ECFA: An Efficient Convergent Firefly Algorithm for Solving Task Scheduling Problems in Cloud-Edge Computing

Lu Yin, *Student Member, IEEE*, Jin Sun, *Member, IEEE*, Junlong Zhou, *Member, IEEE*, Zonghua Gu, *Senior Member, IEEE*, and Keqin Li, *Fellow, IEEE*

***Abstract*—In cloud-edge computing paradigms, the integration of edge servers and task offloading mechanisms has posed new challenges to developing task scheduling strategies. This paper proposes an efficient convergent firefly algorithm (ECFA) for scheduling security-critical tasks onto edge servers and the cloud datacenter. The proposed ECFA uses a probability-based mapping operator to convert an individual firefly into a scheduling solution, in order to associate the firefly space with the solution space. Distinct from the standard FA, ECFA employs a low-complexity position update strategy to enhance computational efficiency in solution exploration. In addition, we provide a rigorous theoretical analysis to justify that ECFA owns the capability of converging to the global best individual in the firefly space. Furthermore, we introduce the concept of boundary traps for analyzing firefly movement trajectories, and investigate whether ECFA would fall into boundary traps during the evolutionary procedure under different parameter settings. We create various testing instances to evaluate the performance of ECFA in solving the cloud-edge scheduling problem, demonstrating its superiority over FA-based and other competing metaheuristics. Evaluation results also validate that the parameter range derived from the theoretical analysis can prevent our algorithm from falling into boundary traps.**

***Index Terms*—Cloud-edge computing, task scheduling, firefly algorithm, convergence proof, trajectory analysis.**

I. Introduction

With the great advance in 5th Generation (5G) communications and Internet of Things (IoT), conventional mobile cloud computing (MCC) is inadequate to satisfy the requirements of high-bandwidth and low-latency due to the centralized mechanism and the increasing number of mobile devices (MDs) [1], [2], [3]. To address this limitation, mobile edge computing (MEC) has become a promising computing paradigm that deploys computing resources and services on network edges to reduce the end-to-end delay [4], [5], [6]. By offloading computation tasks to edge servers instead of the cloud datacenter, MEC can deliver low-latency services and address the energy limitation of MDs. However, edge servers are generally resource-limited and may be incapable of processing computation-intensive applications under tight deadline constraints [7]. Therefore, cloud-edge computing becomes an ideal collaborative platform that takes advantage of both MCC's large-scale computing resources and MEC's low-latency computing services [8]. In case of insufficient computing resources, the edge server can further submit the computation tasks to the resource-rich datacenter to avoid deadline violation. In this context, it would be challenging to schedule deadline-constrained computation tasks considering the collaboration between cloud datacenter and edge servers.

Task scheduling problems are generally modeled as combinatorial optimization problems that seek for the optimal assignment of tasks onto computing resources. Metaheuristic algorithms, especially swarm intelligence algorithms, are popularly used to develop scheduling strategies. In this work, we propose an efficient metaheuristic based upon the solution exploration framework of the firefly algorithm (FA) to solve the aforementioned cloud-edge scheduling problem. FA is essentially a swarm intelligence algorithm inspired by the social behavior of fireflies [9]. Due to its advantages such as few parameters and easy implementation, FA has been used in various fields to solve real-world optimization problems. However, there still exist challenging issues when applying FA to solve the cloud-edge scheduling problem. On the one hand, the heavy computation burden induced by the frequent calculation of distances among fireflies degrades FA's computational efficiency. On the other hand, inappropriate selection of parameter values may cause FA not to converge to the global best solution. With these two challenges in mind, this work attempts to enhance FA's performance in solving scheduling problems by reducing its computational complexity as well as improving the quality of scheduling solutions.

This paper proposes an efficient convergent firefly algorithm (ECFA) for solving the task scheduling problem in a cloud-edge

Manuscript received 14 February 2023; revised 3 June 2023; accepted 22 June 2023. Date of publication 10 July 2023; date of current version 8 October 2023. This work was supported in part by Jiangsu Provincial Key Research and Development Program under Grant BE2022065-2, in part by the National Natural Science Foundation of China under Grant 62172224, in part by the Natural Science Foundation of Jiangsu Province under Grant BK20220138, in part by Kempe Foundation, Sweden, and in part by the Industry-University-Research Innovation Funds for Chinese Universities under Grant 2020ITA03002. Recommended for acceptance by H. Karatza. *(Corresponding author: Jin Sun.)*

Lu Yin, Jin Sun, and Junlong Zhou are with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, Jiangsu 210094, China (e-mail: ylu@njust.edu.cn; sunj@njust.edu.cn; jlzhou@njust.edu.cn).

Zonghua Gu is with the Department of Applied Physics and Electronics, Umeå University, 90187 Umeå, Sweden (e-mail: zonghua.gu@umu.se).

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/TSC.2023.3293048

system. The proposed ECFA uses a probability-based mapping operator to convert an individual firefly into a high-quality scheduling solution, represented by a task sequence, to associate the firefly space with the solution space. Distinct from the standard FA, our proposed ECFA employs a low-complexity position update strategy to significantly reduce the computational burden when updating firefly positions. Based on rigorous theoretical analysis, we justify that ECFA owns the capability of converging to the global best individual in the firefly space. Thus, ECFA is significantly advantageous over the standard FA due to its efficient and convergent characteristics. Furthermore, to fully investigate the movement mechanism for firefly position change, we introduce a new concept of boundary trap for analyzing firefly movement trajectories during the evolutionary procedure, in order to predict whether ECFA would fall into boundary traps and to avoid such occurrences by setting appropriate parameter values. Simulations were performed by comparing ECFA with three FA-based methods and four other metaheuristic-based methods to demonstrate the superiority of our proposed ECFA in terms of both effectiveness and efficiency. Our main contributions are highlighted as follows:

1) We propose an efficient convergent firefly algorithm, which improves the standard FA by designing a low-complexity position update strategy and a probability-based mapping operator.
2) We perform rigorous theoretical analyses on ECFA's convergence by determining an appropriate range of the critical parameter.
3) We introduce the concept of boundary traps for analyzing the fireflies' trajectories during the evolutionary procedure, and further investigate whether the algorithm would fall into boundary traps.
4) Experimental results demonstrate ECFA's superiority and also justify that the parameter range determined by the theoretical analysis can prevent the algorithm from falling into boundary traps.

We organize the remainder of this paper as follows. Section II provides a literature review. Section III introduces the problem model. Section IV provides the details about ECFA. Section V proves the ECFA's convergence. Section VI discusses the firefly's moving trajectory during ECFA's evolutionary procedure. Section VII presents evaluation results, followed by the concluding remarks in Section VIII.

II. Related Work

In the literature, there are a variety of studies dedicated to task scheduling in cloud-edge computing [10], [11], [12]. These studies generally formulate task scheduling problems as combinatorial optimization models or approximate them by convex optimization models, and develop heuristics or analytical methods to solve the optimization problems. Chen et al. [13] formulated a performance-constrained computation offloading problem for minimizing the execution time in cloud-edge computing and introduced a stochastic optimization algorithm. Ning et al. [14] presented a branch and bound algorithm and an iterative heuristic strategy, respectively, to solve task scheduling problems for cloud-edge systems with a single device and multiple devices. Du et al. [3] studied the scheduling problem of the execution delay and total energy consumption optimization in a cloud-edge system and developed a cost-effective sub-optimal algorithm to solve it. Dou et al. [15] studied the computation offloading problem in a hybrid multi-access edge-cloudlet platform and suggested an energy-efficient task scheduling algorithm for the joint optimization of computation offloading, data transmission, and bandwidth assignment. Though recognizing the significance of the above-mentioned studies, these existing methods either use analytical methods that are problem-dependent, or use heuristic methods that may suffer from local optima. Our proposed ECFA, on the one hand, can be easily adapted to solve task scheduling problems in different computing environments by redesigning the task assignment strategy for solution evaluation. On the other hand, the convergence feature of ECFA can prevent it from falling into local optima, leading to high-quality scheduling solutions.

Inspired by the social behaviors of fireflies, Yang et al. [16] proposed the standard FA, which is proven to outperform many other popular metaheuristic algorithms [17]. Since our algorithm is based on the search strategy of FA, in what follows, we investigate existing methods employing FAs to solve optimization problems. Ren et al. [18] proposed a variant firefly algorithm for solving constrained engineering design problems by applying self-adaptive strategies to balance its exploration and exploitation capacities. Altabeeb et al. [19] introduced a multi-population-based firefly algorithm CVRP-CHFA for solving a vehicle routing problem, in which populations communicate and cooperate to maintain population diversity. We further pay special attention to the FA-based methods oriented toward task scheduling problems. Li et al. [20] introduced a multi-objective discrete FA, which divides the continuous position into discretized regions and employs a local search strategy to accelerate the search process. Ren et al. [21] used a selective elimination strategy to improve FA's global search ability and a decision strategy to enhance FA's local search ability. Zhang et al. [22] introduced an improved FA involving a mapping operator to covert a firefly into a scheduling sequence and a new movement scheme to enhance the searching capability. Sekaran et al. [23] developed a dominant FA that uses an adjacency matrix to represent the dominant fireflies' behaviors. Although these FA-based metaheuristics can be used to solve scheduling problems, they may generally suffer from limitations in computational efficiency and/or solution exploration effectiveness. Specifically oriented toward a cloud-edge system, our proposed ECFA focuses on improving FA's searching efficiency as well as providing convergence proof and trajectory analysis for parameter determination. Table I summarizes the main characteristics of existing FA-based scheduling algorithms and highlights the distinguishing properties of ECFA.

III. Problem Description

This section formulates the system model for the cloud-edge scheduling problem with the objective of energy minimization under the deadline constraint. For ease of reference, all the

TABLE I
SUMMARY OF EXISTING STUDIES USING FAS TO SOLVE OPTIMIZATION PROBLEMS

Reference	Studied Problem	Contributions
Ren et al. [18]	constrained engineering design	Self-adaptive strategies.
Altabeeb et al. [19]	capacitated vehicle routing	2-h-opt-based mutation; A multi-population communication strategy.
Li et al. [20]	multi-objective resource provisioning	A discrete region strategy; A local search mechanism.
Ren et al. [21]	virtual machine scheduling	An elimination method; A decision domain strategy.
Zhang et al. [22]	bag-of-tasks application scheduling	A composite initial solution generating heuristic, A slow-movement update strategy.
Sekaran et al. [23]	load balancing in cloud datacenters	A dominant-based search method.
This work	task scheduling in cloud-edge systems	A low-complexity individual update strategy; A probability-based mapping operator; Firefly trajectory analysis; A convergence proof.

TABLE II
ALL NOTATIONS FOR THE CLOUD-EDGE SCHEDULING PROBLEM

Notation	Definition
N	total number of tasks
t_i	task with index i
d_i	data amount of t_i
w_i	computation workload of t_i
w_i^E	encryption computation workload of t_i
w_i^D	decryption computation workload of t_i
F_{device}	CPU operating frequency of the MD
F_{c}	CPU operating frequency of the edge server
F_{s}	CPU operating frequency of the cloud server
x_i	the binary decision variable for t_i
D_i^E	duration time for encrypting t_i's data
D_i^D	duration time for decrypting t_i's data
D_i^P	duration time for executing t_i
T_i^E	completion time for encrypting t_i's data
R	transmission rate of the MD
D_i^T	duration time for transmitting t_i's data
T_i^T	completion time for transmitting t_i's data
T_i^C	completion time for executing t_i
T	completion time for executing all tasks
P_{e}	operating power of the MD
P_{t}	transmission power of the MD
P_{b}	transmission power of the edge server
P_{s}	operating power of the edge server
P_{c}	operating power of the cloud server

notations used for problem formulation and their definitions are listed in Table II.

A. Task Model

With the increasing demands for data security and user privacy, we consider the tasks to be security-critical, requiring data encryption and decryption when making scheduling decisions [24], [25], [26]. To achieve data protection, users can choose different levels of security services implemented by different encryption algorithms to encrypt the tasks before offloading them [25]. Similarly, as in [27], [28], [29], we assume that the time and energy cost of sending back the computation results can be negligible.

There are N independent security-critical tasks on the MD, that need to be offloaded through radio access networks. Each task can be characterized by $t_i =< d_i, w_i > (i \in \{1, 2, \ldots, N\})$,

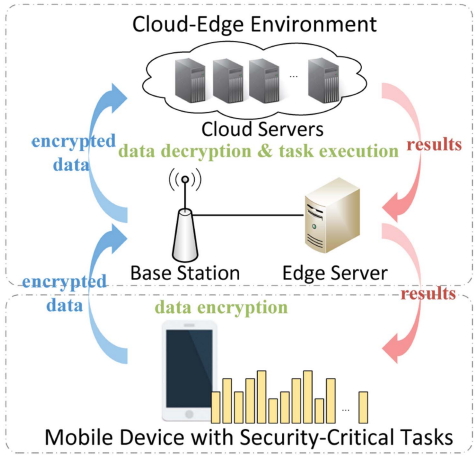


Fig. 1. Security-critical task scheduling in a cloud-edge system.

where d_i the data amount of t_i and w_i is the computation workload (in CPU cycles) for executing t_i. Mobile users can select specific security services implemented by different encryption algorithms to encrypt data before task offloading [25]. We use a set $\{sp_1, sp_2, \ldots, sp_M\}$ to represent several security policies that are available on edge and cloud. Each task is associated with a security policy that can satisfy the user's requirement on security level [5], [26], [30]. For example, the user chooses sp_v to protect t_i, and the computation workloads for encrypting and decrypting t_i can be determined by

$$w_i^E = d_i \times ecw_v, \tag{1}$$

$$w_i^D = d_i \times dcw_v, \tag{2}$$

where ecw_v and dcw_v represent the CPU cycles required for encrypting and decrypting a single data bit, respectively.

B. Cloud-Edge System Model

Fig. 1 illustrates an example of a cloud-edge system consisting of an MD, an edge server deployed along with a base station, and a cloud datacenter. Following [5], [26], we assume that the MD is equipped with one antenna, and thus only one task can be offloaded at a time. For any MD, the resource-limited edge server usually provides a dedicated virtual machine to execute the offloaded tasks in a first come first served (FCFS) manner [31]. Considering some tasks would not finish on time

under such circumstances, some tasks can be further outsourced to the resource-rich cloud datacenter. For a specific task t_i ($i \in \{1, 2, \ldots, N\}$), a binary variable x_i indicates the destination where t_i is scheduled. If t_i is outsourced to the cloud, we have $x_i = 1$, and $x_i = 0$ if t_i is offloaded to the edge server. For ease of understanding, we define two task subsequences $S = (s_1, s_2, \ldots, s_K)$ and $C = (c_1, c_2, \ldots, c_L)$ to represent the set of tasks offloaded to the edge server and outsourced to the cloud, respectively. Apparently, $N = K + L$, where K and L are the number of tasks in S and C, respectively.

C. Time and Energy Models

For any task t_i, its encryption duration, decryption duration, and processing duration can be calculated by $D_i^E = \frac{w_i^E}{F_{\text{device}}}$, $D_i^D = \frac{w_i^D}{F_x}$, and $D_i^P = \frac{w_i}{F_x}$, respectively. Depending on the scheduling decision x_i, we have $F_x = F_{\text{c}}$ if $x_i = 1$, and $F_x = F_{\text{s}}$ otherwise. On the MD, the encryption of t_i starts only after the encryption of its previous task is finished. So, the encryption completion time of t_i is

$$T_i^E = \begin{cases} 0, & i = 0 \\ T_{i-1}^E + D_i^E, & i = 1, 2, \ldots, N. \end{cases} \tag{3}$$

When data encryption of t_i is completed, the MD transmits it to the edge server. Following [32], the transmission rate between the MD and the edge server's base station is

$$R = b \log_2 \left(1 + \frac{g_0 (u_0/u)^\theta p}{b N_0} \right), \tag{4}$$

where definitions of relevant parameters can be referred to in [26]. Then, t_i's transmission duration can be calculated by

$$D_i^T = \begin{cases} \frac{d_i}{R}, & \text{if } x_i = 0 \\ \frac{d_i}{R} + \frac{d_i}{\tau R}, & \text{otherwise,} \end{cases} \tag{5}$$

where τ is a multiplicative factor denoting the ratio of the wired edge-to-cloud transmission rate to the wireless MD-to-edge transmission rate. The value of τ is set within the range [2, 5] by investigating related studies [33], [34], IEEE standards, and cloud providers' websites.

Since a task can only be transmitted after completing its own data encryption and the data transmission of the previous task is completed, the transmission completion time of t_i can be calculated by

$$T_i^T = \begin{cases} 0, & i = 0 \\ \max\{T_{i-1}^T, T_i^E\} + D_i^T, & i = 1, 2, \ldots, N. \end{cases} \tag{6}$$

For the task offloaded to the edge server, it cannot start the data decryption until both its own data transmission and the previous task's execution is completed. Assume that t_i is the k-th task in S, we have

$$T_k^P(S) = \begin{cases} 0, & k = 0 \\ \max\{T_{k-1}^P(S), T_i^T\} & \\ \quad + D_i^D + D_i^P, & k = 1, 2, \ldots, K. \end{cases} \tag{7}$$

On the other hand, the tasks scheduled to the cloud can be processed in parallel as long as their respective data transmissions are completed. Assuming that t_i is the l-th task in C, we have

$$T_l^P(C) = \begin{cases} 0, & l = 0 \\ T_i^T + D_i^D + D_i^P, & l = 1, 2, \ldots, L. \end{cases} \tag{8}$$

Then, t_i's execution completion time T_i^C is

$$T_i^C = \begin{cases} T_k^P(S), & x_i = 0 \\ T_l^P(C), & \text{otherwise.} \end{cases} \tag{9}$$

The completion time of all tasks can be determined by

$$T = \max\left\{T_i^C\right\}, i = 1, 2, \ldots, N. \tag{10}$$

The energy consumption of each task can be decomposed into three parts: E_{device}^i, E_{edge}^i, and E_{cloud}^i. Specifically, E_{device}^i consists of the energy consumption for data encryption and transmission, i.e.,

$$E_{\text{device}}^i = P_{\text{e}} \frac{w_i^E}{F_{\text{device}}} + P_{\text{t}} \frac{d_i}{R}, \tag{11}$$

where P_{e} and P_{t} denote MD's CPU operating power and transmission power, respectively. E_{edge}^i and E_{cloud}^i depends on the scheduling decision on t_i. If t_i is scheduled onto the edge server ($x_i = 0$), E_{edge}^i includes the energy consumption for data receiving, data decryption, and task execution, and $E_{\text{cloud}}^i = 0$. In the opposite scenario ($x_i = 1$), E_{edge}^i includes the energy consumption for transmitting data to the cloud, and E_{cloud}^i involves data receiving, data decryption, and task execution. Thus, we have

$$E_{\text{edge}}^i = P_{\text{b}} \frac{d_i}{R} + (1 - x_i) P_{\text{s}} \left(\frac{w_i^D}{F_{\text{s}}} + \frac{w_i}{F_{\text{s}}} \right) + x_i P_{\text{b}} \frac{d_i}{\tau R}, \tag{12}$$

$$E_{\text{cloud}}^i = x_i \left(P_{\text{c}} \frac{d_i}{R} + P_{\text{c}} \frac{w_i^D}{F_{\text{c}}} + P_{\text{c}} \frac{w_i}{F_{\text{c}}} \right), \tag{13}$$

where P_{b} is the transmission power of the edge server, P_{s} and P_{c} denote the CPU operating power of the edge server and cloud, respectively.

The optimization goal is the weighted sum of all three energy components, which are summed over all tasks, i.e.,

$$E_{\text{sum}} = \sum_{i=1}^{N} \left(\delta E_{\text{device}}^i + \varrho E_{\text{edge}}^i + \eta E_{\text{cloud}}^i \right), \tag{14}$$

where δ, ϱ, and η are weighting factors.

D. Scheduling Model

To summarize, we formulate the cloud-edge scheduling problem as the following constrained optimization problem:

$$\text{minimize} \quad E_{\text{sum}} \tag{15}$$

$$\text{subject to} \quad T \le DL \tag{16}$$

$$\text{variables} \quad \{x_1, x_2, \ldots, x_N\}, \tag{17}$$

where DL is a specified deadline value imposed on task completion time. The scheduling algorithm is to solve the formulated

problem by determining the most appropriate values of decision variables, such that the total energy is minimized while satisfying the deadline constraint.

IV. The Proposed ECFA

This section details the proposed ECFA for solving the task scheduling problem formulated in Section III-D. As the name of ECFA suggests, the main differences between ECFA and standard FA lay in ECFA's efficiency and convergence characteristics. On the one hand, the low-complexity position update strategy in ECFA avoids frequent calculations of distances among firefly individuals, which account for the major computation overhead in standard FA, and significantly improves the computational efficiency. On the other hand, as will be proved in the convergence analysis, ECFA can guarantee its convergence to the global best position in the firefly space. Benefited from this convergence feature, ECFA is advantageous over standard FA and other metaheuristics in escaping from local optima.

It is worth mentioning that, the cloud-edge scheduling problem studied in this work aims at compute-intensive tasks with long execution times, and falls within the scope of static scheduling in which task characteristics are known in advance. As will be revealed in experiments, ECFA can produce the scheduling solutions, for different problem scales, with negligible time overhead compared with the task execution durations.

A. Solution Representation

We define the firefly population as $\Omega = \{f_1, f_2, \ldots, f_{|\Omega|}\}$ where $|\Omega|$ denotes the population size. For each firefly ψ_i in Ω, its position is represented by a n-dimensional vector $x_i =< x_{i1}, x_{i2}, \ldots, x_{iN} >$, where N is the number of all tasks. We use a task sequence $\psi_i =< \varphi_i^1, \varphi_i^2, \ldots, \varphi_i^N >$, i.e., a permutation of all tasks to be scheduled, to represent a scheduling solution. In the FA framework, a firefly with lower brightness would be attracted by and move toward brighter fireflies. To enable ECFA's capability in solving the concerned problem, it is necessary to establish a concrete connection between the firefly position and the scheduling solution and link the brightness of the firefly to the objective value of the corresponding scheduling solution.

B. Mapping Operator

To map a firefly onto a scheduling solution, we propose a probability-based mapping operator to produce high-quality solutions. Different from the sorting-based ranked-order value (ROV) rule [9], [32] that sorts the firefly's position values to obtain a task sequence, our mapping scheme takes into account the position of the current best firefly such that the "good genes" hidden in the current best solution could be inherited.

The input to the mapping operator includes the current best solution ψ_{best} and the position X_i of the firefly to be mapped. The output is the scheduling solution ψ_i of the input firefly. We initialize ψ_i as an empty sequence. For an individual firefly f_i, its position value in each dimension is associated with a probability value

$$\rho_q = \frac{1}{1 + e^{-X_{iq}}}, q = 1, 2, \ldots, N, \tag{18}$$

which is within the range [0, 1]. We copy the q-th task in ψ_{best} to the corresponding position in ψ_i depending on the ρ_q value. The remaining tasks in ψ_{best} would be assigned to ψ_i with an equal probability. In this manner, we can obtain a valid and high-quality scheduling solution ψ_i.

Algorithm 1: Task Assignment Strategy.

Input: A task sequence $\psi_i =< t_i^1, t_i^2, \ldots, t_i^N >$ and the deadline value DL;
Output: The total energy consumption E_{sum};

1 Set $E_{\text{sum}} \leftarrow 0$;
2 **for** *each task* t_i^q **do** /* q = 1 to N */
3 Calculate E^q_{device} by Eq. (11);
4 Set $E_{\text{sum}} \leftarrow E_{\text{sum}} + E^q_{\text{device}}$;
5 Calculate T^q_{edge} and E^q_{edge} by Eqs. (7) and (12);
6 Calculate T^q_{cloud} and E^q_{cloud} by Eqs. (8) and (13);
7 **if** ($T^q_{edge} \le DL$ *and* $T^q_{cloud} \le DL$ *and* $E^q_{edge} \le E^q_{cloud}$) *or* ($T^q_{edge} \le DL$ *and* $T^q_{cloud} > DL$) **then**
8 Dispatch t_i^q to the edge server;
9 Set $E_{\text{sum}} \leftarrow E_{\text{sum}} + E^q_{\text{edge}}$;
10 **if** ($T^q_{edge} \le DL$ *and* $T^q_{cloud} \le DL$ *and* $E^q_{edge} > E^q_{cloud}$) *or* ($T^q_{edge} > DL$ *and* $T^q_{cloud} \le DL$) **then**
11 Dispatch t_i^q onto the cloud;
12 Set $E_{\text{sum}} \leftarrow E_{\text{sum}} + E^q_{\text{cloud}}$;
13 **if** $T^q_{edge} > DL$ *and* $T^q_{cloud} > DL$ **then**
14 Identify ψ_i as an infeasible solution;
15 Set $E_{\text{sum}} \leftarrow +\infty$;
16 **break**;
17 **return** E_{sum};

C. Task Assignment Strategy

To evaluate the fitness value of each firefly, we propose a task assignment strategy (TAS). The pseudocode in Algorithm 1 describes the processing flow of TAS. The input includes a firefly's task sequence ψ_i, which is obtained by the mapping operator as well as the deadline requirement DL. The optimization objective E_{sum} is initialized as 0. TAS traverses the task sequence ψ_i to decide the destination each task should be dispatched onto (lines 2-16). Taking the q-th task in ψ_i (i.e., t_i^q) for example. The energy consumed by t_i^q on the MD can be calculated according to (11) and is then added to E_{sum} (lines 3-4). Next, we calculate the time and energy overheads that would be incurred by assigning t_i^q to the edge server (line 5) and to the cloud (line 6), respectively. The offloading decision is accordingly to select the offloading destination leading to lower energy consumption while fulfilling the deadline requirement (lines 7-12). In case t_i^q cannot be completed ahead of the deadline whichever destination it is scheduled onto, ψ_i is identified as an infeasible solution (lines 13-16). Once having assigned all the tasks in ψ_i, TAS returns the scheduling objective value (i.e., the total energy consumption).

We further use a simple illustrative example to explain the fundamental flow of the task assignment strategy. Assuming that there are three tasks to be scheduled. The input task sequence ψ_i is denoted by $\psi_i = <t^2, t^1, t^3>$. The completion time of each task cannot exceed a pre-specified deadline value $DL = 40$ms. TAS determines the scheduling destinations, one by one, for all tasks in ψ_i. Starting with task t^2, we evaluate the energy consumption and completion time for scheduling it onto the cloud or edge server, respectively. Suppose that in the former case, the two metrics are $E^2_{\text{cloud}} = 5$mJ and $T^2_{\text{cloud}} = 30$ms, whereas those in the latter case are $E^2_{\text{edge}} = 3$mJ and $T^2_{\text{edge}} = 22$ms. Since T^2_{cloud} and T^2_{edge} both satisfy the deadline constraint and E^2_{edge} is lower than E^2_{cloud}, TAS makes the decision that t^2 should be scheduled onto the edge server. For the second task t^1, suppose that E^1_{cloud}, T^1_{cloud}, E^1_{edge}, and T^1_{edge} are 13mJ, 42ms, 8mJ, and 32ms, respectively. Since T^1_{cloud} exceeds the deadline value, TAS has to assign t^1 to the edge server for execution. In the same manner, if E^3_{cloud}, T^3_{cloud}, E^3_{edge}, and T^3_{edge} for task t^3 are 10mJ, 36ms, 7mJ, and 42ms, respectively. TAS would designate the cloud for t^3's execution in order to avoid deadline violation. Once having determined the destinations for all tasks in ψ_i, TAS returns the total energy consumption summed over all tasks.

Algorithm 2: ECFA.

Input: A set of tasks T, population size $|\Omega|$, the maximum number of iterations $G_{\max}$;
Output: The lowest total energy E_{best};
/* Step 1: Initialize firefly population and current best solution */
1 Generate all $|\Omega|$ fireflies in Ω randomly;
2 Create an initial best solution φ_{best} by using the STF rule;
3 Use Algorithm 1 to evaluate φ_{best}'s objective value E_{best};
/* Step2: iterate the search process */
4 Set $g \leftarrow 0$;
5 **while** $g < G_{\max}$ **do**
6 **for** $i = 1$ *to* $|\Omega|$ **do**
7 Update f_i's position by Eq. (21);
8 Map f_i onto its task sequence φ_i;
9 Use Algorithm 1 to calculate φ_i's objective E_i;
10 **if** $E_i < E_{\text{best}}$ **then**
11 Set $E_{\text{best}} \leftarrow E_i$;
12 Set $\varphi_{\text{best}} \leftarrow \varphi_i$;
13 Record f_i as the current best firefly;
14 Set $g \leftarrow g + 1$;
15 **return** E_{best};

D. Position Update Strategy

In the FA framework, each firefly f_i ($i = 1, 2, \ldots, |\Omega|$) moves toward all other brighter fireflies in the population. For any firefly f_j whose fitness value is higher than f_i ($j \neq i$), f_i updates its position in any dimension by

$$x_i^{\text{new}} = x_i + \beta e^{-\gamma r_{ij}^2}(x_j - x_i) + \alpha\left(\varepsilon - \frac{1}{2}\right), \tag{19}$$

where the first term is the current position of f_i, the second term is the movement from f_i's current position to its new position due to f_j's brightness attraction, and the third term stands for a certain random movement. Here, α and β are two coefficients satisfying $\alpha + \beta = 1$, and r_{ij} denotes the euclidean distance between f_i and f_j:

$$r_{ij} = \sqrt{\sum_{l=1}^{N}(x_{il} - x_{jl})^2}, \tag{20}$$

where l indicates the index of position dimension. The major computation burden in (19) is accounted for by the distance calculation with computational complexity $\mathcal{O}(N)$ where N is the total number of tasks. Note that for each firefly individual, the position update in (19) needs to iterate over every other firefly brighter than it. Thus, the calculation of euclidean distance has to be repeated $\frac{|\Omega|(|\Omega|-1)}{2}$ times for the entire population during one iteration.

To avoid the frequent distance calculations analyzed above, we introduce a new linear update scheme in which any firefly only moves toward the firefly with the highest brightness in the population. To be specific, for any dimension, the new update rule used in our proposed ECFA is defined as

$$x_i^{\text{new}} = x_i + \beta(x_{\text{best}} - x_i) + \alpha\left(\varepsilon - \frac{1}{2}\right), \tag{21}$$

where x_{best} denotes the position of the brightest firefly in the population. By replacing the exponential term involving distance calculation by a linear term, the computational complexity of updating one firefly's position is merely $\mathcal{O}(1)$. Compared with the standard FA, the linear update rule in ECFA reduces the computational complexity of updating the positions of all fireflies in the population from $\mathcal{O}(|\Omega|^2 \cdot N)$ to $\mathcal{O}(|\Omega|)$.

E. The Proposed ECFA

We now present our proposed ECFA for solving the cloud-edge scheduling problem, of which the algorithmic details are provided in Algorithm 2. Built upon the probability-based mapping operator, task assignment strategy, and position update rule that have been elaborated previously, ECFA uses an iterative procedure to explore the best scheduling solution leading to the lowest energy consumption.

ECFA starts with initializing the firefly population as well as the current best solution (lines 1-4). Specifically, the initial best solution φ_{best} is obtained by using the shortest task first (STF) rule. The objective value of φ_{best} can be evaluated by performing Algorithm 1. ECFA iterates the following procedure to explore high-quality solutions (lines 5-14). During each iteration, ECFA first updates each firefly's position by the new position update strategy (line 7), converts it into a valid task sequence (line 8), and evaluates its objective value (line 9). Once an individual firefly with lower energy consumption is identified, the current best firefly along with its objective value would be updated accordingly (lines 10-13). The iterative procedure terminates when the max number of iterations is reached. The energy consumption of the best solution explored so far is returned as the final result.

F. Complexity Analysis

The computational complexity of the proposed ECFA is analyzed as follows. First, the complexity of generating the initial firefly population (at random) and initial best solution (by using STF) are $\mathcal{O}(N)$ and $\mathcal{O}(N \cdot \log N)$, respectively. Next, the complexity of the probability-based mapping operator in this work is $\mathcal{O}(N)$, which is significantly lower than that of the ROV rule $\mathcal{O}(N \cdot \log N)$. Then, the position update strategy also has a linear complexity $\mathcal{O}(1)$ due to the linear update rule defined in Section IV-D. Compared with the conventional update rule, which is of $\mathcal{O}(N)$ complexity, ECFA achieves higher computational efficiency, which will be later justified by experimental results. In addition, the complexity of the task assignment heuristic for evaluating solution quality is $\mathcal{O}(N)$. To conclude, the complexity of the entire ECFA flow is determined as $\mathcal{O}(G_{\max} \cdot |\Omega| \cdot N)$, where $G_{\max}$ and $|\Omega|$ stand for the maximum number of generations and the number of fireflies in the population, respectively.

V. Proof of Convergence

This section provides theoretical proof of ECFA's convergence. Similarly as in the convergence analyses given in [35], [36], we assume the global best firefly's position x^* is constant. Also, the subsequent analysis applies to a firefly position of an arbitrary dimension.

Lemma 1: For any firefly f_i ($i \in \{1, 2, \ldots, |\Omega|\}$), its positions among all iterations x_i^t ($t = 0, 1, \ldots, +\infty$) follows a geometric sequence with the common ratio of $1 - \beta$.

Proof: Letting $c = \beta x^* + \alpha(\varepsilon - \frac{1}{2})$, we rewrite the position update equation (i.e., (21)) as

$$\underbrace{x_i^{t+1} - \frac{c}{\beta}}_{\text{(t+1)-th term}} = \underbrace{(1-\beta)}_{\text{common ratio}} \underbrace{\left(x_i^t - \frac{c}{\beta}\right)}_{\text{t-th term}}. \tag{22}$$

We can observe that $x_i^0 - \frac{c}{\beta}$, $x_i^1 - \frac{c}{\beta}$, …, $x_i^t - \frac{c}{\beta}$, $x_i^{t+1} - \frac{c}{\beta}$ form a geometric sequence. The common ratio of this geometric sequence is $1 - \beta$. The general term can be expressed as

$$x_i^{t+1} = (1-\beta)^{t+1}\left(x_i^0 - \frac{c}{\beta}\right) + \frac{c}{\beta}. \tag{23}$$

Based on Lemma 1, we deduce the following theorem.

Theorem 1: When $\beta \in (0, 1) \cup (1, 2)$, ECFA converges to the global optimal position x^*.

Proof: Let q be the common ratio, i.e., $q = 1 - \beta$. According to (23), we have

$$\lim_{t \to +\infty} x_i^{t+1} = \lim_{t \to +\infty} q^{t+1}\left(x_i^0 - \frac{c}{\beta}\right) + \frac{c}{\beta}. \tag{24}$$

To guarantee ECFA's convergence, $\lim_{t\to+\infty} x_i^{t+1}$ must exist. Therefore, q needs to meet $|q| < 1$. Since $q = 1-\beta$, we have $|1-\beta| < 1$, and thus $\beta \in (0, 1) \cup (1, 2)$. Accordingly, (24) can be further expressed as

$$\lim_{t \to +\infty} x_i^{t+1} = \frac{c}{\beta} = x^* + \frac{\alpha}{\beta}\left(\varepsilon - \frac{1}{2}\right). \tag{25}$$

Since ε is a random parameter uniformly distributed within the range [0, 1], its expected value is $\frac{1}{2}$. Therefore, we have $\lim_{t\to+\infty} x_i^{t+1} = x^*$. Thus, each firefly will consequently converge to the optimal position when $\beta \in (0, 1) \cup (1, 2)$.

VI. Trajectory Analysis

This section investigates the firefly's moving trajectories with regard to the optimal firefly position and ECFA-associated parameter values. For this reason, we introduce the concept of boundary trap to characterize the behavior of ECFA's searching procedure getting stuck in local optima. In the same manner, we assume that x^* is constant without any stochastic component [35], [36].

A. Boundary Trap

We use $[x_{\min}, x_{\max}]$ to indicate the boundary of the search region in a certain dimension of the firefly position. During the iterations, in case the firefly position is beyond the upper bound $x_{\max}$, we truncate the position value at $x_{\max}$. Likewise, firefly positions less than the lower bound $x_{\min}$ will be all replaced by $x_{\min}$. We define the boundary traps in the following four situations.

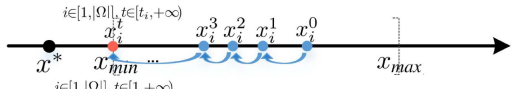
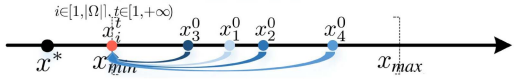
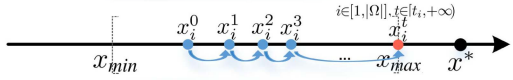
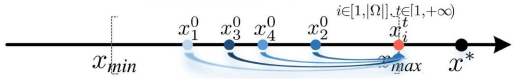
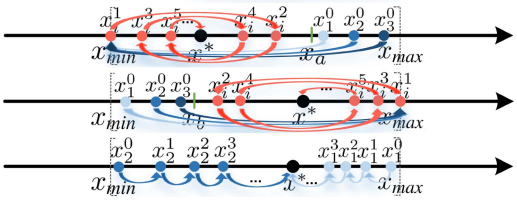
- BT-A: Once $\exists \nu = \arg\min_{\nu \in N^+} x_i^\nu = x_{\min}$, we identify a boundary trap $x_i^t = x_{\min}$, where $t = \nu, \nu + 1, \ldots, +\infty$.
- BT-B: Once $\exists \kappa = \arg\min_{\kappa \in N^+} x_i^\kappa = x_{\max}$, we identify a boundary trap $x_i^t = x_{\max}$, where $t = \kappa, \kappa + 1, \ldots, +\infty$.
- BT-C: All fireflies satisfying $\exists \varsigma_i = \arg\min_{\varsigma_i \in N^+} x_i^{\varsigma_i} = x_{\min}$ form a set $\Phi = \{f_1, f_2, \ldots, f_\varphi\}$. For all fireflies in set Φ, we identify a boundary trap $x_1^{\varsigma_1} = x_2^{\varsigma_2} = \ldots = x_\varphi^{\varsigma_\varphi} = x_{\min}$, $x_1^{\varsigma_1+1} = x_2^{\varsigma_2+1} = \ldots = x_\varphi^{\varsigma_\varphi+1}$, …, $x_1^{\varsigma_1+2} = x_2^{\varsigma_2+2} = \ldots = x_\varphi^{\varsigma_\varphi+2}$, etc.
- BT-D: All fireflies satisfying $\exists \tau_i = \arg\min_{\tau_i \in N^+} x_i^{\tau_i} = x_{\max}$ form a set $\Psi = \{f_1, f_2, \ldots, f_\psi\}$. For all fireflies in set ψ, we identify a boundary trap $x_1^{\tau_1} = x_2^{\tau_2} = \ldots = x_\psi^{\tau_\psi} = x_{\max}$, $x_1^{\tau_1+1} = x_2^{\tau_2+1} = \ldots = x_\psi^{\tau_\psi+1}$, …, $x_1^{\varsigma_1+2} = x_2^{\tau_2+2} = \ldots = x_\psi^{\tau_\psi+2}$, etc.

For the sake of clarity, we summarize the four boundary traps into two categories. BT-A and BT-B can be attributed as one trapping scenario: once a firefly encounters the search boundary ($x_{\min}$ or $x_{\max}$) in a particular iteration, this firefly would be trapped in this boundary in all subsequent iterations. BT-C and BT-D are classified as the other trapping scenario: for the fireflies that encounter the search boundary ($x_{\min}$ or $x_{\max}$), the trajectories of all these fireflies would be identical since the particular iteration in which they each encounter the search boundary.

B. Trajectory Analysis

Based on the definitions of boundary traps in Section VI-A, we analyze in a theoretical manner whether ECFA would fall into

TABLE III
A SUMMARY OF BOUNDARY TRAPPING CASES OF ECFA UNDER DIFFERENT β AND x^* VALUES

	x^* value	β value	boundary trap	illustration of firefly trajectories
Case 1	$(-\infty,\ x_{\min})$	(0, 1)	BT-A	
Case 2	$(-\infty,\ x_{\min})$	(1, 2)	BT-C	
Case 3	$(x_{\max},\ +\infty)$	(0, 1)	BT-B	
Case 4	$(x_{\max},\ +\infty)$	(1, 2)	BT-D	
Case 5	$[x_{\min},\ x_{\max}]$	(1, 2)	BT-C BT-D	
Case 6	$[x_{\min},\ x_{\max}]$	(0, 1)	No	

any boundary traps depending on the position of the optimal firefly and the value of parameter β.

Theorem 2: If $x^* \notin [x_{\min},\ x_{\max}]$, ECFA would fall into boundary traps. To be specific,

2.1: if $x^* < x_{\min}$ and $\beta \in (0,\ 1)$, ECFA would fall into BT-A;

2.2: if $x^* < x_{\min}$ and $\beta \in (1,\ 2)$, ECFA would fall into BT-C;

2.3: if $x^* > x_{\max}$ and $\beta \in (0,\ 1)$, ECFA would fall into BT-B;

2.4: if $x^* > x_{\max}$ and $\beta \in (1,\ 2)$, ECFA would fall into BT-D.

Proof: According to (23), the position of firefly f_i in the t-th iteration is given by

$$x_i^t = \begin{cases} x_{\min}, & \text{if } q^t(x_i^0 - \frac{c}{\beta}) + \frac{c}{\beta} \le x_{\min} \\ q^t\left(x_i^0 - \frac{c}{\beta}\right) + \frac{c}{\beta}, & \text{if } x_{\min} < q^t(x_i^0 - \frac{c}{\beta}) + \frac{c}{\beta} < x_{\max} \\ x_{\max}, & \text{if } q^t(x_i^0 - \frac{c}{\beta}) + \frac{c}{\beta} \ge x_{\max}, \end{cases} \tag{26}$$

where $q = 1 - \beta$ and $x_i^0 \in [x_{\min}, x_{\max}]$ is the initial solution. Due to the fact that $c = \beta x^* + \alpha(\varepsilon - \frac{1}{2})$ and $E[\varepsilon] = \frac{1}{2}$, (26) can be further described as

$$x_i^t = \begin{cases} x_{\min}, & \text{if } x^* + q^t(x_i^0 - x^*) < x_{\min} \\ x^* + q^t(x_i^0 - x^*), & \text{if } x_{\min} \le x^* + q^t(x_i^0 - x^*) \le x_{\max} \\ x_{\max}, & \text{if } x^* + q^t(x_i^0 - x^*) > x_{\max}. \end{cases} \tag{27}$$

Let $\Delta x_i = q^t(x_i^0 - x^*)$, $\forall i \in \{1, 2, \ldots, |\Omega|\}$. We prove the following four theorems based on (27).

Theorem 2.1: In case that $x^* < x_{\min}$ and $\beta \in (0,\ 1)$, since $q = 1 - \beta > 0$ and $x_i^0 - x^* > 0$, we know that $\Delta x_i > 0$ and Δx_i decreases with the increase of t. Once $\Delta x_i = x_{\min} - x^*$, f_i encounters the search boundary $x_{\min}$. Assume that f_i encounters $x_{\min}$ in the t_i-th iteration. We derive that $q^{t_i}(x_i^0 - x^*) = x_{\min} - x^*$ and accordingly $t_i = \lceil \log_q \frac{x_{\min} - x^*}{x_i^0 - x^*} \rceil$. Next, we have $x_i^t = \Delta x_i + x^* \le x_{\min}$, $\forall t \in \{t_i, t_i + 1, t_i + 2, \ldots, +\infty\}$ and thus $x_i^{t_i} = x_i^{t_i+1} = \ldots = x_i^{+\infty} = x_{\min}$. Referring to Case 1 in Table III, we provide an illustration of firefly f_i's trajectory to indicate that f_i would be trapped in $x_{\min}$ since the t_i-th iteration. Moreover, this phenomenon takes place for every individual firefly in the population. Therefore, when $x^* < x_{\min}$ and $\beta \in (0,\ 1)$, ECFA falls into BT-A.

Theorem 2.2: In case that $x^* < x_{\min}$ and $\beta \in (1,\ 2)$, since $x_i^0 \in [x_{\min}, x_{\max}]$ and $q = 1 - \beta < 0$, we have $q(x_i^0 - x^*) < 0$. Then, we can derive $x_i^1 = x^* + q(x_i^0 - x^*) < x^* < x_{\min}$, $\forall i \in \{1, 2, \ldots, |\Omega|\}$, i.e., $x_1^1 = x_2^1 = \ldots = x_{|\Omega|}^1 = x_{\min}$. As $x_i^2 = (1 - \beta)(x_i^1 - \frac{c}{\beta}) + \frac{c}{\beta}$, we next have $x_1^2 = x_2^2 = \ldots = x_{|\Omega|}^2 = (1 - \beta)(x_{\min} - \frac{c}{\beta}) + \frac{c}{\beta}$. In the same manner, we deduce $x_1^3 = x_2^3 = \ldots = x_{|\Omega|}^3, x_1^4 = x_2^4 = \ldots = x_{|\Omega|}^4$, and so on. As indicated by Case 2 in Table III, all fireflies encounter the search boundary $x_{\min}$ after the first iteration, and remain trapped in there in all subsequent iterations. Case 2 in fact can be regarded as a special case of BT-C in which the trajectories of all fireflies merge into an identical position.

Theorem 2.3: In case that $x^* > x_{\max}$ and $\beta \in (0,\ 1)$, since $q = 1 - \beta > 0$, $x_i^0 \in [x_{\min}, x_{\max}]$, and $x^* > x_{\max}$, we know that $\Delta x_i = q^t(x_i^0 - x^*) < 0$ and Δx_i increases with the increase of the iteration t. Similarly as in the proof of Theorem 2.1, firefly f_i will encounter the search boundary $x_{\max}$ in the t_i-th iteration where $t_i = \log_q(\frac{x_{\max} - x^*}{x_i^0 - x^*})$. Due to the fact that $x^* + q^{t_i}(x_i^0 - x^*) > x_{\max}$ after the t_i-th iteration, we have $x_i^{t_i+1} = x_i^{t_i+2} = \ldots = x_i^{+\infty} = x_{\max}$. Therefore, when $x^* > x_{\max}$ and $\beta \in (0,\ 1)$, ECFA falls into BT-B. The corresponding illustration of firefly trajectories can be referred to Case 3 in Table III.

Theorem 2.4: In case that $x^* > x_{\max}$ and $\beta \in (1,\ 2)$, since $x_i^0 \in [x_{\min}, x_{\max}]$ and $q = 1 - \beta < 0$, we have $q(x_i^0 - x^*) > 0$ for any $i \in \{1, 2, \ldots, |\Omega|\}$. We further derive that $x^* + q(x_i^0 - x^*) > x^* > x_{\max}$, and thus $x_1^1 = x_2^1 = \ldots = x_{|\Omega|}^1 = x_{\max}$. Similar to the proof of Theorem 2.2, we can draw the conclusion that all fireflies encounter the search boundary $x_{\max}$ in the very first iteration and remain trapped since then. In other words, ECFA falls into a special case of BT-D in which all fireflies move toward an identical position. An illustration of the fireflies' trajectories in this situation is provided by Case 4 in Table III.

We draw the conclusion from Theorem 2 that, ECFA would inevitably fall into boundary traps if the search space does not contain the optimal solution. Thus, when pre-defining ECFA's search boundary, we have to guarantee that the search space covers the optimal solution to prevent ECFA from falling into boundary traps. We further analyze the trajectories of fireflies and possible boundary traps if this prerequisite can be satisfied. Since parameter β is required to be $\beta \in (0, 1) \cup (1, 2)$ to guarantee ECFA's convergence (refer to Theorem 1), we discuss the two situations in which β value is within (0, 1) and (1, 2), respectively.

Lemma 2: If $x^* \in [x_{\min}, x_{\max}]$ and $\beta \in (1, 2)$, ECFA would fall into boundary traps. To be specific,

2.1: for $x_i^0 \in (x_a, x_{\max})$, ECFA falls into BT-C;

2.2: for $x_i^0 \in (x_{\min}, x_b)$, ECFA falls into BT-D,

where $x_a = \frac{x_{\min}-c}{1-\beta}$, $x_b = \frac{x_{\max}-c}{1-\beta}$, and $c = \beta x^* + \alpha(\varepsilon - \frac{1}{2})$.

Proof: According to (23), we have

$$\begin{aligned} x_i^1 &= (1-\beta)\left(x_i^0 - \frac{c}{\beta}\right) + \frac{c}{\beta} \\ &= q\left(x_i^0 - \frac{c}{\beta}\right) + \frac{c}{\beta} \\ &= x^* + q(x_i^0 - x^*) + \Lambda, \end{aligned} \tag{28}$$

where $q = 1 - \beta$, $c = \beta x^* + \alpha(\varepsilon - \frac{1}{2})$ and $\Lambda = \frac{\alpha}{\beta}(1-q)(\varepsilon - \frac{1}{2})$. Here, Λ can be considered as a random disturbance component with extremely small values that can be neglected in the following analysis. Since $\beta \in (1, 2)$ in this situation, we have $q < 0$. According to (28), there exist the following two cases: (a) if $x_i^0 < x^*$, then $q(x_i^0 - x^*) > 0$, and $x_i^1 > x_*$; (b) if $x_i^0 > x^*$, then $q(x_i^0 - x^*) < 0$ and $x_i^1 < x^*$. Accordingly, we identify that $x_i^1 \notin (x_i^0, x^*)$ (if $x_i^0 < x^*$) or $x_i^1 \notin (x^*, x_i^0)$ (if $x^* < x_i^0$). However, we cannot identify whether $x_i^1 \in (x_{\min}, x_{\max})$ is true. With this concern in mind, we introduce two threshold values, x_a and x_b, to represent the critical situations in which x_i^1 is exactly equal to $x_{\min}$ and $x_{\max}$, respectively. Taking x_a for example, it can be calculated as follows.

$$x_i^1 = (1-\beta)\left(x_a - \frac{c}{\beta}\right) + \frac{c}{\beta} = x_{\min} \tag{29}$$

$$\Rightarrow x_a = \frac{x_{\min} - \frac{c}{\beta}}{1-\beta} + \frac{c}{\beta} = \frac{x_{\min} - c}{1-\beta}. \tag{30}$$

Likewise, x_b is determined by

$$x_b = \frac{x_{\max} - \frac{c}{\beta}}{1-\beta} + \frac{c}{\beta} = \frac{x_{\max} - c}{1-\beta}. \tag{31}$$

Due to the fact that $x_{\min} < x_{\max}$ and $1 - \beta < 0$, we know that $x_b < x_a$. Depending on the value of x_i^0, we discuss the following two possible scenarios:

1) $x_i^0 \in (x_a, x_{\max}]$: (28) and (30) lead to

$$\begin{aligned} x_i^1 &= (1-\beta)\left(x_i^0 - \frac{c}{\beta}\right) + \frac{c}{\beta} \\ &< (1-\beta)\left(\frac{x_{\min} - c}{1-\beta} - \frac{c}{\beta}\right) + \frac{c}{\beta} < x_{\min}. \end{aligned} \tag{32}$$

For any $x_i^0 \in [x_a, x_{\max}]$ ($i \in \{1, 2, \ldots, |\Omega|\}$), we have $x_i^1 = x_{\min}$. Since $x_i^t = (1-\beta)^t(x_i^1 - \frac{c}{\beta}) + \frac{c}{\beta}$, $\forall t = 2, 3, \ldots, +\infty$, we can deduce that for all fireflies whose initial positions are within $[x_a, x_{\max}]$, their trajectories would be exactly the same. As a result, ECFA falls into BT-C in this scenario.

2) $x_i^0 \in [x_{\min}, x_b)$: (28) and (31) lead to

$$\begin{aligned} x_i^1 &= (1-\beta)\left(x_i^0 - \frac{c}{\beta}\right) + \frac{c}{\beta} \\ &> (1-\beta)\left(\frac{x_{\max} - c}{1-\beta} - \frac{c}{\beta}\right) + \frac{c}{\beta} > x_{\max}. \end{aligned} \tag{33}$$

Similarly, for all fireflies whose initial positions are within $[x_{\min}, x_b]$, their trajectories would be identical, indicating that ECFA falls into BT-D.

Case 5 in Table III illustrates the firefly trajectories in the aforementioned two scenarios. Note that, the trajectory in red color represents the identical trajectory that all fireflies whose initial positions are within $[x_a, x_{\max}]$ (or $[x_{\min}, x_b]$) would move along. Lemma 2 suggests that for $\beta \in (1, 2)$, if the initial position x_i^0 is within $[x_{\min}, x_b)$ (or $(x_a, x_{\max}]$), it will be determined as the boundary value $x_{\max}$ (or $x_{\min}$). The reason is that the position in the first iteration is outside of the search space, resulting in identical firefly trajectories. Falling into such boundary traps will significantly degrade the diversity of the population, thereby affecting ECFA's search effectiveness.

We now analyze the only unvisited situation, in which $x^* \in [x_{\min}, x_{\max}]$ and $\beta \in (0, 1)$. We present the following theorem to prove that ECFA is capable of escaping from all boundary traps in this particular situation.

Theorem 3: If $x^* \in [x_{\min}, x_{\max}]$ and $\beta \in (0, 1)$, ECFA would not fall into any boundary traps.

Proof: According to (23), we express the firefly position in the t-th iteration as

$$\begin{aligned} x_i^t &= (1-\beta)^t\left(x_i^0 - \frac{c}{\beta}\right) + \frac{c}{\beta} \\ &= q^t x^* + (1-q^t)\left(x^* + \frac{c}{\beta}\right) \\ &= x^* + q^t(x_i^0 - x^*) + \Xi, \end{aligned} \tag{34}$$

where $\Xi = \frac{\alpha}{\beta}(1-q^t)(\varepsilon - \frac{1}{2})$, which is a negligible random disturbance component. Since β value is restricted within (0, 1), we have $q \in (0, 1)$ and $q^t \in (0, 1)$. Depending on the initial position, if $x_i^0 < x^*, \forall t = 1, 2, \ldots, +\infty$, we can derive $q^t(x_i^0 - x^*) < 0$ and $x_i^0 < x_i^t < x^*$. On the contrary, if $x_i^0 > x^*, \forall t = 1, 2, \ldots, +\infty$, we have $q^t(x_i^0 - x^*) > 0$ and $x_i^0 > x_i^t > x^*$. We observe that regardless of the initial position x_i^0, the value of x_i^t is bounded by x_i^0 and x^*. Since x_i^0 and x^* are both within $[x_{\min}, x_{\max}]$, we obtain that $x_i^t \in [x_{\min}, x_{\max}], \forall t = 1, 2, \ldots, +\infty, \forall i = 1, 2, \ldots, |\Omega|$. In other words, ECFA would not fall into any boundary traps.

Moreover, as $q > 0$, the value of the term $q^t|x_i^0 - x^*|$ decreases with the increasing t value. This observation validates that the fireflies gradually move closer to the position of the optimal solution as the iteration proceeds.

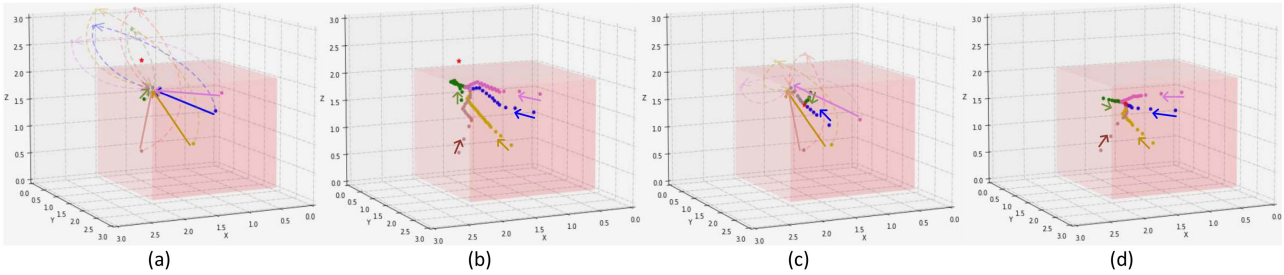


Fig. 2. An illustrative three-dimensional example of firefly trajectories: (a) x^* is outside of search space and $\beta \in (1, 2)$; (b) x^* is outside of search space and $\beta \in (0, 1)$; (c) x^* is inside search space and $\beta \in (1, 2)$; (d) x^* is inside search space and $\beta \in (0, 1)$.

To finalize the trajectory analysis, Table III summarizes all possible boundary trap situations under different values of parameter β and position x^*. Cases 1-4 reveal that ECFA inevitably falls into certain boundary traps as long as the optimal solution is not included in the search space, i.e., $x^* \notin [x_{\min}, x_{\max}]$. In the opposite situation, i.e., $x^* \in [x_{\min}, x_{\max}]$, depending on β value, falling with boundary traps occurs when $\beta \in (1, 2)$. Consequently, ECFA's parameter β must be carefully determined within (0, 1) to avoid any boundary traps. For the purpose of an intuitive interpretation, Fig. 2 further provides an illustrative example of firefly trajectories in a three-dimensional search space in different situations. The red star specifies the position of the optimal solution, and each colored spot denotes the movement of a firefly's position. When the optimal solution is outside of the search space, the selection of β value may guide the firefly trajectories to certain boundary traps, e.g., BT-C/BT-D ($\beta \in (1, 2)$ in Fig. 2(a)) and BT-A/BT-B ($\beta \in (0, 1)$ in Fig. 2(b)). Even if the optimal solution is inside the search space, inappropriate β value may still cause the boundary traps to happen ($\beta \in (1, 2)$ in Fig. 2(c)). Only when $\beta \in (0, 1)$ is also satisfied, all fireflies gradually approach and eventually reach the position of the optimal solution (Fig. 2(d)).

VII. Evaluation Results

To evaluate the performance of our proposed ECFA, we implement simulations by Java programming. All experiments were performed on a desktop machine with a 10-core CPU and 32-GB memory. Experimental settings are as follows.

In terms of testing instance set, we design ten groups of testing instances, in which the number of tasks to be scheduled are 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100, respectively. Moreover, there are ten different testing instances in each group. For all the tasks in the testing instance set, their characteristics d_i and w_i are uniformly distributed over [0, 200] bits and [1000, 2000] CPU cycles/bit, respectively. Six security policies of different protection levels are considered in the experiments, with details of the parameters available in [26]. In the cloud-edge system, the CPU operating frequency of the MD, the edge server, and the cloud are 1.33GHz, 3.3GHz, and 4.0GHz, respectively. The power consumption of the MD transmitting data to the edge server is 0.1W, whereas that of the edge server transmitting data to the cloud is 0.2W [22]. The operating powers of the MD, the edge server, and the cloud are 1.65×10^{-9} Joules/cycle, 2.50×10^{-8} Joules/cycle, and 2.88×10^{-8} Joules/cycle, respectively [5]. Regarding the three energy components in (14), we set their weighting factors as $\delta = 0.5$, $\varrho = 0.3$, and $\eta = 0.2$. For the sake of a fair comparison, all metaheuristic algorithms used for performance evaluation use identical parameter settings, in which the population size is 30 and the maximum number of iterations is 500. The range of firefly position in any dimension is set to [-2, 2]. Other parameters associated with ECFA will be determined in subsequent experiments.

Since the proposed ECFA is a metaheuristic involving randomness, we use the analysis of variance (ANOVA) method [37] to evaluate algorithm performance in a statistical manner. To be specific, we repeat each algorithm to be evaluated for multiple rounds, and record the values of optimization objectives obtained in all rounds. In ANOVA, the metric of relative percentage deviation (RPD) is used to quantify the algorithm's effectiveness, which is defined as

$$RPD = \frac{1}{K}\left(\sum_{k=1}^{K} \frac{E_{\text{sum}}^{k} - E_{\text{sum}}^{*}}{E_{\text{sum}}^{*}}\right) \times 100\%, \tag{35}$$

where E_{sum}^{k} is the total energy obtained in the k-th round and E_{sum}^{*} is the lowest energy in all K replications. The lower the value of RPD is, the better the effectiveness of the algorithm has. We use $K = 5$ in all evaluation experiments.

A. Determination of Important Parameters

1) Parameter β: Parameter β, which represents the attractiveness between any two fireflies, is a critical parameter in our proposed linear update rule (refer to (21)). As justified by the trajectory analysis in Section V, the range of β should be within [0, 1] to ensure ECFA's convergence. For this reason, we investigate the mean RPDs and least-significant difference (LSD) intervals with 95% confidence of ECFA under different β values, i.e., $\beta = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. The results in Fig. 3 verify that ECFA achieves the lowest mean RPD when the β value is 0.1. Thus, we use $\beta = 0.1$ in all subsequent experiments.

2) Parameter m: As one of ECFA's distinguishing properties, the new position update strategy forces each firefly only to move toward the brightest firefly to reduce computational burden (refer to (21)). To validate this point, we use m to denote the number of brightest fireflies that each individual firefly is attracted by, and investigate the impact of parameter m on

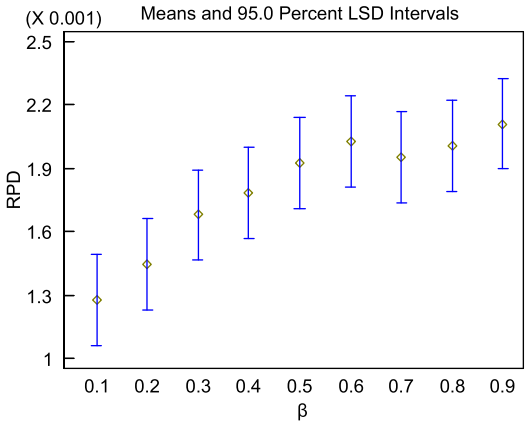


Fig. 3. RPDs obtained by ECFA with different β values.

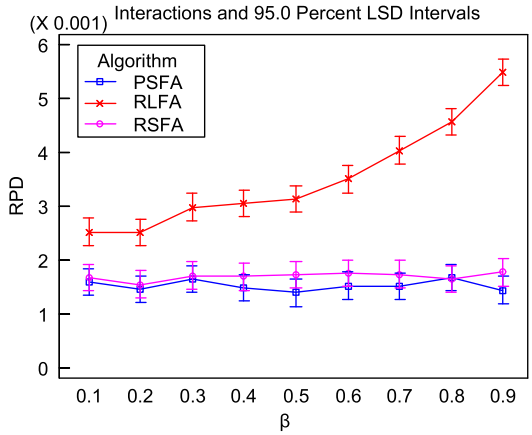


Fig. 6. RPDs obtained by PSFA, RLFA, and RSFA with different β values.

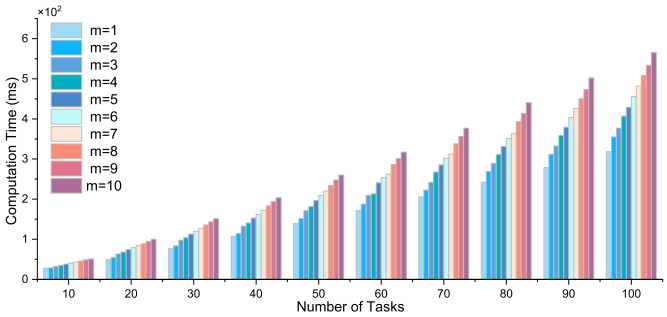


Fig. 4. The runtime statistics for ECFA using different m values on testing instances of different scales.

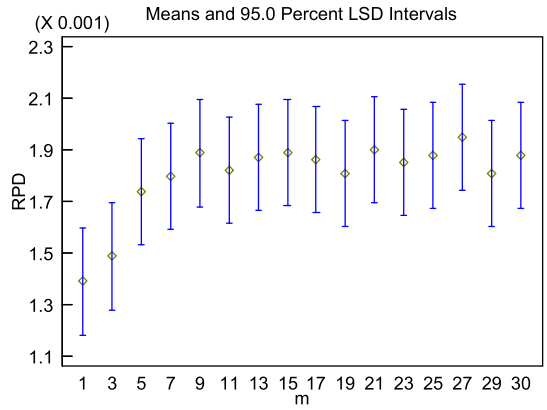


Fig. 5. RPDs obtained by ECFA with different m values.

algorithm performance in terms of effectiveness and efficiency, respectively.

On the one hand, we set different m values and evaluate the mean RPDs and 95% LSD intervals obtained by using ECFA. As indicated by the results in Fig. 5, ECFA achieves the best performance when $m = 1$ as a lower RPD value indicates greater effectiveness. On the other hand, we examine the computation times by using ECFA on various testing instances of different scales. Also, we choose ten m values for evaluation, i.e., $m \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. The runtime statistics in Fig. 4 show that, for each m value, the computation time increases as the number of tasks grows. Also, on the same testing instance, greater m values lead to more overheads in computation time. This overhead becomes more significant as the number of tasks continues to increase. The experiments performed in this section justify that, using only the brightest firefly to attract other fireflies (i.e., $m = 1$) achieves not only the lowest RPD but also the shortest computation time.

B. Evaluation of Scheduling Performance

1) Competing Algorithms: FA-Based Algorithms: We create three FA-based competing algorithms for comparison purposes. To evaluate the computational efficiency of ECFA's new position update strategy, the first FA variant uses the probability-based mapping method and the standard position update rule, and is denoted by PSFA. To evaluate the effectiveness of the mapping operator in ECFA, the second FA variant, denoted by RLFA, uses the conventional ROV mapping method [9], [32] and the new position update strategy. The third competing algorithm is the standard FA that uses ROV mapping and standard position update strategy, which is regarded as the baseline algorithm denoted by RSFA. We also evaluate the parameter setting for these three competing algorithms. As shown in Fig. 6, RLFA achieves the lowest RPD values when $\beta = 0.2$, whereas PSFA and RSFA show insignificant differences in PRD under different β values. Therefore, we set β value to 0.2 for all three competing algorithms in the following experiments. Parameter m is set as 1, which is consistent with ECFA's parameter setting.

Representative Metaheuristic Algorithms: We further compared our ECFA with four other population-based metaheuristics for solving cloud-edge scheduling problems.

- Distributed particle swarm optimization (DPSO) [38]: This metaheuristic divides the original population into multiple sub-populations and applies adaptive parameters to update the velocity and position of particles.
- Genetic algorithm (GA) [39]: This metaheuristic employs standard GA evolutionary mechanisms, i.e., crossover, mutation, and selection operations.

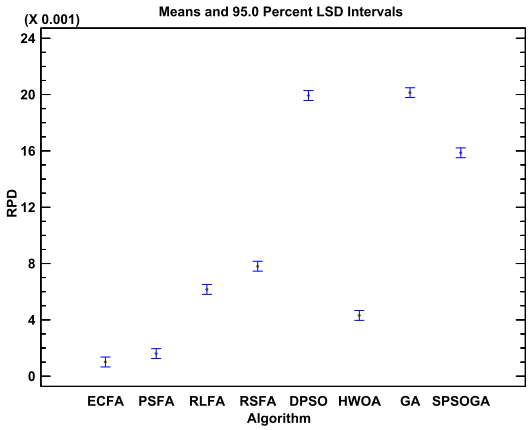


Fig. 7. RPDs obtained by ECFA and competing algorithms.

- Hybrid whale optimization algorithm (HWOA) [40]: This metaheuristic incorporates the position update strategy of the dragonfly algorithm and the mutation operator of GA into the whale optimization algorithm framework.
- Self-adaptive particle swarm optimization algorithm using the genetic algorithm operators (SPSOGA) [7]: This metaheuristic is a hybrid of GA and PSO, which incorporates the mutation and crossover operations into the particle update process.

2) Performance Evaluation: We start with evaluating the effectiveness of ECFA by using the RPD metric. Fig. 7 provides the mean values and LSD intervals of RPDs obtained by our proposed ECFA and all competing algorithms. The non-overlapping LSD intervals among these algorithms indicate that the performance of ECFA is significantly higher than those of all non-FA-based metaheuristics. The reason mainly lies in ECFA's convergence feature, which is beneficial for enhancing ECFA's solution exploration capability. Regarding the comparison with the three FA-based algorithms, ECFA still performs better than PSFS, RLFA, and RSFA. The difference between ECFA and PSFA is the least significant, as PSFA uses the same probability-based mapping operator as in our ECFA. However, the ROV mapping rule used by RLFA and RSFA substantially degrades their performances.

We further compare the computational efficiency between ECFA and other competing algorithms in solving the cloud-edge scheduling problem. To evaluate computational efficiency in a quantitative manner, we introduce a metric called normalized efficiency (NE), which is calculated by

$$NE = \frac{T}{T_{\text{baseline}}}, \tag{36}$$

where T represents the runtime by the algorithm to be evaluated, and T_{baseline} represents that by the baseline algorithm, i.e., RSFA. Each algorithm is repeated for R rounds on every single testing instance, and thus both T and T_{baseline} are averaged over the R replications. As mentioned previously, we chose RSFA as the baseline algorithm. The definition in (36) means that a lower NE value corresponds to a higher computational efficiency.

Table IV lists the average runtimes as well as the NE metrics obtained by different algorithms on all testing instances.

TABLE IV
AVERAGE RUNTIMES AND MEAN NES OF ECFA AND COMPETING ALGORITHMS

	Runtime (ms)	NE
ECFA	123.443	0.139
PSFA	893.418	1.006
RLFA	249.560	0.281
RSFA	888.074	1.000
DPSO	123.778	0.139
HWOA	182.356	0.205
GA	95.040	0.107
SPAOGA	101.156	0.114

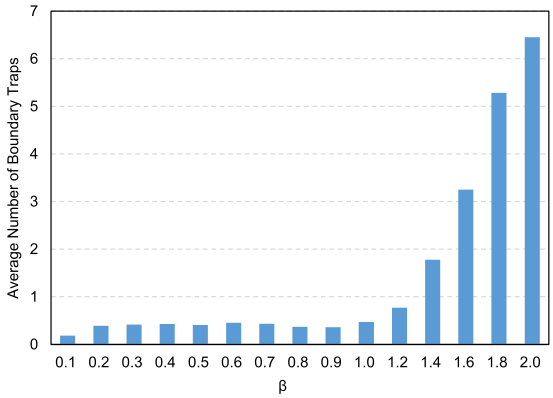


Fig. 8. The average occurrence number of encountering search boundaries with different β values.

The comparison results confirm that our ECFA is the most computationally efficient among all FA-based algorithms. The NE value by using ECFA is 0.139, indicating a 7.19× speedup achieved by ECFA over the baseline RSFA. This is because the new position update strategy in ECFA significantly reduces the computational burden compared to the standard position update strategy. For the same reason, despite the closest performance gap between PSFA and our proposed ECFA, ECFA substantially improves the computational efficiency with a 7.23× speedup over PSFA. On the other hand, the average NE of ECFA is close to those of other non-FA-based metaheuristics. In particular, since GA and SPAOGA use discrete binary encoding for solution representation and require no mapping operator, they are slightly more efficient than ECFA is. However, as revealed in Fig. 7, ECFA outperforms these two algorithms significantly in terms of solution quality.

The last set of experiments is to verify that the value range for parameter β determined by the theoretical analysis can effectively prevent ECFA from falling into boundary traps. We perform ECFA by using different β values and record the number of occurrences of encountering the search boundaries on average for all testing instances. As reported in Fig. 8, for $\beta \in (0, 1)$, the occurrence of ECFA falling into boundary traps is obviously less frequent than those for other parameter settings, which is consistent with our conclusion drawn in the trajectory analysis. In addition, when $\beta \in [1,2)$, we observe an increase in the average number of occurrences of boundary traps as the value of β increases. This observation is because a greater β value would

cause a larger search space and accordingly a higher possibility of encountering the search boundaries.

VIII. Conclusion

This paper proposes an efficient convergent firefly algorithm (ECFA) that incorporates a novel probability-based mapping method and an efficient position update strategy to solve task scheduling problems in cloud-edge computing. The proposed ECFA is capable of improving the searching efficiency as well as guaranteeing the scheduling performance. We provide a theoretical analysis to prove the convergence of ECFA by determining an appropriate range of the critical parameter. We also introduce a new concept of boundary trap to analyze the trajectories of firefly movements, for the purpose of predicting whether ECFA would fall into boundary traps under different parameter settings. Experimental results demonstrate that, compared with other FA-based algorithms and representative population-based metaheuristics, ECFA can produce high-quality scheduling solutions with promising computational efficiency. Evaluation results also justify that the parameter range determined by the trajectory analysis can effectively prevent ECFA from falling into boundary traps.

It is worth emphasizing that, this work can be readily extended to cope with more complicated cloud-edge systems scenarios considering server heterogeneity, device-to-device collaboration, etc. To facilitate the adaption of our proposed ECFA in such scenarios, it is necessary to re-design the task assignment strategy, which is used to calculate scheduling objective value and evaluate solution quality, depending on the specific scheduling scenario. Other fundamental operators in the FA-based evolutionary framework, such as solution representation, firefly mapping, and position update, can remain unchanged. The proof of convergence and trajectory analysis derived in this work would still be effective for the extensions of ECFA.

References

[1] X. Lin, Y. Wang, Q. Xie, and M. Pedram, "Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment," *IEEE Trans. Serv. Comput.*, vol. 8, no. 2, pp. 175–186, Mar./Apr. 2015.

[2] Z. Wu et al., "Scheduling-guided automatic processing of massive hyperspectral image classification on cloud computing architectures," *IEEE Trans. Cybern.*, vol. 51, no. 7, pp. 3588–3601, Jul. 2021.

[3] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Trans. Commun.*, vol. 66, no. 4, pp. 1594–1608, Apr. 2018.

[4] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surveys Tut.*, vol. 19, no. 3, pp. 1657–1681, Third Quarter 2017.

[5] Z. Kuang, L. Li, J. Gao, L. Zhao, and A. Liu, "Partial offloading scheduling and power allocation for mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6774–6785, Aug. 2019.

[6] P.-Q. Huang, Y. Wang, K. Wang, and Z.-Z. Liu, "A bilevel optimization approach for joint offloading decision and resource allocation in cooperative mobile edge computing," *IEEE Trans. Cybern.*, vol. 50, no. 10, pp. 4228–4241, Oct. 2020.

[7] X. Chen, J. Zhang, B. Lin, Z. Chen, K. Wolter, and G. Min, "Energy-efficient offloading for DNN-based smart IoT systems in cloud-edge environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 683–697, Mar. 2022.

[8] D. Kimovski, N. Mehran, C. E. Kerth, and R. Prodan, "Mobility-aware IoT application placement in the cloud – edge continuum," *IEEE Trans. Serv. Comput.*, vol. 15, no. 6, pp. 3358–3371, Nov./Dec. 2022.

[9] 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.

[10] A. Heidari, M. A. J. Jamali, N. J. Navimipour, and S. Akbarpour, "Internet of Things offloading: Ongoing issues, opportunities, and future challenges," *Int. J. Commun. Syst.*, vol. 33, no. 14, 2020, Art. no. e4474.

[11] Y. Laili, F. Guo, L. Ren, X. Li, Y. Li, and L. Zhang, "Parallel scheduling of large-scale tasks for industrial cloud–edge collaboration," *IEEE Internet Things J.*, vol. 10, no. 4, pp. 3231–3242, Feb. 2023.

[12] Z. Miao, P. Yong, Z. Jiancheng, and Y. Quanjun, "Efficient flow-based scheduling for geo-distributed simulation tasks in collaborative edge and cloud environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 3442–3459, Dec. 2022.

[13] Y. Chen, N. Zhang, Y. Zhang, and X. Chen, "Dynamic computation offloading in edge computing for Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4242–4251, Jun. 2019.

[14] Z. Ning, P. Dong, X. Kong, and F. Xia, "A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4804–4814, Jun. 2019.

[15] C. Dou, N. Huang, Y. Wu, and T. Q. Quek, "Energy-efficient hybrid NOMA-FDMA assisted distributed two-tier edge-cloudlet multi-access computation offloading," *IEEE Trans. Green Commun. Netw.*, early access, Feb. 24, 2023, doi: 10.1109/TGCN.2023.3248609.

[16] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *Proc. Int. Symp. Stochastic Algorithms*, 2009, pp. 169–178.

[17] Q.-T. Vien, T. A. Le, X.-S. Yang, and T. Q. Duong, "Enhancing security of MME handover via fractional programming and firefly algorithm," *IEEE Trans. Commun.*, vol. 67, no. 9, pp. 6206–6220, Sep. 2019.

[18] R. Tao, Z. Meng, and H. Zhou, "A self-adaptive strategy based firefly algorithm for constrained engineering design problems," *Appl. Soft Comput.*, vol. 107, 2021, Art. no. 107417.

[19] A. M. Altabeeb, A. M. Mohsen, L. Abualigah, and A. Ghallab, "Solving capacitated vehicle routing problem using cooperative firefly algorithm," *Appl. Soft Comput.*, vol. 108, 2021, Art. no. 107403.

[20] X. Li, C. Gu, Z. Yang, and Y. Chang, "Virtual machine placement strategy based on discrete firefly algorithm in cloud environments," in *Proc. Int. Comput. Conf. Wavelet Act. Media Technol. Inf. Process.*, 2015, pp. 61–66.

[21] C. Ren, Y. Huang, Q. Luo, and X. Li, "Resource scheduling in cloud computing based on firefly algorithm," in *Proc. Int. Conf. Ubiquitous Comput. Commun. Data Sci. Comput. Intell. Smart Comput. Netw. Serv.*, 2019, pp. 638–643.

[22] Y. Zhang, J. Zhou, L. Sun, J. Mao, and J. Sun, "A novel firefly algorithm for scheduling bag-of-tasks applications under budget constraints on hybrid clouds," *IEEE Access*, vol. 7, pp. 151 888–151 901, 2019.

[23] K. Sekaran, M. S. Khan, R. Patan, A. H. Gandomi, P. V. Krishna, and S. Kallam, "Improving the response time of M-learning and cloud computing environments using a dominant firefly approach," *IEEE Access*, vol. 7, pp. 30 203–30 212, 2019.

[24] S. N. Shirazi, A. Gouglidis, A. Farshad, and D. Hutchison, "The extended cloud: Review and analysis of mobile edge computing and fog from a security and resilience perspective," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2586–2595, Nov. 2017.

[25] B. Huang et al., "Security modeling and efficient computation offloading for service workflow in mobile edge computing," *Future Gener. Comput. Syst.*, vol. 97, pp. 755–774, 2019.

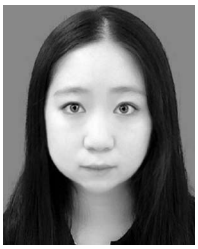
[26] Y. Zhang, Y. Liu, J. Zhou, J. Sun, and K. Li, "Slow-movement particle swarm optimization algorithms for scheduling security-critical tasks in resource-limited mobile edge computing," *Future Gener. Comput. Syst.*, vol. 112, pp. 148–161, 2020.

[27] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.

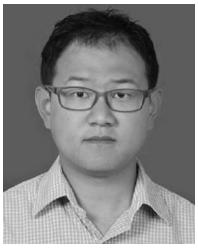
[28] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, and J. Rodriguez, "Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach," *IEEE Trans. Veh. Technol*, vol. 68, no. 4, pp. 3113–3125, Apr. 2019.

[29] U. Saleem, Y. Liu, S. Jangsher, Y. Li, and T. Jiang, "Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 360–374, Jan. 2021.

[30] H. Xing, L. Liu, J. Xu, and A. Nallanathan, "Joint task assignment and resource allocation for D2D-enabled mobile-edge computing," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4193–4207, Jun. 2019.
[31] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 9, pp. 5994–6009, Sep. 2017.
[32] 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.
[33] J. Liu, K. Xiong, D. W. K. Ng, P. Fan, Z. Zhong, and K. B. Letaief, "Max-min energy balance in wireless-powered hierarchical fog-cloud computing networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 11, pp. 7064–7080, Nov. 2020.
[34] M.-H. Chen, M. Dong, and B. Liang, "Resource sharing of a computing access point for multi-user mobile cloud offloading with delay constraints," *IEEE Trans. Mobile Comput.*, vol. 17, no. 12, pp. 2868–2881, Dec. 2018.
[35] F. Van den Bergh and A. P. Engelbrecht, "A study of particle swarm optimization particle trajectories," *Inf. Sci.*, vol. 176, no. 8, pp. 937–971, 2006.
[36] G. Xu, K. Luo, G. Jing, X. Yu, X. Ruan, and J. Song, "On convergence analysis of multi-objective particle swarm optimization algorithm," *Eur. J. Oper. Res.*, vol. 286, no. 1, pp. 32–38, 2020.
[37] L. Stahle and S. Wold, "Analysis of variance (ANOVA)," *Chemometrics Intell. Lab. Syst.*, vol. 6, no. 4, pp. 259–272, 1989.
[38] S. Meng et al., "Security-aware dynamic scheduling for real-time optimization in cloud-based industrial applications," *IEEE Trans. Ind. Inform.*, vol. 17, no. 6, pp. 4219–4228, Jun. 2021.
[39] M. Abbasi, M. Yaghoobikia, M. Rafiee, M. R. Khosravi, and V. G. Menon, "Optimal distribution of workloads in cloud-fog architecture in intelligent vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4706–4715, Jul. 2021.
[40] Y. Kang et al., "HWOA: An intelligent hybrid whale optimization algorithm for multi-objective task selection strategy in edge cloud computing system," *World Wide Web*, vol. 25, no. 5, pp. 2265–2295, 2022.



Lu Yin (Student Member, IEEE) received the BS degree in network engineering from the Nanjing University of Science and Technology, Nanjing, China, in 2018. She is currently working toward the MS degree with the School of Computer Science and Engineering, Nanjing University of Science and Technology. Her research interests include mobile edge computing and task scheduling.



Jin Sun (Member, IEEE) received the BS and MS degrees in computer science from the Nanjing University of Science and Technology, Nanjing, China, in 2004 and 2006, respectively, and the PhD degree in electrical and computer engineering from the University of Arizona in 2011. From 2012 to 2014, he was a member of technical staff with Orora Design Technologies, Inc. He is currently, a professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology. His research interests include cloud computing, edge computing and edge intelligence, and embedded systems.

Junlong Zhou (Member, IEEE) received the PhD degree in computer science and technology from East China Normal University, Shanghai, China, in 2017. He was a visiting scholar with the University of Notre Dame, Notre Dame, IN, USA, during 2014-2015. 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 embedded systems, cloud-edge-IoT, and cyber-physical systems. He has published 100 refereed papers, including nearly 40 in premier IEEE/ACM Transactions. He has held chair positions at many IEEE/ACM conferences. He has been an associate editor for the *Journal of Circuits, Systems, and Computers* and the *IET Cyber-Physical Systems: Theory & Applications*, a subject area editor for the *Journal of Systems Architecture: Embedded Software Design*, and a guest editor for 8 ACM/IET/Elsevier/Wiley Journals such as *ACM Transactions on Cyber-Physical Systems*. He received the Best Paper Award from IEEE CPSCom 2022 and IEEE iThings 2020.

Zonghua Gu (Senior Member, IEEE) received the PhD degree in computer science and engineering from the University of Michigan, Ann Arbor, MI, USA, in 2004. He is currently a professor with the Department of Applied Physics and Electronics, Umeå University, Umeå, Sweden. His research interests include cyber physical systems, real-time embedded systems, and machine learning.

Keqin Li (Fellow, IEEE) is currently a SUNY distinguished professor with the State University of New York and also a National distinguished professor with Hunan University (China). He has authored or coauthored more than 910 journal articles, book chapters, and refereed conference papers. He received several best paper awards from international conferences including PDPTA-1996, NAECON-1997, IPDPS-2000, ISPA-2016, NPC-2019, ISPA-2019, CPSCom-2022. He holds nearly 70 patents announced or authorized by the Chinese National Intellectual Property Administration. He is among the world's top five most influential scientists in parallel and distributed computing in terms of both single-year impact and career-long impact based on a composite indicator of Scopus citation database. He was a 2017 recipient of the Albert Nelson Marquis Lifetime Achievement Award for being listed in Marquis Who's Who in Science and Engineering, Who's Who in America, Who's Who in the World, and Who's Who in American Education for more than twenty consecutive years. He received the Distinguished Alumnus Award of the Computer Science Department at the University of Houston in 2018. He received the IEEE TCCLD Research Impact Award from the IEEE Technical Committee on Cloud Computing in 2022. He is an AAAS fellow, and an AAIA fellow. He is a member of SUNY Distinguished Academy. He is a member of Academia Europaea (Academician of the Academy of Europe).