Dynamic Group Learning Distributed Particle Swarm Optimization for Large-Scale Optimization and Its Application in Cloud Workflow Scheduling

Zi-Jia Wang, *Student Member, IEEE*, Zhi-Hui Zhan, *Senior Member, IEEE*, Wei-Jie Yu, *Member, IEEE*, Ying Lin, *Member, IEEE*, Jie Zhang, Tian-Long Gu, and Jun Zhang, *Fellow, IEEE*

***Abstract*—Cloud workflow scheduling is a significant topic in both commercial and industrial applications. However, the growing scale of workflow has made such a scheduling problem increasingly challenging. Many current algorithms often deal with small- or medium-scale problems (e.g., less than 1000 tasks) and face difficulties in providing satisfactory solutions when dealing with the large-scale problems, due to the curse of dimensionality. To this aim, this article proposes a dynamic group learning distributed particle swarm optimization (DGLDPSO) for large-scale optimization and extends it for the large-scale cloud workflow scheduling. DGLDPSO is efficient for large-scale optimization due to its following two advantages. First, the entire population is divided into many groups, and these groups are coevolved by using the master–slave multigroup distributed model, forming a distributed PSO (DPSO) to enhance the algorithm diversity. Second, a dynamic group learning (DGL) strategy is adopted for DPSO to balance diversity and convergence. When applied DGLDPSO into the large-scale cloud workflow scheduling, an adaptive renumber strategy (ARS) is further developed to make solutions relate to the resource characteristic and to make the searching behavior meaningful rather than aimless. Experiments are conducted on the large-scale benchmark functions set and the large-scale cloud workflow scheduling instances to further investigate the performance of DGLDPSO. The comparison results show that DGLDPSO is better than or at least comparable to other state-of-the-art large-scale optimization algorithms and workflow scheduling algorithms.**

***Index Terms*—Adaptive renumber strategy (ARS), dynamic group learning distributed particle swarm optimization (DGLDPSO), dynamic group learning strategy, large-scale cloud workflow scheduling, master–slave multigroup distributed.**

Manuscript received September 21, 2018; revised January 21, 2019 and June 3, 2019; accepted July 22, 2019. Date of publication September 17, 2019; date of current version May 7, 2020. This work was supported in part by the Outstanding Youth Science Foundation under Grant 61822602, in part by the National Natural Science Foundation of China under Grant 61772207 and Grant 61873097, in part by the Natural Science Foundation of Guangdong Province for Distinguished Young Scholars under Grant 2014A030306038, in part by the Guangdong Natural Science Foundation Research Team under Grant 2018B030312003, and in part by the Guangdong–Hong Kong Joint Innovation Platform under Grant 2018B050502006. This article was recommended by Associate Editor T. Ray. *(Corresponding authors: Zhi-Hui Zhan; Jun Zhang.)*

Z.-J. Wang is with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China.

Z.-H. Zhan is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, also with the State Key Laboratory of Subtropical Building Science, South China University of Technology, Guangzhou 510006, China, and also with the Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information, Guangzhou 510006, China (e-mail: zhanapollo@163.com).

W.-J. Yu is with the School of Information Management, Sun Yat-sen University, Guangzhou 510006, China.

Y. Lin is with the Department of Psychology, Sun Yat-sen University, Guangzhou 510006, China.

J. Zhang is with the School of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China.

T.-L. Gu is with the School of Computer Science and Engineering, Guilin University of Electronic Technology, Guilin 541004, China.

J. Zhang is with Victoria University, Melbourne, VIC 8001, Australia (e-mail: junzhang@ieee.org).

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

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

Digital Object Identifier 10.1109/TCYB.2019.2933499

I. Introduction

WORKFLOW, which contains a set of tasks interconnected via data or computing dependence between each other, is widely used and applicated in many real-world applications [1]. For example, Montage workflow can be used to generate custom mosaics of the sky and CyberShake workflow can be used to characterize earthquake hazards in a region [2]. The workflow scheduling problem is to find the most suitable resource to execute each task of the workflow, so as to satisfy users' quality of service (QoS).

In the past, researchers often studied the workflow scheduling based on distributed environment like grids [3], [4]. With the popularity of cloud computing [5]–[8], workflow scheduling on cloud resource has gradually become a significant research topic in recent years [9]–[11], but it is also more challenging. Different from the fixed and limited computing resources in grid computing, the computing resources in cloud computing are elastic and almost unlimited, which can be leased in any amount at any time and according to the pay-per-use pricing principle. Leasing more resources or expensive resources can shorten the executing time, but also requires a larger cost.

Therefore, taking both the time and cost into consideration is necessary in workflow scheduling on cloud. Rodriguez and Buyya [12] proposed a cost-minimization and deadline-constrained workflow scheduling (CMDCWS) model, where the tasks are required to execute with a minimum cost and within a given deadline constraint. This has become a popular scheduling model because it can find a potential balance between time and cost, with a number of following works on this model [12]–[17]. For example, due to the success of evolutionary algorithms (EAs) [18]–[26], some

EAs have been proposed to deal with the CMDCWS model, such as coevolutionary genetic algorithm (GA) [13], dynamic objective GA (DOGA) [14], cost effective GA [15], particle swarm optimization (PSO) [12], renumber PSO (RNPSO) [16], ant colony system (ACS) [17], and multiobjective ACS (MOACS) [1]. Thus, we also adopt this CMDCWS model in this article.

Even though the above approaches are competitive in solving the small- or medium-scale CMDCWS (e.g., less than 1000 tasks), when the scale of workflow increases, several challenges occur, such as the huge search space and exponentially increasing number of local optima.

To deal with these challenges, this article proposes a dynamic group learning distributed PSO (DGLDPSO) for large-scale optimization and extends it for solving the large-scale CMDCWS. Specifically, three major novel designs and advantages that help DGLDPSO balance diversity and convergence for finding the feasible solution under a tight deadline and minimizing the cost in the large-scale cloud workflow scheduling are described as follows.

1) During the evolutionary process, we randomly divide the entire population into several equal groups in every generation, and these groups are coevolved by using the master–slave multigroup distributed model, forming a distributed PSO (DPSO) framework. This is promising to enhance the population diversity to efficiently deal with the complex search challenge of large-scale optimization.
2) In the particle's evolution, a dynamic group learning (DGL) strategy is adopted for DPSO. On the one hand, the size of each group is dynamically changed in every generation. On the other hand, each group is treated as a big "particle." In this sense, the best particle in the group is regarded as the personal best ***pbest*** of the big particle, while the best ***pbest*** of all big particles (groups) is regarded as the globally best ***gbest*** of the entire population. Besides, only the worst particle in each group is updated by learning from the ***pbest*** of the current group and the ***gbest*** of the entire population. Therefore, in the DGL strategy, the dynamically changed group size can control the learning strength and find a potential balance between diversity and convergence for large-scale optimization. Moreover, only updating the worst particle in the group can save more fitness evaluations (FEs) budget for other particles to search more regions of large-scale space and to further refine the solution accuracy.
3) In the traditional cloud workflow scheduling algorithms, each dimension of the solution represents a task, while the value of each dimension stands for the index of cloud resource that is scheduled to execute the corresponding task. However, the index of resource in fact is only a meaningless index number which does not reflect the characteristic of resource. Learning from the index of resource might make particles sightless. That is when the value x^d on dimension d in a particle moves toward pbest^d or gbest^d, it seems that the resource pbest^d or gbest^d is more suitable for task t_d. However, the resource number between x^d and pbest^d or gbest^d does not represent anything. Therefore, in the particle's learning, an adaptive renumber strategy (ARS) is proposed to make the index of resource meaningful. We adaptively use two metrics that are related to execution time and execution cost, respectively, to sort and renumber the resources in order to make the learning among particles more clear and reasonable.

Therefore, the contributions of this article are two-fold, from both the algorithm design aspect and the real-world application aspect. One the one hand, in the algorithm design aspect, we propose a novel DPSO with DGL strategy, called DGLDPSO, to efficiently solve the large-scale optimization problems. This is an algorithm innovation in both PSO and large-scale optimization. On the other hand, in the real-world application aspect, we have enhanced the DGLDPSO with ARS and extended the algorithm to efficiently solve the large-scale cloud workflow scheduling problems. This is a significant contribution to the cloud computing field.

Experiments are conducted on a large-scale benchmark function set from the CEC2010 test suite and the large-scale instances of cloud workflow scheduling (e.g., up to 5000 tasks). The results obtained by DGLDPSO are better than, or at least comparable to those obtained by the state-of-the-art large-scale optimization algorithms and the cloud workflow scheduling algorithms, showing the effectiveness of our DGLDPSO algorithm.

The remainder of this article is organized as follows. Section II describes the basic PSO algorithm, the application of PSO in cloud workflow scheduling, and the CMDCWS model. Section III describes the proposed DGLDPSO algorithm in detail. In Section IV, experimental results are presented and discussed. Finally, Section V draws the conclusions.

II. PSO and Its Application in Scheduling

A. PSO Framework

In PSO [27], the member of the swarm is called particle, which means a possible solution in the search space. Each particle P_i is associated with two vectors. The vector $\boldsymbol{V}_i = [v_i^1, v_i^2, \ldots, v_i^D]$ means the velocity of P_i, while the vector $\boldsymbol{X}_i = [x_i^1, x_i^2, \ldots, x_i^D]$ means the position of P_i, where D stands for the dimensions of the search space. Moreover, each particle P_i has a memory on its historical best position, called personal best $\boldsymbol{pbest}_i = [pbest_i^1, pbest_i^2, \ldots, pbest_i^D]$. The best one of all the $\boldsymbol{pbest}_i$ is treated as the globally best of the entire population, called $\boldsymbol{gbest} = [gbest^1, gbest^2, \ldots, gbest^D]$. In every generation, each particle P_i adjusts its velocity and position based on its own $\boldsymbol{pbest}_i$ and the $\boldsymbol{gbest}$ of the entire population. The velocity $\boldsymbol{V}_i$ and position $\boldsymbol{X}_i$ of each particle are updated according to the following formulas:

$$v_i^d = \omega \times v_i^d + c_1 \times r1_i^d \times \left(pbest_i^d - x_i^d\right) + c_2 \times r2_i^d \times \left(gbest^d - x_i^d\right) \quad (1)$$

$$x_i^d = x_i^d + v_i^d \quad (2)$$

where ω is the inertia weight to balance global and local search performance. c_1 and c_2 are the acceleration coefficients, where parameter c_1 pulls the particle to its own ***pbest***, ensuring the diversity of the population; while c_2 pushes the swarm to converge to the current ***gbest***, ensuring the speed of convergence. $r1_i^d$ and $r2_i^d$ are the two uniformly distributed random numbers within [0, 1]. A particle's velocity and position on each dimension are clamped in $[-V_{\max}, V_{\max}]$ and $[X_{\min}, X_{\max}]$, respectively.

B. Application of PSO in Cloud Workflow Scheduling

Many improved PSO variants were proposed to solve the tasks scheduling problem in cloud computing, which aims to find the most suitable resource for each task to meet users' QoS, where the tasks are uncorrelated with each other [28]–[35]. For example, Alkhashai and Omara [29] merged the best-fit algorithm into PSO for generating the initial population and utilized tabu search algorithm to do the local search. Zhao *et al.* [34] proposed an adaptive inertia weight PSO (AIWPSO) to relieve the sensitivity of inertia weight and make PSO more adaptive and effective to obtain better scheduling. Apart from the adaptive inertia weight, an adaptation of acceleration coefficients was also used in the mutation time-varying PSO (MTV_PSO) [35] for task scheduling.

Even though these task scheduling algorithms have achieved a great success in cloud computing, the workflow scheduling problems are much more challenging because the tasks are dependent on each other and require a specific execution order.

Over the past few years, many PSO-based cloud workflow scheduling algorithms have been extensively researched [36]–[40]. Some scheduling algorithms focus on reducing the financial cost. For example, Wu *et al.* [37] proposed a revised discrete PSO (RDPSO) to schedule applications among cloud services that takes both data transmission cost and computation cost into account. Pandey *et al.* [39] proposed a novel PSO-based approach to reduce the cost, it shows the PSO-based approach can achieve almost three times cost saving as the compared algorithms. Some scheduling algorithms focus on minimizing the execution time. For instance, Manasrah and Ali [40] proposed a hybrid scheduling algorithm (GA-PSO) which combines PSO and GA to reduce the workflow execution time.

However, cost and time often conflict with each other, the scheduling which can shorten the executing time may also require a larger cost, while a low investment often greatly prolongs the executing time. Therefore, taking both the time and cost into consideration is necessary in workflow scheduling on cloud.

C. CMDCWS Model

To take both the time and cost into consideration, the CMDCWS model is first proposed by Rodriguez and Buyya [12]. It focuses on finding a schedule to execute a workflow so that the total execution cost (TEC) is minimized and the total execution time (TET) should not exceed the deadline. These are depicted in

$$\text{Minimize TEC} \tag{3}$$

$$\text{TET} \leq \text{deadline}. \tag{4}$$

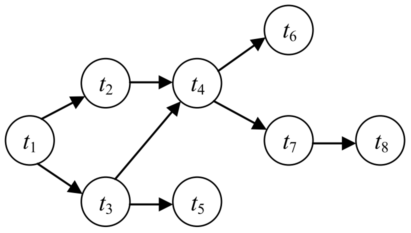


Fig. 1. Example of workflow.

A simple workflow example is shown in Fig. 1. Besides, there is a set of available resources r_j, where each resource has its own specific processing capacity PC_j and cost per unit time C_j. Often, the resource with higher processing capacity can deal with tasks faster, but is relatively more expensive.

In the CMDCWS model, a schedule for each task can be formulated as $S_i = [t_i, r_{x[i]}, \text{ST}_i, \text{ET}_i]$, which means the task t_i is executed on resource $r_{x[i]}$, and is started at the start time ST_i and ended at the end time ET_i. Besides, the resource $r_{x[i]}$ has two important parameters called the lease start time $\text{LST}_{x[i]}$ and lease end time $\text{LET}_{x[i]}$. The equations of calculating TEC and TET are shown as

$$\text{TEC} = \sum_{i=1}^{|T|} C_{x[i]} \times \left(\text{LET}_{x[i]} - \text{LST}_{x[i]}\right) \tag{5}$$

$$\text{TET} = \max\{\text{ET}_i\},\ 1 \leq i \leq |T|. \tag{6}$$

III. DGLDPSO for the Large-Scale Cloud Workflow Scheduling

In order to deal with the large-scale cloud workflow scheduling problems efficiently, DGLDPSO is proposed, together with the following three novel designs and advantages. First, DGLDPSO uses a master–slave multigroup distributed model, forming a DPSO, where multiple groups are coevolved concurrently to enhance the population diversity. Second, the DGL strategy is proposed for DPSO. On the one hand, the size of group is dynamically changed, so as to control the learning strength and find a potential balance between diversity and convergence for large-scale optimization. On the other hand, we treat each group as a big particle and only one update is performed in each group (big particle), which can save FEs to search more regions of the large-scale space and to further refine the solution accuracy. Third, considering the characteristics of cloud workflow scheduling, ARS is further proposed, which can adaptively choose two metrics to sort and renumber the resources in order to make the learning among particles more clear and reasonable.

Before we introduce DGLDPSO, we first describe three critical issues when using PSO for solving the cloud workflow scheduling problem.

The first one is the encoding. Here, the dimension of particle is equal to the number of tasks in the workflow, while the

Particle's position

d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8
2.2	3.5	2.4	4.4	2.7	1.4	3.1	3.8

The corresponding workflow scheduling

$r_2 \rightarrow t_1$	$r_3 \rightarrow t_2$	$r_2 \rightarrow t_3$	$r_4 \rightarrow t_4$	$r_2 \rightarrow t_5$	$r_1 \rightarrow t_6$	$r_3 \rightarrow t_7$	$r_3 \rightarrow t_8$

Fig. 2. Example of the encoding of a particle's position.

$$exetime = \begin{array}{c|cccc} & r_1 & r_2 & r_3 & r_4 \\ \hline t_1 & 2 & 1 & 4 & 3 \\ t_2 & 4 & 6 & 2 & 5 \\ t_3 & 2 & 1 & 1 & 4 \\ t_4 & 5 & 2 & 3 & 1 \\ t_5 & 4 & 4 & 2 & 2 \\ t_6 & 1 & 3 & 2 & 4 \\ t_7 & 2 & 3 & 1 & 5 \\ t_8 & 4 & 3 & 1 & 2 \end{array} \quad transfertime = \begin{array}{c|cccccccc} & t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 \\ \hline t_1 & 0 & 0.3 & 0.2 & 0 & 0 & 0 & 0 & 0 \\ t_2 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 \\ t_3 & 0 & 0 & 0 & 0.4 & 0.3 & 0 & 0 & 0 \\ t_4 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0.2 & 0 \\ t_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ t_6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ t_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 \\ t_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Fig. 3. Example of *exetime* and *transfertime*.

search range of each dimension in particle is the real number between 1.0 and the number of resources plus one. For example, assume the workflow is as Fig. 1 with eight tasks and there are four resources can be leased, then each particle is 8-D and the range of each dimension is [1.0, 5.0). In other words, each dimension stands for each task, and the integer part of its value represents the index of resource scheduled to execute the corresponding task. Fig. 2 shows a simple scheduling, where d_1 stands for task t_1, and its value 2.2 represents that resource r_2 will be assigned to task t_1; d_2 stands for task t_2, and its value 3.5 indicates that resource r_3 will be allocated to task t_2, and so on.

The second one is the FE. As mentioned above, we use (5) and (6) to calculate TEC and TET, respectively. However, in order to obtain TEC and TET, the time that each task is executed on each resource needs to be known. The execution time is stored in the matrix *exetime*, where the *exetime*$[i][j]$ represents the time that task t_i runs on resource r_j. Besides, when the parent task and its child task are executed on different resources, the parent needs time to transfer data to its child. As a result, the transfer time stored in the matrix *transfertime* is used, where the *transfertime*$[i][j]$ indicates the time that the task t_i transfers data to t_j. Fig. 3 presents an example of the matrices *exetime* and *transfertime*. After getting the *exetime* and *transfertime*, the TEC and TET values of a solution can be calculated as the pseudocode shown in Algorithm 1.

The last one is the fitness comparison. As we wish to minimize TEC and let TET not exceed the deadline, so if both solutions are feasible (TET does not exceed the deadline), the one with smaller TEC is better. While if only one solution is feasible, obviously, the feasible solution is preferred. However, if both solutions are infeasible (TET exceeds the deadline), the one with smaller constraint violation (smaller TET) is selected.

A simple workflow scheduling corresponding to Figs. 1–3 is shown in Fig. 4.

A. Master–Slave Multigroup Distributed Framework

The master–slave multigroup distributed framework is illustrated in Fig. 5, where the master node dominates the multiple slave nodes in parallel hardware.

Algorithm 1 Calculate TEC and TET

```
Begin
1.  $TEC = 0$; $TET = 0$;
2.  Initialize each $ST_i$, $ET_i$, $LST_j$, $LET_j$ as 0;
3.  For each task $t_i$
4.      If $t_i$ has no parents
5.          $ST_i = LET_{x[i]}$;
6.      Else
7.          $ST_i = \max\{\max\{ET_p\}, LET_{x[i]}\}$; /* $p$ indicates each parent of $i$ */
8.      End If
9.      $transfer = 0$;
10.     For each $t_c$ /* $c$ indicates each child of $i$ */
11.         If $x[c] \neq x[i]$
12.             $transfer = transfer + transfertime[i][c]$;
13.         End If
14.     End For
15.     $ET_i = ST_i + exetime[i][x[i]] + transfer$;
16.     $LST_{x[i]} = ST_i$;
17.     $LET_{x[i]} = ET_i$;
18.     $TEC = TEC + C_{x[i]} \times (LET_{x[i]} - LST_{x[i]})$;
19. End For
20. $TET = max\{ET_i\}$;
End
```

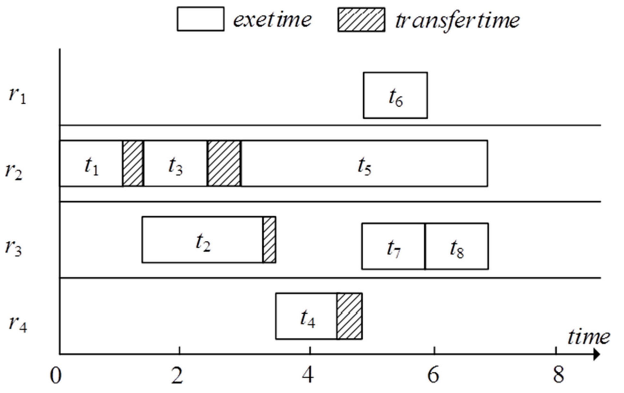


Fig. 4. Example of the workflow scheduling.

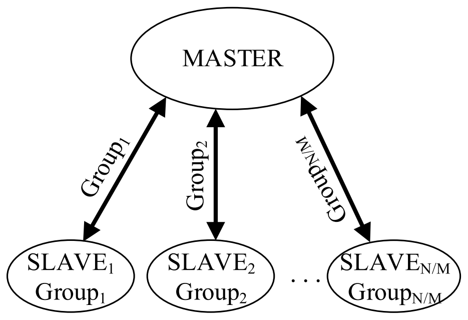


Fig. 5. Master–slave multigroup distributed model.

During the evolutionary process, the master always randomly divides the entire population into N/M equal groups, where N is the population size, and M is the size of the group which will be dynamically changed. Note that if $N\%M \neq 0$, the last group will have $M + N\%M$ particles. After the population partition, the master will send each group to its corresponding slave, and different groups are coevolved concurrently on their slave nodes. After the evolution, each slave sends the updated group back to the master. Up to now, we have finished a loop sequent. The process will be repeated until the termination criterion is satisfied.

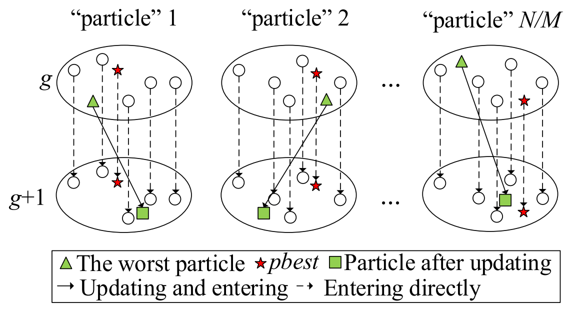


Fig. 6. Learning strategy in DGLDPSO.

As we can see, this master–slave multigroup distributed model can enhance the population diversity by coevolution, which matches the search requirement of the large-scale optimization problems.

B. DGL Strategy

The DGL strategy has the following two novel characteristics.

First, after the population partition, each group is regarded as a big particle, and the best particle in the group is denoted as the ***pbest*** of the big particle (group). Besides, only the worst particle $\boldsymbol{x}_w$ in the group will update its velocity and position by learning from the ***pbest*** in the current big particle (group) and from the ***gbest*** of the entire population, while other particles in the group will enter to the next generation directly. In other words, only N/M particles will be updated in the entire population.

Second, the group size M is dynamically changed in every generation, which is randomly selected from a fixed interval. That is because for a given population size, if the group size is large, the ***pbest*** is selected from a large group of particles, which would be relatively greedy. On the contrary, if the group size is small, ***pbest*** is selected from a small group of particles, which becomes relatively diverse. However, without any prior knowledge about the landscape of a problem, it is hard to precisely determine a proper group size M in different evolutionary stages. Therefore, to make a compromise, the M is randomly selected from a fixed interval. Although this scheme is simple, it is effective and readily applicable to a wide range of uses. For example, if the current group gets trapped in the local optima, the group size M has the opportunity to be smaller to improve the population diversity. In contrast, if the global optima region is found, the group size M can become larger to accelerate the convergence. As a result, we can achieve a potential balance between population diversity and fast convergence. Meanwhile, the random selection also relieves the sensitivity of parameter.

Fig. 6 illustrates the main idea of this novel DGL strategy. The formula of updating velocity of the worst particle is shown as (7), which is similar to (1), but with the following differences:

$$v^d = \omega^d \times v^d + c_1 \times r1^d \times \left(pbest^d - x^d\right) + c_2 \times r2^d \times \left(gbest^d - x^d\right). \quad (7)$$

1) The first part is similar to the inertia term in the traditional PSO. The only difference is that the inertia weight ω in PSO is often set as linearly decreasing from 0.9 to 0.4, while herein it is replaced by the random number in DGLDPSO. The random number is helpful to improve the learning diversity and the population diversity.
2) The second part is also learning from the ***pbest***, which is same to the traditional PSO. However, the ***pbest*** in DGLDPSO is regarded as the best particle of the big particle (group), as we mentioned above.
3) Parameters c_1 and c_2 are often set as 2.0 in the traditional PSO. But in DGLDPSO, c_1 is set as 1.0 and c_2 is set as 0.1, respectively. That is because when dealing with the large-scale optimization problems, we should concentrate more on diversity maintaining to avoid local trapped. In this sense, c_1 and c_2 should be smaller to ensure the population diversity, while c_2 which controls the convergence speed to the ***gbest*** should be much smaller.

Therefore, there are several superiorities of this DGL strategy shown as follows.
1) The random selection of M and redefinition of the big particle and its ***pbest*** can control the learning strength effectively and can find a potential balance between diversity and convergence, which can further improve the performance of DGLDPSO.
2) Only the worst particle in each group is updated, which is helpful to save more FEs for other particles.
3) Resetting the parameters c_1 and c_2 will make DGLDPSO focus more on population diversity, which matches the search requirement of the large-scale optimization problems.

With the above descriptions, the pseudocode of DGLDPSO can be summarized in Algorithm 2.

C. ARS

DGLDPSO performs particularly well on many large-scale optimization problems, which will be further discussed in Section IV-B. However, when applied to the large-scale cloud workflow scheduling, an extra strategy called ARS is incorporated in DGLDPSO due to the characteristics of the cloud workflow scheduling.

Consider a situation. When the cloud provider offers a large amount of resources with random number, the particles will become blind in the traditional PSO during the learning process. That is when the value x^d on dimension d in a particle moves toward pbest^d or gbest^d, it seems that the resource pbest^d or gbest^d is more suitable for task t_d. However, the resource number between x^d and pbest^d or gbest^d does not represent anything. How to make the searching process meaningful is quite important, especially when the scale of workflow is large.

The renumber strategy is first proposed by Li *et al.* [16]. It used the cost per unit time as the metric to renumber the resources where the resource r_i represents the resource with the ith lowest cost per unit time. In that way, when x^d flies toward pbest^d or gbest^d, it predicts a tendency that task t_d

is suitable for a cheaper resource or an expensive one. As a consequence, the searching process will become meaningful.

Even though it achieved the relatively promising results, there is still much room for improvement. As there are two objectives (i.e., time and cost) when evaluate the scheduling, only using the cost metric to renumber the resources is not wise, especially when the deadline is tight. As a result, in this article, we proposed a novel ARS by adaptively selecting the metric to further improve the learning process.

At the beginning, DGLDPSO uses the metric of processing capacity to renumber the resources. This metric is used because that until the population finds a feasible solution, TET is the primary optimization objective we should concern. The metric of processing capacity is more related to the TET, which can help the algorithm find solutions that satisfy the deadline constraint. Once the population finds a feasible solution, what we concerned turns to the cost. At this moment, DGLDPSO adaptively turns to use the metric of cost per unit time to renumber the resources. As a result, DGLDPSO will predict a tendency for each task that the current task is suitable for a cheaper resource or an expensive one, which is useful to minimize the TEC. There are two advantages of our ARS to solve the large-scale cloud workflow scheduling.

1) The renumber strategy can predict a searching tendency for each task, which can make the searching and learning process more meaningful.
2) Adaptively selecting the two metrics can make the population find the feasible solution and minimize the cost more quickly, which is more suitable for the large-scale cloud workflow scheduling, especially when the deadline is tight.

D. Complete Algorithm DGLDPSO

Based on all the components described above, the pseudocode of the complete procedure of DGLDPSO for the large-scale cloud workflow scheduling is outlined in Algorithm 3. The superiority of DGLDPSO is shown as follows.

1) Several groups are coevolved by using the master–slave multigroup distributed model, forming a DPSO, which can enhance the population diversity.
2) The DGL strategy is proposed for DPSO, which can control the learning strength effectively and can find a potential balance between diversity and convergence.
3) ARS is proposed to make the index of the resource and the searching process meaningful, which matches the searching requirements and characteristics of cloud workflow scheduling.

E. Complexity Analysis

Herein, we denote the population size, the dimension of problem, the group size, and the maximum number of FEs as N, D, M, and MaxFEs, respectively. First, in the initialization, the time complexity of DGLDPSO is $O(N \times D) + O(N)$, which is obtained by steps 1–3 in the MASTER process of Algorithm 2. Then, as for individual evolution, since DGLDPSO only updates the worst particle in each group, as a result, the time complexity is $O(N/M \times D) + O(N/M)$ in every generation, as obtained by steps 8–11 in the MASTER process and steps 1–6 in the SLAVE process of Algorithm 2. DGLDPSO will evolve $\text{MaxFEs}/(N/M)$ generations, so the overall time complexity of DGLDPSO is $\text{MaxFEs}/(N/M) \times (O(N/M \times D) + O(N/M))$, which is actually reduced to the $O(\text{MaxFEs} \times D)$ and is similar to many other state-of-the-art large-scale optimization algorithms. Detailed comparisons of time complexity of different large-scale optimization algorithms are listed in Table S.I in the supplementary material.

When apply DGLDPSO to the large-scale cloud workflow scheduling, ARS is proposed and we add two sorting operators in DGLDPSO, as shown in steps 1 and 8 in the MASTER process of Algorithm 3, respectively. Denote the number of resources as N_r, the time complexity of ARS in DGLDPSO is $O(N_r \times \log(N_r))$. Therefore, the overall time complexity of DGLDPSO for the large-scale cloud workflow scheduling is $O(\text{MaxFEs} \times D) + O(N_r \times \log(N_r))$. Detailed comparisons of time complexity of different cloud workflow scheduling algorithms are listed in Table S.II in the supplementary material.

IV. Experimental Results

A. Experimental Setup

To test the performance of DGLDPSO, we conduct the following two experiments. The first one is on the 20 widely used 1000-D optimization benchmark functions from CEC2010 test suite [41], which is used to illustrate the superiority of DGLDPSO for dealing with the large-scale optimization problems. The second one is on the large-scale extension of the CMDCWS model proposed in [12], which is also used to show the preponderance of DGLDPSO for solving the large-scale cloud workflow scheduling. For more details about the test functions and the cloud workflow scheduling model, please refer to [12] and [41], respectively.

For the implementation of DGLDPSO, the master–slave model of DGLDPSO is built in a multiprocessor distributed environment that consists of a number of distributed computing servers. The CPU of each server has eight processors configured with Intel Core i5-7400, 3.00 GHz. Therefore, we obtain the multiprocessor distributed environment and we can assign each group to one processor through MPI. The population size is set as 500 and the interval for the group size M is [2, 10].

In the first experiments on the large-scale benchmark functions, we compare DGLDPSO with seven state-of-the-art large-scale optimization algorithms. The first four competitors are based on the cooperative coevolution (CC) framework, that is, multilevel CC (MLCC) [42], cooperative coevolving PSO (CCPSO2) [43], and cooperative coevolving DE (DECC), including DECC with random grouping (DECC-G) [44] and DECC with differential grouping (DECC-DG) [45]. The other four competitors are non-CC-based large-scale optimization algorithms, including competitive swarm optimizer (CSO) [46], social learning PSO (SL-PSO) [47], dynamic multiswarm PSO (DMS-L-PSO) [48], and dynamic level-based learning swarm optimizer (DLLSO) [49].

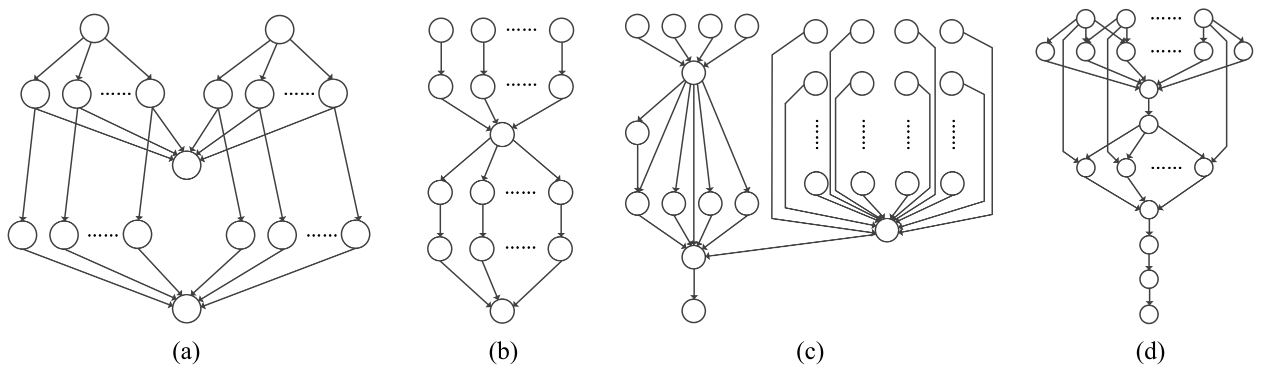


Fig. 7. Topology structures of four different scientific workflows: (a) CyberShake, (b) LIGO, (c) SIPHT, and (d) Montage.

Although DLLSO also has the dynamic technique, it is different that DLLSO dynamic selects a number from a level pool (contain only a few choices) as the number of levels, while DGLDPSO dynamic selects a number from an interval (contain a large number of choices) as the group size. In the second experiments on the large-scale cloud workflow scheduling, we compare DGLDPSO with seven typical cloud scheduling algorithms, including five PSO-based scheduling algorithms, such as PSO [12], RNPSO [16], AIWPSO [34], MTV_PSO [35], and RDPSO [37], and two GA-based scheduling algorithms, such as DOGA [14] and GA-PSO [40]. The parameters of these competitors are set according to their original proposals because they have been well turned for these related problems.

For fair comparisons, the MaxFEs is set as 3 000 000 for all competitors. Moreover, all the experiments run 30 times independently for statistics and the mean results are reported. In addition, the coefficient of variance (C.V) of the 30 runs is calculated to show the stability of the algorithm. Moreover, the Wilcoxon's rank-sum test at $\alpha = 0.05$ between DGLDPSO and other state-of-the-art algorithms is performed to evaluate the statistical significance of the results. The symbols "+," "≈," and "−" indicate DGLDPSO performs significantly better (+), similarly (≈), or significantly worse (−) than the corresponding algorithm in comparison.

B. Comparison Results on Large-Scale Benchmark Functions

These functions are with 1000 dimensions and can be classified into three groups. The first group includes three separable functions f_1–f_3. The second group consists of the following 15 functions f_4–f_{18}, which are partially separable functions. The last group consists of the last two functions f_{19}–f_{20} that are nonseparable functions. All these functions are shifted and rotated, which are more difficult to solve and make our test more comprehensive and convincing. Due to the space limitation, the properties of these functions are given in Table S.III in the supplementary material. For more details about these test functions, please refer to [41].

Table I presents the comparison results where the best results are highlighted in boldface. From Table I, we can see the following.

For the first three separable functions f_1–f_3, DGLDPSO performs significantly better than most of other algorithms, especially on f_1 and f_3. Although it performs slightly worse than MLCC and DLLSO on these three functions, MLCC and DLLSO lose their feasibilities when dealing with the partially separable or nonseparable functions, which will be discussed below.

For the next 15 partially separable functions f_4–f_{18}, DGLDPSO performs the best on five functions (f_6, f_7, f_{11}, f_{13}, and f_{16}). It dominates MLCC, CCPSO2, DECC-G, DECC-DG, CSO, and DMS-L-PSO on at least 11 functions. When compared with SL-PSO and DLLSO, DGLDPSO also significantly performs better than SL-PSO and DLLSO on 8 and 9 functions, respectively, while only dominated by SL-PSO and DLLSO on 6 and 5 functions, respectively.

For the last two nonseparable functions f_{19}–f_{20}, although DGLDPSO performs slightly worse than some algorithms on f_{19}, it almost dominates and performs better than all the compared algorithms on f_{20}.

Overall, DGLDPSO performs better than MLCC, CCPSO2, DECC-G, DECC-DG, CSO, SL-PSO, DMS-L-PSO, and DLLSO on 14, 14, 15, 15, 15, 12, 19, and 11 functions, respectively. Conversely, MLCC, CCPSO2, DECC-G, DECC-DG, CSO, SL-PSO, DMS-L-PSO, and DLLSO can only surpass DGLDPSO on 6, 4, 5, 5, 2, 6, 1, and 8 functions, respectively. Moreover, the C.V values of DGLDPSO are generally smaller than other state-of-the-art compared algorithms, indicating that DGLDPSO always has more stable performance than the competitors.

Therefore, DGLDPSO generally performs better than all these competitors on most of the tested large-scale benchmark functions. The promising performance of DGLDPSO also further confirms that the master–slave multigroup distributed model in coevolution helps the algorithm enhance the population diversity to sufficiently search in the very large space for finding the global optimum.

To further study the evolutionary behavior of different algorithms, we draw their convergence curves to observe their evolutionary processes. Besides, in order to make our comparison more convincing, we choose several typical benchmark functions from all the three groups. Herein, we select separable function f_3, partially separable functions f_6, f_{11}, f_{13}, and f_{16}, and nonseparable function f_{20} as the representative instances. The convergence curves of DGLDPSO, MLCC, CCPSO2, DECC-G, DECC-DG, CSO, SL-PSO, DMS-L-PSO,

TABLE I
EXPERIMENTAL RESULTS OF 1000-D CEC2010 FUNCTIONS f_1–f_{20}

fun	DGLDPSO		MLCC		CCPSO2		DECC-G		DECC-DG	
	Mean±Std	C.V	Mean±Std	C.V	Mean±Std	C.V	Mean±Std	C.V	Mean±Std	C.V
f_1	4.55E-21±2.61E-22	5.74%	**0.00E+00±0.00E+00(-)**	N/A	1.40E+00±1.75E+00(+)	125%	1.19E-14±1.10E-14(+)	92.4%	1.35E+01±6.50E+01(+)	481%
f_2	7.35E+02±4.52E+01	6.15%	**1.66E-01±3.77E-01(-)**	227%	4.21E+00±1.22E+00(-)	29.0%	4.16E+01±2.22E+01(-)	53.4%	4.39E+03±2.02E+02(+)	4.60%
f_3	2.33E-13±1.55E-14	6.65%	8.17E-02±3.11E-01(+)	381%	4.61E-03±1.53E-03(+)	33.2%	1.77E+00±3.86E-01(+)	21.8%	1.66E+01±3.32E-01(+)	2.00%
f_4	3.26E+11±5.98E+10	18.3%	9.15E+12±2.93E+12(+)	32.0%	2.04E+12±1.19E+12(+)	58.3%	1.09E+13±3.52E+12(+)	32.3%	5.04E+12±1.44E+12(+)	28.6%
f_5	2.83E+07±1.06E+06	3.75%	5.30E+08±1.14E+08(+)	21.5%	4.45E+08±1.34E+08(+)	30.1%	2.90E+08±1.05E+08(+)	36.2%	1.44E+08±1.85E+07(+)	12.8%
f_6	**4.14E-09±1.39E-10**	3.36%	1.94E+07±1.21E+06(+)	6.24%	1.80E+07±4.40E+06(+)	24.4%	5.30E+06±1.42E+06(+)	26.8%	1.62E+01±4.07E-01(+)	2.51%
f_7	**2.43E+01±1.00E+01**	41.2%	9.97E+05±7.57E+05(+)	75.9%	3.82E+08±6.95E+08(+)	182%	5.61E+06±1.13E+07(+)	201%	1.35E+04±1.38E+04(+)	102%
f_8	2.84E+07±1.99E+05	0.70%	5.74E+07±3.32E+07(+)	57.8%	2.74E+07±2.79E+07(≈)	102%	6.50E+07±3.40E+07(+)	52.3%	1.61E+07±1.23E+07(-)	76.4%
f_9	4.47E+07±3.44E+06	7.70%	1.24E+08±1.44E+07(+)	11.6%	9.40E+07±3.41E+07(+)	36.3%	2.35E+08±2.40E+07(+)	10.2%	5.56E+07±6.06E+06(+)	10.9%
f_{10}	2.05E+03±1.26E+02	6.15%	4.48E+03±1.51E+03(+)	33.7%	4.87E+03±7.19E+02(+)	14.8%	9.42E+03±5.22E+02(+)	5.54%	4.48E+03±1.22E+02(+)	2.72%
f_{11}	**3.28E-13±1.23E-14**	3.75%	1.93E+02±2.66E+01(+)	13.8%	1.98E+02±3.73E-01(+)	0.19%	2.55E+01±1.38E+00(+)	5.41%	1.06E+01±1.05E+00(+)	9.91%
f_{12}	6.25E+04±3.55E+03	5.68%	3.56E+04±5.21E+03(-)	14.6%	3.72E+04±1.29E+04(-)	34.7%	4.08E+04±5.32E+03(-)	13.0%	**2.70E+03±8.88E+02(-)**	32.9%
f_{13}	**5.70E+02±1.83E+02**	32.1%	2.65E+03±1.91E+03(+)	72.1%	1.30E+03±1.53E+02(+)	11.8%	4.21E+03±3.17E+03(+)	75.3%	5.54E+03±3.56E+03(+)	64.3%
f_{14}	1.46E+08±8.59E+06	5.88%	3.16E+08±2.52E+07(+)	7.97%	2.37E+08±1.03E+08(+)	43.5%	5.85E+08±4.25E+07(+)	7.26%	3.37E+08±2.27E+07(+)	6.74%
f_{15}	1.05E+04±9.96E+01	0.95%	9.44E+03±2.05E+03(-)	21.7%	1.08E+04±1.34E+03(≈)	12.4%	8.23E+03±3.11E+03(-)	37.8%	5.85E+03±7.49E+01(-)	1.28%
f_{16}	**3.58E-13±1.14E-14**	3.18%	3.81E+02±5.80E+01(+)	15.2%	3.96E+02±6.94E-01(+)	0.18%	8.14E+01±1.15E+01(+)	14.1%	7.31E-13±5.13E-14(+)	7.02%
f_{17}	1.84E+06±3.03E+05	16.5%	1.61E+05±1.25E+04(-)	7.76%	1.23E+05±5.22E+04(-)	42.4%	1.63E+05±1.12E+04(-)	6.87%	**4.06E+04±2.62E+03(-)**	6.45%
f_{18}	1.97E+03±6.27E+02	31.8%	8.25E+03±6.34E+03(+)	76.8%	3.06E+03±3.18E+02(+)	10.4%	1.89E+04±1.05E+04(+)	55.6%	1.63E+10±2.52E+09(+)	15.5%
f_{19}	5.66E+06±2.88E+05	5.09%	1.37E+06±8.20E+04(-)	5.99%	1.54E+06±1.08E+05(-)	7.01%	**7.36E+05±3.32E+04(-)**	4.51%	1.74E+06±9.34E+04(-)	5.37%
f_{20}	1.25E+03±1.18E+02	9.44%	2.05E+03±1.59E+02(+)	7.76%	2.14E+03±1.66E+02(+)	7.76%	3.35E+03±1.94E+02(+)	57.9%	6.42E+10±7.46E+09(+)	11.6%
+(DGLDPSO is significantly better)			14		14		15		15	
-(DGLDPSO is significantly worse)			6		4		5		5	
≈			0		2		0		0	
fun	DGLDPSO		CSO		SL-PSO		DMS-L-PSO		DLLSO	
	Mean±Std	C.V	Mean±Std	C.V	Mean±Std	C.V	Mean±Std	C.V	Mean±Std	C.V
f_1	4.55E-21±2.61E-22	5.74%	4.58E-12±6.15E-13(+)	13.4%	8.47E-18±1.63E-18(+)	19.2%	4.88E+09±1.34E+08(+)	2.75%	5.96E-22±6.18E-23(-)	10.3%
f_2	7.35E+02±4.52E+01	6.15%	7.48E+03±1.82E+02(+)	2.43%	1.94E+03±1.07E+02(+)	5.52%	6.17E+03±1.87E+02(+)	3.03%	9.48E+02±4.86E+01(+)	5.06%
f_3	2.33E-13±1.55E-14	6.65%	2.58E-09±2.22E-10(+)	8.60%	1.76E+00±2.17E-01(+)	12.3%	1.74E+01±5.26E-02(+)	0.30%	**2.12E-14±2.66E-15(-)**	12.5%
f_4	3.26E+11±5.98E+10	18.3%	7.21E+11±1.75E+11(+)	24.3%	**2.86E+11±5.93E+10(-)**	20.7%	3.90E+13±4.40E+12(+)	11.3%	5.56E+11±1.00E+11(+)	18.0%
f_5	2.83E+07±1.06E+06	3.75%	**1.13E+07±4.94E+07(-)**	437%	2.61E+07±5.44E+06(≈)	20.8%	1.04E+08±7.83E+06(+)	7.53%	1.32E+07±4.41E+06(-)	33.4%
f_6	**4.14E-09±1.39E-10**	3.36%	8.53E-07±2.25E-08(+)	2.64%	2.00E+01±4.08E+00(+)	20.4%	1.66E+06±1.46E+05(+)	8.80%	6.14E-01±8.45E-01(+)	138%
f_7	**2.43E+01±1.00E+01**	41.2%	2.10E+04±4.87E+03(+)	23.2%	5.97E+04±3.49E+04(+)	58.5%	2.42E+10±1.63E+09(+)	6.74%	8.48E+02±1.23E+03(+)	145%
f_8	2.84E+07±1.99E+05	0.70%	3.86E+07±7.68E+04(+)	0.20%	**7.71E+06±8.64E+05(-)**	11.2%	1.43E+08±3.42E+07(+)	23.9%	3.23E+07±4.68E+05(+)	1.44%
f_9	4.47E+07±3.44E+06	7.70%	6.81E+07±5.58E+06(+)	8.19%	**3.44E+07±3.01E+06(-)**	8.75%	5.79E+09±2.02E+08(+)	3.49%	4.18E+07±4.55E+06(≈)	10.9%
f_{10}	2.05E+03±1.26E+02	6.15%	9.57E+03±8.65E+01(+)	0.90%	3.08E+03±1.83E+03(+)	59.4%	5.88E+03±2.48E+02(+)	4.22%	**8.26E+02±3.40E+01(-)**	4.12%
f_{11}	**3.28E-13±1.23E-14**	3.75%	4.01E-08±4.54E-09(+)	11.3%	2.46E+01±4.47E+00(+)	18.2%	1.82E+02±1.38E+00(+)	0.76%	4.69E+00±5.64E+00(+)	120%
f_{12}	6.25E+04±3.55E+03	5.68%	4.38E+05±5.79E+04(+)	13.2%	1.72E+04±1.59E+04(-)	92.4%	2.84E+06±1.10E+05(+)	3.87%	1.58E+04±2.02E+03(-)	13.1%
f_{13}	**5.70E+02±1.83E+02**	32.1%	5.76E+02±1.30E+02(≈)	22.6%	1.07E+03±5.21E+02(+)	48.7%	9.68E+07±2.62E+07(+)	27.1%	8.26E+02±2.40E+02(+)	29.1%
f_{14}	1.46E+08±8.59E+06	5.88%	2.47E+08±1.37E+07(+)	5.55%	**8.51E+07±5.69E+06(-)**	6.69%	5.02E+09±3.43E+08(+)	6.83%	1.67E+08±8.71E+06(+)	5.22%
f_{15}	1.05E+04±9.96E+01	0.95%	1.01E+04±4.12E+01(≈)	0.41%	1.13E+04±1.08E+02(+)	0.96%	6.21E+03±2.76E+02(-)	4.44%	**8.07E+02±3.68E+01(-)**	4.56%
f_{16}	**3.58E-13±1.14E-14**	3.18%	5.76E-08±4.78E-09(+)	8.29%	2.27E+01±9.99E+00(+)	44.0%	3.39E+02±9.52E-01(+)	0.28%	6.25E+00±2.87E+00(+)	45.9%
f_{17}	1.84E+06±3.03E+05	16.5%	2.19E+06±1.37E+05(+)	6.26%	9.22E+04±2.21E+04(-)	24.0%	2.67E+06±1.54E+05(+)	5.77%	9.28E+04±4.43E+03(-)	4.77%
f_{18}	1.97E+03±6.27E+02	31.8%	**1.66E+03±5.90E+02(-)**	35.5%	2.56E+03±7.56E+02(+)	29.5%	2.82E+09±5.30E+08(+)	18.8%	2.94E+03±7.96E+02(+)	27.1%
f_{19}	5.66E+06±2.88E+05	5.09%	9.97E+06±5.93E+05(+)	5.95%	5.18E+06±7.00E+05(≈)	13.5%	1.63E+07±6.70E+05(+)	4.11%	1.63E+06±9.61E+04(-)	4.98%
f_{20}	1.25E+03±1.18E+02	9.44%	**1.11E+03±2.59E+02(≈)**	23.3%	1.77E+03±1.88E+02(+)	10.6%	4.10E+09±7.56E+08(+)	11.1%	1.74E+03±1.68E+02(+)	9.66%
+(DGLDPSO is significantly better)			15		12		19		11	
-(DGLDPSO is significantly worse)			2		6		1		8	
≈			3		2		0		1	

and DLLSO on these six functions are plotted in Fig. S1 in the supplementary material for saving space.

From Fig. S1(a) in the supplementary material, we can see that only DGLDPSO and DLLSO can converge to good solutions quickly on separable function f_3, while other algorithms occur stagnation in the early stage or evolve very slow. While on the partially separable functions f_6, f_{11}, and f_{13}, shown in Fig. S1(b)–(d) in the supplementary material, DGLDPSO and CSO can find the better results and converge faster than other algorithms. Moreover, DGLDPSO is still more accurate and has a faster convergence speed than CSO, showing DGLDPSO converges the fastest to the best final results. Note that in the partially separable function f_{16} in Fig. S1(e) in the supplementary material, only DGLDPSO and DECC-DG can get the promising results, where DGLDPSO still obtains more accurate result than DECC-DG does. The curves in Fig. S1(f) in the supplementary material also show that DGLDPSO can converge to promising results on the very difficult nonseparable function f_{20}, and its early convergence speed is faster than most of the other algorithms. Overall, DGLDPSO generally has faster convergence speed than these compared large-scale optimization algorithms on these benchmark functions. This may be due to that the DGL strategy in DGLDPSO can change the group size dynamically to control the learning strength effectively and to find a potential balance between diversity and convergence speed.

C. Comparison Results on the Large-Scale Cloud Workflow Scheduling

In this experiment, DGLDPSO with ARS is applied into the large-scale cloud workflow scheduling.

Before we conduct the experiments, some extra parameters in the cloud workflow scheduling model should be set. First, for every type of resource r_j, we define its processing capability PC$_j$ as a uniformly distribution Rand(1, 10) within [1, 10] and its cost per unit time as a normal distribution Normal(PC$_j$, 0.1) with mean PC$_j$ and standard deviation 0.1. That is because the resource with good processing capability is often expensive.

Second, for every task t_i, we define its size s_i as Rand(10, 30) within [10, 30], while its *exetime* on r_j is defined as Normal(s_i/PC$_j$, 0.1). Third, the transfer time *transfertime*[i][c] from parent task t_i to its child t_c is calculated according to the size of task t_i and the *bandwidth*, which can be formulated as

$$\textit{transfertime}[i][c] = \text{Rand}(0.1, 1) \times s_i/\textit{bandwidth} \tag{8}$$

where the *bandwidth* is set as 2000 in our experiment.

Two evaluation criteria called success rate (SR) and MeanTEC are used to evaluate the performance of DGLDPSO and other algorithms. For a given MaxFEs and a deadline, SR denotes the percentage of successful runs out of all runs. Here, a successful run means a run where the algorithm can find the feasible solution. The MeanTEC is measured by the average TEC in all successful runs since the TEC is invalid when the algorithm cannot find a feasible solution within MaxFEs.

1) Comparison Results on the Scientific Workflows: We first test the performance of DGLDPSO on four widely used scientific workflows called CyberShake, LIGO, SIPHT, and Montage. The topology structures of these workflows are shown in Fig. 7. More details of these workflows can be referred to [50].

We generate three test instances for each scientific workflow, where the numbers of tasks are 1000, 1500, and 2000, respectively, and the numbers of resources are 100, 150, and 200, respectively. For each test instance, we set three deadlines which are loose, medium, and tight, respectively, to test whether a given algorithm can find the feasible solution within the MaxFEs. For the test instance with 1000 tasks, we use the deadlines as 200, 170, and 140, respectively. For the test instance with 1500 tasks, we use the deadlines as 300, 250, and 200, respectively. For the test instance with 2000 tasks, we use the deadlines as 400, 340, and 280, respectively.

The detailed comparison results over 30 runs are shown in Table II. For clarity, the best results are highlighted in boldface. Table II shows that with the scale of tasks increases and the deadline becomes tighter, the performances of many algorithms are largely weakened, while the DGLDPSO is still promising.

For the CyberShake workflow, only DGLDPSO, GA-PSO, and DOGA and can always find the feasible solution on all the three test instances and within all the three deadlines. However, DGLDPSO can achieve the best performance on MeanTEC compared with the other algorithms.

For the LIGO workflow, which has a relatively simpler topology structure, DGLDPSO, RNPSO, RDPSO, GA-PSO, and DOGA can all find the feasible solution on all the three test instances and within all the three deadlines. RNPSO performs even better than DGLDPSO on 2000 tasks instance, while DOGA achieves smaller MeanTEC than DGLDPSO on 1000 tasks instance when deadline = 200. Even so, DGLDPSO still outperforms all the other algorithms on other test instances.

For the SIPHT workflow, GA-PSO achieves the smallest MeanTEC and outperforms DGLDPSO on instances with 1000 and 2000 tasks. However, DGLDPSO still outperforms other scheduling algorithms and performs the best on instance with 1500 tasks.

For the Montage workflow, which has a relatively more complicated topology structure, the performance of many compared algorithms is further weakened. MTV_PSO and DOGA achieve the similar performance with DGLDPSO on instances with 1500 and 2000 tasks, while DGLDPSO dominates almost all the other algorithms on instance with 1000 tasks.

Algorithm 2: DGLDPSO

```
Process of MASTER in DGLDPSO
Begin
1.  Randomly initialize the population; FEs=0;
2.  Evaluate the population;
3.  FEs=FEs+N;
4.  While FEs⩽MaxFEs
5.     Determine the gbest;
6.     Randomly select an integer as the group size M;
7.     Randomly divide the population into N/M equal groups;
8.     For each group (big "particle")
9.        Send the current group to the corresponding SLAVE;
10.       Receive the updated group from the corresponding SLAVE;
11.    End For
12. End While
End

Process of SLAVE in DGLDPSO
Begin
1.  Receive a group from MASTER;
2.  Determine the pbest;
3.  Update the worst particle $x_w$ using (7) and (2);
4.  Evaluate $x_w$;
5.  FEs=FEs+1;
6.  Send the updated group to MASTER;
End
```

Algorithm 3: DGLDPSO for Large-Scale Cloud Workflow Scheduling

```
Process of MASTER in DGLDPSO
Begin
1.  Sort and renumber the resources according to the metric of
    processing capacity;
2.  Randomly initialize the population; FEs=0; Find=0;
3.  Evaluate the population using Algorithm 1;
4.  FEs=FEs+N;
5.  While FEs⩽MaxFEs
6.     Determine the gbest;
7.     If the current population find the feasible solution && Find=0
8.        Sort and renumber the resources according to the metric of
          cost per unit time;
9.        Find=1;
10.    End If
11.    Randomly select an integer as the group size M;
12.    Randomly divide the population into N/M equal groups;
13.    For each group (big "particle")
14.       Send the current group to the corresponding SLAVE;
15.       Receive the updated group from the corresponding SLAVE;
16.    End For
17. End While
End

Process of SLAVE in DGLDPSO
Begin
1.  Receive a group from MASTER;
2.  Determine the pbest;
3.  Update the worst particle $x_w$ using (7) and (2);
4.  Evaluate $x_w$ using Algorithm 1;
5.  FEs=FEs+1;
6.  Send the updated group to MASTER;
End
```

Overall, on all the 12 test instances and all the 36 cases with different deadlines, DGLDPSO performs better than PSO, RNPSO, AIWPSO, MTV_PSO, RDPSO, GA-PSO, and DOGA on 36, 33, 34, 34, 34, 27, and 23 cases, respectively. Conversely, RNPSO, MTV_PSO, GA-PSO, and DOGA can only surpass DGLDPSO on 1, 1, 5, and 2 cases, respectively. PSO, AIWPSO, and RDPSO cannot surpass DGLDPSO on any cases. Moreover, the smaller C.V values of DGLDPSO

TABLE II
EXPERIMENTAL RESULTS IN SR AND MEANTEC ON SCIENTIFIC WORKFLOWS

Type	Task	Deadline	DGLDPSO			PSO			RNPSO			AIWPSO		
			SR	MeanTEC	C.V	SR	MeanTEC	C.V	SR	MeanTEC	C.V	SR	MeanTEC	C.V
CyberShake	1000	200	**1.000**	**19333.5**	0.98%	**1.000**	19772.1(+)	4.06%	**1.000**	19541.3(+)	0.47%	**1.000**	19631.4(+)	2.25%
		170	**1.000**	**19432.0**	1.74%	**1.000**	19964.2(+)	1.00%	**1.000**	19616.4(+)	0.21%	**1.000**	19840.8(+)	0.66%
		140	**1.000**	**19496.9**	2.07%	**1.000**	20262.7(+)	2.68%	**1.000**	19722.6(+)	1.94%	**1.000**	20020.3(+)	1.24%
	1500	300	**1.000**	**29188.1**	2.29%	**1.000**	29766.5(+)	5.37%	**1.000**	29605.7(+)	1.11%	**1.000**	29709.1(+)	2.06%
		250	**1.000**	**29279.3**	0.49%	**1.000**	30028.3(+)	2.95%	**1.000**	29882.1(+)	4.55%	**1.000**	29945.2(+)	0.83%
		200	**1.000**	**29378.6**	1.81%	0.933	30230.0(+)	0.81%	**1.000**	29913.0(+)	1.45%	0.867	30072.5(+)	0.84%
	2000	400	**1.000**	**39346.2**	2.90%	**1.000**	39731.4(+)	1.24%	**1.000**	39425.5(+)	1.87%	**1.000**	39651.4(+)	0.32%
		340	**1.000**	**39459.5**	1.16%	0.867	39816.2(+)	3.66%	**1.000**	39669.9(+)	0.90%	0.833	39795.0(+)	0.49%
		280	**1.000**	**39608.8**	1.43%	0.467	40052.7(+)	4.58%	0.967	39842.4(+)	0.33%	0.533	39962.7(+)	0.47%
LIGO	1000	200	**1.000**	19620.1	3.38%	**1.000**	19853.3(+)	2.56%	**1.000**	19770.2(+)	1.17%	**1.000**	19799.9(+)	1.26%
		170	**1.000**	**19708.3**	1.56%	**1.000**	19909.5(+)	0.77%	**1.000**	19834.8(+)	1.39%	**1.000**	19805.6(+)	0.60%
		140	**1.000**	**19801.5**	0.18%	**1.000**	20074.9(+)	1.58%	**1.000**	19931.7(+)	2.85%	**1.000**	20007.5(+)	0.10%
	1500	300	**1.000**	**29447.0**	1.71%	**1.000**	29800.2(+)	2.15%	**1.000**	29625.4(+)	0.18%	**1.000**	29728.1(+)	1.14%
		250	**1.000**	**29485.9**	2.39%	**1.000**	30009.8(+)	1.21%	**1.000**	29885.1(+)	1.08%	**1.000**	29802.6(+)	1.55%
		200	**1.000**	**29527.8**	1.94%	**1.000**	30219.3(+)	0.40%	**1.000**	29956.2(+)	0.70%	**1.000**	30021.4(+)	0.79%
	2000	400	**1.000**	39865.6	2.68%	**1.000**	40095.6(+)	1.16%	**1.000**	**39832.0(≈)**	1.75%	**1.000**	40019.2(+)	1.52%
		340	**1.000**	39972.7	0.64%	0.967	40387.0(+)	3.25%	**1.000**	**39902.6(-)**	2.12%	0.933	40445.3(+)	0.82%
		280	**1.000**	40026.3	1.75%	0.800	40475.1(+)	0.39%	**1.000**	**40011.5(≈)**	0.83%	0.767	40667.8(+)	0.12%
SIPHT	1000	200	**1.000**	19624.5	0.73%	**1.000**	19823.6(+)	3.54%	**1.000**	19705.2(+)	0.44%	**1.000**	19804.0(+)	1.86%
		170	**1.000**	19654.1	0.59%	**1.000**	20007.2(+)	1.13%	**1.000**	19798.9(+)	3.17%	**1.000**	19931.7(+)	0.84%
		140	**1.000**	**19683.4**	1.58%	0.967	20308.4(+)	1.61%	**1.000**	19824.3(+)	2.64%	0.967	20006.9(+)	1.51%
	1500	300	**1.000**	**29156.5**	0.54%	**1.000**	29589.2(+)	2.87%	**1.000**	29468.4(+)	1.85%	**1.000**	29407.5(+)	1.86%
		250	**1.000**	**29236.2**	2.98%	0.900	29665.3(+)	1.32%	**1.000**	29531.4(+)	3.83%	**1.000**	29533.6(+)	0.49%
		200	**1.000**	**29393.6**	0.60%	0.767	29734.2(+)	1.00%	0.867	29624.3(+)	1.43%	0.833	29680.6(+)	2.05%
	2000	400	**1.000**	39296.7	4.25%	**1.000**	39483.9(+)	0.11%	**1.000**	39341.8(+)	0.33%	**1.000**	39455.8(+)	0.90%
		340	**1.000**	39354.7	3.22%	0.733	39654.3(+)	0.15%	0.933	39531.6(+)	0.95%	0.700	39580.2(+)	1.72%
		280	**1.000**	39446.0	1.62%	0.567	39724.8(+)	1.43%	0.867	39696.7(+)	2.01%	0.467	39636.3(+)	0.57%
Montage	1000	200	**1.000**	19584.1	1.04%	**1.000**	19715.2(+)	2.18%	**1.000**	19650.4(+)	0.45%	**1.000**	**19572.4(≈)**	1.38%
		170	**1.000**	**19613.7**	1.06%	0.933	19795.6(+)	2.01%	**1.000**	19695.2(+)	2.18%	**1.000**	19629.8(≈)	0.73%
		140	**1.000**	**19665.5**	0.61%	0.867	19851.1(+)	1.01%	0.967	19785.3(+)	2.41%	0.867	19701.4(+)	1.09%
	1500	300	**1.000**	29234.2	0.79%	**1.000**	29465.0(+)	0.97%	**1.000**	29360.1(+)	0.46%	**1.000**	29477.5(+)	3.48%
		250	**1.000**	29362.9	3.37%	0.967	29593.7(+)	1.14%	0.967	29562.4(+)	2.41%	**1.000**	29512.6(+)	1.59%
		200	**1.000**	29436.3	0.18%	0.800	29679.4(+)	0.49%	0.933	29619.2(+)	0.44%	0.833	29597.8(+)	0.47%
	2000	400	**1.000**	39361.7	0.64%	**1.000**	39522.8(+)	1.57%	**1.000**	39491.6(+)	0.73%	**1.000**	39442.7(+)	0.55%
		340	**1.000**	39417.6	0.35%	0.633	39585.3(+)	0.43%	0.833	39566.0(+)	0.31%	0.800	39511.2(+)	1.21%
		280	**1.000**	**39466.8**	2.56%	0.567	39657.7(+)	1.24%	0.667	39620.8(+)	0.57%	0.733	39563.1(+)	0.61%
+(DGLDPSO is significantly better)						36			33			34		
-(DGLDPSO is significantly worse)						0			1			0		
≈						0			2			2		

Type	Task	Deadline	MTV_PSO			RDPSO			GA-PSO			DOGA		
			SR	MeanTEC	C.V	SR	MeanTEC	C.V	SR	MeanTEC	C.V	SR	MeanTEC	C.V
CyberShake	1000	200	**1.000**	19610.7(+)	0.58%	**1.000**	19408.5(+)	0.79%	**1.000**	19448.5(+)	1.23%	**1.000**	19468.3(+)	1.67%
		170	**1.000**	19737.2(+)	0.96%	**1.000**	19528.1(+)	1.29%	**1.000**	19502.3(+)	4.73%	**1.000**	19504.9(+)	1.38%
		140	**1.000**	19982.5(+)	5.90%	**1.000**	19781.4(+)	1.26%	**1.000**	19703.1(+)	1.02%	**1.000**	19556.2(+)	0.42%
	1500	300	**1.000**	29764.3(+)	1.30%	**1.000**	29654.9(+)	0.56%	**1.000**	29620.0(+)	0.66%	**1.000**	29324.3(+)	2.15%
		250	**1.000**	29950.1(+)	0.59%	**1.000**	29755.2(+)	2.99%	**1.000**	29755.9(+)	1.31%	**1.000**	29441.9(+)	3.27%
		200	0.933	30178.0(+)	3.64%	0.767	30032.3(+)	0.75%	**1.000**	29885.2(+)	1.89%	**1.000**	29568.1(+)	1.73%
	2000	400	**1.000**	39628.9(+)	1.27%	**1.000**	39628.4(+)	2.92%	**1.000**	39509.2(+)	0.57%	**1.000**	39497.6(+)	2.11%
		340	0.967	39841.2(+)	4.69%	0.733	39858.0(+)	3.43%	**1.000**	39605.7(+)	3.56%	**1.000**	39619.7(+)	1.90%
		280	0.833	39938.4(+)	1.44%	0.567	39990.3(+)	1.32%	**1.000**	39759.8(+)	1.24%	**1.000**	39726.3(+)	4.31%
LIGO	1000	200	**1.000**	19774.6(+)	1.56%	**1.000**	19701.8(+)	4.87%	**1.000**	19668.4(+)	1.22%	**1.000**	**19596.0(-)**	0.45%
		170	**1.000**	19847.4(+)	1.15%	**1.000**	19808.7(+)	0.39%	**1.000**	19756.5(+)	0.12%	**1.000**	19714.5(≈)	0.23%
		140	**1.000**	19937.1(+)	0.51%	**1.000**	19888.2(+)	1.50%	**1.000**	19864.3(+)	3.51%	**1.000**	19838.4(≈)	1.37%
	1500	300	**1.000**	29861.4(+)	1.98%	**1.000**	29656.5(+)	2.15%	**1.000**	29661.9(+)	0.99%	**1.000**	29484.6(≈)	0.51%
		250	**1.000**	29985.0(+)	1.13%	**1.000**	29847.6(+)	1.80%	**1.000**	29747.3(+)	4.52%	**1.000**	29557.3(+)	2.89%
		200	**1.000**	30238.8(+)	2.26%	**1.000**	29976.0(+)	0.31%	**1.000**	29871.6(+)	0.87%	**1.000**	29631.1(+)	2.58%
	2000	400	**1.000**	40132.4(+)	0.42%	**1.000**	40031.5(+)	1.79%	**1.000**	39991.9(+)	0.36%	**1.000**	39931.3(+)	2.79%
		340	**1.000**	40312.6(+)	0.30%	**1.000**	40157.7(+)	0.72%	**1.000**	40062.1(+)	1.10%	**1.000**	40011.5(+)	0.43%
		280	0.900	40602.7(+)	2.80%	**1.000**	40279.3(+)	0.36%	**1.000**	40126.8(+)	0.43%	**1.000**	40160.7(+)	0.72%
SIPHT	1000	200	**1.000**	19662.9(+)	3.14%	**1.000**	19656.7(≈)	1.12%	**1.000**	**19509.4(-)**	0.22%	**1.000**	19609.3(≈)	5.46%
		170	**1.000**	19798.5(+)	0.55%	**1.000**	19775.8(+)	0.48%	**1.000**	**19641.2(-)**	0.31%	**1.000**	19655.2(≈)	0.61%
		140	**1.000**	19866.1(+)	0.92%	**1.000**	19876.8(+)	3.51%	**1.000**	19685.0(≈)	0.58%	**1.000**	19696.0(+)	0.55%
	1500	300	**1.000**	29491.6(+)	1.63%	**1.000**	29456.1(+)	1.07%	**1.000**	29323.3(+)	1.65%	**1.000**	29340.5(+)	0.64%
		250	0.900	29608.2(+)	3.25%	0.933	29570.4(+)	3.03%	**1.000**	29428.1(+)	0.20%	**1.000**	29406.4(+)	0.31%
		200	0.733	29770.8(+)	4.16%	0.833	29666.6(+)	4.44%	**1.000**	29507.3(+)	1.52%	**1.000**	29457.8(+)	2.67%
	2000	400	**1.000**	39479.0(+)	0.81%	**1.000**	39306.2(≈)	1.59%	**1.000**	**39194.5(-)**	1.51%	**1.000**	39280.5(≈)	1.27%
		340	0.767	39541.3(+)	3.28%	0.867	39417.9(+)	4.85%	**1.000**	**39300.6(-)**	0.13%	**1.000**	39330.7(≈)	4.37%
		280	0.667	39624.9(+)	0.36%	0.700	39557.8(+)	0.11%	**1.000**	**39401.7(-)**	2.29%	**1.000**	**39365.5(-)**	0.87%
Montage	1000	200	**1.000**	19615.4(+)	2.85%	**1.000**	19653.1(+)	1.79%	**1.000**	19607.9(≈)	2.79%	**1.000**	19631.2(+)	0.81%
		170	**1.000**	19700.6(+)	3.89%	**1.000**	19741.5(+)	3.05%	**1.000**	19682.4(+)	0.30%	**1.000**	19664.3(+)	3.57%
		140	0.933	19755.1(+)	2.83%	0.900	19840.2(+)	1.88%	**1.000**	19747.8(+)	4.02%	**1.000**	19692.9(+)	1.04%
	1500	300	**1.000**	**29229.7(≈)**	2.09%	**1.000**	29365.4(+)	1.90%	**1.000**	29325.2(+)	0.44%	**1.000**	29309.6(+)	1.23%
		250	**1.000**	**29334.6(-)**	1.55%	**1.000**	29564.3(+)	0.53%	**1.000**	29448.0(+)	0.14%	**1.000**	29376.8(≈)	0.87%
		200	0.967	**29408.5(+)**	1.22%	0.900	29578.7(+)	0.22%	**1.000**	29516.3(+)	3.75%	**1.000**	29425.1(+)	1.76%
	2000	400	**1.000**	39485.2(+)	0.35%	**1.000**	39441.9(+)	0.91%	**1.000**	39379.6(≈)	0.17%	**1.000**	**39345.2(≈)**	0.23%
		340	0.867	39566.4(+)	0.72%	0.833	39505.0(+)	3.77%	**1.000**	39423.5(≈)	1.59%	**1.000**	**39408.8(≈)**	0.92%
		280	0.733	39616.0(+)	0.48%	0.700	39567.2(+)	0.97%	**1.000**	39496.1(+)	2.03%	**1.000**	39475.3(≈)	**1.20%**
+			34			34			27			23		
-			1			0			5			2		
≈			1			2			4			11		

also indicate the better stability of DGLDPSO. Therefore, DGLDPSO achieves the best performance on the large-scale scientific workflows.

2) Comparison Results on the Randomly Generated Workflows: To further make our test more convincing and comprehensive, we designed an algorithm shown in

TABLE III
EXPERIMENTAL RESULTS IN SR AND MEANTEC ON TEST INSTANCES T_1–T_9

Task	Type	Deadline	DGLDPSO			PSO			RNPSO			AIWPSO		
			SR	MeanTEC	C.V	SR	MeanTEC	C.V	SR	MeanTEC	C.V	SR	MeanTEC	C.V
1000	T_1	2000	**1.000**	21364.5	0.65%	**1.000**	21775.5(+)	0.95%	**1.000**	**21361.7(≈)**	0.26%	**1.000**	21545.3(+)	1.91%
		1700	**1.000**	**21586.8**	0.89%	**1.000**	21952.3(+)	2.65%	**1.000**	21603.8(+)	1.72%	0.933	21735.0(+)	1.35%
		1400	**1.000**	**22169.7**	1.03%	0.467	22661.2(+)	3.17%	**1.000**	22228.8(+)	1.41%	0.600	22264.5(+)	4.51%
	T_2	2000	**1.000**	**21105.7**	2.40%	**1.000**	21445.2(+)	0.68%	**1.000**	21528.3(+)	1.93%	**1.000**	21422.2(+)	0.66%
		1700	**1.000**	**21290.3**	3.17%	**1.000**	21808.5(+)	0.78%	**1.000**	21586.9(+)	2.97%	**1.000**	21510.1(+)	0.99%
		1400	**1.000**	**21637.6**	2.65%	**1.000**	21946.3(+)	1.15%	**1.000**	21702.8(+)	1.48%	**1.000**	21771.7(+)	3.15%
	T_3	2000	**1.000**	21366.5	4.13%	**1.000**	21839.2(+)	1.13%	**1.000**	**21317.3(≈)**	4.00%	**1.000**	21453.5(+)	0.52%
		1700	**1.000**	**21649.3**	0.98%	**1.000**	21841.8(+)	0.93%	**1.000**	21665.8(+)	0.15%	**1.000**	21895.6(+)	0.29%
		1400	**1.000**	21915.5	3.84%	**1.000**	22162.3(+)	2.45%	**1.000**	21918.4(+)	3.80%	0.967	22001.9(+)	0.58%
1500	T_4	3000	**1.000**	**33359.5**	0.47%	**1.000**	35213.4(+)	0.81%	**1.000**	33491.6(+)	2.38%	**1.000**	34035.4(+)	1.87%
		2500	**1.000**	34321.3	0.83%	0.433	35656.6(+)	4.00%	**1.000**	34369.2(+)	0.17%	0.833	34559.2(+)	1.32%
		2000	**1.000**	35403.3	1.06%	0.000	N/A(+)	N/A	0.933	35522.6(+)	1.70%	0.467	35794.9(+)	4.64%
	T_5	3000	**1.000**	**33043.6**	1.45%	**1.000**	34776.7(+)	3.37%	**1.000**	33064.0(≈)	0.77%	**1.000**	33892.9(+)	2.43%
		2500	**1.000**	33752.1	0.82%	**1.000**	34785.1(+)	2.14%	**1.000**	**33700.5(-)**	1.94%	**1.000**	34565.3(+)	0.79%
		2000	**1.000**	34978.1	1.27%	0.833	35897.3(+)	1.26%	0.967	**34909.7(+)**	0.16%	0.467	35213.8(+)	4.82%
	T_6	3000	**1.000**	34462.5	2.82%	**1.000**	35723.1(+)	0.28%	**1.000**	34516.9(≈)	2.58%	**1.000**	34550.6(+)	1.08%
		2500	**1.000**	**34851.1**	1.25%	**1.000**	35785.4(+)	1.87%	**1.000**	35001.8(+)	0.56%	**1.000**	35038.2(+)	3.78%
		2000	**1.000**	**36022.1**	0.27%	0.933	36393.1(+)	1.77%	0.133	36502.0(+)	5.17%	**1.000**	36449.0(+)	0.10%
2000	T_7	4000	**1.000**	47339.7	0.74%	**1.000**	49683.8(+)	1.68%	**1.000**	47442.8(+)	0.56%	**1.000**	**47182.3(-)**	0.16%
		3400	**1.000**	48471.5	6.82%	**1.000**	49748.7(+)	2.29%	**1.000**	48516.2(+)	1.57%	**1.000**	48549.1(≈)	0.94%
		2800	**1.000**	**50075.9**	1.17%	0.967	51316.5(+)	1.24%	0.967	50546.4(+)	1.07%	**1.000**	50944.5(+)	0.85%
	T_8	4000	**1.000**	48364.0	0.22%	**1.000**	50315.6(+)	0.68%	**1.000**	48757.2(+)	1.85%	**1.000**	48574.8(+)	1.59%
		3400	**1.000**	**49267.6**	0.85%	0.167	50893.2(+)	4.36%	**1.000**	49744.9(+)	1.31%	0.333	49886.3(+)	5.47%
		2800	**1.000**	51301.7	3.23%	0.000	N/A(+)	N/A	0.000	N/A(+)	N/A	0.167	51662.9(+)	6.83%
	T_9	4000	**1.000**	47639.0	0.64%	**1.000**	49650.9(+)	1.65%	**1.000**	**47616.6(≈)**	0.46%	**1.000**	48481.6(+)	2.33%
		3400	**1.000**	**48448.2**	1.80%	0.800	49931.3(+)	4.31%	**1.000**	48550.5(+)	0.19%	0.933	49088.1(+)	2.55%
		2800	**1.000**	**49987.3**	2.35%	0.200	51231.1(+)	5.47%	0.733	50463.1(+)	1.74%	0.433	50616.2(+)	4.99%
+(DGLDPSO is significantly better)						27			21			25		
-(DGLDPSO is significantly worse)						0			1			1		
≈						0			5			1		

Task	Type	Deadline	MTV_PSO			RDPSO			GA-PSO			DOGA		
			SR	MeanTEC	C.V	SR	MeanTEC	C.V	SR	MeanTEC	C.V	SR	MeanTEC	C.V
1000	T_1	2000	**1.000**	21562.1(+)	0.52%	**1.000**	21488.5(+)	1.05%	**1.000**	21428.6(+)	2.83%	**1.000**	21594.2(+)	1.19%
		1700	**1.000**	21702.6(+)	0.63%	**1.000**	21733.2(+)	3.91%	**1.000**	21679.4(+)	1.36%	**1.000**	21811.0(+)	2.83%
		1400	0.767	22237.7(+)	2.40%	**1.000**	21998.4(+)	3.47%	**1.000**	22180.2(≈)	2.34%	**1.000**	22184.3(+)	0.60%
	T_2	2000	**1.000**	21373.2(+)	1.78%	**1.000**	21386.7(+)	1.92%	**1.000**	21291.7(+)	3.02%	**1.000**	21358.1(+)	2.23%
		1700	**1.000**	21565.5(+)	0.41%	**1.000**	21578.1(+)	1.96%	**1.000**	21496.3(+)	0.88%	**1.000**	21467.4(+)	0.69%
		1400	**1.000**	21740.3(+)	0.44%	**1.000**	21696.5(+)	1.07%	**1.000**	21702.1(+)	0.74%	**1.000**	21651.5(+)	2.37%
	T_3	2000	**1.000**	21587.0(+)	1.10%	**1.000**	21533.4(+)	4.35%	**1.000**	21462.8(+)	2.19%	**1.000**	21531.5(+)	4.45%
		1700	**1.000**	21748.9(+)	1.70%	**1.000**	21756.7(+)	0.11%	**1.000**	**21630.9(≈)**	1.34%	**1.000**	21662.8(+)	1.84%
		1400	0.833	21911.4(+)	0.82%	0.933	21948.1(+)	4.22%	**1.000**	**21893.0(-)**	1.23%	**1.000**	21928.1(+)	4.75%
1500	T_4	3000	**1.000**	33975.2(+)	2.19%	**1.000**	33768.8(+)	0.76%	**1.000**	33890.4(+)	1.67%	**1.000**	34118.2(+)	3.37%
		2500	0.767	34545.3(+)	1.85%	**1.000**	34331.5(≈)	4.07%	**1.000**	**34233.6(-)**	1.45%	**1.000**	34502.4(+)	0.32%
		2000	0.367	35855.3(+)	4.99%	0.667	35764.3(+)	4.61%	**1.000**	35549.6(+)	2.33%	**1.000**	**35385.0(-)**	1.59%
	T_5	3000	**1.000**	34071.6(+)	1.05%	**1.000**	33874.2(+)	0.53%	**1.000**	33741.5(+)	3.18%	**1.000**	33553.6(+)	1.01%
		2500	**1.000**	34749.3(+)	2.39%	**1.000**	34719.6(+)	0.73%	**1.000**	34462.4(+)	4.35%	**1.000**	34027.8(+)	0.54%
		2000	0.633	35302.1(+)	3.62%	0.767	35327.0(+)	3.85%	**1.000**	35115.2(+)	1.48%	**1.000**	35028.0(+)	2.75%
	T_6	3000	**1.000**	34759.7(+)	1.92%	**1.000**	34688.9(+)	0.51%	**1.000**	34486.8(≈)	0.24%	**1.000**	**34363.5(-)**	0.99%
		2500	**1.000**	35280.8(+)	0.75%	**1.000**	35271.4(+)	0.21%	**1.000**	34975.6(+)	2.44%	**1.000**	34923.2(+)	0.40%
		2000	0.867	36344.5(+)	1.13%	0.900	36407.6(+)	1.89%	**1.000**	36083.9(+)	3.91%	**1.000**	36079.6(+)	1.21%
2000	T_7	4000	**1.000**	47387.6(≈)	2.09%	**1.000**	47269.1(-)	1.60%	**1.000**	48015.2(+)	1.59%	**1.000**	47973.1(+)	7.14%
		3400	**1.000**	**48408.9(≈)**	2.51%	**1.000**	48442.2(≈)	2.07%	**1.000**	48691.0(+)	0.71%	**1.000**	48749.6(+)	2.56%
		2800	**1.000**	51068.4(+)	3.58%	**1.000**	51037.2(+)	3.52%	**1.000**	50098.8(≈)	0.16%	**1.000**	50322.0(+)	0.97%
	T_8	4000	**1.000**	48502.2(+)	1.26%	**1.000**	48493.7(+)	3.07%	**1.000**	48460.5(+)	0.89%	**1.000**	**48323.0(≈)**	2.28%
		3400	0.267	49600.5(+)	4.96%	0.567	49478.1(+)	4.53%	**1.000**	49280.1(≈)	0.33%	**1.000**	49361.0(+)	1.17%
		2800	0.000	N/A(+)	N/A	0.233	51762.5(+)	5.64%	**1.000**	51423.7(+)	1.95%	**1.000**	**51202.2(-)**	2.66%
	T_9	4000	**1.000**	48205.3(+)	0.48%	**1.000**	48192.5(+)	1.31%	**1.000**	48009.2(+)	1.82%	**1.000**	47770.4(+)	0.52%
		3400	0.967	49333.6(+)	0.17%	**1.000**	48961.0(+)	0.70%	**1.000**	49252.6(+)	1.10%	**1.000**	48488.9(≈)	1.90%
		2800	0.567	50406.7(+)	2.01%	0.867	50385.1(+)	1.12%	**1.000**	50135.4(+)	0.86%	**1.000**	50033.5(+)	2.39%
+			25			24			20			22		
-			0			1			2			3		
≈			2			2			5			2		

Algorithm 4 using the stochastic mechanism similar to the one used in [17] to randomly generate the relatively complicated topological structure of the workflows.

We randomly generate nine test instances to further illustrate the superiority of DGLDPSO when solving the large-scale cloud workflow scheduling problems. The first three test instances T_1–T_3 are with 1000 tasks and 100 resources. The following three test instances T_4–T_6 are with 1500 tasks and 150 resources. The last three test instances T_7–T_9 are with 2000 tasks and 200 resources. For the test instance with 1000 tasks, we use the deadlines as 2000, 1700, and 1400, respectively. For the test instance with 1500 tasks, we use the deadlines as 3000, 2500, and 2000, respectively. For the test instance with 2000 tasks, we use the deadlines as 4000, 3400, and 2800, respectively.

The detailed comparison results over 30 runs are shown in Table III. For clarity, the best results are highlighted in boldface. According to Table III, we can conclude the following.

For the first three test instances T_1–T_3 with 1000 tasks, only DGLDPSO, RNPSO, GA-PSO, and DOGA can always find the feasible solution within all the deadlines. However,

TABLE IV
EXPERIMENTAL RESULTS IN SR AND MEANTEC ON 5000-D TEST INSTANCES T_1–T_3

Task	Type	Deadline	DGLDPSO			PSO			RNPSO			AIWPSO		
			SR	MeanTEC	C.V	SR	MeanTEC	C.V	SR	MeanTEC	C.V	SR	MeanTEC	C.V
5000	T_1	11000	**1.000**	**158214.0**	1.35%	**1.000**	160097.6(+)	3.37%	**1.000**	159767.6(+)	2.39%	**1.000**	160078.7(+)	0.54%
		10000	**1.000**	**159973.3**	1.18%	0.800	162325.0(+)	4.14%	**1.000**	161610.9(+)	0.89%	0.933	162053.7(+)	4.68%
		9000	**1.000**	**167720.8**	4.67%	0.533	171356.1(+)	4.48%	0.767	169811.4(+)	2.79%	0.833	169959.1(+)	4.17%
	T_2	11000	**1.000**	**159496.2**	0.97%	**1.000**	160945.1(+)	1.56%	**1.000**	160486.2(+)	1.69%	**1.000**	160501.4(+)	0.22%
		10000	**1.000**	**160859.2**	0.89%	0.767	162306.6(+)	2.55%	0.933	161747.5(+)	4.53%	0.833	161632.6(+)	2.75%
		9000	**1.000**	**169178.3**	0.51%	0.567	171150.9(+)	1.95%	0.833	170199.1(+)	0.27%	0.667	170530.8(+)	3.78%
	T_3	11000	**1.000**	**158685.2**	2.16%	**1.000**	160551.6(+)	4.99%	**1.000**	159627.1(+)	4.06%	**1.000**	160058.4(+)	4.86%
		10000	**1.000**	**160232.9**	4.47%	0.833	162577.5(+)	1.95%	0.933	161344.3(+)	0.69%	0.833	161504.0(+)	4.77%
		9000	**1.000**	**169982.3**	4.59%	0.600	171308.7(+)	3.57%	0.800	170902.2(+)	3.09%	0.700	170572.9(+)	1.15%
+(DGLDPSO is significantly better)						9			9			9		
-(DGLDPSO is significantly worse)						0			0			0		
≈						0			0			0		

Task	Type	Deadline	MTV_PSO			RDPSO			GA-PSO			DOGA		
			SR	MeanTEC	C.V	SR	MeanTEC	C.V	SR	MeanTEC	C.V	SR	MeanTEC	C.V
5000	T_1	11000	**1.000**	159767.5(+)	0.50%	**1.000**	159848.6(+)	2.45%	**1.000**	159326.6(+)	0.91%	**1.000**	159283.2(+)	3.12%
		10000	0.867	161768.6(+)	3.65%	0.867	161611.0(+)	3.23%	**1.000**	160843.4(+)	0.62%	**1.000**	160734.8(+)	3.83%
		9000	0.667	170147.2(+)	3.75%	0.633	170007.7(+)	4.18%	**1.000**	168956.3(+)	1.99%	0.933	169028.4(+)	3.86%
	T_2	11000	**1.000**	160573.6(+)	2.76%	**1.000**	160602.4(+)	4.90%	**1.000**	160051.4(+)	1.61%	**1.000**	159886.3(+)	1.05%
		10000	0.833	161590.1(+)	3.10%	0.83	161977.0(+)	1.80%	**1.000**	161120.9(+)	1.65%	**1.000**	160966.2(≈)	3.14%
		9000	0.700	170257.2(+)	2.46%	0.833	170831.1(+)	3.47%	0.900	171043.9(+)	2.07%	**1.000**	170156.9(+)	2.51%
	T_3	11000	**1.000**	159709.3(+)	4.19%	**1.000**	159531.6(+)	3.70%	**1.000**	159312.8(+)	1.64%	**1.000**	159542.0(+)	2.43%
		10000	0.800	161243.9(+)	3.31%	0.933	161132.5(+)	1.41%	0.967	160983.3(+)	1.78%	**1.000**	160925.7(+)	4.64%
		9000	0.667	170417.5(+)	3.12%	0.767	170616.2(+)	2.95%	0.867	170633.9(+)	3.56%	0.967	170251.4(+)	1.72%
+			9			9			9			8		
-			0			0			0			0		
≈			0			0			0			1		

Algorithm 4 Generate the Workflow Topology Structure

```
Begin
1.  For $i = 1 : T$ //$T$ is the number of tasks in the workflow
2.      $P_c = 0.1 + 0.2 \times i/T$;
3.      For $c = i + 1 : T$
4.          If $Rand(0, 1) \le P_c$
5.              task $t_c$ is the child of task $t_i$;
6.          End If
7.      End For
8.  End For
End
```

DGLDPSO can achieve the best performance on MeanTEC compared with the other algorithms.

For the following three test instances T_4–T_6 with 1500 tasks, PSO, RNPSO, AIWPSO, MTV_PSO, and RDPSO all lose their feasibilities when the deadline is tight. Especially, when deadline = 2000, PSO is totally unfeasible on T_4. Although GA-PSO and DOGA can achieve the similar performance on SR compared with DGLDPSO, they are totally dominated by DGLDPSO on MeanTEC on T_5, no matter on which deadline.

For the last three test instances T_7–T_9 with 2000 tasks, the performance of other compared algorithms is further weakened. When deadline = 2800, PSO, RNPSO, and MTV_PSO cannot find any feasible solution on T_8. However, DGLDPSO still achieves the better or at least comparable MeanTEC compared with the other algorithms on all the three test instances.

Overall, on all the 9 test instances and all the 27 cases, DGLDPSO performs better than PSO, RNPSO, AIWPSO, MTV_PSO, RDPSO, GA-PSO, and DOGA on 27, 21, 22, 25, 20, 25, and 24 cases, respectively. Conversely, RNPSO, AIWPSO, RDPSO, GA-PSO, and DOGA, can only surpass DGLDPSO on 1, 1, 1, 2, and 3 cases, respectively. PSO and MTV_PSO cannot surpass DGLDPSO on any cases. From the results of C.V, we can also see that the performance of DGLDPSO is more stable than other compared algorithms. Therefore, DGLDPSO generally achieves the best performance on the large-scale randomly generated cloud workflow scheduling.

3) Scalability of DGLDPSO: In order to investigate the scalability of DGLDPSO, we further randomly generated three test instances with 5000 tasks and 500 resources using Algorithm 4 to compare the performance of DGLDPSO with these algorithms. Three deadlines are set as 11 000, 10 000, and 9000, respectively. The detail experimental results on the 5000-D scheduling can be seen in Table IV. As we can see, with the increasing scales, the performance of GA-PSO and DOGA is greatly deteriorated. Only DGLDPSO can find all the feasible solutions on all the three test instances and within all the three deadlines. Besides, DGLDPSO performs significantly better and achieves smaller MeanTEC than all the other scheduling algorithms on almost of all the nine cases. Moreover, the performance of DGLDPSO is more stable than the compared algorithms according to the C.V values. These results show that DGLDPSO can also remain good performance when the scale increases to 5000.

From Tables II–IV, DGLDPSO shows its superiority and good scalability, on both the scientific workflows and the randomly generated workflows. As the scale of tasks increases and the deadline becomes tighter, the superiorities of DGLDPSO are increasingly obvious. Therefore, the DGLDPSO not only has a promising performance on the large-scale benchmark functions but also has a promising performance on the large-scale regular scientific workflows, complex randomly generated workflows, and even very large workflows up to 5000 tasks. This may be due to that the multi-group distributed coevolution of DGLDPSO has very strong global search ability in the large-scale optimization. Moreover, the ARS incorporated in DGLDPSO can also make the index of resource meaningful and the learning among particles more clear and more reasonable, helping to solve the large-scale cloud workflows scheduling problems efficiently.

D. Effects of ARS on the Large-Scale Cloud Workflow Scheduling

In order to investigate the effect of ARS, we consider a DGLDPSO variant without ARS, called DGLDPSO-noARS, and compare it with DGLDPSO on all the nine randomly generated workflow instances, the same used in Section IV-C2. The detailed comparison results with respect to SR and MeanTEC between DGLDPSO and DGLDPSO-noARS are listed in Table S.IV in the supplementary material.

As we can see, DGLDPSO can always find the feasible solution on all these nine test instances and on all the deadlines, while DGLDPSO-noARS loses its feasibility when the deadline is tight, such as when deadline $= 1400$ on T_1 and when deadline $= 2000$ on T_4. Meanwhile, DGLDPSO always achieves better MeanTEC when compared with DGLDPSO-noARS. In all, DGLDPSO dominates DGLDPSO-noARS on 25 cases, and the C.V values of DGLDPSO are generally smaller than DGLDPSO-noARS, which fully illustrates the effectiveness of the ARS.

To further investigate the applicability of ARS, we also adopt the ARS in all the other compared algorithms and term their variants with ARS as algorithm-ARS. For example, PSO with ARS is called PSO-ARS. Note that RNPSO with ARS is also called PSO-ARS. The results between these algorithms with and without the ARS are compared in Table S.V in the supplementary material. The results show that the performance of the PSO-based scheduling algorithms, including PSO, RNPSO, AIWPSO, MTV_PSO, and RDPSO, is greatly improved by using the ARS. This indicates that ARS can guide the flying of particles more clearly and make the learning among particles more reasonable. While the performance of the GA-based scheduling algorithms, including DOGA and GA-PSO, is also improved by using the ARS, but not obvious. That may be due to that the evolutionary operators in GA, including selection, crossover, and mutation, do not have the learning and flying process. Therefore, ARS has relatively less influence on the GA-based scheduling algorithms. As the compared algorithms have been enhanced by ARS, we further compare the results obtained by their ARS variants with those obtained by DGLDPSO (which is also with ARS) in Table S.VI in the supplementary material. From Table S.VI in the supplementary material, we can see that DGLDPSO still generally outperforms other algorithms variants with ARS.

Therefore, ARS has the promising effectiveness on the cloud scheduling algorithms, which can be widely applied into other cloud scheduling algorithms and further improve their performance, especially on the PSO-based scheduling algorithms.

V. Conclusion

This article develops a DGLDPSO for large-scale optimization and extends it with ARS for tackling large-scale cloud workflow scheduling. Three major novel designs are developed to improve the performance of the algorithm: 1) master–slave multigroup distributed model; 2) dynamic group learning strategy; and 3) ARS.

Several groups are coevolved by using the master–slave multigroup distributed model, forming a DPSO framework, which can enhance the population diversity. Furthermore, the DGL strategy in DGLDPSO can control the learning strength effectively and can find a potential balance between diversity and convergence. Finally, the ARS can make the index of the resource and the searching process meaningful.

Based on these three novel designs, DGLDPSO can achieve a promising performance when dealing with the large-scale benchmark functions and the large-scale cloud workflow scheduling problems.

References

[1] Z.-G. Chen *et al.*, "Multiobjective cloud workflow scheduling: A multiple populations ant colony system approach," *IEEE Trans. Cybern.*, vol. 49, no. 8, pp. 2912–2926, Aug. 2019.

[2] *Montage: An Astronomical Image Engine*. Accessed: Aug. 2019. [Online]. Available: http://montage.ipac.caltech.edu

[3] W.-N. Chen and J. Zhang, "An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 39, no. 1, pp. 29–43, Jan. 2009.

[4] S. Abrishami, M. Naghibzadeh, and D. H. J. Epema, "Cost-driven scheduling of grid workflows using partial critical paths," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 8, pp. 1400–1414, Aug. 2012.

[5] Z.-H. Zhan *et al.*, "Cloudde: A heterogeneous differential evolution algorithm and its distributed cloud version," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 3, pp. 704–716, Mar. 2017.

[6] G.-P. Liu, "Predictive control of networked multiagent systems via cloud computing," *IEEE Trans. Cybern.*, vol. 47, no. 8, pp. 1852–1859, Aug. 2017.

[7] Y. H. Song, X. Y. He, Z. J. Liu, W. He, C. Y. Sun, and F. Y. Wang, "Parallel control of distributed parameter systems," *IEEE Trans. Cybern.*, vol. 48, no. 12, pp. 3291–3301, Dec. 2018.

[8] X.-F. Liu, Z.-H. Zhan, J. D. Deng, Y. Li, T. L. Gu, and J. Zhang, "An energy efficient ant colony system for virtual machine placement in cloud computing," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 113–128, Feb. 2018.

[9] Z.-H. Zhan, X.-F. Liu, Y.-J. Gong, J. Zhang, H. S. H. Chung, and Y. Li, "Cloud computing resource scheduling and a survey of its evolutionary approaches," *ACM Comput. Surveys*, vol. 47, no. 4, pp. 1–33, Jul. 2015.

[10] F. Zhang, J. W. Cao, K. Hwang, K. Q. Li, and S. U. Khan, "Adaptive workflow scheduling on cloud computing platforms with iterative ordinal optimization," *IEEE Trans. Cloud Comput.*, vol. 3, no. 2, pp. 156–168, Apr./Jun. 2015.

[11] M. Mao and M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal.*, 2011, pp. 1–12.

[12] M. A. Rodriguez and R. Buyya, "Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds," *IEEE Trans. Cloud Comput.*, vol. 2, no. 2, pp. 222–235, Apr. 2014.

[13] L. Liu, M. Zhang, R. Buyya, and Q. Fan, "Deadline-constrained coevolutionary genetic algorithm for scientific workflow scheduling in cloud computing," *Concurrency Comput. Pract. Exp.*, vol. 29, no. 5, pp. 1–12, 2017.

[14] Z.-G. Chen, K.-J. Du, Z.-H. Zhan, and J. Zhang, "Deadline constrained cloud computing resources scheduling for cost optimization based on dynamic objective genetic algorithm," in *Proc. IEEE Congr. Evol. Comput.*, 2015, pp. 708–714.

[15] J. Meena, M. Kumar, and M. Vardhan, "Cost effective genetic algorithm for workflow scheduling in cloud under deadline constraint," *IEEE Access*, vol. 4, pp. 5065–5082, 2016.

[16] H.-H. Li, Y.-W. Fu, Z.-H. Zhan, and J.-J. Li, "Renumber strategy enhanced particle swarm optimization for cloud computing resource scheduling," in *Proc. IEEE Congr. Evol. Comput.*, 2015, pp. 870–876.

[17] Z.-G. Chen *et al.*, "Deadline constrained cloud computing resources scheduling through an ant colony system approach," in *Proc. IEEE Int. Conf. Cloud Comput. Res. Innov.*, 2015, pp. 112–119.

[18] Z.-J. Wang, Z.-H. Zhan, and J. Zhang, "Solving the energy efficient coverage problem in wireless sensor networks: A distributed genetic algorithm approach with hierarchical fitness evaluation," *Energies*, vol. 11, no. 12, pp. 1–14, 2018.
[19] X.-F. Liu *et al.*, "Neural network based information transfer for dynamic optimization," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published. doi: 10.1109/TNNLS.2019.2920887.
[20] Z.-J. Wang *et al.*, "Dual-strategy differential evolution with affinity propagation clustering for multimodal optimization problems," *IEEE Trans. Evol. Comput.*, vol. 22, no. 6, pp. 894–908, Dec. 2018.
[21] X.-F. Liu, Z.-H. Zhan, Y. Gao, J. Zhang, S. Kwong, and J. Zhang, "Coevolutionary particle swarm optimization with bottleneck objective learning strategy for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 23, no. 4, pp. 587–602, Aug. 2019.
[22] Z.-J. Wang *et al.*, "Automatic niching differential evolution with contour prediction approach for multimodal optimization problems," *IEEE Trans. Evol. Comput.*, to be published. doi: 10.1109/TEVC.2019.2910721.
[23] X.-W. Luo, Z.-J. Wang, R.-C. Guan, Z.-H. Zhan, and Y. Gao, "A distributed multiple populations framework for evolutionary algorithm in solving dynamic optimization problems," *IEEE Access*, vol. 7, pp. 44372–44390, 2019.
[24] Z.-J. Wang, Z.-H. Zhan, and J. Zhang, "Distributed minimum spanning tree differential evolution for multimodal optimization problems," *Soft Comput.*, pp. 1–11, Mar. 2019. doi: 10.1007/s00500-019-03875-x.
[25] H. Han, W. Lu, L. Zhang, and J. Qiao, "Adaptive gradient multiobjective particle swarm optimization," *IEEE Trans. Cybern.*, vol. 48, no. 11, pp. 3067–3079, Nov. 2018.
[26] X.-F. Liu *et al.*, "Historical and heuristic-based adaptive differential evolution," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published. doi: 10.1109/TSMC.2018.2855155.
[27] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, 1995, pp. 1942–1948.
[28] S. Abdi, S. A. Motamedi, and S. Sharifian, "Task scheduling using modified PSO algorithm in cloud computing environment," in *Proc. Int. Conf. Mach. Learn. Elect. Mech. Eng.*, 2014, pp. 37–41.
[29] H. M. Alkhashai and F. A. Omara, "An enhanced task scheduling algorithm on cloud computing environment," *Int. J. Grid Distrib. Comput.*, vol. 9, no. 7, pp. 91–100, 2016.
[30] A. Al-Maamari and F. A. Omara, "Task scheduling using PSO algorithm in cloud computing environments," *Int. J. Grid Distrib. Comput.*, vol. 8, no. 5, pp. 245–256, 2015.
[31] A. I. Awad, N. A. El-Hefnawy, and H. M. Abdel_kader, "Enhanced particle swarm optimization for task scheduling in cloud computing environments," *Procedia Comput. Sci.*, vol. 65, pp. 920–929, Sep. 2015.
[32] T. Kaur and S. Pahwa, "An upgraded algorithm of resource scheduling using PSO and SA in cloud computing," *Int. J. Comput. Appl.*, vol. 74, no. 8, pp. 28–32, 2013.
[33] V. Popov, "Particle swarm optimization technique for task-resource scheduling for robotic clouds," *Appl. Mech. Mater.*, vol. 565, pp. 243–246, Jun. 2014.
[34] S. S. Zhao, X. L. Fu, H. H. Li, G. Dong, and J. R. Li, "Research on cloud computing task scheduling based on improved particle swarm optimization," *Int. J. Perform. Eng.*, vol. 13, no. 7, pp. 1063–1069, 2017.
[35] S. Zhao, X. Lu, and X. Li, "Quality of service-based particle swarm optimization scheduling in cloud computing," in *Proc. Int. Conf. Comput. Eng. Netw.*, 2015, pp. 235–242.
[36] N. Sadhasivam and P. Thangaraj, "Design of an improved PSO algorithm for workflow scheduling in cloud computing environment," *Intell. Autom. Soft Comput.*, vol. 23, no. 3, pp. 493–500, 2017.
[37] Z. Wu, Z. Ni, L. Gu, and X. Liu, "A revised discrete particle swarm optimization for cloud workflow scheduling," in *Proc. IEEE Int. Conf. Comput. Intell. Security*, 2010, pp. 184–188.
[38] S. Xue, W. Shi, and X. Xu, "A heuristic scheduling algorithm based on PSO in the cloud computing environment," *Int. J. U e-Service Sci. Technol.*, vol. 9, no. 1, pp. 349–362, 2016.
[39] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments," in *Proc. IEEE Int. Conf. Comput. Intell. Security*, 2010, pp. 400–407.
[40] A. M. Manasrah and H. B. Ali, "Workflow scheduling using hybrid GA-PSO algorithm in cloud computing," *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 1–16, Jan. 2018.
[41] 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, Rep., 2009. [Online]. Available: http://staff.ustc.edu.cn/Ÿketang/cec2012/lsgo_competition.htm
[42] Z. Y. Yang, K. Tang, and X. Yao, "Multilevel cooperative coevolution for large scale optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2008, pp. 1663–1670.
[43] 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.
[44] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Inf. Sci.*, vol. 178, no. 15, pp. 2985–2999, 2008.
[45] M. 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.
[46] 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.
[47] R. Cheng and Y. Jin, "A social learning particle swarm optimization algorithm for scalable optimization," *Inf. Sci.*, vol. 291, pp. 43–60, Jan. 2015.
[48] 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.*, 2008, pp. 3845–3852.
[49] Q. Yang, W.-N. Chen, J. D. Deng, Y. Li, T. L. Gu, and J. Zhang, "A Level-based learning swarm optimizer for large-scale optimization," *IEEE Trans. Evol. Comput.* vol. 22, no. 4, pp. 578–594, Aug. 2018.
[50] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M. H. Su, and K. Vahi, "Characterization of scientific workflows," in *Proc. 3rd Workshop Workflows Support Large Scale Sci.*, Austin, TX, USA, 2008, pp. 1–10.

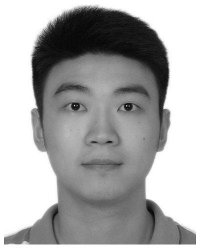
Zi-Jia Wang (S'15) received the B.S. degree in automation from Sun Yat-sen University, Guangzhou, China, in 2015, where he is currently pursuing the Ph.D. degree.

His current research interests include evolutionary computation algorithms like differential evolution, particle swarm optimization, and their applications in design and optimization, such as cloud computing resources scheduling.

Zhi-Hui Zhan (M'13–SM'18) 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.

From 2013 to 2015, he was a Lecturer and an Associate Professor with the Department of Computer Science, Sun Yat-sen University. Since 2016, he has been a Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, where he is also the Changjiang Scholar Young Professor and the Pearl River Scholar Young Professor. 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.

Prof. Zhan was a recipient of the Outstanding Youth Science Foundation from National Natural Science Foundation of China in 2018, the Wu Wen Jun Artificial Intelligence Excellent Youth from the Chinese Association for Artificial Intelligence in 2017, and the China Computer Federation Outstanding Dissertation and the IEEE Computational Intelligence Society Outstanding Dissertation for his doctoral dissertation. He is listed as one of the Most Cited Chinese Researchers in Computer Science. He is currently an Associate Editor of *Neurocomputing*.



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 an Associate Professor 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'13) received the B.Sc. degree in computer science and the Ph.D. degree in computer applied technology from Sun Yat-sen University, Guangzhou, China, in 2007 and 2012, respectively.

She is currently an Associate Professor with the Department of psychology, Sun Yat-sen University. Her current research interests include computational intelligence and its applications in psychology.

Jie Zhang received the master's degree in computer science from the China University of Mining and Technology, Xuzhou, China, in 1999.

She is currently an Associate Professor with the Beijing University of Chemical Technology, Beijing, China. Her current research interests include formal verification, and PHM for power and embedded system design.

Ms. Zhang is a member of the Chinese Institute of Electronics Embedded Expert Committee.

Tian-Long Gu received the M.Eng. degree from Xidian University, Xi'an, China, in 1987, and the Ph.D. degree from Zhejiang University, Hangzhou, China, in 1996.

From 1998 to 2002, he was a Research Fellow with the School of Electrical and Computer Engineering, Curtin University of Technology, Perth, WA, Australia, and a Post-Doctoral Fellow with the School of Engineering, Murdoch University, Perth. He is currently a Professor with the School of Computer Science and Engineering, Guilin University of Electronic Technology, Guilin, China. His current research interests include formal methods, data and knowledge engineering, software engineering, and information security protocol.

Jun Zhang (F'17) received the Ph.D. degree from the City University of Hong Kong, Hong Kong, in 2002.

He is currently a Visiting Scholar with Victoria University, Melbourne, VIC, Australia. His current research interests include computational intelligence, cloud computing, high performance computing, operations research, and power electronic circuits.

Dr. Zhang was a recipient of the Changjiang Chair Professor from the Ministry of Education, China, in 2013, the China National Funds for Distinguished Young Scientists from the National Natural Science Foundation of China in 2011, and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON CYBERNETICS, and the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS.