A Two-Stage Estimation of Distribution Algorithm With Heuristics for Energy-Aware Cloud Workflow Scheduling

Yi Xie, Xue-Yi Wang, Zi-Jun Shen, Yu-Han Sheng, and Gong-Xing Wu

***Abstract*—With the enormous increase in energy usage by cloud data centers for handling various workflow applications, the energy-aware cloud workflow scheduling has become a hot issue. However, there is still a need and room for improvement in both the model for estimating workflow energy consumption and the algorithm for energy-aware cloud workflow scheduling. To fill these gaps, a new model for estimating the energy consumption of the cloud workflow execution and a novel Two-Stage Estimation of Distribution Algorithm with heuristics (TSEDA) for energy-aware cloud workflow scheduling are proposed based on the relationships among scheduling scheme, host load and power. In particular, in the proposed TSEDA, a new probability model and its updating mechanism are presented, and a two-stage coevolution strategy with some novel heuristic methods for individual generation, decoding and improvement is designed. Extensive experiments are conducted on workflow applications with various sizes and types, and the results show that the proposed TSEDA outperforms conventional algorithms.**

***Index Terms*—Workflow, estimation of distribution algorithm, scheduling, cloud computing, energy consumption.**

I. Introduction

WORKFLOWS have been widely used to represent large-scale scientific, business and industrial applications, where nodes stand for tasks and edges for precedence relationships between tasks [1], [2], [3]. Cloud computing has become one of the most promising computing paradigms that fulfill ever-growing computing and storage requirements for the execution of Big Data workflow applications [4], and scheduling the enormous number of user-submitted workflows is an important aspect of cloud computing [5]. Thus scheduling workflow in cloud computing (namely cloud workflow scheduling) has become a vital and popular research area [6], [7], [8], and has attracted more and more attentions from both academia and industry [3], [9], [10].

The increasing demand for process automation and the continuous maturity of cloud computing technology have led to a dramatic increase in the number and variety of workflow applications processed on cloud computing platforms [11], [12]. As a result, more and more data centers have been built across the globe to serve the growing needs of computing and storage for handling those workflow applications from science, business, and industry and other fields [13]. This has led to an enormous increase in energy usage by cloud data centers, which is not only a financial burden but also increases the environmental hazards by emitting large amounts of carbon dioxide and decreases the reliability of system components [5], [13], [14], [15]. During the past decade, cloud data centers have consumed about 2.4% of the global electricity supply [16], and it is estimated that the power consumption of global cloud data centers is increasing by between 15% and 20% per year [16]. Thus, the energy problem hence becomes one of the major concerns in cloud environments [4], [15], [17], so it is necessary to develop an energy efficient cloud workflow scheduling method to reduce the Energy Consumption (EC) as much as possible [5], [17].

In response to these requirements, based on the relationships among scheduling scheme, host load and power, this study presents a new EC model and develops a novel Two-Stage Estimation of Distribution Algorithm with heuristics (TSEDA) for energy-aware cloud workflow scheduling to reduce the EC of cloud workflow execution.

The remainder of this study is organized as follows: Section II reviews the related works and summarizes the main contributions of this study compared to the current studies. Section III formulates the cloud workflow scheduling problem and presents a new EC model. Section IV presents the proposed TSEDA for energy-aware cloud workflow scheduling. Extensive experiments are conducted to evaluate the performance of the proposed TSEDA in Section V. Finally, Section VI concludes our work and points out future work.

II. Related Work

Cloud workflow scheduling refers to assigning workflow tasks to suitable computing resources in the cloud environment and determining their execution orders on each computing resource

Manuscript received 7 March 2023; revised 28 July 2023; accepted 31 August 2023. Date of publication 5 September 2023; date of current version 13 December 2023. This work was supported by the National Social Science Fund of China under Grant 17BGL237. Recommended for acceptance by S. Deng. *(Corresponding authors: Yi Xie; Gong-Xing Wu.)*

Yi Xie is with the School of Management Engineering & E-Business, Zhejiang Gongshang University, Hangzhou, Zhejiang 310018, China, and also with the Contemporary Business and Trade Research Center, Zhejiang Gongshang University, Hangzhou, Zhejiang 310018, China (e-mail: xieyi@mail.zjgsu.edu.cn).

Xue-Yi Wang, Zi-Jun Shen, Yu-Han Sheng, and Gong-Xing Wu are with the School of Management Engineering & E-Business, Zhejiang Gongshang University, Hangzhou, Zhejiang 310018, China (e-mail: 2623648585@qq.com; 3077466069@qq.com; 473289669@qq.com; ywwgx@zjgsu.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TSC.2023.3311785, provided by the authors.

Digital Object Identifier 10.1109/TSC.2023.3311785

without violating the precedence relationships among tasks, so that some desired performance criteria (e.g., makespan, cost, energy, reliability) are satisfied or/and optimized [1], [18], [19]. Thus there are two main issues in cloud workflow scheduling. One is how to select an appropriate computing resource for each task, namely, how to determine the Task-to-Resource Mapping (TRM), and the other is how to determine the Task Scheduling Order (TSO) without violating the precedence relationships among tasks.

Since cloud workflow scheduling is a well-known NP-hard problem [10], [19], [20], [21], [22], [23], there is no algorithm that can obtain the optimal solution in polynomial time. Therefore, Heuristic Algorithms (HAs) and Metaheuristic Algorithms (MAs) are the preferred options to solve this problem [19], [22].

HAs are problem-dependent and deterministic, which can give exact solutions for specific problems in a finite amount of time using the rules set by the developers according to their intuitions and experiences. Many HAs have been developed for various types of cloud workflow scheduling problems in [3], [4], [5], [10], [11], [12], [14], [15], [16], [17], [19], [20], [24], [25], [26], [27], [28], [29], [30], [31], [32]. However, they usually fit only a particular type of problems, so it is impossible for them to always give a good and consistent result for various problems, especially for hard and large-scale optimization problems [33], [34], [35].

Unlike HAs, MAs are problem-independent and non-deterministic, which can combine the knowledge obtained from previous search results with random choices to explore the search space. They can usually produce better scheduling results than HAs [7]. Many MAs have also been developed in [1], [2], [13], [18], [21], [22], [23], [33], [34], [35], [36], [37], [38], [39], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51] to solve cloud workflow scheduling problems by employing Particle Swarm Optimization (PSO), Genetic Algorithm (GA), Ant Colony Optimization (ACO), Chemical Reaction Optimization (CRO).

In order to reduce the search space and computational cost, only the TRM is considered in [13], [22], [36], [37], [38], [39], while the TSO is ignored in them. The TSO is fixed in advance in them, so they could not find the best solution [40].

In [1], [2], [33], [34], [35], [41], [42], [43], the TRM or the TSO is evolved (searched/generated) by a metaheuristic approach, while the other is decoded/determined by a heuristic approach. As a result, the search spaces of MAs presented in [1], [2], [33], [34], [35], [41], [42], [43] are incomplete and the optimal solution may not be found by them.

In [18], [21], [23], [44], [45], [46], [47], [48], [49], [50], [51], both the TRM and the TSO are evolved by a metaheuristic approach (GA or PSO). However, the Hierarchical Coding Technology (HCT), where the change of the TSO is limited in the respective levels, is employed in [18], [44], [45], [46] to maintain its legality and to facilitate the genetic operations, and in [47], theoretically not all the task orders can be gone through because the search space of a task is in the fixed range. Thus, the search spaces of the MAs presented in [18], [44], [45], [46], [47] are also incomplete and the optimal solution may not be found by them. In addition, in [48], the TRM is only used to represent and evolve the type of resources, and the instance of resources is still determined by a heuristic approach, so the search space of the MA presented in [48] is also incomplete in the problem where both the type and the instance of resources need to be considered. The MAs presented in [21], [23], [49], [50], [51] employ GA and have a complete search space, but it is necessary to design some more sophisticated genetic operators to maintain the legality of the TSO encoding based on topological sorting in [21], [23], [49], [50], and the redundant search space inevitably exists in the MA presented in [51] because of the real encoding is employed in it, which further increases the high computational or search cost caused by the large and complete search space.

In terms of optimization objectives, most of the existing studies (e.g., [1], [2], [3], [10], [16], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [33], [34], [35], [36], [37], [38], [39], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51]) focus on makespan and/or cost in cloud workflow scheduling, and only a small minority of them (e.g., [4], [5], [11], [12], [13], [14], [15], [17], [32]) consider the EC. Furthermore, almost all of these aforementioned studies considering EC are based on the Dynamic Voltage and Frequency Scaling (DVFS) enabled environment or technique to estimate and optimize EC. But, due to the frequent adjustments of the operating voltage and frequency of the hosts, the DVFS would not only lead to the degradation in response time and reliability but also cause additional execution costs [14], [15], moreover its application scope is usually limited either to the particular resource site or within the homogeneous cluster [14].

Moreover, the Estimation of Distribution Algorithm (EDA) has been proven to be effective and its variants have been successfully used to solve some optimization problems, such as the project scheduling [52], flow/job shop scheduling [53], [54], facility layout [55], vehicle routing [56], hyperparameter optimization [57], etc. However, to the best of our knowledge, it is rarely applied in cloud workflow scheduling, in particular, an EDA with the two-stage coevolution strategy and some novel heuristic methods for aware-energy cloud workflow scheduling, which can efficiently perform a global search in a complete solution space, have not yet been developed.

As a result, there is still a need and room for improvement in both the model for estimating the workflow EC and the algorithm for energy-aware cloud workflow scheduling. To fill these gaps, this study not only proposes a new model for estimating the workflow EC but also develops a novel TSEDA for energy-aware cloud workflow scheduling based on the relationships among scheduling scheme, host load (CPU utilization) and power. These current studies are summarized and compared with our study as shown in Appendix 1, available online.

The main contributions of this study are summarized as follows:

1) The EC of the cloud workflow execution is estimated and optimized based on the relationships among scheduling scheme, host load and power for the first time. Different from these existing methods based on DVFS, it can reduce/optimize the EC without changing the frequency and voltage of Virtual Machines (VMs)/hosts, which allows the proposed method to have a wider scope of application.

2) A novel TSEDA considering both the TRM and the TSO is developed for energy-aware cloud workflow scheduling for the first time. Different from the traditional EDA and other MAs, a two-stage coevolution strategy with some novel heuristic methods to generate, decode and improve individuals is designed in the proposed TSEDA, so that it can perform efficient search on the complete solution space.

III. Problem Modeling and Formulation

This section introduces the application and system models and basic definitions, then presents the EC model and defines the problem. For ease of reference, Appendix 2, available online, lists the main notations used in this paper and their definitions.

A. Application and System Models

Workflow scheduling can be divided into two categories: static scheduling and dynamic scheduling. In static scheduling, all information about the workflow, such as the number of tasks and resources, the structure of the workflow, the execution time of each task on the resources, and the communication/transfer time between the tasks/resources, is assumed to be known by the scheduling algorithm at compile time (before scheduling). Whereas, in dynamic scheduling, all of the above information can only be known at run time. Comparative studies show that static scheduling outperforms dynamic scheduling in most cases from various perspectives [8]. The reason is that static scheduling obtains more information than dynamic scheduling and can search globally in the solution space using this available prior information [8]. Therefore, this study focuses on the static scheduling of the workflow. Tasks and the precedence relationships between them in a workflow can be represented as a directed acyclic graph $G = \{T, E\}$. $T = \{t_1, \ldots, t_I\}$ is the finite set of tasks, where t_i is task i, and I is the number of tasks. E is the set of directed edges between tasks. A directed edge from $t_{\overleftarrow{i}}$ to t_i is denoted by $e_{\overleftarrow{i},i}$. Each directed edge $e_{\overleftarrow{i},i} \in E$ represents the precedence relationship between $t_{\overleftarrow{i}}$ and t_i, where $t_{\overleftarrow{i}}$ is said to be the parent task of t_i, and t_i is said to be the child task of $t_{\overleftarrow{i}}$. All input data of a task must be received before its execution can begin, and none of its output data is available until the execution of a task is finished. Let l_i be the length (the number of instructions required to be executed) of t_i. Workflows process data in the form of files, thus let IFL_i and OFL_i be respectively the lists of input files and output files of t_i. Thus, the workflow model is defined as $WM = \{G, \{l_i, IFL_i, OFL_i\}\}$. Let PR_i be the set of indexes of parent tasks of t_i and SC_i be the set of indexes of child tasks of t_i, thus $PR_i = \{\overleftarrow{i} | e_{\overleftarrow{i},i} \in E\}$, $SC_i = \{\overrightarrow{i} | e_{i,\overrightarrow{i}} \in E\}$, $i \in PR_{\overrightarrow{i}} \Leftrightarrow \overrightarrow{i} \in SC_i$.

In a cloud computing environment, when a workflow is executed, VMs are used as the basic processing unit to receive and process workflow tasks [31]. A Virtual Machine (VM) is created/deployed in a single host that allocates computing power and bandwidth to it. In cloud computing, the number of VMs is very large and usually regarded as "infinite", but the type of resource (host and/or VM) is finite [48]. On the other hand, each task can only be executed once by a VM during workflow execution. Thus, the number of resources (hosts and/or VMs) that can be used or need to be considered during workflow execution is finite. In fact, for workflow scheduling in cloud computing, the size of the resource pool (resources that can be used or need to be considered during workflow execution) is usually determined in advance by heuristic methods (such as [26], etc.) or set directly to $mpt \times rtp$ to maintain the integrity of all possible solutions and reduce the search space (such as [21], [22], [36], [38], [39], etc.), where rtp is the number of available VM types and mpt is the maximum number of tasks that can be executed in parallel during workflow execution. Therefore, without loss of generality, let $VM = \{vm_1, \ldots, vm_J\}$ be the set of VMs, where vm_j is VM j, and J is the number of VMs. Let $HT = \{ht_1 \ldots, ht_K\}$ be the set of hosts, where ht_k is host k, and K is the number of hosts. Let c_k^{ht} be the processing capacity of ht_k. Let $p_k(ld)$ be the load-power function of ht_k, namely, the power (EC per unit time) of ht_k when its load is ld. $p_k(ld)$ can be obtained by linear interpolation according to the data provided by the SPECpower benchmark available at http://www.spec.org/power_ssj2008/results/power_ssj2008.html. Let c_j^{vm} and b_j^{vm} be the processing capacity and the bandwidth of vm_j respectively. Let hid_j be the index of host where vm_j is created/deployed. Let $VM_k = \{j | hid_j = k\}$ be the set of indexes of VMs created in ht_k. Thus, the resource model for EC in cloud computing environment is defined as $CR = \{VM, HT, \{c_k^{ht}, p_k(ld), VM_k\}, \{c_j^{vm}, b_j^{vm}\}\}$. Let $st_j(\tau) \in \{0, 1\}$ be the state of vm_j at time τ, $st_j(\tau) = 1$ indicates that vm_j is processing a task at time τ, and $st_j(\tau) = 0$ indicates that vm_j is free/available at time τ.

Compared with the peer-to-peer model where the files are transferred directly from the VM running the task to the VM running its child tasks, the global storage model is not only easy to recover in case of failure but also beneficial to improve resource utilization, reduce VM/host rental costs and avoid redundant computing [58]. Thus, similar to [28], [29], [30], the global storage model is used in this study to share the intermediate files between tasks, such as Amazon S3 and Amazon EFS. In this model, tasks store their outputs in the global storage and retrieve their inputs from it. It should be noted that for a task there is no need to transfer/read these input files produced by its parent task when this task and its parent task run on the same VM.

B. Basic Definitions

The intermediate files to be transferred from a task ($t_{\overleftarrow{i}}$) to its child task (t_i) are produced by $t_{\overleftarrow{i}}$ and received/used by t_i, and their size can be calculated by (1). The original (external input) files of a task refer to those input files that are not produced by its parent tasks, and their size can be calculated by (2). Where, d_f is the size of file f.

$$d^{if}_{\overleftarrow{i},i} = \sum\nolimits_{f \in IFL_i \wedge f \in OFL_{\overleftarrow{i}}} d_f \tag{1}$$

$$d^{ef}_{i} = \sum\nolimits_{f \in IFL_i \wedge f \notin \cup_{\overleftarrow{i} \in PR_i} OFL_{\overleftarrow{i}}} d_f \tag{2}$$

The input files of a task include the intermediate files produced by its parent tasks and the original files. Furthermore, when this task and its parent task run on the same VM, the intermediate files between them do not need to be transferred. Thus, when a task (t_i) is executed, the time it takes to transfer/read the required input files from the global storage is calculated by (3), and the time it takes to transfer/write the output files to the global storage is calculated by (4), where vid_i be the index of the VM executing t_i.

$$tt_i^{in} = 8 \times d_i^{ef} \Big/ b_{vid_i}^{vm} + \sum\nolimits_{\overleftarrow{i} \in PR_i \wedge vid_{\overleftarrow{i}} \neq vid_i} 8 \times d_{\overleftarrow{i},i}^{if} \Big/ b_{vid_i}^{vm} \tag{3}$$

$$tt_i^{out} = \sum_{f \in OFL_i} 8 \times d_f \big/ b_{vid_i}^{vm} \tag{4}$$

The total execution time of a task (t_i) includes the time it takes to transfer/read the required input files from the global storage, the task's processing time, and the time it takes to transfer/write the output files to the global storage. Thus, it can be calculated by (5).

$$et_i = l_i \big/ c_{vid_i}^{vm} + tt_i^{in} + tt_i^{out} \tag{5}$$

There are two policies to find resource available time for a task execution in the scheduling: non-insertion policy and insertion policy. In the non-insertion policy, the available time of a resource for a task execution is the finish time of its last assigned task. Whereas, in the insertion policy, the possible insertion of a task in an earliest idle time slot between two already scheduled tasks on a resource may be considered. The insertion policy can fully utilize the idle time slots of VMs, so it can produce the better scheme than the non-insertion policy. In addition, tasks can be scheduled forward or backward with respect to the TSO. In the Forward Scheduling based on Insertion Policy (FSIP) only those tasks without unscheduled parent tasks can be scheduled to start as early as possible, starting from a task without parent tasks. Whereas, in the Backward Scheduling based on Insertion Policy (BSIP) only those tasks without unscheduled child tasks can be scheduled to start as late as possible, starting from a task without child tasks. Let τ_i^s and τ_i^f be the start time and finish time of t_i. Thus, the ready time of t_i, denoted by τ_i^r, is the earliest time at which t_i can be executed without considering resource constraints, and can be is calculated in the FSIP and BSIP respectively by (6) and (7).

$$\tau_i^r = \begin{cases} 0 & PR_i = \emptyset \\ \max\limits_{\overleftarrow{i} \in PR_i} \{\tau_{\overleftarrow{i}}^f\} & PR_i \neq \emptyset \end{cases} \tag{6}$$

$$\tau_i^r = \begin{cases} 0 & SC_i = \emptyset \\ \max\limits_{\overrightarrow{i} \in SC_i} \{\tau_{\overrightarrow{i}}^f\} & SC_i \neq \emptyset \end{cases} \tag{7}$$

Let $ITL_j = [u_j^1, v_j^1] \cup \cdots \cup [u_j^{Q_j}, v_j^{Q_j}] = \cup_{q=1}^{Q_j} [u_j^q, v_j^q]$ be the idle time-slot lists of vm_j that is its current available time lists for tasks, where $u_j^1 < v_j^1 < \cdots < u_j^{Q_j} < v_j^{Q_j} = \infty$, $[u_j^q, v_j^q]$ is the qth idle time-slot and Q_j is the number of idle time-slots in ITL_j. The VM available time for t_i, denoted by τ_i^a, is the start time of the earliest idle time-slot which can be used to execute t_i, and can be calculated by:

$$\tau_i^a = \min\{u_{vid_i}^q \,|\, v_{vid_i}^q - \max\{u_{vid_i}^q, \tau_i^r\} \geq et_i, q = 1, \ldots, Q_{vid_i}\} \tag{8}$$

Thus, the start time and the finish time of t_i can be calculated by (9) and (10).

$$\tau_i^s = \max\{\tau_i^r, \tau_i^a\} \tag{9}$$

$$\tau_i^f = \tau_i^s + et_i \tag{10}$$

It should be noted that the actual start and finish time of t_i are $\max\{\tau_1^f, \ldots, \tau_I^f\} - \tau_i^f$ and $\max\{\tau_1^f, \ldots, \tau_I^f\} - \tau_i^s$ respectively when the BSIP is used.

The average execution time of t_i is defined as follows:

$$\overline{et_i} = \sum_{j=1}^{J} \left(l_i \big/ c_j^{vm} + \sum_{f \in IFL_i \cup OFL_i} 8 \times d_f \Big/ b_j^{vm} \right) \Big/ J \tag{11}$$

The upward rank of t_i, which is the average length of the critical execution path from t_i to the exit tasks and reflects the average remaining time to finish all tasks after t_i starts up, is defined recursively as follows:

$$ur_i = \begin{cases} \overline{et_i} & SC_i = \emptyset \\ \overline{et_i} + \max\limits_{\overrightarrow{i} \in SC_i} \{ur_{\overrightarrow{i}}\} & SC_i \neq \emptyset \end{cases} \tag{12}$$

C. Energy Consumption Model

The EC and power of the host are closely related to its load, and the load depends on the state of the VM which is determined by the scheduling scheme. Therefore, different from the existing DVFS-based estimation method, a new model for estimating the EC of cloud workflow execution is presented based on the relationships among scheduling scheme, host load and power as follows:

First, the load of ht_k at time τ, which is the ratio of the processing capacity of these VMs that are created in ht_k and are in use at time τ to the processing capacity of ht_k, can be calculated by (13).

$$ld_k(\tau) = \sum_{j \in VM_k} c_j^{vm} \cdot st_j(\tau) \Big/ c_k^{ht} \tag{13}$$

Then, as shown in (14), the energy consumed by ht_k in any time period $[\tau_1, \tau_2]$ can be estimated by integrating the load-power function of ht_k at $[\tau_1, \tau_2]$.

$$e_k\big|_{\tau_1}^{\tau_2} = \int_{\tau_1}^{\tau_2} p_k(ld_k(\tau)) d\tau \tag{14}$$

Finally, as shown in (15), the EC of cloud workflow execution can be estimated by summing the energy consumed by all hosts during workflow execution.

$$e^{we} = \sum_{k=1}^{K} \int_{\min_{i \in \{i' | vid_{i'} \in VM_k\}} \{\tau_i^s\}}^{\max_{i \in \{i' | vid_{i'} \in VM_k\}} \{\tau_i^f\}} p_k(ld_k(\tau)) d\tau \tag{15}$$

In this paper, the scheduling problem is how to select an appropriate VM for each task and determine the execution order

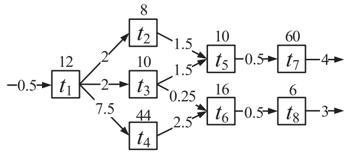

Fig. 1. The workflow application with 8 tasks.

of tasks on VMs without violating the precedence relationships between tasks so that the EC can be minimized.

D. A Numerical Case

To illustrate our problem and models with an example and to provide a case for illustrating the encoding in Section IV.A, the IFBSS and the LBCAS in Section IV.C, this section presents a numerical case.

Fig. 1 shows a workflow application with 8 tasks ($t_1, \ldots, t_8$). The rectangle represents the task, and the number above the rectangle is the length of the task which is the processing time (unit: s) of the task on a Standard Computation Service (SCS), namely,l_1= 12, $\ldots, l_8 = 6$. The arrow represents the input/output file, and the number on the arrow is the size of the file. For example, t_1 has an original (workflow) input file of 0.5 GB from external and three output files of 2 GB, 2 GB, 7.5 GB respectively transferred to t_2, t_3, t_4; and t_8 has an input file of 0.5 GB from t_6 and a final (workflow) output file of 3 GB transferred to external. Thus, we can obtain $d_1^{ef} = 0.5$, $d_2^{ef} = \ldots = d_8^{ef} = 0$, $d_{1,2}^{if} = 2, \ldots, d_{6,8}^{if} = 0.5$. Three VMs ($vm_1, vm_2, vm_3$) created in one host ($ht_1$: NEC Corporation Express5800/GT110f-S) are used to perform the workflow application. The processing capacities of ht_1, vm_1, vm_2, vm_3 are 10SCS, 2SCS, 4SCS, 4SCS respectively, namely,$c_1^{ht} = 10$, $c_1^{vm} = 2$, $c_2^{vm} = 4$, $c_3^{vm} = 4$. Thus, in this case the host has only 6 loads: 0.0, 0.2, 0.4, 0.6, 0.8, 1.0, and the corresponding powers are 15.9J/s, 22.4J/s, 27.2J/s, 33.0J/s, 39.5J/s, 45.1J/s respectively according to the data provided by the SPECpower benchmark, namely, $p_1(0) = 15.9, p_1(0.2) = 22.4, p_1(0.4) = 27.2, p_1(0.6) = 33.0$, $p_1(0.8) = 39.5$, $p_1(1) = 45.1$. The bandwidths of vm_1, vm_2, vm_3 are 2 Gbit/s, 4 Gbit/s, 4 Gbit/s respectively, namely, $b_1^{vm} = 2$, $b_2^{vm} = 4$, $b_3^{vm} = 4$.

The Gantt chart of a solution and the host load and EC at different times in the solution are shown in Fig. 2, in this solution, $t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8$ are assigned to $vm_2, vm_1, vm_3, vm_2, vm_3, vm_2, vm_3, vm_2$ respectively; $\tau_1^s = 0$, $\tau_1^f = 27$, $\tau_2^s = 27$, $\tau_2^f = 45$, $\tau_3^s = 27$, $\tau_3^f = 37$, $\tau_4^s = 27$, $\tau_4^f = 43$, $\tau_5^s = 45$, $\tau_5^f = 51.5$, $\tau_6^s = 43$, $\tau_6^f = 48.5$, $\tau_7^s = 51.5$, $\tau_7^f = 74.5$, $\tau_8^s = 48.5$, $\tau_8^f = 56$; and $makespan = 74.5$, $EC = 27.2 \times 27 + 45.1 \times 10 + 33.0 \times 8 + 39.5 \times 11 + 27.2 \times 18.5 = 2387.1$.

IV. The Proposed Algorithm

Since the cloud workflow scheduling problem is an NP-hard problem whose solution space can increase exponentially with the number of tasks and resources. it is necessary to develop an MA to find optimal or near-optimal solutions in a reasonable time. Therefore, the TSEDA, which can search efficiently in

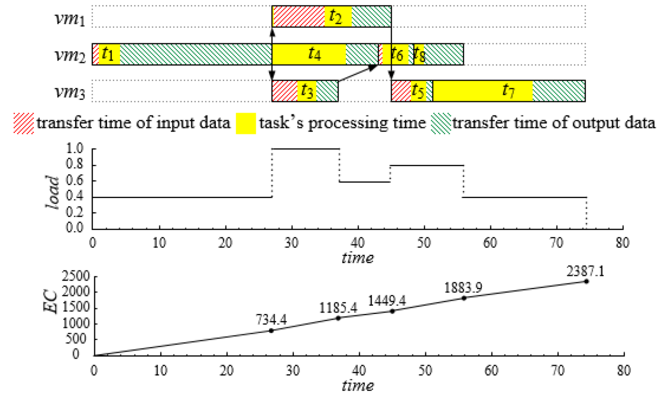

Fig. 2. The Gantt chart of a solution and the host load and EC at different times in the solution.

the complete solution space, is developed for the energy-aware cloud workflow scheduling problem.

A. Individual Encoding and Decoding

In the TSEDA, each individual (also called chromosome) $ch = \{g_1, \ldots, g_I; g_{I+1}, \ldots, g_{2I}\}$ comprises $2I$ genes. $\{g_1, \ldots, g_I\}$ is the TRM to determine the resources allocated to tasks and $\{g_{I+1}, \ldots, g_{2I}\}$ is the TSO to determine the order of tasks to be scheduled, where $g_i \in \{1, \ldots, J\}$ $(i = 1, \ldots, I)$ represents the index of VM allocated to t_i and $g_{I+i} (i = 1, \ldots, I)$ represents the index of the ith scheduled task. For example, $g_1 = 3$ indicates that vm_3 is allocated to t_1, and $g_{I+1} = 2$ indicates that the first scheduled task is t_2. Both the TRM and the TSO are included in the individual code, thus, compared with these algorithms where the TSO is fixed in advance or one of the TSO and TRM is completely decoded/determined by a heuristic approach, the proposed TSEDA has a complete search space. The TSO is a forward or backward topological sort corresponding to $G = \{T, E\}$. If a task has parent tasks, in the forward topological sort the index of this task must occur after the indexes of all its parent tasks, on the contrary, in the backward topological sort the index of this task must occur before the indexes of all its parent tasks.

Algorithm 1 shows how to decode an individual based on the insertion policy. In Algorithm 1, the scheduling scheme is generated first, that is, the start time and finish time of all tasks are determined according to (1)–(10) (Lines 1–10), then the EC is calculated section by section according to (13)–(15) (Lines 11–20). In particular, if the individual's TSO is a forward topological sort, the FSIP should be employed, otherwise the BSIP should be employed.

For example, for the case described in Section III.D, the individual corresponding to the solution/scheme shown in Fig. 2 can be encoded as $\{2, 1, 3, 2, 3, 2, 3, 2; 1, 2, 3, 4, 6, 5, 8, 7\}$, whose TSO is a forward topological sort. Conversely, this individual can also be decoded into the solution/scheme shown in Fig. 2 by Algorithm 1 using the FSIP.

Algorithm 1: Decode An Individual Based on the Insertion Policy.

procedure $Dcd(\&ch)$
input: individual $ch = \{g_1, \cdots, g_{2I}\}$;
output: τ_i^s , τ_i^f , e^{we} ;
$ITL_1, \cdots, ITL_J \leftarrow [0, \infty]$;
for $\delta \leftarrow 1, \cdots, I$ **do**
  $i \leftarrow g_{I+\delta}$; $j, vid_i \leftarrow g_i$;
  calculate et_i according to (1)–(5);
  if the TSO of ch is a forward topological sort **then**
    calculate τ_i^r according to (6);
  else // the TSO of ch is a backward topological sort
    calculate τ_i^r according to (7);
  calculate τ_i^s , τ_i^f according to (8)–(10);
  $ITL_j \leftarrow ITL_j - [\tau_i^s, \tau_i^f]$; //update ITL_j ;
sort τ_i^s , τ_i^f in ascending order, let them be $\tau_1 \le \tau_2 \le \cdots \le \tau_{2I}$; $e^{we} \leftarrow 0$;
$\tau_k^{lts} \leftarrow \min\left\{\tau_i^s \middle| i \in \{i' | hid_{g_{i'}} = k\}\right\}$, $\tau_k^{ltf} \leftarrow \max\left\{\tau_i^f \middle| i \in \{i' | hid_{g_{i'}} = k\}\right\}$, $k = 1, \cdots, K$;
for $\delta \leftarrow 1, \cdots, 2I-1$ **do**
  $ld_k \leftarrow 0$, $k = 1, \cdots, K$;
  for $i \leftarrow 1, \cdots, I$ **do**
    if $[\tau_\delta, \tau_{\delta+1}] \subset [\tau_i^s, \tau_i^f]$ **then**
      $ld_{hid_{g_i}} \leftarrow ld_{hid_{g_i}} + c_{g_i}^{vm} / c_{hid_{g_i}}^{ht}$;
  for $k \leftarrow 1, \cdots, K$ **do**
    if $[\tau_\delta, \tau_{\delta+1}] \subset [\tau_k^{lts}, \tau_k^{ltf}]$ **then**
      $e^{we} \leftarrow e^{we} + (\tau_{\delta+1} - \tau_\delta) \times p_k(ld_k)$;

B. Probability Model and Updating Mechanism

In the EDA, the probability model is used to describe the distribution of the solutions in the search space and to generate new solutions by sampling, and the updating mechanism is used to adjust the probability model so that the search procedure traces the promising search area as much as possible. The probability model and its updating mechanism have a great impact on the performance of EDA, so it is crucial to design them. Since a cloud workflow scheduling scheme involves the TRM and the TSO, in order to ensure a complete search space and reflect the characteristics of the problem and make it easy to implement, as shown in (16) and (17), the probability model for the cloud workflow scheduling problem includes the Mapping Probability Model (MPM) and the Scheduling order Probability Model (SPM), which are respectively designed to describe the probability distributions of tasks assigned to different VMs and tasks arranged in different scheduling orders.

$$MPM(k) = [\alpha_{i,j}(k)]_{I \times J} \tag{16}$$

where $\alpha_{i,j}(k)$ is the probability that t_i is assigned to vm_j in the kth generation, $\alpha_{i,1}(k) + \ldots + \alpha_{i,J}(k) = 1, i = 1, \ldots, I$.

$$SPM(k) = [\beta_{i',i}(k)]_{I \times I} \tag{17}$$

where $\beta_{i',i}(k)$ is the probability that the ith scheduled task is $t_{i'}$ in the kth generation when the FSIP is employed, $\beta_{1,i}(k) + \ldots + \beta_{I,i}(k) = 1, i = 1, \ldots, I$.

The initial MPM is defined as follows:

$$MPM(1) = [1/J]_{I \times J} \tag{18}$$

Let $STS_i = \{t_{i'} | \xi_{i'} < i \le I - \zeta_{i'}\}$ be the set of tasks that may be scheduled in the ith scheduling, where $\zeta_{i'}$ is the number of descendant tasks of $t_{i'}$, $\xi_{i'}$ is the number of ancestor tasks of $t_{i'}$. Thus, the initial SPM is defined as follows:

$$SPM(1) = [\gamma_{i',i} / |STS_i|]_{I \times I} \tag{19}$$

where, if $t_{i'} \in STS_i$ ($t_{i'}$ may be scheduled in the ith scheduling), $\gamma_{i',i} = 1$, otherwise, $\gamma_{i',i} = 0$.

Let ch_{bt} be the best individual found when the population evolves to the k-generation. The MPM and the SPM are updated as follows:

$$\alpha_{i,j}(k+1) = (1 - \theta_1)\alpha_{i,j}(k) + \theta_1 \lambda'_{i,j}(k) \tag{20}$$

$$\beta_{i',i}(k+1) = (1 - \theta_2)\beta_{i',i}(k) + \theta_2 \lambda''_{i',i}(k) \tag{21}$$

where, if t_i is assigned to vm_j in ch_{bt}, $\lambda'_{i,j}(k) = 1$, otherwise $\lambda'_{i,j}(k) = 0$; and if the TSO of ch_{bt} is a forward topological sort and the ith scheduled task is $t_{i'}$ in ch_{bt}, or the TSO of ch_{bt} is a backward topological sort and the I-i+1th scheduled task is $t_{i'}$ in ch_{bt}, $\lambda''_{i',i}(k) = 1$, otherwise $\lambda''_{i',i}(k) = 0$. $\theta_1, \theta_2 \in (0, 1)$ are the update rates of the MPM and the SPM respectively.

The updating process is a kind of incremental learning, where the second term on the right side of (20) and (21) represents the learning information from the best individuals of previous generations. Based on this updating mechanism, the probability model can be well adjusted in each generation to reveal the general distribution with the characteristics of these best individuals and make the search procedure always trace the most promising search area.

C. Individual Generation and Improvement

Unlike the GA which generates offspring by crossover and mutation operation, the EDA does it by sampling according to a probability model. Moreover, different from the traditional EDA, this paper designs and uses a two-stage coevolution strategy and some novel heuristic methods/information to generate, decode and improve individuals. In stage 1, the TSO of the individual is first generated by sampling according to the SPM and the dynamic heuristic information; then the TRM of the individual is generated and the individual is decoded by a heuristic method, so that the proposed algorithm can converge to the vicinity of the optimal solution as soon as possible. In stage 2, besides the TSO, the TRM of the individual is also generated by sampling according to the MPM, and the individual is decoded by the FSIP, furthermore, an Iterative Forward and Backward Scheduling Strategy (IFBSS) with reordering the TSO and a Load Balance and Communication Avoidance Strategy (LBCAS) are used to expand the neighborhood search for improving the individual.

In particular, different from the traditional EDA, the dynamic heuristic information of tasks based on upward ranks ($\eta_i = (ur_i / \max\{ur_1, \ldots, ur_I\})^{\varphi(1 - cn/tn)}$, $i = 1, \ldots, I$) and the heuristic method based on the Task-Minimum-EC (called as TMEC) are designed and used to generate better individuals in the TSEDA. Where, $\varphi > 0$ is the factor of dynamic heuristic information, cn is the current time or number of evolution iterations, tn is the total time or number of evolution iterations, and the TMEC assigns the selected task to the VM that can minimize the increased EC after the task is assigned. Algorithm

Algorithm 2: Generate the TSO of An Individual by Sampling.

function $GnrSchLst(SPM, \{\eta_1, \cdots, \eta_I\})$
input: SPM, $\{\eta_1, \cdots, \eta_I\}$;
output: the TSO $\{g_{I+1}, \cdots, g_{2I}\}$ of individual ch;
$upr_i \leftarrow |PR_i|$; $RTI \leftarrow \{i | upr_i = 0\}$; $ch = \{g_1, \cdots, g_{2I}\} \leftarrow \{null, \cdots, null\}$;
for $\delta \leftarrow 1, \cdots, I$ **do**
  $A_i \leftarrow \eta_i \beta_{i,\delta} / \sum_{i' \in RTI} \eta_{i'} \beta_{i',\delta}$, $i \in RTI$;
  randomly select a task index $\hat{i}$ from RTI using a roulette wheel selection scheme according to A_i.
  $g_{I+\delta} \leftarrow \hat{i}$; $RTI \leftarrow RTI - \{\hat{i}\}$;
  for each $i' \in SC_{\hat{i}}$ **do**
    $upr_{i'} \leftarrow upr_{i'} - 1$;
    if $upr_{i'} = 0$ **then**
      $RTI \leftarrow RTI \cup \{i'\}$;
return ch;

Algorithm 3: Generate the TRM of an Individual and Decode it by the TMEC.

procedure $HrsDcd(\&ch)$
input: individual $ch = \{g_1, \cdots, g_{2I}\}$;
output: the TRM $\{g_1, \cdots, g_I\}$ of ch, τ_i^s, τ_i^f, e^{we};
$e^{we} \leftarrow 0$; $ITL_1, \cdots, ITL_J \leftarrow [0, \infty]$;
for $\delta \leftarrow 1, \cdots, I$ **do**
  $i \leftarrow g_{I+\delta}$;
  for $j \leftarrow 1, \cdots, J$ **do**
    find the start time $\tau_{i,j}^s$ and the finish time $\tau_{i,j}^f$ of t_i using FSIP when t_i is assigned to vm_j;
    calculate the increased EC $\Delta e_{i,j}^{we}$ after t_i is assigned to vm_j;
  $j' \leftarrow \arg\min_{j=1,\cdots,J} \{\Delta e_{i,j}^{we}\}$, $g_i \leftarrow j'$;
  $\tau_i^s \leftarrow \tau_{i,j'}^s$, $\tau_i^f \leftarrow \tau_{i,j'}^f$; $e^{we} \leftarrow e^{we} + \Delta e c_{i,j'}^{we}$; $ITL_{j'} \leftarrow ITL_{j'} - [\tau_i^s, \tau_i^f]$;

Algorithm 4: Generate the TRM of an Individual by Sampling.

procedure $GnrRscLst(\&ch = \{g_1, \cdots, g_{2I}\}, MPM)$
input: individual $ch = \{g_1, \cdots, g_{2I}\}$, MPM;
output: the TRM $\{g_1, \cdots, g_I\}$ of ch;
for $i \leftarrow 1, \cdots, I$ **do**
  randomly select a VM index j' using a roulette wheel selection scheme according to $\alpha_{i,j}$.
  $g_i \leftarrow j'$;

2 shows how to generate the TSO of an individual by sampling according to the SPM and η_i $(i = 1, \ldots, I)$. Algorithm 3 shows how to generate the TRM of an individual and decode it by the TMEC. Algorithm 4 shows how to generate the TRM of an individual by sampling according to the MPM.

From the mechanism of the EDA, it can be seen that the EDA stresses more on the exploration among the space, while the exploitation capability within a certain area should be further enhanced [54]. Thus, different from the traditional EDA, the IFBSS with reordering the TSO shown in Algorithm 5 and the LBCAS shown in Algorithm 6 are developed in the proposed TSEDA to intensify the search in promising regions for finding better individuals.

Algorithm 5: The IFBSS.

procedure $IFBSS(\&ch)$
input: individual $ch = \{g_1, \cdots, g_{2I}\}$;
output: the improved ch by the IFBSS;
$ch' = \{g'_1, \cdots, g'_{2I}\} \leftarrow ch$;
repeat
  $ch'' = \{g''_1, \cdots, g''_{2I}\} \leftarrow ch'$; $UTI \leftarrow \{1, \cdots, I\}$;
  for $\delta \leftarrow 1, \cdots, I$ **do** //reorder the TSO in the descent of $\tau_i''^f$;
    $\hat{i} \leftarrow \arg\max_{i \in UTI} \{\tau_i''^f\}$; $g'_{I+\delta} \leftarrow \hat{i}$; $UTI \leftarrow UTI - \{\hat{i}\}$;
  $Dcd(ch')$;
until $e'^{we} \geq e''^{we}$
if $e'^{we} = e''^{we}$ and the TSO of ch' is a forward topological sort **then**
  $ch \leftarrow ch'$;
else
  $ch \leftarrow ch''$;

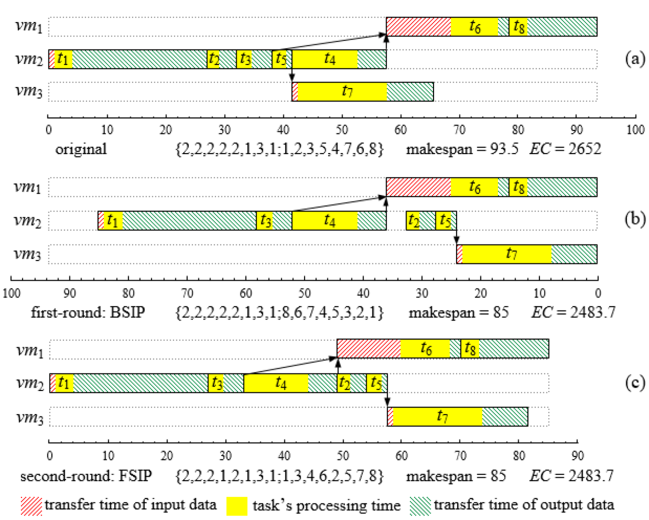

Fig. 3. The illustration of the IFBSS.

The IFBSS is designed to find a better TSO for improving the individual. The instinctive rationale behind the IFBSS is that some tasks in a feasible scheduling scheme usually have some free slack that they can be shifted without affecting the remaining tasks to be executed on schedule, and the workflow makespan and other time-related performance (e.g., EC and cost) can be reduced/optimized by the forward and backward scheduling that shifts the tasks to the far left and right of their free slack and reorders the TSO. Basically, as shown in Algorithm 5 the IFBSS iteratively reorders the TSO of the individual in the descent of the finish time of tasks and uses the FSIP or BSIP to decode the individual until there is no further improvement in the EC.

For the case in Section III.D, Fig. 3 illustrates how the IFBSS is performed on $\{2, 2, 2, 2, 2, 1, 3, 1; 1, 2, 3, 5, 4, 7, 6, 8\}$. First, $\{1, 2, 3, 5, 4, 7, 6, 8\}$ is reordered as $\{8, 6, 7, 4, 5, 3, 2, 1\}$ because of $\tau_8^f > \tau_6^f > \tau_7^f > \tau_4^f > \tau_5^f > \tau_3^f > \tau_2^f > \tau_1^f$ in the scheduling scheme shown in Fig. 3(a), and $\{2, 2, 2, 2, 2, 1, 3, 1; 8, 6, 7, 4, 5, 3, 2, 1\}$is decoded by the BSIP to obtain its scheduling scheme shown in Fig. 3(b). Then, $\{8, 6, 7, 4, 5, 3, 2, 1\}$ is reordered again as $\{1, 3, 4, 6, 2, 5, 7, 8\}$ because of $\tau_1^f > \tau_3^f >$

Algorithm 6: The LBCAS.

procedure $LBCAS(\&ch)$
input: individual $ch = \{g_1, \cdots, g_{2I}\}$;
output: the improved ch by the LBCAS;
$ld_j \leftarrow \sum_{g_i = j} et_i$, $j = 1, \cdots, J$; $j' \leftarrow \arg\min_{j=1,\cdots,J}\{ld_j\}$;
$ST_{j'} \leftarrow \bigcup_{g_i = j'}(SC_i \cup PR_i) - \{i \mid g_i = j'\}$;
if $ST_{j'} = \varnothing$ **then**
  $ST_{j'} \leftarrow \{1, \cdots, I\} - \{i \mid g_i = j'\}$;
$i' \leftarrow \arg\max_{i \in ST_{j'}}\{ld_{g_i}\}$;
$ch' \leftarrow \{g'_1, \cdots, g'_{2I}\}$, where $g'_i = \begin{cases} g_i & i \neq i' \\ j' & i = i' \end{cases}$;
$Dcd(ch')$; $IFBSS(ch')$;
if $e'^{we} < e^{we}$ **then**
  $ch \leftarrow ch'$;

Algorithm 7: The TSEDA.

input: WM, RM;
output: the best individual/solution ch_{bt};
1: initialize MPM, SPM according to (18)–(19);
2: calculate ur_i according to (11)–(12); $mur \leftarrow \max\{ur_1, \cdots, ur_I\}$;
3: generate two individuals ch', ch'' according to the HMEC and the HEFT;
4: $ch_{bt} \leftarrow$ the best of ch', ch''; $Pop \leftarrow \varnothing$;
5: **while** the termination condition of stage 1 is not met **do**
6:   $\eta_i \leftarrow (ur_i / mur)^{\varphi(1-cn/tn)}$, $i = 1, \cdots, I$;
7:   **while** $|Pop| < N$ **do**
8:     $ch \leftarrow GnrTskLst(SPM, \{\eta_1, \ldots, \eta_I\})$;
9:     $HrsDcd(ch)$; $Pop \leftarrow Pop \cup \{ch\}$;
10:     **if** ch is better than ch_{bt} **then**
11:       $ch_{bt} \leftarrow ch$;
12:   update MPM and SPM according to ch_{bt}; $Pop \leftarrow \varnothing$;
13: **while** the termination condition of the TSEDA is not met **do**
14:   $\eta_i \leftarrow (ur_i / mur)^{\varphi(1-cn/tn)}$, $i = 1, \cdots, I$;
15:   **while** $|Pop| < N$ **do**
16:     $ch \leftarrow GnrTskLst(SPM, \{\eta_1, \ldots, \eta_I\})$; $GnrRscLst(ch, MPM)$;
17:     $Dcd(ch)$; $Pop \leftarrow Pop \cup \{ch\}$;
18:     **if** ch is better than ch_{bt} **then**
19:       $ch_{bt} \leftarrow ch$;
20:   **for** $ch \in$ top $\lceil Np_i \rceil$ individuals in Pop **do**
21:     $IFBSS(ch)$; $LBCAS(ch)$;
22:   update MPM and SPM according to ch_{bt}; $Pop \leftarrow \varnothing$;
23: return ch_{bt};

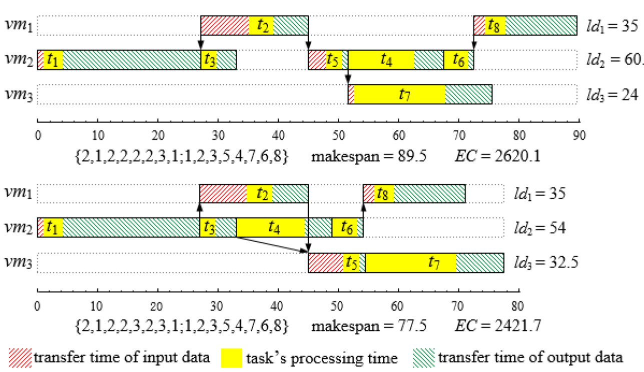

Fig. 4. The illustration of the LBCAS.

$\tau_4^f > \tau_6^f > \tau_2^f > \tau_5^f > \tau_7^f > \tau_8^f$ in the scheduling scheme shown in Fig. 3(b), and $\{2, 2, 2, 2, 2, 1, 3, 1; 1, 3, 4, 6, 2, 5, 7, 8\}$ is decoded by the FSIP to obtain its scheduling scheme shown in Fig. 3(c). Finally, since there is no further improvement in the EC, the IFBSS is terminated. Thus, its TSO is improved from $\{1, 2, 3, 5, 4, 7, 6, 8\}$ to $\{1, 3, 4, 6, 2, 5, 7, 8\}$, and correspondingly its makespan and EC are reduced from 93.5 and 2652 to 85 and 2483.7.

The LBCAS is designed to find a better TRM for improving the individual. The instinctive rationale behind the LBCAS is that balancing the load among VMs and avoiding communication (data/file transfer) by reassigning the tasks with direct dependencies to the same VM can shorten the execution time of the workflow, thereby optimizing other time-related performance of workflow execution (e.g., EC and cost). As shown in shown in Algorithm 6, the loads among VMs are balanced by reassigning a task to the VM with the least load. In addition, in order to avoid the communication between VMs as much as possible, when selecting a task to be reassigned, the preference is given to the parent or child tasks of the tasks assigned to the VM with the least load.

For the case in Section III.D, Fig. 4 illustrates how the LBCAS is performed on $\{2, 1, 2, 2, 2, 2, 3, 1; 1, 2, 3, 5, 4, 7, 6, 8\}$. First, the loads of the VMs are calculated and the VM with the lowest load (vm_3, namely $j' = 3$) is identified. Then, the parent or child tasks of the tasks assigned to vm_3 (t_5, namely $ST_3 = \{5\}$) are found. Finally, because there is only an element in ST_3, this task (t_5) is selected and reassigned to vm_3. Thus its TRM is improved from $\{2, 1, 2, 2, 2, 2, 3, 1\}$ to $\{2, 1, 2, 2, 3, 2, 3, 1\}$, correspondingly, the makespan and EC are reduced from 89.5 and 2620.1 to 77.5 and 2421.7.

D. Complete Algorithm

Now, the complete TSEDA shown in Algorithm 7 can be formed by combining all of the individual algorithms discussed above. First, the probability model (MPM and SPM) is initialized according to (18)–(19) and the best individual ch_{bt} is initialized as the best of two individuals generated according to the Heterogeneous Earliest-Finish-Time (HEFT) [24] and the Heterogeneous Minimum EC (HMEC) respectively (Lines 1–4), where the HMEC, like the HEFT, arranges the TSO in the descent order of ur_i, but it allocates the VMs according to the TSO by the TMEC instead of the heuristic method based on the task-earliest-finish-time used in the HEFT. Then, a two-stage coevolution strategy is employed. In stage 1 (Lines 5–12), the TSO of the individual is generated by sampling, after that the TRM of the individual is generated and the individual is decoded by the TMEC (Lines 8–9), which can make the TSEDA converge to the vicinity of the optimal solution as soon as possible. In stage 2 (Lines 13–22), both the TSO and TRM of the individual are generated by sampling, and the individual is decoded by the FSIP (Lines 16–17); moreover, the IFBSS and LBCAS are used to expand the neighborhood search for finding better individuals (Lines 20–21), where, N is the population size, and $p_i \in (0, 1)$ is the improvement rate. Finally, ch_{bt} is returned (Line 23).

The time complexity of the proposed TSEDA is analyzed as follows: The complexity of initializing MPM, SPM, and ch_{bt} (Lines 1–4) is $O(IJ) + O(I^3) + O(I^3J) \approx O(I^3J)$. The complexity of calculating $\eta_1, \ldots, \eta_I$ (Lines 6 and 14) is $O(I)$. The complexities of $GnrSchLst(SPM, \{\eta_1, \ldots, \eta_I\})$, $HrsDcd(p)$, $GnrRscLst(p, MPM)$, $Dcd(p)$, $IFBSS(p)$, $LBCAS(p)$ are $O(I^2)$, $O(I^3J)$, $O(IJ)$, $O(I^2)$, $O(I^2)$, $O(I^2)$. The complexity of updating probability model (Lines 12 and 22) is $O(IJ + I^2) \approx O(I^2)$. Thus, the complexity of the proposed TSEDA is evaluated as follows: $O(I^3J)+G_1 \cdot (O(I) + N \cdot O(I^2) + N \cdot O(I^3J) + O(I^2))+ G_2 \cdot (O(I) + 2N \cdot O(I^2) + N \cdot O(IJ) + 2Np_i \cdot O(I^2) + O(I^2)) \approx O(G_1NI^3J)$, where G_1 and G_2 are respectively the number of iterations/generations in stage 1 and stage 2.

V. Evaluation

In this section, the extensive experiments are conducted on various real and random workflow applications to verify the effectiveness of the proposed TSEDA by comparing it with the other algorithms in terms of the quality (workflow EC) of the solutions found by them. All algorithms are implemented in C++ and run on a personal computer with 3.2GHz∗4 CPU and 3.8G memory, and the source code and original experimental data (results) are available at https://github.com/TSEDA-C/TSEDA.

A. Compared Algorithms

Since the EC of cloud workflow execution in this study is estimated and optimized based on the relationships among scheduling scheme, host load and power rather than the DVFS-enabled environment or technique, very few research works have been done that deal with exactly the same problem as mentioned in this paper. However, there are still some studies that can be compared with this paper. The metrics to select an algorithm as a competitor from the list of papers in each category are as follows: 1) it considers the TRM and the TSO and can be applied to solve our problem; 2) it is detailed and accurate enough to be implemented for the comparison; 3) it is the latest published or has been widely used. According to these metrics, the HEFT [24], Hybrid GA (HGA) [43], New GA (NGA) [41], Level Workflow Scheduling based on GA (LWSGA) [18], Adaptive Decoding Biased Random Key GA (ADBRKGA) [51], cloud workflow scheduling approach combining PSO and idle time slot-aware rules (called HPSO) [48] are chosen as typical representatives of various algorithms to compare with our algorithm. The HEFT first arranges the TSO in the descent order of the upward ranks of tasks, then according to the TSO allocates the resource/processor that can minimize the earliest finish time of the selected task with the insertion policy, and it is the most classical and widely used HA. The HGA employs GA to evolve the TRM and employs an upward-ranking heuristic to decode/determine the TSO. The NGA employs GA to evolve the TSO and employs a list-based heuristic to decode/determine the TRM. Both the LWSGA and the ADBRKGA employ GA to evolve the TRM and the TSO, however, the LWSGA employs the HCT which limits the TSO to be changed only in the respective layers. The HPSO employs PSO to evolve the TRM and the TSO. These algorithms have achieved some valuable results in solving the workflow scheduling problem and are often used as comparative algorithms in current literature. It should be noted that: 1) unlike the original HPSO, the TRM in the HPSO in our experiment is modified to represent the VM instance rather than the type to fit the scheduling problem in this paper, because in our problem the resources can be determined in advance by some methods; 2) for an objective and fair evaluation of these algorithms, except that the optimization objectives and the individual evaluation/decoding of these algorithms are adapted for EC, the other operations (e.g., selection, crossover and mutation, particle update) are the same as those of these original algorithms.

B. Numerical Case Study

For the case in Section III.D, Table I lists the best solutions that can be found by the HEFT [24], HGA [43], NGA [41], LWSGA [18], ADBRKGA [51], HPSO [48], TSEDA and the Average scheduling/Running Time (ART) that they take to find their best solutions in the case. In fact, the best solutions found by the TSEDA, ADBRKGA, and HPSO are the optimal solution because their search spaces are complete. While the HGA, LWSGA, NGA and HEFT cannot find the optimal solution because it is not in their search spaces (or paths) in this case. Moreover, the proposed TSEDA can find the optimal solution in shorter time than the ADBRKGA and HPSO, because it has higher search efficiency and optimization ability in the complete search space.

C. Real-World Case Study

In order to further evaluate the performance of these algorithms, more extensive experiments are conducted on real-word cases as below.

1) Workflow Applications and Resource Configurations: For a fair and objective evaluation on different scheduling algorithms, the realistic workflow applications with all different types (C: CyberShake, E: Epigenomics, L: LIGO's Inspiral Analysis, M: Montage, S: SIPHT) and three sizes (S: about 30 tasks, M: about 50 tasks, L: about 100 tasks) are selected from the workflow library that has been widely used by researchers to measure the performance of workflow scheduling algorithms. These workflow applications come from different fields and have different structure, communication and computing characteristics [2]. They are available at https://confluence.pegasus.isi.edu/display/pegasus/Deprecated+Workflow+Generator from which the processing time of each task on an SCS, the input/output files of each task, the sizes of these input/output files, and the precedence relationships between tasks can be obtained. As a result, $5 \times 3 = 15$ experimental cases are used in our experiment. Let $cs^{ty,sz}$ be the experimental case whose workflow type is ty, workflow size is sz, where, $ty \in TY = \{C, E, L, M, S\}$, $sz \in SZ = \{S, M, L\}$, For example, $cs^{M,L}$ represents the experimental case whose workflow type is Montage, workflow size is about 100 tasks.

To reflect and simulate the heterogeneity of resources in cloud computing, there are three different types of hosts

TABLE I
THE BEST SOLUTIONS AND THE THEIR SCHEDULING TIME

Algorithm	t_1			t_2			t_3			t_4			t_5			t_6			t_7			t_8			EC (J)	ART (s)
	s_1	f_1	vid_1	s_2	f_2	vid_2	s_3	f_3	vid_3	s_4	f_4	vid_4	s_5	f_5	vid_5	s_6	f_6	vid_6	s_7	f_7	vid_7	s_8	f_8	vid_8		
HEFT	0	27	2	27	45	1	27	37	3	27	43	2	45	51.5	3	43	48.5	2	51.5	74.5	3	48.5	56	2	2387.10	0.00018
HGA	0	27	2	27	36	3	27	33	2	33	49	2	36	42.5	3	49	54	2	42.5	65.5	3	54	61.5	2	2205.95	0.05477
NGA	0	27	2	27	36	3	27	33	2	33	49	2	36	42.5	3	49	54	2	42.5	65.5	3	54	61.5	2	2205.95	0.00298
LWSGA	0	27	2	27	36	3	27	33	2	33	49	2	36	42.5	3	49	54	2	42.5	65.5	3	54	61.5	2	2205.95	0.09293
ADBRKGA	0	27	2	27	32	2	32	38	2	41.5	57.5	2	38	41.5	2	57.5	62.5	2	41.5	65.5	3	62.5	70	2	2199.20	0.01155
HPSO	0	27	3	27	32	3	32	38	3	41.5	57.5	3	38	41.5	3	57.5	62.5	3	41.5	65.5	2	62.5	70	3	2199.20	0.29322
TSEDA	0	27	2	27	32	2	32	38	2	41.5	57.5	2	38	41.5	2	57.5	62.5	2	41.5	65.5	3	62.5	70	2	2199.20	0.00197

TABLE II
THE POWERS OF HOSTS AT DIFFERENT LOADS

load		0	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
power (w)	ht_1	15.9	20.3	22.4	24.4	27.2	29.8	33.0	36.8	39.5	42.6	45.1
	ht_2	13.1	17.5	20.5	23.0	25.4	28.2	31.9	37.3	44.5	51.5	56.1
	ht_3	16.8	20.0	22.8	26.0	29.9	35.0	40.4	46.3	55.6	68.8	87.2

TABLE III
THE DEPLOYMENT OF VMS IN THE HOSTS

host	VM		
	S-type	M-type	L-type
hty_1	1	2	0
hty_2	1	1	1
hty_3	1	1	2

(hty_1: NEC Corporation Express5800/GT110f-S; hty_2: FUJITSU Server PRIMERGY TX1320 M3; and hty_3: Lenovo Global Technology ThinkSystem SR150) available for these workflow executions. The processing capacities of hty_1, hty_2, hty_3 are 10SCS, 14SCS, 22.2SCS respectively, and their load-power data is listed in Table II, which are available at http://www.spec.org/power_ssj2008/results/power_ssj2008.html. The power corresponding to the load not listed in Table II is calculated by linear interpolation. For example, when the load is 25%, the power of ht_1 is $22.4 + (24.4-22.4) * (25\%-20\%) / (30\%-20\%) = 23.4$W. Moreover, there are also three different types of VMs: S-type, M-type and L-type as the basic processing unit for processing workflow tasks. The processing capacities of S-type, M-type and L-type are 2000 MI/s (2SCS), 4000 MI/s (4SCS), and 8000 MI/s (8SCS), and the bandwidths of S-type, M-type and L-type are 2 Gbit/s, 4 Gbit/s and 8 Gbit/s. These three types of VMs are deployed in these hosts according to the following real-world rules: 1) Use the processing capacities of these hosts as much as possible, 2) The number of VMs of each type created should be balanced as much as possible. Thus the VMs are deployed in these hosts as shown in Table III.

2) Experiment and Result Analysis: Because the quality of the solution found by an MA is not only related to its own performance, but also related to its scheduling/Running Time (RT), in order to evaluate these MAs objectively and fairly, each of these MAs will run at the same RT (namely, process time of CPU) $rt^{ty,sz}$ for each $cs^{ty,sz}$. $rt^{ty,sz}$ is the maximum of $act^{ty,sz}_{HGA}$, $act^{ty,sz}_{NGA}$, $act^{ty,sz}_{LWSGA}$, $act^{ty,sz}_{ADBRKGA}$, $act^{ty,sz}_{HPSO}$, where $act^{ty,sz}_{HGA}$, $act^{ty,sz}_{NGA}$, $act^{ty,sz}_{LWSGA}$, $act^{ty,sz}_{ADBRKGA}$, $act^{ty,sz}_{HPSO}$ are the average convergence times of the HGA, NGA, LWSGA, ADBRKGA, HPSO. In our experiment the algorithm is considered to converge when the current best individual is still not improved after continuously searching for $20\sqrt{IJ}$ individuals.

In general, the parameter setting will affect the performance of an MA [51]. The parameters of the TSEDA include: the population size factor: $\varepsilon = N/I$, the update rates of probability model: θ_1 and θ_2, the factor of dynamic heuristic information: φ, the improvement rate: p_i, and the proportion of time the first stage runs: ψ. The orthogonal experimental design is a highly efficient, fast and economical method to study multi factors and multi levels, so it is employed to determine an optimal combination of parameter values for the proposed TSEDA, which are $\varepsilon = 1.8$, $\theta_1 = 0.35$, $\theta_2 = 0.25$, $\varphi = 0.8$, $p_i = 0.03$, $\psi = 0.75$. The HEFT is an HA where no parameters need to be set, while the parameter settings of the HGA, NGA, LWSGA, ADBRKGA and HPSO refer to their original papers.

First, all algorithms are evaluated in the environment with one host for each type (hty_1, hty_2, and hty_3) where 3 VMs of S-type, 4 VMs of M-type, and 3 VMs of L-type are available for these experimental cases. In such a given resource environment, for each $cs^{ty,sz}$, the workflow ECs of the solutions found by the HEFT and its RT in different cases are listed in Table IV. Since MAs are non-deterministic, for an objective and fair comparison, they run 100 times at the same RT $rt^{ty,sz}$ for each $cs^{ty,sz}$. In addition, to estimate the impact of the two-stage coevolution strategy on the overall performance, the TSEDA without it (called TSEDA-TS) also run 100 times at the same RT time $rt^{ty,sz}$. The averages of the workflow ECs of the solutions found by these MAs are calculated as shown in Table V. Due to the workflow EC varies greatly in different cases, to eliminate this effect, all the workflow ECs in the experiment are divided by the workflow ECs of the solutions found by the HEFT to obtain the relative workflow ECs. Figs. 5 and 6 respectively show the averages and boxplot of the relative workflow ECs of the solutions found by different algorithms in different cases in the given resource environment.

To estimate the specific degree of improvement of TSEDA, we also define the average improvement rate of the TSEDA over the compared algorithms as follows:

$$AIR_{***} = \frac{1}{|TY|\,|SZ|}\sum_{ty\in TY}\sum_{sz\in SZ}\frac{\overline{e}^{we}_{***}(ty,sz) - \overline{e}^{we}_{TSEDA}(ty,sz)}{\overline{e}^{we}_{***}(ty,sz)} \tag{22}$$

where, $\overline{e}^{we}_{TSEDA}(ty,sz)$ is the average workflow EC of the solutions found by the TSEDA for $cs^{ty,sz}$, and $\overline{e}^{we}_{***}(ty,sz)$ is the

TABLE IV
THE WORKFLOW ECS OF THE SOLUTIONS FOUND BY THE HEFT AND ITS RT IN DIFFERENT CASES IN THE GIVEN RESOURCE ENVIRONMENT

Case	$cs^{C,S}$	$cs^{C,M}$	$cs^{C,L}$	$cs^{E,S}$	$cs^{E,M}$	$cs^{E,L}$	$cs^{L,S}$	$cs^{L,M}$	$cs^{L,L}$	$cs^{M,S}$	$cs^{M,M,}$	$cs^{M,L}$	$cs^{S,S}$	$cs^{S,M,}$	$cs^{S,L}$
EC(J)	5950.65	7925.17	11955.72	73435.81	168802.16	1627624.70	28267.15	47310.51	85198.63	974.05	2083.41	4511.48	32484.40	48755.29	69483.88
RT(s)	0.00061	0.00102	0.00215	0.00042	0.00077	0.00161	0.00063	0.00101	0.00193	0.00058	0.00110	0.00225	0.00301	0.00653	0.01024

TABLE V
THE AVERAGES OF THE WORKFLOW ECS OF THE SOLUTIONS FOUND BY THE MAS IN THE GIVEN RESOURCE ENVIRONMENT

Case	EC(J)							RT(s)
	HGA	NGA	LWSGA	ADBRKGA	HPSO	TSEDA-TS	TSEDA	
$cs^{C,S}$	4822.48	5577.09	4821.87	5019.89	4582.20	4574.61	4442.69	1.389
$cs^{C,M}$	6885.38	7614.77	6917.28	7091.02	6844.84	6041.20	5964.93	2.334
$cs^{C,L}$	10733.38	11702.82	10740.99	10744.37	10659.06	8885.41	8774.38	7.944
$cs^{E,S}$	55481.72	57233.51	55855.30	55011.00	54902.35	55661.31	54812.66	0.818
$cs^{E,M}$	141507.31	154179.25	130370.39	127944.78	127682.96	126224.21	126082.59	2.599
$cs^{E,L}$	1456572.31	1538868.48	1315199.60	1226630.81	1299832.43	1223220.98	1222515.08	8.112
$cs^{L,S}$	21994.70	23797.88	22211.59	21424.50	20594.40	20256.56	19975.64	1.069
$cs^{L,M}$	42296.18	44356.95	42382.95	41971.97	38062.72	36092.33	35499.04	1.936
$cs^{L,L}$	78311.42	81287.37	79297.26	75874.45	77828.79	63646.29	63444.82	6.165
$cs^{M,S}$	771.09	800.09	786.26	720.81	720.81	706.22	705.69	0.794
$cs^{M,M}$	1907.63	1986.75	1956.30	1827.44	1844.15	1557.06	1556.25	1.805
$cs^{M,L}$	4165.96	4309.03	4318.70	4138.00	4163.29	3296.29	3294.76	5.258
$cs^{S,S}$	18637.51	23297.21	18342.13	18423.78	18410.30	18152.43	18152.43	1.438
$cs^{S,M}$	41149.49	38910.90	37094.66	36955.17	35494.42	35289.05	35268.57	4.940
$cs^{S,L}$	64349.09	58496.39	57242.76	55570.62	53899.81	52886.38	52439.90	15.177

TABLE VI
THE RESOURCES (HOSTS AND VMS) ARE CONFIGURED FOR EACH CASE

Case		$cs^{C,S}$	$cs^{C,M}$	$cs^{C,L}$	$cs^{E,S}$	$cs^{E,M}$	$cs^{E,L}$	$cs^{L,S}$	$cs^{L,M}$	$cs^{L,L}$	$cs^{M,S}$	$cs^{M,M,}$	$cs^{M,L}$	$cs^{S,S}$	$cs^{S,M,}$	$cs^{S,L}$
mpt		14	24	49	5	10	24	7	12	24	9	28	62	21	42	73
host	hty_1	5	8	17	2	4	8	3	4	8	3	10	21	7	14	25
	hty_2	5	8	17	2	4	8	3	4	8	3	10	21	7	14	25
	hty_3	5	8	17	2	4	8	3	4	8	3	10	21	7	14	25
VM	S-type	15	24	51	6	12	24	9	12	24	9	30	63	21	42	75
	M-type	20	32	68	8	16	32	12	16	32	12	40	84	28	56	100
	L-type	15	24	51	6	12	24	9	12	24	9	30	63	21	42	75

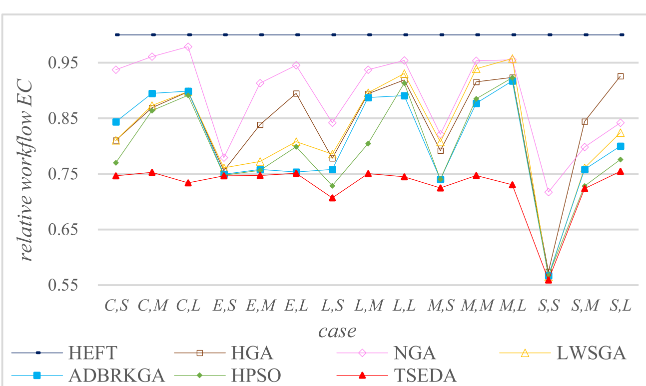

Fig. 5. The averages of the relative workflow ECs of the solutions found by different algorithms in different cases in the given resource environment.

average workflow EC of the solutions found by the compared algorithm (the HEFT, HGA, NGA, LWSGA, ADBRKGA, HPSO) for $cs^{ty,sz}$.

It can be seen from Tables IV, V and Figs. 5, 6 that: 1) The TSEDA can find better solutions than the HGA, NGA, LWSGA, ADBRKGA, HPSO at the same RT in all cases in the given resource environment. The average improvement rates of the TSEDA over the HGA, NGA, LWSGA, ADBRKGA, HPSO in all cases are 13.01%, 17.79%, 11.15%, 9.03%, 7.50%, respectively. 2) The MAs can usually find better solutions than the HAs (e.g., the HEFT) in the given resource environment. In particular, the TSEDA can find better solutions than the HEFT in all cases, and the average improvement rate of the TSEDA over the HEFT is 27.21%. 3) The TSEDA performs the best in terms of stability and robustness among all MAs in the given resource environment, and the standard deviation of the relative workflow ECs of the solutions found by the HGA, NGA, LWSGA, ADBRKGA, HPSO, TSEDA are 2.44%, 0.44%, 2.68%, 1.83%, 1.48%, and 0.02%, respectively. 4) The average improvement rates of the TSEDA over the TSEDA-TS ($AIR_{TSEDA-TS}$) are 0.77%, which shows that the two-stage coevolution strategy can improve the overall performance of the proposed TSEDA.

Furthermore, since MAs are random search techniques, we also calculate the probabilities of the proposed TSEDA outperforming the HEFT, HGA, NGA, LWSGA, ADBRKGA, HPSO in the given resource environment, which are 100.00%, 100.00%, 100.00%, 100.00%, 100.00%, 99.87%, respectively.

Moreover, in order to reflect the "infinite" nature of resources in cloud computing, all algorithms are also evaluated in the

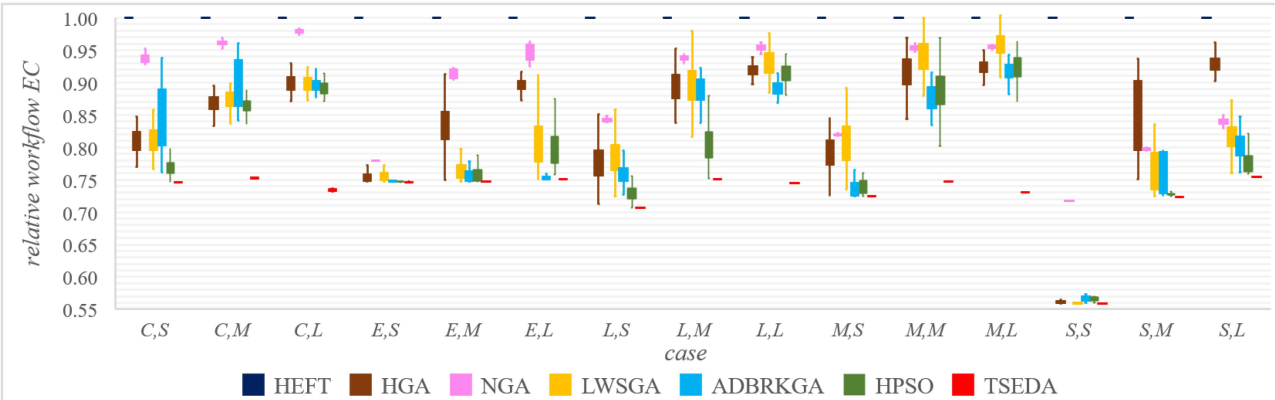

Fig. 6. The boxplot of the relative workflow ECs of the solutions found by different algorithms in different cases in the given resource environment.

TABLE VII
The Workflow ECs of the Solutions Found By the HEFT and Its RT in Different Cases in the Sufficient Resource Environment

Case	$cs^{C,S}$	$cs^{C,M}$	$cs^{C,L}$	$cs^{E,S}$	$cs^{E,M}$	$cs^{E,L}$	$cs^{L,S}$	$cs^{L,M}$	$cs^{L,L}$	$cs^{M,S}$	$cs^{M,M,}$	$cs^{M,L}$	$cs^{S,S}$	$cs^{S,M,}$	$cs^{S,L}$
EC(J)	5761.73	7543.82	11048.07	62784.57	147064.15	1358159.54	24313.71	42985.03	75873.66	798.84	1800.40	3842.95	38301.88	68940.54	86753.24
RT(s)	0.00111	0.00232	0.00741	0.00054	0.00127	0.00370	0.00096	0.00152	0.00402	0.00087	0.00276	0.00883	0.00386	0.00885	0.01599

TABLE VIII
The Averages of the Workflow ECs of the Solutions Found By the MAs in Different Cases in the Sufficient Resource Environment

Case	EC(J)						RT(s)
	HGA	NGA	LWSGA	ADBRKGA	HPSO	TSEDA	
$cs^{C,S}$	4764.39	5691.62	4860.29	5304.15	4804.42	4442.62	3.039
$cs^{C,M}$	6440.83	7466.33	6577.49	6715.23	6776.25	5964.25	16.348
$cs^{C,L}$	9432.30	10897.57	9652.88	9740.06	10489.84	8772.92	293.321
$cs^{E,S}$	56570.71	61340.48	56545.01	55665.23	55500.03	54812.40	1.177
$cs^{E,M}$	131743.68	138611.55	130083.79	129240.84	136689.64	126081.12	6.459
$cs^{E,L}$	1239152.07	1316658.15	1247706.50	1240817.11	1318190.02	1222511.43	126.112
$cs^{L,S}$	22040.56	23587.74	22309.50	21358.34	22937.94	19975.45	1.789
$cs^{L,M}$	38677.98	41373.91	38945.41	38688.35	40625.19	35497.51	8.966
$cs^{L,L}$	67733.63	73718.47	69162.73	72063.71	73948.69	63442.90	95.097
$cs^{M,S}$	769.70	760.45	826.10	786.02	780.18	705.67	1.113
$cs^{M,M}$	1725.56	1748.40	1812.32	1768.63	1788.22	1556.17	21.875
$cs^{M,L}$	3643.21	3777.48	3669.18	3808.61	3828.16	3293.14	344.133
$cs^{S,S}$	18541.69	28157.49	18448.29	18524.66	18344.29	18152.43	5.193
$cs^{S,M}$	37973.78	48389.27	37489.40	38003.61	39572.29	35268.16	41.285
$cs^{S,L}$	56393.93	68229.15	55270.77	54961.23	53999.27	52439.58	345.426

environment with sufficient/infinite resources available for workflow execution. First, similar to [21], [22], [36], [38], [39], etc., in order to maintain the integrity of all possible solutions and reduce the search space, the number of VMs of each type is set to a minimum value of not less than mpt, so the resources (hosts and VMs) for each case $cs^{ty,sz}$are configured as shown in Table VI. For example, there are 25 hosts of hty_1, 25 hosts of hty_2, and 25 hosts of hty_3 that create 250 VMs including 75 VMs of S-type, 100 VMs of M-type, and 75 VMs of L-type available for the case $cs^{S,L}$ because its mpt is 73. Then, in the sufficient resource environment, for each $cs^{ty,sz}$, the workflow ECs of the solutions found by the HEFT and its RT are listed in Table VII, and the averages of the workflow ECs of the solutions found by running these MAs 100 times at the same running time are shown in Table VIII.

It can be seen from Tables VII, VIII that the experimental results in the sufficient/infinite resource environment are consistent with those in the given resource environment. In particular, the TSEDA can still find better solutions than the HEFT, HGA, NGA, LWSGA, ADBRKGA, HPSO in all cases in the sufficient resource environment, the average improvement rates of the TSEDA over the HEFT, HGA, NGA, LWSGA, ADBRKGA, HPSO in all cases are 22.24%, 6.52%, 16.58%, 7.63%, 7.94%, 9.55%, respectively. The TSEDA still performs the best in terms of stability and robustness among all MAs in the sufficient resource environment, and the standard deviation of the relative workflow ECs of the solutions found by the HGA, NGA, LWSGA, ADBRKGA, HPSO, TSEDA are 1.23%, 0.22%, 2.39%, 1.69%, 2.23%, and 0.02%, respectively. The probabilities of the proposed TSEDA outperforming the HEFT, HGA, NGA, LWSGA, ADBRKGA, HPSO in the sufficient/infinite resource environment are all 100.00%.

In order to find better solutions, the RT of MAs is usually larger than that of HAs (e.g., the HEFT). However, the proposed

TSEDA can also find the same solution at almost the same RT as that of the HEFT or the HMEC because the best of two individuals generated according to the HEFT and the HMEC respectively is seeded into the population as the initial best individual.

The experimental results have shown that the proposed TSEDA outperforms the conventional approaches (e.g., the HEFT, HGA, NGA, LWSGA, ADBRKGA, HPSO).

VI. Conclusion

This article studies the problem of energy-aware cloud workflow scheduling. First, a model for estimating the EC of cloud workflow execution is presented based on the relationships among scheduling scheme, host load and power. Then, the TSEDA is developed for energy-aware cloud workflow scheduling, where some novel strategies and methods including two-stage coevolution strategy, individual generation and decoding based on heuristic information/method, the IFBSS and the LBCAS are designed to enhance its performance and efficiency. Finally, in simulation experiments with real-world data, the proposed TSEDA is evaluated by comparing it with conventional algorithms. The results show that the proposed TSEDA has robust and stable performance and outperforms these conventional approaches (e.g., the HEFT, HGA, NGA, LWSGA, ADBRKGA, HPSO).

The advantages and novelties of the TSEDA are summarized as follows: 1) Different from these existing methods based on DVFS, based on the relationships among scheduling scheme, host load and power, it can reduce/optimize EC without changing the frequency and voltage of VMs/hosts, which allows the proposed method for a wider scope of application. 2) Compared with these algorithms where the TSO is fixed in advance or one of the TSO and TRM is completely decoded/determined by a heuristic approach, the proposed TSEDA has a complete search space. 3) A two-stage coevolution strategy with some novel heuristic methods to generate, decode and improve individuals is designed in the proposed TSEDA, so that it can perform efficient search on the complete solution space.

This study aims to optimize the EC of cloud workflow execution, however, the proposed TSEDA is a high-level problem-independent algorithm in nature, which can be used to solve other optimization problems, even the multi-objective optimization problems with constraints, by adapting the individual decoding/evaluation, the heuristic methods (the TMEC, IFBSS, LBCAS, etc.) and incorporating the multi-objective optimization framework and the constraint handling strategy, thus further extending our method/algorithm to solve these problems can be done first for future research. Second, in our study the resource configuration (host and VM relationship) is fixed in advance, how to deploy the VMs on the hosts is another problem that is beyond the scope of this study, but it can also be done for future research, because the integration of resource configuration and workflow scheduling in cloud environment should produce a larger optimization space and achieve better optimization results. Third, this study focuses on the static workflow scheduling, where all information about a workflow is known before scheduling. However, this is not always true in real systems due to the uncertainty of workflow task arrival and the performance fluctuation of resources [51]. Therefore, dynamic scheduling that can handle these uncertainties is another problem worth investigating in our future. Fourth, to further improve the search efficiency of our algorithm, other effective strategies and schemes, such as multipopulation distributed coevolutionary strategy [59], [60], region encoding scheme [61], are worth studying in our future. Finally, this study focuses on how to make the algorithm better, however, other relevant theories, approaches and techniques, such as deeper theoretical analysis on the components of the evolutionary computation algorithms [62], the reducing problem difficulty [63], are worth studying in the future.

References

[1] Q. Wu, M. Zhou, Q. Zhu, Y. Xia, and J. Wen, “MOELS: Multiobjective evolutionary list scheduling for cloud workflows,” *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 1, pp. 166–176, Jan. 2020.

[2] Q. Wu, F. Ishikawa, Q. Zhu, Y. Xia, and J. Wen, “Deadline-constrained cost optimization approaches for workflow scheduling in clouds,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 12, pp. 3401–3412, Dec. 2017.

[3] I. Gupta, M. S. Kumar, and P. K. Jana, “Efficient Workflow Scheduling Algorithm for Cloud Computing System: A Dynamic Priority-Based Approach,” *Arab*,” *J. Sci. Eng.*, vol. 43, pp. 7945–7960, Dec. 2018.

[4] W. Ahmad, B. Alam, and A. Atman, “An energy-efficient Big Data workflow scheduling algorithm under budget constraints for heterogeneous cloud environment,” *J. Supercomput.*, vol. 77, no. 10, pp. 11946–11985, Oct. 2021.

[5] M. Safari and R. Khorsand, “Energy-aware scheduling algorithm for time-constrained workflow tasks in DVFS-enabled cloud environment,” *Simul. Model. Pract. Theory*, vol. 87, pp. 311–326, Sep. 2018.

[6] M. Masdari and M. Zangakani, “Efficient task and workflow scheduling in inter-cloud environments: Challenges and opportunities,” *J. Supercomput.*, vol. 76, no. 1, pp. 499–535, Jan. 2020.

[7] M. Masdari, F. Salehi, M. Jalali, and M. Bidaki, “A survey of PSO-based scheduling algorithms in cloud computing,” *J. Netw. Syst. Manag.*, vol. 25, no. 1, pp. 122–158, Jan. 2017.

[8] F. Wu, Q. Wu, and Y. Tan, “Workflow scheduling in cloud: A survey,” *J. Supercomput.*, vol. 71, no. 9, pp. 3373–3418, Sep. 2015.

[9] M. Hosseinzadeh, M. Y. Ghafour, H. K. Hama, B. Vo, and A. Khoshnevis, “Multi-objective task and workflow scheduling approaches in cloud computing: A comprehensive review,” *J. Grid Comput.*, vol. 18, no. 3, pp. 327–356, Sep. 2020.

[10] M. S. Kumar, I. Gupta, S. K. Panda, and P. K. Jana, “Granularity-based workflow scheduling algorithm for cloud computing,” *J. Supercomput.*, vol. 73, no. 12, pp. 5440–5464, Dec. 2017.

[11] G. L. Stavrinides and H. D. Karatza, “An energy-efficient, QoS-aware and cost-effective scheduling approach for real-time workflow applications in cloud computing systems utilizing DVFS and approximate computations,” *Future Gener. Comput. Syst.*, vol. 96, pp. 216–226, Jul. 2019.

[12] L. Zhang, L. Wang, Z. Wen, M. Xiao, and J. Man, “Minimizing energy consumption scheduling algorithm of workflows with cost budget constraint on heterogeneous cloud computing systems,” *IEEE Access*, vol. 8, pp. 205099–205110, Nov. 2020.

[13] Z. Wen et al., “Running industrial workflow applications in a software-defined multicloud environment using green energy aware scheduling algorithm,” *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5645–5656, Aug. 2021.

[14] R. Garg, M. Mittal, and L. H. Son, “Reliability and energy efficient workflow scheduling in cloud environment,” *Clust. Comput.*, vol. 22, no. 4, pp. 1283–1297, Dec. 2019.

[15] T. Wu, H. Gu, J. Zhou, T. Wei, X. Liu, and M. Chen, “Soft error-aware energy-efficient task scheduling for workflow applications in DVFS-enabled cloud,” *J. Syst. Archit.*, vol. 84, pp. 12–27, Mar. 2018.

[16] X. Li, W. Yu, R. Ruiz, and J. Zhu, “Energy-aware cloud workflow applications scheduling with geo-distributed data,” *IEEE Trans. Serv. Comput.*, vol. 15, no. 2, pp. 891–903, Mar./Apr. 2022.

[17] Z. Li, J. Ge, H. Hu, W. Song, H. Hu, and B. Luo, “Cost and energy aware scheduling algorithm for scientific workflows with deadline constraint in clouds,” *IEEE Trans Serv. Comput.*, vol. 11, no. 4, pp. 713–726, Jul./Aug. 2018.

[18] H. Khajemohammadi, A. Fanian, and T. A. Gulliver, “Efficient workflow scheduling for grid computing using a leveled multi-objective genetic algorithm,” *J. Grid Comput.*, vol. 12, no. 4, pp. 637–663, Dec. 2014.

[19] B. P. Rimal and M. Maier, “Workflow scheduling in multi-tenant cloud computing environments,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 1, pp. 290–304, Jan. 2017.

[20] Z. Sun, B. Zhang, C. Gu, R. Xie, B. Qian, and H. Huang, “ET2FA: A hybrid heuristic algorithm for deadline-constrained workflow scheduling in cloud,” *IEEE Trans Serv. Comput.*, vol. vol. 16, no. 3, pp. 1807–1821, Aug. 2023, doi: 10.1109/TSC.2022.3196620.

[21] N. Rizvi, R. Dharavath, L. Wang, and A. Basava, “A workflow scheduling approach with modified fuzzy adaptive genetic algorithm in IaaS clouds,” *IEEE Trans Serv. Comput.*, vol. 16, no. 2, pp. 872–885, Mar./Apr. 2023, doi: 10.1109/TSC.2022.3174112.

[22] W. Guo, B. Lin, G. Chen, Y. Chen, and F. Liang, “Cost-driven scheduling for deadline-based workflow across multiple clouds,” *IEEE Trans. Netw. Serv. Manag.*, vol. 15, no. 4, pp. 1571–1585, Dec. 2018.

[23] Z. Zhu, G. Zhang, M. Li, and X. Liu, “Evolutionary multi-objective workflow scheduling in cloud,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 5, pp. 1344–1357, May 2016.

[24] H. Topcuoglu, S. Hariri, and M. Wu, “Performance-effective and low-complexity task scheduling for heterogeneous computing,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260–274, Mar. 2002.

[25] V. Arabnejad, K. Bubendorfer, and B. Ng, “Budget and deadline aware e-science workflow scheduling in clouds,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 1, pp. 29–44, Jan. 2019.

[26] H. R. Faragardi, M. R. Sedghpour, S. Fazliahmadi, T. Fahringer, and N. Rasouli, “GRP-HEFT: A budget-constrained resource provisioning scheme for workflow scheduling in IaaS clouds,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 6, pp. 1239–1254, Jun. 2020.

[27] J. Sahni and D. P. Vidyarthi, “A cost-effective deadline-constrained dynamic scheduling algorithm for scientific workflows in a cloud environment,” *IEEE Trans. Cloud Comput.*, vol. 6, no. 1, pp. 2–18, First Quarter 2018.

[28] R. Ghafouri, A. Movaghar, and M. Mohsenzadeh, “A budget constrained scheduling algorithm for executing workflow application in infrastructure as a service clouds,” *Peer-to-Peer Netw. Appl.*, vol. 12, no. 1, pp. 241–268, Jan. 2019.

[29] M. A. Rodriguez and R. Buyya, “Scheduling dynamic workloads in multi-tenant scientific workflow as a service platforms,” *Future Gener. Comput. Syst.*, vol. 79, pp. 739–750, Feb. 2018.

[30] M. A. Rodriguez and R. Buyya, “Budget-driven scheduling of scientific workflows in IaaS clouds with fine-grained billing periods,” *ACM Trans. Auton. Adapt. Syst.*, vol. 12, no. 2, May 2017, Art. no. 5.

[31] G. Yao, X. Li, Q. Ren, and R. Ruiz, “Failure-aware elastic cloud workflow scheduling,” *IEEE Trans Serv. Comput.*, vol. 16, no. 3. pp. 1846–1859, May/Jun. 2023, doi: 10.1109/TSC.2022.3188414.

[32] B. Hu, Z. Cao, and M. Zhou, “Energy-minimized scheduling of real-time parallel workflows on heterogeneous distributed computing systems,” *IEEE Trans. Serv. Comput.*, vol. 15, no. 5, pp. 2766–2779, Sep./Oct. 2022.

[33] Y. Wen, H. Xu, and J. Yang, “A heuristic-based hybrid genetic-variable neighborhood search algorithm for task scheduling in heterogeneous multiprocessor system,” *Inform. Sci.*, vol. 181, no. 3, pp. 567–581, Feb. 2011.

[34] Y. Xu, K. Li, J. Hu, and K. Li, “A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues,” *Inform. Sci.*, vol. 270, pp. 255–287, Jun. 2014.

[35] Y. Xu, K. Li, L. He, L. Zhang, and K. Li, “A hybrid chemical reaction optimization scheme for task scheduling on heterogeneous computing systems,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 12, pp. 3208–3222, Dec. 2015.

[36] 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, Second Quarter 2014.

[37] Z. J. Wang et al., “Dynamic group learning distributed particle swarm optimization for large-scale optimization and its application in cloud workflow scheduling,” *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2715–2729, Jun. 2020.

[38] Y. H. Jia et al., “An intelligent cloud workflow scheduling system with time estimation and adaptive ant colony optimization,” *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 51, no. 1, pp. 634–649, Jan. 2021.

[39] 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.

[40] Q. Xiao, J. Zhong, L. Feng, L. Luo, and J. Lv, “A cooperative coevolution hyper-heuristic framework for workflow scheduling problem,” *IEEE Trans. Serv. Comput.*, vol. 15, no. 1, pp. 150–163, Jan./Feb. 2022.

[41] B. Keshanchi, A. Souri, and N. J. Navimipour, “An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: Formal verification, simulation, and statistical testing,” *J. Syst. Softw.*, vol. 124, pp. 1–21, Feb. 2017.

[42] M. H. Shirvani, “A hybrid meta-heuristic algorithm for scientific workflow scheduling in heterogeneous distributed computing systems,” *Eng. Appl. Artif. Intell.*, vol. 90, Apr. 2020, Art. no. 103501.

[43] S. G. Ahmad, C. S. Liew, E. U. Munir, A. T. Fong, and S. U. Khan, “A hybrid genetic algorithm for optimization of scheduling workflow applications in heterogeneous computing systems,” *J. Parallel Distrib. Comput.*, vol. 87, pp. 80–90, Jan. 2016.

[44] M. Akbari, H. Rashidi, and S. H. Alizadeh, “An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems,” *Eng. Appl. Artif. Intell.*, vol. 61, pp. 35–46, May 2017.

[45] A. Mahmood and S. A. Khan, “Hard real-time task scheduling in cloud computing using an adaptive genetic algorithm,” *Computers*, vol. 6, no. 2, Apr. 2017, Art. no. 15.

[46] X. Ma, H. Gao, H. Xu, and M. Bian, “An IoT-based task scheduling optimization scheme considering the deadline and cost-aware scientific workflow for cloud computing,” *EURASIP J. Wirel. Commun. Netw.*, Nov. 2019, Art. no. 249.

[47] H. Hu et al., “Multi-objective scheduling for scientific workflow in multicloud environment,” *J. Netw. Comput. Appl.*, vol. 114, pp. 108–122, Jul. 2018.

[48] Y. Wang and X. Zuo, “An effective cloud workflow scheduling approach combining PSO and idle time slot-aware rules,” *IEEE/CAA J. Autom. Sinica*, vol. 8, no. 5, pp. 1079–1094, May 2021.

[49] L. Wang, H. J. Siegel, V. P. Roychowdhury, and A. A. Maciejewski, “Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach,” *J. Parallel Distrib. Comput.*, vol. 47, no. 1, pp. 8–22, Nov. 1997.

[50] Y. Xie, F. X. Gui, W. J. Wang, and C. F. Chien, “A two-stage multi-population genetic algorithm with heuristics for workflow scheduling in heterogeneous distributed computing environments,” *IEEE Trans. Cloud Comput.*, vol. 11, no. 2, pp. 1446–1460, Second Quarter 2023, doi: 10.1109/TCC.2021.3137881.

[51] Y. Xie, Y. Sheng, M. Qiu, and F. Gui, “An adaptive decoding biased random key genetic algorithm for cloud workflow scheduling,” *Eng. Appl. Artif. Intell.*, vol. 112, Jun. 2022, Art. no. 104879.

[52] Y. Zhou, J. Miao, B. Yan, and Z. Zhang, “Stochastic resource-constrained project scheduling problem with time varying weather conditions and an improved estimation of distribution algorithm,” *Comput. Ind. Eng.*, vol. 157, Jul. 2021, Art. no. 107322.

[53] Y. Du, J. Q. Li, C. Luo, and L. L. Meng, “A hybrid estimation of distribution algorithm for distributed flexible job shop scheduling with crane transportations,” *Swarm Evol. Comput.*, vol. 62, Apr. 2021, Art. no. 100861.

[54] S. Wang, L. Wang, M. Liu, and Y. Xu, “A hybrid estimation of distribution algorithm for the semiconductor final testing scheduling problem,” *J. Intell. Manuf.*, vol. 26, pp. 861–871, Oct. 2015.

[55] A. Utamima, “A comparative study of hybrid estimation distribution algorithms in solving the facility layout problem,” *Egyptian Informat. J.*, vol. 22, no. 4, pp. 505–513, Dec. 2021.

[56] R. Pérez-Rodríguez and A. Hernández-Aguirre, “A hybrid estimation of distribution algorithm for the vehicle routing problem with time windows,” *Comput. Ind. Eng.*, vol. 130, pp. 75–96, Apr. 2019.

[57] J. Y. Li, Z. H. Zhan, J. Xu, S. Kwong, and J. Zhang, “Surrogate-assisted hybrid-model estimation of distribution algorithm for mixed-variable hyperparameters optimization in convolutional neural networks,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 5, pp. 2338–2352, May 2023.

[58] M. A. Rodriguez and R. Buyya, “A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments,” *Concurrency Computat.: Pract. Exper.*, vol. 29, no. 8, Apr. 2017, Art. no. e4041.

[59] 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, doi: 10.1109/TEVC.2018.2875430.

[60] Z. H. Zhan, Z. J. Wang, H. Jin, and J. Zhang, “Adaptive distributed differential evolution,” *IEEE Trans. Cybern.*, vol. 50, no. 11, pp. 4633–4647, Nov. 2020, doi: 10.1109/TCYB.2019.2944873.

[61] J. R. Jian, Z. G. Chen, Z. H. Zhan, and J. Zhang, "Region encoding helps evolutionary computation evolve faster: A new solution encoding scheme in particle swarm for large-scale optimization," *IEEE Trans. Evol. Comput.*, vol. 25, no. 4, pp. 779–793, Aug. 2021, doi: 10.1109/TEVC.2021.3065659.
[62] J. Y. Li, Z. H. Zhan, and J. Zhang, "Evolutionary computation for expensive optimization: A survey," *Mach. Intell. Res.*, vol. 19, no. 1, pp. 3–23, Jan. 2022.
[63] Z. H. Zhan, L. Shi, K. C. Tan, and J. Zhang, "A survey on evolutionary computation for complex continuous optimization," *Artif. Intell. Rev.*, vol. 55, pp. 59–110, Jan. 2022.

Yi Xie received MS and PhD degrees in industrial engineering from Zhejiang University, in 2003 and 2011, respectively. He is a professor with Zhejiang Gongshang University. He is the author of one book, more than 20 articles and 10 inventions. His research interests include intelligent computing, and scheduling optimization.

Xue-Yi Wang received BS degree in information management and information system from the Shandong University of Finance and Economics, in 2022. She is currently working toward the MS degree with Zhejiang Gongshang University. Her research interests include intelligent computing, scheduling optimization.

Zi-Jun Shen received BS degree in E-commerce from Zhejiang Gongshang University in 2021. He is currently working toward the MS degree with Zhejiang Gongshang University. His research interests include intelligent computing, and scheduling optimization.

Yu-Han Sheng received BS degree in information management and information system from Zhejiang Gongshang University, in 2022. He is currently working toward the MS degree with Zhejiang Gongshang University. His research interests include intelligent computing and scheduling optimization.

Gong-Xing Wu received MS degrees in computer application technology from Hohai University, in 2003 and received PhD degrees in Statistics from Zhejiang Gongshang University in 2022. He is an associate professor in Zhejiang Gongshang University. He is the author of one book and more than 10 articles. His research interests include business Big Data analysis and software engineering.