IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 17 July 2023, accepted 6 September 2023, date of publication 13 September 2023, date of current version 20 September 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3314735*

RESEARCH ARTICLE

# Hybrid Teaching–Learning-Based Optimization for Workflow Scheduling in Cloud Environment

**JIEGUANG HE AND XIAOLI LIU**
College of Computer Science, Guangdong University of Petrochemical Technology, Maoming 525011, China
Corresponding author: Xiaoli Liu (lilianlaugdupt@163.com)

This work was supported in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2020A1515010727 and Grant 2023A1515011913.

**ABSTRACT** At present, workflow scheduling in cloud computing environment is still a challenging optimization topic due to its NP-complete characteristics. In order to obtain better scheduling results, researchers are constantly coming up with new methods. In this study, we offer a hybrid metaheuristic for solving workflow scheduling in cloud to minimize the makespan of the workflow considering the heterogeneity of virtual resources. This hybrid approach combines the excellent optimization properties of Heterogeneous Earliest Finish Time (HEFT), Teaching–Learning-Based Optimization (TLBO), Opposition-Based Learning (OBL), and genetic manipulations, which is named Hybrid TLBO (HTLBO). Firstly, a HEFT-based method is proposed to produce the high-quality diverse initial population. Secondly, a Mixed OBL (MOBL) model is designed, in which the boundary search information and the population historical search information are systematically taken into account. Finally, an enhanced learner stage using genetic operations are added to effectively help the algorithm to jump out of the local optima. Rigorous experiments over various scientific workflows are conducted to validate HTLBO's performance. The obtained results are compared to HEFT and some state-of-the-art hybrid metaheuristics in terms of average makespan, running time and non-parametric statistics. A significant improvement in schedule quality demonstrates that HTLBO can increase population diversity and achieve a good balance between scheduling effectiveness and efficiency.

**INDEX TERMS** Workflow scheduling, cloud computing, teaching–learning-based optimization, opposition-based learning, search boundary, population information.

## I. INTRODUCTION

Workflow technologies are emerging to handle complex, data-intensive experimentations and simulations that are run by computational scientists. Usually, such scientific workflows are composed of a huge number of tasks which have dependencies among each other and produce large-scale data while executed. Apparently, the execution of a workflow demands a lot of resources for computation, communication and data storage of its tasks. Cloud computing systems can provide such an infrastructure to run the workflow through a suite of distributed virtualized resources interconnected by high-speed networks.

The associate editor coordinating the review of this manuscript and approving it for publication was Nitin Gupta.

Workflow scheduling in cloud computing systems has gained extensive attention in both academia and industry because the results of scheduling can greatly influence the performance of computing environment. Scheduling is the process of determining the priority of the tasks and mapping the tasks to the available virtual machines properly, so as to minimize the total execution time of the computation. In fact, the workflow scheduling has been proven to be an NP-complete problem [1]. Currently, hybridization of multiple heuristic algorithms demonstrates much power to produce better results than a single metaheuristic or traditional list-based heuristics [2], [3].

In this paper, we propose a hybrid algorithm, named as Hybrid Teaching–Learning-Based Optimization (HTLBO), which hybridizes Heterogeneous Earliest Finish Time

(HEFT) [4], Opposition-Based Learning (OBL) [5], genetic operations and Teaching–Learning-Based Optimization (TLBO) [6] for workflow scheduling problem in cloud computing systems. The major contributions of this study are listed below.

- We develop the HTLBO algorithm, which can be applied to a wide range of workflow applications. The proposed algorithm can achieve a good balance between effectiveness (makespan and convergence speed) and efficiency (running time).
- We propose a new HEFT-based approach to generate the initial population, which is divided into two parts, including high-quality solutions generated by HEFT and quality-based crossover operation, and randomly generated solutions. A small group of high-quality solutions can help HTLBO find better solutions faster than it can from the population composed of pure solutions generated randomly. On the other side, the remaining random solutions can keep a good diversity of the population and help HTLBO to reach part of the feasible search space as large as possible.
- We design a Mixed Opposition-Based Learning (MOBL) modal that combines boundary search information and historical population search information.
- Besides the teacher phase and learner phase of the original TLBO, we add a new phase of enhanced learning to HTLBO, which is implemented using genetic operations. Moreover, combining the MOBL modal with these three stages and considering the different characteristics of each stage, we design three different calculation methods of opposition-based solutions according to mean individual, randomly selected individual, and best individual, respectively.
- We demonstrate the better performance of the algorithm through comprehensive experiments and comparisons with state-of-the-art heuristic and hybrid approaches.

The rest of the paper is organized as follows.

Section II reviews the related workflow scheduling algorithms and techniques used in this study. The workflow scheduling problem in cloud computing environment is briefly presented in Section III. The original TLBO and the proposed HTLBO are described in Sections IV and V, respectively. Section VI elaborates on the experimental results and their discussion. Finally, the conclusion and some future works are discussed in Section VII.

## II. RELATED WORK

### A. OPTIMIZATION METHODS FOR SOLVING WORKFLOW SCHEDULING IN CLOUD COMPUTING ENVIRONMENT

Number of heuristics and evolutionary algorithms have been developed to address the workflow scheduling problems. In this paper, we classify these algorithms into three categories: heuristic-based methods, metaheuristics, and hybrid metaheuristics.

#### 1) HEURISTIC-BASED METHODS

The heuristic-based algorithms mostly belong to the list scheduling class, and the list scheduling algorithms, in fact, are based on priority schemes. This kind of algorithms consists of two stages. The first stage is to establish an ordered list of tasks by the priorities calculated through certain criteria, and the second stage is to assign each task according to its pre-defined order on a proper resource that can finish it as quickly as possible. Some well-known list scheduling heuristics are Heterogeneous Earliest Finish Time (HEFT) [4], Critical Path on Processor (CPOP) [4], Performance Effective Task Scheduling (PETS) [7], Predict Earliest Finish Time (PEFT) [8], Standard Deviation Based Task Scheduling (SDBATS) [9], Heterogeneous Dynamic List Task Scheduling (HDLTS) [10], and efficient priority and relative distance (EPRD) [11]. On the other side, some Quality-of-Service (QoS) based heuristic algorithms for workflow scheduling in cloud environment have been proposed in recent years. This type of algorithms focuses on optimizing several performance metrics or optimizing a performance metric while satisfying some QoS constraints. Zhu and Tang [12] developed a list-scheduling framework for scheduling multi-resource workflow and proposed a deadline-constrained scheduling algorithm based on the framework to minimize the execution cost. Amoon et al. [13] proposed a workflow task scheduling method, which is known as Improved Cost Task Scheduling (ICTS) and includes three stages: level sorting, task-prioritizing and virtual machine selection, to minimize workflow execution time and cost. Zhou et al. [14] developed a fuzzy dominance sort based HEFT (FDHEFT) algorithm to jointly optimize the cost and time of cloud workflow scheduling. Faragardi et al. [15] proposed a greedy resource provisioning and improved HEFT method GRP-HEFT for the hourly-based cost model in IaaS cloud to optimize workflow execution time under budget constraint. Quan et al. [16] studied the scheduling problem of energy-constrained parallel applications on heterogeneous computing systems, and proposed a weight-based mechanism to pre-allocate the energy consumption of unassigned tasks to minimize the scheduling length or maximize reliability under the energy consumption or/and deadline constraints. The greatest advantage of these approaches is that their complexity is basically less than any other scheduling metaheuristics, thus having good efficiency and practicability. And for small-scale problems, namely smaller search space, they also possess good performance, because the solutions got by them are almost optimal or near-optimal. However, for medium or large-scale problems whose search space increases dramatically, the quality of the solutions which just are feasible in most cases declines sharply because the effectiveness of the heuristic-based list scheduling algorithms heavily relies on the heuristic rules used. Hence, they are not likely to produce satisfactory results for all kinds of scheduling situations.

#### 2) METAHEURISTICS

As for metaheuristics, there are a lot of methodologies, mainly focusing on different evolutionary strategies, to solve the workflow scheduling problems. Some traditional and brand-new algorithms of these types include Genetic

Algorithms (GAs) [17], [18], Particle Swarm Optimization (PSO) [19], [20], Ant Colony Optimization (ACO) [21], [22], Artificial Bee Colony (ABC) algorithm [23], [24], Cuckoo Search (CS) algorithm [25], Chemical Reaction Optimization (CRO) [26], Whale Optimization Algorithm (WOA) [27], and Grey Wolf Optimizer (GWO) [28], etc. These techniques start from an initial population, and then generate next population iteratively by some individual evolutionary operations that combine the information from previous search results and some randomizing strategies. Metaheuristics can generally get high quality solutions due to the abilities of searching the problem space as widely as possible; however, their scheduling times are inevitably much higher than those of the heuristic-based algorithms. Additionally, almost all metaheuristics have more or less disadvantages related to their own characteristics, such as bad global or local search ability, apt to fall into local optima, slow convergence speed, complicated computation formula, and too many control parameters to adjust for different problems.

#### 3) HYBRID METAHEURISTICS

Considering the merits and demerits of a single list-based heuristic or metaheuristic, more and more researchers have proposed hybrid metaheuristics for tackling workflow scheduling in cloud environments.

Among the others, GAs have received more attention for hybridization due to their widespread use in antecedent scheduling problems, e.g., project scheduling, job/flow shop scheduling, routine scheduling, and vehicle scheduling. Daoud and Kharma addressed the task scheduling problem by utilization of a Longest Dynamic Critical Path (LDCP) heuristic and a GA for scheduling (GAS), and the combination is called Hybrid Heuristic-Genetic Scheduling (H2GS) [29]. First the LDCP generate a high quality schedule that will be injected into the initial population of the GAS, and then the GAS proceeds to evolve shorter schedules. Wang et al. [30] proposed a Hybrid Successor Concerned Heuristic-Genetic Scheduling (HSCGS) algorithm which incorporates a Successor Concerned List Heuristic Scheduling (SCLS) into a GA. The SCLS generates a high quality schedule using the up-ward rank that concerns the impact of a task's successor, and then the GA incorporate this schedule to produce new generations. In [2], a resource-load balancing hybrid algorithm, named as Hybrid Genetic Algorithm (HGA), was proposed to schedule the workflows, where the HEFT heuristic is integrated with a GA by the way of hybridization similar to those of H2GS and HSCGS. Recently, another hybridization of GA and HEFT (HEFTGA) was proposed in [31], which was used to optimize both makespan and cost of workflows.

Besides the GAs, PSO is another algorithm that gains much popularity to be treated as main method while mixed. In a QoS-based hybrid PSO scheme (GHPSO) [32], Xue et al. introduced a GA and a hill climbing algorithm into the PSO to improve the diversity of the population and the local search ability. Verma and Kaushal [33] presented a non-dominance sort based Hybrid Particle Swarm Optimization (HPSO) algorithm to handle the workflow scheduling problem with two conflicting objectives of makespan and cost. Mansouri et al. [34] proposed a hybrid task scheduling algorithm named FMPSO mixing Fuzzy system and Modified Particle Swarm Optimization technique to enhance load balancing and cloud throughput. By taking advantage of both PSO and GWO, Arora and Banyal [35] proposed a hybrid PSO-GWO to reduce the total executing cost and total execution time for workflow scheduling.

Some other up-to-date outstanding hybrids that are used for workflow-concerned scheduling are briefly expressed as follows. Xu et al. [36] proposed a Hybrid Chemical Reaction Optimization (HCRO), in which some novel heuristic approaches and a new selection strategy are integrated with the original CRO, to solve the DAG-based task scheduling problem. Vasile et al. [37] proposed a resource-aware hybrid scheduling algorithm for different types of application: batch jobs and workflows, where tasks are first assigned to groups of resources and then a classical scheduling algorithm is used to schedule them using the allocated group resources. Abdullahi and Ngadi [38] merged Symbiotic Organisms Search (SOS) and SA to form a hybrid algorithm SASOS. This approach uses SOS for covering global exploration and SA for finding better solutions on local solution regions. With the combination of SA and Taguchi method in Cat Swarm Optimization (CSO), a Cloud Scalable Multi-Objective Cat Swarm Optimization Based Simulated Annealing (CSM-CSOSA) algorithm for scheduling tasks in cloud datacenter for the purpose of ensuring consumers QoS expectations was proposed by Gabi et al. [39]. Choudhary et al. [40] proposed a GSA based hybrid algorithm for bi-objective workflow scheduling in cloud computing that is a hybridization of Gravitational Search Algorithm (GSA) and HEFT. Mohammadzadeh et al. [41] developed a hybridization of the Ant Lion Optimizer (ALO) algorithm with a Sine Cosine Algorithm (SCA) algorithm and used it multi-objectively to solve the problem of scheduling scientific workflows.

These hybrid methods are able to utilize the advantages of one or more auxiliary algorithms to make up for the shortcomings of the main algorithm. Specifically, they can be further divided into two categories. One class is realized by combining a metaheuristic algorithm with some heuristic rules, such as HEFT, SCLS, LDCP, Fuzzy system. This kind of hybrid can achieve a good balance between scheduling quality and scheduling efficiency, and usually introduce few additional parameters. However, because heuristic rules are only combined with certain stages of metaheuristic algorithms, their performance improvement to hybrid algorithms is limited. The other class is achieved by combining multiple metaheuristics. Due to the full play of the advantages of multiple metaheuristic algorithms, this hybridization can obtain the best solution results, but it consumes a lot of time and usually introduces a large number of control parameters.

### B. TEACHING–LEARNING-BASED OPTIMIZATION AND OPPOSITION-BASED LEARNING

Since the HTLBO proposed in this paper has incorporated the methods of TLBO and OBL, some brief reviews of them are addressed as below.

TLBO, proposed by Rao et al. [6], is a kind of population-based optimization algorithm inspired by the principle of improving students' grades through teachers teaching and students learning procedure. Owning to its virtues in no needing any algorithm-specific tuning parameters with respect to other metaheuristics, TLBO and its variants have been broadly applied in function optimization [42], [43], [44], flow/job shop scheduling [45], [46], [47], [48], engineering design optimization [49], [50], thermoelectric cooler optimization [51], quadratic assignment problem [52], workflow scheduling [53] and some other fields.

OBL is a new technology proposed in the field of computing intelligence in recent years. It is primarily used to improve the global search ability of an algorithm. Currently, the hybridization of OBL in TLBO is found in the following applications. Mukhopadhyay et al. [54] integrated OBL with TLBO to form an Opposition Teaching–Learning-Based Optimization (OTLBO) and used it to solve the optimal power flow problem. Shao et al. [55] proposed an extended teaching–learning based optimization algorithm to solve the no-wait flow shop scheduling problem, where OBL is added in the previewing stage before class to improve the diversity of the population. Xu et al. [56] proposed a novel TLBO variant named dynamic-opposite learning TLBO (DOLTLBO), which employs a new Dynamic-Opposite Learning (DOL) strategy to overcome premature convergence. Liu et al. [57] proposed a hybrid metaheuristic teaching–learning-based optimization (HTLBO) for the travel route optimization problem alongside the urban railway line, where OBL is embedded for enhancing HTLBO's exploration ability.

Based on the above-mentioned references, it can be found that:

(1) Many hybrids integrate HEFT to enhance the convergent performance of the original algorithms by seeding a HEFT-generated individual into the initial population. Nevertheless, the effectiveness of this kind of hybridization is not obvious since only one high-quality solution is added into the initial population.

(2) The lack of good balance between global exploration and local exploitation is still a key problem faced by TLBO and its variants. In addition, there are only a few TLBO-based algorithms used to solve the cloud workflow scheduling problems. The high dimensionality in workflow scheduling makes it easy for these methods to fall into local optima.

(3) As for OBL, most of the current researches have employed the boundary values of the search space to calculate the opposition individuals, which can enhance the diversity of the population effectively, but in the later stage of evolution, it is not conductive to the local search, thus reducing the convergence speed and accuracy.

Hence, in this paper, we propose a new hybrid algorithm HTLBO to solve the workflow scheduling problem in cloud computing systems considering the heterogeneity of virtual resources. The new method has the characteristics of (1) absorbing the merits of TLBO, OBL and HEFT; (2) just needing to set the values of two parameters, namely proportion of high-quality individuals in initial population and mutation probability in genetic operation; (3) designing a mixed opposition-based learning model combining boundary information and intra-population information; (4) achieving a good balance between effectiveness and efficiency of the scheduling.

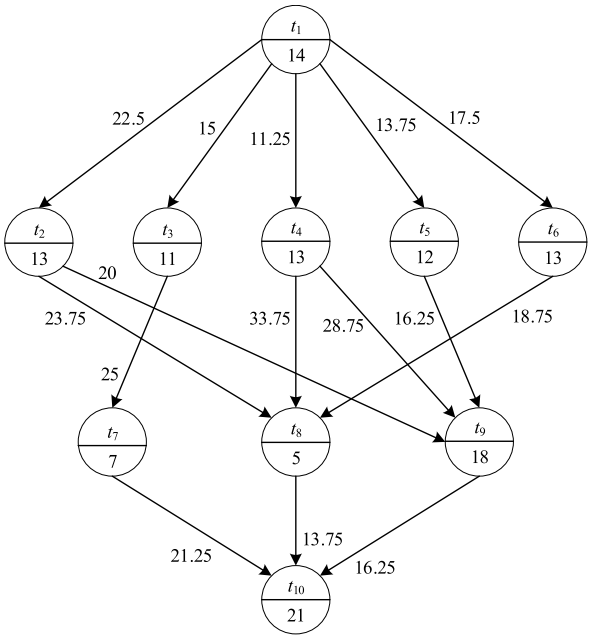


**FIGURE 1.** Workflow sample.

## III. PROBLEM STATEMENT

A workflow application can be represented in the form of a Directed Acyclic Graph (DAG) [40], $W = (T, E)$ as shown in Fig. 1 [4], where $T = \{t_1, t_2, \ldots, t_n\}$ is the set of tasks and $E$ is the set of edges. An edge is denoted as a pair of tasks $(t_i, t_j)$ which illustrates the precedence restraint between the predecessor $t_i$ and the successor $t_j$. That is to say, task $t_i$ must finish before task $t_j$ can start. In a DAG, a task without any predecessor is called an entry task and a task without any successor is called an exit task. In reality, some workflows may have multiple entry tasks and/or multiple exit tasks.

Cloud computing system comprises of a set $VM = \{vm_1, vm_2, \ldots, vm_m\}$ of $m$ independent different types of virtual machine (VM), namely with different processing capacity (CPU, memory, bandwidth), fully interconnected by a high-speed network. The computation capacity of $vm_k$ is presented as $C_{comp}(vm_k)$. The bandwidth, namely the data transfer rate, of the links between different VMs in a cloud computing system may be different, too. The transfer rate is represented by an $m \times m$ matrix, $B_{m\times m}$, where $B(vm_k, vm_l)$ denotes the bandwidth between VMs $vm_k$ and $vm_l$. In addition, $E_d(t_i)$ represents the amount of data to be

executed by task $t_i$, and $O_d(t_i, t_j)$ indicates the output data generated by $t_i$ and passed to $t_j$.

Therefore, the computation time $T_{comp}(t_i, p_k)$ of task $t_i$ executed on VM $vm_k$ can be calculated by Eq. (1), and the communication time $T_{comm}(t_i(vm_k), t_j(vm_l))$ of edge $(t_i, t_j)$, which means the time consumption in transferring data from task $t_i$ scheduled on $vm_k$ to task $t_j$ scheduled on $vm_l$, can be obtained by Eq. (2).

$$T_{comp}(t_i, vm_k) = \frac{E_d(t_i)}{C_{comp}(vm_k)} \tag{1}$$

$$T_{comm}(t_i(vm_k), t_j(vm_l)) = \frac{O_d(t_i, t_j)}{B(vm_k, vm_l)} \tag{2}$$

The workflow scheduling problem is defined as the assignment of a set of tasks $T$ to the set of VMs $VM$, ensuring that the precedence constraint is satisfied and the makespan of the whole workflow is reduced as much as possible.

## IV. ORIGINAL TLBO

TLBO searches for the optimal value of the problem by simulating the teaching and learning process in a class, corresponding to the two phases of teacher phase and learner phase. In the teacher phase, the teacher imparts knowledge to the students so as to enhance the mean grades of the whole class, whereas in the learner phase, the students learn from each other so that every one of the class can improve his/her grades through interaction with others. In this section, only the two main phases of the algorithm are addressed.

In the teacher phase, the teacher's task is to do his/her best to increase the average marks of the class. Here, a class represents a population, where the best individual is the teacher, and the rest can be the learners. Assume an objective function $f(x)$ needed to be minimized and with $n$-dimensional variables, the $i$th individual can be presented as $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})$. Based on the average level of the class, the teacher utilizes the following formula to teach the learners.

$$X_i^{\text{New}} = X_i + r_i \cdot (X_{\text{Teacher}} - T_F X_{\text{Mean}}) \tag{3}$$

$$T_F = \text{round}(1 + \text{rand}(0, 1)) \tag{4}$$

$$X_{\text{Mean}} = \frac{1}{N} \sum_{i=1}^{N} X_i \tag{5}$$

where rand (0, 1) and $r_i$ are the uniform random numbers between 0 and 1; round($x$) represents $x$ is rounded to the nearest integer; $T_F$ is a teaching factor that decides the value of the mean to be changed; $N$ is the total number of the individuals; $X_{\text{Teacher}}$ is the teacher and the $X_{\text{Mean}}$ is the mean individual of the whole population. If the new individual $X_i^{\text{New}}$ is better than the old one $X_i$, $X_i$ will be replaced. Otherwise, $X_i$ is not changed.

After finishing the teaching process, the learners learn knowledge through interaction between themselves. A learner interacts randomly with others by the means of communications, discussions and consultations, etc. In the learner phase, every learner enhances himself/herself by the following operations:

$$X_i^{\text{New}} = \begin{cases} X_i + r_i \cdot (X_i - X_j) & f(X_i) < f(X_j) \\ X_i + r_i \cdot (X_j - X_i) & f(X_i) > f(X_j), \end{cases} \quad i \neq j. \tag{6}$$

where $X_j$ stands for any randomly selected individual that is different from $X_i$. If the new individual $X_i^{\text{New}}$ is better than the old one $X_i$, $X_i$ will be accepted. Otherwise, $X_i$ is not replaced.

| Task | $t_1$ | $t_2$ | $t_3$ | $t_4$ | $t_5$ | $t_6$ | $t_7$ | $t_8$ | $t_9$ | $t_{10}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| VM | 2 | 4 | 4 | 1 | 3 | 2 | 1 | 1 | 4 | 3 |

**FIGURE 2.** Example of an individual representation.

## V. PROPOSED ALGORITHM HTLBO

We first introduce the overall framework of the HTLBO, then in turn details each part of it.

### A. OVERALL FRAMEWORK

The overall framework of HTLBO algorithm is given in Algorithm 1.

**Algorithm 1** Hybrid Teaching–Learning-Based Optimization (HTLBO) for Workflow Scheduling

**Input:** population size $N$, workflow application and VMs specification in cloud computing system, max iteration $T$
**Output:** the best task-VM mapping $X^{\text{Best}}$
**1:** HEFT-based initial population generation
**2:** $t \leftarrow 1$
**3: While** $t <= T$
**4:** Teacher phase with mean individual OBL
**5:** Learner phase with best individual OBL
**6:** Enhanced learner phase with random individual OBL and genetic operations
**7:** $t \leftarrow t + 1$
**8: End while**
**9:** Find $X^{\text{Best}}$ with the minimum makespan
**10: Return** $X^{\text{Best}}$

### B. INDIVIDUAL REPRESENTATION AND DECODING

In the proposed algorithm, an individual is represented as a mapping of the tasks of a given workflow to the VMs. For a workflow having $n$ tasks and a cloud computing system having $m$ VMs, the $i$th individual can be expressed as:

$$X_i = (x_{i1}, x_{i2}, \ldots, x_{id}, \ldots, x_{in}) \tag{7}$$

where $x_{id}$ indicates the VM $x_{id}$ is assigned to the task $t_d$. It is noteworthy that $x_{id}$ is an integer and lies in $[1, m]$. Fig. 2 illustrates an example of an individual assigning 10 tasks on 4 VMs, in which the workflow of Fig. 1 is used.

While decoding an individual to get the corresponding schedule, namely deciding the start and finish times of each task, the precedence constraints between the tasks must be satisfied. Hence, we decode an individual according to the tasks' descending order of upward ranks [4] which completely satisfies the precedence constraints of the given workflow. After decoding an individual, the makespan of it can be obtained.

### C. MIXED OPPOSITION-BASED LEARNING MODEL

OBL is an intelligent technology proposed by Tizhoosh. The main idea is to evaluate the current solution and its opposition solution simultaneously and keep the better one at last, so as to enhance the search scope and ability of the algorithm [5]. In this paper, a Mixed Opposition-Based Learning model (MOBL) with the change of evolution is proposed, which is defined as follows:

$$\overline{X} = (1 - \frac{t}{T_{\max}}) \cdot \overline{X}^{\mathrm{B}} + \frac{t}{T_{\max}} \cdot \overline{X}^{\mathrm{P}} \tag{8}$$

where $\overline{X}$ means the mixed opposition solution (individual) of $X$; $t$ is the current iteration number, and $T_{\max}$ is the maximum number of iterations; $\overline{X}^{\mathrm{B}}$ is the boundary opposition solution of $\overline{X}$ calculated by the upper and lower bounds of the search space, and $\overline{X}^{\mathrm{P}}$ is the population-based opposition solution of $\overline{X}$ calculated using intra-population information, which can be mean individual, random individual, the best individual, etc. The mathematical models of $\overline{X}^{\mathrm{B}}$ and $\overline{X}^{\mathrm{P}}$ are as follows:

$$\overline{X}^{\mathrm{B}} = LB + UB - r_1 \cdot X \tag{9}$$

$$\overline{X}^{\mathrm{P}} = 2R - r_2 \cdot X \tag{10}$$

where $LB = (lb_1, lb_2, \ldots, lb_n)$ represents the lower bound vector, and apparently, in workflow scheduling, $lb_1 = lb_2, \ldots, = lb_n = 1$; $UB = (ub_1, ub_2, \ldots, ub_n)$ represents the upper bound vector, and $ub_1 = ub_2, \ldots, = ub_n = m$; $R$ is a reference individual, which denotes the intra-population information and can be mean individual, random individual, best individual, etc.; $r_1$ and $r_2$ are two random vectors lying in [0, 1], which are used to control the contraction of the original $X$.

It can be seen from Eq. (8) that in the early stage of the algorithm, $t$ is small, so the coefficient $(1 - t/T_{\max})$ before $\overline{X}^{\mathrm{B}}$ is large, and $\overline{X}^{\mathrm{B}}$ plays a leading role. At this time, the mixed opposition solution generated is not easily affected by the population history search information, which is conducive to the global search of the algorithm and facilitates the finding of more local optimal solutions. However, in the anaphase of evolution, as the value of $t$ increases, so does the value of $t/T_{\max}$, which tends to be 1. At this point, $\overline{X}^{\mathrm{P}}$ plays a leading role, and the mixed opposition solution obtained is mainly affected by the population history search information, which is in flavor of enhancing the local search ability of the algorithm and facilitating the further excavation near the existing local optimal solutions, thus improving the search accuracy and speed of the algorithm.

In view of this, the MOBL model proposed in this paper takes into account both the global exploration at the early stage and the local exploitation at the later stage, which not only ensures the wide search range of the algorithm (jumping out of the local optima), but also guarantees the fine search at the small area (getting closer to the global optimum).

## VI. PHASES OF HTLBO

First, a new enhanced learner phase is added into the HTLBO. Then, according to the learning characteristics of different phases of HTLBO, the reference points for calculating opposition solutions in different stages are set. Finally, multiple opposition learning strategies based on MOBL model are designed to improve the search capacity of HTLBO. The specific implementation of each stage is described as follows.

### A. HEFT-BASED INITIAL POPULATION GENERATION

We use a HEFT-based method to generate an initial population. The steps of this method are described as Algorithm 2.

**Algorithm 2** HEFT-Based Initial Population Generation

**Input:** population size $N$, high-quality individual proportion $\alpha$, workflow application and VMs specification in cloud computing system
**Output:** Initial population *Pop*
**1:** Set $Pop = \Phi$, $num = 0$
**2:** Generate an individual $\lambda_{\mathrm{HEFT}}$ by using HEFT algorithm and get its makespan $l_{\lambda_{\mathrm{HEFT}}}$. Put $\lambda_{\mathrm{HEFT}}$ into *Pop*
**3: While** $num < \lfloor N * \alpha \rfloor - 1$ **do**
**4:** Randomly generate an individual $\lambda_{\mathrm{rand}}$, and calculate its makespan $l_{\lambda_{\mathrm{rand}}}$
**5:** **If** $l_{\lambda_{\mathrm{rand}}} > l_{\lambda_{\mathrm{HEFT}}}$ **then**
**6:** Calculate the quality-based replacement rate $c = \frac{l_{\lambda_{\mathrm{rand}}} - l_{\lambda_{\mathrm{HEFT}}}}{l_{\lambda_{\mathrm{rand}}}}$
**7:** **For each** element $p$ in $\lambda_{\mathrm{rand}}$ **do**
**8:** Generate a uniform random number $r$ between 0 and 1
**9:** **If** $r < c$ **then**
**10:** Set $p = q$, where $q$ is the element of $\lambda_{\mathrm{HEFT}}$ that has the same position as $p$
**11:** **End if**
**12:** **End for**
**13:** **End if**
**14:** Put $\lambda_{\mathrm{rand}}$ into *Pop*
**15:** Set $num = num + 1$
**16: End while**
**17:** Set $num = 0$
**18: While** $num < N - (\lfloor N * \alpha \rfloor - 1)$ **do**
**19:** Randomly generate an individual $\lambda_{\mathrm{rand}}$, and calculate its makespan $l_{\lambda_{\mathrm{rand}}}$. Put $\lambda_{\mathrm{rand}}$ into *Pop*
**20:** Set $num = num + 1$
**21: End while**
**22: Return** *Pop*

At line 6 of Algorithm 2, the quality-based replacement rate $c$ indicates the schedule length improvement of $\lambda_{\mathrm{HEFT}}$ while comparing to $\lambda_{\mathrm{rand}}$. That is to say, in order to enhance the quality of $\lambda_{\mathrm{rand}}$, about $\lfloor c \times n \rfloor$ elements in $\lambda_{\mathrm{rand}}$ should be replaced with the elements that have the same positions as them in $\lambda_{\mathrm{HEFT}}$.

### B. TEACHER PHASE WITH MEAN INDIVIDUAL OPPOSITION-BASED LEARNING

Analyzing the Eq. (1), it can be observed that the teacher phase of TLBO is to search for the best individual by improving the average score of the class. Therefore, the mean individual is chosen as the reference point of opposition learning, that is, in Eqs. (9) and (10), letting $X = X_{\mathrm{Best}}$ and $R = X_{\mathrm{Mean}}$. In this way, the teacher phase based on the mean individual opposition learning is constructed, and its operation steps are shown in Algorithm 3.

**Algorithm 3** Teacher Phase With Mean Individual OBL

**Input:** Population $Pop$
**Output:** New population $Pop^{\text{New}}$
**1: For** each individual $X_i$ in $Pop$ **do**
**2:** Calculate the new solution $X_i^{\text{New}}$ according to Eq. (3)
**3:** Decode $X_i^{\text{New}}$, and get the makespan $l_{X_i^{\text{New}}}$
**4:** **If** $l_{X_i^{\text{New}}} < l_{X_i}$ **then**
**5:** $X_i \leftarrow X_i^{\text{New}}$
**6:** **End if**
**7: End for**
**8:** Calculate the mean individual $X_{\text{Mean}}$ of the population
**9:** Set $R = X_{\text{Mean}}$ and $X = X_{\text{Best}}$ in Eqs. (9) and (10)
**10:** Calculate the opposition solution $\overline{X_{\text{Best}}}$ of the individual $X_{\text{Best}}$ according to Eq. (8)
**11:** Decode $\overline{X_{\text{Best}}}$, and get the makespan $l_{\overline{X_{\text{Best}}}}$
**12: If** $l_{\overline{X_{\text{Best}}}} < l_{X_{\text{Best}}}$ **then**
**13:** $X_{\text{Best}} \leftarrow \overline{X_{\text{Best}}}$
**14: End if**
**15: Return** $Pop^{\text{New}}$

### C. LEARNER PHASE WITH BEST INDIVIDUAL OPPOSITION-BASED LEARNING

As can be seen from Eq. (4), in the learner stage of TLBO, individuals improve themselves by learning from randomly selected individuals, while this may make it difficult for the worst individual in the population to improve. Hence, in this stage, OBL is applied to improve the worst individual by learning the best in the opposite direction.

The best individual $X_{\text{Best}}$ is chosen as the reference point of opposition learning here, namely, in Eqs. (9) and (10), letting $X = X_{\text{Worst}}$ and $R = X_{\text{Best}}$. The operation steps of the learner phase combined with the best individual opposition learning are described in Algorithm 4.

**Algorithm 4** Learner Phase With Best Individual OBL

**Input:** Population $Pop$
**Output:** New population $Pop^{\text{New}}$
**1: For** each individual $X_i$ in $Pop$ **do**
**2:** Calculate the new solution $X_i^{\text{New}}$ according to Eq. (6)
**3:** Decode $X_i^{\text{New}}$, and get the makespan $l_{X_i^{\text{New}}}$
**4:** **If** $l_{X_i^{\text{New}}} < l_{X_i}$ **then**
**5:** $X_i \leftarrow X_i^{\text{New}}$
**6:** **End if**
**7: End for**
**8:** Set $R = X_{\text{Best}}$ and $X = X_{\text{Worst}}$ in Eqs. (9) and (10)
**9:** Calculate the opposition solution $\overline{X_{\text{Worst}}}$ of the individual $X_{\text{Worst}}$ according to Eq. (8)
**10:** Decode $\overline{X_{\text{Worst}}}$, and get the makespan $l_{\overline{X_{\text{Worst}}}}$
**11: If** $l_{\overline{X_{\text{Worst}}}} < l_{X_{\text{Worst}}}$ **then**
**12:** $X_{\text{Worst}} \leftarrow \overline{X_{\text{Worst}}}$
**13: End if**
**14: Return** $Pop^{\text{New}}$

### D. ENHANCED LEARNER PHASE WITH RANDOM INDIVIDUAL OPPOSITION-BASED LEARNING AND GENETIC OPERATIONS

In order to further enhance the diversity of the population in the later stage of evolution, an enhanced learner phase is added to the HTLBO, and its implementation is shown in Algorithms 5–7.

**Algorithm 5** Enhanced Learner Phase With Random Individual OBL and Genetic Operations

**Input:** Population $Pop$
**Output:** New population $Pop^{\text{New}}$
**1:** Set $X = X_{\text{Best}}$ in Eqs. (9) and (10)
**2: For** each individual $X_i$ in $Pop$ **do**
**3:** Randomly choose an individual $X_{\text{Rand}}$
**4:** Set $R = X_{\text{Rand}}$ in Eq. (10)
**5:** Calculate the opposition solution $\overline{X_{\text{Best}}}$ of the individual $X_{\text{Best}}$ according to Eq. (8)
**6:** **If** choice = crossover **then**
**7:** Apply two-point crossover operation on $X_i$ and $\overline{X_{\text{Best}}}$ to produce a new individual $X_i^{\text{New}}$
**8:** **Else**
**9:** Apply mutation operation on $X_i$ to produce a new individual $X_i^{\text{New}}$
**10:** **End if**
**11:** Decode $X_i^{\text{New}}$, and get the makespan $l_{X_i^{\text{New}}}$
**12:** **If** $l_{X_i^{\text{New}}} < l_{X_i}$ **then**
**13:** $X_i \leftarrow X_i^{\text{New}}$
**14:** **End if**
**15: End for**
**16: Return** $Pop^{\text{New}}$

**Algorithm 6** Two-Point Crossover

**Input:** $X_i$ and $\overline{X_{\text{Best}}}$
**Output:** $X_i^{\text{New}}$
**1:** Randomly generate two crossover positions $pos_1$ and $pos_2$ ($pos_1 \neq pos_2$)
**2:** $X_i^{\text{New}} \leftarrow X_i$
**3: For** each element $p$ whose position is between $pos_1$ and $pos_2$ in $X_i^{\text{New}}$
**4:** Set $p = q$, where $q$ is the element of $\overline{X_{\text{Best}}}$ that has the same position as $p$
**5: End for**
**6: Return** $X_i^{\text{New}}$

To balance the population diversity and convergence accuracy, the best individual and randomly chosen individual are used for calculating the opposition solution at this stage, namely, setting $R = X_{\text{Rand}}$ and $X = X_{\text{Best}}$. Obviously, for each $X_i$, the opposition solution $\overline{X_{\text{Best}}}$ is different due to randomly chosen reference point $X_{\text{Rand}}$, so the offspring obtained from the crossover of $X_i$ and $\overline{X_{\text{Best}}}$ is conducive to maintaining the diversity of the population. Furthermore, the mutation with small probability done on $X_i$ is helpful for the algorithm to jump out of the local optima.

## VII. TIME COMPLEXITY ANALYSIS

The time complexity of the algorithm is calculated by summation of the time complexity of the most time-consuming operations in HTLBO. Suppose the size of the population is $N$, the maximum number of iterations is $T$. $\alpha$ is the high-quality individual proportion of the initial population. The computational complexity of decoding an individual, namely generating a schedule and obtaining its makespan, is set to be $O(f)$. HTLBO has three individual updating methods, which are real number operation in full dimensions, crossover operation in partial dimensions and mutation operation in a small number of dimensions. For the sake of simplicity, their time complexities are set to $O(g)$, $O(h)$ and $O(k)$, respectively. In the initialization stage, the time complexity of producing an initial population is

**Algorithm 7** Mutation

**Input:** $X_i$ and mutation probability $\beta$
**Output:** $X_i^{\text{New}}$
**1:** $X_i^{\text{New}} \leftarrow X_i$
**2: For** each element $p$ in $X_i^{\text{New}}$
**3:** Generate a uniform random number $r$ between 0 and 1
**4:** **If** $r < \beta$
**5:** Randomly choose a VM whose number is $q$
**6:** Set $p = q$
**7:** **End if**
**8: End for**
**9: Return** $X_i^{\text{New}}$

$O(Ng)$, and the time complexity of transforming $N\alpha$ ordinary individuals into high-quality individuals is $O(N\alpha h)$. Hence, the total time complexity of initialization stage is $O(Ng + N\alpha h + Nf)$. Each iteration of HTLBO consists of three search stages. The teacher and learner phases use real number operation to update the individuals, so the time complexity of both is $O(Ng + Nf)$. The enhanced leaner phase uses crossover and mutation to update individuals with equal probability, so the time complexity can be presented as $O(\frac{1}{2}Nh + \frac{1}{2}Nk + Nf)$. To sum up, the total time complexity of HTLBO is:

$$O_{\text{HTLBO}}(Ng + N\alpha h + Nf + (2Ng + \frac{1}{2}Nh + \frac{1}{2}Nk + 3Nf)T) \tag{11}$$

Furthermore, since all algorithms in this study use max number of generated schedules ($maxGS$) to terminate the algorithms, we also provide $maxGS$-based time complexity of HTLBO. Apparently, $N + 3NT = maxGS$, while $N$ is much less than $3NT$, so $3NT = maxGS$ can be obtained. Replacing $T$ with $\frac{maxGS}{3N}$, the time complexity of HTLBO can finally be:

$$O_{\text{HTLBO}}(Ng + N\alpha h + Nf + (\frac{2}{3}g + \frac{1}{6}h + \frac{1}{6}k + f)maxGS) \tag{12}$$

Using the similar method, the time complexity of the original TLBO can be expressed as:

$$O_{\text{TLBO}}(Ng + Nf + (g + f)maxGS) \tag{13}$$

Since $O(g) > O(h) > O(k)$, the complexity of HTLBO in Eq. (12) can be reduced to Eq. (14) as follows:

$$O_{\text{HTLBO}}(Ng + Nf + (\frac{2}{3}g + f)maxGS) \tag{14}$$

Therefore, the time complexity of HTLBO is slightly less than that of TLBO, which is proved by the actual running time comparison in the following experiment.

## VIII. AN ILLUSTRATION

We illustrate the HTLBO process using the workflow in Fig. 1, which has the same structure as the reference [4]. In the diagram, the number in a task node represents the computational amount of the task, in million instructions (MI). And the data next to the edge represents the data traffic between nodes, in MB. The experimental environment is set up as described in the next section.

| $\lambda_{\text{HEFT}}$ | 3 | 4 | 1 | 3 | 2 | 3 | 2 | 3 | 4 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|

**FIGURE 3.** Individual $\lambda_{\text{HEFT}}$ computed by HEFT.

| $\lambda_{\text{rand}}$ | 2 | 2 | 2 | 2 | 1 | 1 | 3 | 3 | 3 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|

**FIGURE 4.** Random individual $\lambda_{\text{rand}}$.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $\lambda_{\text{rand}}$ | 2 | 2 | 1 | 2 | 2 | 1 | 3 | 3 | 3 | 2 |
| $\lambda_{\text{HEFT}}$ | 3 | 4 | 1 | 3 | 2 | 3 | 2 | 3 | 4 | 3 |
| $r$ | 0.0249 | 0.3138 | 0.1840 | 0.6431 | 0.4611 | 0.1936 | 0.2087 | 0.6309 | 0.2575 | 0.1127 |
| improved $\lambda_{\text{rand}}$ $r < c$ | 3 | 4 | 1 | 2 | 2 | 3 | 2 | 3 | 4 | 3 |

**FIGURE 5.** Implementation of improved $\lambda_{\text{rand}}$.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $R = X_{\text{Mean}}$ | 2 | 2 | 2 | 3 | 2 | 2 | 2 | 2 | 3 | 2 |
| $X_{\text{Best}}$ | 3 | 3 | 2 | 4 | 1 | 2 | 1 | 2 | 3 | 3 |
| $\overline{X_{\text{Best}}}$ | 2 | 4 | 4 | 3 | 4 | 3 | 3 | 3 | 2 | 3 |

**FIGURE 6.** Implementation of $\overline{X_{\text{Best}}}$ using the mean individual as reference.

In the population initialization phase, an individual $\lambda_{\text{HEFT}}$ for the given workflow is computed using HEFT, which is shown in Fig. 3 in a simplified mode. The task scheduling order is $(t_1, t_4, t_2, t_5, t_3, t_6, t_9, t_7, t_8, t_{10})$. Thus, the makespan of $\lambda_{\text{HEFT}}$ can be calculated to be 60.14. For a random individual $\lambda_{\text{rand}}$ shown in Fig. 4 with the makespan of 91.02, the quality-based replacement rate $c$ is calculated to be 0.3393. An improved $\lambda_{\text{rand}}$ can be obtained using Algorithm 2, and its implementation is shown in Fig. 5, where $r$ is random numbers lying [0, 1]. Apparently, by replacing some of the elements in $\lambda_{\text{rand}}$ with the corresponding values in $\lambda_{\text{HEFT}}$, the quality of $\lambda_{\text{rand}}$ is significantly improved, and its makespan is reduced to 73.33.

In the teacher phase, suppose that the mean individual obtained in a certain iteration is $X_{\text{Mean}}$, and set it as the reference individual $R$. For the current best individual $X_{\text{Best}}$, calculate its opposition solution $\overline{X_{\text{Best}}}$, whose implementation is illustrated in Fig. 6. The makespan of $X_{\text{Best}}$ is 59.76, but the makespan of $\overline{X_{\text{Best}}}$ is 75.67, so $\overline{X_{\text{Best}}}$ is not used to replace $X_{\text{Best}}$.

In the learner phase, $X_{\text{Best}}$ is set to be the reference individual $R$. For a given worst individual $X_{\text{Worst}}$, the implementation of its opposition solution $\overline{X_{\text{Worst}}}$ is described as follows. The makespan of $X_{\text{Worst}}$ is 73.72, and the makespan of $\overline{X_{\text{Worst}}}$ is 62.19. Since the improvement of $\overline{X_{\text{Worst}}}$ is obvious, it can be used to replace $X_{\text{Worst}}$.

In the enhanced learner phase, an individual $X_{\text{Rand}}$ is randomly chosen and used as the reference point to calculate the opposition solution $\overline{X_{\text{Best}}}$ of $X_{\text{Best}}$. An example is shown in Fig. 8.

Individuals can be improved by crossing with $\overline{X_{\text{Best}}}$ or mutating themselves. Examples of crossover and mutation are shown in Figs. 9 and 10, respectively. After crossover, the

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $R = X_{\text{Best}}$ | 3 | 3 | 2 | 4 | 1 | 2 | 1 | 2 | 3 | 3 |
| $X_{\text{Worst}}$ | 2 | 2 | 3 | 3 | 4 | 3 | 3 | 3 | 4 | 3 |
| $\overline{X_{\text{Worst}}}$ | 3 | 4 | 2 | 3 | 3 | 1 | 3 | 3 | 4 | 2 |

**FIGURE 7.** Implementation of $\overline{X_{\text{Worst}}}$ using the best individual as reference.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $R = X_{\text{Rand}}$ | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 2 | 1 |
| $X_{\text{Best}}$ | 3 | 3 | 2 | 4 | 2 | 3 | 3 | 2 | 4 | 3 |
| $\overline{X_{\text{Best}}}$ | 4 | 4 | 4 | 2 | 3 | 3 | 3 | 4 | 3 | 4 |

**FIGURE 8.** Implementation of $\overline{X_{\text{Best}}}$ using a randomly selected individual as reference.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $X$ | 3 | 3 | 3 | 1 | 4 | 2 | 3 | 3 | 2 | 3 |
| $\overline{X_{\text{Best}}}$ | 4 | 4 | 4 | 2 | 3 | 3 | 3 | 4 | 3 | 4 |
| $X^{\text{New}}$ | 3 | 4 | 4 | 2 | 3 | 3 | 3 | 3 | 2 | 3 |

**FIGURE 9.** Two-point crossover of $X$ and $\overline{X_{\text{Best}}}$.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $X$ | 3 | 3 | 2 | 1 | 2 | 4 | 2 | 2 | 3 | 3 |
| $X^{\text{New}}$ | 3 | 3 | 2 | 1 | 2 | 4 | 4 | 2 | 3 | 3 |

**FIGURE 10.** Mutation of $X$.

**TABLE 1.** Mian iterations of HTLBO.

| Iteration | Makespan (s) |
|---|---|
| 1 | 60.14 |
| 2 | 60.14 |
| …… | …… |
| 17 | 59.68 |
| …… | …… |
| **55** | **59.11** |
| …… | …… |
| 65 | 59.11 |

| Task | $t_1$ | $t_2$ | $t_3$ | $t_4$ | $t_5$ | $t_6$ | $t_7$ | $t_8$ | $t_9$ | $t_{10}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| VM | 3 | 3 | 1 | 2 | 4 | 4 | 2 | 4 | 3 | 3 |

**FIGURE 11.** Resultant schedule of the workflow sample.

makespan of $X^{\text{New}}$ is shortened from the original 64.47 to 63.44. Mutation can sometimes improve the quality of an individual, as shown in Fig. 10, where the makespan of $X^{\text{New}}$ is reduced from 60.96 to 60.09.

Table 1 shows the details of the best individual identified in main iterations with respect to the makespan. The resultant schedule is presented in Fig. 11.

**TABLE 2.** Size of the workflow based on the number of tasks.

| Number of Tasks | Size |
|---|---|
| 24 to 60 | Small |
| 100 to 400 | Medium |
| 800 to 2000 | Large |

**TABLE 3.** Workflow scheduling simulation parameters.

| Parameters | Value |
|---|---|
| Number of VMs | 4 |
| Heterogeneity | (0.8, 1.2) |
| RAM (MB) | 512 |
| Basic MIPS | 1000 |
| Basic BW (Mbps) | 2000 |
| Pes Number | 1 |

## IX. EXPERIMENTAL RESULTS AND DISCUSSION

This section presents the simulation results and discussion of the proposed algorithm from the perspectives of control parameter settings, algorithm comparison and non-parametric statistics.

### A. EXPERIMENTAL SETUP

The WorkflowSim Framework based on CloudSim [31] is chosen to carry out the experiments. HTLBO is evaluated on various scientific workflows obtained from the Pegasus workflow repository [58]. These real-work workflows include CyberShake, Epigenomics, Inspiral, Montage, and Sipht. We divide them into three sizes based on the number of tasks in the workflow [40], which is shown in Table 2. Moreover, the workflow scheduling simulation parameters are listed in Table 3, where MIPS, BW and Pes number represent million instructions per second, bandwidth, and number of CPUs, respectively. It should be noted that in this study, in order to better reflect the authenticity of the cloud computing environment, the heterogeneity factor is set for each VM resource, whose value is a random number of (0.8, 1.2). For the four VMs in this study, the heterogeneity factors are set as (0.8140, 0.9647, 1.1832, 1.0196), so the true MIPS and BW for each VM are obtained by multiplying the basic MIPS and BW by the corresponding heterogeneity factor.

In order to verify the comprehensive performance of HTLBO, some outstanding workflow scheduling algorithms are chosen and benchmarked for comparison. They are original TLBO [6], HEFT [4], HEFTGA [31], HGA [2], and HGSA [40]. Among these algorithms, HEFT is a popular heuristic algorithm for solving workflow scheduling; TLBO is a metaheuristic; and HEFTGA, HGA, HGSA and HTLBO are hybrid metaheuristics integrated with HEFT. The settings for all algorithms compared are identical to those recommended in their original works. Table 4 shows the parameter settings for algorithms involved.

All algorithms were implemented using Java coding environment on a Core i7 processor running at 2.4GHz and equipped with 16GB of main memory. The max number

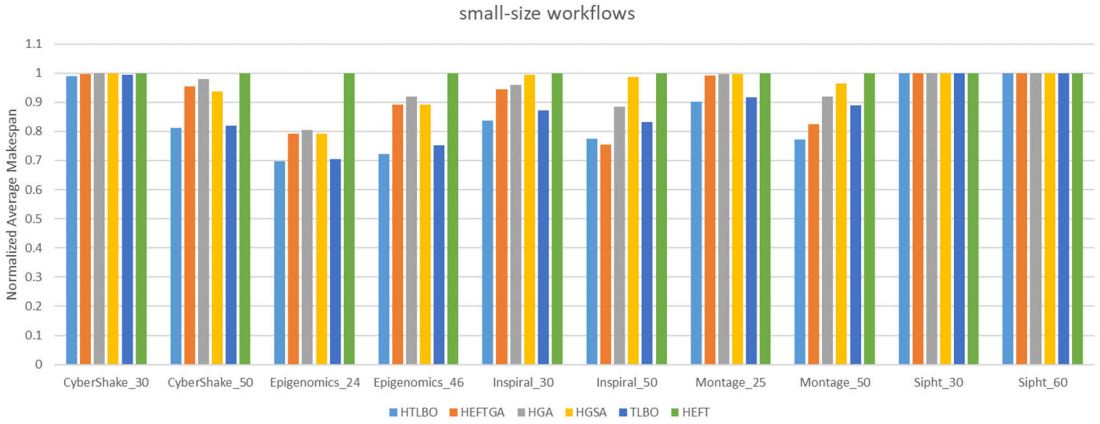


**FIGURE 12.** Comparison of small-size workflows.

**TABLE 4.** Parameter settings for algorithms involved.

| Algorithms | Parameters | Values |
|---|---|---|
| HTLBO | High-quality individual proportion $\alpha$ of initial population | 0.3 |
| | Mutation probability $\beta$ | 0.05 |
| TLBO | ~ | ~ |
| HEFTGA | Tournament size $nbr$ | 4 |
| HGA | Elitism rate $\alpha$ | 0.2 |
| | Mutation rate $\beta$ | 0.02 |
| HGSA | Gravitational constant $G_0$ | 5 |
| | Small constant used in gravity $\gamma$ | 0.3 |
| | Mass threshold for inferior agents $\delta$ | 0.1 |
| | Small constant used in force $\varepsilon$ | 10 |
| HEFT | ~ | ~ |

of generated schedules (maxGS), which are obtained after individual decoding, is used as the criterion for terminating each algorithm, with a maximum of 10000 schedules. The population size of each algorithm is set as 50. All results are recorded and compared based on the algorithms' average performance over 30 independent runs.

### B. SETTINGS OF HIGH-QUALITY INDIVIDUAL PROPORTION $\alpha$ AND MUTATION PROBABILITY $\beta$

The high-quality individual proportion $\alpha$ is applied to control the proportion of HEFT-based individuals in the initial population. Apparently, a larger $\alpha$ increases the number of high-quality individuals in the population, thus speeding up the convergence of the algorithm. However, this situation reduces the diversity of the population, making the algorithm easy to fall into local optima early and difficult to continue to progress in the late iterations. Therefore, the value of $\alpha$ should not be set too large. In this study, $\alpha$ ranges from the set $\{0.1, 0.2, 0.3, 0.4, 0.5\}$.

The mutation probability $\beta$ is used to help the algorithm jump out of the local optima and increase the probability that the algorithm can still find some better solutions in the later iterations. Generally, a small number of mutation operations can significantly increase the diversity of the population, so $\beta$ should be set to a small value. In this study, $\beta$ takes values from $\{0.05, 0.10, 0.15, 0.20\}$.

We select five different types of medium-size workflows as representatives to test the scheduling performance (average makespan) of HTLBO under different combinations of $\alpha$ and $\beta$. And the Friedman test is utilized to determine the best combination. The results are presented in Table 5, where makespan is measured in seconds.

As can be seen from the table, when $\alpha = 0.3$ and $\beta = 0.05$, the mean rank is the smallest. Hence, the thresholds of $\alpha$ and $\beta$ in HTLBO are set to 0.3 and 0.05, respectively.

### C. COMPARISON OF HTLBO WITH OTHER ALGORITHMS

The performance of HTLBO is estimated against HEFT, TLBO, HEFTGA, HGA, and HGSA with respect to normalized average makespan. For the sake of easy comparison and visualization of the overall quality of the results, the max-normalization is used to normalize the average makespan as calculated in Eq. (15).

$$avgMS_i^{\mathrm{Nor}} = \frac{avgMS_i}{\max\limits_{j=1 \text{ to } J}(avgMS_j)} \tag{15}$$

where $avgMS_i^{\mathrm{Nor}}$ is the normalized value for average makespan $avgMS_i$ obtained by algorithm $i$. Figs. 12–14 show the bar charts for the normalized average makespan, so as to demonstrate the effectiveness of HTLBO compared to other algorithms. *It should be noted that* in order to compare the optimization characteristics of various algorithms more fairly, we also added HEFT to the original TLBO for generating the initial population. In other words, TLBO is a hybrid algorithm mixed with HEFT in this study.

From the figures, it can be observed that HTLBO obtains the best results in 31 out of 35 workflow instances. To be specific, for the small-size workflows, HTLBO performs better than or equal to other algorithms except for Inspiral_50. This means that HTLBO has good local exploitation ability, that is, in the workflow scheduling problems with small search space, it can mine the solutions with higher accuracy than the competitors. And for medium-size workflows, HTLBO

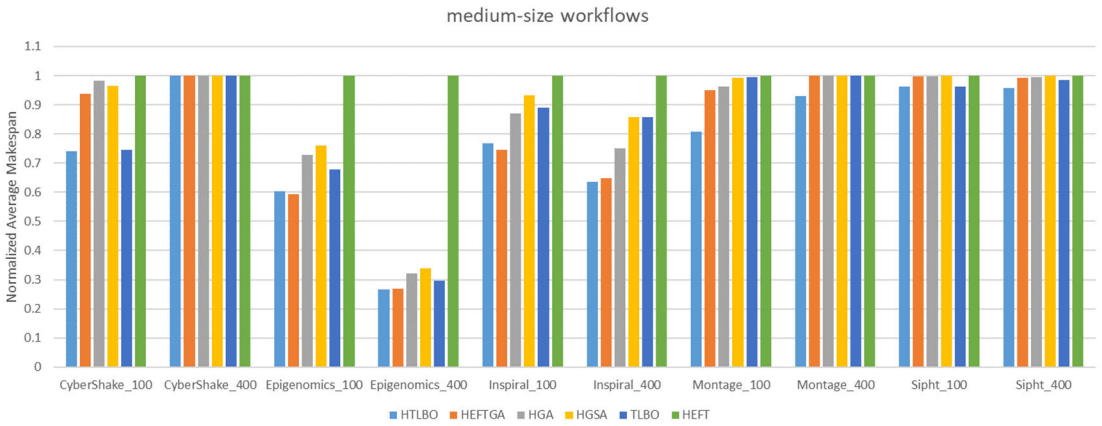


**FIGURE 13.** Comparison of medium-size workflows.

**TABLE 5.** Combination tests of different $\alpha$ and $\beta$.

| $\alpha$ | $\beta$ | CyberShake_100 | Epigenomics_100 | Inspiral_100 | Montage_100 | Sipht_100 | Mean rank | Final rank |
|---|---|---|---|---|---|---|---|---|
| 0.1 | 0.05 | 602.34 | 63375.14 | 4087.82 | 235.45 | 5019.94 | 4.4 | 2 |
| | 0.10 | 611.73 | 64608.55 | 4152.89 | 240.25 | 5030.48 | 12.4 | 12 |
| | 0.15 | 612.02 | 65196.20 | 4122.73 | 239.68 | 4960.09 | 9.2 | 7 |
| | 0.20 | 612.91 | 64689.22 | 4221.54 | 242.38 | 5009.19 | 13.4 | 16 |
| 0.2 | 0.05 | 596.43 | 63451.09 | 4087.87 | 239.36 | 5029.68 | 6.6 | 3 |
| | 0.10 | 610.47 | 64976.27 | 4117.12 | 237.41 | 5014.75 | 8.6 | 6 |
| | 0.15 | 610.31 | 64461.02 | 4196.80 | 239.23 | 5002.42 | 9.2 | 8 |
| | 0.20 | 616.19 | 66061.73 | 4140.85 | 243.66 | 4999.90 | 13.6 | 17 |
| 0.3 | 0.05 | 606.49 | 64325.27 | 4043.41 | 234.56 | 5012.55 | **4.2** | **1** |
| | 0.10 | 607.20 | 65622.60 | 4108.57 | 237.81 | 5005.12 | 7.6 | 5 |
| | 0.15 | 613.58 | 65841.56 | 4095.20 | 243.84 | 5024.68 | 13.6 | 18 |
| | 0.20 | 612.30 | 67115.12 | 4122.25 | 242.19 | 5000.63 | 12.6 | 13 |
| 0.4 | 0.05 | 603.15 | 64406.94 | 4018.05 | 244.23 | 5037.51 | 9.4 | 9 |
| | 0.10 | 608.95 | 65864.85 | 4119.96 | 241.30 | 5013.41 | 11 | 10 |
| | 0.15 | 618.88 | 64961.04 | 4150.63 | 239.58 | 5027.90 | 13.2 | 15 |
| | 0.20 | 626.96 | 67397.14 | 4172.66 | 241.36 | 5001.68 | 15.4 | 20 |
| 0.5 | 0.05 | 604.40 | 64430.62 | 4068.63 | 238.67 | 5037.67 | 7.4 | 4 |
| | 0.10 | 614.15 | 65925.21 | 4154.61 | 240.19 | 5020.51 | 14.4 | 19 |
| | 0.15 | 619.18 | 64714.30 | 4203.04 | 239.07 | 4996.81 | 11 | 11 |
| | 0.20 | 620.58 | 66678.09 | 4121.92 | 242.06 | 4994.22 | 12.8 | 14 |

still maintains the performance advantage and outperforms the comparison algorithms in the vast majority of cases except for Epigenomics_100 and Inspiral_100. Moreover, with the increase of workflow size, the performance advantage of HTLBO becomes more obvious, as shown in the scheduling results of large-scale workflows. In particular, when solving workflows Montage and Sipht with complex structure and large scale, the results obtained by HTLBO are significantly better than those of other algorithms. However, other hybrid metaheuristics can hardly get better results than those got by HEFT because of the huge problem search space. Thus, HTLBO also possesses good global exploration ability. It can find more excellent solutions than other algorithms when encountering large search space. To be concluded, the proposed HTLBO achieves a good balance between global exploration and local exploitation.

Furthermore, the running times of compared algorithms on various types of workflows are given in Table 6, where the reported times are the average running times of the workflows.

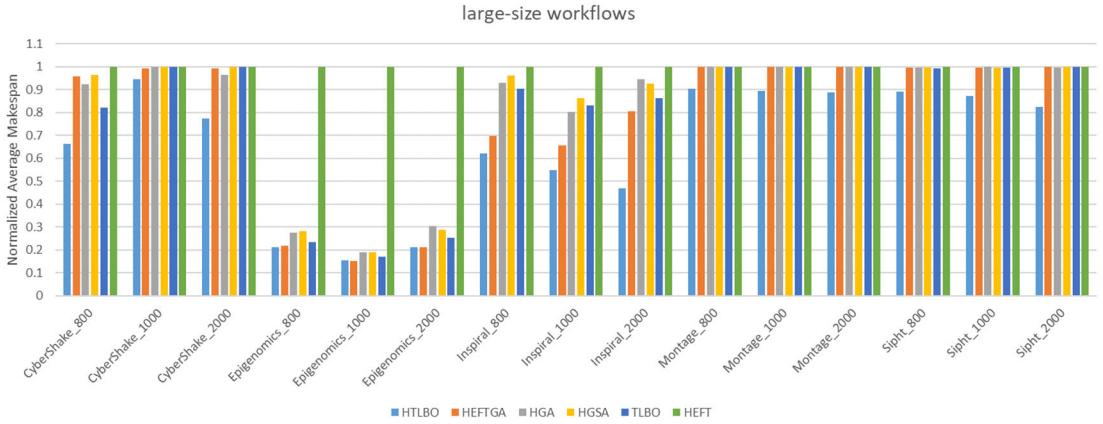


**FIGURE 14.** Comparison of large-size workflows.

**TABLE 6.** Running times of compared algorithms.

| Workflow sizes | Algorithms | Running times (s) | Rank |
|---|---|---|---|
| Small-size | HTLBO | 0.1313 | 4 |
| | HEFTGA | 0.1088 | 3 |
| | HGA | 0.1058 | 2 |
| | HGSA | 1.3093 | 6 |
| | TLBO | 0.1381 | 5 |
| | HEFT | 0.0001 | 1 |
| Medium-size | HTLBO | 1.2979 | 4 |
| | HEFTGA | 1.1689 | 3 |
| | HGA | 1.1290 | 2 |
| | HGSA | 48.6604 | 6 |
| | TLBO | 1.3901 | 5 |
| | HEFT | 0.0015 | 1 |
| Large-size | HTLBO | 32.4858 | 4 |
| | HEFTGA | 26.7836 | 3 |
| | HGA | 26.0089 | 2 |
| | HGSA | 1526.1598 | 6 |
| | TLBO | 33.7278 | 5 |
| | HEFT | 0.1472 | 1 |

As can be seen from the table, HTLBO ranks fourth among all algorithms regardless of the workflow size. The consuming time of HTLBO is slightly less than that of TLBO, which confirms the above time complexity analysis. HEFT is the least time-consuming algorithm of all, because as a heuristic, it produces only one schedule. The running times of HEFTGA and HGA are less than that of HTLBO, because both are hybrid metaheuristic algorithms based on GA, which use partial-dimension updates to generate new individuals. HGSA has the longest running time among all algorithms, as it generates new individuals using full-dimension updates and computes a large number of Euclidean distances in each iteration.

All in all, HTLBO maintains good execution efficiency under the condition of obtaining outstanding scheduling results, and its running time is completely acceptable. Thus, HTLBO achieves a good balance between effectiveness and efficiency.

**TABLE 7.** Non-parametric statistical tests.

| Algorithms | +/=/- | Mean rank | Final rank |
|---|---|---|---|
| HTLBO | | **1.3143** | **1** |
| HEFTGA | 24/11/0 | 2.6857 | 2 |
| HGA | 32/3/0 | 3.6571 | 4 |
| HGSA | 32/3/0 | 4.5571 | 5 |
| TLBO | 31/4/0 | 3.1714 | 3 |
| HEFT | 32/3/0 | 5.6143 | 6 |

### D. NON-PARAMETRIC STATISTICAL TESTS

Wilcoxon signed rank test and Friedman test [59] were used to further verify the effectiveness and superiority of the proposed algorithm HTLBO from the perspective of non-parametric statistical analysis. The significance levels of the tests are set to 0.05, and the test results are shown in Table 7. The symbols "+, −, =" indicate the times that HTLBO is superior, inferior, or equal to the compared algorithms, respectively, which are obtained by Wilcoxon signed rank test. For example, the result of "24/11/0" represents that HTLBO outperforms HEFTGA on 24 of 35 cases, is equal to it on 11 cases, and is inferior to it on 0 case. The mean ranks are calculated using Friedman test.

As can be seen in the table, in the Wilcoxon signed rank test, HTLBO is not inferior to the comparison algorithms in any case, and the performance advantages are significant. In the Friedman test, HTLBO obtains the smallest mean rank. Therefore, it can be demonstrated that HTLBO has a prominent advantage in non-parametric statistics compared with other algorithms.

## X. CONCLUSION AND FUTURE WORK

In this study, a hybrid metaheuristic HTLBO for real-world scientific workflow scheduling in heterogeneous cloud environment is proposed. As a solution of this problem, the proposed algorithm integrates HEFT, OBL and genetic operations into the original TLBO to achieve a minimal makespan of the workflow scheduling. A HEFT-based method is applied to generate a high-quality initial population for enhancing the convergence accuracy and speed of the algorithm. Based on the principle of balancing the global exploration and local exploitation of the algorithm as much as possible, a mixed OBL model combining the boundary search information and the population history search information is designed to generate some opposition-based solutions for expanding the search range in each stage. In order to improve the diversity of the population and avoid falling into local optima too early, an enhanced learner phase implemented using genetic operations is also added to the algorithm. Experiments on real-world workflows as CyberShake, Epigenomics, Inspiral, Montage, and Sipht with various task sizes demonstrate that HTLBO has outperformed HEFT, original TLBO and some state-of-the-art hybrid metaheuristics including HEFTGA, HGA, HGSA in terms of average makespan while achieving acceptable running times. Furthermore, the non-parametric statistics also show that the difference between HTLBO and these comparison algorithms is significant. Therefore, the proposed HTLBO can achieve a good balance between scheduling effectiveness and efficiency.

It should be noted that HTLBO aims to shorten the workflow makespan as much as possible when considering the sharing and heterogeneity of cloud resources. However, in practical applications, some scenarios about task deadline, multi-objective optimization, workflow budget constraints, and cloud resource reliability constraints are not taken into account in HTLBO.

Hence, in future work, we plan to use HTLBO for tackling the problems of multi-objective optimization with multiple constraints when scheduling workflows in cloud environment. Another work ahead is to further improve HTLBO by introducing a dual encoding method with complete search space, which consists of a discrete coding part based on task permutation and a continuous coding part based on resource allocation.

## REFERENCES

[1] A. Iranmanesh and H. R. Naji, "DCHG-TS: A deadline-constrained and cost-effective hybrid genetic algorithm for scientific workflow scheduling in cloud computing," *Cluster Comput.*, vol. 24, no. 2, pp. 667–681, Jun. 2021.

[2] S. G. Ahmad, C. S. Liew, E. U. Munir, T. F. Ang, 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.

[3] F. Tao, Y. Feng, L. Zhang, and T. W. Liao, "CLPS-GA: A case library and Pareto solution-based hybrid genetic algorithm for energy-aware cloud service scheduling," *Appl. Soft Comput.*, vol. 19, pp. 264–279, Jun. 2014.

[4] H. Topcuoglu, S. Hariri, and M.-Y. 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.

[5] H. R. Tizhoosh, "Opposition-based learning: A new scheme for machine intelligence," in *Proc. Int. Conf. Comput. Intell. Modeling, Control Autom. Int. Conf. Intell. Agents, Web Technol. Internet Commerce (CIMCA-IAWTIC)*, Nov. 2005, pp. 695–701.

[6] R. V. Rao, V. J. Savsani, and D. P. Vakharia, "Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems," *Comput.-Aided Des.*, vol. 43, no. 3, pp. 303–315, Mar. 2011.

[7] E. Ilavarasan and P. Thambidura, "Low complexity performance effective task scheduling algorithm for heterogeneous computing environments," *J. Comput. Sci.*, vol. 3, no. 2, pp. 94–103, Feb. 2007.

[8] H. Arabnejad and J. G. Barbosa, "List scheduling algorithm for heterogeneous systems by an optimistic cost table," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 682–694, Mar. 2014.

[9] E. U. Munir, S. Mohsin, A. Hussain, M. W. Nisar, and S. Ali, "SDBATS: A novel algorithm for task scheduling in heterogeneous computing systems," in *Proc. IEEE Int. Symp. Parallel Distrib. Process., Workshops Phd Forum*, May 2013, pp. 43–53.

[10] M. Qasim, T. Iqbal, E. U. Munir, N. Tziritas, S. U. Khan, and L. T. Yang, "Dynamic mapping of application workflows in heterogeneous computing environments," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW)*, May 2017, pp. 462–471.

[11] L. Zhang, L. Zhou, and A. Salah, "Efficient scientific workflow scheduling for deadline-constrained parallel tasks in cloud computing environments," *Inf. Sci.*, vol. 531, pp. 31–46, Aug. 2020.

[12] Z. Zhu and X. Tang, "Deadline-constrained workflow scheduling in IaaS clouds with multi-resource packing," *Future Gener. Comput. Syst.*, vol. 101, pp. 880–893, Dec. 2019.

[13] M. Amoon, N. El-Bahnasawy, and M. ElKazaz, "An efficient cost-based algorithm for scheduling workflow tasks in cloud computing systems," *Neural Comput. Appl.*, vol. 31, no. 5, pp. 1353–1363, May 2019.

[14] X. Zhou, G. Zhang, J. Sun, J. Zhou, T. Wei, and S. Hu, "Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based HEFT," *Future Gener. Comput. Syst.*, vol. 93, pp. 278–289, Apr. 2019.

[15] H. R. Faragardi, M. R. S. 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.

[16] Z. Quan, Z.-J. Wang, T. Ye, and S. Guo, "Task scheduling for energy consumption constrained parallel applications on heterogeneous computing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 5, pp. 1165–1182, May 2020.

[17] N. Rizvi, D. Ramesh, L. Wang, and A. Basava, "A workflow scheduling approach with modified fuzzy adaptive genetic algorithm in IaaS clouds," *IEEE Trans. Services Comput.*, vol. 16, no. 2, pp. 872–885, Mar. 2023.

[18] 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.

[19] Z.-J. Wang, Z.-H. Zhan, W.-J. Yu, Y. Lin, J. Zhang, T.-L. Gu, and J. Zhang, "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.

[20] X. Tang, C. Shi, T. Deng, Z. Wu, and L. Yang, "Parallel random matrix particle swarm optimization scheduling algorithms with budget constraints on cloud computing systems," *Appl. Soft Comput.*, vol. 113, Dec. 2021, Art. no. 107914.

[21] Y.-H. Jia, W.-N. Chen, H. Yuan, T. Gu, H. Zhang, Y. Gao, and J. Zhang, "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.

[22] A. Muteeh, M. Sardaraz, and M. Tahir, "MrLBA: Multi-resource load balancing algorithm for cloud computing using ant colony optimization," *Cluster Comput.*, vol. 24, no. 4, pp. 3135–3145, Dec. 2021.

[23] M. R. Thanka, P. U. Maheswari, and E. B. Edwin, "An improved efficient: Artificial bee colony algorithm for security and QoS aware scheduling in cloud computing environment," *Cluster Comput.*, vol. 22, no. S5, pp. 10905–10913, Sep. 2019.

[24] B. Kruekaew and W. Kimpan, "Enhancing of artificial bee colony algorithm for virtual machine scheduling and load balancing problem in cloud computing," *Int. J. Comput. Intell. Syst.*, vol. 13, no. 1, p. 496, 2020.

[25] H. Singh and R. Randhawa, "Cuckoo search based workflow scheduling on heterogeneous cloud resources," in *Proc. 7th Int. Conf. Cloud Comput., Data Sci. Eng. Confluence*, Jan. 2017, pp. 65–70.
[26] Y. Xu, K. Li, L. He, and T. K. Truong, "A DAG scheduling scheme on heterogeneous computing systems using double molecular structure-based chemical reaction optimization," *J. Parallel Distrib. Comput.*, vol. 73, no. 9, pp. 1306–1322, Sep. 2013.
[27] S. R. Thennarasu, M. Selvam, and K. Srihari, "A new whale optimizer for workflow scheduling in cloud computing environment," *J. Ambient Intell. Humaniz. Comput.*, vol. 12, no. 3, pp. 3807–3814, Mar. 2021.
[28] B. H. Abed-Alguni and N. A. Alawad, "Distributed grey wolf optimizer for scheduling of workflow applications in cloud environments," *Appl. Soft Comput.*, vol. 102, Apr. 2021, Art. no. 107113.
[29] M. I. Daoud and N. Kharma, "A hybrid heuristic–genetic algorithm for task scheduling in heterogeneous processor networks," *J. Parallel Distrib. Comput.*, vol. 71, no. 11, pp. 1518–1531, Nov. 2011.
[30] C. Wang, J. Gu, Y. Wang, and T. Zhao, "A hybrid heuristic-genetic algorithm for task scheduling in heterogeneous multi-core system," in *Proc. Int. Conf. Algorithms Archit. Parallel Process.*, Sep. 2012, pp. 153–170.
[31] H. Aziza and S. Krichen, "A hybrid genetic algorithm for scientific workflow scheduling in cloud environment," *Neural Comput. Appl.*, vol. 32, no. 18, pp. 15263–15278, Sep. 2020.
[32] S.-J. Xue and W. Wu, "Scheduling workflow in cloud computing based on hybrid particle swarm algorithm," *TELKOMNIKA Indonesian J. Electr. Eng.*, vol. 10, no. 7, pp. 1560 –1566, Nov. 2012.
[33] A. Verma and S. Kaushal, "A hybrid multi-objective particle swarm optimization for scientific workflow scheduling," *Parallel Comput.*, vol. 62, pp. 1–19, Feb. 2017.
[34] N. Mansouri, B. M. H. Zade, and M. M. Javidi, "Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory," *Comput. Ind. Eng.*, vol. 130, pp. 597–633, Apr. 2019.
[35] N. Arora and R. K. Banyal, "Workflow scheduling using particle swarm optimization and gray wolf optimization algorithm in cloud computing," *Concurrency Comput., Pract. Exper.*, vol. 33, no. 16, Aug. 2021, Art. no. e6281.
[36] 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.
[37] M.-A. Vasile, F. Pop, R.-I. Tutueanu, V. Cristea, and J. Kołodziej, "Resource-aware hybrid scheduling algorithm in heterogeneous distributed computing," *Future Gener. Comput. Syst.*, vol. 51, pp. 61–71, Oct. 2015.
[38] M. Abdullahi and M. A. Ngadi, "Hybrid symbiotic organisms search optimization algorithm for scheduling of tasks on cloud computing environment," *PLoS ONE*, vol. 11, no. 6, Jun. 2016, Art. no. e0158229.
[39] D. Gabi, A. S. Ismail, A. Zainal, Z. Zakaria, and A. Al-Khasawneh, "Hybrid cat swarm optimization and simulated annealing for dynamic task scheduling on cloud computing environment," *J. Inf. Commun. Technol.*, vol. 17, no. 3, pp. 435–467, 2018.
[40] A. Choudhary, I. Gupta, V. Singh, and P. K. Jana, "A GSA based hybrid algorithm for bi-objective workflow scheduling in cloud computing," *Future Gener. Comput. Syst.*, vol. 83, pp. 14–26, Jun. 2018.
[41] A. Mohammadzadeh, M. Masdari, F. S. Gharehchopogh, and A. Jafarian, "A hybrid multi-objective metaheuristic optimization algorithm for scientific workflow scheduling," *Cluster Comput.*, vol. 24, no. 2, pp. 1479–1503, Jun. 2021.
[42] Y. Kumar, N. Dahiya, S. Malik, and S. Khatri, "A new variant of teaching learning based optimization algorithm for global optimization problems," *Informatica*, vol. 43, no. 1, p. 65, Mar. 2019.
[43] A. Taheri, K. RahimiZadeh, and R. V. Rao, "An efficient balanced teaching-learning-based optimization algorithm with individual restarting strategy for solving global optimization problems," *Inf. Sci.*, vol. 576, pp. 68–104, Oct. 2021.
[44] A. K. Shukla, "Chaos teaching learning based algorithm for large-scale global optimization problem and its application," *Concurrency Comput., Pract. Exper.*, vol. 34, no. 1, Jan. 2022, Art. no. e6514.
[45] D. Lei, L. Gao, and Y. Zheng, "A novel teaching-learning-based optimization algorithm for energy-efficient scheduling in hybrid flow shop," *IEEE Trans. Eng. Manag.*, vol. 65, no. 2, pp. 330–340, May 2018.
[46] R. Buddala and S. S. Mahapatra, "Two-stage teaching-learning-based optimization method for flexible job-shop scheduling under machine breakdown," *Int. J. Adv. Manuf. Technol.*, vol. 100, nos. 5–8, pp. 1419–1432, Feb. 2019.
[47] L. Jin, C. Zhang, X. Wen, C. Sun, and X. Fei, "A neutrosophic set-based TLBO algorithm for the flexible job-shop scheduling problem with routing flexibility and uncertain processing times," *Complex Intell. Syst.*, vol. 7, no. 6, pp. 2833–2853, Dec. 2021.
[48] H. Tang, B. Fang, R. Liu, Y. Li, and S. Guo, "A hybrid teaching and learning-based optimization algorithm for distributed sand casting job-shop scheduling problem," *Appl. Soft Comput.*, vol. 120, May 2022, Art. no. 108694.
[49] Y. Zhang, Z. Jin, and Y. Chen, "Hybrid teaching–learning-based optimization and neural network algorithm for engineering design optimization problems," *Knowl.-Based Syst.*, vol. 187, Jan. 2020, Art. no. 104836.
[50] C. Zhong, G. Li, and Z. Meng, "A hybrid teaching–learning slime mould algorithm for global optimization and reliability-based design optimization problems," *Neural Comput. Appl.*, vol. 34, no. 19, pp. 16617–16642, Oct. 2022.
[51] R. V. Rao and V. Patel, "Multi-objective optimization of two stage thermoelectric cooler using a modified teaching–learning-based optimization algorithm," *Eng. Appl. Artif. Intell.*, vol. 26, no. 1, pp. 430–445, Jan. 2013.
[52] T. Dokeroglu, "Hybrid teaching–learning-based optimization algorithms for the quadratic assignment problem," *Comput. Ind. Eng.*, vol. 85, pp. 86–101, Jul. 2015.
[53] S. D. K. Ram, S. Srivastava, and K. K. Mishra, "A variant of teaching-learning-based optimization and its application for minimizing the cost of workflow execution in the cloud computing," *Concurrency Comput., Pract. Exper.*, vol. 33, no. 21, Nov. 2021, Art. no. e6425.
[54] P. Mukhopadhyay, S. Dutta, and P. K. Roy, "Optimal location of TCSC using opposition teaching learning based optimization," *Int. J. Energy Optim. Eng.*, vol. 4, no. 1, pp. 85–101, Jan. 2015.
[55] W. Shao, D. Pi, and Z. Shao, "An extended teaching-learning based optimization algorithm for solving no-wait flow shop scheduling problem," *Appl. Soft Comput.*, vol. 61, pp. 193–210, Dec. 2017.
[56] Y. Xu, Z. Yang, X. Li, H. Kang, and X. Yang, "Dynamic opposite learning enhanced teaching–learning-based optimization," *Knowl.-Based Syst.*, vol. 188, Jan. 2020, Art. no. 104966.
[57] F. Liu, C. Liu, Q. Zhao, and C. He, "A hybrid teaching-learning-based optimization algorithm for the travel route optimization problem alongside the urban railway line," *Sustainability*, vol. 13, no. 3, p. 1408, Jan. 2021.
[58] (2022). *Workflow Management System*. [Online]. Available: https://pegasus.isi.edu/
[59] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, Mar. 2011.

**JIEGUANG HE** received the Ph.D. degree in mechanical engineering from the Guangdong University of Technology, in 2015. He is currently an Associate Professor with the Guangdong University of Petrochemical Technology. He has published more than 30 papers in relevant journals and conferences. His current research interests include cloud computing, intelligent optimization algorithms, and project scheduling.

**XIAOLI LIU** received the M.E. degree in public administration from Nanchang University, in 2022. She is currently a Research Intern with the Guangdong University of Petrochemical Technology. Her current research interests include business management and intelligent optimization algorithms.