Future Generation Computer Systems 112 (2020) 148–161

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Slow-movement particle swarm optimization algorithms for scheduling security-critical tasks in resource-limited mobile edge computing☆

Yi Zhang [a,b], Yu Liu [a], Junlong Zhou [a], Jin Sun [a,*], Keqin Li [c]

[a] School of Computer Science and Engineering, Nanjing University of Science and Technology, 200 Xiaolingwei Street, Nanjing 210094, China
[b] Lian Yungang E-Port Information Development Co., Ltd, Lian Yungang, China
[c] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

ARTICLE INFO

Article history:
Received 14 December 2019
Received in revised form 16 April 2020
Accepted 15 May 2020
Available online 21 May 2020

Keywords:
Mobile edge computing
Security-critical tasks
Scheduling algorithms
Particle swarm optimization

ABSTRACT

Mobile edge computing (MEC) allows mobile devices to offload computation tasks to nearby MEC servers for achieving low latency and energy efficiency. This paper aims at scheduling security-critical tasks, which require data encryption and thus incur extra runtime and energy costs, in a MEC system consisting of multiple resource-limited MEC servers. The scheduling objective is to minimize task completion time as well as the mobile device's energy consumption. We propose two slow-movement particle swarm optimization algorithms to solve the resultant NP-hard problem. Specifically, we develop a position-based mapping scheme to map particles onto scheduling solutions. The mapping method relies on the current best solution and a position-based probability model to generate high-quality solutions that can inherit the good schemata from the current best solution. To prevent the significant change in particles' positions, we further propose a novel particle updating strategy to slow down particles' movements, in order to explore more high-quality solutions under the guide of personal best particle and global best particle. Experimental results demonstrate that, the proposed algorithms significantly outperform the conventional particle swarm optimization algorithm in terms of both effectiveness and efficiency. Performance of the mapping method and the particle updating strategy are also investigated.

1. Introduction

The rapid development of mobile applications has imposed a heavy demand of computation capability of mobile devices. However, resource-limited mobile devices in general have limited computing resources and thus cannot meet this demand. Mobile edge computing (MEC)[1] [1] has emerged to address this issue, by allowing mobile devices to offload computation tasks to nearby MEC servers for achieving low latency and energy efficiency [2].

In a MEC system, multiple MEC servers are deployed along with base stations, and mobile devices communicate with base stations by radio access networks to offload tasks to MEC servers. Since it is impractical to deploy and maintain too many physical servers along with base stations, MEC servers are considered to have limited resources, in contrast to cloud platforms with infinity resources. On the other hand, due to the benefits of MEC systems, mobile users are willing to offload their tasks, leading to the fierce competition for the computing resources of MEC servers. To cope with this resource competition, MEC providers have to restrict user parallelism on MEC servers. In other words, concurrent processing tasks from a single user on any MEC server is not allowed in such MEC systems with multiple resource-limited MEC servers [3,4]. Technically, for any user, any MEC server can provide a virtual machine (or an application container) to execute the user's offloaded tasks following a certain policy, e.g. the commonly used first come first serve (FCFS) policy [5,6].

Security-critical tasks are a special type of tasks that require data encryption before offloading. When choosing the MEC server to offload these tasks, users tend to trust MEC providers with good reputations, rather than those with no or even bad reputations. In addition, users may choose different encryption algorithms, which in turn provide different levels of data protection, before offloading the tasks [7]. The key challenge in

☆ This work was supported in part by the National Natural Science Foundation of China under Grants 61872185 and 61802185, in part by the Natural Science Foundation of Jiangsu Province of China under Grant BK20180470, in part by the Fundamental Research Funds for the Central Universities under Grants 30919011233, 30919011402, and 30920021132, in part by the Jiangsu Planned Projects for Postdoctoral Research Funds under Grant 2019K025, and in part by the Open Research Project of the Hubei Key Laboratory of Intelligent Geo-Information Processing under Grant KLIGIP-2018A04.

* Corresponding author.
E-mail address: sunj@njust.edu.cn (J. Sun).

[1] MEC is also known as multi-access edge computing in the literature.

https://doi.org/10.1016/j.future.2020.05.025
0167-739X/

this scenario is how to schedule a user's multiple independent security-critical tasks in a resource-limited MEC system, with the objective of minimizing the completion time of all tasks (as known as makespan) as well as the energy consumption of the user's mobile device.

To tackle the above-mentioned challenge, this paper studies and solves the security-critical task scheduling problem in a resource-limited MEC system. We establish a rigorous optimization model for the considered problem, and prove it to be NP-hard. Accordingly, we propose two novel particle swarm optimization (PSO) algorithms using a newly constructed slow-movement particle updating strategy to solve the problem. The proposed algorithms employ a task permutation to represent a valid scheduling solution (or solution for simplicity). To evaluate the particle's fitness, we propose a position-based mapping scheme to convert a particle into a solution. A greedy heuristic is constructed to calculate the converted solution's objective that is regarded as the particle's fitness. The mapping method relies on the current best solution and a position-based probability model to generate a high-quality solution that can inherit good schemata from the current best solution. Moreover, we present a theoretically analytical approach to determine appropriate boundary values of particles' positions to ensure the inheritance effectiveness. Furthermore, to prevent the significant change in particles' positions, we propose a novel particle updating strategy to slow down particles' movements, in order to explore more high-quality solutions under the guide of the personal best particle and the global best particle. Since the proposed mapping method relies on the current best solution to perform particle mapping, we use two task sorting policies, longest task first (LTF) and biggest data task first (BTF), to generate the initial best solution, and define the corresponding slow-movement PSO algorithms as SPSO1 and SPSO2, respectively. We compare the proposed algorithms with the conventional PSO algorithm using the well-known ranked-order value (ROV) rule [8–10] and the standard particle updating strategy. Simulation results demonstrate that, both SPSO1 and SPSO2 generate better solutions to the considered problem. In addition, the proposed algorithms achieve over 2× speedups in computation time. Specifically, the mapping scheme outperforms the ROV rule in terms of both effectiveness and efficiency. The slow-movement strategy is significantly superior to the standard particle updating strategy. The LTF and BTF policies are capable of generating a qualified initial best solution, contributing to the high quality of the final scheduling solution. In summary, the main contributions of this paper are listed below.

- We establish a rigorous optimization model to formulate the security-critical task scheduling problem and prove it to be NP-hard.
- We present a position-based mapping scheme that utilizes particles' positions as well as the current best solution to convert particles into high-quality solutions.
- We develop a slow-movement particle updating strategy to improve the solution exploration effectiveness of the proposed scheduling algorithms.
- We propose two novel scheduling algorithms by employing the LTF/BTF solution initialization methods, the new position-based mapping scheme and the novel slow - movement particle updating strategy.

The reminder of this paper is organized as follows. Section 2 discusses existing scheduling algorithms for MEC systems. Section 3 formulates the optimization model for the considered problem. Sections 4 and 5 present the proposed algorithms and simulation results, respectively. Finally, concluding remarks and future works are given in Section 6.

2. Related work

In the literature, there are a number of existing algorithms aiming at task scheduling in single-user MEC systems [3,4,7, 11–14], multi-user MEC systems [19–37], and MEC plus clouds systems [15,38–40]. Comprehensive surveys are conducted in [2, 41,42]. Since the attention of this paper is focused on using PSO-based algorithms for scheduling security-critical tasks in a single-user resource-limited MEC system, in this section we discuss in detail existing studies that are closely related to this work.

Mao et al. [3] studied the joint task offloading scheduling and transmit power allocation problem in a single-user resource-limited MEC system. Independent tasks are not allowed to execute concurrently on a MEC server and have to be processed in a FCFS manner. This problem is formulated as a mixed integer nonlinear program. A heuristic based on the two-state flowshop scheduling theory and convex optimization techniques is developed to solve it. Kuang et al. [4] addressed the partial offloading scheduling for independent tasks and power allocation problem in a single-user resource-limited MEC system, and formulated it as a non-convex mixed-integer optimization problem. A family of heuristics are developed to solve this problem by employing Lagrangian dual decomposition and convex optimization techniques. Qin et al. [11] investigated a resource allocation problem for maximizing the power-constrained available processing capacity with unpredictable tasks in both single-user and multi-user MEC systems. The authors also provided a theoretical approach to derive the optimal solution to the scheduling problem in a single-user MEC system. Sahni et al. [12] proposed a multi-stage greedy adjustment algorithm to tackle a data-aware task scheduling problem in a resource-limited MEC system for reducing latency. In this algorithm, both task placement and network flows adjustment are taken into consideration to avoid data transferring congestion. Xing et al. [13] studied a joint task assignment and resource allocation problem in a D2D-Enabled MEC system in which multiple mobile devices serve as a MEC server. This problem was solved by a heuristic based on the convex optimization and constraint relaxation. Different from aforementioned methods that scheduling multiple independent tasks, Zhang et al. [14] proposed a heuristic for scheduling dependent tasks among which the relations are characterized by a concept of directed call graph. The scheduling algorithms discussed above are security-unaware, while we schedule security-critical tasks in a single-user resource-limited MEC system in this paper.

It is worth mentioning the scheduling algorithm in [7], which was developed for scheduling security-critical tasks of workflow applications in a single-user MEC system consisting of resource-sufficient servers. In this work, the relations among the tasks to be scheduled are represented by a directed acyclic graphs (DAG). Tasks not associated with any DAG edges are assumed to be independent. Since this work assumes sufficient resources on MEC servers, independent tasks can be processed in different virtual machines on a MEC server concurrently. Based on a multi-level security model, the tackled problem was formulated as a mixed integer linear programming problem and solved by a genetic algorithm (GA). Different from this work, we schedule independent security-critical tasks in a single-user resource-limited MEC system and propose novel slow-movement PSOs to solve it. Similar to the scheduling models in [3,4], concurrent task processing is not allowed due to the resource constraint. For this reason, the approach in [7] is not able to cope with our considered independent security-critical task scheduling problem in a resource-limited MEC system. Table 1 provides a summary of aforementioned scheduling algorithms for single-user MEC systems.

Table 1
Summary of existing related work for task scheduling in a single-user MEC system.

Reference	Environment	Application Model	Security-Critical	Methodology
Mao et al. [3]	Resource-limited MEC system	Independent Tasks	No	Heuristic
Kuang et al. [4]	Resource-limited MEC system	Independent Tasks	No	Heuristic
Qin et al. [11]	Resource-sufficient MEC system	Independent Tasks	No	Theoretical Analysis
Sahni et al. [12]	Resource-limited MEC system	Workflow	No	Heuristic
Xing et al. [13]	D2D-Enabled MEC system	Independent Tasks	No	Heuristic
Zhang et al. [14]	Resource-sufficient MEC system	Workflow	No	Heuristic
Huang et al. [7]	Resource-sufficient MEC system	Workflow	Yes	Genetic Algorithm
This paper	Resource-limited MEC system	Independent Tasks	Yes	Slow-movement PSO

Table 2
Summary of existing work using PSOs to solve optimization problems in MEC environments.

Reference	Tackled problem	Innovations
Xie et al. [15]	Schedule tasks of a workflow in a cloud-edge system	A non-linear inertia weight updating method, a selection policy and a mutation operator are used.
Jiang et al. [16]	Minimize the total energy consumption by optimizing the positions of ground vehicles and unmanned aerial vehicles in a hybrid MEC system	A deep learning method is integrated.
Yadav et al. [17]	Allocate service in a MEC system	Crossover and mutation operators are incorporated.
Mseddi et al. [18]	Jointly optimize the container placement and the flexible service provisioning in a MEC system	A particle repair mechanism is employed.
This paper	Schedule independent security-critical tasks in a resource-limited MEC system	A position-based mapping scheme and a slow-movement particle updating strategy are proposed, and LTF and BTF are used to initialize solutions.

We now discuss existing approaches employing PSOs to solve optimization problems in MEC environments. Xie et al. [15] developed a directional and non-local-convergent particle swarm optimization (DNCPSO) algorithm to schedule workflow tasks in a cloud-edge system. This algorithm includes a non-linear inertia weight updating method, a selection policy, and a mutation operator to achieve the joint optimization of makespan and cost. Jiang et al. [16] considered a hybrid MEC system consisting of ground stations, ground vehicles, and unmanned aerial vehicle that are all deployed with MEC servers to enable offloading of computation tasks from users' devices. To minimize the total energy consumption of all devices, a deep learning-based PSO approach was proposed to dynamically determine the optimal positions of ground vehicles and unmanned aerial vehicles. Yadav et al. [17] employed a GA-based PSO algorithm to optimally allocate services in a MEC system. This algorithm incorporates two well-known GA operators, crossover and mutation, into PSO framework to avoid being trapped into local optima. Mseddi et al. [18] presented a PSO-based approach to solve the joint optimization problem of container placement and flexible service provisioning in a MEC system. For a particle violating any constraint(s) in the optimization model, a repair mechanism is utilized to repair this particle by changing its position in the particle space. However, the PSOs in above-mentioned researches do not take into account the sequence of tasks/services/resources/containers on MEC servers. A distinguishing property in our considered problem is that task sequence is regarded as one of the decision variables. Therefore, in our SPSO1 and SPSO2 algorithms, we develop a position-based mapping operator to convert each particle into a valid task sequence representing the scheduling solution. We also develop a slow-movement particle updating strategy to improve the solution exploration capability of the proposed algorithms. Table 2 summarizes the application scenarios as well as the kernel innovation of existing PSO-based algorithms developed for MEC systems.

3. Problem description

We consider a resource-limited MEC system consisting of a mobile device and multiple resource-limited MEC servers, each of which is connected to a base station. Base stations are connected by a core cellular network implemented by software defined network (SDN) technology, which is capable of managing a network in a logical center called SDN controller. The SDN controller installs forwarding rules and routing tables to completely control and manage this SDN-based cellular network. The mobile device can offload computation tasks to all MEC servers [43]. On each resource-limited MEC server, there is a dedicated virtual machine (or an application container) with a single CPU core to process the offloaded tasks in a FCFS manner [3,4]. Fig. 1 provides an overview of this resource-limited MEC system.

The mobile device has n independent security-critical computing tasks to be offloaded to the m MEC servers. Following [3,4], we assume that the mobile device has one single antenna and has to offload only one task each time. We use a task permutation $\zeta = (\zeta_1, \zeta_2, \ldots, \zeta_n)$ to define the order that the mobile device follows to offload tasks. Each task ζ_i can be described by a tuple $\langle d_i, c_i \rangle$, where d_i represents the amount of input data (in bits), and c_i indicates the average computation workload (in CPU cycles/bit). The values of d_i and c_i can be evaluated and obtained by measurement tools [3,43]. The total computation workload w_i can be calculated by

$$w_i = c_i \times d_i, \ \ i = 1, 2, \ldots, n. \tag{1}$$

If a task ζ_i is assigned to a MEC server s_j ($j = 1, 2, \ldots, m$), it cannot be started execution until all its data have been transmitted to the server. However, due to the task's security-critical feature, it is necessary to encrypt its input data and transfer encrypted data via the network. As mentioned previously, users may trust MEC servers in different levels. Generally, users trust those usually-used MEC servers whose providers have good reputations, rather than the unusually-used ones whose providers have no or even bad reputations. Similar to [7,44], we define a multi-level security policy set $\{p_1, p_2, \ldots, p_k\}$, in which each security policy p_l is associated with a security level l, an encryption algorithm, its encryption computation workload per bit α_l (in CPU cycles/bit), and its energy consumption per bit γ_l (in mJ/bit), as well as the decryption computation workload per bit β_l (in CPU cycles/bit). Obviously, users select low-/high-level security policy

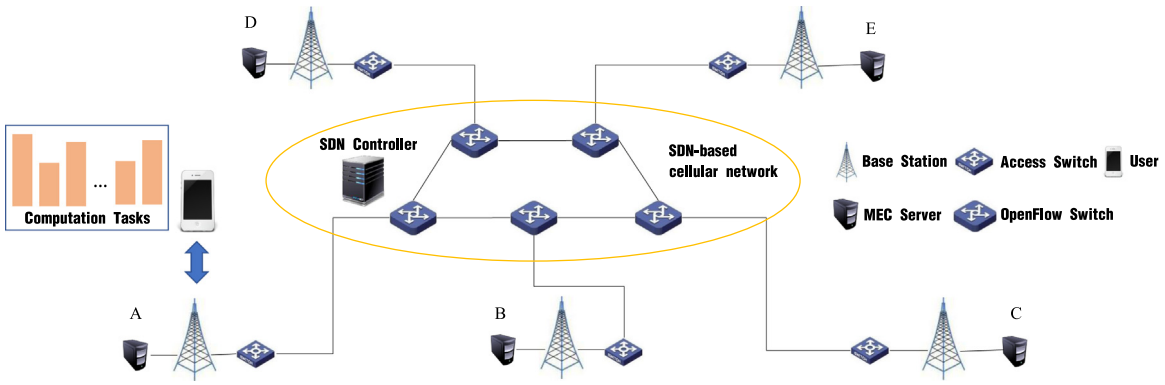


Fig. 1. The overview of the resource-limited MEC system.

and use the corresponding encryption algorithm for data encryption when tasks are assigned to trusted/untrusted MEC servers. We accordingly define a variable h_{jl} to denote the relationship between MEC servers and security policies: $h_{jl} = 1$ indicates users employ the security policy p_l to protect data when tasks are assigned to MEC server s_j, and $h_{jl} = 0$ otherwise. We further define a decision variable x_{ij} to indicate the assignment between tasks and MEC servers: $x_{ij} = 1$ means the task ζ_i is assigned to the MEC server s_j, and $x_{ij} = 0$ otherwise. In other words, the index of the assigned MEC server is $y = \{j | x_{ij} = 1,\ j = 1, 2, \ldots, m\}$ and its corresponding security policy is $z = \{l | h_{yl} = 1,\ l = 1, 2, \ldots, k\}$. Based on y and z, for a specific task ζ_i, the total computation workload of encryption and decryption (represented as w_i^E and w_i^D) can be calculated by

$$w_i^E = \alpha_z d_i,\ i = 1, 2, \ldots, n, \tag{2}$$

$$w_i^D = \beta_z d_i,\ i = 1, 2, \ldots, n. \tag{3}$$

Data encryption is performed on the mobile device, and data decryption and task execution are completed on the assigned MEC server. Let f_0 and f_j be the CPU operating frequency of the mobile device and the MEC server s_j, respectively. The durations of encryption data, decryption data and task processing for task ζ_i, denoted by D_i^E, D_i^D, and D_i^P, can be calculated by

$$D_i^E = \frac{w_i^E}{f_0},\ i = 1, 2, \ldots, n, \tag{4}$$

$$D_i^D = \frac{w_i^D}{f_y},\ i = 1, 2, \ldots, n, \tag{5}$$

$$D_i^P = \frac{w_i}{f_y},\ i = 1, 2, \ldots, n, \tag{6}$$

Also, its completion time of data encryption C_i^E is defined by

$$C_i^E = \begin{cases} 0, & \text{if } i = 0. \\ C_{i-1}^E + D_i^E, & \text{if } i = 1, 2, \ldots, n. \end{cases} \tag{7}$$

When data encryption has been finished, the mobile device needs to transmit the encrypted data to the assigned MEC server. According to [45], for most encryption algorithms, the amount of encrypted data is similar to that of original data. Thus, we regard that the amount of encrypted data is still d_i. Similar to [3,4], the transmission rate R that the mobile device transmits the encrypted data to the base station coupled with the direct-connected MEC server (i.e., MEC server A in Fig. 1) via radio access networks can be obtained by

$$R = b \log_2 \left(1 + \frac{g_0 (u_0/u)^\theta p}{b N_0}\right), \tag{8}$$

where b is the channel bandwidth, g_0 is the path loss constant, u_0 is the reference distance, u is the distance between the mobile device and the base station, θ is path loss exponent, p is the transmission power of the mobile device, N_0 is noise power spectral density. Due to SDN-based technology used, the transmission rate between any two MEC servers can be controlled and thus considered as a constant [43]. Without the loss of generality, we assume that the direct-connected MEC server of the mobile device is s_1. As a result, R_{1j} ($j = 2, 3, \ldots, m$) is a constant, and we define $R_{11} = \infty$ for uniform formulation. The duration of the data transmission D_i^T is given by

$$D_i^T = \frac{d_i}{R} + \frac{d_i}{R_{1y}},\ i = 1, 2, \ldots, n, \tag{9}$$

in which the first term is the transmission duration from the mobile device to the directly connected MEC server s_1 and the second term represents the transmission duration from the directly connected MEC server to the assigned MEC server. Note that if the assigned MEC server is the directly connected one, the value of the second term is 0 owing to $R_{11} = \infty$. A task's data can be transmitted if its data encryption has been finished and the data transmission of its predecessor task in ζ has been completed. Accordingly, the completion time of data transmission C_i^T can be obtained by

$$C_i^T = \begin{cases} 0, & \text{if } i = 0. \\ \max\{C_{i-1}^T, C_i^E\} + D_i^T, & \text{if } i = 1, 2, \ldots, n. \end{cases} \tag{10}$$

Once data transmission has been completed, the assigned MEC server decrypts the received encrypted data and process the task. Let φ be a subsequence of ζ and φ includes all the tasks assigned to the same MEC server as ζ_i does (i.e., the MEC server s_y). Assume that ζ_i is the tth task in φ. The completion time of ζ_i (denoted by C_i^P) is equivalent to that of φ_t (denoted by $C_t^P(\varphi)$), i.e.,

$$C_i^P = C_t^P(\varphi) = \begin{cases} 0, & \text{if } t = 0. \\ \max\{C_{t-1}^P(\varphi), C_i^T\} & \\ \quad + D_i^D + D_i^P, & \text{otherwise.} \end{cases} \tag{11}$$

where $C_{t-1}^P(\varphi)$ is the completion time of φ_t's predecessor task in φ. The completion time of all tasks is determined as

$$C_{\max} = \max\{C_i^P\},\ i = 1, 2, \ldots, n. \tag{12}$$

The energy consumption of the mobile device includes the energy consumed for data encryption and the energy consumed for transmitting the encrypted data, and can be determined as

$$E_i = \gamma_z d_i + p \frac{d_i}{R}. \tag{13}$$

Note that the energy consumption is independent on the order of tasks in ζ, but relies on decision variable x_{ij}. The total energy for offloading all n tasks is given by

$$E = \sum_{i=1}^{n} E_i. \tag{14}$$

As mentioned previously, the mobile device offloads tasks to MEC servers for reducing latency and saving energy. Following [3,4], we define the scheduling objective as the weighted sum of makespan and energy consumption. We summarize below our considered security-critical task scheduling problem:

$$\text{minimize} \quad obj = C_{\max} + \eta E \tag{15}$$

$$\text{subject to} \quad \sum_{j=1}^{m} x_{ij} = 1, \ i = 1, 2, \ldots, n \tag{16}$$

$$\sum_{l=1}^{k} h_{jl} = 1, \ j = 1, 2, \ldots, m \tag{17}$$

Eq. (15) is the objective function, where η is the weighting factor that can be used for the tradeoff between the makespan and the energy consumption of the mobile device. Eq. (16) ensures that a task only can be assigned to a MEC server. Eq. (17) imposes a constraint that each MEC server uses a specified security policy. The optimization problem in Eqs. (15)–(17) is NP-hard, which can be justified by the following theorem.

Theorem 1. *The considered scheduling problem is NP-hard.*

Proof. We prove this theorem by reducing the well-known parallel machine scheduling problem (PMSP), which has been proved to be NP-hard [46,47], to the considered problem. In PMSP, there are n different jobs to be assigned to m identical machines, with the objective of makespan minimization. PMSP can be reduced to our considered problem by the following four assumptions (1) all the MEC servers have the same CPU operating frequency; (2) the user trust all the MEC servers and no data encryption and decryption are required; (3) the transmission rate R and R_{1j} ($j = 2, 3, \ldots, m$) are considered to be infinity, and the duration of data transmission can be regarded as 0; (4) the weighting factor η is set to be 0. Since PMSP is NP-hard, our considered problem is also NP-hard. ■

4. Proposed algorithms

In this section, we describe the proposed slow-movement particle swarm optimization algorithms SPSO1 and SPSO2 for solving the considered scheduling problem. All the key operators and strategies are discussed in detail.

4.1. Representation and initialization

Each task permutation is considered as a solution in our proposed scheduling algorithms. A valid task permutation should include all scheduled n tasks without duplications. We further define a swarm Ψ consisting of all $|\Psi|$ particles. Each particle ψ^t ($t = 1, 2, \ldots, |\Psi|$) is associated with a position p^t and a velocity v^t, which can be represented as n-dimensional tuples $(p_1^t, p_2^t, \ldots, p_n^t)$ and $(v_1^t, v_2^t, \ldots, v_n^t)$, respectively. In addition, the position of the personal best particle (i.e., *pbest*) is denoted by $p^{t*} = (p_1^{t*}, p_2^{t*}, \ldots, p_n^{t*})$. The position of the global best particle tracked by the swarm (i.e., *gbest*) is denoted by $g^* = (g_1^*, g_2^*, \ldots, g_n^*)$. Like most PSO algorithms in the literature, e.g., [8–10], to avoid excessive roaming of particles outside the search space, we restrict the values of p_i^t ($i = 1, 2, \ldots, n$) and v_i^t ($i = 1, 2, \ldots, n$) within intervals $[p_{\min}, p_{\max}]$ and $[v_{\min}, v_{\max}]$, respectively.

We randomly initialize all particles in the swarm. As aforementioned, in order to evaluate particle's fitness, we need to map each particle onto a valid solution (i.e., a task permutation). For this purpose, we propose a position-based mapping operator that depends on the current best solution. Therefore, it is necessary to first generate an initial solution and use it as the initial value of the current best solution. Long task first (LTF) is a well-known sorting policy for solution initialization, as LTF is verified to be effective for solving the PMSP problem [47] that can be reduced to our considered problem (refer to Theorem 1). We employ a task's total computation workload (i.e., w_i in Eq. (1)) to evaluate its duration. LTF arranges all tasks in a non-ascending order of w_i value. In this work, however, in addition to the total computation workload, data amount also has significant impact upon the execution time of data encryption, transmission and encryption, and in turn upon the total duration time. Hence, we develop a new big data task first (BTF) sorting policy that arranges all tasks according to their data amounts d_i in a non-ascending order. The proposed SPSO1 and SPSO2 employ LTF and BTF to generate initial best solutions, respectively.

4.2. Position-based mapping method

As mentioned above, we propose a position-based mapping method to map particles onto their corresponding solutions. We use the values of particles' positions to calculate the probabilities that tasks in the current best solution are copied to the solution converted from a particle. In this manner, good schemata concealed in the current best solution can be inherited by the mapped solution, for the purpose of ensuring its high quality. To summarize, during the evolutionary procedure of the proposed algorithms, solutions are explored in a probabilistic manner for enabling the inheritance of good schemata.

We use ζ^* to denote the current best solution. For each particle ψ^t ($t = 1, 2, \ldots, |\Psi|$), the proposed mapping method starts with an empty corresponding solution ζ^t and makes it a complete solution by a two-stage procedure. At the first stage, each task in ζ^* is appended into ζ^t with a probability $\rho = e^{-p_i^t}$. Accordingly, $\rho \in (0, 1)$ should be true as long as $p_i^t > 0$. In order to achieve this target, we restrict the minimal value of position $p_{\min} > 0$. We then eliminate all those appended tasks from a copy of the current best solution (denoted by $\zeta^{\text{copy}*}$) so as to avoid the destruction on ζ^*, which is used to produce corresponding solution for each particle. In the second stage, each remaining task in $\zeta^{\text{copy}*}$ is inserted into ζ^t in a uniformly distributed manner, i.e., each task in $\zeta^{\text{copy}*}$ is located at each position with the probability $\frac{1}{|\zeta^t|}$. Consequently, ζ^t becomes a complete task permutation and is the final result of the mapping method. The complete procedure of the position-based mapping method is given in Algorithm 1, in which lines 2–6 represent the first stage, and lines 7–8 denote the second stage.

According to the preceding analysis, the inheritance of good schemata by the generated solution mainly take place at the first stage, as those appended tasks in the mapped solutions follow their relative order in the current best solution. In order to inherit good schemata more effectively, we set $0 < p_i^t \leq -\ln \frac{1}{2}$ ($i = 1, 2, \ldots, n$) for any particle ψ^t, such that more than half tasks can be appended to the mapped solution without changing their relative order. It is worth noting that, as solutions corresponding to all particles in the swarm are generated based on the current best solution, if all mapped solutions are identical to the current best solution, our proposed algorithms would fail to explore new solutions For this reason, it is necessary to analyze whether this situation would happen. We have the following theorem.

Theorem 2. *For a particle ψ^t (with position $p_i^t \in [p_{\min}, p_{\max}]$) mapped onto its corresponding solution ζ^t by the proposed position-based mapping method, by defining the probability of ζ^t identical to ζ^* as $\rho(\zeta^t = \zeta^*)$, if $0 < p_i^t \leq -\ln \frac{1}{2}$, we have $\rho(\zeta^t = \zeta^*) \in [B_n(1 - e^{-p_{\max}})^n, B_n(e^{-p_{\min}})^n]$, where $B_n = \sum_{l=0}^{n} \frac{1}{(n-l)!}$.*

Algorithm 1: Position-Based Mapping Method

Input: The current best solution ζ^*;
Output: A complete solution ζ^t;
Generate an empty solution ζ^t;
for $i = 1$ *to* n **do**
  Create a uniformly distributed variable ε within the interval [0, 1];
  if $\varepsilon \leq e^{-p_i^t}$ **then**
    Append ζ_i^* to ζ^t ; /* ζ_i^* denotes the i-th task in ζ^* */
  end
end
Eliminate tasks appended to ζ^t from $\zeta^{\text{copy}*}$; /* $\zeta^{\text{copy}*}$ is a copy of ζ^* */
for $i = 1$ *to* $|\zeta^{copy*}|$ **do**
  Insert $\zeta_i^{\text{copy}*}$ into ζ_t at each position with the probability $\frac{1}{|\zeta_t|}$; /* $\zeta_i^{\text{copy}*}$ denotes the i-th task in $\zeta^{\text{copy}*}$ */
end
return ζ_t;

Proof. According to Algorithm 1, if l ($l = 0, 1, \ldots, n$) tasks are appended to ζ^t at the first stage, and the remaining $n - l$ tasks are inserted at their original locations at the second stage, the generated solution ζ^t must be identical to ζ^*. Let ρ_l be the probability of $\zeta^t = \zeta^*$ for this particular case. The probability of ζ^t identical to ζ^* (i.e., $\rho\left(\zeta^t = \zeta^*\right)$) should be equal to the sum of the probabilities in all cases, i.e.,

$$\rho\left(\zeta^t = \zeta^*\right) = \sum_{l=0}^{n} \rho_l. \tag{18}$$

In order to calculate ρ_l, we define a task set $\mathbb{T}_l$ consisting of all the l tasks appended into ζ^t at the first stage. Obviously, the number of possible $\mathbb{T}_l$ is C_n^l (where $l = |\mathbb{T}_l|$), i.e., the number of task combinations with the length l selected from n tasks. Let δ_l be the probability of tasks inserted at their original locations in the second stage. We have $\delta_l = 1/A_n^{n-l}$, in which A_n^{n-l} represents the total number of task permutations with the length $(n-l)$ selected from n tasks. Consequently, by defining a set $\mathbb{T}$ including all tasks, ρ_l can be calculated by

$$\rho_l = \delta_l C_n^l \exp\left\{-\sum_{\forall \zeta_i^t \in \mathbb{T}_l} p_i^t\right\} \cdot \prod_{\forall \zeta_i^t \in (\mathbb{T}-\mathbb{T}_l)} \left(1 - e^{-p_i^t}\right). \tag{19}$$

Since $0 < p_i^t \leq -\ln\frac{1}{2}$, we have $e^{-p_i^t} \geq \frac{1}{2} \geq 1 - e^{-p_i^t}$. Referring to Eq. (19), the following two equations hold.

$$\begin{aligned}
\rho_l &\leq \delta_l C_n^l \exp\left\{-\sum_{\forall \zeta_i^t \in \mathbb{T}_l} p_i^t\right\} \cdot \exp\left\{-\sum_{\forall \zeta_i^t \in (\mathbb{T}-\mathbb{T}_l)} p_i^t\right\} \\
&= \frac{l!}{n!} \cdot \frac{n!}{l!(n-l)!} \cdot \exp\left\{\sum_{i=1}^{n} p_i^t\right\} \\
&\leq \frac{1}{(n-l)!} \cdot \left(e^{-p_{\min}}\right)^n.
\end{aligned} \tag{20}$$

$$\begin{aligned}
\rho_l &\geq \delta_l C_n^l \prod_{\forall \zeta_i^t \in \mathbb{T}_l} \left(1 - e^{-p_i^t}\right) \cdot \prod_{\forall \zeta_i^t \in (\mathbb{T}-\mathbb{T}_l)} \left(1 - e^{-p_i^t}\right) \\
&= \frac{l!}{n!} \cdot \frac{n!}{l!(n-l)!} \cdot \prod_{i=1}^{n} \left(1 - e^{-p_i^t}\right) \\
&\geq \frac{1}{(n-l)!} \cdot \left(1 - e^{-p_{\max}}\right)^n.
\end{aligned} \tag{21}$$

By combining Eqs. (18), (20), and (21), we have

$$B_n \left(1 - e^{-p_{\max}}\right)^n \leq \rho\left(\zeta^t = \zeta^*\right) \leq B_n \left(e^{-p_{\min}}\right)^n. \tag{22}$$

That is, $\rho\left(\zeta^t = \zeta^*\right) \in [B_n \left(1 - e^{-p_{\max}}\right)^n, B_n \left(e^{-p_{\min}}\right)^n]$. ■

Theorem 2 indicates that the lower-bound and upper-bound of the probability that the mapped solution is identical to the current best solution not only depend on the problem size n (i.e., the number of tasks to be scheduled), but also rely on the values of $p_{\max}$ and $p_{\min}$. In fact, this probability value is small for large-size problems, and is close to 0 when the size n is sufficiently large. This conclusion can be verified by the following corollary. Before providing the corollary, we first introduce a lemma to prove the corollary.

Lemma 1. $\lim_{n\to\infty} B_n = e$.

Proof. This lemma is very well-known and its proof can be found on most advanced math books. For the sake of completeness, we provide the proof in brief. We define $f(x) = e^x$ and apply the following Taylor series expansion:

$$f(x) = f(0) + \frac{f'(0)}{1!} + \frac{f^{(2)}(0)}{2!} + \cdots + \frac{f^{(n)}(0)}{n!} + o(x^n). \tag{23}$$

Let $x = 1$, we have

$$f(1) = \sum_{l=0}^{n} \frac{1}{(n-l)!} + o(x^n) = B_n + o(x^n) = e.$$

Therefore, we can draw the conclusion. ■

Corollary 1. $\lim_{n\to\infty} \rho\left(\zeta^t = \zeta^*\right) = 0$.

Proof. According to Eq. (22), we have

$$\begin{aligned}
\lim_{n\to\infty} B_n \left(1 - e^{-p_{\max}}\right)^n &\leq \lim_{n\to\infty} \rho\left(\zeta^t = \zeta^*\right) \\
&\leq \lim_{n\to\infty} B_n \left(e^{-p_{\min}}\right)^n.
\end{aligned}$$

Due to the fact that $0 < 1 - e^{-p_{\max}} < 1$ and $0 < e^{-p_{\min}} < 1$, the following two equations hold.

$$\begin{aligned}
\lim_{n\to\infty} \rho\left(\zeta^t = \zeta^*\right) &\geq \lim_{n\to\infty} B_n \left(1 - e^{-p_{\max}}\right)^n \\
&= \lim_{n\to\infty} B_n \lim_{n\to\infty} \left(1 - e^{-p_{\max}}\right)^n \\
&= e \times 0 = 0.
\end{aligned}$$

$$\begin{aligned}
\lim_{n\to\infty} \rho\left(\zeta^t = \zeta^*\right) &\leq \lim_{n\to\infty} B_n \left(e^{-p_{\min}}\right)^n \\
&= \lim_{n\to\infty} B_n \lim_{n\to\infty} \left(e^{-p_{\min}}\right)^n \\
&= e \times 0 = 0.
\end{aligned}$$

Consequently, we have $0 \leq \lim_{n\to\infty} \rho\left(\zeta^t = \zeta^*\right) \leq 0$, and we conclude that $\lim_{n\to\infty} \rho\left(\zeta^t = \zeta^*\right) = 0$. ■

Corollary 1 shows that for large-size problems, the solutions generated by the proposed mapping method are rarely identical to the current best solution without respect to the values of particles' positions. However, for small- and medium-size problems, it is necessary to manipulate the boundary values of particles' positions by setting appropriate $p_{\min}$ and $p_{\max}$ values. As indicated by Theorem 2, in order to maintain a small value of $\rho\left(\zeta^t = \zeta^*\right)$, we need to set relatively small lower-bound and upper-bound values. On one hand, $p_{\max}$ cannot be set as a large value, since a larger $p_{\max}$ value would lead to a larger lower-bound value. By taking into account the condition $0 < p_i^t \leq -\ln\frac{1}{2}$, we set

$p_{\max} = -\ln \frac{1}{2}$. For the lower-bound value, on the other hand, $p_{\min}$ cannot be set as an extremely small value, in order to avoid an overly large upper-bound value. Thus, we set $p_{\min} = 0.05$.

4.3. Greedy heuristic

In order to calculate the objective value of a solution generated in previous mapping procedure, we propose a greedy heuristic, in which tasks of the given solution are assigned to MEC servers one by one in a greedy manner, and the objective can be eventually obtained when the assignment is finished. In this heuristic, for a task ζ_i^t in a specific solution ζ^t, the index of the assigned MEC server y can be obtained by

$$y = \left\{ j \middle| \min_{\forall \zeta_i^t \text{ assigned to } s_j} \{obj_{ij}^t\},\ j = 1, 2, \ldots, m \right\}, \tag{24}$$

where obj_{ij}^t represents the objective value by assigning the task ζ_i^t to MEC server s_j. Algorithm 2 summarizes the algorithmic flow of the proposed greedy heuristic.

Algorithm 2: Greedy Heuristic

Input: A solution ζ^t;
Output: The objective value of ζ^t;
1 **for** $i = 1$ *to* n **do**
2 Get the i-th task ζ_i^t from ζ^t;
3 Set $E = 0$; /* the total energy cost */
4 Set $obj^t = 0$; /* the objective value of ζ^t */
5 **for** $j = 1$ *to* m **do** /* the loop for calculating the objective value */
6 Calculate w_i, w_i^E, w_i^D, D_i^E, D_i^D, D_i^P, C_i^E, R, D_i^T, C_i^T, and C_i^P by Eqs. (1)–(11), respectively;
7 Calculate E_i by Eq. (13);
8 Calculate $obj_{ij}^t = C_i^P + \eta(E + E_i)$;
9 **end**
10 Determine the index of the assigned MEC server y by Eq. (24);
11 Update $E = E + E_y$ and $obj^t = obj_{iy}^t$;
12 **end**
13 **return** obj^t;

4.4. Slow-movement particle updating strategy

In PSO framework, particles move towards both the personal best particle and the global best particle by the conventional particle updating strategy [48]. During each iteration, the velocity v^t and the position p^t of a particle ψ^t are updated by Eqs. (25) and (26), respectively.

$$v_i^t = \omega v_i^t + c_1 r_1 (p_i^t - p_i^{t*}) + c_2 r_2 (p_i^t - g_i^*), \tag{25}$$

$$p_i^t = p_i^t + v_i^t. \tag{26}$$

In Eq. (25), ω represents the inertia weight, whereas c_1 and c_2 indicate the individual cognition component and the social communication component, respectively [48]. r_1 and r_2 are two uniformly distributed numbers within [0, 1]. According to these two equations, the particle's movement velocity in the ith dimension (i.e., v_i^t) mainly depends on the values of $p_i^t - p_i^{t*}$ and $p_i^t - g_i^*$. As a result, the movement velocity could have a large value, leading to a significant change in the particle position. Since the position values are used to calculate the probability of tasks in the current best solution being inherited by the mapped solutions, the significant variations of particles' positions may cause the proposed algorithms to miss certain good solutions during the evolutionary procedure.

To address the above-mentioned issue, we propose a slow-movement particle updating strategy to slow down particles' movements. In this strategy, the velocity is updated as follows:

$$v_i^t = \omega v_i^t + c_1 r_1 \cdot sgn(p_i^t, p_i^{t*})\lambda + c_2 r_2 \cdot sgn(p_i^t, g_i^*)\lambda, \tag{27}$$

$$sgn(x, y) = \begin{cases} +, & \text{if } x \le y. \\ -, & \text{if } x > y. \end{cases} \tag{28}$$

where sgn indicates the sign function and λ is a positive constant with a small value determined by experiments in Section 5.2. The position updating rule is the same as in conventional PSO. According to Eqs. (26) and (27), we can observe that if $p_i^t \le p_i^{t*}$ and $c_1 r_1 \cdot sgn(p_i^t, p_i^{t*})\lambda > 0$, the values of positions will increase. Otherwise, $c_1 r_1 \cdot sgn(p_i^t, p_i^{t*})\lambda < 0$, denotes the decrease of positions' values. A similar observation can be made for $c_2 r_2 \cdot sgn(p_i^t, g_i^*)\lambda$. Therefore, by our proposed particle updating strategy, particles also move towards both the personal best particle and the global best particle, enabling to explore good solutions. This capability is similar to that of the conventional one. However, in a distinct manner, we employ a small-value positive constant λ rather than $p_i^t - p_i^{t*}$ and $p_i^t - g_i^*$ for the purpose of maintaining a small-value movement velocity and further avoiding the substantial variation of the positions' values. Consequently, the proposed particle updating strategy not only enables the proposed algorithms to discover high-quality solutions under the guide of both personal best particles and global best particle, but also helps proposed algorithms to avoid the trap of missing good solutions by maintaining a small-value movement velocity.

4.5. Descriptions of proposed algorithms

The proposed SPSO1 and SPSO2 start with the parameter initialization procedure (line 1), in which MG represents the maximal number of generations. All particles in the initial swarm are generated randomly (line 2), and each particle is regarded as its personal best particle (line 3). SPSO1/SPSO2 employs LTF/BTF to initialize the current best solution ζ^* (line 4), of which the objective value can be calculated by the proposed greedy heuristic (line 5). All particles are evaluated by an iterative procedure described in lines 6–15, in which the global best particle is initialized. The evolutionary procedure is shown by lines 16–29, in which each particle is updated by the proposed particle updating strategy (line 18) and mapped onto the corresponding solution (line 19) that is evaluated by the proposed greedy heuristic (line 20). In this procedure, the personal best particle is updated if a better one is discovered (line 21). If a new better solution is found (line 22), the current best solution and its objective value as well as the global best particle are updated (lines 23–25), accordingly. The complete algorithmic procedure of SPSO1 and SPSO2 is given in Algorithm 3.

4.6. Complexity analysis

We analyze the computational complexities of the proposed operators and algorithms as follows. The complexities of LTF and BTF both are $\mathcal{O}(n \cdot \log n)$ since they are implemented by using the quick sort method. The mapping operator described in Algorithm 1 has a linear complexity $\mathcal{O}(n)$, which is lower than the complexity of the ROV rule $\mathcal{O}(n \cdot \log n)$. As will be justified by experimental results, the proposed algorithms achieve good computational efficiency by employing this low-complexity mapping operator. In addition, according to Algorithm 2, the complexity of the greedy heuristic used for evaluating the quality of a candidate solution is $\mathcal{O}(m \cdot n)$. To conclude, the complexities of our proposed algorithms are both $\mathcal{O}(MG \cdot |\Psi| \cdot mn^2)$, where MG and $|\Psi|$ represent the maximum number of generations and swarm size, respectively.

Algorithm 3: Slow-movement Particle Swarm Optimization Algorithms SPSO1/SPSO2

Input: A task set $\mathbb{T}$;
Output: The objective of the obtained best solution obj^*;
Initialize all parameters including MG, p_{min}, p_{max}, v_{min}, v_{max}, ω, c_1, c_2 and λ, respectively;
Generate all $|\Psi|$ particles in the swarm Ψ, randomly;
Set each particle to be its personal best particle;
Generate the initial best solution ζ^* by LTF/BTF;
/* SPSO1 uses LTF and SPSO2 uses BTF. */
Call Algorithm 2 to calculate solution ζ^*'s objective value obj^*;
for $t = 1$ *to* $|\Psi|$ **do**
  Get the i-th particle ψ^t;
  Call Algorithm 1 to map ψ^t onto its corresponding solution ζ^t;
  Call Algorithm 2 to calculate ζ^t's objective obj^t;
  if $obj^t < obj^*$ **then**
    $\zeta^* = \zeta^t$;
    $obj^* = obj^t$;
    Set ψ^t as the global best particle;
  end
end
while $g < MG$ **do**
  for $t = 1$ *to* $|\Psi|$ **do**
    Update ζ^t by the proposed particle updating strategy;
    Call Algorithm 1 to map ψ^t onto its corresponding solution ζ^t;
    Call Algorithm 2 to calculate ζ^t's objective obj^t;
    Update the personal best particle if ζ^t is better than the original personal best particle;
    if $obj^t < obj^*$ **then**
      $\zeta^* = \zeta^t$;
      $obj^* = obj^t$;
      Set ψ^t as the global best particle;
    end
  end
  $g = g + 1$;
end
return obj^*;

5. Simulation results

In this section, like most existing approaches (e.g., [3,4,7,11–14,40]), we conduct extensive simulation experiments to evaluate the performance of the proposed SPSO1 and SPSO2, in terms of both effectiveness and efficiency. We first describe the testing instances and parameter determination criteria, followed by the detailed evaluation results.

5.1. Testing instances

To thoroughly evaluate the performance of scheduling algorithms for problems of different sizes, we generate a testing instance set consisting of 5 groups with the sizes (i.e., the number of tasks n) in {10, 20, 30, 40, 50}. Each group contains 10 different testing instances. In total, there are 50 testing instances used for performance evaluation. Following [3,4], for each task, the amount of input data d_i and average computation workload c_i are uniformly distributed within $[0, 2d_{\text{avg}}]$ and $[0, 2c_{\text{avg}}]$, where $d_{\text{avg}} = 1000$ bits and $c_{\text{avg}} = 797.5$ cycles/bit, respectively.

We construct a multi-level security policy set that takes into account six policies, of which the details are provided in Table 3. In [44], six encryption algorithms and their corresponding execution durations (in ms/KB) as well as energy consumption (in mJ/KB) were introduced. In this work, we employ these encryption policies and express the energy consumption values in units of mJ/bit. In addition, as we use computation workload (in CPU cycles/bit) α_l to differentiate encryption algorithms, we define α_l values according to the observation reported in [44] that the execution duration of an encryption algorithm increases as the security level raises. The values of decryption computation workloads (in CPU cycles/bit) β_l are determined relying on the relations between the decryption durations and encryption as reported in [49–51]. For instance, referring to the IDEA encryption algorithm in [51], the durations of decryption and encryption are nearly identical. Therefore, for security policy p_4 employing the IEA encryption algorithm, we set decryption computation workload $\beta_4 = 300$, which is identical to the α_4's value. Decryption computation workloads for the other security policies are set in a similar way.

We consider a resource-limited MEC system with six MEC servers. The CPU operating frequencies (in GHz) of all MEC servers are uniformly distributed within [2, 4]. In order to use all six security policies listed in Table 3, all MEC servers randomly selects one of the six security policies in a one–one mapping manner. In [43], the transmission durations between any two MEC servers s_i and s_j $(i \neq j)$ are set as $\tau_{ij} \times D_R$, where D_R represents the transmission duration between the mobile device and the base station and τ_{ij} is uniformly distributed within [1, 1.5]. In this work, the transmission rate (R_{1j}) between the direct-connected MEC server s_1 and another one s_j $(j = 2, 3, \ldots, 6)$ is set as $\tau_j \times R$, where τ_j is uniformly distributed within the interval [0.67, 1.0]. The parameters for calculating R are set identically to those in [3,4], i.e., $b = 1$ MHz, $g_0 = -40$ dB, $u_0 = 1$ m, $u = 100$ m, $\theta = 4$ and $N_0 = -174$ dBm/Hz. For the mobile device in the MEC system, its CPU operating frequency and transmission power are set as 1 GHz and 100 mW, respectively.

5.2. Parameter determination

In this section, we determine the values of several important parameters, including the inertia weight (ω), the weights of individual cognition and social communication (c_1 and c_2), and λ for updating the movement velocity in our proposed particle updating strategy (refer to Eq. (27)). As indicated in [52], the inertia weight ω should be within the interval [0.2, 1.2]. In this work, we set $\omega \in \{0.2, 0.4, 0.6, 0.8, 1.0, 1.2\}$. Also, following existing PSO-based algorithms [9,10,48,52], c_1 and c_2 are set as identical values. Thus, we use $c_1 = c_2 = c \in \{1, 2, 3, 4, 5\}$. In addition, according to the analysis in Section 4.2, we set $p_{\min} = 0.05$ and $p_{\max} = -\ln\frac{1}{2}$. The values of v_i^t $(i = 1, 2, \ldots, n)$ are restricted within interval $[\ln\frac{1}{2}, -\ln\frac{1}{2}]$, i.e., $v_{\min} = \ln\frac{1}{2}$ and $v_{\max} = -\ln\frac{1}{2}$.

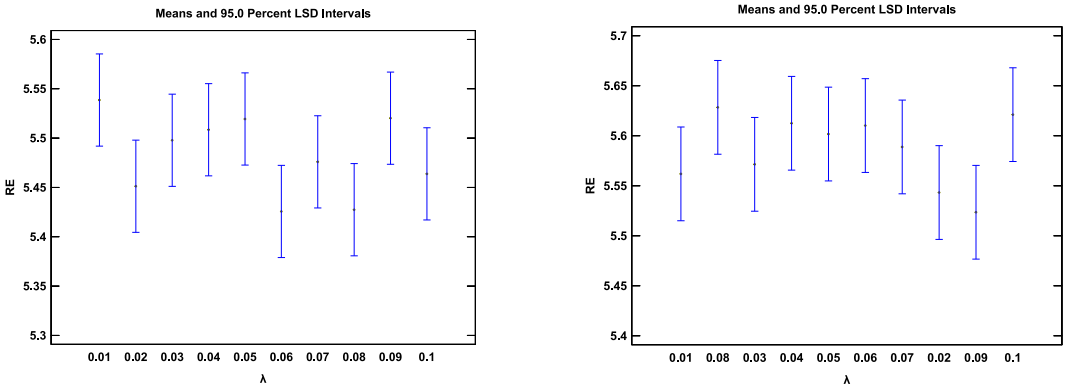
For each testing instance, we use the well-known relative error (RE) metric to evaluate algorithm effectiveness, which is defined as

$$RE = \left(\sum_{r=1}^{R} \frac{obj_r - obj^*}{obj^*}\right) \Big/ R \times 100\% \tag{29}$$

where R is the total number of replications, obj_r represents the scheduling result produced by a specific algorithm in the rth replication, and obj^* denotes the best one among the results produced by all compared algorithms during R replications. Due to the same obj^* value used in Eq. (29), RE is capable of evaluating the scheduling effectiveness of different algorithms. The lower the RE value is, the better the corresponding effectiveness is.

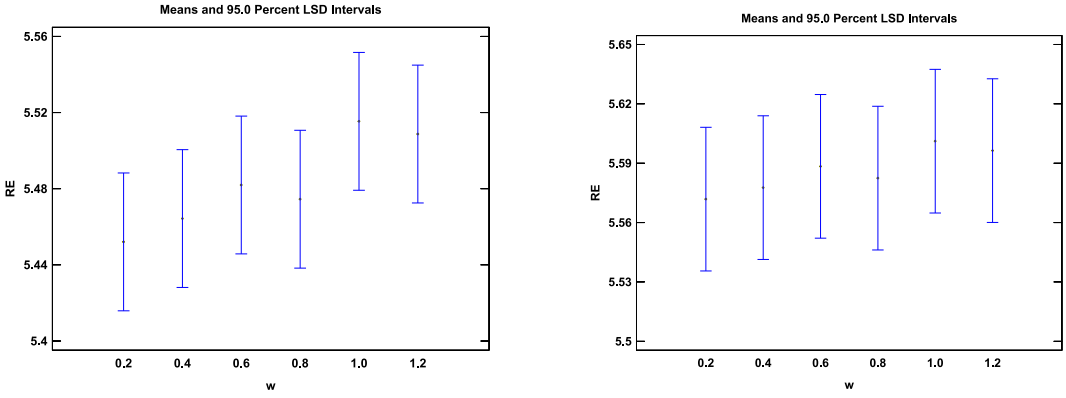
Table 3
Details about security policies.

Security policy	Security level	Encryption algorithm	α_l (CPU cycles/bit)	γ_l (10^{-4}mJ/bit)	β_l (CPU cycles/bit)
p_1	1	RC4	100	2.5296	90
p_2	2	RC5	200	5.0425	280
p_3	3	BLOWFISH	250	6.8370	350
p_4	4	IDEA	300	7.8528	300
p_5	5	SKIPJACK	350	8.7073	400
p_6	6	3DES	1050	26.3643	1700



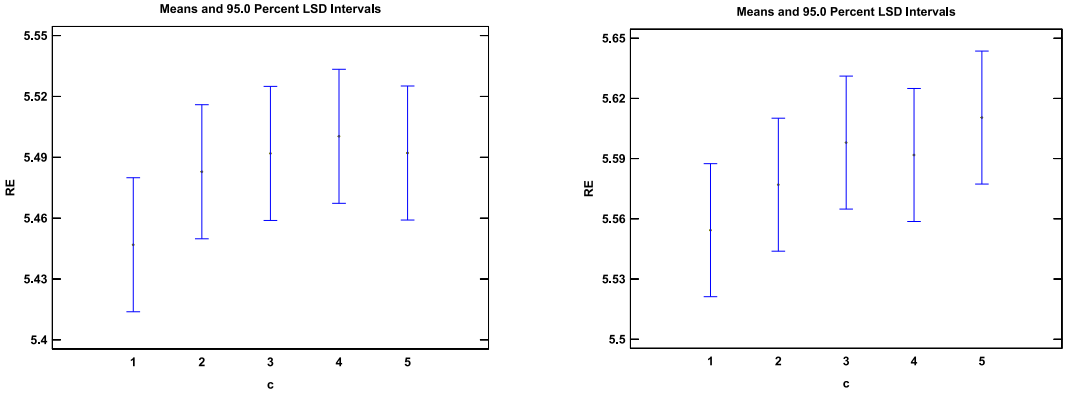
(a) SPSO1 (b) SPSO2

Fig. 2. Mean REs and 95% LSD intervals obtained by SPSO1 and SPSO2 with different λ values.



(a) SPSO1 (b) SPSO2

Fig. 3. Mean REs and 95% LSD intervals obtained by SPSO1 and SPSO2 with different ω values.



(a) SPSO1 (b) SPSO2

Fig. 4. Mean REs and 95% LSD intervals obtained by SPSO1 and SPSO2 with different c values.

As mentioned in Section 4.4, λ denotes the movement step length and should be determined appropriately. On the one hand, a small λ value could cause the proposed algorithms to converge slowly. On the other hand, a large λ value leads to a high velocity that may cause the metaheuristics to miss certain good solutions during the evolutionary procedure. For this reason, we restrict $\lambda \in \{0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1\}$. The minimum and maximum values of λ are of the same magnitudes as $p_{\min}$ and $p_{\max}$ values, respectively. Thus, there are totally $6 \times 5 \times 10 = 300$ combinations for parameters ω, c, and λ. Considering that SPSO1 and SPSO2 both are metaheuristics involving randomness, each algorithm is performed on each instance with each parameter combination for 5 separate replications (i.e., $R = 5$). Accordingly, in total, each algorithm is performed $50 \times 5 \times 300 = 75000$ rounds, providing sufficient samples to perform a statistical analysis. We employ the well-known multi-factor analysis of variance (ANOVA) method to evaluate algorithm effectiveness in a statistical way. In order to fine-tune all the parameters, we run the proposed algorithms with the weighting factor value $\eta \in \{0, 0.2, 0.4, 0.6, 0.8, 1, 2, 3, 4, 5\}$. We first fix η value at 1, i.e., the makespan and the amount of energy consumption are equally weighted. Figs. 2–4 plot the mean REs and least-significant difference (LSD) intervals with 95% confidence obtained by SPSO1 and SPSO2 with regard to λ, ω, and c, respectively. Non-overlapping intervals between any two plots indicate that the observed differences in mean values are statistically significant at the specified confidence level. The results indicate that SPSO1 achieves the lowest mean RE with parameter combination $\{\lambda = 0.06, \omega = 0.2, c = 1\}$, and the best parameter combination for SPSO2 is $\{\lambda = 0.09, \omega = 0.2, c = 1\}$. This conclusion is due to the fact that with $\omega = 0.2$ and $c = 1$, both algorithms are capable of maintaining a relatively low velocity of each particle. Similar observations can be made for other η values, and similar conclusions can be drawn. Due to limited space, we do not provide the complete comparison for all η values.

5.3. Performance evaluation

We now evaluate the performance of the proposed SPSO1 and SPSO2 in terms of both effectiveness and efficiency. Considering that there is no existing algorithm for solving the considered problem, we select the conventional particle swarm optimization algorithm (CPSO) as the baseline. The parameters of CPSO are set identically as those in the original work [53]. The well-known ROV rule is employed by CPSO to map particles onto solutions. All three algorithms to be compared, i.e., SPSO1, SPSO2, and CPSO, are implemented by Java and performed on the same aforementioned computer with different η values ($\eta \in \{0, 0.2, 0.4, 0.6, 0.8, 1, 2, 3, 4, 5\}$). The RE metric in Eq. (29) is used for effectiveness evaluation and each algorithm is run for five replications ($R = 5$).

For the sake of fair comparison, we use unified experimental setup and parameter settings for all algorithms to be evaluated. The population size is set as 30 and the maximum number of generations is set as 1000. In other words, we evaluate each algorithm by using $30 \times 1000 = 30{,}000$ particles for each run. All algorithms were implemented by Java programming with Java Runtime Environment 1.8.0 on Windows 10 64-bit operating system. In addition, all algorithms were performed on a machine equipped with an eight-core CPU operating at 1.8 GHz and 8-GB memory. We first present and analyze the evaluation results for a typical case in which $\eta = 1$, indicating that the makespan and energy consumption have identical weights. We will subsequently discuss the results for the cases with other η values.

Fig. 5 provides the comparison of mean REs and LSD intervals with 95% confidence level for the three compared algorithms when $\eta = 1$. The results show that the mean REs of CPSO, SPSO1 and SPSO2 are 1.58, 0.34 and 0.71, respectively, indicating that SPSO1 and SPSO2 are both more effective than CPSO. More importantly, as there is no overlapping between their LSD intervals, we can conclude that SPSO1 and SPSO2 both outperform CPSO significantly. Fig. 6 plots the mean REs and 95% LSD confidence intervals for the three compared algorithms when η is set as other values. The comparison results justify the advantage of SPSO1 and SPSO2 over CPSO for all investigated η values. Accordingly, we can draw the final conclusion that the proposed SPSO1 and SPSO2 can achieve good effectiveness and they outperform CPSO significantly regardless of the η value.

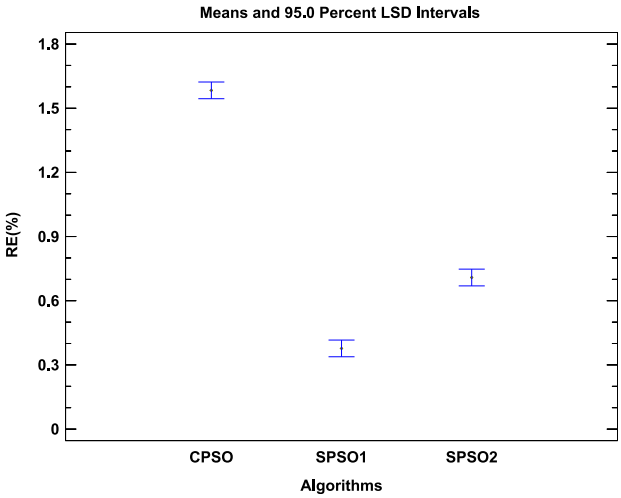


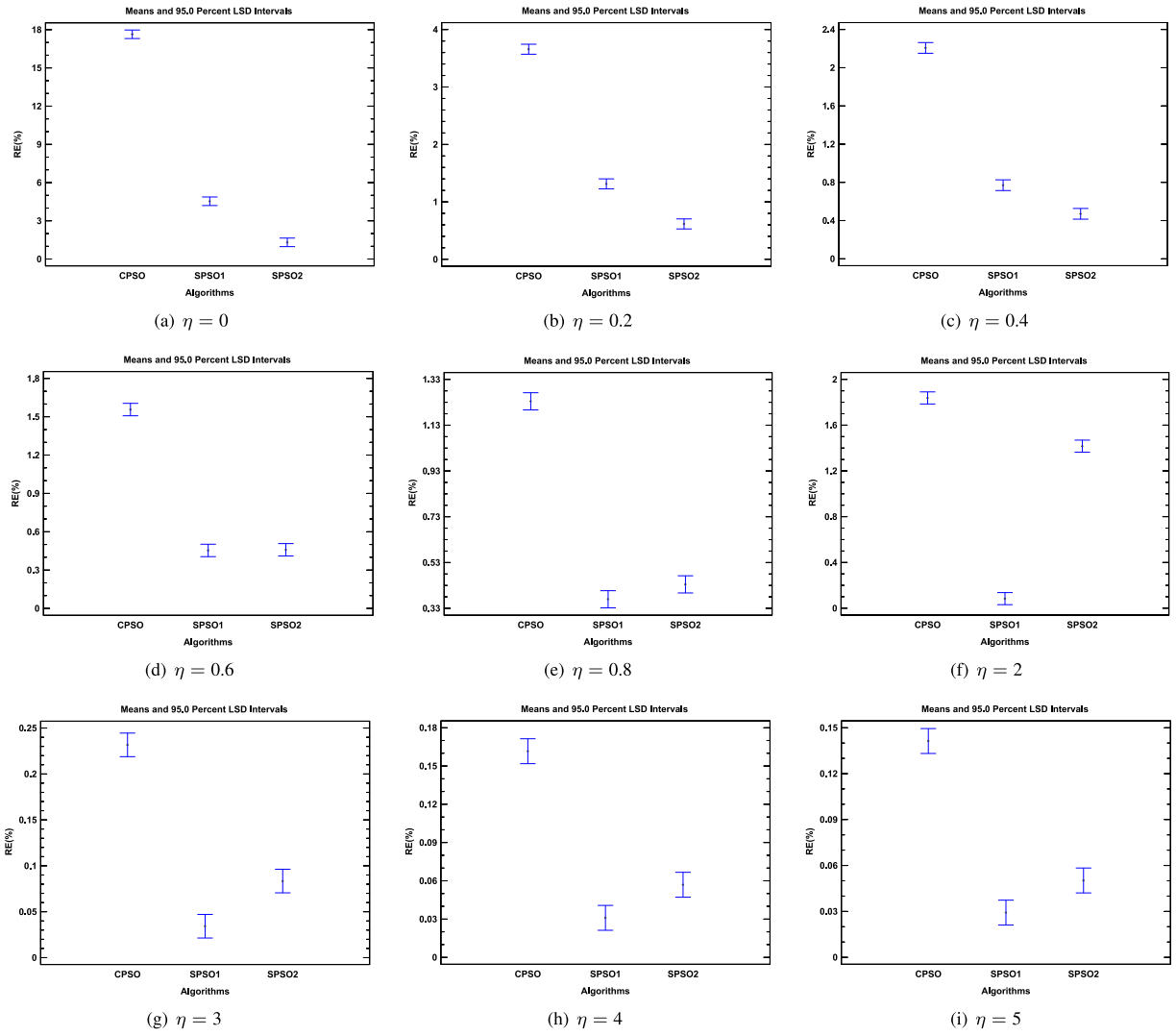
Fig. 5. Mean REs and 95% LSD intervals obtained by CPSO, SPSO1, and SPSO2 with $\eta = 1$.

Table 4
Computation times (in milliseconds) of CPSO, SPSO1, and SPSO2.

Groups	CPSO	SPSO1	SPSO2
$n = 10$	130.20	88.06	94.42
$n = 20$	416.80	208.34	228.56
$n = 30$	953.46	342.94	392.92
$n = 40$	2135.60	685.58	790.42
$n = 50$	3219.18	832.30	991.88
Average	1371.05	431.44	499.64

The high effectiveness of both SPSO1 and SPSO2 is due to the use of the position-based mapping method, the slow-movement particle updating strategy and LTF/BTF policy in these two algorithms. First, in the proposed mapping scheme, good schemata contained in the current best solution can be inherited by the generated solutions to ensure their high qualities. Second, for achieving effective inheritance, we determine appropriate maximum and minimum values for particles' positions by an analytical procedure. Based on these boundary values, the slow-movement particle updating strategy not only can discover high-quality solutions under the guide of personal best particle and global best particle, but also can avoid missing good solutions by maintaining small-value movement velocity. Finally, LTF/BTF provides SPSO1/SPSO2 with a good initial solution, contributing to the high quality of the final scheduling solution. These points will be justified by subsequent experiments for evaluating the impacts of several key operators upon the scheduling results.

Having verified algorithm effectiveness, we now compare the computational efficiency of the three algorithms to be evaluated. The computation times in Table 4 indicate that, for five groups of different sizes, CPSO, SPSO1, and SPSO2 consume on average 1371.05 ms, 431.44 ms, and 499.64 ms to solve the scheduling problem, respectively. Considering that the computation times



(a) $\eta = 0$ (b) $\eta = 0.2$ (c) $\eta = 0.4$

(d) $\eta = 0.6$ (e) $\eta = 0.8$ (f) $\eta = 2$

(g) $\eta = 3$ (h) $\eta = 4$ (i) $\eta = 5$

Fig. 6. Mean REs and 95% LSD intervals obtained by CPSO, SPSO1, and SPSO2 with different η values.

for large-size groups dominate the average value, for the sake of fair comparison, we introduce a new performance metric that is independent on group size to evaluate the efficiencies of scheduling algorithms. We define normalized efficiency (NE) achieved by each algorithm for each group of testing instances as

$$NE = \frac{d}{d_{\text{baseline}}} \tag{30}$$

where d denotes the computation time of an algorithm to be evaluated, and d_{baseline} means the computation time of the baseline algorithm. Clearly, a lower NE value indicates higher computational efficiency. As in the previous set of experiments, CPSO is selected as the baseline. When calculating the NE value, since each algorithm is performed R rounds on each testing instance, d and d_{baseline} are determined as their mean values during R replications. Fig. 7 illustrates the mean NEs for all compared algorithms on all testing instances when $\eta = 1$. The NE values achieved by SPSO1, SPSO2, and CPSO are 0.42, 0.47, and 1.0, respectively. In other words, SPSO1 and SPSO2 both are more computationally efficient than CPSO by achieving 2.38× and 2.13× speedups, respectively. The evaluation results for other η values support the same conclusion, and thus are not detailed due to limited space. The main reason for the high efficiency of SPSO1 and SPSO2 is that, both algorithms are implemented by employing the low-complexity position-based mapping method to convert a particle into a scheduling solution.

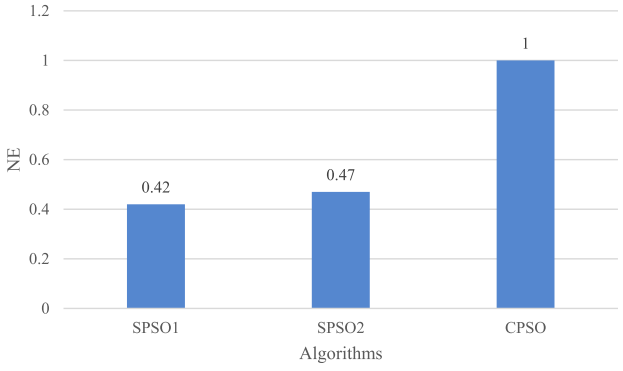


Fig. 7. Mean NEs obtained by SPSO1, SPSO2, and CPSO with $\eta = 1$.

5.4. Impacts of key operators

We perform further experiments to evaluate the impacts of these key operators upon the scheduling results. We compare SPSO1/SPSO2 with SPSO1_R/SPSO2_R, in which ROV rule is used instead of the position-based mapping method to map particles onto solutions, and other operators remain unchanged. In addition, we construct two new algorithms SPSO1_C and SPSO2_C

by using the conventional particle updating strategy to replace the proposed slow-movement particle updating strategy. Furthermore, for the purpose of investigating the impacts of LTF and BTF, we establish three new algorithms SPSO3, SPSO4, and SPSO5 by replacing LTF/BTF in SPSO1/SPSO2 with the well-known shortest task first (STF) policy, the smallest data task first (MTF) policy, and the random policy to generate the initial best solution, respectively. The parameters of these seven newly constructed algorithms are assigned the same values as in SPSO1 and SPSO2. We perform these nine algorithms with all candidate η values (i.e., $\eta \in \{0, 0.2, 0.4, 0.6, 0.8, 1, 2, 3, 4, 5\}$, and observer similar results in all cases. For the sake of conciseness, we select a typical case, in which $\eta = 1$, to show comparison results and in turn analyze the impacts of key operators. Figs. 8 and 9 present the mean REs and 95% LSD confidence levels, and the mean NEs for the nine compared algorithms, respectively. Table 5 further lists the computation times of all algorithms taken into account for evaluating the impacts of key operators. The impacts of key operators are analyzed as follows.

- Position-based mapping method: Fig. 8 shows that SPSO1/SPSO2 outperforms SPSO1_R/SPSO2_R significantly, demonstrating that the proposed position-based mapping operator is superior to the ROV rule. During the particle mapping procedure, good schemata in the current best solution can be inherited to produce a high-quality mapped solution. On the other hand, the complexity of the proposed mapping method is $\mathcal{O}(n)$, which is lower than ROV's complexity $\mathcal{O}(n \cdot \log n)$. The comparison of computation times in Table 5 and comparison of mean NEs in Fig. 9 confirm that our proposed mapping method is more efficient than ROV. In this set of experiments, SPSO2_R is considered as the baseline and all the nine algorithms built upon the proposed mapping method achieve better efficiency than the two algorithms using the ROV rule. In brief, the position-based mapping method not only improves the scheduling effectiveness, but also enhances the computational efficiency.
- Slow-movement particle updating strategy: The results in Fig. 8 also show that SPSO1/SPSO2 outperforms SPSO1_C/SPSO2_C significantly. This conclusion indicates that the proposed slow-movement particle updating strategy is more effective than the conventional one. The reason is that, the proposed particle updating strategy not only enables the proposed algorithms to discover high-quality solutions under the guide of the personal best particle and the global best particle, but also helps the proposed algorithms to avoid missing good solutions by maintaining small-value movement velocity.
- Solution initialization policy: We can further observe from Fig. 8 that, SPSO1 outperforms SPSO3 and SPSO5, and SPSO2 is more effective than SPSO4 and SPSO5. This observation indicates that LTF and BTF are more effective than STF, MTF and the random policy. As our proposed algorithms are highly dependent on the initial best solution, both LTF and BTF are capable of generating high-quality initial best solutions, and in turn make substantial contributions to the high effectiveness of SPSO1 and SPSO2.

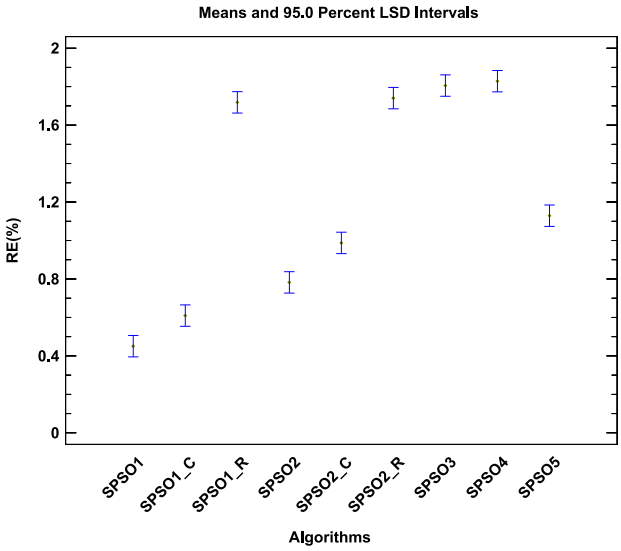


Fig. 8. Mean REs and 95% LSD intervals obtained by different algorithms with $\eta = 1$.

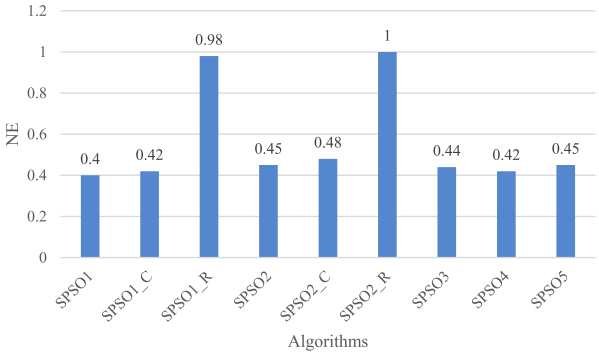


Fig. 9. Mean NEs obtained by different algorithms with $\eta = 1$.

6. Conclusions and future work

This paper presents two slow-movement particle swarm optimization algorithms for scheduling security-critical tasks in a resource-limited MEC system. The scheduling problem is formulated as a NP-hard optimization model, with the objective of minimizing the makespan and energy consumption. The proposed algorithms employ effective task ordering policies to generate the initial best solutions, based upon which the position-based mapping method can map each particle into a high-quality solution to the considered problem. A slow-movement particle updating strategy is further developed to enhance the ability of the proposed algorithms in exploring high-quality solutions. Extensive simulations are performed to justify that, compared with the conventional PSO algorithm using the ROV task sorting policy and standard particle updating strategy, the scheduling algorithms presented in this paper not only can produce better scheduling solutions but also can achieve higher computational efficiency.

It is also necessary to discuss several potential limitations when applying our proposed algorithms to cope with practical problems. SPSO1 and SPSO2 are oriented toward the MEC environment in which the mobile device has one single antenna and can only offload one task each time. As such, our proposed algorithms may not be applicable to solving the problem considering mobile devices with multiple antennas. In future, we plan to tackle this challenge by reformulating the optimization model and redesigning the solution evaluation heuristic. In addition, our algorithms assume that some input parameters are constant and known in advance, such as the duration of task execution and the transmission rate between the mobile device and the base station. For this reason, we need to rely on measurement tools or profiling methods to determine the values of these parameters. However, in practical scenarios, task durations and transmission rates may

Table 5
Computation times (in milliseconds) of different algorithms for evaluating the impacts of key operators.

Groups	SPSO1	SPSO1_C	SPSO1_R	SPSO2	SPSO2_C	SPSO2_R	SPSO3	SPSO4	SPSO5
$n = 10$	88.06	93.28	134.62	94.42	101.44	137.88	96.34	93.14	97.30
$n = 20$	208.34	216.66	424.94	228.56	242.80	434.82	224.96	216.82	230.38
$n = 30$	342.94	359.80	979.46	392.92	419.48	1000.94	378.06	359.30	390.68
$n = 40$	685.58	723.20	2218.02	790.42	861.90	2261.28	768.40	722.20	789.24
$n = 50$	832.30	877.94	3306.90	991.88	1082.00	3378.46	946.88	874.74	976.22
Average	431.44	454.18	1412.79	499.64	541.52	1442.68	482.93	453.24	496.76

vary slightly due to environmental uncertainty and unpredictable latency issues. To extend the applicability of our algorithms in the presence of such variations, we need to incorporate stochastic optimization techniques to cope with variation-aware scheduling problems.

CRediT authorship contribution statement

Yi Zhang: Conceptualization, Methodology, Software. **Yu Liu:** Software, Validation. **Junlong Zhou:** Investigation, Writing - review & editing. **Jin Sun:** Supervision, Writing - original draft, Project administration. **Keqin Li:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, D. Sabella, On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration, IEEE Commun. Surv. Tutor. 19 (3) (2017) 1657–1681.

[2] Z. Tao, Q. Xia, Z. Hao, C. Li, L. Ma, S. Yi, Q. Li, A survey of virtual machine management in edge computing, Proc. IEEE 107 (8) (2019) 1482–1499.

[3] Y. Mao, J. Zhang, K.B. Letaief, Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems, in: IEEE Wireless Communications and Networking Conference, 2017.

[4] Z. Kuang, L. Li, J. Gao, L. Zhao, A. Liu, Partial offloading scheduling and power allocation for mobile edge computing systems, IEEE Internet Things J. 6 (4) (2019) 6774–6785.

[5] Y. Zhang, D. Niyato, P. Wang, Offloading in mobile cloudlet systems with intermittent connectivity, IEEE Trans. Mob. Comput. 14 (12) (2015) 2516–2529.

[6] M. Jia, J. Cao, W. Liang, Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks, IEEE Trans. Cloud Comput. 5 (4) (2017) 725–737.

[7] B. Huang, Z. Li, P. Tang, S. Wang, J. Zhao, H. Hu, W. Li, V. Chang, Security modeling and efficient computation offloading for service workflow in mobile edge computing, Future Gener. Comput. Syst. 97 (2019) 755–774.

[8] M.K. Marichelvam, T. Prabaharan, S.Y. Xin, A discrete firefly algorithm for the multi-objective hybrid flowshop scheduling problems, IEEE Trans. Evol. Comput. 18 (2) (2014) 301–305.

[9] X. Zuo, G. Zhang, W. Tan, Self-adaptive learning PSO-based deadline constrained task scheduling for hybrid IaaS cloud, IEEE Trans. Autom. Sci. Eng. 11 (2) (2014) 564–573.

[10] M.F. Tasgetiren, Y.C. Liang, M. Sevkli, G. Gencyilmaz, A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem, European J. Oper. Res. 177 (3) (2007) 1930–1947.

[11] M. Qin, L. Chen, N. Zhao, Y. Chen, F.R. Yu, G. Wei, Power-constrained edge computing with maximum processing capacity for IoT networks, IEEE Internet Things J. 6 (3) (2019) 4330–4343.

[12] Y. Sahni, J. Cao, L. Yang, Data-aware task allocation for achieving low latency in collaborative edge computing, IEEE Internet Things J. 6 (2) (2019) 3512–3524.

[13] H. Xing, L. Liu, J. Xu, A. Nallanathan, Joint task assignment and resource allocation for D2D-enabled mobile-edge computing, IEEE Trans. Commun. 67 (6) (2019) 4193–4207.

[14] W. Zhang, Z. Zhang, S. Zeadally, H. Chao, Efficient task scheduling with stochastic delay cost in mobile edge computing, IEEE Commun. Lett. 23 (1) (2019) 4–7.

[15] Y. Xie, Y. Zhu, Y. Wang, Y. Cheng, R. Xu, A.S. Sani, D. Yuan, Y. Yang, A novel directional and non-local-convergent particle swarm optimization based workflow scheduling in cloud-edge environment, Future Gener. Comput. Syst. 97 (2019) 361–378.

[16] F. Jiang, K. Wang, L. Dong, C. Pan, W. Xu, K. Yang, Deep learning based joint resource scheduling algorithms for hybrid MEC networks, IEEE Internet Things J. (2019) http://dx.doi.org/10.1109/JIOT.2019.2954503.

[17] V. Yadav, B.V. Natesha, R.M.R. Guddeti, GA-PSO: Service allocation in fog computing environment using hybrid bio-inspired algorithm, in: TENCON 2019 - 2019 IEEE Region 10 Conference, TENCON, 2019, pp. 1280–1285.

[18] A. Mseddi, W. Jaafar, H. Elbiaze, W. Ajib, Joint container placement and task provisioning in dynamic fog computing, IEEE Internet Things J. 6 (6) (2019) 10028–10040.

[19] W. Na, S. Jang, Y. Lee, L. Park, N. Dao, S. Cho, Frequency resource allocation and interference management in mobile edge computing for an internet of things system, IEEE Internet Things J. 6 (3) (2019) 4910–4920.

[20] C. Yi, J. Cai, Z. Su, A multi-user mobile computation offloading and transmission scheduling mechanism for delay-sensitive applications, IEEE Trans. Mob. Comput. 19 (1) (2020) 29–43.

[21] C. Liu, K. Li, J. Liang, K. Li, COOPER-SCHED: A cooperative scheduling framework for mobile edge computing with expected deadline guarantee, IEEE Trans. Parallel Distrib. Syst. (2019) http://dx.doi.org/10.1109/TPDS.2019.2921761.

[22] L. Lei, H. Xu, X. Xiong, K. Zheng, W. Xiang, Joint computation offloading and multiuser scheduling using approximate dynamic programming in NB-IoT edge computing system, IEEE Internet Things J. 6 (3) (2019) 5345–5362.

[23] Z. Liang, Y. Liu, T. Lok, K. Huang, Multiuser computation offloading and downloading for edge computing with virtualization, IEEE Trans. Wireless Commun. 18 (9) (2019) 4298–4311.

[24] W. Chen, D. Wang, K. Li, Multi-user multi-task computation offloading in green mobile edge cloud computing, IEEE Trans. Serv. Comput. 12 (5) (2019) 726–738.

[25] L. Lei, H. Xu, X. Xiong, K. Zheng, W. Xiang, X. Wang, Multi-user resource control with deep reinforcement learning in IoT edge computing, IEEE Internet Things J. 6 (6) (2019) 10119–10133.

[26] J. Zhang, L. Zhou, Q. Tang, E.C. Ngai, X. Hu, H. Zhao, J. Wei, Stochastic computation offloading and trajectory scheduling for UAV-assisted mobile edge computing, IEEE Internet Things J. 6 (2) (2019) 3688–3699.

[27] T. Zhu, T. Shi, J. Li, Z. Cai, X. Zhou, Task scheduling in deadline-aware mobile edge computing systems, IEEE Internet Things J. 6 (3) (2019) 4854–4866.

[28] M. Hu, L. Zhuang, D. Wu, Y. Zhou, X. Chen, L. Xiao, Learning driven computation offloading for asymmetrically informed edge computing, IEEE Trans. Parallel Distrib. Syst. 30 (8) (2019) 1802–1815.

[29] J. Wang, J. Hu, G. Min, W. Zhan, Q. Ni, N. Georgalas, Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning, IEEE Commun. Mag. 57 (5) (2019) 64–69.

[30] D. Zhang, L. Tan, J. Ren, M.K. Awad, S. Zhang, Y. Zhang, P. Wan, Near-optimal and truthful online auction for computation offloading in green edge-computing systems, IEEE Trans. Mob. Comput. 19 (4) (2020) 880–893.

[31] M.G.R. Alam, M.M. Hassan, M.Z. Uddin, A. Almogren, G. Fortino, Autonomic computation offloading in mobile edge for IoT applications, Future Gener. Comput. Syst. 90 (2019) 149–157.

[32] J. Xu, L. Chen, S. Ren, Online learning for offloading and autoscaling in energy harvesting mobile edge computing, IEEE Trans. Cogn. Commun. Netw. 3 (3) (2017) 361–373.

[33] H. Guo, J. Liu, J. Zhang, W. Sun, N. Kato, Mobile-edge computation offloading for ultradense IoT networks, IEEE Internet Things J. 5 (6) (2018) 4977–4988.

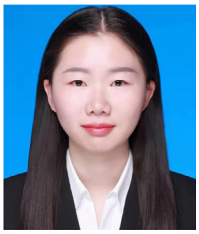
[34] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, M. Bennis, Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning, IEEE Internet Things J. 6 (3) (2019) 4005–4018.

[35] L. Chen, S. Zhou, J. Xu, Computation peer offloading for energy-constrained mobile edge computing in small-cell networks, IEEE/ACM Trans. Netw. 26 (4) (2018) 1619–1632.

[36] C. You, Y. Zeng, R. Zhang, K. Huang, Asynchronous mobile-edge computation offloading: Energy-efficient resource management, IEEE Trans. Wireless Commun. 17 (11) (2018) 7590–7605.

[37] J. Zhang, X. Hu, Z. Ning, E.C.. Ngai, L. Zhou, J. Wei, J. Cheng, B. Hu, Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks, IEEE Internet Things J. 5 (4) (2018) 2633–2645.
[38] H. Rafique, M.A. Shah, S.U. Islam, T. Maqsood, S. Khan, C. Maple, A novel bio-inspired hybrid algorithm (NBIHA) for efficient resource management in fog computing, IEEE Access 7 (2019) 115760–115773.
[39] X. Ma, S. Wang, S. Zhang, P. Yang, C. Lin, X.S. Shen, Cost-efficient resource provisioning for dynamic requests in cloud assisted mobile edge computing, IEEE Trans. Cloud Comput. (2019) http://dx.doi.org/10.1109/TCC.2019.2903240.
[40] Z. Ning, P. Dong, X. Kong, F. Xia, A cooperative partial computation offloading scheme for mobile edge computing enabled internet of things, IEEE Internet Things J. 6 (3) (2019) 4804–4814.
[41] M. Mukherjee, L. Shu, D. Wang, Survey of fog computing: Fundamental, network applications, and research challenges, IEEE Commun. Surv. Tutor. 20 (3) (2018) 1826–1857.
[42] C. Mouradian, D. Naboulsi, S. Yangui, R.H. Glitho, M.J. Morrow, P.A. Polakos, A comprehensive survey on fog computing: State-of-the-art and research challenges, IEEE Commun. Surv. Tutor. 20 (1) (2018) 416–464.
[43] H.A. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, C. Assi, Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing, IEEE J. Sel. Areas Commun. 37 (3) (2019) 668–682.
[44] W. Jiang, K. Jiang, X. Zhang, Y. Ma, Energy aware real-time scheduling policy with guaranteed security protection, in: 2014 19th Asia and South Pacific Design Automation Conference, ASP-DAC, 2014, pp. 317–322.
[45] N.R. Potlapally, S. Ravi, A. Raghunathan, N.K. Jha, A study of the energy consumption characteristics of cryptographic algorithms and security protocols, IEEE Trans. Mob. Comput. 5 (2) (2006) 128–143.
[46] M.R. Garey, D.S. Johnson, Computers and Intractability, A Guide to the Theory Of NP-Completeness, 1979.
[47] P. Brucker, Scheduling Algorithms, Springer-Verlag, 2004.
[48] F.V.D. Bergh, A. Engelbrecht, A study of particle swarm optimization particle trajectories, Inform. Sci. 176 (8) (2006) 937–971.
[49] N. Singhal, J. Raina, Comparative analysis of AES and RC4 algorithms for better utilization, Int. J. Comp. Trends Technol. (2011) 177–181.
[50] A. Trad, A.A. Bahattab, S. Ben Othman, Performance trade-offs of encryption algorithms for Wireless Sensor Networks, in: 2014 World Congress on Computer Applications and Information Systems, WCCAIS, 2014, pp. 1–6.
[51] M.I. Alam, M.R. Khan, Performance and efficiency analysis of different block cipher algorithms of symmetric key cryptography, Int. J. Adv. Res. Comput. Sci. Softw. Eng. 3 (10) (2013) 713–720.
[52] Y. Shi, R.C. Eberhart, Parameter selection in particle swarm optimization, in: International Conference on Evolutionary Programming, 1998, pp. 591–600.
[53] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995, pp. 39–43.

Yi Zhang received the B.S. and Ph.D. degrees in School of Computer Science and Engineering, Southeast University, Nanjing, China, in 2005 and 2011, respectively. From July 2009 to December 2009, he did an internship at the IBM China Research Laboratory after he was awarded the IBM Ph.D. Fellowship. In 2011, he joined the Huawei Tech. Co., 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, China. His research interests include project scheduling, workflow optimization, resource management and allocation in cloud computing and mobile computing.



Yu Liu received the B.S. degree in software engineering from Nanjing University of Science and Technology, Nanjing, China, in 2019. She is currently working toward the M.S. degree in the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. Her research interests include cloud computing and task scheduling.

Junlong Zhou received the Ph.D. degree in Computer Science 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 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 has published 60 refereed papers in his research areas, most of which are published in premium conferences and journals including 18 IEEE/ACM Transactions. He has served as publication chairs, publicity chairs, section chairs, and TPC members for numerous conferences. He has been an Associate Editor for the Journal of Circuits, Systems, and Computers, and serves as a Guest Editor for several special issues of the ACM Transactions on Cyber–Physical Systems, IET Cyber–Physical Systems: Theory & Applications, and Journal of Systems Architecture: EMBEDDED SOFTWARE DESIGN. He is a member of IEEE.

Jin Sun received the B.S. and M.S. degrees in computer science from 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. 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 cloud and edge computing, stochastic modeling and analysis, and cyber-physical systems. He is a member of IEEE.

Keqin Li is a SUNY Distinguished Professor of computer science with the State University of New York. He is also a Distinguished Professor at Hunan University, China. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber–physical systems, heterogeneous computing systems, big data computing, high-performance computing, CPU–GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, intelligent and soft computing. He has published over 690 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He currently serves or has served on the editorial boards of the IEEE Transactions on Parallel and Distributed Systems, the IEEE Transactions on Computers, the IEEE Transactions on Cloud Computing, the IEEE Transactions on Services Computing, and the IEEE Transactions on Sustainable Computing. He is a Fellow of IEEE.