Distributed Differential Evolution Based on Adaptive Mergence and Split for Large-Scale Optimization

Yong-Feng Ge, *Student Member, IEEE*, Wei-Jie Yu, *Member, IEEE*, Ying Lin, *Member, IEEE*, Yue-Jiao Gong, *Member, IEEE*, Zhi-Hui Zhan, *Member, IEEE*, Wei-Neng Chen, *Member, IEEE*, and Jun Zhang, *Fellow, IEEE*

***Abstract*—Nowadays, large-scale optimization problems are ubiquitous in many research fields. To deal with such problems efficiently, this paper proposes a distributed differential evolution with adaptive mergence and split (DDE-AMS) on subpopulations. The novel mergence and split operators are designed to make full use of limited population resource, which is important for large-scale optimization. They are adaptively performed based on the performance of the subpopulations. During the evolution, once a subpopulation finds a promising region, the current worst performing subpopulation will merge into it. If the merged subpopulation could not continuously provide competitive solutions, it will be split in half. In this way, the number of subpopulations is adaptively adjusted and better performing subpopulations obtain more individuals. Thus, population resource can be adaptively arranged for subpopulations during the evolution. Moreover, the proposed algorithm is implemented with a parallel master–slave manner. Extensive experiments are conducted on 20 widely used large-scale benchmark functions. Experimental results demonstrate that the proposed DDE-AMS could achieve competitive or even better performance compared with several state-of-the-art algorithms. The effects of DDE-AMS components, adaptive behavior, scalability, and parameter sensitivity are also studied. Finally, we investigate the speedup ratios of DDE-AMS with different computation resources.**

***Index Terms*—Adaptive population model, distributed differential evolution (DDE), large-scale optimization.**

I. Introduction

In THE last decades, various kinds of evolutionary algorithms (EAs) such as differential evolution (DE) [1]–[4], particle swarm optimization (PSO) [5], [6], ant colony optimization [7], and artificial bee colony [8] have been developed. These algorithms have achieved great success on many numerical and combinatorial optimization problems in recent years [9]–[14]. However, the performance of classical EAs often deteriorates rapidly as the dimensionality of the problem increases [15]. Since the optimization problems in science and engineering are increasingly complex nowadays, research into designing EAs that are capable of tackling large-scale optimization problems has drawn increasing attention [16]–[18].

To enhance the performance of EAs for large-scale optimization, both decomposition and nondecomposition approaches have been studied [19]–[22]. In the decomposition approaches, the divide-and-conquer strategy is adopted. Cooperative co-evolution (CC) proposed by Potter and De Jong [23] is a famous approach to decompose large-scale optimization problems. Due to the simplicity and efficiency of DE, various decomposition strategies based on DE have been proposed, such as random dynamic grouping [24], multilevel dynamic grouping [25], and differential grouping [26]. It is obvious that if the objective function of the problem at hand is separable, the problem decomposition can be trivial, while for nonseparable functions the problem decomposition could be a difficult task. In addition, the performance of CC approach is highly sensitive to the decomposition strategies.

Several approaches to tackle large-scale optimization problems without decomposition have also been considered, such as designing new operators [27]–[29], embedding local search [30], [31], and introducing structured population [32]–[34]. In the DEs with structured population, which is referred as the distributed DE (DDE), the population is partitioned into several subpopulations and each subpopulation evolves independently. This paper aims at designing an efficient DDE algorithm for the large-scale optimization problems.

The population model of DDE should be sophisticatedly designed since it has significant influence on the performance of the algorithm. In most existing DDEs [35]–[37], the population models are static, which means that the number of individuals in each subpopulation and the number of subpopulations are fixed during the evolution. In this way, the population resource could not be effectively reallocated according to the various situations during optimization. For example,

Manuscript received May 3, 2017; accepted July 9, 2017. Date of publication July 31, 2017; date of current version June 14, 2018. This work was supported by the National Natural Science Foundation of China under Grant 61502544 and Grant 61332002. This paper was recommended by Associate Editor P. N. Suganthan. *(Yong-Feng Ge and Wei-Jie Yu contributed equally to this work.) (Corresponding authors: Wei-Jie Yu; Jun Zhang.)*

Y.-F. Ge, W.-J. Yu, and Y. Lin are with Sun Yat-sen University, Guangzhou 510275, China (e-mail: ywj21c@163.com).

Y.-J. Gong, Z.-H. Zhan, W.-N. Chen, and J. Zhang are with the South China University of Technology, Guangzhou 510641, China (e-mail: junzhang@ieee.org).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org provided by the authors. The file contains additional figures and tables. The material is 3.14 MB in size.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2017.2728725

the promising search regions should be assigned with more population resource, while redundant population resource in bad performing subpopulations should be avoided. Although dynamic population models have been proposed [38]–[42], existing methods are not effective enough to fully utilize the population resource to improve the search efficiency. For example, in [38], [39], and [42], the population size is monotonically reduced, which may cause the insufficient of population resource in the latter stage of evolution. In [40] and [41], the individuals are dynamically arranged over the subpopulations. However, the overall number of subpopulations and their sizes are fixed, which could not adapt the population model to fit for different optimization scenarios.

To address the above issues, in this paper, a novel DDE with adaptive population model is proposed. Two novel operators named mergence and split are proposed for dynamically rearranging the population resource among subpopulations. Moreover, an adaptive strategy based on the contribution of subpopulations to the evolution is proposed to control the execution of mergence and split. The best performing subpopulation is more likely to capture at a promising search region and it should be rewarded by more population resource for a deeper search. On the contrary, if the merged subpopulation could not continuously provide competitive solutions, half of its individuals will be reallocated by the split operator. In this way, better performing subpopulations obtain more population resource for enhancing the search efficiency. The population resource is adaptively arranged and the effectiveness of population resource is maintained through the entire evolution process. In addition, the proposed algorithm is implemented in parallel to reduce the computation time.

The main contribution of this paper is listed as follows.

1) An adaptive population model is proposed, in which the population resource is adaptively arranged to improve the search efficiency.
2) Two novel operators named mergence and split are proposed to help dynamically arrange the individuals among the subpopulations.
3) An adaptive contribution-based strategy based on the performance of the subpopulations is designed for mergence and split.

Numerical experiments are conducted on 20 benchmark functions from CEC'2010 competition and comparisons have been made against several state-of-the-art approaches for large-scale optimization. Experimental results show that the proposed DDE with adaptive mergence and split (DDE-AMS) algorithm outperforms the competitors in terms of quality of solutions, convergence speed and statistical tests. Furthermore, the effects of DDE-AMS components, search behavior of DDE-AMS, parameter sensitivity of DDE-AMS, and speedup ratio of DDE-AMS are investigated.

The remainder of this paper is organized as follows. In Section II, a brief review of related work on large-scale optimization is presented. Subsequently, Section III briefly introduces the traditional DDE. Section IV describes the proposed DDE-AMS algorithm in detail, including the novel mergence and split operators and the adaptive contribution-based strategy. Extensive experiments with discussion are provided in Section V. Finally, Section VI draws the conclusion.

II. Related Work

Although the classical EAs have achieved great success in solving many numerical and combinatorial optimization problems, they often lose their efficiency and advantages when applied to large and complex problems. Based on this condition, various modified EAs for large-scale optimization have been proposed. Recently, DE has shown effectiveness in the existing approaches for large-scale optimization. This section presents a brief review on the representative existing DE-based approaches for tackling large-scale optimization. This review includes CC algorithms with problem decomposition strategy and nondecomposition approaches, in which the large-scale problems are solved without decomposition.

A. Cooperative Co-Evolution Approaches

CC is an effective method for solving large-scale optimization problems through a divide-and-conquer paradigm. Since DE is simple and efficient, various DE-based CC approaches have been proposed. The CC approaches can be divided into two categories: static and dynamic CC models. In the static CC model, n-dimensional problem is partitioned into k s-dimensional subproblems [23], [43]. Once the subcomponents are identified, they undergo the entire optimization. It is clear that the static approaches do not scale well as the dimensionality increases.

Therefore, a lot of studies have been interested in developing dynamic grouping approaches, such as random dynamic grouping and learning-based dynamic grouping. In the random dynamic grouping approaches, a problem is decomposed into k s-dimensional subproblems and the variables are randomly allocated to subcomponents in every co-evolutionary cycle [24], [25], [44], [45]. However, this approach is ineffective when the number of interacting variables grows. In practice, without prior knowledge about the problem, it is not clear how the problem should be decomposed. For utilizing the prior knowledge of interaction between variables, learning-based dynamic grouping approaches are designed, such as differential grouping [26], contribution-based grouping [46], and adaptive partitioning [47].

B. Nondecomposition Approaches

Yet another possibility of solving the large-scale optimization problems is to enhance the algorithm's performance and solve the problems without decomposition, which is referred as the nondecomposition approach. These nondecomposition approaches usually focus on the alteration such as modifying new operators, embedding local search, and introducing structured population to enhance the performance [48], [49]. The JADE mutation strategy "DE/current-to-$p$best" [50] was utilized to enhance a self-adaptive DE [31] for large-scale optimization. A DE based on generalized opposition-based learning [28] was proposed and a memetic DE combined with the multiple offspring sampling was proposed in [51].

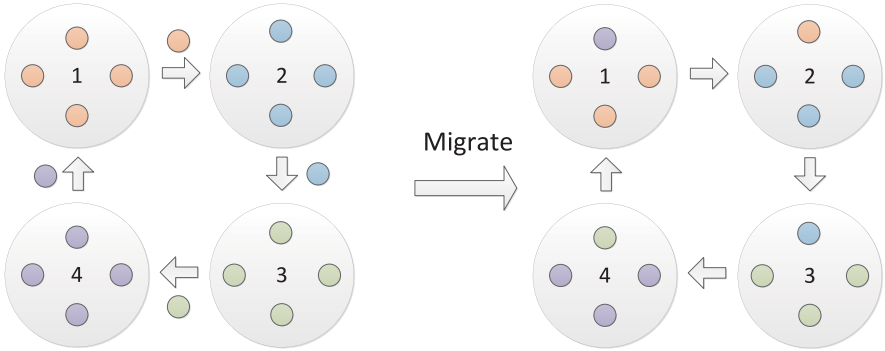


Fig. 1. DDE.

In the DE algorithm with structured population, i.e., DDE, the population is divided into several subpopulations and each subpopulation evolves independently. With a given probability, the individuals in the subpopulations are exchanged to maintain the population diversity, which is referred to migration. Structured population in the DE-based algorithms could help balance the exploration and exploitation search ability. Regarding the performance of DDE, migration and population model are two main research directions.

Migration is an important factor to prevent the search process from premature convergence or stagnation through injecting new individuals into subpopulations. Kozlov and Samsonov [52] introduced a new migration scheme in which the oldest member of the subpopulation is replaced by the received individual, instead of the worst one. A novel migration mechanism was proposed [53] that the substitution occurs only when the incoming solution is better than the one chosen to be replaced. De Falco *et al.* [54] introduced a migration model, which is inspired by the phenomenon known as biological invasion. In the novel model, the solutions with better fitness than the average value of subpopulation are sent to all neighbor subpopulations and stochastic universal sampling is applied for selection. Izzo *et al.* [55] studied an asynchronous migration strategy, in which subpopulations evolve independently according to the termination condition. Whenever the evolving process of one subpopulation stops, the best solution in this subpopulation is sent to the neighbor subpopulation. Weber *et al.* [39] employed a fitness diversity measure to activate the migration between subpopulations. Motivated by the principle "not too similar, not too different," Cheng *et al.* [56] proposed an enhanced DDE algorithm with multicultural migration.

The population model is a key factor in the success of DDE. To maintain the effectiveness of population resource, the population model in DDE, such as the population topology and the sizes of subpopulations, have been studied in various ways for improving the search efficiency. Zaharie and Petcu [35] presented a parallel distributed self-adaptive DE algorithm, which uses a random topology as the communication structure. To improve both speed and performance, a DDE with unidirectional ring topology was presented by Tasoulis *et al.* [32]. De Falco *et al.* [36], [37] utilized toroidal mesh topology as the population model to form DDE, in which a chosen solution is sent to all the neighbor subpopulations. In [57], the population is divided into subpopulations according to the von Neumann topology and local search is embedded for enhancing the balance between exploration and exploitation. Jeyakumar and Velayutham [58] and Thangavelu and Velayutham [59] studied the performance of cooperative DDEs, in which mixing classical DEs independently evolve in subpopulations. In [60] and [61], the influence of size and number of subpopulations in DDE was studied and analyzed in detail with help of standard benchmarks. Penas *et al.* [62] designed an asynchronous parallel implementation of DDE. However, all these population models are static, i.e., the sizes of subpopulations and the population topologies are unchanged during the evolution.

Other than the static population models, dynamic population models for DE have been designed to dynamically arrange the population resource. Unlike the static population models, the sizes or the number of subpopulations in these population models are dynamically changed during the optimization, which is helpful in maintaining the effectiveness of population resource during the evolution. Weber *et al.* [40] proposed a novel operator named shuffling, which helps randomly rearrange the individuals over the subpopulations. Experimental results exhibit that the novel operator is effective in promoting the balances between exploitation and exploration search process. In [41], a novel DDE simultaneously consisting of three mutation strategies was proposed. The big reward subpopulation is allocated to the best performing mutation strategy. Although individuals are dynamically rearranged over subpopulations [40], [41], the overall number of subpopulations and their sizes are fixed. Unlike these two, some approaches with dynamic subpopulation sizes have been proposed. In [39], the subpopulations are grouped into two families. Subpopulations in the first family arranged according to a ring topology have constant population size, and employ a best-random like migration strategy. The second family is composed of a subpopulation with dynamic population size, which is progressively reduced. Hendershot [38] proposed an extension of DE named MultiDE. In MultiDE, the number of subpopulations is kept variable, i.e., subpopulations can dynamically emerge and disappear. When an element from a subpopulation is similar to an element from population 0, the former is no longer considered for further evolution. Zamuda *et al.* [42] combined

the population reduction-based jDE [63] with two mutation schemes by using a structured population. However, the monotonically reduced population size may cause the insufficiency of population resource in the latter stage of evolution.

III. DDE

In this section, the classical DDE is introduced. In DDE, the population is uniformly partitioned into a number of islands at the beginning. During the evolution, DE performs independently in each island. We also refer to the populations in the islands as subpopulations of DDE, as it is commonly done in the literature. The subpopulations are arranged in a topology. This means that every subpopulation can send information to only one other subpopulation, and in the same way it can only receive information from one single subpopulation. With a given probability ϕ, some individuals are migrated to the neighbor subpopulations. When a subpopulation receives the migrated individual, it is inserted into its subpopulation replacing one individual. Through migration, the information is exchanged. Typically, the best individual in each subpopulation is sent to the neighbor and a random individual, which is different from the best individual will be replaced by the migrated individual. This process will allow the algorithm to better benefit from the diversity of solutions in different subpopulations.

Since each subpopulation evolves independently, it is natural to implement DDE in parallel. "Master–slave" is a popular mode for the implementation of DDE. As shown in Algorithm 1, master node charges for the global communication such as updating the best individual and migration between subpopulations. Each subpopulation is assigned to a slave node for independent evolution including mutation, crossover, and selection.

Algorithm 1 Pseudo-Code of DDE Algorithm

```
procedure GLOBAL CONTROLLER (AT MASTER NODE)
    set the generation counter g = 0
    spawn N subpopulations, each one on a different processor
    while not termination condition do
        if rand(0, 1) < φ then
            receive an individual from each subpopulation
            for each received individual do
                send the individual to its neighbor subpopulation in the ring
            end for
        end if
        g = g + 1
    end while
end procedure

procedure SUBPOPULATION (AT SLAVE NODE)
    for each generation do
        perform a DE step
        send a copy of the best individual to the master node
        if a migrated individual has been received then
            replace a random individual, different from the best, by the migrated individual
        end if
        if a termination signal has been received then
            terminate the execution
        end if
    end for
end procedure
```

A. Mutation

The evolution of individuals begins with the mutation operation. At each generation g, the mutation operation is applied to each individual $\mathbf{x}_i^g$ in the current population to create its corresponding mutant individual $\mathbf{v}_i^g$. The frequently used mutation strategies are listed as follows.

DE/rand/1

$$\mathbf{v}_i^g = \mathbf{x}_{r1}^g + F \cdot \left(\mathbf{x}_{r2}^g - \mathbf{x}_{r3}^g\right). \tag{1}$$

DE/current-to-best/1

$$\mathbf{v}_i^g = \mathbf{x}_i^g + F \cdot \left(\mathbf{x}_{\text{best}}^g - \mathbf{x}_i^g\right) + F \cdot \left(\mathbf{x}_{r1}^g - \mathbf{x}_{r2}^g\right). \tag{2}$$

DE/best/1

$$\mathbf{v}_i^g = \mathbf{x}_{\text{best}}^g + F \cdot \left(\mathbf{x}_{r1}^g - \mathbf{x}_{r2}^g\right). \tag{3}$$

DE/best/2

$$\mathbf{v}_i^g = \mathbf{x}_{\text{best}}^g + F \cdot \left(\mathbf{x}_{r1}^g - \mathbf{x}_{r2}^g\right) + F \cdot \left(\mathbf{x}_{r3}^g - \mathbf{x}_{r4}^g\right). \tag{4}$$

DE/rand/2

$$\mathbf{v}_i^g = \mathbf{x}_{r1}^g + F \cdot \left(\mathbf{x}_{r2}^g - \mathbf{x}_{r3}^g\right) + F \cdot \left(\mathbf{x}_{r4}^g - \mathbf{x}_{r5}^g\right) \tag{5}$$

where $\mathbf{x}_{\text{best}}^g$ indicates the best individual in the current population; $r1$, $r2$, $r3$, $r4$, and $r5$ are distinct integers randomly selected from the indexes of individuals, and they are all different from the index i; the factor F is a positive control parameter called differential factor for weighting the difference vectors. It can be seen that the mutant individual v_i^g is generated by combing a base vector with one or two scaled difference vectors.

B. Crossover

In order to enhance the population diversity, the mutation individual $\mathbf{v}_i^g$ is recombined with the target individual $\mathbf{x}_i^g$ to generate a trial individual $\mathbf{u}_i^g$. The process can be formulated as

$$\mathbf{u}_{i,j}^g = \begin{cases} \mathbf{v}_{i,j}^g, & \text{if rand } (0, 1) \le Cr \text{ or } j = j_{\text{rand}} \\ \mathbf{x}_{i,j}^g, & \text{otherwise} \end{cases} \tag{6}$$

where rand(0, 1) is a uniformly distributed random number; j_{rand} is a random integer generated once for each individual to make sure that at least one component of $\mathbf{u}_i^g$ is different from the target individual; and Cr indicates the crossover rate, which determines the fraction of $\mathbf{u}_i^g$ from the mutation individual.

C. Selection

After mutation and crossover, selection operation is carried out to compare the fitness of target and trial individuals. For

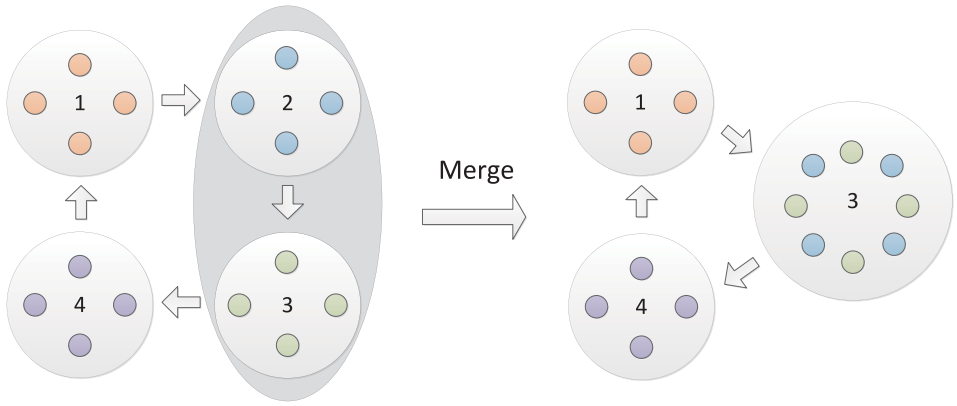


Fig. 2. Illustration of mergence operator. Subpopulation SP-2 is merged to SP-3.

a minimization problem, the selection can be expressed as follows:

$$\mathbf{x}_i^{g+1} = \begin{cases} \mathbf{u}_i^g, & \text{if } f(\mathbf{u}_i^g) \le f(\mathbf{x}_i^g) \\ \mathbf{x}_i^g, & \text{otherwise} \end{cases} \tag{7}$$

where $f(x)$ is the objective function of the optimization problem.

IV. DDE-AMS

In this section, a novel DDE algorithm named DDE-AMS with adaptive population model is proposed. Two operators named mergence and split are designed for dynamically arranging the population resource among subpopulations. Moreover, a contribution-based adaptive strategy for controlling mergence and split is proposed. Finally, the proposed DDE-AMS is algorithmically illustrated.

In the traditional DDE algorithm, the number of individuals in the subpopulations is fixed during the evolution. However, the static population model could not dynamically arrange the population resource to the subpopulations accordingly, which may cause the population resource lose effectiveness.

In the proposed DDE-AMS, to achieve a dynamic population arrangement, two operators named mergence and split are designed. On the one hand, to enhance the search efficiency at promising regions, individuals in the bad performing subpopulation will be merged into the good performing subpopulation. On the other hand, if a merged subpopulation could not continuously provide competitive solutions, these merged big subpopulations are split and half of the individuals will be reinitialized to maintain the diversity.

Furthermore, to indicate the performance of each subpopulation, an adaptive contribution value is defined. A higher contribution value means the subpopulation contributes more to the entire evolution. On the contrary, a lower contribution value indicates the subpopulation has less positive effect on the optimization process. Based on the contribution value, mergence and split are adaptively executed. If contribution value of a subpopulation keeps high, more individuals are allocated by mergence for enhancing the search efficiency. If contribution value of a merged subpopulation continuously decreases,

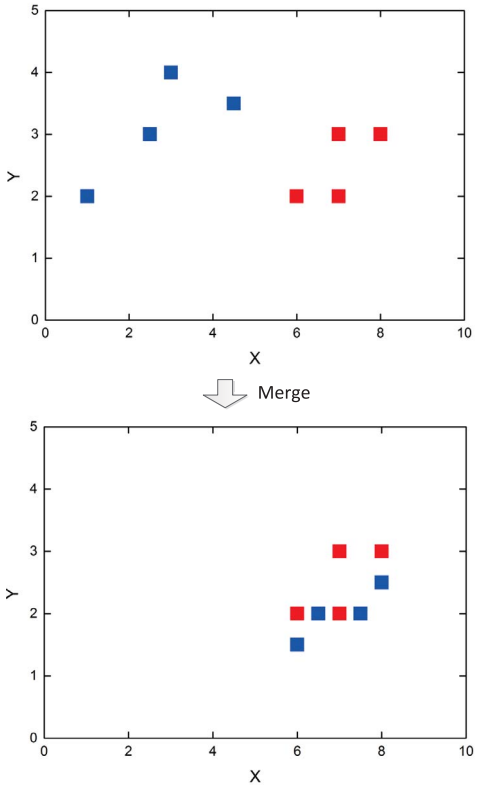


Fig. 3. Numerical example of merging two subpopulations in a 2-D search space. Individuals in the blue subpopulation are merged into the red subpopulation.

half of its individuals are released by split. Thus, the population resource is adaptively arranged to the subpopulations during the evolution.

A. Mergence Operator

Mergence operator assigns the population resource in bad performing subpopulations to promising ones. Suppose the condition for mergence is satisfied, all the individuals in one bad performing subpopulation are moved to one good performing subpopulation. Fig. 2 shows an example of mergence process, in which four individuals of subpopulation SP-2 are rearranged into subpopulation SP-3. As a result, the bad

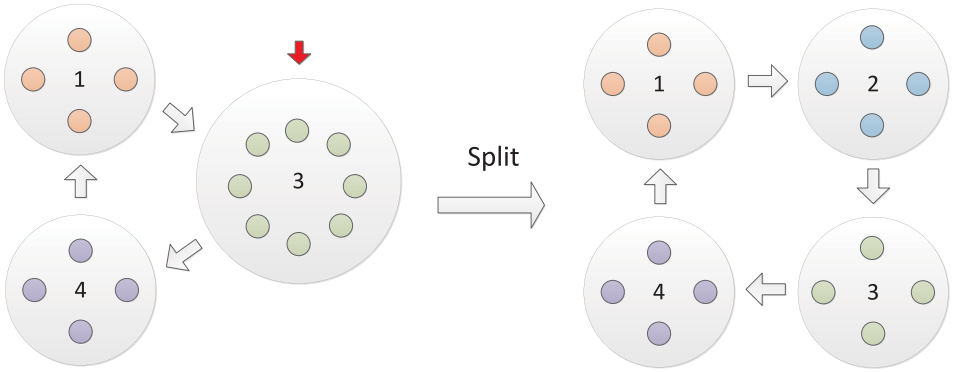


Fig. 4. Illustration of split operator. Subpopulation SP-3 is split into SP-2 and SP-3.

performing subpopulation SP-2 is removed and two subpopulations are combined to form one big subpopulation SP-3. There are three subpopulations remained in total.

Mutation operator "DE/best/1" is adopted for moving the individuals in bad performing subpopulation to the search region of good performing subpopulation. DE/best/1 is utilized due to the purpose of locating these new individuals around the best individual of good performing subpopulation. The new position of each individual i from the bad performing subpopulation is generated by the position of the best individual and two selected random individuals in the good performing subpopulation

$$\mathbf{x}_i^{g+1} = \mathbf{x}_{\text{best}}^g + F \cdot \left(\mathbf{x}_{r1}^g - \mathbf{x}_{r2}^g\right). \tag{8}$$

Fig. 3 shows a numerical example of merging two subpopulations in a 2-D search space. Suppose individuals in the red subpopulation have provided competitive performance, it is chosen as the good performing subpopulation. For merging the individuals in the bad performing blue subpopulation to the good performing red subpopulation, the mutation operator mentioned above is utilized. The best individual and selected random individuals in red subpopulation are used for calculating each new position of the blue individuals. As a result, the individuals from blue subpopulation are moved to the search region of red subpopulation.

B. Split Operator

Split operator helps release the redundant population resource in bad performing subpopulations. Suppose the condition for split is met by one subpopulation, half of its individuals are randomly selected and reinitialized to form a new subpopulation. Fig. 4 shows an example of the split process. Subpopulation SP-3 is split into two subpopulations with equal size, i.e., SP-2 and SP-3. Subsequently, the new subpopulation SP-2 is inserted into the unidirectional ring topology and involved in the communication among other subpopulations. Thus, one subpopulation is split into two subpopulation and the number of subpopulations increases to four. The position of each individual i in new subpopulation is randomly generated within the predefined boundaries of search space

$$\mathbf{x}_i^{g+1} = \mathbf{x}_{\min} + \text{rand}\ (0, 1) \cdot (\mathbf{x}_{\max} - \mathbf{x}_{\min}). \tag{9}$$

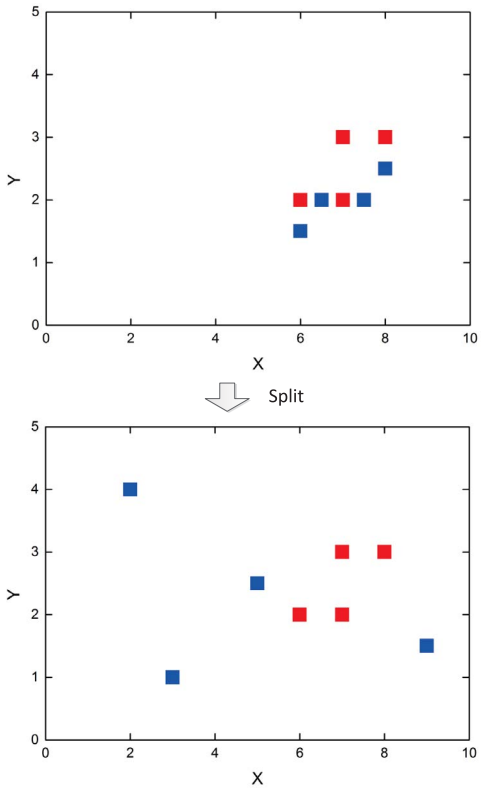


Fig. 5. Numerical example of splitting one subpopulation in a 2-D search space. The blue individuals are randomly selected and reinitialized to form a new subpopulation.

Fig. 5 shows an example of splitting a subpopulation containing eight individuals in a 2-D search space. As mentioned above, four blue individuals in the subpopulation are randomly selected to form a new subpopulation. Through the split process, four blue individuals in the new subpopulation are reinitialized in the entire search space.

C. Contribution-Based Strategy for Mergence and Split

First, an adaptive value is designed for indicating the contribution of each subpopulation to the entire evolution. Based on the contribution value, the adaptive execution of mergence and split is achieved. For subpopulation *si*, the value of

contribution is indicated by Con_{si}. For every U_p generations, the value of Con_{si} is updated. The initial value of Con_{si} is 0 and updated as follows:

$$\text{Con}_{si}^{g} = \begin{cases} \text{Con}_{si}^{g-1} + U_p * (1 - D_r) \\ \text{if subpopulation } i \text{ contains the best individual} \\ \text{Con}_{si}^{g-1} - U_p * D_r \\ \text{otherwise} \end{cases} \tag{10}$$

where D_r indicates the value Con_{si} decreases in every generation. The value of D_r is set between 0 and 1. According to (10), at a certain generation, Con_{si} of subpopulation i increases only when subpopulation i contains the best individual. For the other subpopulations, the value of Con_{si} decreases. If the value of Con_{si} is lower than 0, its value is reinitialized as 0. The definition of Con_{si} is based on the following considerations. On the one hand, if subpopulation i continuously generates competitive solutions, its value of Con_{si} will increase. A higher value of Con_{si} indicates that subpopulation i contributes more to the entire evolution. On the other hand, if the promising subpopulation i could not continuously provide competitive solutions, its value of Con_{si} decreases. A lower value of Con_{si} indicates subpopulation i has less positive effect on the optimization process.

In short, with the help of the decay rate D_r, the value of Con_{si} could be adaptively changed. Based on the analysis above, the value of Con_{si} effectively indicates the contribution as well as the evolutionary status of subpopulation i. Next, the adaptive strategy for execution of mergence and split is introduced.

1) When to Merge: Once the contribution value Con_{si} of subpopulation i exceeds the predefined threshold T, mergence is applied. Through comparing the best individuals in each subpopulation, the subpopulation w containing the worst one among these best individuals is chosen. Then, individuals in subpopulation w are sent to subpopulation i. According to the definition of Con_{si}, if the value of Con_{si} exceeds the threshold, it means subpopulation i has continuously provided competitive solutions. It is very likely that the search region of subpopulation i is promising. For enhancing the search at the promising region as well as maintaining the effectiveness of population resource, individuals from the bad performing subpopulation are merged. Since multiple subpopulations are helpful in maintaining population diversity, we predefine a minimal number of subpopulations N_m. Mergence operator is valid only when the number of subpopulations is greater than N_m.

2) When to Split: Split operator is also contribution driven. If the size of subpopulation i is bigger than the initial value, which means it has been merged at least once, it is defined as a big subpopulation. If Con_{si} of big subpopulation i decreases to the initial value 0, it will be selected as the split subpopulation and split up. According to the definition of Con_{si}, if the value of Con_{si} continuously decreases, it means subpopulation i could not provide competitive solutions lately. The main reason is that the search region of subpopulation i is not promising. For example, the search region of current big subpopulation contains a local optimum. Since the big subpopulation occupies more population resource, it is split up and half of the population resource is reinitialized to prevent the population resource from being wasted.

D. Overall Process

As shown in Algorithm 2, we use the master–slave parallel mode to implement the proposed DDE-AMS. For public use, the implementation is provided online. Compared with the traditional DDE, the additional part is the proposed adaptive mergence and split (AMS). U_p indicates the period to update the value of Con_{si}. At the beginning of each loop, the master node receives and updates the value of Con_{si}. According to "when to merge" and "when to split" mentioned in the last section, master node sends the mergence or split signal to the corresponding subpopulations in slave nodes for rearranging the population resource. Thus, the proposed DDE-AMS with adaptive population resource assignment is realized.

V. Experimental Results

A. Experimental Setup and Benchmark Functions

In this section, experiments are carried out to evaluate the performance of the proposed DDE-AMS. We use 20 benchmark functions chosen from IEEE CEC'2010 [64] special session on large-scale global optimization and the associated competition. These benchmark functions are classified into the following five groups.

1) Separable functions (F_1–F_3).
2) Single-group m-nonseparable functions (F_4–F_8).
3) $(n/2m)$ group m-nonseparable functions (F_9–F_{13}).
4) (n/m) group m-nonseparable functions (F_{14}–F_{18}).
5) Nonseparable functions (F_{19} and F_{20}).

n is the dimensionality of the problem and m is the number of variables in each nonseparable subcomponent. n and m are set to 1000 and 50, respectively. The characteristics of these functions and *MaxNFEs* are briefly summarized in Table SI of the supplementary file, available online at http://ieeexplore.ieee.org and more properties can be found in [64].

The parameters setting of DDE-AMS are summarized in Table I. To be specific, the overall population size NP is set to 300; the initial number of subpopulations is set as 10 and the minimal number of subpopulations is set as 4; U_p, T, and D_r are set as 25, 80, and 0.3, respectively; F and Cr in DE are set as 0.5 and 0.9 according to [2] and [65]; and ϕ is set as 0.05. In addition, each experiment runs 25 times independently.

Moreover, the code is written in C++. All the DDE-based algorithms are implemented based on Open MPI (open source high performance computing). Each subpopulation is assigned to one core and evolves in parallel. The proposed and all the compared algorithms are performed on a PC cluster system (Intel 4-core i5 CPU).

In the following experiments, the proposed DDE-AMS is compared with several state-of-the-art algorithms in terms of various performance metrics. Furthermore, the

Algorithm 2 Pseudo-Code of DDE-AMS Algorithm

```
procedure GLOBAL CONTROLLER (AT MASTER NODE)
    set the generation counter g = 0 and spawn N subpopulations, each one on a different processor
    while not termination condition do
        if rand(0, 1) < φ then
            receive an individual from each subpopulation
            for each received individual do
                send the individual to its neighbor subpopulation in the ring
            end for
        end if
        if g % U_p = 0 then
            receive Con_si from each subpopulation
            update Con_si for each subpopulation
            if condition of mergence operator is met then
                send mergence signal signal to the two corresponding subpopulations
            end if
            if condition of split operator is met then
                send split signal to the corresponding subpopulation
            end if
        end if
        g = g + 1
    end while
end procedure

procedure SUBPOPULATION (AT SLAVE NODE)
    for each generation do
        perform a DE step
        send a copy of the best individual to the master node
        if a migrated individual has been received then
            replace a random individual, different from the best, by the migrated individual
        end if
        if g % U_p = 0 then
            send Con_si to the master node
            if a mergence signal is received by bad performing subpopulation then
                send all the individuals to the mergence subpopulation
            end if
            if a mergence signal is received by good performing subpopulation then
                receive individuals from the cleared subpopulation
            end if
            if a split signal is received then
                send half of the randomly selected individuals to form a new subpopulation
            end if
        end if
    end for
end procedure
```

effects of DDE-AMS components, adaptive behavior, scalability, parameter sensitivity, and speedup ratios of DDE-AMS are investigated. Each experiment is run 25 times independently.

B. Performance Metrics

To evaluate the performance of DDE-AMS, several performance metrics are considered and they can be classified into the following groups.

1) *Solution Quality:* We use mean and standard deviation of errors to evaluate the solution quality. The error of a solution is defined as the difference between the final solution and the optimal value. The optimum for each function can be referred to [64].
2) *Statistic Test:* To compare the solution quality from an statistic angle, Wilcoxon rank-sum test [66] at a significance level 0.05 is employed to compare the performance of two algorithms from a statistical perspective. It is utilized to check whether there is a

TABLE I
PARAMETER SETTING

Parameter	Definition	Value
NP	Overall population size	300
N_i	Initial number of sub-populations	10
N_m	Minimal number of sub-populations	4
U_p	Update period	25
T	Threshold	80
D_r	Decay rate	0.3
F	Mutation rate	0.5
Cr	Crossover rate	0.9
ϕ	Migration probability	0.05

significant difference between two algorithms on one function using the errors over the total runs.

3) *Convergence Speed:* Convergence curve is employed to investigate the convergence speed of the algorithms. The convergence curve of the average errors over the number of generations is used to illustrate the convergence performance of an algorithm.
4) *Running Time and Speedup Ratio:* Since the proposed DDE-AMS is implemented in parallel, running time and speedup ratio are two important factors which should be investigated. Speedup ratio is defined as the ratio of the serial running time to the parallel running time.

Due to the space limit, all numerical values of the simulations are presented in the supplementary file, available online at http://ieeexplore.ieee.org.

C. Effect of Adaptive Mergence and Split

The main component of DDE-AMS is the AMS. To investigate its effectiveness, two experiments are performed in this section. In the first part, to show the effectiveness of AMS, six DDE-AMS algorithms with different DE mutation strategies are implemented and compared with their corresponding DDE algorithms without AMS. The DDE-AMS with DE/best/1 mutation strategy, for example, is denoted as DDE-AMS-best-1. These six DDE-AMS algorithms are listed as follows.

1) DDE-AMS-best-1.
2) DDE-AMS-rand-1.
3) DDE-AMS-current-to-best.
4) DDE-AMS-best-2.
5) DDE-AMS-rand-2.
6) DDE-AMS-current-to-rand.

Tables SII and SIII of the supplementary file show the comparisons of experimental results where the better results are highlighted in boldface. The performance of all the six traditional DDEs is enhanced by the proposed AMS operators. To show the advantage of the proposed DDE-AMS algorithm in a statistical sense, Wilcoxon rank-sum test at a significance level 0.05 is performed. The errors for each function over 25 independent runs are used to conduct the test. As shown in the tables, data in each cell is represented in a "$-/\approx/+$" manner, where "$-$," "$\approx$," and "$+$" denote that the algorithm is significantly worse than, equivalent to, and better than its corresponding traditional DDE algorithm, respectively. It is clear that the DDE with AMS are able to obtain significantly better results than the original versions on the majority of functions.

Furthermore, to select the best DDE-AMS, we compare DDE-AMS-best-1 with other DDE-AMS algorithms. In Table SIV of the supplementary file, the DDE-AMS-Best-1's numbers of wins, ties, and loses against the other DDE-AMS variants are measured. We could conclude that the DDE-AMS approach using DE/best/1 could offer significantly better performance than the others in terms of solution accuracy. Hence, DDE-AMS-best-1 is adopted as the representative for the rest experiments.

In the second part, considering our proposed DDE-AMS could adaptively change the number of subpopulations during the evolution, the effectiveness of the adaptive population model can be investigated by comparing DDE-AMS with traditional DDEs with different fixed number of subpopulations. The DDE algorithm with ten fixed subpopulations, for example, is denoted as DDE-10.

The experimental results obtained by these algorithms are listed in Table SV of the supplementary file. In general, the evolution in DDEs with smaller number of subpopulations is relatively greedy. As a result, these DDEs outperform on unimodal functions such as F_1, F_7, and F_{12}. On the contrary, with a bigger number of subpopulations, DDEs evolve with a higher population diversity and could achieve better performance on multimodal functions such as F_2, F_8, and F_{20}. Since the population resource is adaptively arranged in the proposed DDE-AMS algorithm, it is effective in various search space. To show the advantage of the dynamic population model in a statistical sense, the same significant test is performed. It is clear that the proposed DDE-AMS could achieve significantly better results than all the compared DDEs on most of test functions.

D. Comparisons With State-of-the-Art DDE Variants

In this section, we compare the proposed DDE-AMS algorithm with three well-known DDE variants to further reveal its advantage. The population models of the first two DDE algorithms are static while the last one is dynamic. These three DDE variants are listed as follows and their parameters are set according to their original papers.

1) *PDE [32]:* This variant adopts a modified best-random migration strategy. With a given probability, migration is executed. The best individual in the emigrated subpopulation takes the place of a randomly selected individual except for the best one in the immigrated subpopulation.
2) *DDEM [56]:* This variant makes use of two migration selection approaches to maintain a high diversity in the subpopulations, i.e., target individual-based migration selection and representative individual-based migration selection, respectively. In addition, the diversity amongst the individuals is controlled by means of an affinity-based replacement strategy.
3) *SOUPDE [40]:* In this variant, a novel operator named shuffling is proposed, which helps randomly rearrange the individuals over the subpopulations.

The mean and standard deviations of the errors over 25 independent runs for all test functions are presented in Table SVI of the supplementary file, where the best results are highlighted in boldface. Our proposed DDE-AMS algorithm comprehensively outperforms these three in terms of solution accuracy. DDE-AMS achieves the best results on 16 functions, which is much higher than the figures 0, 4, and 0 of PDE, DDEM, and SOUPDE, respectively.

To show the advantage of proposed DDE-AMS algorithm in a statistical sense, Wilcoxon rank-sum test is performed. DDE-AMS is significantly better than PDE, DDEM, and SOUPDE on 16, 16, and 20 functions, while significantly worse than PDE, DDEM, and SOUPDE on only 4, 4, and 0 functions, respectively.

In addition, the convergence curves of four DDE algorithms on six typical functions are plotted in Fig. S1 of the supplementary file by taking the average of 25 independent runs. F_7, F_{12}, and F_{19} are unimodal functions while F_8, F_{13}, and F_{18} are multimodal functions. Convergence curves of these six typical functions clearly show that DDE-AMS converges fastest to achieve the highest solution accuracy among the four compared algorithms. On the one side, for unimodal functions, the proposed algorithm shows the best convergence performance. On the other side, when optimizing the multimodal functions, DDE-AMS exhibits much stronger global search ability than the other three algorithms.

E. Comparisons With State-of-the-Art DE-Based CC Algorithms

Since CC is a famous approach to tackle large-scale optimization problems, experiments are carried out to compare DDE-AMS with three state-of-the-art DE-based CC algorithms, namely, CC with random grouping (DECC-G) [24], multilevel CC (MLCC) [25], and CC with differential grouping (DECC-DG) [26]. These three state-of-the-art DE-based CC algorithms are listed as follows and their parameters are set according to their original papers.

1) *DECC-G:* Random grouping decomposes a problem into k s-dimensional subproblems, but instead of using a static grouping, it randomly allocates the decision variables to subcomponents in every co-evolutionary cycle. It was shown mathematically that with random grouping the probability of placing two interacting variables in the same subcomponent for several cycles is reasonably high.
2) *MLCC:* This algorithm designed a multilevel CC framework for large-scale optimization problems. The self-adapted mechanism was proposed to select a decomposer according to its historical performance at the start of each cycle.
3) *DECC-DG:* In this algorithm, an automatic decomposition strategy is proposed, called differential grouping that can uncover the underlying interaction structure of the decision variables and form subcomponents such that the interdependence between them is kept to a minimum.

In Table SVII of the supplementary file, the mean and standard deviations of the errors over 25 independent runs are presented and the best results are highlighted in boldface. Overall, DDE-AMS achieves the best results on 12 functions. To show the advantage of the proposed DDE-AMS algorithm in a statistical sense, Wilcoxon rank-sum test is performed and result are listed in Table SVII of the supplementary file.

On closer inspection, one can find that DECC-G and MLCC outperform DDE-AMS on all the separable functions. However, on nearly all the nonseparable functions, DDE-AMS could achieve better performance with the help of adaptive population resource arrangement. For DECC-DG, with the help of differential grouping, it outperforms on part of the nonseparable functions, namely, F_5, F_6, F_{15}, and F_{16}. On all the separable functions and other nonseparable functions, the differential grouping could not provide equivalent help. Overall, DDE-AMS achieves significantly better performance than the compared DECC-G, MLCC, and DECC-DG on most of the functions.

In addition, Fig. S2 of the supplementary file shows the convergence curves of four approaches on the same six typical functions. Each point on the plot is calculated by taking the average of 25 independent runs. Take the convergence curves DECC-DG as an example, with the help of differential grouping, it performs better in F_7, F_{12}, and F_{19} than the other two DE-based CC algorithms. However, in F_8, F_{13}, and F_{18}, the search converges in early stage of evolution, which is also because of the grouping strategy. The performance of these DE-based CC algorithms depends a lot on the separability of the test functions. Through the adaptive population resource arrangement, DDE-AMS exhibits the best search performance on all these six typical functions. To summarize, by using the adaptive population model, the proposed DDE-AMS is a very promising algorithm for large-scale optimization in terms of solution accuracy and search efficiency.

F. Comparisons With Other State-of-the-Art Algorithms

In this section, experiments are carried out to compare the proposed DDE-AMS algorithm with another five state-of-the-art algorithms including the winner of CEC'2010, i.e., MA-SW-Chains. These five state-of-the-art algorithms are listed as follows.

1) *MA-SW-Chains [67]:* This memetic algorithm is the top ranked algorithm in the CEC'2010 special session and competition on large-scale global optimization. It assigns each individual a local search intensity that depends on its features, by chaining different local search applications.
2) *CSO [16]:* This PSO variant introduces a new competitive learning strategy for PSO and shows its promising performance in handling large-scale optimization.
3) *SL-PSO [68]:* SL-PSO is another PSO variant for large-scale optimization, which is embedded by a social learning strategy.
4) *DMS-L-PSO [69]:* DMS-L-PSO algorithm is a multiswarm PSO variant, where multiple swarms are dynamically formed during each generation.

5) *CCPSO2 [70]:* CCPSO2 is a PSO-based CC algorithm, in which the grouping method is random grouping and the group size is randomly selected from a pool.

In Table SVIII of the supplementary file, the experimental results over 25 independent runs are presented and the best results are highlighted in boldface. It can be seen that our proposed DDE-AMS comprehensively outperforms these five state-of-the-art algorithms. DDE-AMS achieves the best results on eight functions, which is much higher than the figures 5, 4, 1, 1, and 1 of MA-SW-Chains, CSO, SL-PSO, DMS-L-PSO, and CCPSO2, respectively.

In addition, we make use of the Wilcoxon rank-sum test to evaluate the statistical significance of the results. Overall, DDE-AMS achieves significantly better results on more functions. It is significantly better than MA-SW-Chains, CSO, SL-PSO, DMS-L-PSO, and CCPSO2 on 10, 13, 13, 15, and 17 functions, Conversely, MA-SW-Chains, CSO, SL-PSO, DMS-L-PSO, and CCPSO2 surpass DDE-AMS on 8, 7, 6, 5, and 3 functions, respectively.

To sum up, our proposed DDE-AMS remains very competitive with these five state-of-the-art algorithms for large-scale optimization.

G. Scalability to Higher Dimensionality

To further evaluate the scalability of DDE-AMS to higher dimensionality, experiments are conducted on DDE-AMS for optimizing 3000-D and 5000-D by extending the dimensions of test functions in the CEC'2010 to 3000 and 5000.

The proposed DDE-AMS algorithm is compared with three state-of-the-art DE-based CC algorithms. Here, state-of-the-art DE-based CC algorithms are chosen because these algorithms are based on the divide-and-conquer strategy, which has been proven to have good scalability.

Tables SIX and SX of the supplementary file exhibit the comparison results on 3000-D and 5000-D functions, respectively. The mean and standard deviations of the errors over 25 independent runs are presented with highlighted best mean values. For most test functions, the performance of DDE-AMS remains superior to that of the compared CC algorithms. Our proposed DDE-AMS achieves the best results on 13 functions with 3000-D and 14 functions with 5000-D, respectively.

Wilcoxon rank-sum test is also performed to show the advantage of DDE-AMS algorithm in a statistical sense. On the 3000-D functions, DDE-AMS is significantly better than DECC-G, MLCC, and DECC-DG on 16, 17, and 16 functions, respectively. On the 5000-D functions, DDE-AMS significantly outperforms the compared CC algorithms on 16, 17, and 17 functions.

On closer inspection, we can find that DECC-G and MLCC outperform our proposed algorithm on all three separable functions due to their divide-and-conquer strategy. However, on all the nonseparable functions expect F_{10}, DDE-AMS achieves the best performance. For DECC-DG, it cannot obtain any solution (denoted by N/A in Tables SIX and SX of the supplementary file) on all the three separable functions and two nonseparable functions (5000-D F_5 and 5000-D F_6). This is because the differential grouping strategy of DECC-DG costs more evaluations than the predefined maximum number of evaluations.

In summary, this series of experiments demonstrate the good scalability of our proposed DDE-AMS algorithm to higher dimensionality.

H. Effect of Adaptive Population Model on Existing Well-Known DDE Variants

In this section, experiments are carried out to verify the effectiveness of the proposed adaptive population model on DDE variants with static population models, namely, PDE [32] and DDEM [56]. The algorithm named SOUPDE [40] mentioned above is not included here, because it has its own dynamic population model. Two algorithms, namely, PDE-AMS and DDEM-AMS, are implemented by embedding the proposed AMS into the corresponding DDE variants. The AMS parameters are set according to DDE-AMS and the other parameters of two algorithms are set according to their original papers.

For each approach, 25 independent runs are carried out. Table SXI of the supplementary file shows the results and the better results are marked in bold. It can be observed that, with the help of AMS, the performance of both PDE and DDEM algorithms is enhanced. PDE-AMS is significantly better than PDE on all the 20 test functions. DDEM-AMS could achieve significantly better results in 19 functions, while significantly worse than DDEM on only one function. To sum up, our proposed adaptive population model is effective for these two DDE variants with static population model.

I. Adaptive Arrangement of Population Resource

Through observing the variation of subpopulation number during the evolution, the adaptive population resource arrangement of DDE-AMS is investigated. The variation on three typical unimodal functions (F_1, F_{12}, and F_{19}) and three typical multimodal functions (F_5, F_{11}, and F_{16}) is plotted in Fig. S3 of the supplementary file. In the figure, the x-axis represents the number of generations and the y-axis represents the number of subpopulations.

According to the Fig. S3 of the supplementary file, for the three unimodal functions, the number of subpopulations monotonically decreases. This is because the subpopulation near the global optimum could continuously provide competitive solutions and obtain more individuals from the bad performing subpopulations. In this way, the population resource is attracted by the best optimum individual, which is beneficial for the unimodal functions. The variation is quite different on the three multimodal functions. Mergence is also applied on the best performing subpopulations. However, these best performing subpopulations might be trapped in local optima and could not continuously provide competitive solutions. In this case, some of the merged subpopulations are split. With the help of adaptive contribution-based strategy, mergence and split perform alternative to enhance the search efficiency at the promising search region. Such kind of population arrangement is effective for the optimization of multimodal functions.

J. Sensitivity Analysis

In DDE-AMS, the population resource is adaptively arranged with the help of contribution-based strategy. In this case, the performance of DDE-AMS may be sensitive to the selection of parameters in contribution-based strategy, namely, update period U_p, threshold T, decay rate D_r, and migration probability ϕ. If the update period is too low, due to the cost of communication between subpopulations, the running time of DDE-AMS is long. On the contrary, if the evolutionary status of subpopulations could not be updated in time, the execution of mergence or split might be delayed. In this case, the effectiveness of population arrangement could not be maintained. In addition, if the threshold or decay rate is too low, merge is frequently applied and population resource cannot be assigned in time to the most promising search region. If the threshold or decay rate is too high, the condition of mergence operator is hard to satisfy and thus the population resource could not be allocated to the promising search region. If the migration probability is too low, there would be little information exchanged between subpopulations. On the contrary, if the migration probability is too high, migration is frequently applied and the population diversity would be low.

To test the sensitivity of these four parameters, we varied one parameter at a time while keeping the other two parameters fixed as the values summarized in Table I. Table SXII of the supplementary file shows the results of the sensitivity tests. We performed 25 independent runs for every set of parameters. Note that Wilcoxon rank-sum test with a 0.05 significance level is additionally performed. The labels "wins," "ties," and "loses" indicates whether the corresponding parameters is, respectively, better than, equal to, or worse than the results obtained by the compared versions. The results presented in Table SXII suggest that although tuning all of these parameters affects the quality of the obtained results, these differences are not significant. In addition, the chosen combination of these four parameters in our DDE-AMS, namely, $U_p = 25$, $T = 80$, $D_r = 0.3$, and $\phi = 0.05$ could achieve the best performance.

Also, performance of DDE-AMS may be sensitive to the selected overall population size NP and initial number of subpopulations N_i. If the overall population size is large and the initial number of subpopulations is small, which means the best individual is selected from a large group of individuals, the evolution would be relatively greedy. On the contrary, if the best individual is selected from a small group of individuals and the initial number of subpopulations is large, the evolution would be relatively diverse. In order to investigate the sensitivity of DDE-AMS to the combination of the overall population size and the initial number of subpopulations, we compare DDE-AMS with different values of these two parameters.

Each combination of NP and N_i runs on all the 20 test functions and the ranks are averaged. The average ranks achieved by 12 combinations are plotted in Fig. S4 of the supplementary file. Through comparing these average ranks, we could find that the average ranks of approaches based on five initial subpopulations are high. This is mainly because of the insufficient of population diversity in the early stage of optimization. However, the ranks obtained by other combinations do not make big differences. To sum up, the performance of DDE-AMS is not so sensitive to the overall population size and the initial number of subpopulations. Overall, a population size of 300 and initially dividing the population into 10 subpopulations is a suitable combination to achieve the lowest average rank.

K. Speedup Ratio

One of the advantages of DDE is the easy parallel implementation. Since each subpopulation in DDE can evolve independently, our proposed DDE-AMS is implemented in parallel and each subpopulation is assigned to a computing node. Thus, the initial number of subpopulations affects the parallel granularity of DDE-AMS during the evolution.

The serial running time of DDE-AMS with one subpopulation and parallel running time of DDE-AMS algorithms with different numbers of initial subpopulations (2, 5, 10, 15, 20, and 30) are first recorded. Note that the overall population sizes of these algorithms are uniformly set to 300. Then, speedup ratios achieved by DDE-AMS with different initial numbers of subpopulations are calculated and reported in Table SXIII of the supplementary file. It can be seen that the speedup ratios vary with the initial number of subpopulations and the test function. As we expect, the DDE-AMS with larger initial number of subpopulations could achieve higher speedup ratios on most of the test functions.

Speedup ratio is a metric to measure run time performance of a parallel algorithm, and it is sensitive to the computation cost of test function. Fig. S5 of the supplementary file is plotted to further visualize the variation tendency of the speedup ratios on six typical functions F_3, F_6, F_{10}, F_{11}, F_{15}, and F_{16} with increasing computation costs. It can be observed that the speedup ratios consistently grows with the increase of the initial number of subpopulations on all six typical functions. On closer inspection, we can find that the increase of speedup ratio slows down when the initial number of subpopulations increases from 20 to 30, this can be due to the following reason. With the increase of number of nodes, the parallel computation cost reduces while the communication cost between different nodes increases. When the number of nodes increases to a certain value, the increase amount of communication cost may exceed the reduction of computation cost. In this case, the parallel running time may increase in reverse and cause the decrease of speedup ratio. For this experiment, when the number of nodes increases from 20 to 30, the increase rate of communication cost is higher than the decrease rate of computation cost. Thus, the increase of speedup ratio slows down. However, the speedup ratio still keeps growing in the interval between 20 and 30.

To sum up, our proposed DDE-AMS could achieve a high speedup ratio due to the good balance between computation and communication overhead.

VI. Conclusion

In this paper, a novel DDE algorithm with adaptive population model named DDE-AMS has been proposed for large-scale optimization. Two novel operators, namely,

mergence and split are designed and performed according to the performance of the subpopulations. With the help of mergence, the recently best performing subpopulation is rewarded by more population resource. If the merged subpopulation could not continuously provide competitive solutions, split is applied to release the redundant population resource. Driven by the contribution-based adaptive strategy, adaptive population resource arrangement is achieved.

Numerical experiments have been conducted on 20 benchmark large-scale optimization functions. The experimental results show that the performance of DDE is enhanced by the adaptive arrangement of population resource. Moreover, comparisons have been made against several state-of-the-art DDE algorithms and DE-based CC algorithms. It can be concluded that the proposed DDE-AMS generally outperforms the competitors in terms of solution quality, convergence speed, and statistical tests. Furthermore, mergence, split, and contribution-based adaptive strategy have been investigated and shown to be effective. The parameters of DDE-AMS are not sensitive and speedup ratio achieved by DDE-AMS is relatively high.

For future work, we will apply the DDE-AMS algorithm to real-world large-scale optimization applications. In addition, we can incorporate the proposed adaptive population model into other DEAs to further verify its effectiveness.

REFERENCES

[1] K. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Berlin, Germany: Springer-Verlag, 2005.

[2] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Glob. Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[3] J.-H. Zhong *et al.*, "A differential evolution algorithm with dual populations for solving periodic railway timetable scheduling problem," *IEEE Trans. Evol. Comput.*, vol. 17, no. 4, pp. 512–527, Aug. 2013.

[4] W.-J. Yu *et al.*, "Differential evolution with two-level parameter adaptation," *IEEE Trans. Cybern.*, vol. 44, no. 7, pp. 1080–1099, Jul. 2014.

[5] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. Int. Symp. Micro Mach. Human Sci.*, vol. 1. Nagoya, Japan, 1995, pp. 39–43.

[6] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, Perth, WA, Australia, 1995, pp. 1942–1948.

[7] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 26, no. 1, pp. 29–41, Feb. 1996.

[8] W.-J. Yu, Z.-H. Zhan, and J. Zhang, "Artificial bee colony algorithm with an adaptive greedy position update strategy," *Soft Comput.*, to be published, doi: 10.1007/s00500-016-2334-4.

[9] C.-F. Juang, T.-L. Jeng, and Y.-C. Chang, "An interpretable fuzzy system learned through online rule generation and multiobjective ACO with a mobile robot control application," *IEEE Trans. Cybern.*, vol. 46, no. 12, pp. 2706–2718, Dec. 2016.

[10] B. Xue, M. Zhang, and W. N. Browne, "Particle swarm optimization for feature selection in classification: A multi-objective approach," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 1656–1671, Dec. 2013.

[11] N. García-Pedrajas, C. Hervás-Martínez, and J. Muñoz-Pérez, "COVNET: A cooperative coevolutionary model for evolving artificial neural networks," *IEEE Trans. Neural Netw.*, vol. 14, no. 3, pp. 575–596, May 2003.

[12] S. Jiang and S. Yang, "An improved multiobjective optimization evolutionary algorithm based on decomposition for complex Pareto fronts," *IEEE Trans. Cybern.*, vol. 46, no. 2, pp. 421–437, Feb. 2016.

[13] Y.-J. Gong, Y.-F. Ge, J.-J. Li, J. Zhang, and W. H. Ip, "A splicing-driven memetic algorithm for reconstructing cross-cut shredded text documents," *Appl. Soft Comput.*, vol. 45, pp. 163–172, Aug. 2016.

[14] W.-J. Yu, J.-Z. Li, W.-N. Chen, and J. Zhang, "A parallel double-level multiobjective evolutionary algorithm for robust optimization," *Appl. Soft Comput.*, vol. 59, pp. 258–275, Oct. 2017.

[15] T. Weise, R. Chiong, and K. Tang, "Evolutionary optimization: Pitfalls and booby traps," *J. Comput. Sci. Technol.*, vol. 27, no. 5, pp. 907–936, 2012.

[16] R. Cheng and Y. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 191–204, Feb. 2015.

[17] Q. Yang *et al.*, "Segment-based predominant learning swarm optimizer for large-scale optimization," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2016.2616170.

[18] W. Chu, X. Gao, and S. Sorooshian, "A new evolutionary search strategy for global optimization of high-dimensional problems," *Inf. Sci.*, vol. 181, no. 22, pp. 4909–4927, 2011.

[19] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "Test problems for large-scale multiobjective and many-objective optimization," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2016.2600577.

[20] Y.-F. Zhang and H.-D. Chiang, "A novel consensus-based particle swarm optimization-assisted trust-tech methodology for large-scale global optimization," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2016.2577587.

[21] D. Molina, M. Lozano, A. M. Sánchez, and F. Herrera, "Memetic algorithms based on local search chains for large scale continuous optimisation problems: MA-SSW-Chains," *Soft Comput.*, vol. 15, no. 11, pp. 2201–2220, 2011.

[22] L.-Y. Tseng and C. Chen, "Multiple trajectory search for large scale global optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2008, pp. 3052–3059.

[23] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *Parallel Problem Solving From Nature—PPSN III*. Berlin, Germany: Springer-Verlag, 1994, pp. 249–257.

[24] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Inf. Sci.*, vol. 178, no. 15, pp. 2985–2999, 2008.

[25] Z. Yang, K. Tang, and X. Yao, "Multilevel cooperative coevolution for large scale optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2008, pp. 1663–1670.

[26] M. N. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 378–393, Jun. 2014.

[27] B. Kazimipour, X. Li, and A. K. Qin, "Effects of population initialization on differential evolution for large scale optimization," in *Proc. IEEE Congr. Evol. Comput.*, Beijing, China, 2014, pp. 2404–2411.

[28] H. Wang, Z. Wu, and S. Rahnamayan, "Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems," *Soft Comput.*, vol. 15, no. 11, pp. 2127–2140, 2011.

[29] Z. Yang, K. Tang, and X. Yao, "Scalability of generalized adaptive differential evolution for large-scale continuous optimization," *Soft Comput.*, vol. 15, no. 11, pp. 2141–2155, 2011.

[30] A. Caponio, A. V. Kononova, and F. Neri, "Differential evolution with scale factor local search for large scale problems," in *Computational Intelligence in Expensive Optimization Problems*. Berlin, Germany: Springer-Verlag, 2010, pp. 297–323.

[31] S.-Z. Zhao, P. N. Suganthan, and S. Das, "Self-adaptive differential evolution with multi-trajectory search for large-scale optimization," *Soft Comput.*, vol. 15, no. 11, pp. 2175–2185, 2011.

[32] D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis, "Parallel differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, vol. 2. Portland, OR, USA, 2004, pp. 2023–2029.

[33] M. Weber, F. Neri, and V. Tirronen, "A study on scale factor in distributed differential evolution," *Inf. Sci.*, vol. 181, no. 12, pp. 2488–2511, 2011.

[34] M. Weber, V. Tirronen, and F. Neri, "Scale factor inheritance mechanism in distributed differential evolution," *Soft Comput.*, vol. 14, no. 11, pp. 1187–1207, 2010.

[35] D. Zaharie and D. Petcu, "Parallel implementation of multi-population differential evolution," in *Proc. 2nd Workshop Concurrent Inf. Process. Comput.*, Timişoara, Romania, 2003, pp. 223–232.

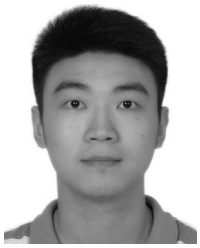
[36] I. De Falco, A. D. Cioppa, D. Maisto, U. Scafuri, and E. Tarantino, "Satellite image registration by distributed differential evolution," in *Applications of Evolutionary Computing*. Berlin, Germany: Springer-Verlag, 2007, pp. 251–260.

[37] I. De Falco, U. Scafuri, E. Tarantino, and A. D. Cioppa, "A distributed differential evolution approach for mapping in a grid environment," in *Proc. IEEE Int. Conf. Parallel Distrib. Netw. Based Process.*, Naples, Italy, 2007, pp. 442–449.

[38] Z. V. Hendershot, "A differential evolution algorithm for automatically discovering multiple global optima in multidimensional, discontinuous spaces." in *Proc. MAICS*, Chicago, IL, USA, 2004, pp. 92–97.
[39] M. Weber, F. Neri, and V. Tirronen, "Distributed differential evolution with explorative–exploitative population families," *Genet. Program. Evol. Mach.*, vol. 10, no. 4, pp. 343–371, 2009.
[40] M. Weber, F. Neri, and V. Tirronen, "Shuffle or update parallel differential evolution for large-scale optimization," *Soft Comput.*, vol. 15, no. 11, pp. 2089–2107, 2011.
[41] G. Wu, R. Mallipeddi, P. N. Suganthan, R. Wang, and H. Chen, "Differential evolution with multi-population based ensemble of mutation strategies," *Inf. Sci.*, vol. 329, pp. 329–345, Feb. 2016.
[42] A. Zamuda, J. Brest, and E. Mezura-Montes, "Structured population size reduction differential evolution with multiple mutation strategies on CEC 2013 real parameter optimization," in *Proc. IEEE Congr. Evol. Comput.*, Cancún, Mexico, 2013, pp. 1925–1931.
[43] F. Van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 225–239, Jun. 2004.
[44] M. N. Omidvar, X. Li, Z. Yang, and X. Yao, "Cooperative co-evolution for large scale optimization through more frequent random grouping," in *Proc. IEEE Congr. Evol. Comput.*, Barcelona, Spain, 2010, pp. 1–8.
[45] Z. Yang, J. Zhang, K. Tang, X. Yao, and A. C. Sanderson, "An adaptive coevolutionary differential evolution algorithm for large-scale optimization," in *Proc. IEEE Congr. Evol. Comput.*, Trondheim, Norway, 2009, pp. 102–109.
[46] M. N. Omidvar, X. Li, and X. Yao, "Smart use of computational resources based on contribution for cooperative co-evolutionary algorithms," in *Proc. Conf. Genet. Evol. Comput.*, Dublin, Ireland, 2011, pp. 1115–1122.
[47] H. K. Singh and T. Ray, "Divide and conquer in coevolution: A difficult balancing act," in *Agent-Based Evolutionary Search*. Berlin, Germany: Springer-Verlag, 2010, pp. 117–138.
[48] A. LaTorre, S. Muelas, and J.-M. Peña, "A comprehensive comparison of large scale global optimizers," *Inf. Sci.*, vol. 316, pp. 517–549, Sep. 2015.
[49] S. Mahdavi, M. E. Shiri, and S. Rahnamayan, "Metaheuristics in large-scale global continues optimization: A survey," *Inf. Sci.*, vol. 295, pp. 407–428, Feb. 2015.
[50] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, Oct. 2009.
[51] A. LaTorre, S. Muelas, and J.-M. Peña, "A MOS-based dynamic memetic differential evolution algorithm for continuous optimization: A scalability test," *Soft Comput.*, vol. 15, no. 11, pp. 2187–2199, 2011.
[52] K. N. Kozlov and A. M. Samsonov, "New migration scheme for parallel differential evolution," in *Proc. Int. Conf. Bioinformat. Genome Regul. Struct.*, 2006, pp. 141–144.
[53] J. Apolloni, G. Leguizamón, J. García-Nieto, and E. Alba, "Island based distributed differential evolution: An experimental study on hybrid testbeds," in *Proc. IEEE Int. Conf. Hybrid Intell. Syst.*, Barcelona, Spain, 2008, pp. 696–701.
[54] I. De Falco, A. D. Cioppa, D. Maisto, U. Scafuri, and E. Tarantino, "Biological invasion–inspired migration in distributed evolutionary algorithms," *Inf. Sci.*, vol. 207, pp. 50–65, Nov. 2012.
[55] D. Izzo, M. Ruciński, and C. Ampatzis, "Parallel global optimisation meta-heuristics using an asynchronous island-model," in *Proc. IEEE Congr. Evol. Comput.*, Trondheim, Norway, 2009, pp. 2301–2308.
[56] J. Cheng, G. Zhang, and F. Neri, "Enhancing distributed differential evolution with multicultural migration for global numerical optimization," *Inf. Sci.*, vol. 247, pp. 72–93, Oct. 2013.
[57] C. Zhang, J. Chen, and B. Xin, "Distributed memetic differential evolution with the synergy of Lamarckian and Baldwinian learning," *Appl. Soft Comput.*, vol. 13, no. 5, pp. 2947–2959, 2013.
[58] G. Jeyakumar and C. S. Velayutham, "Distributed heterogeneous mixing of differential and dynamic differential evolution variants for unconstrained global optimization," *Soft Comput.*, vol. 18, no. 10, pp. 1949–1965, 2014.
[59] S. Thangavelu and C. S. Velayutham, "An investigation on mixing heterogeneous differential evolution variants in a distributed framework," *Int. J. Bio Inspired Comput.*, vol. 7, no. 5, pp. 307–320, 2015.
[60] L. Singh and S. Kumar, "Parallel evolutionary asymmetric subsethood product fuzzy-neural inference system: An island model approach," in *Proc. Int. Conf. Comput. Theory Appl.*, Kolkata, India, 2007, pp. 282–286.
[61] J. Apolloni, J. García-Nieto, E. Alba, and G. Leguizamón, "Empirical evaluation of distributed differential evolution on standard benchmarks," *Appl. Math. Comput.*, vol. 236, pp. 351–366, Jun. 2014.
[62] D. R. Penas, J. R. Banga, P. González, and R. Doallo, "Enhanced parallel differential evolution algorithm for problems in computational systems biology," *Appl. Soft Comput.*, vol. 33, pp. 86–99, Aug. 2015.
[63] A. Zamuda and J. Brest, "Population reduction differential evolution with multiple mutation strategies in real world industry challenges," in *Swarm and Evolutionary Computation*. Berlin, Germany: Springer-Verlag, 2012, pp. 154–161.
[64] K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise, "Benchmark functions for the CEC'2010 special session and competition on large scale global optimization," Nat. Inspired Comput. Appl. Lab., Univ. Sci. Technol. China, Hefei, China, Tech. Rep., 2009. [Online]. Available: http://titan.csit.rmit.edu.au/~e46507/publications/lsgo-cec10.pdf
[65] J. Ronkkonen, S. Kukkonen, and K. V. Price, "Real-parameter optimization with differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, Edinburgh, U.K., 2005, pp. 506–513.
[66] 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 Evol. Comput.*, vol. 1, no. 1, pp. 3–18, 2011.
[67] D. Molina, M. Lozano, and F. Herrera, "MA-SW-Chains: Memetic algorithm based on local search chains for large scale continuous global optimization," in *Proc. IEEE Congr. Evol. Comput.*, Barcelona, Spain, 2010, pp. 1–8.
[68] R. Cheng and Y. Jin, "A social learning particle swarm optimization algorithm for scalable optimization," *Inf. Sci.*, vol. 291, pp. 43–60, Jan. 2015.
[69] S.-Z. Zhao, J. J. Liang, P. N. Suganthan, and M. F. Tasgetiren, "Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization," in *Proc. IEEE Congr. Evol. Comput.*, Hong Kong, 2008, pp. 3845–3852.
[70] X. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 16, no. 2, pp. 210–224, Apr. 2012.

Yong-Feng Ge (S'16) received the bachelor's degree in software engineering from Sun Yat-sen University, Guangzhou, China, in 2015, where he is currently pursuing the M.S. degree.

His current research interests include evolutionary computation algorithms and their applications on real-world problems, large-scale optimization algorithms, and distributed evolutionary algorithms and their applications on real-world problems.



Wei-Jie Yu (S'10–M'14) received the bachelor's and Ph.D. degrees from Sun Yat-sen University, Guangzhou, China, in 2009 and 2014, respectively.

He is currently a Lecturer with the School of Information Management, Sun Yat-sen University. His current research interests include computational intelligence and its applications on intelligent information processing, big data, and cloud computing.

Ying Lin (M'12) received the Ph.D. degree in computer applied technology from Sun Yat-sen University, Guangzhou, China, in 2012.

She is currently an Assistant Professor with the Department of Psychology, Sun Yat-sen University. Her current research interests include computational intelligence and its applications in network analysis and cognitive diagnosis.

Yue-Jiao Gong (M'15) received the B.S. and Ph.D. degrees in computer science from Sun Yat-sen University, Guangzhou, China, in 2010 and 2014, respectively.

From 2015 to 2016, she was a Post-Doctoral Research Fellow with the Department of Computer and Information Science, University of Macau, Macau, China. She is currently an Associate Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. Her current research interests include evolutionary computation and machine learning methods, as well as their applications to big data and intelligent transportation scheduling. She has published over 50 papers in the above areas.

Dr. Gong currently serves as a Reviewer for the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON NEURAL NETWORK AND LEARNING SYSTEMS, and the IEEE TRANSACTIONS ON CYBERNETICS.

Zhi-Hui Zhan (S'09–M'13) received the bachelor's and Ph.D. degrees from the Department of Computer Science, Sun Yat-sen University, Guangzhou, China, in 2007 and 2013, respectively.

He is currently a Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. His current research interests include evolutionary computation algorithms, swarm intelligence algorithms, and their applications in real-world problems, and in environments of cloud computing and big data.

Dr. Zhan was a recipient of the IEEE Computational Intelligence Society Outstanding Ph.D. Dissertation, the China Computer Federation Outstanding Ph.D. Dissertation, the Natural Science Foundation for Distinguished Young Scientists of Guangdong Province, China, in 2014, the Pearl River New Star in Science and Technology in 2015, and the Youth Talent in Science and Technology Innovation of Guangdong Province, China, in 2016. He is also appointed as the Pearl River Scholar Young Professor in 2016. He is listed as one of the Most Cited Chinese Researchers in Computer Science.

Wei-Neng Chen (S'07–M'12) received the bachelor's and Ph.D. degrees from Sun Yat-sen University, Guangzhou, China, in 2006 and 2012, respectively.

He is currently a Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. He has published over 70 papers in international journals and conferences. His current research interests include swarm intelligence algorithms and their applications on cloud computing, operations research, and software engineering.

Dr. Chen was a recipient of the IEEE Computational Intelligence Society Outstanding Dissertation Award for his doctoral thesis in 2016, and the National Science Fund for Excellent Young Scholars in 2016.

Jun Zhang (M'02–SM'08–F'16) received the Ph.D. degree in electrical engineering from the City University of Hong Kong, Hong Kong, in 2002.

He is currently a Professor with the South China University of Technology, Guangzhou, China. His current research interests include computational intelligence, cloud computing, wireless sensor networks, operations research, and power electronic circuits. He has authored seven research books and book chapters, and over 50 IEEE TRANSACTIONS papers in the above areas.

Prof. Zhang was a recipient of the National Science Fund for Distinguished Young Scholars in 2011, and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He was also appointed as the Changjiang Chair Professor in 2013. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, and the IEEE TRANSACTIONS ON CYBERNETICS. He is the Founding and Current Chair of the IEEE Guangzhou Subsection, the IEEE Beijing (Guangzhou) Section Computational Intelligence Society Chapters, and ACM Guangzhou Chapter.