IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 14, 2019, accepted October 9, 2019, date of publication October 21, 2019, date of current version October 30, 2019.

*Digital Object Identifier 10.1109/ACCESS.2019.2948468*

# A Novel Firefly Algorithm for Scheduling Bag-of-Tasks Applications Under Budget Constraints on Hybrid Clouds

**YI ZHANG, JUNLONG ZHOU, (Member, IEEE), LULU SUN, JINGJING MAO, AND JIN SUN, (Member, IEEE)**
School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China

Corresponding author: Jin Sun (sunj@njust.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61872185 and Grant 61802185, in part by the Natural Science Foundation of Jiangsu Province of China under Grant BK20180470, in part by the Jiangsu Planned Projects for Postdoctoral Research Funds under Grant 2019K025, and in part by the Jiangsu Provincial Key Research and Development Program under Grant BE2016768.

**ABSTRACT** This paper aims at scheduling bag-of-tasks (BoT) applications under budget constraints on hybrid clouds for minimizing the makespan. To solve this NP-hard problem, we propose a novel firefly algorithm (NFA) in which the evaluation of a firefly consists of two steps: (1) mapping a firefly to a scheduling solution (a task sequence); (2) calculating the solution's objective (its corresponding makespan). In the first step, different from the well-known ranked-order value (ROV) rule, we propose a distance-based mapping operator that relies on the distance between a firefly and the brightest one to determine the mapping relationship between a firefly and a solution. We use a probability model in which solutions corresponding to fireflies closer to the brightest one would have higher probabilities to inherit tasks from the current best solution. In this manner, these solutions can inherit more ''good genes'' hidden inside the current best solution to evolve into more high-quality solutions. In the second step, we employ an effective heuristic to evaluate solution objectives. We further develop a composite heuristic to generate the initial best solution, providing the proposed NFA with a good start. We also establish a new movement scheme such that fireflies distant from the brightest one can explore a wide range in the search space, whereas fireflies nearby the brightest one can search in a small neighborhood. Experimental results show that, by employing the above-mentioned strategies, NFA outperforms the standard firefly algorithm and the existing best algorithm, in terms of scheduling effectiveness and computational efficiency. Specifically, the distance-based mapping operator is verified to be both more effective and more efficient than the ROV rule. The composite heuristic is capable of generating a good initial solution, leading to the high quality of the final schedule. The movement scheme can further reduce the makespan of BoT applications.

**INDEX TERMS** Bag-of-tasks applications, hybrid clouds, makespan, firefly algorithms.

## I. INTRODUCTION

Cloud computing is a popular distributed computing paradigm that can deliver a massive amount of computing resources in a pay-as-you-go manner. In other words, users only need to pay what they have actually used. As a result, users can temporally use computing resources of public clouds while the private cloud cannot provide sufficient resources. In order to achieve this target, cloud vendors offer hybrid cloud solutions that integrates a private cloud with public clouds seamlessly. As illustrated in Figure 1, with the aid of hybrid cloud solutions, administrators or programs of the private cloud can manage the computing resources of the hybrid cloud (i.e., the private cloud and public clouds) via unified interfaces. For example, in case that the available computing resources of the private cloud cannot afford an instance of a ''large'' virtual machine (VM) type for processing a specific task, an instance of the same VM type can be

The associate editor coordinating the review of this manuscript and approving it for publication was Asad Waqar Malik.

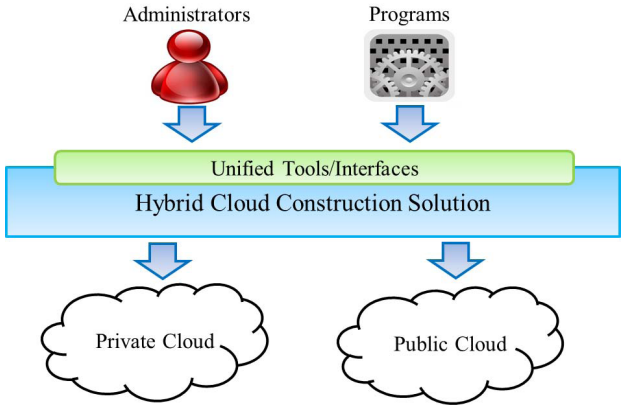


**FIGURE 1.** An illustrative example of hybrid cloud environment.

created on a public cloud to tackle this task. In addition, many cloud providers now can deliver and charge VM instances in seconds (e.g., QingCloud[1] and TecentCloud[2]) instead of in hours. This new charging mechanism makes hybrid clouds more popular for cloud customers, since they would not waste a certain fraction of the last hour that has to be paid entirely in traditional charging mechanisms.

Bag-of-tasks (BoT) applications consist of many independent tasks that can be processed in parallel without synchronization [1]. BoT applications are widespread in a variety of fields,such as computer imaging, parameter sweeps, and big data applications [2]–[4]. For instance, in parameter sweep application, a program have to be executed repeatedly and independently with different input configurations. Cloud computing represents a natural solution for executing BoT applications, considering its high-performance computing capability as well as flexible pricing policy. In hybrid cloud environments, if the private cloud cannot provide sufficient resources to accomplish all applications, it is a wise choice for users to outsource certain BoT applications and process them on public clouds. Under this circumstance, the key challenge is how to optimally schedule BoT applications' tasks given a user-specified budget for public cloud usage, such that the completion time of all tasks (a.k.a. makespan) can be minimized.

The focus of this paper is to address the above-mentioned BoT scheduling problem under budget constraints on hybrid clouds to minimize the makespan, which can be formulated as a NP-hard integer programm (IP) [5], [6]. We propose a novel firefly algorithm (NFA) to solve the considered problem, in which a task sequence is used to represent a scheduling solution. The standard firefly algorithms (SFA) is an evolutionary algorithm originally developed for solving continuous optimization problems [7]. The proposed NFA adopts the representation of SFA, i.e., each firefly is represented by a fixed-dimensional tuple, in which each element is a real number. Accordingly, we propose a two-step procedure to evaluate a firefly. In the first step, each firefly is mapped onto a scheduling solution (i.e. a task sequence). Different from the well-known ranked-order value (ROV) rule [8],[3] we propose a distance-based mapping operator that relies on the distance between a firefly and the brightest one to determine the mapping relationship between a firefly and a solution. During the evolving procedure, each firefly moves toward the brightest firefly which indicates the best solution. For this reason, the proposed mapping operator uses a probability model in which solutions corresponding to fireflies closer to the brightest one would have higher probabilities to inherit tasks from the current best solution. In this manner, these solutions can inherit more "good genes" hidden inside the current best solution to evolve into more high-quality solutions. The second step is to calculate the solution's objective (i.e., its corresponding makespan) by an existing effective heuristic. Meanwhile, we develop a composite heuristic to generate the initial best solution, providing the proposed NFA with a good start. We also establish a new movement scheme such that fireflies distant from the brightest one can explore a wide range in the search space, whereas fireflies nearby the brightest one can search in a small neighborhood. This scheme is for the purpose of maintaining a high probability in the mapping operator to inherit most good genes in the current best solution when they are close to the brightest one. Experimental results show that, by employing the above-mentioned strategies, the proposed NFA outperforms the standard firefly algorithm and the existing best algorithm [5] when solving the considered problem, in terms of scheduling effectiveness and computational efficiency. Specifically, the distance-based mapping operator is verified to be both more effective and more efficient compared with the ROV rule. The composite heuristic is capable of generating a good initial solution, leading to the high quality of the final scheduling solution. The movement scheme can further improve the effectiveness of NFA in scheduling BoT applications.

The rest of this paper is organized as follows. Section II investigates existing approaches related to this work. Section III formulates the optimization model for the considered BoT scheduling problem. Section IV detailed the proposed NFA for solving the formulated problem. Section V presents our experimental setup and simulation results. Finally, the concluding remarks are summarized in Section VI.

## II. RELATED WORK

In the literature, a considerable number of efforts have been made to study BoT scheduling on clouds [2], [5], [9]–[31]. Since the attention of this paper is paid to scheduling BoT application in hybrid cloud environments, in this section we briefly discuss existing scheduling methods related to hybrid clouds.

[1]https://www.qingcloud.com/
[2]https://www.qcloud.com/

[3]ROV rule is also referred to as smallest position value (SPV) rule in the literature.

In [22], the authors addressed a deadline-constrained BoT scheduling problem and formulated it as an IP. Two effective heuristics were constructed in [23] to solve the scheduling problem considering the cost in both computation and data transmission, whereas similar heuristics were introduced in [24]. Different from other works regarding private clouds as resources pools, Wang *et al.* [26] considered private clouds as multiple discrete physical machines, and designed a greedy heuristic to schedule tasks among these physical machines. The problem of scheduling BoT applications with varying tasks' runtimes was studied in [27]. A method for estimating tasks' runtimes was developed to update the scheduling results according to the estimated task durations. Abdi *et al.* [28] considered BoT scheduling on a hybrid cloud platform and employed CPLEX solver to solve the problem. The authors in [29] and [30] employed particle swarm optimization algorithms to schedule BoT applications on hybrid clouds under deadline constraints.

Different from the cost-minimization deadline-constrained approaches summarized above, our previous work [5] studied makespan-minimization budget-constrained BoT scheduling problem on hybrid clouds. This scheduling problem was formulated as an IP and was solved by a fast iterated local search (FILS) algorithm. This work addresses BoT scheduling problem in the same scenario. However, we propose a novel scheduling algorithm NFA that outperforms FILS in terms of both effectiveness and efficiency.

A firefly algorithm (FA) is a novel metaheuristic inspired by the social behaviors of fireflies. FA was originally designed to solve continuous optimization problems [7], [32]–[34], and can be extended to cope with discrete optimization problems. In [35] and [36], two discrete FAs were proposed to solve the well-known traveling salesman problem (TSP). In these two FAs, fireflies are represented as permutations of city names, and are regarded as solutions. Two movement schemes were proposed based on the firefly representation to modify the permutations. Distance between two fireflies is defined as the number of different edges between them. Li *et al.* [37] used similar representation and distance definition to construct a new FA for solving asymmetric multi-depot vehicle routing problem. In [38], an improved FA was presented to solve the unrelated parallel machines scheduling problem with sequence-dependent setup times. A special movement scheme was proposed for updating fireflies by employing adjusted processing times matrix. In these FAs, each firefly represents a solution to the problem and can thus be evaluated directly. However, it is necessary to design specific movement schemes that are applicable for the corresponding firefly representations.

Two FAs were proposed in [39] and [40] to address the placement problem of active power conditioners. In these two FAs, fireflies were encoded in real numbers and then mapped onto solutions represented in binary notations. Two mapping operators were proposed to achieve this target. In [8], a FA encoded in real numbers was proposed to handle multi-objective hybrid flowshop scheduling problems. The ROV rule was used to map a firefly to a permutation of jobs, which can be regarded as a solution to the problem. In these real-encoded FAs, mapping operators are required for evaluating fireflies. For these problems whose solutions are represented in permutations, ROV is widely accepted and used. However, sorting-based ROV is time-consuming when solving large-scale problems. In this paper, we propose a distance-based mapping operator, in which the current best solution is taken into account and its ''good genes'' can be inherited by subsequent solutions. As a result, more high-quality solutions can be explored during the search procedure. Moreover, the proposed mapping operator is not sorting-based, and therefore has a lower complexity than ROV does.

## III. SCHEDULING MODEL

In this section, we present in detail the optimization model for the considered BoT scheduling problem, in which several variable definitions are similar to those in [41], [42]. We consider a hybrid cloud environment that includes a private cloud $CP_0$ and $m$ public clouds $CP_1, CP_2, \cdots, CP_m$. The private cloud and all public clouds can provide $k$ VM types $VM_1, VM_2, \cdots, VM_k$. Each VM type is associated with a CPU capacity $CPU_q$ and a memory capacity $Mem_q$. Users can use the resources of the private cloud for free. However, if tasks are outsourced to be processed on public clouds, an extra cost would be incurred. We use $p_{qh}$ $(q = 1, 2, \cdots, k;\ h = 0, 1, \cdots, m)$ to represent the unit price for using a $VM_q$ instance on a cloud provider $CP_h$. We have $p_{q0} = 0$ for the private cloud since its resources are free to use.

There are $n$ applications to be scheduled on hybrid clouds, and each application consists of $T_i$ tasks. The total number of tasks is $T = \sum_{i=1}^{n} T_i$. Similar to the scheduling model in [29], each application $a_i$ requires a VM type specified by the user. In other words, all tasks belonging to this application require a same VM type. For each task $t_j$ $(j = 1, 2, \cdots, T)$, we define a binary variable $x_{jq}$ to denote the mapping relationship between this task and the VM type it demands. $x_{jq} = 1$ indicates $t_j$ demands $VM_q$, and $x_{jq} = 0$ otherwise. Note that all tasks in an application share the same value of $x_{jq}$. For each task $t_j$, we define $r_j$ to denote its execution duration. Following previous studies [22]–[24], [29], we assume that each task can be executed in only one VM instance and each VM instance can only process one task at any time slot, in order to avoid interference or resource contention. We further assume that the tasks are executed continuously without any preemption. The setup times for VM instances are assumed to be zero. As mentioned previously, cloud providers nowadays can deliver VM instances in seconds. Therefore, the time overhead for creating a VM instance can be neglected when compared with task runtimes (generally in hours). In addition, considering that public cloud providers can charge resources in seconds as well, the entire time frame for the scheduling problem is divided into time slots with the granularity of one second. Specifically for the private cloud, the amount of resources consumed by BoT applications cannot exceed the upper-bound limit $CPU^*$ and $Mem^*$.

With all above-mentioned definitions, the completion time of all tasks (i.e., the makespan) $c_{\max}$ can be determined as the largest value among all the tasks' completion times, i.e.,

$$c_{\max} = \max\left\{c_j;\ j=1, 2, \cdots, T\right\}, \tag{1}$$

where $c_j$ denotes the completion time of task $t_j$, and can be calculated by

$$c_j = st_j + r_j. \tag{2}$$

In this equation, $st_j$ indicates the start time of task $t_j$. We further use $S = c_{\max}$ to denote the maximum number of time slots for all tasks.

Two decision variables are required to formulate the scheduling model. Binary variable $y_{jh} = 1$ means that task $t_j$ is dispatched to $CP_h$, and $y_{jh} = 0$ otherwise. Binary variable $z_{js} = 1$ denotes that $t_j$ is in execution at time slot $s$ on the private cloud, and $z_{js} = 0$ otherwise. Based on these two binary variables, $t_j$'s start time $st_j$ can be determined as

$$st_j = \begin{cases} \arg\min\left\{s|z_{js}=1\right\}, & \text{if } y_{j0} = 1. \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Obviously, if task $t_j$ is dispatched to the private cloud (i.e., $y_{j0} = 1$), its start time $st_j$ would be $\arg\min\{s|z_{js} = 1\}$. To be specific, assume that task $t_j$ is in execution on a VM instance at time slots 3, 4, and 5. Its start time $st_j$ is therefore $\arg\min\{3, 4, 5\} = 3$. On the other hand, if task $t_j$ is outsourced to a public cloud, it can start execution at time slot 0 since the setup times for VM instances are negligible and are assumed to be 0.

The total cost for executing all BoT applications can be then determined as

$$Cost = \sum_{j=1}^{T}\sum_{h=0}^{m}\sum_{q=1}^{k} x_{jq}y_{jh}r_j p_{qh}. \tag{4}$$

This cost cannot exceed a user-specified budget limit $B$.

Putting it all together, we can formulate the considered scheduling problem as the following IP.

Minimize the makespan:

$$c_{\max} = \max\left\{c_j;\ j=1, 2, \cdots, T\right\} \tag{5}$$

Subject to:

$$c_j = st_j + r_j, \quad j = 1, 2, \cdots, T \tag{6}$$

$$\sum_{j=1}^{T}\sum_{h=0}^{m}\sum_{q=1}^{k} x_{jq}y_{jh}r_j p_{qh} \le B \tag{7}$$

$$\sum_{h=0}^{m} y_{jh} = 1, \quad j = 1, 2, \cdots, T \tag{8}$$

$$\sum_{q=1}^{k}\sum_{j=1}^{T} z_{js}x_{jq}CPU_q \le CPU^*, \quad s = 0, 1, \cdots, S \tag{9}$$

$$\sum_{q=1}^{k}\sum_{j=1}^{T} z_{js}x_{jq}Mem_q \le Mem^*, \quad s = 0, 1, \cdots, S \tag{10}$$

The scheduling objective in Equation (5) is to minimize the makespan. Equation (6) ensures that tasks are executed consecutively. Equation (7) imposes the budget constraint. Equation (8) is to guarantee the uniqueness of mapping from tasks to clouds. Equations (9) and (10) impose a set of constraints that the amount of consumed resources cannot exceed the corresponding resource limit. For ease of reference, all associated parameters are listed in Table 1.

## IV. PROPOSED NOVEL FIREFLY ALGORITHM (NFA)

The proposed NFA follows the evolutionary mechanism of SFAs, inspired by social behaviors of fireflies. A firefly uses flash to attract other fireflies. According to [7], [32], [33], for any two fireflies, the firefly's attractiveness is proportional to its brightness (or called light intensity). The less bright one will be attracted by the brighter one and will move toward the brighter one. Also, the brightness of a firefly is associated with the objective of the considered problem. In other words, less bright fireflies with worse objective values are attracted by other brighter fireflies with better objective values. All fireflies will be eventually around the brightest firefly with the best objective value.

Different from SFAs that were originally proposed for solving continuous optimization problems, in order to map real-encoded fireflies to solutions (i.e., task sequences), our proposed NFA employs a distance-based mapping operator, which not only considers the distance between a firefly and the brightest one, but also takes into account the current best solution. The mapped solutions can then be evaluated by an efficient task assignment strategy FTA [5]. For the purpose of giving NFA a good start, we construct an effective composite heuristic to generate a highly-qualified solution that is regarded to be the initial best solution. More importantly, we develop a new movement scheme that can maintain a high probability in the proposed mapping operator to inherit most "good genes" in the current best solution to generated solutions when they are close to the brightest one. The flowchart of the proposed NFA is illustrated in Figure 2. In what follows, we discuss in detail the proposed mapping operator, composite heuristic and movement scheme, which are all fundamental procedures in NFA.

### A. SOLUTION REPRESENTATION AND EVALUATION

Task sequences are permutations of all tasks and considered as solutions in this paper. For example, given four tasks $t_1$, $t_2$, $t_3$, and $t_4$ in total, the following three permutations of tasks $\pi_1 - \pi_3$ are all valid solutions: $\pi_1 = (t_1, t_2, t_3, t_4)$, $\pi_2 = (t_3, t_1, t_2, t_4)$ and $\pi_3 = (t_2, t_1, t_4, t_3)$. Meanwhile, given a solution $\pi$ with the total number of tasks $T$, we define $\pi_{[j]} (j = 1, 2, \cdots, T)$ to be the $j$-th task in it. For instance, $\pi_{1[1]} = t_1$, $\pi_{2[2]} = t_1$ and $\pi_{3[3]} = t_4$.

For a firefly $i$ ($i = 1, 2, \cdots, |\Omega|$) where $\Omega$ is the population, its location $X_i$ can be represented as a $T$-dimensional tuple, i.e., $X_i = \{X_{i1}, X_{i2}, \cdots, X_{iT}\}$ where $X_{il} (l = 1, 2, \cdots, T)$ is a real number. Each firefly can be mapped onto a solution by our proposed mapping operator. In addition, the brightness

**TABLE 1. Notations for problem formulation.**

| Notation | Definition | Notation | Definition |
|---|---|---|---|
| $c_{\max}$ | the makespan of the application | $CP_h$ | the $h$-th cloud provider |
| $m$ | the total number of public clouds | $VM_q$ | the $q$-th VM type |
| $k$ | the total number of VM types | $CPU_q$ | the number of CPUs on $VM_q$ |
| $Mem_q$ | the amount of memory on $VM_q$ | $p_{qh}$ | the unit price of $VM_q$ provided by $CP_h$ |
| $a_i$ | the $i$-th application | $n$ | the total number of applications |
| $T_i$ | the number of tasks for application $a_i$ | $T$ | the total number of tasks |
| $t_j$ | the $j$-th task | $r_j$ | the runtime of $t_j$ |
| $st_j$ | the start time of $t_j$ | $c_j$ | the completion time of $t_j$ |
| $CPU^*/Mem^*$ | CPU/memory capacity of the private cloud | $s$ | a time slot |
| $B$ | the budget for cloud usage cost | $x_{jq}$ | a variable: $x_{jq}=1$ if $t_j$ demands $VM_q$, and $x_{jq}=0$ otherwise. |
| $y_{jh}$ | a decision variable: $y_{jh}=1$ if $t_j$ is allocated to $CP_h$, and $y_{jh}=0$ otherwise. | $z_{js}$ | a decision variable: $z_{js}=1$ if $t_j$ is in execution at slot $s$ on the private cloud, and $z_{js}=0$ otherwise. |

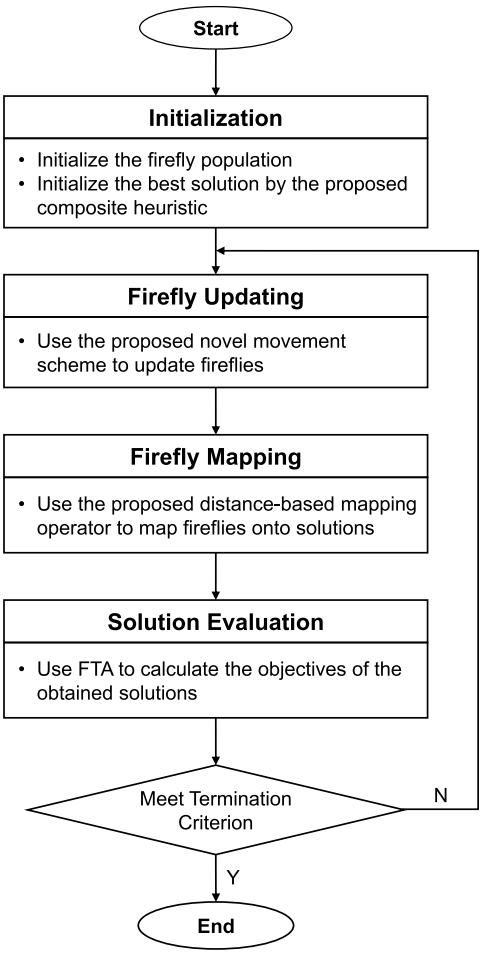


**FIGURE 2. The flowchart of the proposed NFA.**

of a firefly is associated with the optimization objective [7], [32], [33]. As the objective of our considered problem is to minimize of the makespan, the brightness (or light intensity) $I_i$ of a firefly $i$ is defined as the reciprocal of its corresponding solution's objective, i.e., $I_i = 1/f(\pi_i)$ where $f$ is the objective function, and $\pi_i$ is the firefly's corresponding solution obtained by the proposed mapping operator. We employ the fast task assignment (FTA) strategy presented in our previous work [5] to evaluate scheduling solutions (i.e., to calculate solutions' objective values), due to its good effectiveness and efficiency. In other words, we have $f(\pi_i) = \text{FTA}(\pi_i)$.

### B. DISTANCE-BASED MAPPING OPERATOR

Fireflies are mapped onto solutions by a distance-based mapping operator that takes into account the distance between the mapped firefly and the brightest one, and the current best solution. This feature is distinct from the existing mapping operators (e.g., ROV) that only depend on mapped fireflies' positions. The proposed mapping operator is inspired by the evolving principle of FAs, i.e., during the evolving procedure, each firefly moves close to the brightest one corresponding to the best solution. In other words, fireflies that are closer to the brightest one (i.e., having shorter distances) may be mapped to better solutions in larger possibilities. Thus, the mapping operator uses a probability model, in which solutions corresponding to fireflies closer to the brightest one have bigger probabilities to inherit tasks from the current best solution, directly. As a result, more ''good genes'' hidden inside of the current best solution could be inherited by these solutions, beneficial for exploring more high-quality solutions.

Let $\pi^*$ and $r_i^*$ be the current best solution and the distance between the firefly $i$ and the brightest one, respectively. For a firefly $i$, the mapping operator first creates an empty solution $\pi_i$ and then constructs $\pi_i$ to be a complete solution by two steps. In Step 1, each task in $\pi^*$ is appended to $\pi_i$ with the probability $p = e^{-r_i^*}$. As $r_i^* \geq 0$, we have $p \in (0, 1]$. Afterwards, all the tasks that have been appended into $\pi_i$ are removed from $\pi^*$. Accordingly, the smaller $r_i^*$ is, the bigger $p$ is. In other words, solutions corresponding to fireflies closer to the brightest one have bigger probabilities to inherit tasks from the current best solution, directly. Consequently, more

**Algorithm 1** Distance-Based Mapping Operator

**Input**: $r_i^*$ and $\pi^*$;
**Output**: A solution $\pi_i$;
1 Create an empty solution $\pi_i$;
2 **for** $j = 1$ *to* $T$ **do**
3  Generate a uniform random number $\lambda \in [0, 1]$;
4  **if** $\lambda \leq e^{-r_i^*}$ **then**
5  Append $\pi^*_{[j]}$ to $\pi_i$;
6 Remove all the tasks that have been appended into $\pi^*$ from $\pi^*_{temp}$;
7 **for** $j = 1$ *to* $|\pi^*|$ **do**
8  Add $\pi^*$ into $\pi_i$ in a uniform distribution;
9 **return** $\pi_i$;

''good genes'' hidden inside of the current best solution could be inherited by these solutions. This is the reason for the good effectiveness of the proposed mapping operator. In Step 2, each left task in $\pi^*$ is added into $\pi_i$ in a uniform distribution. In other words, each task $j$ in $\pi^*$ is added to each position in $\pi_i$ with the identical probability of $\frac{1}{|\pi_i|}$. Finally, $\pi_i$ becomes a complete solution and is returned. The algorithmic procedure of the mapping operator is presented in Algorithm 1. Lines 2-5 denote Step 1, while Lines 7-8 correspond to Step 2.

The complexity of the proposed mapping operator is $\mathcal{O}(T)$. In comparison, the ROV rule is sorting-based and has a complexity of $\mathcal{O}(T \cdot \log T)$ in the best case. We draw the conclusion that our mapping operator is more computationally efficient. Since solutions are constructed from the current best solution $\pi^*$ based on the probability $p$, if all the obtained solutions are identical to $\pi^*$, NFA loses its exploration capability and traps into the local optimum. Therefore, it is necessary to discuss the probability that the obtained solution is identical to $\pi^*$. We prove that the following theorem is true.

*Theorem 1:* For a firefly $i$ that is not identical to the brightest firefly, when it is mapped onto a solution $\pi_i$ by the proposed mapping operator based on the current best solution $\pi^*$, we have $\lim_{T\to\infty} P(\pi_i = \pi^*) = 0$.

*Proof:* There are two independent cases which could lead to $\pi_i = \pi^*$. (1) All the tasks in $\pi^*$ are inherited by $\pi_i$ in Step 1. (2) Part of tasks in $\pi^*$ are inherited by $\pi_i$ and removed from $\pi^*$ in Step 1, and the other left tasks are inserted into $\pi_i$ at their original positions in Step 2. Let the probabilities of these two cases be $P_1$ and $P_2$, respectively. Then $P(\pi_i = \pi^*) = P_1 + P_2$.

(1) In the first case, since all the tasks in $\pi^*$ are inherited by $\pi_i$ in Step 1, we have $P_1 = p^T$.

(2) Assume $L$ to be the number of tasks that are inherited from $\pi^*$ by $\pi_i$. The expectation of $L$ $E(L)$ should be $T \cdot p$. Let $\mu$ be the probability that all left tasks in $\pi^*$ are occasionally inserted at their original positions. Then, $\mu = 1/A_T^{T-L}$, where $A_T^{T-L}$ be the number of $(T - L)$-length permutations of $T$ tasks. So, we have $P_2 = \mu p^L (1-p)^{T-L}$. Therefore, we have $P(\pi_i = \pi^*) = P_1 + P_2 = p^T + \mu p^L (1-p)^{T-L}$. Due to $E(L) = T \cdot p$, the following equation is true.

$$P(\pi_i = \pi^*) = p^T + \mu p^{T \cdot p} (1-p)^{T \cdot (1-p)} \tag{11}$$

Accordingly, we have $\lim_{T\to\infty} P(\pi_i = \pi^*) = \lim_{T\to\infty} p^T + \lim_{T\to\infty} \mu p^{T \cdot p} (1-p)^{T \cdot (1-p)}$. As this firefly $i$ is not identical to the brightest firefly, we have $r_i^* \neq 0$. In other words, $0 < p < 1$, $0 < p^p < 1$, and $0 < (1-p)^{(1-p)} < 1$. Thus, $\lim_{T\to\infty} p^T = 0$, $\lim_{T\to\infty} \mu = 0$, $\lim_{T\to\infty} p^{T \cdot p} = \lim_{T\to\infty} (p^p)^T = 0$, and $\lim_{T\to\infty} (1-p)^{T \cdot (1-p)} = \lim_{T\to\infty} ((1-p)^{(1-p)})^T = 0$. Therefore, we have $\lim_{T\to\infty} P(\pi_i = \pi^*) = 0$. ■

Theorem 1 justifies that for large-scale problems, the probability of generating solutions identical to $\pi^*$ is substantially small (close to zero) regardless of $p$ value. However, for small-scale problems, if fireflies are extremely close to the brightest one (leading to a large $p$ value), the probability cannot be regardless according to Equation (11). In other words, these fireflies may be mapped to solutions that are remarkably similar (even identical) to $\pi^*$, and the diversification would be reduced. In order to avoid this terrible situation, we set the maximum of $p$ as 0.95.

### C. COMPOSITE HEURISTIC

As mentioned previously, the mapping operator relies on the current best solution. In order to give NFA a good start, we employ a composite heuristic to initialize a highly qualified best solution. The proposed composite heuristic consists of three steps. In the first step, an initial solution is generated by the well-known longest task first rule (LTF), in which the tasks to be scheduled are arranged by their durations in a non-ascending order.

In the second step, we use an insertion-based reconstruction method (IRM) to generate a new solution by reconstructing the obtained LTF result (denoted by $\pi_{LTF}$). We first create an empty solution $\pi_{IRM}$. The $j$-th task in $\pi_{LTF}$ is then inserted into $\pi_{IRM}$ at each possible position to generate $j$ candidate partial solutions, among which the best one is used to update $\pi_{IRM}$. Then, IRM handles the $(j+1)$-th task in the same manner. When all the tasks in $\pi_{LTF}$ have been inserted into $\pi_{IRM}$ and $\pi_{IRM}$ becomes a complete solution, IRM returns $\pi_{IRM}$ and terminates.

To be specific, we provide an example to explain the IRM procedure. Assume that $\pi_{LTF}$ has four tasks. In the first round, $\pi_{IRM}$ is set to empty, and $\pi_{LTF[1]}$ is inserted into $\pi_{IRM}$ with only one candidate partial solution obtained (i.e., $\pi^1_{IRM} = (\pi_{LTF[1]})$), which is naturally adopted to update $\pi_{IRM}$ (i.e., $\pi_{IRM} \leftarrow \pi^1_{IRM}$). In the second round, $\pi_{IRM} = (\pi_{LTF[1]})$ and $\pi_{LTF[2]}$ is inserted into $\pi_{IRM}$ with two candidate partial solutions achieved (i.e. $\pi^1_{IRM} = (\pi_{LTF[2]}, \pi_{LTF[1]})$ and $\pi^2_{IRM} = (\pi_{LTF[1]}, \pi_{LTF[2]})$). We assume that $\pi^1_{IRM}$ is better than $\pi^2_{IRM}$ and $\pi^1_{IRM}$ is adopted to update $\pi_{IRM}$ (i.e. $\pi_{IRM} \leftarrow \pi^1_{IRM}$), accordingly. In the third round, $\pi_{IRM} = (\pi_{LTF[2]}, \pi_{LTF[1]})$ and

$\pi_{LTF[3]}$ is inserted into $\pi_{IRM}$ with three candidate partial solution obtained (i.e. $\pi^1_{IRM} = (\pi_{LTF[3]}, \pi_{LTF[2]}, \pi_{LTF[1]})$, $\pi^2_{IRM} = (\pi_{LTF[2]}, \pi_{LTF[3]}, \pi_{LTF[1]})$ and $\pi^3_{IRM} = (\pi_{LTF[2]}, \pi_{LTF[1]}, \pi_{LTF[3]})$). We assume that $\pi^3_{IRM}$ is the best and $\pi^3_{IRM}$ is adopted to update $\pi_{IRM}$ (i.e., $\pi_{IRM} \leftarrow \pi^3_{IRM}$), accordingly. Finally, since all the tasks in $\pi_{LTF}$ has been processed and $\pi_{IRM}$ now becomes a complete solution, IRM returns $\pi_{IRM}$ and stops.

In the third step, we iterate a swap-based search method (SSM) to improve the result of IRM $\pi_{IRM}$ until no improvement can be obtained. In SSM, two new solutions $\pi_{SSM}$ and $\pi_{tem}$ are first created and are both set as $\pi_{IRM}$. Then, SSM swaps the $j$-th task in $\pi_{tem}$ with all its subsequent tasks and generates $(T - j)$ candidate solutions. If the best one among them is better than $\pi_{SSM}$, it will be used to update both $\pi_{tem}$ and $\pi_{SSM}$. Otherwise, $\pi_{tem}$ and $\pi_{SSM}$ remain unchanged. Then, SSM proceeds to handle the ($j$+1)-th task in $\pi_{SSM}$. Once the processing of the ($T$−1)-th task is completed, SSM returns $\pi_{SSM}$ as the result and terminates.

Likewise, we also use an example to clarify the SSM procedure. We assume that $T = 4$. Both $\pi_{tem}$ and $\pi_{SSM}$ are set as $\pi_{IRM} = (\pi_{IRM[1]}, \pi_{IRM[2]}, \pi_{IRM[3]}, \pi_{IRM[4]})$, initially. In the first round, $\pi_{tem[1]}$ (i.e. $\pi_{IRM[1]}$) is swapped with $\pi_{tem[2]}$, $\pi_{tem[3]}$ and $\pi_{tem[4]}$, respectively. The following three candidate solutions are generated:

$$\pi^1_{tem} = (\pi_{IRM[2]}, \pi_{IRM[1]}, \pi_{IRM[3]}, \pi_{IRM[4]}),$$
$$\pi^2_{tem} = (\pi_{IRM[3]}, \pi_{IRM[2]}, \pi_{IRM[1]}, \pi_{IRM[4]}),$$
$$\pi^3_{tem} = (\pi_{IRM[4]}, \pi_{IRM[2]}, \pi_{IRM[3]}, \pi_{IRM[1]}).$$

Assume that $\pi^3_{tem}$ is the best one among these three solutions, and is also better than $\pi_{SSM}$. We set $\pi_{SSM} \leftarrow \pi_{tem} \leftarrow \pi^3_{tem}$. In the second round, $\pi_{tem} = (\pi_{IRM[4]}, \pi_{IRM[2]}, \pi_{IRM[3]}, \pi_{IRM[1]})$ and $\pi_{tem[2]}$ is swapped with $\pi_{tem[3]}$ and $\pi_{tem[4]}$, respectively. We can further obtain the following two candidate solutions:

$$\pi^1_{tem} = (\pi_{IRM[4]}, \pi_{IRM[3]}, \pi_{IRM[2]}, \pi_{IRM[1]}),$$
$$\pi^2_{tem} = (\pi_{IRM[4]}, \pi_{IRM[1]}, \pi_{IRM[3]}, \pi_{IRM[2]}).$$

Assume that $\pi^2_{tem}$ is the better one, but is worse than $\pi_{SSM}$. In this case, $\pi_{SSM}$ remains unchanged and $\pi_{tem}$ is reset to $\pi_{SSM}$ (i.e., $\pi_{tem} \leftarrow \pi_{SSM}$). In the third round, $\pi_{tem} = \pi_{SSM} = (\pi_{IRM[4]}, \pi_{IRM[2]}, \pi_{IRM[3]}, \pi_{IRM[1]})$, $\pi_{tem[3]}$ is swapped with $\pi_{tem[4]}$ with one candidate solution obtained, i.e., $\pi^1_{tem} = (\pi_{IRM[4]}, \pi_{IRM[2]}, \pi_{IRM[1]}, \pi_{IRM[3]})$. Assume that $\pi^1_{tem}$ outperforms $\pi_{SSM}$. $\pi_{SSM}$ is set as $\pi^1_{tem}$, i.e., $\pi_{SSM} \leftarrow \pi^1_{tem}$. Finally, since the ($T$−1)-th task has been processed, SSM returns $\pi_{SSM}$ as the result and terminates.

The procedure of the composite heuristic is given in Algorithm 2. Line 1 denotes the first step, i.e., using LTF to generate a solution $\pi_{LTF}$, whereas Line 2 means the second step, i.e., employing IRM to reconstruct $\pi_{LTF}$ to obtain $\pi_{IRM}$. The loop in Lines 5-13 is the third step, i.e., iterating SSM to further improve $\pi_{IRM}$. $\pi_{CH}$ is used to record the best found solution and returned in Line 14 finally.

**Algorithm 2** Composite Heuristic

**Input**: A task set $\Psi$ including all applications' tasks;
**Output**: A solution $\pi_{CH}$;

```
1  π_LTF ←LTF(Ψ) ;          /* The first step */
2  π_IRM ←IRM(π_LTF) ;      /* The second step */
3  Set Improved ← TRUE;
4  Set π_CH ← π_tem ← π_IRM;
   /* The "while" loop is the third step
   */
5  while (Improved) do
6      Set Improved ← FALSE;
7      π_SSM ←SSM(π_tem);
8      if (π_SSM outperforms π_CH) then
9          π_tem ← π_SSM;
10         π_CH ← π_SSM;
11         Set Improved ← TRUE;
12     else
13         π_tem ← π_CH;
14 return π_CH;
```

We perform a brief complexity analysis as follows. As aforementioned, we employ FTA to evaluate all solutions. However, the complexity of FTA is difficult to determine explicitly (refer to [5] for details), and is denoted as $\mathcal{O}(FTA)$. The complexities of LTF, IRM and SSM are $\mathcal{O}(T \cdot \log T \cdot O(FTA))$, $\mathcal{O}(T \cdot (T+1) \cdot \mathcal{O}(FTA))$ and $\mathcal{O}(T \cdot (T-1) \cdot \mathcal{O}(FTA))$, respectively. The complexity of the composite heuristic is hard to determine, since we do not know the actual number of iterations in the "while" loop.

### D. NOVEL MOVEMENT SCHEME

The proposed novel movement scheme is constructed based on the standard movement scheme of SFA, which is introduced below. In the standard movement scheme, when a firefly $i$ moves toward a brighter firefly $j$, its positions are updated by the following equation.

$$X_i = X_i + \beta_0 e^{-\gamma r_{ij}^2}(X_j - X_i) + \alpha\left(\varepsilon - \frac{1}{2}\right). \quad (12)$$

where the first term denotes the current location of the firefly $i$, and the second one represents the movement of the firefly $i$ from its current location to a new one because of the attraction. $\beta_0 e^{-\gamma r_{ij}^2}$ is defined as the attractiveness function, and its value indicates the level that the firefly $j$ attracts the firefly $i$. The parameter $\beta_0$ represents the original attractiveness of fireflies, while $\gamma$ is the light absorption coefficient. $r_{ij}$ is the Euclidean distance between the two fireflies, and can be calculated by Equation (13). Generally, $\beta_0 \in [0, 1]$ as reported in [7], [32]–[34]. When $\beta_0 = 0$, only random search is employed and fireflies would not move toward the brightest one representing the best solution. On the other hand, while $\beta_0 = 1$, cooperative search is applied such that the brightest firefly uses its large light intensity for the purpose

of attracting other fireflies to move toward it. In other words, the brightest firefly affects the positions of other fireflies significantly. $\gamma$ reflects the variation of attractiveness corresponding to the variation of distance between the two fireflies. $\gamma = 0$ indicates constant attractiveness, whereas $\gamma = \infty$ means negligible attractiveness (close to zero), leading to a random search. According to the conclusion given in [7], [32], [33], in general, $\gamma \in [0.01, 100]$. The conclusion drawn in [34] shows that $\gamma \in [0, 10]$. As a result, we have $\gamma \in [0.01, 10]$. The third term is the randomization representing the random search, which allows a firefly to move randomly. $\alpha$ is a randomization parameter and $\varepsilon$ is a random number with the value uniformly distributed in [0, 1].

$$r_{ij} = \sqrt{\sum_{l=1}^{d} (X_{il} - X_{jl})^2}. \tag{13}$$

It is obvious that in the standard movement scheme, the value domain of the randomization is $[-\frac{1}{2}\alpha, \frac{1}{2}\alpha]$. Accordingly, once the value of $\alpha$ is initialized, the range of the random search is constant, without respect to the distance between the firefly and the brightest one. In fact, when a firefly is distant from the brightest one, we need to allow the firefly to search in a wide range. On the other hand, when a firefly is close to the brightest one, we need to allow the firefly to search in a little neighborhood around the brightest firefly in order to maintain a high probability $p$ in the proposed mapping operator. As a result, most good genes hidden inside of the current best solution can be inherited by the solutions generated by the mapping operator, benefiting the high qualities of the obtained solutions and improving NFA's effectiveness. According to this conclusion, we propose a new movement scheme, in which a firefly's position is updated by Equation (14) when this firefly moves toward a brighter firefly $j$.

$$X_i = X_i + \beta_0 e^{-\gamma r_{ij}^2}(X_j - X_i) + r_i^* \left(\varepsilon - \frac{1}{2}\right). \tag{14}$$

Equation (14) indicates that the value domain of the randomization is $[-\frac{1}{2}r_i^*, \frac{1}{2}r_i^*]$, relying on the value of $r_i^*$. Thus, when a firefly is far away from the brightest one (i.e., a great $r_i^*$ value), a wide searching range can be obtained. On the other hand, when a firefly is close to the brightest one (i.e., a small $r_i^*$ value), a small neighborhood around the brightest firefly can also be achieved.

### E. DESCRIPTION OF NFA

In the proposed NFA, the evaluation of fireflies is a two-step procedure that first maps fireflies onto solutions, and then employs FTA to calculate the corresponding makespan of mapped solutions. However, the mapping operator in NFA depends on the distance between the mapped firefly and the brightest one. In other words, the initialization of the brightest firefly relies on the mapping operator, which is in turn dependent on the determination of the brightest firefly. We employ a simple strategy to address this issue, by randomly selecting the brightest firefly from the initial population. Although this strategy cannot find the real brightest firefly, it can be discovered after all the fireflies in the initial population have been evaluated.

The overall flow of NFA is summarized in Algorithm 3. Line 2 is the population initialization. Line 3 defines the light intensity of the firefly $i$ as $1/f(\pi_i)$. Line 4 is the initialization of the best solution. In Line 5, the brightest firefly is selected from the initial population, randomly. The evaluation procedure of all the fireflies in the initial population is given in Lines 6-13, where we can see that the brightest firefly is updated if a new best solution is found. Lines 14-27 represent the evolving procedure. If firefly $j$ is brighter than firefly $i$ (i.e., $I_j > I_i$ in Line 17), firefly $i$ is required to move toward the firefly $j$ and its location $X_i$ will be updated by Equation (14). Note that $I_j > I_i$ indicates that the solution corresponding to firefly $j$ is better than that corresponding to firefly $i$, i.e., $f(\pi_j) < f(\pi_i)$.

## V. EXPERIMENTAL RESULTS

In this section, the performance of scheduling algorithms is evaluated and impacts of important factors are investigated by exhaustive simulation experiments.

### A. TESTING INSTANCES

The hybrid cloud environment used in experiments consists of a private cloud and four public clouds. Table 2 describes the resource capabilities and unit prices for various VM types provided by public clouds. It is worth mentioning that, the prices in Table 2 are charged in hours. When calculating the total cost in the scheduling algorithm, we need to convert them into unit prices charged in seconds. The two testing instance sets for parameter determination and performance evaluation were constructed in the same way as in our previous works [5]. The testing instances are organized into three groups in which the number of tasks to be scheduled are 20, 50, and 100, respectively. Each group is further composed of three subgroups representing small-size problem, medium-size problem, and large-size problem, respectively. The details about the construction of testing instances can be referred to [5].

The total cost of using cloud resources must satisfy a budget constraint. We use $VM_b$ to denote the best VM type among all available types, and $p_b$ is its price. The budget limit is accordingly specified by

$$B = T \times p_b \times \lambda, \tag{15}$$

where $\lambda$ is tuning parameter for adjusting the budget value and is referred to as *budget factor* in this work. $T$ denotes the number of tasks to be scheduled. Testing instances of larger sizes (i.e., with greater numbers of tasks) would be assigned higher budget values. For the private cloud with limited resources, its CPU and memory capacities are specified in a similar manner:

$$CPU^* = T \times CPU_b \times \rho, \tag{16}$$

$$Mem^* = T \times Mem_b \times \rho, \tag{17}$$

**TABLE 2.** VM types provided by GoGrid and Amazon EC2.

| VM Type | CPU | Memory | Prices (GoGrid) | Prices (EC2 us-east) | Prices (EC2 us-west) | Prices (EC2 eu-east) |
|---|---|---|---|---|---|---|
| t2.micro | 1 | 1 | 0.02 | 0.013 | 0.017 | 0.014 |
| t2.small | 1 | 2 | 0.03 | 0.026 | 0.034 | 0.028 |
| t2.medium | 2 | 4 | 0.06 | 0.052 | 0.068 | 0.056 |
| m3.medium | 1 | 3.75 | 0.09 | 0.070 | 0.077 | 0.077 |
| m3.large | 2 | 7.5 | 0.17 | 0.140 | 0.154 | 0.154 |
| m3.xlarge | 4 | 15 | 0.34 | 0.280 | 0.308 | 0.308 |
| m3.2xlarge | 8 | 30 | 0.68 | 0.560 | 0.616 | 0.616 |

where $\rho$ is another tuning parameter referred to as *capacity factor*. For testing instances of larger sizes, they would have more resources available on the private cloud.

### B. PARAMETER DETERMINATION

In the proposed NFA, there are two parameters $\beta_0$ and $\gamma$, both of which are used to update fireflies' positions in Equation (14). As indicated in Section IV-D, $\beta_0 \in [0, 1]$ in general. Meanwhile, $\beta_0 = 1$ indicates that the brightest firefly uses its large light intensity to attract other fireflies to move toward it. As NFA regards fireflies that are closer to the brightest one may be mapped to better solutions in higher possibilities, the brightest firefly is required to try its best to attract other fireflies to move toward it. Accordingly, we set $\beta_0 = 1$. Also, $\gamma$ cannot be set as a large value, since a greater $\gamma$ value denotes lower attractiveness. Due to $\gamma \in [0.01, 10]$ in general, we select 10 candidate values in the first half of this value domain, i.e., $\gamma \in \{0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5\}$.

Considering the use of randomness in the scheduling algorithms, we use the well-known multi-factor analysis of variance (ANOVA) method to evaluate the scheduling performance in a statistical sense. For this reason, we use different values of the two tuning factors, $\lambda \in \{1, 5, 10\}$ and $\rho \in \{0.05, 0.1, 0.15\}$ to create nine different parameter configurations. NFA is evaluated on all possible configurations. The population size in NFA (i.e., the number of fireflies) is set as $|\Omega| = 30$. For each firefly, its position value in each dimension is randomly selected within the interval $[-2, 2]$. We terminate the NFA procedure if the number of iterations reaches 1000. For each testing instance, the scheduling algorithm is performed with multiple rounds. We record the makespan corresponding to the scheduling solution in each round, and use relative error (RE) to assess the scheduling quality, which is defined as

$$RE = \left( \sum_{r=1}^{R} \frac{c_r - c_{LB}}{c_{LB}} \right) \Big/ R \times 100\%, \tag{18}$$

where $R$ is the number of rounds that the scheduling algorithm is repeated, $c_r$ denotes the obtained makespan in the $r$-th round, and $c_{LB}$ is the lower-bound limit for makespan, which can be determined by assuming that the private cloud has infinite resources and there is a unlimited budget for public cloud usage. Referring to Equation (18), a smaller RE indicates a higher scheduling quality as the makespan is closer to the lower-bound limit. We then use ANOVA method to analyze the statistical significance of the observed average REs. Figure 3 presents the mean REs and 95% confidence least-significant difference (LSD) intervals with different $\gamma$ values ranging from 0.5 to 5. We can observe that the REs of NFA with $\gamma = 1$, 1.5, 2, 2.5, 3 are similar, and are lower than those of NFA with other $\gamma$ values. We set $\gamma = 2$ since this value leads to the lowest RE.

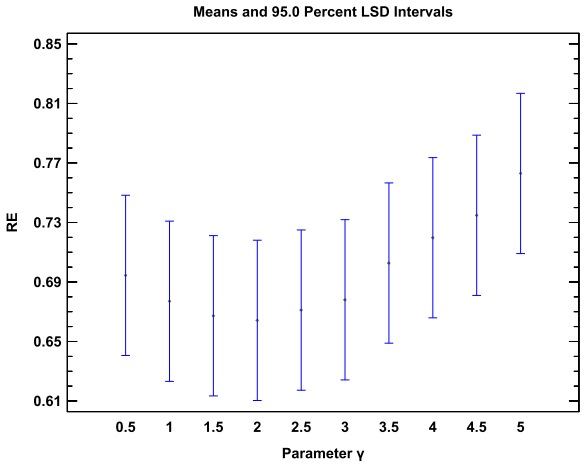


**FIGURE 3.** Mean REs and LSD Intervals for different $\gamma$ values.

### C. ALGORITHM EVALUATION

For comparison purposes, we use FILS [5], which is the existing best algorithm for solving the considered problem, as the baseline in performance evaluation. The standard firefly algorithm (SFA) [7] is also taken into account. However, SFA was originally introduced to solve continuous optimization problems. In order to make SFA capable of solving the considered problem in this paper, ROV is employed by SFA to map fireflies to solutions. Same as NFA, SFA uses FTA to evaluate the obtained solutions. All the parameters of SFA and FILS are set as the same in [5], [7], respectively. For fair comparison, the termination criteria of all the three compared algorithms are set as 1000 generations. Same as those in Section V-B, the two factors are set as $\lambda \in \{1, 5, 10\}$ and $\rho \in \{0.05, 0.1, 0.15\}$, respectively. All three scheduling algorithms used in comparative experiments are repeated

**Algorithm 3** Novel Firefly Algorithm (NFA)

**Input**: A task set $\Psi$ including all applications' tasks;
**Output**: The objective of the best solution $f(\pi^*)$;
Set the values of *MaxGeneration*, $\alpha$, $\beta_0$ and $\gamma$, respectively;
Initialize the population of fireflies $\Omega = \{X_1, X_2, \cdots, X_{|\Omega|}\}$, randomly;
Define the light intensity $I_i$ of the firefly $i$ as $1/f(\pi_i)$;
Employ the composite heuristic to generate the initial best solution $\pi^*$;
Select a random firefly from the population as the brightest one and assume its index as $b$;
**for** $i = 1$ *to* $|\Omega|$ **do**
    Calculate $r_i^*$;
    Generate a solution $\pi_i$ corresponding to the firefly $i$ by the proposed mapping operator with $r_i^*$ and $\pi^*$;
    Calculate the objective $f(\pi_i)$ of the obtained solution by performing FTA on $\pi_i$;
    **if** $f(\pi_i) < f(\pi^*)$ **then**
        $b \leftarrow i$;
        $\pi^* \leftarrow \pi_i$;
        $f(\pi^*) \leftarrow f(\pi_i)$;
**while** $t <$ *MaxGeneration* **do**
    **for** $i = 1$ *to* $|\Omega|$ **do**
        **for** $j = 1$ *to* $|\Omega|$ **do**
            **if** $I_j > I_i$ **then**
                Calculate distance $r_{ij}$ by Equation (13);
                Move the firefly $i$ toward the firefly $j$ by Equation (14);
            Calculate $r_i^*$;
            Generate a solution $\pi_i$ corresponding to the firefly $i$ by the proposed mapping operator with $r_i^*$ and $\pi^*$;
            Calculate the objective $f(\pi_i)$ of the obtained solution by performing FTA on $\pi_i$;
            Update $I_i$;
            **if** $f(\pi_i) < f(\pi^*)$ **then**
                $b \leftarrow i$;
                $\pi^* \leftarrow \pi_i$;
                $f(\pi^*) \leftarrow f(\pi_i)$;
**return** $f(\pi^*)$;

with 5 rounds (i.e. $R$=5). Figure 4 illustrates the comparison results of mean REs and LSD intervals at a 95% confidence level by different algorithms.

We can observe from Figure 4 that, the mean REs of FILS, NFA and SFA are 0.96, 0.66 and 1.03, respectively. We conclude that NFA outperforms both FILS and SFA. The reasons for NFA's good effectiveness are explained as follows. First, NFA employs an effective composite heuristic to initialize a high-quality best solution. Moreover, NFA uses the distance-based mapping operator to map fireflies onto scheduling

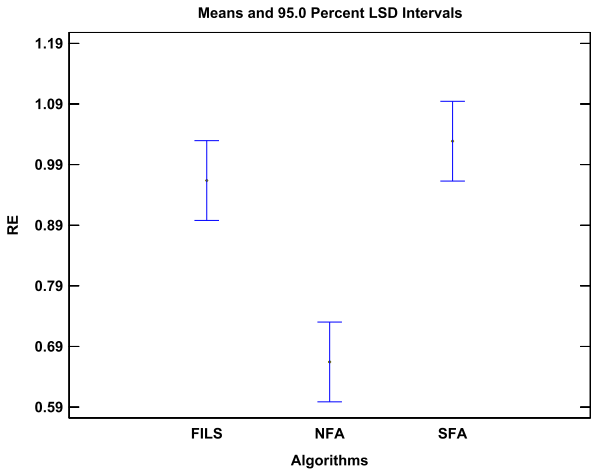


**FIGURE 4.** Comparison of mean REs and LSD Intervals among different algorithms.

solutions. The good genes hidden inside the current best solution are inherited to the resultant solutions. Finally, NFA utilizes the proposed movement scheme, in which fireflies can explore a wide range while they are distant from the brightest one, whereas fireflies can search in a small neighborhood around the brightest one for the purpose of maintaining a high probability in the mapping operator to inherit most good genes in the current best solution to generated solutions when they are nearby the brightest one. All these conclusions will be verified in Section V-E.

In order to evaluate the efficiency, we use normalized efficiency (NE) to denote scheduling algorithm's computational efficiency, which is defined as the ratio of its computation time to the computation time of a baseline algorithm. The lower NE value is, the higher computational efficiency the scheduling algorithm achieves. In this set of experiments, FILS is used as the baseline algorithm. All scheduling algorithms are performed with $R$ rounds, and computation times are averaged to calculate NE. Figure 5 provides the mean NEs achieved by different algorithms for evaluation. The NE values by FILS, NFA and SFA are 1, 0.32, and 0.69, respectively, indicating that NFA achieves the highest efficiency. More importantly, NFA is more efficient than SFA. The reason is that the complexity of the proposed distance-based mapping operator employed by NFA is $O(T)$, lower than that of ROV used by SFA $O(T \cdot \log T)$.

### D. IMPACTS OF KEY FACTORS

We perform another set of experiments to analyze the impacts of several key factors, including the budget factor $\lambda$, capacity factor $\rho$, and the number of tasks $T$, upon the scheduling results. As in previous experiments, we observe the mean REs and LSD intervals by different algorithm when the factors increases their values. We can make several observations from the evaluation results presented in Figures 6-8. As the $\lambda$ value increases, the mean REs by scheduling algorithms decrease with a higher budget limit. On the other hand, with

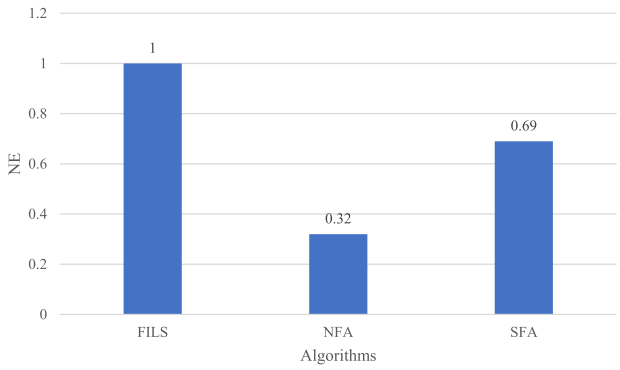


**FIGURE 5.** Mean NEs achieved by different algorithms.

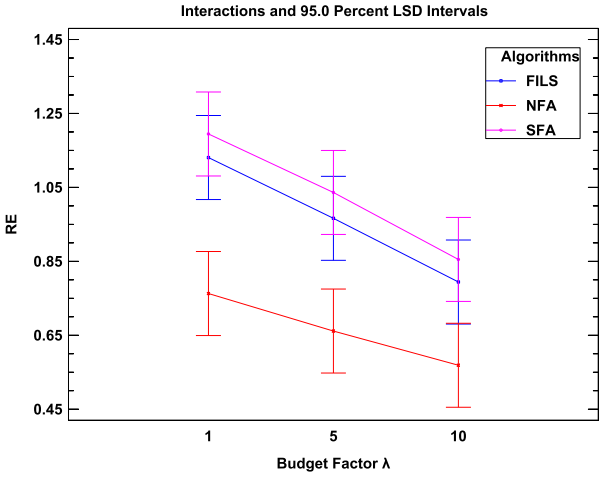


**FIGURE 6.** Mean REs and LSD intervals w.r.t $\lambda$ value.

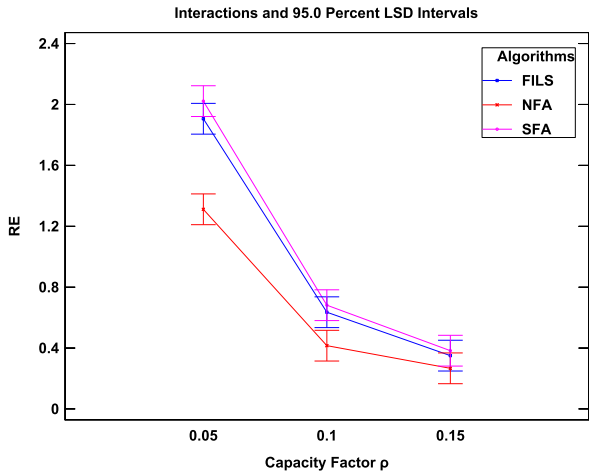


**FIGURE 7.** Mean REs and LSD intervals w.r.t $\rho$ value.

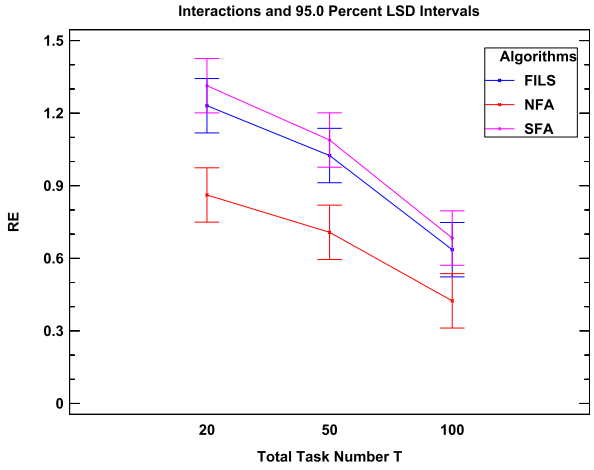


**FIGURE 8.** Mean REs and LSD intervals w.r.t $T$ value.

a larger $\rho$ value, we can also observe a decrease in the mean REs, since the parallelism of task execution has been enhanced with more resources provided by the private cloud. On the side of the total task number $T$, Figure 8 shows that the proposed NFA achieves the lowest REs with regardless of the value of $T$. In other words, NFA maintains stably good effectiveness for all scales of problems.

### E. IMPACTS OF KEY OPERATORS

We now investigate the impacts of several main procedures of NFA, including the mapping operator, composite heuristic and movement scheme upon scheduling results. The composite heuristic consisting of LTF, IRM, and SSM. The effectiveness of LTF has been verified in our previous work [5]. We now evaluate the effectiveness of IRM, SSM and the overall composite heuristic. For this reason, we construct three new NFAs, NFA_RAN, NFA_H1 and NFA_H2 for evaluation. RAN uses a random method to generate the initial best solution. H1 is a heuristic that uses LTF to generate the initial best solution and utilizes IRM to improve the obtained solution. H2 also employs LTF to produce the initial best solution, but uses SSM to improve it. Obviously, the effectiveness of IRM, SSM, and the overall composite heuristic can be shown by comparing NFA with NFA_H2, NFA_H1 and NFA_RAN, respectively. In order to evaluate the distance-based mapping operator, we design NFA_ROV by replacing the mapping operator with the ROV rlue in NFA. To verify the proposed movement scheme, we establish NFA_SM, in which the standard movement scheme is employed instead of our proposed movement scheme. These algorithms are compared with NFA. Parameters of those new constructed algorithms are set the same as those of NFA. Note that NFA_SM has another parameter $\alpha$, whose value is set as 0.8 by an experiment in which $\alpha \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$, and NFA_SM with $\alpha = 0.8$ achieves the best effectiveness. All the compared algorithms are run with the maximal number of generations 1000 and $R = 5$. The results of mean REs and LSD intervals (at a 95% confidence level) by different algorithms are given in Figure 9.

We can observe from Figure 9 that NFA is more effective than NFA_ROV in terms of scheduling quality. The results show that the proposed mapping operator is superior to the ROV rule. The reason is that in the proposed

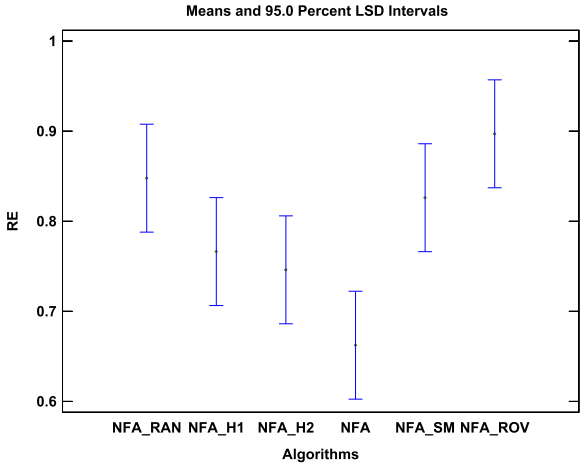


**FIGURE 9.** Plot of mean REs and LSD intervals for the compared algorithms.

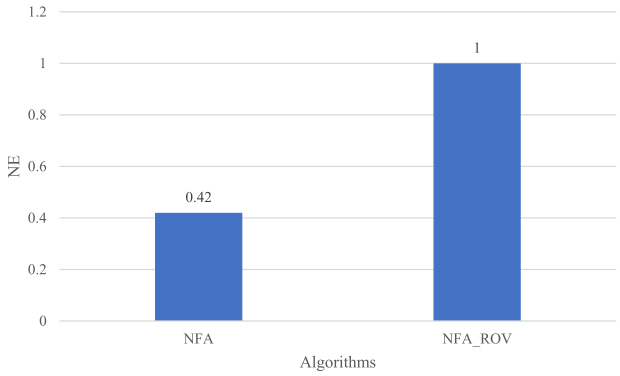


**FIGURE 10.** Plot of mean NEs for NFA and NFA_ROV.

distance-based mapping operator, good genes hidden inside of the current best solution can be inherited to those generated solutions probabilistically, which is beneficial for the high quality of the obtained solutions. In addition, as aforementioned, the complexity of the mapping operator is lower than that of ROV. This conclusion can be verified by the resutls in Figure 10. The mean NEs of NFA and NFA_ROV are 0.42 and 1.00, respectively.

Figure 9 also shows that NFA significantly outperforms NFA_RAN, indicating that composite heuristic has good effectiveness and makes a substantial contribution to the high performance of NFA. Although NFA leads to a lower RE than NFA_H1 and NFA_H2 do, we notice that NFA does not outperform NFA_H1 and NFA_H2 significantly. Thus, we draw the conclusion that both IRM and SSM are effective but do not make significant contributions. Nevertheless, IRM and SSM both play important roles, which can be verified by the fact that NFA_H1 and NFA_H2 are better than NFA_RAN in an insignificant way but NFA outperforms NFA_RAN significantly. The only difference between NFA and NFA_H1/NFA_H2 is that, NFA employs both IRM and SSM but NFA_H1/NFA_H2 solely uses IRM/SSM. Judging by this point, the good effectiveness of NFA depends on both IRM and SSM.

One more observation can be made from Figure 9 that NFA significantly outperforms NFA_SM, denoting that the proposed movement scheme is advantageous over the standard movement scheme. The reason is that we allow fireflies to move randomly in a wide/narrow range when they have long/short distances to the brightest firefly. As a result, when fireflies are distant from the brightest one, a wide range in the search space can be explored. When fireflies are close to the brightest one, they search in a small neighborhood around of the brightest firefly. In other words, these fireflies can maintain a short distance to the brightest one, leading to a high probability $p$ of inheriting most good genes from the current best solution to generated solutions. Therefore, high-quality solutions can be obtained by the mapping operator, and the effectiveness of NFA can be improved.

## VI. CONCLUSION

This paper addressed the problem of scheduling bag-of-tasks (BoT) applications with budget constraints on hybrid clouds for the purpose of minimizing the makespan. We proposed a novel firefly algorithm (NFA) to solve it. In NFA, fireflies are evaluated by two steps: (1) use a distance-based mapping operator for mapping fireflies to solutions; (2) employ an existing heuristic FTA to calculate solutions' objectives. We also proposed a composite heuristic to generate the initial best solution and a new movement scheme to update fireflies' positions. Experimental results showed that NFA outperforms the standard firefly algorithm and the existing best algorithm for the considered problem FILS, in terms of scheduling effectiveness and computational efficiency.

## REFERENCES

[1] M. Hu and B. Veeravalli, "Requirement-aware scheduling of bag-of-tasks applications on grids with dynamic resilience," *IEEE Trans. Comput.*, vol. 62, no. 10, pp. 2108–2114, Oct. 2013.

[2] G. Terzopoulos and H. D. Karatza, "Bag-of-task scheduling on power-aware clusters using a DVFS-based mechanism," in *Proc. IEEE Int. Symp. Workshops Parallel Distrib. Process.*, May 2014, pp. 833–840.

[3] L. Thai, B. Varghese, and A. Barker, "A survey and taxonomy of resource optimisation for executing bag-of-task applications on public clouds," *Future Gener. Comput. Syst.*, vol. 82, pp. 1–11, May 2018.

[4] P. Varshney and Y. Simmhan, "AutoBoT: Resilient and cost-effective scheduling of a bag of tasks on spot VMs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 7, pp. 1512–1527, Jul. 2019. doi: 10.1109/TPDS.2018.2889851.

[5] Y. Zhang, J. Sun, Z. Wu, S. Xie, and R. Xu, "Scheduling parallel intrusion detecting applications on hybrid clouds," *Secur. Commun. Netw.*, vol. 2018, Oct. 2018, Art. no. 2863793. doi: 10.1155/2018/2863793.

[6] P. Brucker, *Scheduling Algorithms*. New York, NY, USA: Springer-Verlag, 2004.

[7] X. S. Yang, *Nature-Inspired Metaheuristic Algorithms*. Beckington, U.K.: Luniver Press, 2008.

[8] M. K. Marichelvam, T. Prabaharan, and X. S. Yang, "A discrete firefly algorithm for the multi-objective hybrid flowshop scheduling problems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 2, pp. 301–305, Apr. 2014.

[9] K. H. Kim, R. Buyya, and J. Kim, "Power aware scheduling of bag-of-tasks applications with deadline constraints on DVS-enabled clusters," in *Proc. 7th IEEE Int. Symp. Cluster Comput. Grid*, May 2007, pp. 541–548.

[10] R. N. Calheiros and R. Buyya, "Energy-efficient scheduling of urgent bag-of-tasks applications in clouds through DVFS," in *Proc. IEEE 6th Int. Conf. Cloud Comput. Technol. Sci.*, Dec. 2014, pp. 342–349.

[11] Y. Zhang, Y. Wang, and C. Hu, "Cloudfreq: Elastic energy-efficient bag-of-tasks scheduling in DVFS-enabled clouds," in *Proc. IEEE Int. Conf. Parallel Distrib. Syst.*, Dec. 2015, pp. 585–592.
[12] A.-M. Oprescu and T. Kielmann, "Bag-of-tasks scheduling under budget constraints," in *Proc. 2nd IEEE Int. Conf. Cloud Comput. Technol. Sci.*, Nov./Dec. 2010, pp. 351–359.
[13] A.-M. Oprescu, T. Kielmann, and H. Leahu, "Stochastic tail-phase optimization for bag-of-tasks execution in clouds," in *Proc. IEEE Int. Conf. Utility Cloud Comput.*, Nov. 2012, pp. 204–208.
[14] M.-A. Vasile, F. Pop, R.-I. Tutueanu, and V. Cristea, "HySARC$^2$: Hybrid scheduling algorithm based on resource clustering in cloud environments," in *Proc. 13th Int. Conf. Algorithms Archit. Parallel Process.* Springer, Dec. 2013, pp. 416–425.
[15] J. O. Gutierrez-Garcia and K. M. Sim, "A family of heuristics for agent-based elastic Cloud bag-of-tasks concurrent scheduling," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1682–1699, Sep. 2013.
[16] L. Thai, B. Varghese, and A. Barker, "Executing bag of distributed tasks on the cloud: Investigating the trade-offs between performance and cost," in *Proc. IEEE 6th Int. Conf. Cloud Comput. Technol. Sci.*, Dec. 2014, pp. 400–407.
[17] L. Thai, B. Varghese, and A. Barker, "Budget constrained execution of multiple bag-of-tasks applications on the cloud," in *Proc. IEEE Int. Conf. Cloud Comput.*, Jun./Jul. 2015, pp. 975–980.
[18] M. Mao, J. Li, and M. Humphrey, "Cloud auto-scaling with deadline and budget constraints," in *Proc. IEEE/ACM Int. Conf. Grid Comput.*, Oct. 2010, pp. 41–48.
[19] M. H. Farahabady, Y. C. Lee, and A. Y. Zomaya, "Non-clairvoyant assignment of bag-of-tasks applications across multiple clouds," in *Proc. Int. Conf. Parallel Distrib. Comput., Appl. Technol.*, Dec. 2012, pp. 423–428.
[20] I. A. Moschakis and H. D. Karatza, "Multi-criteria scheduling of bag-of-tasks applications on heterogeneous interlinked clouds with simulated annealing," *J. Syst. Softw.*, vol. 101, pp. 1–14, Mar. 2015.
[21] I. A. Moschakis and H. D. Karatza, "A meta-heuristic optimization approach to the scheduling of bag-of-tasks applications on heterogeneous clouds with multi-level arrivals and critical jobs," *Simul. Model. Pract. Theory*, vol. 57, pp. 1–25, Sep. 2015.
[22] R. Van den Bossche, K. Vanmechelen, and J. Broeckhove, "Cost-optimal scheduling in hybrid IaaS clouds for deadline constrained workloads," in *Proc. IEEE Int. Conf. Cloud Comput.*, Jul. 2010, pp. 228–235.
[23] R. Van den Bossche, K. Vanmechelen, and J. Broeckhove, "Cost-efficient scheduling heuristics for deadline constrained workloads on hybrid clouds," in *Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci.*, Nov./Dec. 2011, pp. 320–327.
[24] R. Van den Bossche, K. Vanmechelen, and J. Broeckhove, "Online cost-efficient scheduling of deadline-constrained workloads on hybrid clouds," *Future Gener. Comput. Syst.*, vol. 29, no. 4, pp. 973–985, 2013.
[25] M. Malawski, K. Figiela, and J. Nabrzyski, "Cost minimization for computational applications on hybrid cloud infrastructures," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1786–1794, 2013.
[26] B. Wang, Y. Song, Y. Sun, and J. Liu, "Managing deadline-constrained bag-of-tasks jobs on hybrid clouds," in *Proc. 24th High Perform. Comput. Symp.*, Apr. 2016, Art. no. 22.
[27] V. Pelaez, A. Campos, D. F. Garcia, and J. Entrialgo, "Autonomic scheduling of deadline-constrained bag of tasks in hybrid clouds," in *Proc. Int. Symp. Perform. Eval. Comput. Telecommun. Syst.*, Jul. 2016, pp. 1–8.
[28] S. Abdi, L. Pourkarimi, M. Ahmadi, and F. Zargari, "Cost minimization for deadline-constrained bag-of-tasks applications in federated hybrid clouds," *Future Gener. Comput. Syst.*, vol. 71, pp. 113–128, Jun. 2017.
[29] X. Zuo, G. Zhang, and W. Tan, "Self-adaptive learning PSO-based deadline constrained task scheduling for hybrid IaaS cloud," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 2, pp. 564–573, Apr. 2014.
[30] Y. Zhang and J. Sun, "Novel efficient particle swarm optimization algorithms for solving QoS-demanded bag-of-tasks scheduling problems with profit maximization on hybrid clouds," *Concurrency Comput., Pract. Exper.*, vol. 29, no. 21, 2017, Art. no. e4249. doi: 10.1002/cpe.4249.
[31] Y. Zhang, J. Sun, and Z. Wu, "An heuristic for bag-of-tasks scheduling problems with resource demands and budget constraints to minimize makespan on hybrid clouds," in *Proc. 5th Int. Conf. Adv. Cloud Big Data*, Aug. 2017, pp. 39–44.
[32] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *Proc. Int. Symp. Stochastic Algorithms*, 2009, pp. 169–178.
[33] X.-S. Yang, "Multiobjective firefly algorithm for continuous optimization," *Eng. Comput.*, vol. 29, no. 2, pp. 175–184, 2013.
[34] K. Chandrasekaran and S. P. Simon, "Optimal deviation based firefly algorithm tuned fuzzy design for multi-objective UCP," *IEEE Trans. Power Syst.*, vol. 28, no. 1, pp. 460–471, Feb. 2013.
[35] G. K. Jati and Suyanto, "Evolutionary discrete firefly algorithm for travelling salesman problem," in *Proc. Int. Conf. Adapt. Intell. Syst.*, 2011, pp. 393–403.
[36] G. K. Jati, R. Manurung, and Suyanto, "Discrete firefly algorithm for traveling salesman problem: A new movement scheme," in *Swarm Intelligence and Bio-Inspired Computation*. Amsterdam, The Netherlands: Elsevier, 2013, pp. 295–312.
[37] J. Li, T. Li, Y. Yu, Z. Zhang, P. M. Pardalos, Y. Zhang, and Y. Ma, "Discrete firefly algorithm with compound neighborhoods for asymmetric multi-depot vehicle routing problem in the maintenance of farm machinery," *Appl. Soft Comput.*, vol. 81, Aug. 2019, Art. no. 105460. doi: 10.1016/j.asoc.2019.04.030.
[38] A. E. Ezugwu and F. Akutsah, "An improved firefly algorithm for the unrelated parallel machines scheduling problem with sequence-dependent setup times," *IEEE Access*, vol. 6, pp. 54459–54478, 2018. doi: 10.1109/ACCESS.2018.2872110.
[39] M. Farhoodnea, A. Mohamed, H. Shareef, and H. Zayandehroodi, "Optimum placement of active power conditioner in distribution systems using improved discrete firefly algorithm for power quality enhancement," *Appl. Soft Comput.*, vol. 23, pp. 249–258, Oct. 2014.
[40] M. Farhoodnea, A. Mohamed, H. Shareef, and H. Zayandehroodi, "Optimum placement of active power conditioners by a dynamic discrete firefly algorithm to mitigate the negative power quality effects of renewable energy-based generators," *Int. J. Elect. Power Energy Syst.*, vol. 61, pp. 305–317, Oct. 2014.
[41] C. Liu, K. Li, and K. Li, "Minimal cost server configuration for meeting time-varying resource demands in cloud centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 11, pp. 2503–2513, Nov. 2018.
[42] C. Liu, K. Li, J. Liang, and K. Li, "COOPER-MATCH: Job offloading with a cooperative game for guaranteeing strict deadlines in MEC," *IEEE Trans. Mobile Comput.*, to be published. doi: 10.1109/TMC.2019.2921713.

**YI ZHANG** received the B.S. and Ph.D. degrees from the School of Computer Science and Engineering, Southeast University, Nanjing, China, in 2005 and 2011, respectively.

From July 2009 to December 2009, he was an Intern with the IBM China Research Laboratory. He has received the Ph.D. Fellowship from IBM. In 2011, he joined Huawei Tech., Comapny as a member of Technical Research Staff. He is currently an Assistant Professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing. His research interests include project scheduling, workflow optimization, resource management, and allocation in cloud computing and mobile computing.

**JUNLONG ZHOU** (S'15–M'17) received the Ph.D. degree in computer science from East China Normal University, Shanghai, China, in 2017.

He has published 50 refereed articles in his research areas, most of which are published in premium conferences and journals including the ACM/IEEE DATE, the IEEE TC, the IEEE TPDS, the IEEE TCAD, the IEEE TCAS, the IEEE TR, and the IEEE TAES. He was a Visiting Scholar with the University of Notre Dame, Notre Dame, IN, USA, from 2014 to 2015. He is currently an Assistant Professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. His research interests include real-time embedded systems, cloud computing, and cyber physical systems.

Dr. Zhou received the Reviewer Award of the *Journal of Circuits, Systems, and Computers*, in 2016. He has been an Associate Editor of the *Journal of Circuits, Systems, and Computers*. He serves as a Guest Editor for several special issues of the *ACM Transactions on Cyber-Physical Systems*, *IET Cyber-Physical Systems: Theory & Applications*, and the *Journal of Systems Architecture* (Elsevier).

**LULU SUN** received the B.S. degree in software engineering from the Nanjing University of Science and Technology, Nanjing, China, in 2017, where she is currently pursuing the M.S. degree with the School of Computer Science and Engineering. Her research interests include cloud computing and task scheduling.

**JINGJING MAO** received the B.S. degree in software engineering from the Nanjing University of Science and Technology, Nanjing, China, in 2016, where she is currently pursuing the M.S. degree with the School of Computer Science and Engineering. Her research interests include cloud computing and task scheduling.

**JIN SUN** (M'17) received the B.S. and M.S. degrees in computer science from the Nanjing University of Science and Technology, Nanjing, China, in 2004 and 2006, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Arizona, in 2011.

From 2012 to 2014, he was a Technical Staff Member of Orora Design Technologies, Inc., Redmond, WA, USA. He is currently an Associate Professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. His research interests include high-performance computing, integrated circuit modeling and analysis, and computer-aided design.