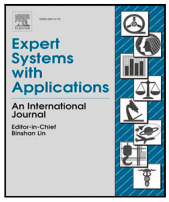
Expert Systems With Applications 250 (2024) 123896

Contents lists available at ScienceDirect

Expert Systems With Applications

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



Partial and cost-minimized computation offloading in hybrid edge and cloud systems☆

Haitao Yuan [a,*], Jing Bi [b], Ziqi Wang [b], Jinhong Yang [c], Jia Zhang [d]

[a] School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China
[b] School of Software Engineering in Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China
[c] CSSC Systems Engineering Research Institute, Beijing, China
[d] Department of Computer Science in the Lyle School of Engineering at Southern Methodist University, Dallas, TX 75205, USA

ARTICLE INFO

Keywords:
Edge computing
Cloud data centers
Computation offloading
Particle swarm optimization
Genetic algorithm

ABSTRACT

Nowadays, numerous mobile devices (MDs) provide nearly anytime and anywhere services, running on top of various computation-intensive applications. However, bearing limited battery, bandwidth, computing, and storage resources, MDs cannot completely execute all tasks of such applications in real-time. Cloud data centers (CDCs) possess enormous resources and energy, which can help execute tasks offloaded from MDs. Nonetheless, CDCs reside in remote sites, thereby leading to long transmission latency. In recent years, small base stations (SBSs) have emerged to offer close proximity, high bandwidth, and low latency services to their nearby and limited MDs. However, it becomes a new challenge to minimize the total system cost in a complex and heterogeneous architecture. To address it, this work proposes an energy-minimized partial computation offloading technique. First, a limited optimization problem of cost minimization for the system is formulated. Afterward, an improved hybrid meta-heuristic algorithm is developed, which synergistically combines a Metropolis acceptance criterion of simulated annealing and genetic operations. One uniqueness of the proposed algorithm is that it simultaneously determines task allocation among MDs, an SBS, and a CDC, the transmission power of MDs, and bandwidth allocation of wireless channels between MDs and SBS. Experiments with real-life tasks from Google data centers have shown that the proposed Genetic Simulated-annealing-based Particle swarm optimization (GSP) significantly achieves lower system cost and faster convergence speed than benchmark peers.

1. Introduction

Over recent years, mobile devices (MDs) have supported a growing number of heterogeneous mobile applications in different fields, including social networking, online games, *etc.*, which have become essential components in our daily lives (Bi, Wang et al., 2023; Cong et al., 2020). With the continuous advancements of mobile computing techniques, present MDs may offer many computation-intensive applications. For example, speech recognition and online games are increasingly developed and supplied on high-speed wireless networks and advanced MDs (Qi et al., 2023). The applications often consume a lot of CPU, memory, and battery energy. Yet, MDs only have limited wireless bandwidth, battery capacity, computing, and storage resources. Thus, it is impossible for MDs to completely execute such resource-hungry mobile applications in real-time due to their resource limits. In addition, their lifetime can be significantly shortened if they execute too many tasks because a lot of energy is consumed.

The cloud computing infrastructure, *e.g.*, Amazon EC2, and Rackspace, on the other side, possesses huge storage and computing resources, thus being able to overcome the processing and battery limits of MDs. In reality, MDs may allocate and offload some computing and memory-intensive tasks of applications to remote cloud data centers (CDCs) for execution (Bi et al., 2022a; Zhang et al., 2020; Zhu et al., 2021). In other words, tasks of applications are often executed in MDs or offloaded to CDCs in parallel. The energy consumed by MDs is significantly decreased by offloading some tasks to remote CDCs (Sohaib et al., 2023; Yuan & Zhou, 2021) In this way, MDs can guarantee the performance of their applications without increasing their energy consumption (Bozorgchenani et al., 2020). Yet, centralized CDCs are

☆ This work was supported in part by the Beijing Natural Science Foundation under Grants 4232049 and L233005, the National Natural Science Foundation of China under Grants 62173013 and 62073005, and the Fundamental Research Funds for the Central Universities, China under Grant YWF-22-L-1203.

* Corresponding author.
E-mail addresses: yuan@buaa.edu.cn (H. Yuan), bijing@bjut.edu.cn (J. Bi), ziqi_wang@emails.bjut.edu.cn (Z. Wang), yangjinhong.66@163.com (J. Yang), jiazhang@smu.edu (J. Zhang).

https://doi.org/10.1016/j.eswa.2024.123896
Received 22 December 2023; Received in revised form 27 February 2024; Accepted 1 April 2024
Available online 6 April 2024

often located in remote places far from MDs. Therefore, performance may not be guaranteed if many tasks of delay-sensitive applications (*e.g.*, mobile video conference, and mobile online games) are offloaded to remote centralized CDCs. Furthermore, in a market with a rapidly increasing number of MDs, such an MD-CDC model tends to cause enormous network traffic and congestion.

To address these drawbacks, in recent years, edge computing has emerged to enable some pervasive and flexible network and computing resources at the network edge to nearby MDs (Bi et al., 2022b; Yu et al., 2020), in the format of edge servers or so-called small base stations (SBSs). SBSs typically enable a partial computation offloading strategy (Bi et al., 2021; Saleem et al., 2020; Silva et al., 2021), to execute a portion of tasks of aforementioned latency-sensitive applications from resource-constrained MDs to nearby edge servers with richer resources (Lin et al., 2020). In this way, SBSs provide MDs with proximity, high bandwidth, and low latency access to resources in edge servers. However, resources on the edge are usually less sufficient compared with CDCs. Therefore, SBSs and remote CDCs are usually connected with fiber links with low latency. For tasks demanding significant computing power, SBSs will, in turn, allocate and offload them to centralized CDCs. Thus, edge computing triggers a new MD-SBS-CDC model representing a hybrid cloud and edge system.

However, the extra process of task offloading from MDs to SBSs, and from SBSs to CDCs unavoidably causes additional latency and energy consumption. Moreover, additional delays in waiting, communication, and processing in CDCs may be incurred. As a result, the total system cost of a hybrid cloud and edge system depends on the execution cost of all involving MDs, SBSs, and CDCs. This work considers the total system cost, which mainly includes the energy cost of running tasks in MDs, the cost of running offloaded tasks in SBS, and that of running offloaded tasks in CDC. Therefore, it has been recognized that it is critical to achieve the total system cost minimization for an edge computing system in such a complex and heterogeneous environment while guaranteeing delay limits of mobile applications (Sahni et al., 2021).

Above all, the main contributions of this work are four-fold.

1. This work designs a pervasive and fundamental architecture for studying hybrid edge and cloud systems, including MDs, SBS, and CDC. By analyzing the realistic characteristics of the different devices, MDs, SBS, and CDC are characterized by heterogeneous triple queueing models to analyze their overall cost and actual performance.
2. This work formulates and establishes the total cost minimization of hybrid edge and cloud systems as a constrained optimization problem, which considers several real-world constraints, including limits of task latency, maximum transmission power of each MD, available bandwidth of each channel between each MD and SBS, available energy of each MD and SBS, available CPU and memory resources, and task queue stability of MDs, SBS, and CDC to extend the generalization of the proposed method.
3. The cost minimization problem is solved by a novel hybrid optimization algorithm named Genetic Simulated-annealing-based Particle swarm optimization (GSP) that synergistically combines a Metropolis acceptance rule of simulated annealing (SA), and genetic operations of genetic algorithm (GA) into particle swarm optimization (PSO). It well balances the exploration and exploitation abilities and has great robustness when solving our formulated complex optimization problems.
4. Realistic data-driven experiments and comparisons with several state-of-the-art algorithms have demonstrated that our proposed GSP significantly achieves smaller costs and higher convergence speed when solving the proposed constrained optimization problem.

The remainder of this article is given here. Section 2 introduces the related work. The cost minimization problem is formulated in Section 3. Section 4 gives GSP's details. Section 5 explains the simulation results to evaluate GSP over real-life tasks from Google data centers. Section 6 summarizes this article.

2. Related work

This section discusses the related work from two aspects, including energy-efficient task offloading and resource allocation in cloud/edge systems.

2.1. Energy-efficient task offloading

Recently, the energy-efficient task offloading problem has been increasingly investigated in cloud and edge computing environments. The work in Feng et al. (2022) reveals that it is essential to minimize the total cost of a heterogeneous edge–cloud architecture due to the energy-intensive and resource-hungry services. In this case, the authors first construct an edge–cloud system. Then, the computation offloading problem of multi-function service requests is formulated and solved by a cost-minimized computation offloading algorithm with reconfiguration. This method jointly considers the resource allocation and service reconfiguration to minimize the total system cost. However, the edge–cloud architecture does not consider the mobility of users. Moreover, this work does not take into account the transmission power of users that we have considered. The work in Dai et al. (2020) adopts a deep reinforcement learning (DRL) method to design an effective policy of computation offloading and resource allocation, to decrease the energy consumed by hybrid networks. A Markov decision process is first leveraged to optimize computation offloading and resource allocation. Then, DRL is used to reduce the consumed energy. In contrast to their approach, our work considers a different scenario and adopts an improved meta-heuristic algorithm to decrease the total energy. The work in Ale et al. (2021) targets increasing the number of finished tasks and decreasing the consumed energy. They design a DRL method to determine the optimal edge server to execute offloaded tasks, and an optimal number of resources is specified to maximize the expected utility. Unlike their work, our research considers hybrid cloud and edge systems, including MDs, SBS, and CDC. The work in Wu et al. (2020) proposes a computation offloading method to reduce the energy consumed by a fog system to meet tasks' delay constraints. A mixed nonlinear integer program is formulated and handled by a method of Benders decomposition to yield the optimal offloading strategy. Unlike their work, our research considers more real-life factors and constraints in hybrid cloud and edge systems. The work in Yuan, Hu et al. (2022) jointly optimizes task partitioning, task offloading, and user associations to achieve the total cost minimization of hybrid cloud–edge systems. They mainly consider applications, which are divisible and can be split into several dependent subtasks completed in MDs, edge nodes, and CDC. Different from it, this work considers a single SBS and integrates the execution cost of offloaded tasks in SBS in the problem formulation. The work in Su et al. (2024) focuses on the energy consumption of a mobile edge computing system. Specifically, a general cloud–edge collaborative computation offloading model includes multiple mobile terminals, multiple edge servers, and a remote cloud center. Moreover, the energy consumption for edges and cloud is calculated separately by considering both transmission and computational energy consumption. Finally, an offloading strategy with near real-time decision-making is proposed to minimize the energy consumption of the system. Unlike their work, our research considers more real-life constraints, *e.g.*, limits of task latency, maximum transmission power of each MD, and available bandwidth of each channel between each MD and SBS to extend the generalization of the proposed edge–cloud model. The work in Mu et al. (2020) develops a dynamic approach for achieving computation offloading in an edge network, where many

MDs share common resources to support their delay-sensitive applications. They aim to seek the optimal trade-off between energy consumed by MDs and tasks' delay, such that given out-of-service probability is guaranteed. Unlike their work, our research aims to minimize the cost of hybrid cloud and edge systems in which the formulated problem is more realistic and complicated.

In summary, different from these aforementioned studies, we formulate a fundamental edge–cloud architecture including MDs, SBS, and CDC and design an enhanced computation offloading approach. This method optimizes resource allocation and reduces the energy consumption of the system. It focuses on delay-intensive applications in such systems. It considers many real-life constraints, including limits of task latency, maximum transmission power of each MD, available bandwidth of each channel between each MD and SBS, available energy of each MD and SBS, available CPU and memory resources, and task queue stability of MDs, SBS, and CDC to satisfy user requirements.

2.2. Resource allocation in hybrid systems

More studies have been given on resource allocation in hybrid edge/cloud systems. The work in Zaw et al. (2023) considers the allocation of both communication and computing resources in a multi-access edge computing system. The authors first formulate the allocation models of uplink, downlink, and computing resources for this system. Then, the resource allocation problem is solved by two decentralized algorithms that adopt a penalty function method to handle the device performance constraints. However, their architecture does not involve cloud computing, thus failing to establish an edge–cloud collaboration architecture. Moreover, the overall complexity of this method is relatively high, thus making it challenging to be applied in practical applications. The work in Xu et al. (2021) proposes strategies to realize beam selection and user scheduling in a cloud–edge system. A constrained process of Markov decision is formulated to minimize long-term average network delay, such that users' quality of service is guaranteed. Unlike their work, we minimize the total cost of the hybrid system. The work in Casola et al. (2020) presents a cloud–edge allocation mechanism for an industrial environment. It considers performance, cost, and security caused by on-demand service delivery and dynamically deployed applications. In contrast, we design a more fine-grained optimization model in which more realistic factors are considered. The work in Wang et al. (2017) develops a paradigm for multimedia sensing services in multimedia communication systems. High flexibility and diversity of data are provided in its application layer. Then, the allocation of wireless uplink resources is optimized accordingly to minimize the energy cost of wireless communication. Different from their work, we focus on energy-efficient partial computation offloading. The work in Sun et al. (2024) considers the resource allocation problem in a vehicular edge computing architecture. The authors construct a hierarchical framework including vehicle, edge, control, and cloud layers. Moreover, cooperative resource allocation and task offloading algorithms are proposed to coordinate the heterogeneity among tasks and servers to improve the resource utilization of the system and service satisfaction for vehicles. However, this work does not consider the fairness of resource scheduling and allocation among different vehicles. In addition, their considered constraints do not include the limit of maximum transmission power of each vehicle. The work in Ren et al. (2019) formulates an allocation problem of computation and communication resources for minimizing the weighted-sum latency of MDs. It is then transformed into a convex optimization one, and a closed-form task offloading policy is obtained accordingly. However, it aims to minimize the delay by using a collaborative cloud and edge computing manner. The work in Bi et al. (2021) designs a computation offloading approach to minimize the energy consumption of MDs and an edge server by optimizing task offloading ratios, CPU running speeds of MDs, channel bandwidth allocation, and transmission power of MDs. It ignores the cloud in its hybrid edge computing system including multiple MDs and an edge server. Different from it, this work considers a more complicated system including MDs, SBS, and CDC.

Different from the aforementioned studies, we formulate an edge–cloud collaborative system and adopt $M/M/1$, $M/M/c$ and $M/M/\infty$ queues to analyze and monitor MDs, SBS, and CDC, respectively. Our approach considers the limited resources of both users and the edge. It simultaneously determines the allocation of bandwidth resources in transmission networks, task offloading among MDs, SBS, and CDC, and MDs' data transmission power. Moreover, this method also minimizes the total cost of hybrid cloud and edge systems, including MDs, SBS, and CDC.

3. Problem formulation

We formulate the cost minimization problem based on our proposed architecture for hybrid cloud and edge systems. Core notations used in this work are listed in Table 1.

To study the cost minimization problem of hybrid cloud and edge systems, we propose an architecture, as illustrated in Fig. 1. Without losing generality and making it easier to build a model, the architecture comprises multiple MDs, an SBS, and a CDC. Such an architecture illustrates a typical partial computation offloading architecture of hybrid cloud and edge systems. Each task may run in MDs, SBS, and/or CDC. In this architecture, tasks are offloaded to SBS/CDC through wireless channels shared by multiple MDs. The SBS is connected to the CDC through high-speed fiber links. In this work, we consider computation-intensive tasks that are independent of each other and executed in MDs or SBS/CDC by using partial computation offloading. Moreover, in the system, the offloading strategy is determined in the SBS. The reasons are given as follows. First, SBS has more computational resources than MDs. Therefore, it has enough computational resources and performance to execute GSP. Second, SBS communicates with MDs directly, which reports their current states, like residual energy, to SBS. In this case, SBS can collect the performance metrics of MDs.

To analyze and monitor the MD-SBS-CDC architecture over partial computation offloading scenarios, we construct a triple queueing model. As illustrated in Fig. 1, there are N MDs in the system. Each MD supports a specific type of application and yields tasks continuously. Thus, we decide to select a model of $M/M/1$ queue (Gong et al., 2020) to evaluate the behavior of an MD. Taking into account their different roles in the partial computation offloading scenarios, the SBS is analyzed as a model of $M/M/c$ queue (Whaiduzzaman et al., 2018), and the CDC is analyzed as an $M/M/\infty$ queue model (Li et al., 2019).

To study the cost minimization problem of such an architecture, we assume that tasks from each MD i arrive in a process of Poisson (Chetlur & Dhillon, 2020), and the average task arriving rate is λ_i. p_i^o $(0 \le p_i^o \le 1)$ means the ratio of tasks offloaded from MD i. Similarly, tasks processed in MD i also conform to the Poisson process, and their average arriving rate is $(1-p_i^o)\lambda_i$. Likewise, tasks scheduled to the SBS and CDC also conform to the process of Poisson, and their average arriving rate is $p_i^o\lambda_i$.

In the next subsections, we first formulate the modeling of MDs, the SBS, and the CDC, formulate the cost and latency models for the architecture, and finally formulate a constrained optimization problem.

3.1. Modeling of MDs

This work adopts a model of $M/M/1$ queue to analyze the behavior of each MD. μ_i^M means the task execution rate of MD i, and l_i^M means its CPU utilization. According to Bi et al. (2021), the task arriving rate of MD i is $(1-p_i^o)\lambda_i$, and its task service rate is $\mu_i^M(1-l_i^M)$. T_i^M means the average time of locally running tasks in MD i, which is obtained as:

$$T_i^M = \frac{1}{\mu_i^M(1-l_i^M)-(1-p_i^o)\lambda_i} \tag{1}$$

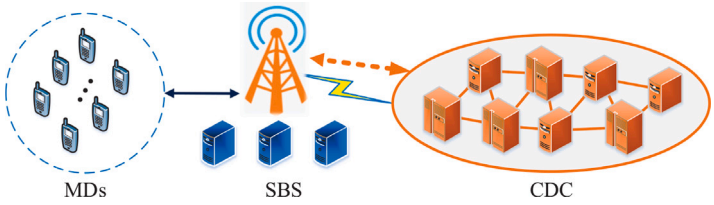


Fig. 1. Partial computation offloading architecture of hybrid cloud and edge systems.

Table 1
Main notations.

Notation	Definition
N	Number of MDs
λ_i	Mean arriving rate of tasks from MD i
p_i^o	Ratio of offloaded tasks from MD i
T_i^M	Average time for running tasks in MD i
L_{max}	Response time limit of tasks in MDs
μ_i^M	Service rate for MD i
l_i^M	CPU utilization for MD i.
θ_i	Time series of workload smoothed with a Savitzky–Golay filter
E_i^1	Energy consumption of tasks finished in MD i
ϕ_i^M	Power of running tasks in MD i
f_i^M	Running speed (CPU cycles/s) of MD i
k_i^M	Constant for the chip architecture of MD i
$\hat{E}_i^M$	Maximum energy of MD i
E_i^2	Energy consumption of transmitting tasks from MD i to SBS
T_i^t	Transmission time of delivering tasks from MD i to SBS
P_i	Transmission power of delivering tasks from MD i to SBS
γ_i^M	Proportion of bandwidth allocated to MD i between it and SBS
P_i^{max}	Maximum power of transmission for MD i
R_i	Transmission rate (bits/s) between MD i and SBS
S	Channel number between MDs and SBS
W	Channel bandwidth
h_i	Circularly symmetric and complex Gaussian constant
ω_0	White Gaussian noise power
d_i	Distance between MD i and SBS
υ	Path loss coefficient
ψ^{SBS}	Proportion of tasks offloaded to SBS
α_i	CPU cycle number for running each bit of data in MD i
$\hat{A}$	Number of available CPU cycles
g_i	Number of memories needed by executing the input data of MD i
$\hat{G}$	Maximum number of memories
λ_p^{SBS}	Arriving rate of tasks in SBS
λ_{max}^{SBS}	Maximum execution rate of task of SBS
c	Number of homogeneous servers
u_c^{SBS}	Task execution rate in a server of SBS
T_{wait}^{SBS}	Average time of waiting for tasks offloaded to SBS
T_b^{SBS}	Expected waiting time for executed results in SBS
$\hat{E}^{SBS}$	Maximum energy in SBS
k^{SBS}	Constant for the chip architecture in SBS
f^{SBS}	Working speed of SBS
u_b^C	Task transmission rate in CDC
F_M	Cost of executing tasks in MDs
r^M	Price ($/kWh) of energy in MDs
u_b^{SBS}	Transmission rate of tasks in SBS
F_{SBS} (F_C)	Cost of executing tasks offloaded in SBS (CDC)
r^{SBS} (r^C)	Cost of executing each task in SBS (CDC)
F	Total cost of the system
T_i^o	Average latency of tasks executed in SBS and CDC
T_{wait}^C	Average waiting time of tasks offloaded in CDC
u^C	Task processing rate of CDC
T_b^C	Expected waiting time of execution results in CDC

For MD i, the tasks it handles cannot exceed its processing capacity. Thus, task arriving rate must be less than or equal to its task execution rate, *i.e.*,

$$(1-p_i^o)\lambda_i < \mu_i^M(1-l_i^M),\ 1 \le i \le N \tag{2}$$

E_i^1 denotes the energy consumed by executed tasks in MD i. ϕ_i^M means the power of running tasks in MD i. For each MD i, its CPU running speed can be adjusted with the technology of dynamic voltage and frequency scaling by following the users' workload. Thus, energy consumed by MDs can be decreased by reducing their CPU running speeds. According to Wang et al. (2016), this work models the power consumption (ϕ_i^M) of the CPU in MD i as $k_i^M(f_i^M)^3$, where f_i^M is the running speed (CPU cycles/s) of MD i, and k_i^M is a coefficient depending on its chip architecture. Then, E_i^1 is obtained as:

$$E_i^1 = \phi_i^M T_i^M = k_i^M (f_i^M)^3 \frac{1}{\mu_i^M(1-l_i^M)-(1-p_i^o)\lambda_i} \tag{3}$$

For MD i, its energy consumption for running its tasks must be less than or equal to its upper limit, *i.e.*,

$$k_i^M (f_i^M)^3 \frac{1}{\mu_i^M(1-l_i^M)-(1-p_i^o)\lambda_i} \le \hat{E}_i^M,\ 1 \le i \le N \tag{4}$$

where $\hat{E}_i^M$ denotes the maximum energy of MD i.

E_i^2 and T_i^t denote the energy consumption and the time of delivering data from MD i to the SBS. θ_i is the input data size (bits) of a task of MD i. E_i^2 is given as:

$$E_i^2 = P_i T_i^t = \frac{P_i p_i^o \lambda_i \theta_i}{R_i} \tag{5}$$

where P_i is the transmission power of delivering data from MD i to the SBS, and R_i denotes the transmission rate (bits/second) of delivering data between MD i and the SBS.

For MD i, the transmission power of delivering data from it to the SBS must be less than or equal to its limit, *i.e.*,

$$0 < P_i < P_i^{max} \tag{6}$$

where P_i^{max} is the maximum transmission power of MD i.

For MD i, its allocation proportion of the bandwidth of the channel between it and the SBS must be less than or equal to 1, *i.e.*,

$$0 \le \gamma_i^M \le 1,\ 1 \le i \le N \tag{7}$$

Moreover, to ensure better utilization of bandwidth resources between MDs and SBS, the sum of bandwidth allocation proportions of all channels between all MDs and the SBS must equal 1, *i.e.*,

$$\sum_{i=1}^{N} \gamma_i^M = 1 \tag{8}$$

Let R_i denote the transmission rate (bits/second) of delivering data between MD i and the SBS. Following Bi et al. (2021),

$$R_i = S\gamma_i^M W \log\left(2\left(1+\frac{P_i(d_i)^{-\upsilon}|h_i|^2}{\omega_0}\right)\right) \tag{9}$$

where S is the channel number between MDs and the SBS, W is the total channel bandwidth, h_i is a circularly symmetric and complex Gaussian constant, ω_0 is the white Gaussian noise power, d_i is a distance between MD i to the SBS, and υ is a path loss coefficient.

In addition, the size of completed results for typical applications is much smaller than the input data (Luo et al., 2021). The assumption that the size of output is significantly smaller than the input data is reasonable similar to Khayyat et al. (2020) and Zhang et al. (2021). Thus, this work does not consider the energy of sending completed results to MDs. The total energy of MD i consists of that for tasks executed in MDs and that for transmitting data to the SBS. E_i is the energy consumption of MD i, *i.e.*,

$$E_i = (1 - p_i^o)E_i^1 + p_i^o E_i^2 \tag{10}$$

3.2. Models of SBS and CDC

ψ^{SBS} denotes the ratio of tasks offloaded to the SBS and it is obtained as:

$$\psi^{SBS} = \begin{cases} 1, & \lambda_{max}^{SBS} \geq \sum_{i=1}^{N} \lambda_i p_i^o \\ \frac{\lambda_{max}^{SBS}}{\sum_{i=1}^{N} \lambda_i p_i^o}, & \text{otherwise} \end{cases} \tag{11}$$

The average arriving rate of tasks scheduled to the SBS and CDC is $p_i^o \lambda_i$. Given ψ^{SBS}, the average arriving rate of tasks scheduled to the SBS is $p_i^o \lambda_i \psi^{SBS}$. Thus, the total input data size (bits) of all MD i's tasks offloaded to the SBS is $\left(p_i^o \lambda_i \psi^{SBS}\right) \theta_i$. α_i is the CPU cycle number for running one bit of input data for tasks from MD i. The total number of CPU cycles for running all bits of input data for tasks from MD i offloaded to the SBS is $\left(p_i^o \lambda_i \psi^{SBS}\right) \theta_i \alpha_i$, which cannot exceed the limit, *i.e.*,

$$\sum_{i=1}^{N} \left(\lambda_i p_i^o \psi^{SBS}\right) \theta_i \alpha_i \leq \hat{A} \tag{12}$$

where $\hat{A}$ denotes the number of total CPU cycles.

Similarly, the number of memories needed by tasks in the SBS cannot exceed its upper limit, *i.e.*,

$$\sum_{i=1}^{N} \theta_i \lambda_i p_i^o \psi^{SBS} g_i \leq \hat{G} \tag{13}$$

where g_i is the number of memories needed by executing each bit for MD i's tasks, and $\hat{G}$ is the total number of memories.

λ_p^{SBS} is the task arriving rate in the SBS. According to properties of different independent Poisson processes for multiple MDs, the arriving rate of all tasks from MDs in the SBS, λ_p^{SBS}, is given as:

$$\lambda_p^{SBS} = \begin{cases} \sum_{i=1}^{N} \lambda_i p_i^o, & \lambda_{max}^{SBS} \geq \sum_{i=1}^{N} \lambda_i p_i^o \\ \lambda_{max}^{SBS}, & \text{otherwise} \end{cases} \tag{14}$$

where N denotes the MD number, λ_{max}^{SBS} is SBS's maximum task execution rate.

Besides, the arriving rate of tasks in SBS cannot exceed its task execution rate and task transmission rate, *i.e.*,

$$\lambda_p^{SBS} < u_c^{SBS} \tag{15}$$

$$\lambda_p^{SBS} < u_b^{SBS} \tag{16}$$

where u_c^{SBS} and u_b^{SBS} denote the task execution rate and task transmission rate of the SBS, respectively.

Following Jia et al. (2017), we adopt a model of $M/M/c$ queue to analyze the behavior of the SBS. We assume that there are c homogeneous servers in the SBS. u^{SBS} is the task execution rate of a server in the SBS. T_{wait}^{SBS} is the average waiting time of all tasks offloaded to the SBS, *i.e.*,

$$T_{wait}^{SBS} = \frac{\Delta}{u^{SBS}c - \lambda_p^{SBS}} + \frac{1}{u^{SBS}} \tag{17}$$

$$\Delta = \frac{\left(\frac{c\rho^{SBS}}{u^{SBS}c - \lambda_p^{SBS}}\right)\left(\frac{1}{1-\rho^{SBS}}\right)}{\sum_{k=0}^{c-1} \frac{(c\rho^{SBS})^k}{k!} + \left(\frac{c\rho^{SBS}}{u^{SBS}c - \lambda_p^{SBS}}\right)\left(\frac{1}{1-\rho^{SBS}}\right)} \tag{18}$$

$$\rho^{SBS} = \frac{\lambda_p^{SBS}}{u^{SBS}c} \tag{19}$$

In addition, T_b^{SBS} is the average waiting time of results in the SBS before they are completely transmitted, *i.e.*,

$$T_b^{SBS} = \frac{1}{u_b^{SBS} - \lambda_p^{SBS}} \tag{20}$$

Finally, the amount of energy consumed by SBS cannot exceed its limit, *i.e.*,

$$\sum_{i=1}^{N} k^{SBS} (f^{SBS})^3 (T_{wait}^{SBS} + T_b^{SBS}) \leq \hat{E}^{SBS} \tag{21}$$

where $\hat{E}^{SBS}$ is the maximum energy in the SBS, k^{SBS} is a constant of the chip architecture of the SBS, and f^{SBS} is the running speed of the SBS.

Similar to Li et al. (2019), we use a model of $M/M/\infty$ queue to evaluate the behavior of the CDC. For the CDC, it has abundant computational resources. Thus, we do not consider its computational resource constraints. However, the task arriving rate cannot exceed its maximum transmission rate of tasks, *i.e.*,

$$\left(\sum_{i=1}^{N} \lambda_i p_i^o\right) - \lambda_p^{SBS} < u_b^C \tag{22}$$

where u_b^C is the transmission rate of tasks in the CDC.

3.3. Total cost

The cost of the system is closely related to its energy consumption. F_M denotes the cost of running tasks in MDs, *i.e.*,

$$F_M = r^M \sum_{i=1}^{N} E_i \tag{23}$$

where r^M is the energy price ($/kWh) of MDs.

F_{SBS} is the cost of running offloaded tasks in SBS, *i.e.*,

$$F_{SBS} = r^{SBS} \lambda_p^{SBS} \tag{24}$$

where r^{SBS} denotes the cost of running a task in the SBS.

F_C is the cost of running offloaded tasks in the CDC, *i.e.*,

$$F_C = r^C \left(\sum_{i=1}^{N} \lambda_i p_i^o - \lambda_p^{SBS}\right) \tag{25}$$

where r^C denotes the cost of running a task in the CDC.

Finally, F is the total system cost including F_M, F_{SBS} and F_C.

$$F = F_M + F_{SBS} + F_C \tag{26}$$

3.4. Latency model

Similar to Ren et al. (2019), we ignore the time for sending results back to MDs. T_i^o is the average time of executing tasks in the SBS and CDC, *i.e.*,

$$T_i^o = T_i^t + \psi^{SBS}(T_{wait}^{SBS} + T_b^{SBS}) + (1 - \psi^{SBS})(T_{wait}^C + T_b^C) \tag{27}$$

T_{wait}^C is the average waiting time for offloaded tasks in the CDC. T_{wait}^C includes the time of transmitting data from the SBS to the CDC and the running time in the CDC. Thus,

$$T_{wait}^C = T^o + \frac{1}{u^C} \tag{28}$$

where u^C is the task execution rate of the CDC.

After the CDC executes offloaded tasks, the CDC sends the finished results back to the SBS, which in turn forwards them back to MDs. T_b^C means the average waiting time for the finished results in the CDC before they are completely sent out. Then,

$$T_b^C = \frac{1}{u_b^C - \left(\sum_{i=1}^{N} \lambda_i p_i^o - \lambda_p^{SBS}\right)} \tag{29}$$

It is assumed that tasks in MDs and those offloaded to the SBS and the CDC are finished in parallel. The average time of tasks in MD i, L_i, cannot exceed its limit L_{max}, *i.e.*,

$$L_i = \max\left(T_i^M, T_i^o\right) \le L_{max} \tag{30}$$

3.5. Limited optimization problem

According to above discussion, we aim to minimize $_F$, *i.e.*,

$$\underset{p_i^o, P_i, \gamma_i^M}{\mathbf{Min}} \ \{_F\} \tag{31}$$

subject to

$$\max\left(T_i^M, T_i^o\right) \le L_{max}, \ 1 \le i \le N \tag{32}$$

$$(1 - p_i^o)\lambda_i < \mu_i^M (1 - l_i^M), \ 1 \le i \le N \tag{33}$$

$$k_i^M (f_i^M)^3 \frac{1}{\mu_i^M (1 - l_i^M) - (1 - p_i^o)\lambda_i} \le \hat{E}_i^M, \ 1 \le i \le N \tag{34}$$

$$\lambda_p^{SBS} < u_c^{SBS} \tag{35}$$

$$\lambda_p^{SBS} < u_b^{SBS} \tag{36}$$

$$\sum_{i=1}^{N} \theta_i \lambda_i p_i^o \psi^{SBS} \alpha_i \le \hat{A} \tag{37}$$

$$\sum_{i=1}^{N} \theta_i \lambda_i p_i^o \psi^{SBS} g_i \le \hat{G} \tag{38}$$

$$\sum_{i=1}^{N} k^{SBS} (f^{SBS})^3 (T_{wait}^{SBS} + T_b^{SBS}) \le \hat{E}^{SBS} \tag{39}$$

$$\left(\sum_{i=1}^{N} \lambda_i p_i^o\right) - \lambda_p^{SBS} < u_b^C \tag{40}$$

$$0 < P_i < P_i^{max} \tag{41}$$

$$0 \le p_i^o \le 1 \tag{42}$$

$$0 \le \gamma_i^M \le 1 \tag{43}$$

$$\sum_{i=1}^{N} \gamma_i^M = 1 \tag{44}$$

4. Genetic Simulated-annealing-based Particle swarm optimization (GSP)

The aforementioned offloading scenario designs the objective function and its constraints. We aim to yield the optimal values of decision variables (ratio of tasks offloaded by MDs, power of transmitting data from MDs to SBS, and proportion of bandwidth allocated to MDs in the channel between them and SBS) to yield our offloading strategy. The final offloading strategy minimizes the total energy consumption. Given the constrained optimization problem (***P***), our proposed solution algorithm is given as follows.

$_F$ is nonlinear in terms of P_i^o, P_i, and γ_i^M. Thus, it is a constrained nonlinear optimization problem. To process the aforementioned constraints, we adopt a mechanism of penalty function (Panda & Pani, 2016) to transform all constraints into the penalty. ***P*** is transformed into an unconstrained problem, *i.e.*,

$$\underset{\boldsymbol{x}}{\mathbf{Min}} \left\{\widetilde{\phi} = \overset{\infty}{\mathcal{N}} \mho + \phi\right\} \tag{45}$$

where $\boldsymbol{x}$ denotes a vector of P_i^o, P_i, and γ_i^M.

$\widetilde{\phi}$ is an augmented objective and $\overset{\infty}{\mathcal{N}}$ is a large and positive constant. $\mho$ denotes the sum of all penalties, *i.e.*,

$$\mho = \sum_{p=1}^{\mathbb{N}^{\neq}} \left(\max\{0, -g_p(\boldsymbol{x})\}\right)^{\overset{0}{\gamma}_1} + \sum_{q=1}^{\mathbb{N}^{=}} \left|h_q(\boldsymbol{x})\right|^{\overset{0}{\gamma}_2} \tag{46}$$

In (46), $\mathbb{N}^{\neq}$ and $\mathbb{N}^{=}$ are numbers of inequality and equality constraints. $\overset{0}{\gamma}_1$ and $\overset{0}{\gamma}_2$ denote two positive numbers. An inequality constraint p is converted into $g_p(\boldsymbol{x}) \ge 0$. The penalty of p is $\left(\max\{0, -g_p(\boldsymbol{x})\}\right)^{\overset{0}{\gamma}_1}$ if it is not satisfied, and it is zero otherwise. Similarly, an equality constraint q is converted into $h_q(\boldsymbol{x}) = 0$. The penalty of q is $\left|h_q(\boldsymbol{x})\right|^{\overset{0}{\gamma}_2}$ if it is not satisfied, and 0 otherwise. For instance, (36) is transformed into $u_b^{SBS} - \lambda_p^{SBS} > 0$, and the penalty is $(\max\{0, -(u_b^{SBS} - \lambda_p^{SBS})\})^{\overset{0}{\gamma}_1}$. Then, an unconstrained problem is yielded.

Some typical algorithms are available to solve this problem. They often require that optimization problems must have specific mathematical properties. For instance, they demand first and second-order derivatives. To avoid such aforementioned disadvantages, existing studies use meta-heuristic algorithms because of their significant merits, *e.g.*, fast convergence, easy implementation, strong robustness, and ability to handle nonlinearities and discontinuities. In reality, they have been adopted to address various real-life problems in different areas. Each meta-heuristic optimization algorithm may have to bear some disadvantages. For instance, although the convergence of PSO is fast, it often easily falls into local optima when solving constrained problems (Bi, Zhai et al., 2023). SA owns a rule of Metropolis acceptance, allowing directions to worsen search values (Tsai et al., 2020). Thus, SA can escape from local optima and find global ones by setting the optimal temperature cooling rate. However, SA suffers from very slow convergence (Vincent et al., 2017). To avoid such drawbacks, we design a hybrid and improved algorithm called GSP. Since GA provides genetic operations and has high individual diversity, it improves both search efficiency and accuracy.

In PSO, each particle changes its position and velocity with its learning experience and the current population (Zeng et al., 2020). $|\mathbb{X}|$ is the population size, $\mathbf{x}_i$ is a position of particle i ($i = 1, 2, \ldots, |\mathbb{X}|$), and v_i is a velocity of particle i. $\mathbb{N}^D$ denotes the dimension of each position. P_i^o is stored in the first N entries, P_i is stored in the next N entries, γ_i^M is stored in the next N entries, and $\widetilde{\phi}$ is stored in the last entry. Thus, $\mathbb{N}^D = 3N + 1$. $\check{\mathbf{x}}_i$ is a locally best position of particle i. $\hat{\mathbf{x}}$ is a globally best position. Then, $\mathbf{x}_i$ and v_i are updated as:

$$\mathbf{x}_i = \mathbf{x}_i + v_i \tag{47}$$

$$v_i = \theta_1 \cdot v_i + \check{\theta}_2 w_1 \left(\check{\mathbf{x}}_i - \mathbf{x}_i\right) + \hat{\theta}_2 w_2 \left(\hat{\mathbf{x}} - \mathbf{x}_i\right) \tag{48}$$

where w_1 and w_2 denote two random numbers in $(0, 1)$. θ_1 denotes an inertia weight. $\hat{\theta}_2$ and $\check{\theta}_2$ are social and individual acceleration constants, reflecting the influence of $\check{\mathbf{x}}_i$ and $\hat{\mathbf{x}}$.

The optimization process of PSO oscillates if $\check{\mathbf{x}}_i$ and $\hat{\mathbf{x}}$ differ significantly. Genetic operations produce superior particles for enhancing the global search ability. Therefore, $\acute{\mathbf{x}}_i$ denotes a position of a superior one for particle i. Specifically, $\acute{\mathbf{x}}_i$ combines $\check{\mathbf{x}}_i$ and $\hat{\mathbf{x}}$, *i.e.*,

$$\acute{\mathbf{x}}_i = \frac{\check{\theta}_2 w_1 \check{\mathbf{x}}_i + \hat{\theta}_2 w_2 \hat{\mathbf{x}}}{\check{\theta}_2 w_1 + \hat{\theta}_2 w_2} \tag{49}$$

Then, v_i and $\mathbf{x}_i$ are updated as:

$$v_i = \theta_1 \cdot v_i + \theta_3 w_3 \left(\acute{\mathbf{x}}_i - \mathbf{x}_i\right) \tag{50}$$

$$\mathbf{x}_i = \mathbf{x}_i + v_i \tag{51}$$

where θ_3 denotes an acceleration constant, and w_3 is a vector of random numbers in $(0, 1)$.

Furthermore, $\check{\mathbf{x}}_i$ and $\hat{\mathbf{x}}$ are encoded by using a binary encoding method. θ_5 denotes the possibility of mutation. The single-point operation of crossover is performed on $\check{\mathbf{x}}_i$ and $\hat{\mathbf{x}}$ to yield offspring $\check{\mathbf{x}}_i$. Afterward, the mutation is performed on $\check{\mathbf{x}}_i$ with a given probability θ_5, thereby reducing the possibility of falling into locally best positions. Finally, the selection is performed to select $\check{\mathbf{x}}_i$ or $\acute{\mathbf{x}}_i$.

If $\widetilde{\phi}(\check{\mathbf{x}}_i) \leq \widetilde{\phi}(\acute{\mathbf{x}}_i)$, $\check{\mathbf{x}}_i$ is selected as a superior one for particle i; otherwise, $\acute{\mathbf{x}}_i$ is selected. $\mathbf{x}_i^g$ and $\mathbf{x}_i^{g+1}$ are positions of particle i in iterations g and $g+1$. $\mathbf{x}_i^{g+1}$ is updated as:

$$v_i = \theta_1 \cdot v_i + \theta_3 \cdot w_3 \cdot \left(\acute{\mathbf{x}}_i - \mathbf{x}_i^g\right) \tag{52}$$

$$\mathbf{x}_i^{g+1} = \mathbf{x}_i^g + v_i \tag{53}$$

Moreover, a Metropolis acceptance rule of SA is adopted in GSP to enhance the diversity of the population, thereby helping it to step out of the local optima. It allows accepting the inferior solutions for the next iteration. Specifically, if $\widetilde{\phi}(\mathbf{x}_i^{g+1}) \leq \widetilde{\phi}(\mathbf{x}_i^g)$, $\mathbf{x}_i^{g+1}$ is used; otherwise, it is conditionally used if

$$e^{\left(\frac{\widetilde{\phi}\left(\mathbf{x}_i^g\right)-\widetilde{\phi}\left(\mathbf{x}_i^{g+1}\right)}{\theta_4^g}\right)} > w_4 \tag{54}$$

where w_4 and θ_4^g denote a random number in (0,1) and the temperature in iteration g.

Algorithm 1 GSP

```
1: Initialize velocities and positions of particles
2: Change φ̃ of with (45)
3: Change x̌_i and x̂
4: Initialize θ5, θ4^1, θ7, ǒ2, ô2, θ3, ô1, ĝ, ǒ1, and |𝕏|
5: g ← 1
6: while g ≤ ĝ do
7:    Conduct crossover on x̌_i and x̂ to produce offspring x̌_i
8:    Conduct mutation on x̌_i with a probability of θ5
9:    Conduct selection to select x̌_i or x́_i
10:   Change velocities and positions with (52),(53) and (54)
11:   Calculate φ̃ for each particle with (45)
12:   Change x̌_i and x̂
13:   θ4^g ← θ4^g · θ7
14:   θ1 ← (ô1 − ǒ1) · (ĝ − g)/ĝ + ǒ1
15:   g ← g + 1
16: end while
17: return x̂
```

Algorithm 1 gives GSP's details. The line 1 randomly initializes velocities and positions of particles in PSO. Line 2 changes $\widetilde{\phi}$ with (45). Line 3 changes $\check{\mathbf{x}}_i$ and $\hat{\mathbf{x}}$. θ_4^1 denotes the starting temperature, and θ_7 denotes its temperature cooling rate. Line 4 initializes θ_5, θ_4^1, θ_7, $\check{o}_2$, $\hat{o}_2$, θ_3, $\hat{o}_1$, $\check{o}_1$, $\hat{g}$, and $|\mathbb{X}|$. $\hat{g}$ is the total number of iterations. The loop of **while** terminates when $g > \hat{g}$ in Line 6. Line 7 conducts the crossover on $\check{\mathbf{x}}_i$ and $\hat{\mathbf{x}}$ to yield offspring $\check{\mathbf{x}}_i$. Line 8 conducts the mutation on $\check{\mathbf{x}}_i$ with a probability of θ_5. Line 9 conducts the selection to select $\check{\mathbf{x}}_i$ or $\acute{\mathbf{x}}_i$ for particle i. Line 10 updates velocities and positions with (52), (53) and (54). Line 11 updates $\widetilde{\phi}$ for each particle i with (45). Line 12 changes $\check{\mathbf{x}}_i$ and $\hat{\mathbf{x}}$. Line 13 reduces the temperature by θ_7. $\hat{o}_1$ and $\check{o}_1$ are upper and lower bounds of θ_1. Line 14 linearly decreases θ_1 from $\hat{o}_1$ to $\check{o}_1$. Line 17 returns the final $\hat{\mathbf{x}}$.

We give the time complexity of Algorithm 1. The computation cost is caused by the loop of **while**, which terminates when the number of iterations reaches $\hat{g}$. In Lines 7–15, GSP's time complexity in an iteration is $\mathcal{O}(|\mathbb{X}|\mathbb{N}^D)$ where $\mathbb{N}^D = 3N + 1$. Thus, the time complexity in an iteration is $\mathcal{O}(|\mathbb{X}|(3N+1))$. Consequently, GSP's overall time complexity is $\mathcal{O}(\hat{g}|\mathbb{X}|N)$.

Table 2
Parameter setting of our model.

Parameter	Value
f_i^M	$[3 \times 10^8, 3.5 \times 10^8]$ cycles/s
r^M	0.005 $/kWh
λ_i	3 MIPS
k_i^M	10^{-26}
μ_i^M	5 MIPS
l_i^M	0.3
h_i	0.98
θ_i	3.2×10^6 bits
$\hat{E}_i^M$	2 J
P_i^{max}	0.1 W
α_i	40
g_i	0.06
d_i	50 m
r^{SBS}	0.001
λ_{max}^{SBS}	8 MIPS
u^{SBS}	10 MIPS
u_b^{SBS}	15 MIPS
$\hat{A}$	14×10^9 cycles/s
$\hat{G}$	2048 GB
$\hat{E}^{SBS}$	3 J
k^{SBS}	10^{-27}
S	64
f^{SBS}	8×10^8 cycles/s
W	10 MHZ
υ	4
c	3
ω_0	1.6×10^{-11}
r^C	0.003 $
u_b^C	26 MIPS
u^C	26 MIPS

Table 3
Parameter setting of GSP.

Parameter	Value
$\check{\theta}_2$	0.5
$\hat{\theta}_2$	0.5
θ_3	1.496
$\hat{\theta}_1$	0.95
$\check{\theta}_1$	0.4
θ_5	0.01
θ_4^1	10^8
θ_7	0.95
$\hat{g}$	10^3
$\lvert\mathbb{X}\rvert$	100
$\overset{\infty}{\mathcal{N}}$	10^{10}
$\overset{0}{\gamma}_1$	2
$\overset{0}{\gamma}_2$	1

5. Performance evaluation

We evaluate GSP with realistic tasks from Google data centers for one day.[1] Input data (bits) for MDs are sampled every five minutes. GSP is implemented in MATLAB 2021, running in a server with an Intel(R) Core(TM) i7-10700F CPU with 2.90 GHz and 16-GB memory.

5.1. Parameter setting

According to Wang et al. (2016) and Yuan, Bi et al. (2022), the setting of parameters for MDs, SBS, and CDC is given in Table 2. In addition, similar to studies in Gao et al. (2019), a design approach named Taguchi is used to determine the best setting of parameters of GSP in Table 3.

[1] https://github.com/google/cluster-data (accessed on March 19, 2021).

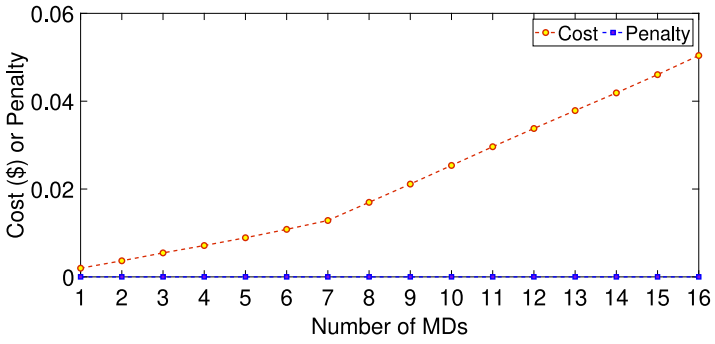


Fig. 2. Total cost and its corresponding penalty w.r.t. varying N.

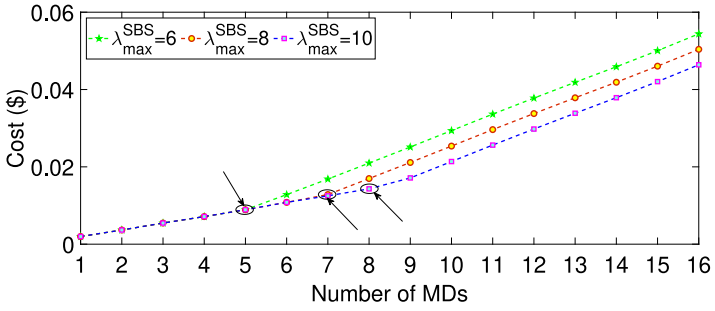


Fig. 3. Total cost w.r.t varying N and λ_{max}^{SBS}.

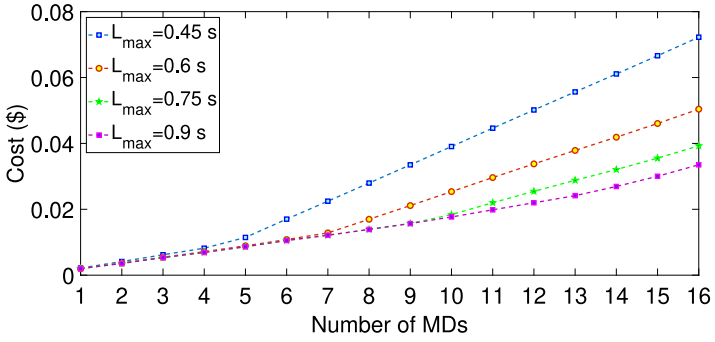


Fig. 4. Total cost w.r.t different L_{max}.

5.2. Experimental results

Fig. 2 shows the total system cost and its corresponding penalty with respect to different N. It is observed that given each number of MDs, the penalty obtained by the proposed GSP is zero. Thus, Fig. 2 demonstrates the constraints in $\boldsymbol{P}$ are strictly met.

Fig. 3 illustrates the total cost with respect to different N and λ_{max}^{SBS}. It is shown that the total cost increases as N becomes larger. In Fig. 3, there is a point of inflection for each curve, which is marked as a separate circle. After each inflection point, the total cost increases faster with a bigger slope as N increases. This is due to the limit of the task execution rate of SBS, *i.e.*, λ_{max}^{SBS}, is limited. For example, when $\lambda_{max}^{SBS} =$ 6 tasks/s, more tasks are offloaded to CDC, thereby increasing the total system cost.

Fig. 4 shows the total system cost in terms of varying L_{max}, which means the response time limit of tasks in MDs. It can be observed that when L_{max} = 0.45 s, the total system cost of the system is much larger than the cases given L_{max} = 0.6, 0.75, or 0.9 s This is because smaller L_{max} requires that more tasks are offloaded to SBS and CDC, thus increasing the total system cost.

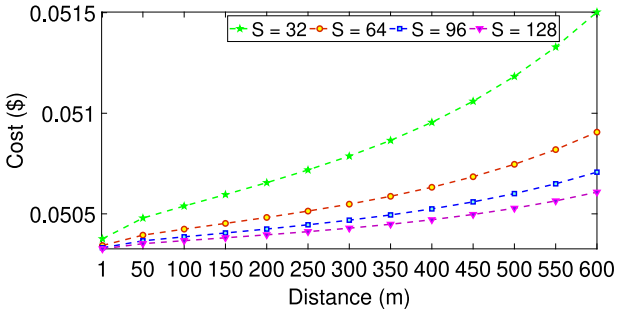


Fig. 5. Total cost w.r.t different S and d_i.

Fig. 5 shows the total system cost in terms of varying S and d_i when there are 16 MDs. It is shown that the total system cost increases as d_i becomes larger. Besides, it also shows that the total system cost reduces as S becomes larger. This is because the allocated bandwidth for each MD increases as S increases. Thus, the data transmission rate

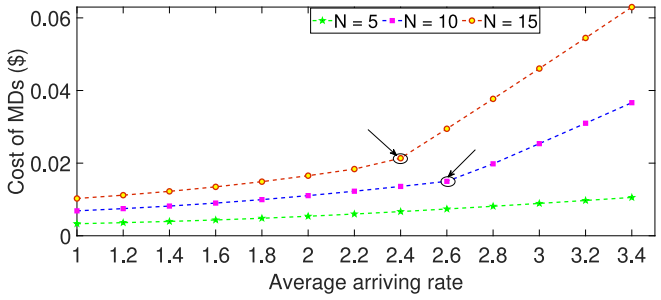


Fig. 6. Cost of MDs w.r.t varying arriving rates.

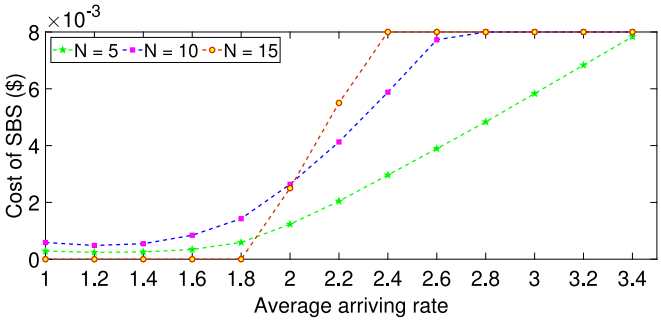


Fig. 7. Cost of SBS w.r.t varying arriving rates.

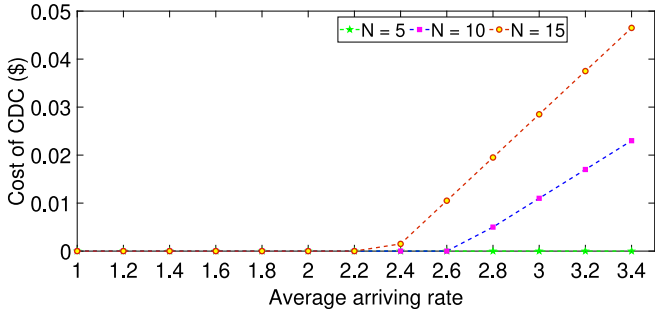


Fig. 8. Cost of CDC w.r.t varying arriving rates.

between each MD and SBS increases based on (9). Thus, following (5), the energy consumption of transmitting data from an MD to SBS decreases, which reduces the total system cost.

Figs. 6–8 illustrate the total system cost of MDs, SBS, and CDC in terms of varying arriving rates. It can be observed that the total system cost of MDs, SBS, and CDC all increase with the increase of arriving rates. When $N = 5$, the close SBS has sufficient processing capacity to execute the tasks offloaded from MDs. As illustrated in Fig. 8, the total system cost of CDC is 0 when $N = 5$. When $N = 10$, there is an inflection point when the arriving rate is 2.4 tasks/s. As illustrated in Fig. 7, when the arriving rate is greater than 2.4 tasks/s, SBS cannot execute all tasks offloaded to itself. Excessive tasks are offloaded to the CDC for further processing. Moreover, it is illustrated in Fig. 8 that the system cost of CDC increases linearly from 0 as the arriving rate reaches 2.4 tasks/s.

We further compare GSP with its typical benchmark algorithms, including Genetic Learning-based Bat Algorithm (GLBA), SA-based PSO (SAPSO), SA, and GA. The reasons are given as follows.

1. SAPSO (Yuan et al., 2017) integrates merits of SA and PSO. It inherits quick convergence of PSO and strong global search of SA. Consequently, the comparison between SAPSO and GSP proves the convergence and search accuracy of GSP.

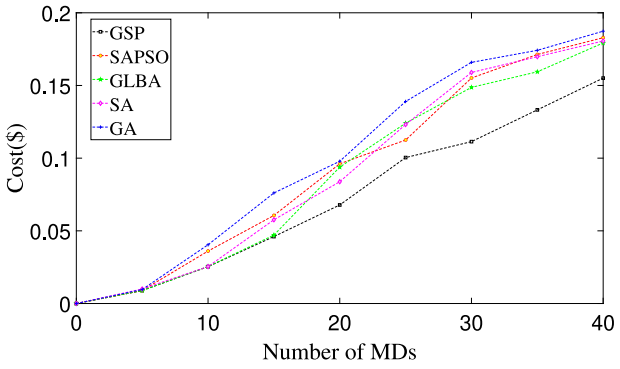


Fig. 9. Cost comparison of five algorithms w.r.t. varying numbers of MDs.

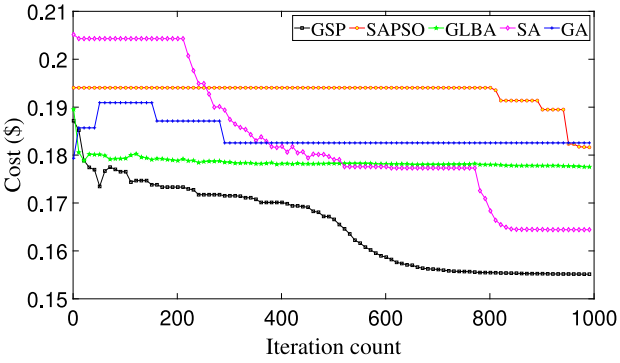


Fig. 10. Cost of GSP, SAPSO, GLBA, SA, and GA.

2. GLBA (Yue & Zhang, 2020) integrates genetic operations into BA, thus inheriting GA's high individual diversity and BA's fast convergence. Thus, their comparison proves the search accuracy and convergence of GSP.
3. SA (Lyden & Haque, 2016) obtains global optima by selecting the suitable temperature cooling rate. The comparison between SA and GSP proves the global search for GSP.
4. GA (Metawa et al., 2017) has high individual diversity and search accuracy. Thus, the comparison proves the search accuracy of GSP.

Fig. 9 illustrates the total system cost of five algorithms regarding varying numbers of MDs. GSP's cost is the smallest when N changes from 1 to 40. Besides, GSP's cost increases linearly as N increases. Moreover, it is shown that the system cost of five algorithms achieves similar results when there are five MDs. However, their cost starts to increase after the case with 15 MDs. The cost of SA increases sharply with 10 MDs, and GLBA increases after the case with 15 MDs. SAPSO also shows a similar trend after the case with five MDs. GA's cost is the largest in terms of varying N. Finally, GA, SAPSO, GLBA, and SA achieve similar results with 40 MDs, and their cost are all higher than that of GSP.

Moreover, Figs. 10 and 11 illustrate the cost and the penalty of SAPSO, GSP, GLBA, SA, and GA, respectively, when there are 40 MDs. It is illustrated that GSP has the lowest system cost (0.1553 $) after 1000 iterations. The final cost (0.1644 $) of SA is less than GA, GLBA, and SAPSO, but it is still greater than GSP. Moreover, SA obtains its best result after 820 iterations, which is much larger than GSP. SAPSO, GLBA, and BA perform poorly when there are about 40 MDs. Furthermore, it is shown in Fig. 11 that the penalty of SAPSO cannot decrease to zero after 1000 iterations. It is worth noting that the penalty of GA also fails to achieve the penalty of zero at the end, and it is 15 934 $ in Fig. 11 after 800 iterations. The result proves that GA and SAPSO

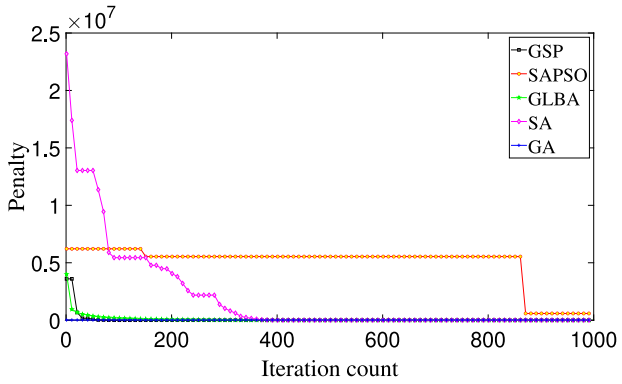


Fig. 11. Penalty of GSP, SAPSO, GLBA, SA, and GA.

cannot generate satisfactory solutions meeting all the constraints. The penalties of GSP, GLBA, and SA all reduce to zero at the end of the search processes. This proves that the solutions finally obtained by these algorithms are valid, and they strictly meet all constraints.

However, SA yields a satisfactory solution after 350 iterations. GLBA has a similar iteration curve to GSP, but it also achieves the penalty of zero after 400 iterations. GSP has a lower penalty at the beginning of iterations, and it achieves the penalty of zero after 100 iterations, which outperforms other compared algorithms. As a result, the compared algorithms or other similar ones can also be used to solve the problem. However, it is hard for them to balance the search efficiency and the search accuracy. Specifically, SAPSO and GA cannot find the desired solutions that meet the constraints after their required total iterations. GLBA and SA find their valid solutions but they have low search efficiency. GSP has higher search efficiency and accuracy than state-of-art metaheuristics, and it can better solve the considered offloading problem.

Furthermore, Figs. 12–15 illustrate the system cost of MDs, SBS, CDC, and the total system cost of the proposed strategy concerning the number of MDs. We also show the total system cost of local offloading, random offloading, and full offloading. It is shown that the proposed strategy outperforms random and local computing concerning the cost of MDs in Fig. 12. Its cost is higher than that of full offloading yet leads to larger transmission latency. We aim to minimize the total system cost such that the latency requirement is strictly met. Thus, the latency of full offloading is not acceptable. Fig. 13 illustrates that the system cost of SBS with the proposed offloading strategy is 0.008 $ after the case with 10 MDs. The reason is that the total tasks exceed the maximum processing capacity of SBS, and excessive ones are further offloaded to CDC. It is worth noting that the cost of SBS with local computing is zero because no tasks are offloaded to SBS and that full offloading maintains the maximum value. Fig. 14 shows that the proposed strategy outperforms the random and full offloading with respect to the system cost of CDC. It is higher than that with local computing that does not involve CDC. Furthermore, Fig. 15 illustrates the total system cost with the proposed offloading strategy with respect to the number of MDs. It is observed that our offloading strategy yields the least cost among all the strategies.

5.3. Experimental discussion

In summary, Fig. 2 shows the system cost and its corresponding penalty for different values of N. It is shown that the penalties are all zero with different values of N, which proves the robustness and excellent optimization ability of the GSP. Figs. 3–5 show the total system cost with different experimental parameters (N, λ_{max}^{SBS}, L_{max}, S and d_i). The final cost shows an increasing trend as these variables increase. This is because more MDs and higher task arriving rates

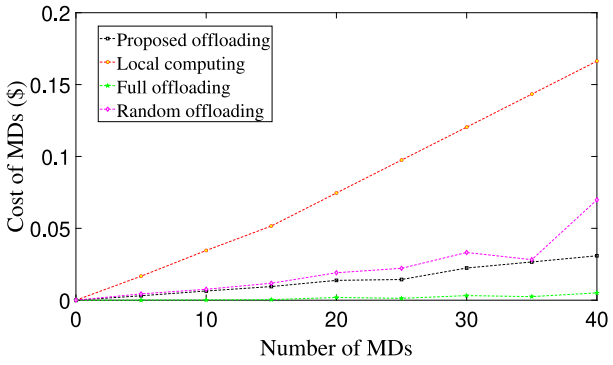


Fig. 12. Cost of MDs w.r.t. varying numbers of MDs.

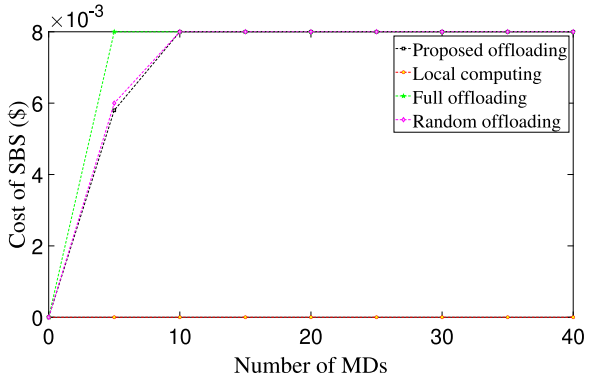


Fig. 13. Cost of SBS w.r.t. varying numbers of MDs.

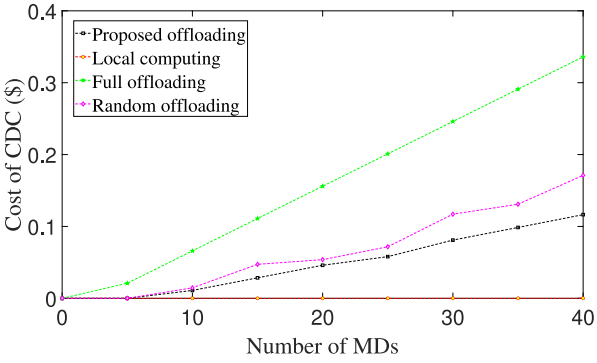


Fig. 14. Cost of CDC w.r.t. varying numbers of MDs.

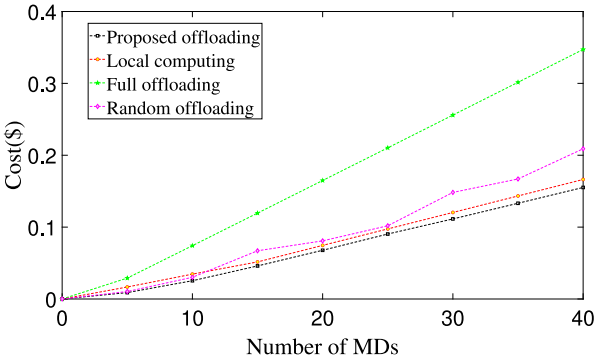


Fig. 15. Total cost w.r.t. varying numbers of MDs.

bring more tasks that need to be executed, thus causing additional energy consumption of the system. Moreover, larger d_i causes more communication energy between MD i and the SBS. Furthermore, the total cost of the system shows a linear increase as the above variables increase, which proves that the proposed GSP can well handle the computation offloading under different situations in this hybrid cloud and edge systems. Moreover, Figs. 6–8 illustrate the cost of MDs, SBS, and the CDC with different task arriving rates, respectively. They show that as the task arriving rate increases, MDs cannot process these tasks locally and then offload them to the SBS for execution. Moreover, when the processing limit of the SBS is reached, SBS further offloads those exceeding tasks to the CDC. This demonstrates the process of computation offloading and justifies the correctness of system modeling.

Figs. 9–11 compare our proposed GSP with four other metaheuristic optimization algorithms (SAPSO, GLBA, SA, and GA). Specifically, Fig. 9 illustrates that GSP achieves the lowest system cost with different numbers of MDs compared to its peers. Figs. 10 and 11 show the iteration curves and corresponding penalty curves of these algorithms. They prove that GSP can obtain valid optimization results in a shorter time. Moreover, Figs. 12–15 compare the proposed GSP with three different offloading strategies (local computing, full offloading, and random offloading). They show the cost of MDs, SBS, CDC, and the system with different numbers of MDs. In addition, the cost of each strategy shows an increasing trend as the number of MDs increases. This is because more MDs in the system bring more pending tasks, and they need additional energy to be processed. Finally, the total system cost of GSP is the lowest among all strategies.

It is worth noting that the dimension of the proposed optimization problem reaches one hundred. Moreover, the dimension of the problem will continue to increase in the future as the number of users keeps growing. In this case, we will further consider enhancing our GSP by incorporating dimension reduction tools, *e.g.*, principal component analysis, and autoencoder. In addition, we will also consider combining a surrogate model with our GSP. Surrogate models can replace a part of true models for the function evaluation, thereby saving computational resources of GSP when solving high-dimensional problems.

6. Conclusion

With the development of wireless communication and hardware technologies, mobile devices (MDs) can support various resource-hungry applications. However, MDs only have constrained resources and energy. Current hybrid cloud and edge systems provide a partial computation offloading strategy for executing all tasks within their delay bounds. Existing studies have not systematically studied the minimization problem of the total cost of all tasks of complex applications in such hybrid systems. To fill the gap, this work designs an architecture of a hybrid cloud and edge system, which consists of multiple MDs, one small base station (SBS), and one centralized cloud data center (CDC). A constrained total cost minimization problem for such a system is established, which is addressed by a proposed optimization algorithm called Genetic Simulated-annealing-based Particle swarm optimization (GSP). It seamlessly combines the merits of genetic operations and the global search ability of simulated annealing and jointly optimizes the task offloading among MDs, SBS, and CDC, as well as a holistic collection of related factors. Real trace data-based experiments prove that GSP achieves less total system cost in fewer iterations than its typical peers.

In the future, we will explore further work in two directions. First, we plan to study a more complicated architecture for hybrid cloud and edge systems. Our current architecture comprises multiple MDs, one SBS, and one CDC. When one SBS is not powerful enough to support its nearby MDs, enabling sharing (Chen et al., 2021) among SBSs will be beneficial. We plan to study a more complex hybrid system with multiple heterogeneous SBSs. Second, we plan to evaluate our GSP in newly emerging 5G/6G edge systems to study its broader applicability further.

CRediT authorship contribution statement

Haitao Yuan: Methodology, Investigation, Formal analysis, Writing – original draft, Funding acquisition. **Jing Bi:** Resources, Supervision, Project administration. **Ziqi Wang:** Software, Visualization, Investigation. **Jinhong Yang:** Data curation, Validation. **Jia Zhang:** Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Haitao Yuan and Jing Bi report financial support was provided by National Natural Science Foundation of China. Haitao Yuan and Jing Bi report financial support was provided by Beijing Natural Science Foundation. Haitao Yuan reports financial support was provided by Fundamental Research Funds for the Central Universities. If there are other authors, they 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

Ale, L., Zhang, N., Fang, X., Chen, X., Wu, S., & Li, L. (2021). Delay-aware and energy-efficient computation offloading in mobile edge computing using deep reinforcement learning. *IEEE Transactions on Cognitive Communications and Networking*, *7*, 881–892.

Bi, J., Wang, Z., Yuan, H., Qiao, J., Zhang, J., & Zhou, M. (2023). Self-adaptive teaching-learning-based optimizer with improved RBF and sparse autoencoder for complex optimization problems. In *2023 IEEE international conference on robotics and automation* (pp. 7966–7972).

Bi, J., Yuan, H., Duanmu, S., Zhou, M., & Abusorrah, A. (2021). Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization. *IEEE Internet of Things Journal*, *8*, 3774–3785.

Bi, J., Yuan, H., Zhang, K., & Zhou, M. (2022a). Energy-minimized partial computation offloading for delay-sensitive applications in heterogeneous edge networks. *IEEE Transactions on Emerging Topics in Computing*, *10*, 1941–1954.

Bi, J., Yuan, H., Zhang, J., & Zhou, M. (2022b). Green energy forecast-based bi-objective scheduling of tasks across distributed clouds. *EEE Transactions on Sustainable Computing*, *7*, 619–630.

Bi, J., Zhai, J., Yuan, H., Wang, Z., Qiao, J., Zhang, J., & Zhou, M. (2023). Multi-swarm genetic gray wolf optimizer with embedded autoencoders for high-dimensional expensive problems. In *2023 IEEE international conference on robotics and automation* (pp. 7265–7271).

Bozorgchenani, A., Mashhadi, F., Tarchi, D., & Salinas, S. (2020). Multi-objective computation sharing in energy and delay constrained mobile edge computing environments. *IEEE Transactions on Mobile Computing*, *20*, 2992–3005.

Casola, V., Benedictis, A., Martino, S., Mazzocca, N., & Starace, L. (2020). Security-aware deployment optimization of cloud-edge systems in industrial IoT. *IEEE Internet of Things Journal*, *8*, 12724–12733.

Chen, S., Li, Q., Zhou, M., & Abusorrah, A. (2021). Recent advances in collaborative scheduling of computing tasks in an edge computing paradigm. *Sensors*, *21*, 1–22.

Chetlur, V., & Dhillon, H. (2020). On the load distribution of vehicular users modeled by a Poisson line Cox process. *IEEE Wireless Commun. Lett.*, *9*, 2121–2125.

Cong, P., J., Zhou., Li, L., Cao, K., Wei, T., & Li, K. (2020). A survey of hierarchical energy optimization for mobile edge computing: A perspective from end devices to the cloud. *ACM Computing Surveys*, *53*, 1–44.

Dai, Y., Zhang, K., Maharjan, S., & Zhang, Y. (2020). Edge intelligence for energy-efficient computation offloading and resource allocation in 5G beyond. *IEEE Transactions on Vehicular Technology*, *69*, 12175–12186.

Feng, C., Han, P., Zhang, X., Zhang, Q., Zong, Y., Liu, Y., & Guo, L. (2022). Cost-minimized computation offloading of online multifunction services in collaborative edge-cloud networks. *IEEE Transactions on Network and Service Management*, *20*, 292–304.

Gao, S., Zhou, M., Wang, Y., Cheng, J., Yachi, H., & Wang, J. (2019). Dendritic neuron model with effective learning algorithms for classification, approximation, and prediction. *IEEE Transactions on Neural Networks and Learning Systems*, *30*, 601–614.

Gong, H., Li, R., An, J., Bai, Y., & Li, K. (2020). Quantitative modeling and analytical calculation of anelasticity for a cyber-physical system. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *50*, 4746–4761.

Jia, M., Cao, J., & Liang, W. (2017). Optimal cloudlet placement and user to cloudlet allocation in Wireless Metropolitan Area networks. *IEEE Transactions on Cloud Computing*, *5*, 725–737.

Khayyat, M., Elgendy, I., Muthanna, A., Alshahrani, A., Alharbi, S., & Koucheryavy, A. (2020). Advanced deep learning-based computational offloading for multilevel vehicular edge-cloud computing networks. *IEEE Access*, *8*, 137052–137062.

Li, X., Zhang, C., Gu, B., Yamori, K., & Tanaka, Y. (2019). Optimal pricing and service selection in the mobile cloud architectures. *IEEE Access*, *7*, 43564–43572.

Lin, C., Han, G., Qi, X., Guizani, M., & Shu, L. (2020). A distributed mobile fog computing scheme for mobile delay-sensitive applications in SDN-enabled vehicular networks. *IEEE Transactions on Vehicular Technology*, *69*, 5481–5493.

Luo, Q., Li, C., Luan, T., & Shi, W. (2021). Minimizing the delay and cost of computation offloading for vehicular edge computing. *IEEE Transactions on Services Computing*, *15*, 2897–2909.

Lyden, S., & Haque, M. (2016). A simulated annealing global maximum power point tracking approach for PV modules under partial shading conditions. *IEEE Transactions on Power Electronics*, *31*, 4171–4181.

Metawa, N., Hassan, M., & Elhoseny, M. (2017). Genetic algorithm based model for optimizing bank lending decisions. *Expert Systems with Applications*, *80*, 75–82.

Mu, S., Zhong, Z., & Zhao, D. (2020). Energy-efficient and delay-fair mobile computation offloading. *IEEE Transactions on Vehicular Technology*, *69*, 15746–15759.

Panda, A., & Pani, S. (2016). A symbiotic organisms search algorithm with adaptive penalty function to solve multi-objective constrained optimization problems. *Applied Soft Computing*, *46*, 344–360.

Qi, S., Chen, J., Chen, P., Wen, P., Niu, X., & Xu, L. (2023). An efficient GAN-based predictive framework for multivariate time series anomaly prediction in cloud data centers. *The Journal of Supercomputing*, *80*, 1268–1293.

Ren, J., Yu, G., He, Y., & Li, J. (2019). Collaborative cloud and edge computing for latency minimization. *IEEE Transactions on Vehicular Technology*, *68*, 5031–5044.

Sahni, Y., Cao, J., Yang, L., & Ji, Y. (2021). Multi-hop multi-task partial computation offloading in collaborative edge computing. *IEEE Transactions on Parallel and Distributed Systems*, *32*, 1133–1145.

Saleem, U., Liu, Y., Jangsher, S., Tao, X., & Li, Y. (2020). Latency minimization for D2D-enabled partial computation offloading in mobile edge computing. *IEEE Transactions on Vehicular Technology*, *69*, 4472–4486.

Silva, L., Magaia, A., Sousa, B., Kobusińska, A., Casimiro, A., Mavromoustakis, C., Mastorakis, G., & Albuquerque, V. (2021). Computing paradigms in emerging vehicular environments: A review. *IEEE/CAA Journal of Automatica Sinica*, *8*, 491–511.

Sohaib, M., Jeon, W., & W., Yu. (2023). Hybrid online–offline learning for task offloading in mobile edge computing systems. *IEEE Transactions on Wireless Communication*.

Su, Q., Zhang, Q., Li, W., Zhang, X., & Edge Collaboration (2024). Primal-dual-based computation offloading method for energy-aware cloud. *IEEE Transactions on Mobile Computing*, *23*, 1534–1549.

Sun, Z., Sun, G., Liu, Y., Wang, J., & Cao, D. (2024). BARGAIN-MATCH: A game theoretical approach for resource allocation and task offloading in vehicular edge computing networks. *IEEE Transactions on Mobile Computing*, *23*, 1655–1673.

Tsai, C., Hsia, C., Yang, S., Liu, S., & Fang, Z. (2020). Optimizing hyperparameters of deep learning in predicting bus passengers based on simulated annealing. *Applied Soft Computing*, *88*, 11–22.

Vincent, F., Redi, A., Hidayat, Y., & Wibowo, O. (2017). A simulated annealing heuristic for the hybrid vehicle routing problem. *Applied Soft Computing*, *53*, 119–132.

Wang, Y., Sheng, X., Wang, L., & J., Li. (2016). Mobile-edge computing: Partial computation offloading using dynamic voltage scaling. *IEEE Transactions on Communications*, *64*, 4268–4282.

Wang, W., Wang, Q., & Sohraby, K. (2017). Multimedia sensing as a service (MSaaS): Exploring resource saving potentials of at cloud-edge IoT and fogs. *IEEE Internet of Things Journal*, *4*, 487–495.

Whaiduzzaman, M., Naveed, A., & Gani, A. (2018). MobiCoRE: Mobile device based cloudlet resource enhancement for optimal task response. *IEEE Transactions on Services Computing*, *11*, 144–154.

Wu, Z., Li, L., Fei, Z., Zheng, Z., Li, L., & Z., Han. (2020). Energy-efficient robust computation offloading for fog-IoT systems. *IEEE Transactions on Vehicular Technology*, *69*, 4417–4425.

Xu, C., Liu, S., Zhang, C., Huang, Y., Lu, Z., & Yang, L. (2021). Multi-agent reinforcement learning based distributed transmission in collaborative cloud-edge systems. *IEEE Transactions on Vehicular Technology*, *70*, 1658–1672.

Yu, Y., Gong, Y., Gong, S., & Guo, Y. (2020). Joint task offloading and resource allocation in UAV-enabled mobile edge computing. *IEEE Internet of Things Journal*, *7*, 3147–3159.

Yuan, H., Bi, J., & M., Zhou. (2022). Geography-aware task scheduling for profit maximization in distributed green data centers. *IEEE Transactions on Cloud Computing*, *10*, 1864–1874.

Yuan, H., Bi, J., Tan, W., Zhou, M., Li, B., & Li, J. (2017). TTSA: An effective scheduling approach for delay bounded tasks in hybrid clouds. *IEEE Transactions on Cybernetics*, *47*, 3658–3668.

Yuan, H., Hu, Q., Wang, M., Bi, J., & Zhou, M. (2022). Cost-minimized user association and partial offloading for dependent tasks in hybrid cloud–edge systems. In *2022 IEEE 18th international conference on automation science and engineering* (pp. 1059–1064). Mexico City, Mexico.

Yuan, H., & Zhou, M. (2021). Profit-maximized collaborative computation offloading and resource allocation in distributed cloud and edge computing systems. *IEEE Transactions on Automation Science and Engineering*, *18*, 1277–1287.

Yue, X., & Zhang, H. (2020). Modified hybrid bat algorithm with genetic crossover operation and smart inertia weight for multilevel image segmentation. *Applied Soft Computing*, *90*, 1–43.

Zaw, C., Tran, M., Han, Z., & Hong, C. (2023). Radio and computing resource allocation in co-located edge computing: A generalized Nash equilibrium model. *IEEE Transactions on Mobile Computing*, *22*, 2340–2352.

Zeng, N., Wang, Z., Liu, W., Zhang, H., Hone, K., & Liu, X. (2020). A dynamic neighborhood-based switching particle swarm optimization algorithm. *IEEE Transactions on Cybernetics*, *52*, 9290–9301.

Zhang, W., Elgendy, A., Hammad, M., Iliyasu, A., Du, X., Guizani, M., & Abd El-Latif, A. (2021). Secure and optimized load balancing for multitier IoT and edge-cloud computing systems. *IEEE Internet of Things Journal*, *8*, 8119–8132.

Zhang, Y., Lan, X., Ren, J., & Cai, L. (2020). Efficient computing resource sharing for mobile edge-cloud computing networks. *IEEE/ACM Transactions on Networking*, *28*, 1227–1240.

Zhu, Q., Tang, H., Huang, J., & Hou, Y. (2021). Task scheduling for multi-cloud computing subject to security and reliability constraints. *IEEE/CAA Journal of Automatica Sinica*, *8*, 848–865.

Haitao Yuan received the Ph.D. degree in Computer Engineering from New Jersey Institute of Technology (NJIT), Newark, NJ, USA in 2020. He is currently an Associate Professor at the School of Automation Science and Electrical Engineering at Beihang University, Beijing, China. His research interests include cloud computing, edge computing, data centers, big data, machine learning, deep learning, and optimization algorithms. He received the Chinese Government Award for Outstanding Self-Financed Students Abroad, the 2021 Hashimoto Prize from NJIT, and the Best Paper Award in the 17th ICNSC. He serves as an associate editor for Expert Systems with Applications.

Jing Bi received her B.S. and Ph.D. degrees in Computer Science from Northeastern University, Shenyang, China, in 2003 and 2011, respectively. From 2013 to 2015, she was a Post-doc researcher in the Department of Automation at Tsinghua University, Beijing, China. From 2011 to 2013, she was a research scientist at the Beijing Research Institute of Electronic Engineering Technology in Beijing, China. From 2009 to 2010, she was a research assistant and participated in research on cloud computing at IBM Research, Beijing, China. From 2018 to 2019, she was a Visiting Research Scholar with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, USA. She is a Professor at the Faculty of Information Technology, School of Software Engineering, Beijing University of Technology, Beijing, China. She has over 150 publications in international journals and conference proceedings. Her research interests include distributed computing, cloud computing, large-scale data analytics, machine learning, and performance optimization. Dr. Bi received the IBM Fellowship Award, the Best Paper Award at the 17th IEEE International Conference on Networking, Sensing and Control, and the First-Prize Progress Award of the Chinese Institute of Simulation Science and Technology. She is now an Associate Editor of IEEE Transactions on Systems Man and Cybernetics: Systems. She is a senior member of the IEEE.

Ziqi Wang is currently a Master's student in the Faculty of Information Technology, School of Software Engineering, Beijing University of Technology, Beijing, China. Before that, he received his B.E. degree in Internet of Things from Beijing University of Technology in 2022. His research interests include cloud computing, task scheduling, intelligent optimization algorithms, and machine learning.

Jinhong Yang received the Ph.D. degree in Computer Application Technology from Harbin Engineering University, Harbin, Heilongjiang, China in 2017. She is a Senior Engineer with CSSC Systems Engineering Research Institute in Beijing, China. Her research interests include machine learning, data mining, knowledge reasoning, deep learning, and intelligent optimization. She is a reviewer for Expert Systems With Applications, International Journal of Machine Learning and Cybernetics, *etc*.

Jia Zhang received the Ph.D. degree in computer science from the University of Illinois at Chicago. She is currently the Cruse C. and Marjorie F. Calahan Centennial Chair in Engineering, Professor of Department of Computer Science in the Lyle School of Engineering at Southern Methodist University. Her research interests emphasize the application of machine learning and information retrieval methods to tackle data science infrastructure problems, with a recent focus on scientific workflows, provenance mining, software discovery, knowledge graph, and interdisciplinary applications of all of these interests in earth science. She is a senior member of the IEEE.