed grouping.

MOLSOPs differ from LSGOPs in that no single solution can optimize all the conflicting objectives, instead, a set of solutions should be generated to approximate the PF. In MOLSOPs, variables have different properties [42] that can be classified as follows:

1. position variables, which affect only the diversity of the solution set;
2. distance variables, which affect only the convergence of the solution set; and
3. mixed variables, which affect both the diversity and the convergence of the solution set.

Therefore, position variables should be permuted to approximate the PF as comprehensively as possible. However, distance variables should be optimized so they can closely approach the PF.

To identify these variable types, the multi-objective evolutionary algorithm based on decision variable analyses (MOEA/DVA) [30] proposed a mechanism that used decision variable analyses (DVA) to categorize the position and mixed variables as diversity-related variables and to categorize distance variables as convergence-related variables. The convergence-related variables were separated into several groups that were then optimized under the CC framework.

Using multiple populations can contribute to the optimization performance. In cooperative multi-objective differential evolution (CMODE) [43], each objective was optimized by a subpopulation, and an archive was used to maintain good solutions and optimize all objectives. This approach achieved good experimental results.

Compared to MOSSOPs, designing parallel/distributed MOEAs for MOLSOPs will be more beneficial. In this paper, we propose the distributed parallel cooperative coevolutionary multi-objective large-scale immune algorithm (DPCCMOLSIA), which is aimed at solving MOLSOPs in an effective and efficient manner.

The contributions of this paper can be summarized as follows:

1. Each objective is optimized by a subpopulation. Thus, the exploration with respect to each objective is enhanced, and all objectives are comprehensively optimized by an archive. Variables are grouped according to their properties and interactions, contributing to effective optimization.

2. The idea of IA is introduced, and more computational resources are used to explore sparse areas in the objective space. When combined with *SBX*, the performance can be enhanced.
3. We construct a three-layer parallel structure. The evaluations of individuals in different groups of multiple populations can then be performed in parallel, which greatly reduces the computation time.

The remainder of this paper is organized as follows: Section 2 provides some preliminary information required for this paper. The details of the DPCCMOLSIA are discussed in Section 3. Then, in Section 4, we describe the experimental study and present the corresponding analyses. Finally, Section 5 concludes this paper.

2. Preliminaries

2.1. MOP and Variable Properties

An MOP involves several objectives that usually conflict with each other; therefore, solving an MOP involves obtaining a set of solutions that approximate the PF. For the minimization problem, we have the following formula:

$$\text{Minimize } F\left(\boldsymbol{X}\right) = \left\{f_1\left(\boldsymbol{X}\right), f_2\left(\boldsymbol{X}\right), ..., f_M\left(\boldsymbol{X}\right)\right\} \tag{1}$$

where $\boldsymbol{X} = (X_1, X_2, ..., X_D)$ is a point in the solution space $\Re^D$. Here, D is the number of variables, f_i, $i = 1, 2, ..., M$, represents the objectives, and $F\left(\boldsymbol{X}\right)$ denotes the point that corresponds to $\boldsymbol{X}$ in the objective space, $\Re^M$.

Due to the conflicts among the objectives, the types of the different variables involved can vary: these types can be classified as position, distance, and mixed variables. For instance, consider the following MOP:

$$\begin{gathered} \left\{ \begin{array}{l} f_1 = x_1 + \sin\left(4\pi x_2\right) + e^{x_3(x_4 - 0.05)} + x_5^2 \\ f_2 = 1 - x_1 - \cos\left(4\pi x_2\right) + x_3^2 + x_4^3 + x_5^2 \end{array} \right. \\ s.t.\ x_i \in [0,1],\ i = 1,2,3,4,5. \end{gathered} \tag{2}$$

where f_1 and f_2 are two objectives, and x_1, x_2, x_3, x_4 and x_5 are decision variables.

Fig. 1 illustrates the sampled solution sets by varying each variable individually while holding the others constant at 0.5. From the image, we can determine the properties of the variables: x_1 is a position variable, because it influences only the diversity; x_2 is a mixed variable because it influences both the diversity and the convergence; x_3 and x_4 are distance variables, yet their

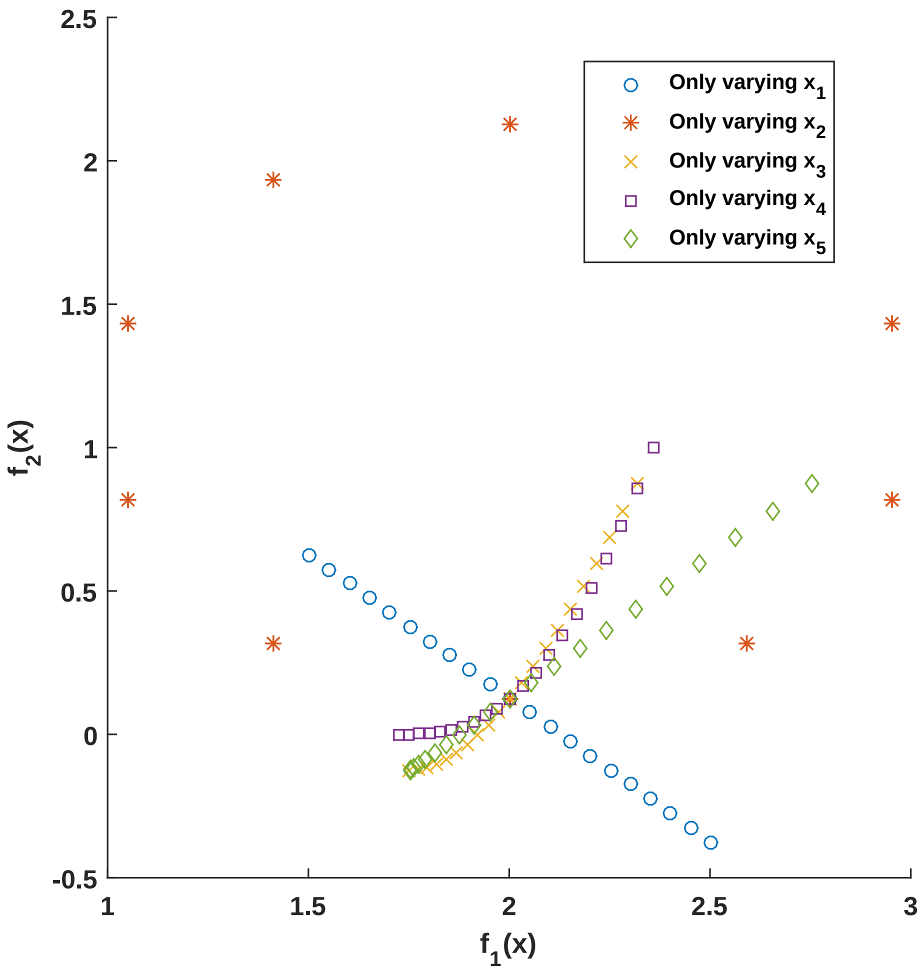


Figure 1: Image of solution sets for the MOP formulated in Eq. 2 by varying one variable while fixing the others to 0.5.

relative positions change a little with varying values; and x_5 is a distance variable, because it influences only the convergence.

2.2. CC

CC [34] divides a large number of variables into multiple subcomponents that are optimized separately. For the fitness evaluation, the target subcomponent is recombined with the representatives of the other components to form a complete solution.

2.3. Immune Algorithm

CLONALG was proposed in [8]; its process is detailed in Algorithm 1. In CLONALG, an *antibody* denotes a candidate solution, the optimal solution is seen as the *antigen*, and the *affinity* represents the fitness.

3. The Proposed Algorithm: DPCCMOLSIA

Algorithm 2 lists the main steps in the framework of DPCCMOLSIA. These main steps are described in detail in the following subsections.

3.1. Variable Property and Interaction Analyses

Variables are classified as position variables, distance variables and mixed variables according to their influences on diversity and convergence. At the end of this process, the position variables and mixed variables are categorized as diversity-related variables and the distance variables are categorized as convergence-related variables. For the MOP formulated in Eq. 2, x_1 and x_2 are classified as diversity-related variables, while x_3, x_4 and x_5 are classified as convergence-related variables.

3.2. Variable Grouping

Because more than one objective exists, the interactions among variables are obtained with respect to all the objectives by adopting the idea of gDG [40]. The diversity-related variables are separated into a single group. We group the convergence-related variables according to the following idea: if two variables interact with each other for any objective to be optimized in the current subpopulation/archive, we consider them to be interacting. For example, for the MOP formulated in Eq. 2, x_1 and x_2 are diversity-related variables, so they are allocated to a single group. For the convergence-related variables, x_3 and x_4 interact in f_1 and act independently in f_2, so we allocate them to a single group in subpopulation 1 (only optimizing f_1), to separate groups in subpopulation 2 (only optimizing f_2), and to the same group in the archive (optimizing both f_1 and f_2); x_5 is independent from other variables for both f_1 and f_2, so it is in another separate group.

Algorithm 1: CLONALG

```
Input: number of variables: $D$;
       number of antibodies: $N_{Ab}$;
       number of generations: $N_{gen}$;
       antibodies: $POP_{Ab}$;
       number of antibodies to be selected: $N_{sel}$.
Output: final antibodies: $POP_{Ab}$;
        final affinities: $AFF_{Ab}$.
/* Initialization                                              */
1  $G = 0$;
2  Randomly initialize $POP_{Ab}$;
3  Selected antibodies $POP_{sel} = \phi$, $AFF_{sel} = \phi$;
4  Reproduced antibodies $POP_{rep} = \phi$, $AFF_{rep} = \phi$;
/* Main Loop                                                   */
5  while $G < N_{gen}$ do
6      $AFF_{Ab}^{G} = f\left(POP_{Ab}^{G}\right)$;
7      Selection according to $AFF_{Ab}^{G}$:
       $POP_{Ab}^{G} \rightarrow POP_{sel}^{G},\ AFF_{Ab}^{G} \rightarrow AFF_{sel}^{G}$;
8      Cloning according to $AFF_{sel}^{G}$:
9      $POP_{sel}^{G} \rightarrow POP_{rep}^{G}$;
10     Hypermutation:
11     $POP_{rep}^{G} \rightarrow POP_{rep}^{G+1}, AFF_{rep}^{G+1} = f\left(POP_{rep}^{G+1}\right)$;
12     Insertion:
13     $POP_{Ab}^{G} + POP_{rep}^{G+1} \rightarrow POP_{Ab}^{G+1}$;
14     $G++$;
```

Algorithm 2: DPCCMOLSIA

```
1  Initialization;
2  Variable property and interaction analyses;
3  Variable Grouping;
4  Parallelism implementation;
5  Optimization;
```

3.3. Parallelism Implementation

For MOLSOPs, especially expensive ones, parallelism can be beneficial. DPCCMOLSIA is a distributed parallel algorithm implemented using the MPI. In DPCCMOLSIA, the parallel structure has three layers.

Assuming that there are N^{CPU} CPU resources available, the variables are divided to N_i^G groups. Here, $i = 1, 2, ..., M + 1$—that is to say, the subpopulations are represented by $i = 1, 2, ..., M$ and the archive is represented by $i = M + 1$. There are NP individuals in each subpopulation and in the archive population. And the importances of each subpopulation and the archive population are ω_{SUB} and ω_{ARC}, respectively. Then, we have the following equation:

$$\begin{gathered} N_i^{CPU} = \frac{N_i^G \times \omega_i}{\sum N_i^G \times \omega_i} \times N^{CPU} \\ s.t.\ i = 1, 2, ..., M + 1. \end{gathered} \tag{3}$$

where

$$\omega_i = \begin{cases} \omega_{SUB} & \text{if } i = 1, 2, ..., M \\ \omega_{ARC} & \text{if } i = M + 1 \end{cases} \tag{4}$$

and N_i^{CPU} is the number of CPUs allocated to the i-th subpopulation or the archive.

$$\begin{gathered} N_{i,j}^{CPU} = \frac{N_i^{CPU}}{N_i^G} \\ s.t.\ j = 1, 2, ..., N_i^G. \end{gathered} \tag{5}$$

where $N_{i,j}^{CPU}$ is the number of CPUs allocated to the j-th group in the i-th subpopulation or the archive.

The evaluations of the individuals are allocated across the multiple CPUs in each group.

$$\begin{gathered} N_{i,j,k}^{CPU} = \frac{NP}{N_{i,j}^{CPU}} \\ s.t.\ k = 1, 2, ..., N_{i,j}^{CPU}. \end{gathered} \tag{6}$$

where $N_{i,j,k}^{CPU}$ is the number of individuals that are assigned to the k-th CPU of the j-th group in the i-th subpopulation or the archive.

Therefore, based on the three-layer parallel structure, the evaluations of the individuals in each group of all $M + 1$ populations are conducted in parallel, which substantially reduces the computation time.

To guarantee the optimization performance, information must be shared among the groups. The communication strategy should be properly designed [44, 45], for this purpose, we adopt von Neumann topology.

3.4. Evolution Combined with the Idea of IA

The overall evolution process is provided by Algorithm 3. The evolution of each group in the subpopulations (Algorithm 4) or in the archive (Algorithm 5) is described in the following subsections.

3.4.1. Subpopulations

In Line 2 of Algorithm 4, in the evolution, tour selection is employed to choose 2 individuals from the full population. Then in Lines 3 and 4, we use *SBX* to evolve variables in the target group and integrate with other variables to form a complete individual.

$$\overline{X}_{i,j} = \begin{cases} SBX\left(X_i, X_{r_1}, X_{r_2}, j\right) & \text{if } j \in index \\ X_{r_3,j} & \text{otherwise} \end{cases} \tag{7}$$

where $\overline{X}_i$ is the generated new solution, X_i is the target parent individual, X_{r_1} and X_{r_2} are the 2 reference individuals, *index* is the set of variables optimized by the current group, and X_{r_3} is integrated with the optimized variables to form a complete solution, which has the following form:

$$r_3 = \begin{cases} i & \text{if } r < \dfrac{G}{N_{gen}} \\ r_4 & \text{else if } r' < 0.5 \\ r_5 & \text{otherwise} \end{cases} \tag{8}$$

Algorithm 3: Evolution

```
Input: generation number: $N_{gen}$.
Output: final population: $POP_{final}$.
1 for $G = 1 \rightarrow N_{gen}$ do
2     Evolve all variable groups in the subpopulations (Algorithm 4) and
      the archive (Algorithm 5) in parallel;
3     Exchange information among the groups;
4 Gather all the individuals from all groups to generate the final
  population $POP_{final}$;
```

Algorithm 4: Evolution of One Variable Group in Subpopulations

```
  Input: number of individuals: $NP$;
         population: $POP_1$.
  Output: new population: $POP_{new1}$.
  /* Evolution                                                  */
1 for $i = 1 \rightarrow NP$ do
2     Select 2 reference individuals;
3     Use $SBX$ to generate offspring $i$;
4     Integrate other variables with the generated offspring to form a
      complete solution;
5     Perform polynomial mutation;
  /* Evaluation                                                 */
6 Allocate the generated solutions to the CPU resources in the group
  and perform the evaluations in the CPUs in parallel;
7 Collect the fitness values from the CPUs;
  /* Refinement                                                 */
8 Combine the generated solutions with the old population;
9 Obtain $NP$ individuals based on their fitness values to the considered
  objective$\rightarrow POP_{new1}$;
```

where G is the number of the current generation and N_{gen} is the number of the maximum generation. Here, r and r' are uniform random numbers in the range of $[0.0, 1.0]$ and r_4 and r_5 are 2 selected individuals through tour selection. Then, in Line 5, *polynomial mutation* is performed.

In Lines 6 and 7, to evaluate the newly generated solutions, we use parallelism to alleviate the computational burden. This is the third layer of the parallel structure of DPCCMOLSIA.

Finally, in Lines 8 and 9, the NP best individuals with respect to the considered objective are preserved.

Algorithm 5: Evolution of One Variable Group in Archive

```
   Input: number of individuals: $NP$;
          population: $POP_2$;
          maximum number of individuals to be selected: $N_{sel}$.
   Output: new population: $POP_{new2}$.
   /* Selection                                              */
1  Select $N_{sel}$ individuals according to the Pareto dominance and
   crowding distance;
   /* Clone                                                  */
2  Clone the selected individuals to a total number of $NP$;
   /* Evolution                                              */
3  for $i = 1 \rightarrow NP$ do
4      Select 2 reference individuals;
5      Use $SBX$ to generate the offspring $i$;
6      Integrate other variables to the generated offspring to form a
       complete solution;
7      Perform polynomial mutation;
   /* Evaluation                                             */
8  Allocate the generated solutions to the CPU resources in the group
   and perform evaluations on the CPUs in parallel;
9  Collect the fitness values from the CPUs;
   /* Non-dominated sorting                                  */
10 Combine the generated solutions with the old population;
11 Obtain $NP$ individuals according to the Pareto dominance and
   crowding distance$\rightarrow POP_{new2}$;
```

3.4.2. Archive

Traditionally, in each generation, all individuals take part in evolution. However, this paper introduces the idea of IA, in which, in each generation, we select several best individuals and produce NP offspring, the whole process of which is illustrated in Algorithm 5. In detail, the selection of individuals in Line 1 is determined by two criteria: non-dominance and crowding distance. If the number of nondominated individuals is less than N_{sel}, we select them all for cloning; otherwise, we select the N_{sel} individuals that have larger crowding distances. In the cloning process in Line 2, the number of clones of each selected individual is determined by the crowding distance.

$$N_i^C = \frac{dist_i}{\sum_{i=1}^{N_{sel}} dist_i} \times NP, \tag{9}$$

where N_i^C represents the replications of selected individual i and $dist_i$ is its crowding distance in the population, which is calculated as follows:

$$dist_i = \sum_{m=1}^{M} dist_i^m, \tag{10}$$

where, $dist_i^m$ is the crowding distance of individual i with respect to objective m,

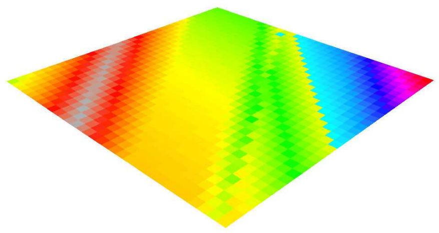
$$dist_i^m = \begin{cases} \infty & \text{if } (i)^* = 1 \\ \dfrac{\widetilde{f}_m^{(i)^*+1} - \widetilde{f}_m^{(i)^*-1}}{\widetilde{f}_m^{NP} - \widetilde{f}_m^{1}} & \text{otherwise} \end{cases} \tag{11}$$

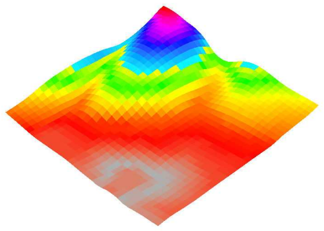
and $\widetilde{f}_m^{(i)^*}$ is the f_m^i sorted in ascending order. Finally, $(i)^*$ is the new index of individual i in the sorted sequence.

$$dist_i = \begin{cases} 2 \times dist_i^{max} & \text{if } dist_i = \infty \\ dist_i & \text{otherwise,} \end{cases} \tag{12}$$

and $dist_i^{max}$ is the maximum crowding distance. Because there are ∞ values assigned to crowding distances, to calculate N_i^C, we have to convert them.

In Line 4 in the evolution process, we select 2 individuals from among the N_{sel} selected individuals if $N_{sel} > 2$; otherwise, the selection scope is the whole population. Then in Lines 5 and 6, we use SBX to generate the target individual. For the integration, r_4 and r_5 (Eq. 8) are 2 randomly selected individuals from the N_{sel} best individuals used for cloning when $N_{sel} > 2$ or


(a) Plain Terrain


(b) Hilly Terrain

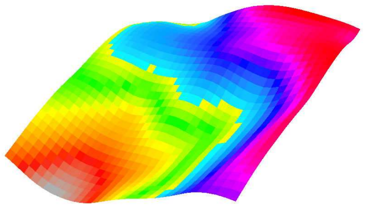

(c) Mountainous Terrain

Figure 2: Illustration of 3D terrain data.

from the whole population when $N_{sel} \leq 2$. Then Line 7 performs *polynomial mutation*.

Finally, in Lines 8 and 9, we combine the new individuals with the current population to obtain the NP best individuals according to the Pareto dominance and crowding distance. When the number of nondominated individuals is less than NP, several dominated individuals are preserved.

4. Experimental Research: Application to 3D Terrain Deployment of Heterogeneous Directional Sensor Networks

4.1. 3D Deployment Problem and Terrain Data

We use the 3D deployment problem proposed in [2], which includes three objectives: *Coverage*, *Connectivity Uniformity* and *Deployment Cost*. We also use the same real-world 3D terrain data (Fig. 2), which is composed of plain (Fig. 2(a)), hilly (Fig. 2(b)) and mountainous (Fig. 2(c)) terrains. These three terrains have different characteristics that are used to verify the proposed algorithm with respect to various conditions.

4.2. Parameter Setup

We compare DPCCMOLSIA with CCGDE3 [41], CMODE [43] and the nondominated sorting genetic algorithm III (NSGA-III) [46] in addressing the deployment optimization problem.

For all the algorithms, the optimization process is performed 20 times. The fitness evaluations (FEs) are set to $10^4 \times D$: here, $D = 10^2$.

To ensure a fair comparison, we set the population size, NP, to 120 for all algorithms. Specifically, for CCGDE3, the population is split into 2 subpopulations, each of which has 60 individuals. For CMODE, because there are 3 objectives that must be optimized, we used 3 subpopulations, each of which has 20 individuals, and set the maximum size of the archive to 120; for NSGA-III, we simply set NP to 120. For DPCCMOLSIA, each of the subpopulations and the archive population has 120 individuals, while the importance ratio of the subpopulation and the archive population is set to $\omega_{SUB} : \omega_{ARC} = 1 : 6$, and we finally select 120 individuals.

DE is used in CCGDE3, and F and CR are set to 0.5 and 1.0, respectively. *SBX* and *polynomial mutation* are used in NSGA-III and DPCCMOLSIA, and the distribution indexes are set to $\eta_c = \eta_m = 20$. The probabilities of crossover and mutation are set to $p_c = 1.0$ and $p_m = 1.0/D$, respectively.

Additionally, for DPCCMOLSIA, we set $N_{sel} = 0.1 \times NP$, and the number of CPUs used is 72.

4.3. Performance Indicator

Because the optimal solutions are unknown, we use the hypervolume (HV) indicator [47] and visualize all the obtained solutions. The higher is the HV indicator value, the better is the optimization performance.

4.4. Results and Analyses

First, we demonstrate all the obtained final nondominated solutions after 20 runs of each algorithm on each of the three terrains in Fig. 3. Here, $P - *$ denotes the results on plain terrain, $H - *$ denotes the results on hilly terrain, and $M - *$ denotes the results on mountainous terrain.

As Fig. 3 shows, the characteristics are quite different for the different terrains, while for the various algorithms on the same terrain, the solutions are only slightly different.

In general, for the plain terrain, all the algorithms perform better on the *Coverage* objective. For the hilly terrain, the algorithms tend to obtain good performance on the *Deployment Cost* objective. Finally, on the mountainous

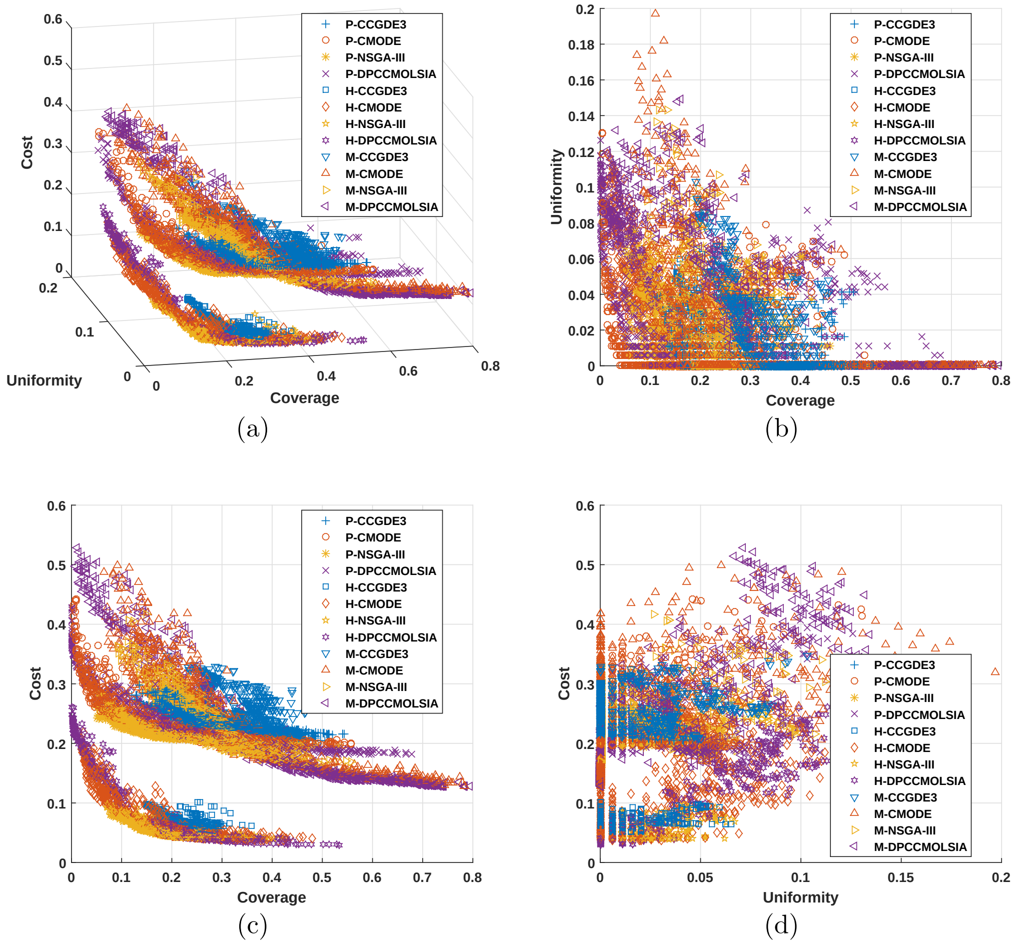


Figure 3: Visualization of solutions on all terrains.

terrain, the algorithms performances are all inferior to their performances on the other two terrains. We can comment on the above phenomena as follows:

1. Because the plain terrain is flatter than the other two terrains, it is easier to achieve better *Coverage*.
2. The hilly terrain has few changes in elevation, and algorithms tend to deploy the sensor nodes in the low-lying areas, thus guaranteeing better *Deployment Cost*.
3. The mountainous terrain has severe elevation changes, which makes it much more difficult to address compared with the other two terrains; consequently, the algorithms exhibit poor performances on this terrain.

In the following, we analyze the performances of the different algorithms on each terrain in detail.

4.4.1. Plain Terrain

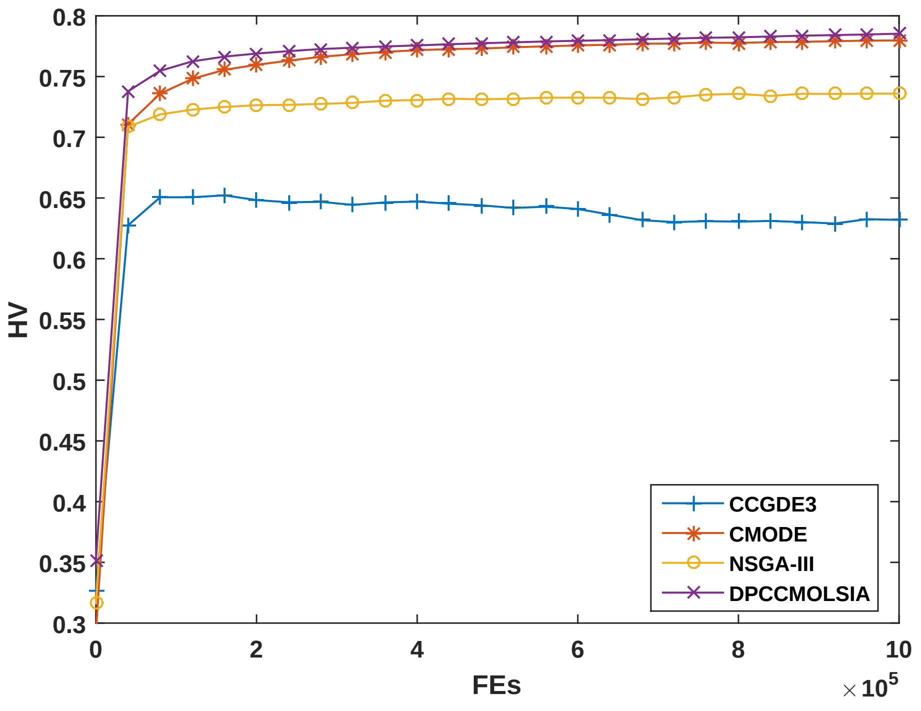
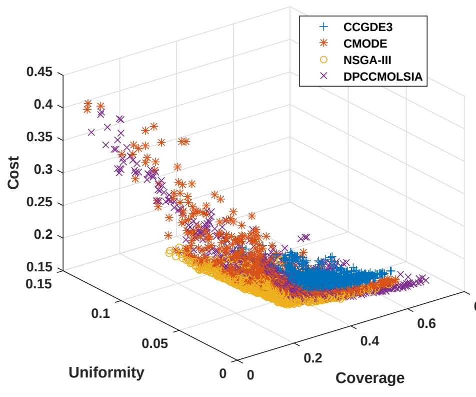
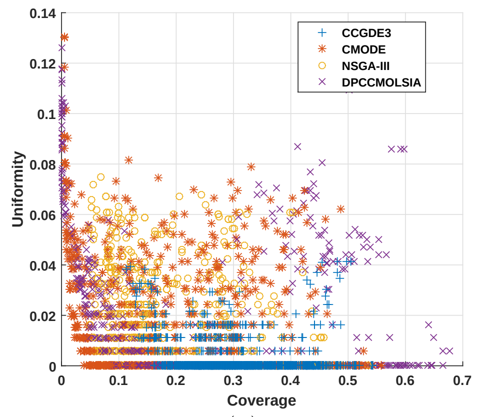


Figure 4: Convergence curves of HV on plain terrain.

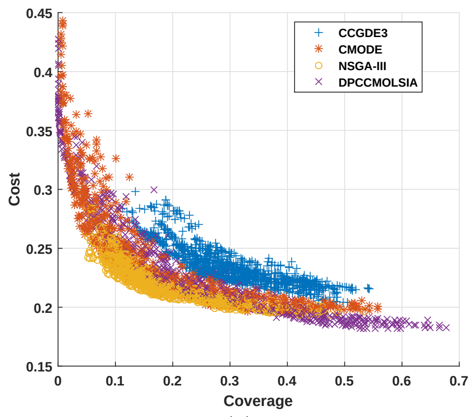
The convergence curves of the HV indicator are illustrated in Fig. 4. We can see that DPCCMOLSIA performs the best (0.785290), CMODE slightly worse (0.779786), NSGA-III is third (0.735985), and CCGDE3 performs the worst (0.631979). Moreover, DPCCMOLSIA has the fastest convergence speed, but improves less later in the process, similar to CMODE.



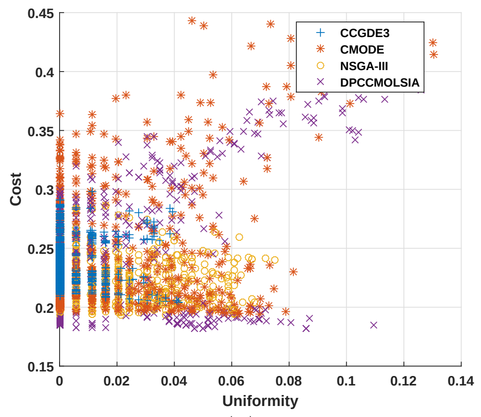
(a)



(b)



(c)



(d)

Figure 5: Visualization of solutions of plain terrain.

The visualization is shown in Fig. 5. In accordance with the HV indicator and considering the diversity and convergence of solutions, the overall performance of DPCCMOLSIA is the best.

Coverage is an important factor to consider in WSN deployment problems. From the visualization, we can see that DPCCMOLSIA is able to obtain a very low fitness value (good performance) for the *Coverage* objective, which validates its performance. Because the plain terrain is quite flat, it is easier to optimize the objectives *Connectivity Uniformity* and *Deployment Cost*.

On the whole, the performances of all the algorithms on the plain terrain can be ordered as follows: DPCCMOLSIA > CMODE > NSGA-III > CCGDE3.

4.4.2. Hilly Terrain

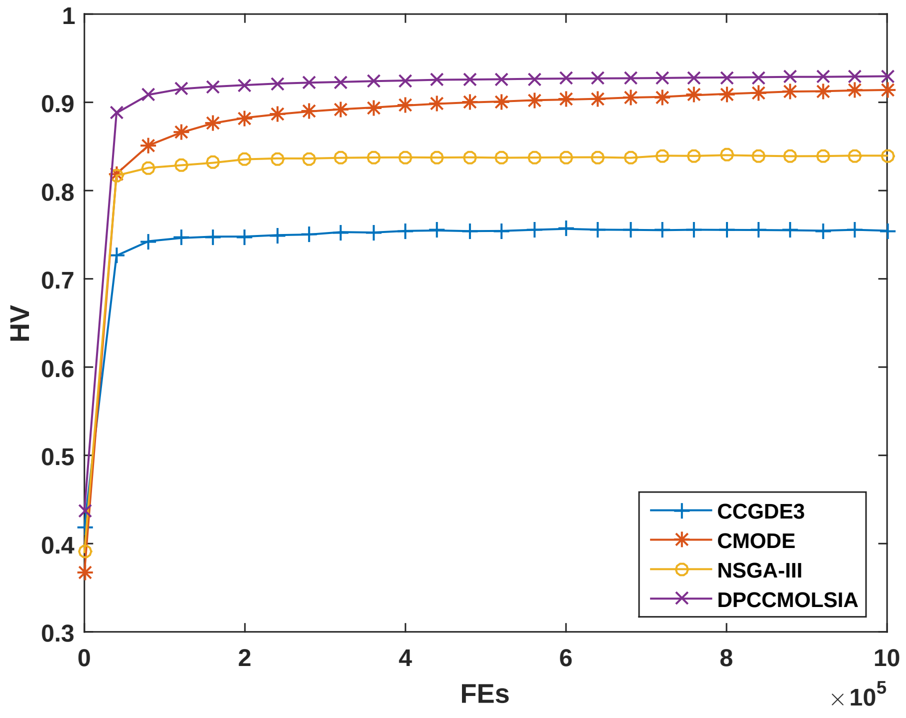


Figure 6: Convergence curves of HV on hilly terrain.

The convergence curves of the HV indicator for all the algorithms on the hilly terrain are illustrated in Fig. 6.

From the HV indicator, again, DPCCMOLSIA performs best (0.929553); CMODE is second (0.914022); NSGA-III is third (0.839551), and CCGDE3 is far worse (0.754544). The characteristics of all the algorithms are similar to those described above for the plain terrain.

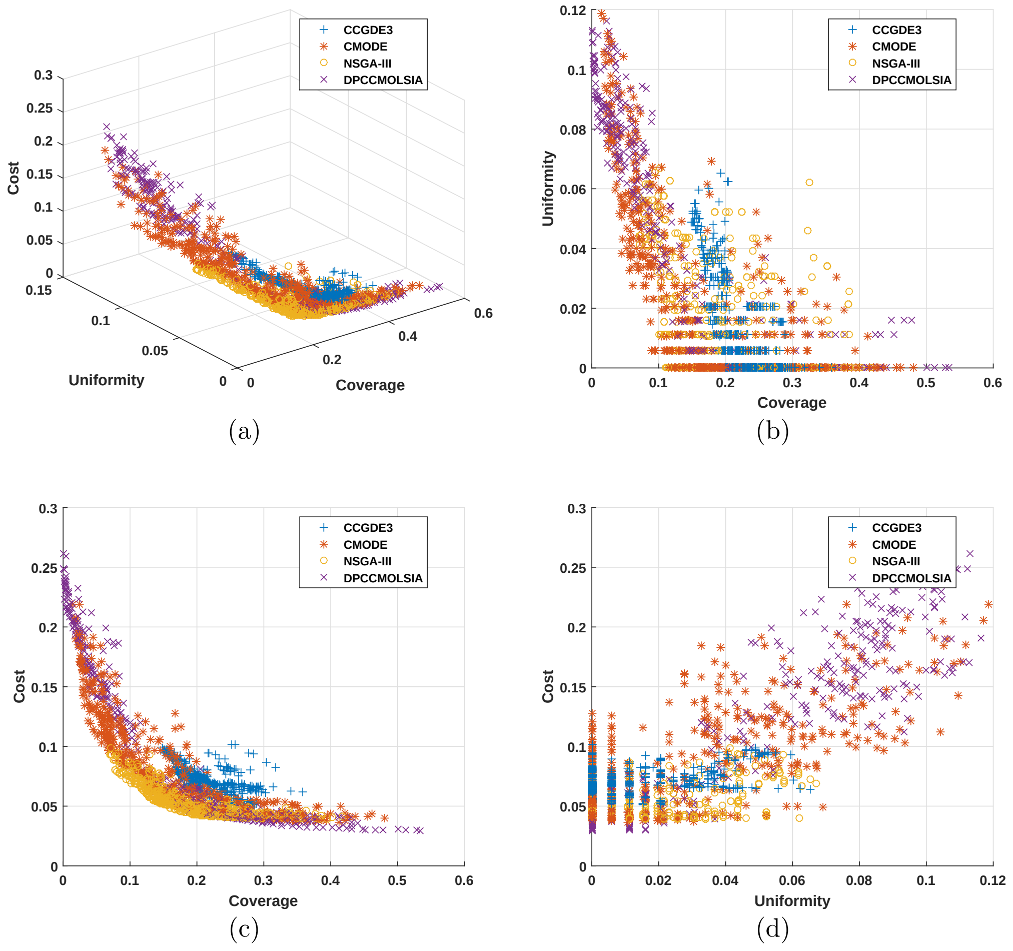


Figure 7: Visualization of solutions of hilly terrain.

The visualization of the solutions are shown in Fig. 7. Generally, DPCC-MOLSIA more comprehensively approximates the optimal PF and still guarantees good *Coverage*. As mentioned above, because the elevation changes in the hilly terrain are relatively small, the algorithms obtain a relatively good *Deployment Cost*. However, to achieve better *Coverage*, the sensor nodes should be deployed in higher areas, which results in a sharp increase in the fitness of the objective *Deployment Cost*, as can be observed in Fig. 7(c).

Overall, the performances of the algorithms on hilly terrain can be ordered as follows: DPCCMOLSIA > CMODE > NSGA-III > CCGDE3.

4.4.3. Mountainous Terrain

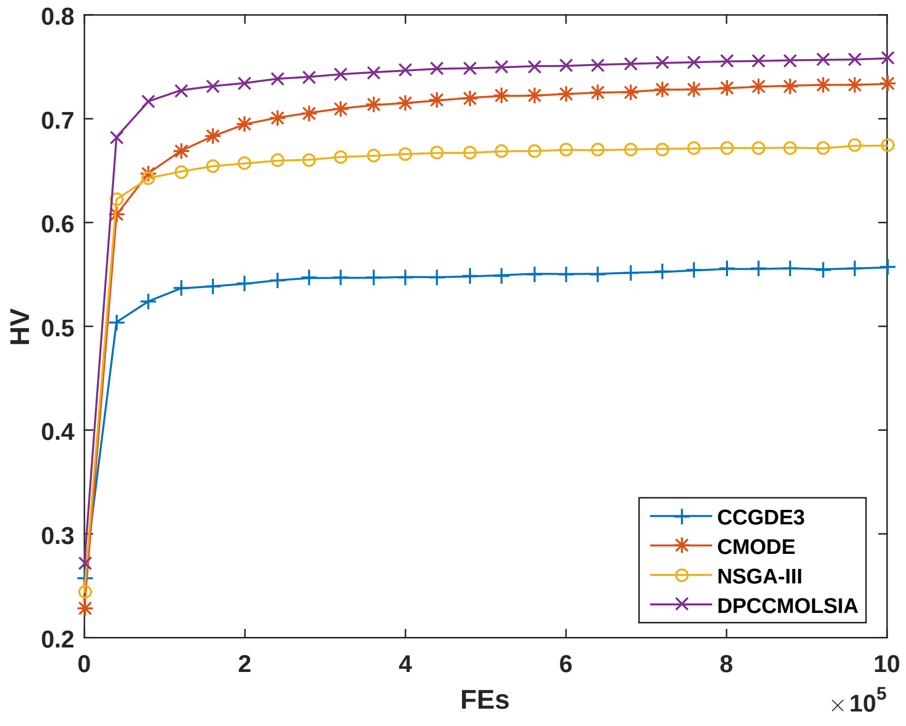


Figure 8: Convergence curves of HV on mountainous terrain.

The convergence curves of the HV indicator of DPCCMOLSIA, CMODE, NSGA-III and CCGDE3 on mountainous terrain are illustrated in Fig. 8.

DPCCMOLSIA again obtains the highest HV indicator value (0.758215), CMODE is a little worse (0.733522), NSGA-III is third (0.674049), and CCGDE3 is the worst (0.556730). The characteristics of the different algorithms are similar to those on the plain and hilly terrains.

Visualizations of the obtained solutions of all algorithms are shown in Fig. 9. Overall, the DPCCMOLSIA algorithm performs the best. Because mountainous terrain has severe altitude variations, it is much more difficult for the algorithms to achieve a good optimization performance.

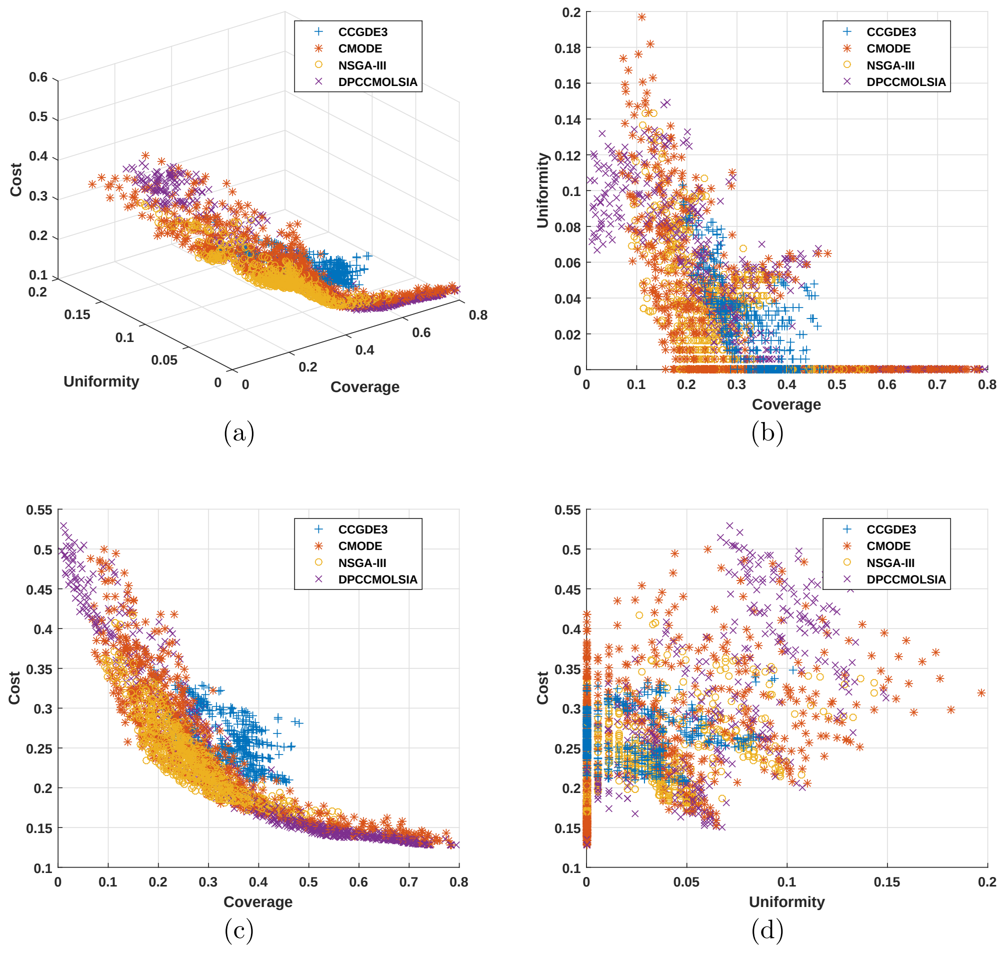


Figure 9: Visualization of solutions of mountainous terrain.

The performances of all four algorithms on mountainous terrain can be ordered as follows: DPCCMOLSIA > CMODE > NSGA-III > CCGDE3.

Overall, comprehensively considering all the tested terrains, DPCCMOLSIA achieves the best optimization results; CMODE is a little worse; NSGA-III is third; and CCGDE3 is well behind.

Table 1: Average Computation Time of CCGDE3, CMODE, NSGA-III and DPCCMOLSIA and the Speedup Ratios with Respect to DPCCMOLSIA

AVERAGE TIME	**CCGDE3**	**CMODE**	**NSGA-III**	**DPCCMOLSIA**
Plain terrain	2.32E+03	2.63E+03	2.52E+03	**8.48E+01**[1]
Hilly terrain	3.40E+03	3.58E+03	3.67E+03	**1.25E+02**
Mountainous terrain	3.06E+03	3.25E+03	3.28E+03	**1.13E+02**
All terrains	2.93E+03	3.15E+03	3.16E+03	**1.07E+02**
Speedup Ratio	2.73E+01	2.94E+01	2.94E+01	/

[1] Values in bold denote better performance.

Table 1 summarizes the computation time required by the various algorithms. Compared to the serial algorithms, the computation time of DPCCMOLSIA is substantially reduced.

5. Conclusion and Prospect

In this paper, we proposed a distributed parallel cooperative coevolutionary multi-objective large-scale immune algorithm (DPCCMOLSIA), which uses a three-layer parallel structure to substantially reduce the computation time. By decomposing the objectives and variables, the original complex MOLSOP is transformed into simpler, small-scale problems that are easier to address. We verified the effectiveness and efficiency of DPCCMOLSIA by testing it on a real-world problem in comparison with several other algorithms (CCGDE3, CMODE and NSGA-III). In the future, we will plan to continue the improvement of DPCCMOLSIA and test it on additional real-world problems.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant No. 61303001, in part by the Foundation of Key Laboratory of Machine Intelligence and Advanced Computing of the Ministry of Education under Grant No. MSC-201602A, in part by the Opening Project of Guangdong High Performance Computing Society under Grant No. 2017060101. This work was conducted at National Supercomputer Center in Guangzhou (NSCC-GZ) and National Supercomputer Center in Tianjin, and the calculations were performed on TianHe-2 and TianHe-1(A). The staff from Supercomputer Centers and the engineers from Beijing Paratera Technology Co., Ltd. provided effective support and made the computation smoothly. Thanks for all the support.

References

[1] S. A. Kauffman, Origins of Order in Evolution: Self-Organization and Selection, Springer Netherlands, Dordrecht, 1992, pp. 153–181. `doi:10.1007/978-94-015-8054-0_8`.
URL `http://dx.doi.org/10.1007/978-94-015-8054-0_8`

[2] B. Cao, J. Zhao, Z. Lv, X. Liu, 3D terrain multiobjective deployment optimization of heterogeneous directional sensor networks in security monitoring, Vol. PP, 2017, pp. 1–1. `doi:10.1109/TBDATA.2017.2685581`.

[3] B. Wang, X. Gu, L. Ma, S. Yan, Temperature error correction based on BP neural network in meteorological wireless sensor network, International Journal of Sensor Networks (IJSNET) 23 (4) (2017) 265 – 278. `doi:10.1504/IJSNET.2017.083532`.

[4] J. Shen, S. Chang, J. Shen, Q. Liu, X. Sun, A lightweight multi-layer authentication protocol for wireless body area networks, Future Generation Computer Systems`doi:http://dx.doi.org/10.1016/j.future.2016.11.033`.
URL `http://www.sciencedirect.com/science/article/pii/S0167739X16306963`

[5] Y. Zhang, X. Sun, B. Wang, Efficient algorithm for k-barrier coverage based on integer linear programming, China Communications 13 (7) (2016) 16–23. `doi:10.1109/CC.2016.7559071`.

[6] Z. Zhou, Q. J. Wu, F. Huang, X. Sun, Fast and accurate near-duplicate image elimination for visual sensor networks, International Journal of Distributed Sensor Networks 13 (2) (2017) 1550147717694172. arXiv:http://dx.doi.org/10.1177/1550147717694172, doi:10.1177/1550147717694172.
URL http://dx.doi.org/10.1177/1550147717694172

[7] J. Zhang, J. Tang, T. Wang, F. Chen, Energy-efficient data-gathering rendezvous algorithms with mobile sinks for wireless sensor networks, International Journal of Sensor Networks (IJSNET) 23 (4) (2017) 248 – 257. doi:10.1504/IJSNET.2017.10004216.

[8] L. N. de Castro, F. J. V. Zuben, Learning and optimization using the clonal selection principle, IEEE Transactions on Evolutionary Computation 6 (3) (2002) 239–251. doi:10.1109/TEVC.2002.1011539.

[9] C. A. C. Coello, N. C. Cortés, An approach to solve multiobjective optimization problems based on an artificial immune system, in: International Conference on Artificial Immune Systems, 2002, pp. 212–221.

[10] Y. Xue, J. Jiang, B. Zhao, T. Ma, A self-adaptive artificial bee colony algorithm based on global best for global optimization, Soft Computing (8) (2017) 1–18. doi:10.1007/s00500-017-2547-1.
URL https://doi.org/10.1007/s00500-017-2547-1

[11] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the strength Pareto evolutionary algorithm, Tech. rep., Eidgenössische Technische Hochschule Zürich (ETH), Institut für Technische Informatik und Kommunikationsnetze (TIK) (2001). doi:10.3929/ethz-a-004284029.

[12] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197. doi:10.1109/4235.996017.

[13] Q. Zhang, H. Li, MOEA/D: A multiobjective evolutionary algorithm based on decomposition, IEEE Transactions on Evolutionary Computation 11 (6) (2007) 712–731. doi:10.1109/TEVC.2007.892759.

[14] T. Zhu, W. Luo, C. Bu, L. Yue, Accelerate population-based stochastic search algorithms with memory for optima tracking on dynamic power

systems, IEEE Transactions on Power Systems 31 (1) (2016) 268–277. doi:10.1109/TPWRS.2015.2407899.

[15] C. Bu, W. Luo, L. Yue, Continuous dynamic constrained optimization with ensemble of locating and tracking feasible regions strategies, IEEE Transactions on Evolutionary Computation 21 (1) (2017) 14–33. doi:10.1109/TEVC.2016.2567644.

[16] C. Bu, W. Luo, T. Zhu, L. Yue, Solving online dynamic time-linkage problems under unreliable prediction, Applied Soft Computing 56 (2017) 702 – 716. doi:http://dx.doi.org/10.1016/j.asoc.2016.11.005. URL http://www.sciencedirect.com/science/article/pii/S1568494616305749

[17] Y. Zhang, W. Luo, Z. Zhang, B. Li, X. Wang, A hardware/software partitioning algorithm based on artificial immune principles, Applied Soft Computing 8 (1) (2008) 383 – 391. doi:http://dx.doi.org/10.1016/j.asoc.2007.03.003. URL http://www.sciencedirect.com/science/article/pii/S1568494607000257

[18] W. Luo, J. Sun, C. Bu, H. Liang, Species-based particle swarm optimizer enhanced by memory for dynamic optimization, Applied Soft Computing 47 (2016) 130 – 140. doi:http://dx.doi.org/10.1016/j.asoc.2016.05.032. URL http://www.sciencedirect.com/science/article/pii/S1568494616302423

[19] J. Yoo, P. Hajela, Immune network simulations in multicriterion design, Structural optimization 18 (2) (1999) 85–94. doi:10.1007/BF01195983. URL http://dx.doi.org/10.1007/BF01195983

[20] M. Gong, L. Jiao, H. Du, L. Bo, Multiobjective immune algorithm with nondominated neighbor-based selection, Evolutionary Computation 16 (2) (2008) 225–255. doi:10.1162/evco.2008.16.2.225.

[21] Q. Lin, J. Chen, Z. H. Zhan, W. N. Chen, C. A. C. Coello, Y. Yin, C. M. Lin, J. Zhang, A hybrid evolutionary immune algorithm for multiobjective optimization problems, IEEE Transactions on Evolutionary Computation 20 (5) (2016) 711–729. doi:10.1109/TEVC.2015.2512930.

[22] Y.-J. Gong, W.-N. Chen, Z.-H. Zhan, J. Zhang, Y. Li, Q. Zhang, J.-J. Li, Distributed evolutionary algorithms and their models, Appl. Soft Comput. 34 (C) (2015) 286–300. `doi:10.1016/j.asoc.2015.04.061`. URL `http://dx.doi.org/10.1016/j.asoc.2015.04.061`

[23] B. Cao, J. Zhao, Z. Lv, X. Liu, A distributed parallel cooperative co-evolutionary multiobjective evolutionary algorithm for large-scale optimization, IEEE Transactions on Industrial Informatics 13 (4) (2017) 2030–2038. `doi:10.1109/TII.2017.2676000`.

[24] Z. H. Zhan, X. F. Liu, H. Zhang, Z. Yu, J. Weng, Y. Li, T. Gu, J. Zhang, Cloudde: A heterogeneous differential evolution algorithm and its distributed cloud version, IEEE Transactions on Parallel and Distributed Systems 28 (3) (2017) 704–716. `doi:10.1109/TPDS.2016.2597826`.

[25] D. A. V. Veldhuizen, J. B. Zydallis, G. B. Lamont, Considerations in engineering parallel multiobjective evolutionary algorithms, IEEE Transactions on Evolutionary Computation 7 (2) (2003) 144–173. `doi:10.1109/TEVC.2003.810751`.

[26] A. J. Nebro, J. J. Durillo, A Study of the Parallelization of the Multi-Objective Metaheuristic MOEA/D, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 303–317. `doi:10.1007/978-3-642-13800-3_32`. URL `http://dx.doi.org/10.1007/978-3-642-13800-3_32`

[27] J. J. Durillo, Q. Zhang, A. J. Nebro, E. Alba, Distribution of Computational Effort in Parallel MOEA/D, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 488–502. `doi:10.1007/978-3-642-25566-3_38`. URL `http://dx.doi.org/10.1007/978-3-642-25566-3_38`

[28] J. Ge, Z. Chen, Y. Wu, Y. E, H-SOFT: a heuristic storage space optimisation algorithm for flow table of openflow, Concurrency and Computation: Practice and Experience 27 (13) (2015) 3497–3509, cPE-13-0288.R1. `doi:10.1002/cpe.3206`.
URL `http://dx.doi.org/10.1002/cpe.3206`

[29] B. Cao, J. Zhao, Z. Lv, X. Liu, S. Yang, X. Kang, K. Kang, Distributed parallel particle swarm optimization for multi-objective and many-objective large-scale optimization, IEEE Access 5 (2017) 8214–8221. `doi:10.1109/ACCESS.2017.2702561`.

[30] X. Ma, F. Liu, Y. Qi, X. Wang, L. Li, L. Jiao, M. Yin, M. Gong, A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables, IEEE Transactions on Evolutionary Computation 20 (2) (2016) 275–298. doi:10.1109/TEVC.2015.2455812.

[31] J. J. Escobar, J. Ortega, J. González, M. Damas, Assessing Parallel Heterogeneous Computer Architectures for Multiobjective Feature Selection on EEG Classification, Springer International Publishing, Cham, 2016, pp. 277–289. doi:10.1007/978-3-319-31744-1_25.
URL http://dx.doi.org/10.1007/978-3-319-31744-1_25

[32] A. G. D. Nuovo, M. Palesi, V. Catania, Multi-objective evolutionary fuzzy clustering for high-dimensional problems, in: 2007 IEEE International Fuzzy Systems Conference, 2007, pp. 1–6. doi:10.1109/FUZZY.2007.4295660.

[33] Y. Zuo, M. Gong, J. Zeng, L. Ma, L. Jiao, Personalized recommendation based on evolutionary multi-objective optimization [research frontier], IEEE Computational Intelligence Magazine 10 (1) (2015) 52–62. doi:10.1109/MCI.2014.2369894.

[34] M. A. Potter, K. A. De Jong, A cooperative coevolutionary approach to function optimization, Springer Berlin Heidelberg, Berlin, Heidelberg, 1994, pp. 249–257. doi:10.1007/3-540-58484-6_269.
URL http://dx.doi.org/10.1007/3-540-58484-6_269

[35] F. van den Bergh, A. P. Engelbrecht, A cooperative approach to particle swarm optimization 8 (3) (2004) 225–239. doi:10.1109/TEVC.2004.826069.

[36] M. N. Omidvar, X. Li, X. Yao, Cooperative co-evolution with delta grouping for large-scale non-separable function optimization, in: Proc. IEEE Congr. Evol. Comput., 2010, pp. 1–8. doi:10.1109/CEC.2010.5585979.

[37] X. Li, X. Yao, Cooperatively coevolving particle swarms for large-scale optimization, IEEE Trans. Evol. Comput. 16 (2) (2012) 210–224. doi:10.1109/TEVC.2011.2112662.

[38] M. N. Omidvar, X. Li, Y. Mei, X. Yao, Cooperative co-evolution with differential grouping for large-scale optimization, IEEE Trans. Evol. Comput. 18 (3) (2014) 378–393. doi:10.1109/TEVC.2013.2281543.

[39] Y. Mei, M. N. Omidvar, X. Li, X. Yao, A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization, ACM Trans. Math. Softw. 42 (2) (2016) 13:1–13:24. doi:10.1145/2791291.
URL http://doi.acm.org/10.1145/2791291

[40] Y. Ling, H. Li, B. Cao, Cooperative co-evolution with graph-based differential grouping for large scale global optimization, in: 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), 2016, pp. 95–102. doi:10.1109/FSKD.2016.7603157.

[41] L. M. Antonio, C. A. C. Coello, Use of cooperative coevolution for solving large scale multiobjective optimization problems, in: 2013 IEEE Congress on Evolutionary Computation, 2013, pp. 2758–2765. doi:10.1109/CEC.2013.6557903.

[42] S. Huband, P. Hingston, L. Barone, L. While, A review of multiobjective test problems and a scalable test problem toolkit, IEEE Trans. Evol. Comput. 10 (5) (2006) 477–506. doi:10.1109/TEVC.2005.861417.

[43] J. Wang, W. Zhang, J. Zhang, Cooperative differential evolution with multiple populations for multiobjective optimization, IEEE Transactions on Cybernetics 46 (12) (2016) 2848–2861. doi:10.1109/TCYB.2015.2490669.

[44] Y. Wu, G. Min, K. Li, B. Javadi, Modeling and analysis of communication networks in multicluster systems under spatio-temporal bursty traffic, IEEE Transactions on Parallel and Distributed Systems 23 (5) (2012) 902–912. doi:10.1109/TPDS.2011.198.

[45] Y. Wu, G. Min, D. Zhu, L. T. Yang, An analytical model for on-chip interconnects in multimedia embedded systems, ACM Trans. Embed. Comput. Syst. 13 (1s) (2013) 29:1–29:19. doi:10.1145/2536747.2536751.
URL http://doi.acm.org/10.1145/2536747.2536751

[46] K. Deb, H. Jain, An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints, IEEE Transactions on Evolutionary Computation 18 (4) (2014) 577–601. `doi:10.1109/TEVC.2013.2281535`.

[47] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, V. G. da Fonseca, Performance assessment of multiobjective optimizers: An analysis and review, IEEE Transactions on Evolutionary Computation 7 (2) (2003) 117–132. `doi:10.1109/TEVC.2003.810758`.