Cost-Efficient Task Offloading in Mobile Edge Computing With Layered Unmanned Aerial Vehicles

Haitao Yuan, *Senior Member, IEEE*, Meijia Wang, *Student Member, IEEE*, Jing Bi, *Senior Member, IEEE*, Shuyuan Shi, Jinhong Yang, Jia Zhang, *Senior Member, IEEE*, MengChu Zhou, *Fellow, IEEE*, and Rajkumar Buyya, *Fellow, IEEE*

***Abstract*—Mobile edge computing (MEC) paradigm supports cloud-like computing capabilities at the edge of the network and offers low-latency services. Proxy servers of MEC with mobility and limited computing, *e.g.,* flying unmanned aerial vehicles (UAVs) have emerged as competitors in providing services. This work considers a task offloading problem for an UAV-assisted MEC system and designs an integrated cloud–edge network with multiple mobile users (MUs) and layered UAVs to improve MEC with a network of UAVs. In our system, edge UAVs (EUAVs) and the cloud collaborate to provide caching and computing services for MUs. We consider static and dynamic applications that support task offloading. Our proposed approach minimizes the weighted cost of latency and energy consumption by jointly optimizing caching and offloading, deployment of EUAVs, and allocation of computation resources. Simultaneously, this work also considers UAVs' caching and computation capacities while meeting MUs' latency and energy constraints. Thus, a constrained mixed integer nonlinear program for a layered UAV-assisted hybrid cloud-edge system is formulated. To solve it, this work designs a hybrid metaheuristic algorithm named adaptive and genetic simulated annealing (SA)-based particle swarm optimization (AGSP). Experimental results with a real-life dataset verify that the AGSP's system energy consumption and task latency are reduced by at least 7.4% and 8.46%, respectively, compared with the state-of-the-art algorithms, thus proving that AGSP greatly enhances the energy and latency of the system.**

***Index Terms*—Computation offloading, mobile edge computing (MEC), particle swarm optimization (PSO), unmanned aerial vehicles (UAVs), wireless caching.**

Manuscript received 5 April 2024; accepted 23 May 2024. Date of publication 22 July 2024; date of current version 25 September 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62173013 and Grant 62073005; in part by the Beijing Natural Science Foundation under Grant 4232049 and Grant L233005; in part by the Fundamental Research Funds for the Central Universities under Grant YWF-23-03-QB-015; and in part by Beihang World TOP University Cooperation Program. *(Corresponding author: Haitao Yuan.)*

Haitao Yuan and Meijia Wang are with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China (e-mail: yuan@buaa.edu.cn; meijia@buaa.edu.cn).

Jing Bi is with the School of Software Engineering, Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China (e-mail: bijing@bjut.edu.cn).

Shuyuan Shi is with the Fert Beijing Institute, MIIT Key Laboratory of Spintronics, and the School of Integrated Circuit Science and Engineering, Beihang University, Beijing 100191, China (e-mail: smeshis@buaa.edu.cn).

Jinhong Yang is with CSSC Systems Engineering Research Institute, Beijing 100036, China (e-mail: yangjinhong.66@163.com).

Jia Zhang is with the Department of Computer Science, Southern Methodist University, Dallas, TX 75206 USA (e-mail: jiazhang@smu.edu).

MengChu Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: zhou@njit.edu).

Rajkumar Buyya is with the Cloud Computing and Distributed Systems (CLOUDS) Lab, School of Computing and Information Systems, University of Melbourne, Melbourne, VIC 3010, Australia (e-mail: rbuyya@unimelb.edu.au).

This article has supplementary downloadable material available at https://doi.org/10.1109/JIOT.2024.3408216, provided by the authors.

Digital Object Identifier 10.1109/JIOT.2024.3408216

I. Introduction

THE CURRENT demand for energy-efficient and fast services of mobile users (MUs) intensifies due to their dramatic increase [1]. Mobile edge computing (MEC) emerges, where MUs can partially offload computation-intensive tasks to the network's edge [2] for reducing congestion of a backbone network in an energy-saving and cost-effective manner [3]. In recent years, a new paradigm of unmanned aerial vehicle (UAV)-enabled MEC has been proposed, where UAVs provide high mobility, lightweight features, and the capability to offer uninterrupted services to MUs, irrespective of the geographical limits [4]. UAV-supported MEC networks have the following advantages. First, UAVs can be deployed flexibly even in places that are unreliable to build terrestrial MEC networks [5]. Second, UAVs provide short-range Line of Sight (LoS) links for offloading tasks and transmitting their results [6]. Third, UAVs have mobility, flexibility, and maneuverability, and therefore, their trajectories can be optimized arbitrarily to realize dynamic planning in different heterogeneous scenarios [7].

Several global companies, such as Google, Facebook, Amazon, and Huawei have launched projects for the UAV-assisted MEC systems. Unlike the traditional MEC servers, UAVs realize real-time scheduling with their computation resources *via* trajectory planning. Yet, they have to handle dramatic variations in both temporal and spatial aspects. In a network with multiple MUs and UAVs, multiple MUs often transmit data simultaneously, inevitably causing interference and resource competition among them [8]. Thus, data transmissions of all MUs suffer from loss because of mutual influence. Besides, multiple MUs compete to offload computing tasks to UAVs. Their offloading performance is significantly influenced by resource sharing and wireless bandwidth allocation of UAVs [9]. In addition, in a multi-UAV

2327-4662 © 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: University of Melbourne. Downloaded on September 26,2024 at 01:07:41 UTC from IEEE Xplore. Restrictions apply.

environment, there are many UAVs available to each MU, and the performance of computation offloading largely depends on the selection of UAVs [10].

Current multilayer UAVs can be divided into two types of architectures.

1) *Multilayer Centralized Architecture:* It includes a layer of MUs on the ground, a layer of distributed UAVs, and a centralized UAV layer [11]. MUs transmit their tasks to the distributed UAV layer. Then, these distributed UAVs relay the queued tasks to the centralized UAV, which acts as the decision center. This architecture imposes high-performance requirements on the centralized UAV with large battery and computation capacities. This architecture is advantageous for applications requiring centralized data processing and immediate responsiveness, *e.g.,* the surveillance systems.
2) *Multilayer Distributed Architecture:* In this architecture, multilayer UAVs of various types and flying altitudes utilize directional antennas to communicate with MUs [12]. UAVs in each layer maintain a fixed height but are distributed horizontally. This architecture places less performance requirements on individual UAVs. It focuses on the communication connectivity, association probability, and coverage probability within the multitier UAV networks. Depending on specific application requirements and network conditions, it offers different tradeoffs regarding communication overhead, scalability, and fault tolerance. This architecture is suitable in scenarios characterized by extensive coverage and flexible deployment, *e.g.,* the disaster relief systems.

There are a few studies on computing and caching in an UAV-assisted MEC network [13], [14], [15]. Yet, few studies have comprehensively considered both of them. Integrating caching into computing enhances resource reuse to improve the performance of MUs. Investigating correlations between the caching and computing in a layer of UAVs and a cloud becomes crucial. In addition, it is also critical to balance the latency and the energy consumption in the system. However, a challenge arises from the computing tasks of uneven MUs and system heterogeneity, which complicates the selection of cached data and offloading locations. Unlike the existing studies, this work focuses on a layered UAV-assisted MEC system with multiple MUs. These UAVs collaborate with a cloud to offer caching and computation offloading services to MUs and leverage the edge layer's caching capacity and computing resources. The main contributions of the work are threefold.

1) It formulates an offloading problem in such a system, and innovatively considers static and dynamic applications that support offloading their tasks. A mixed integer nonlinear program (MINLP) is formulated to minimize the weighted cost of latency and energy consumption of the system.
2) It designs a hybrid optimization algorithm called adaptive genetic SA-based particle swarm optimization (AGSP), which integrates genetic operations of the genetic algorithm (GA) and a Metropolis acceptance criterion of SA into a nonlinear and dynamic particle swarm optimizer (PSO) with adaptive inertial weights.
3) It finds a close-to-optimal strategy that jointly optimizes computing resources, transmission power, associations between MUs and UAVs, task offloading ratios, and coordinates of MUs and UAVs to minimize the weighted cost of latency and energy consumption of the system.

The remainder of this article is structured as follows. Section II gives the related work. Section III introduces our system model. Section IV describes the proposed AGSP. Section V presents the numerical results. Finally, Section VI concludes the work.

II. Related Work

This section reviews the UAV-assisted MEC system. The integration of UAVs and MEC is classified according to the roles of UAVs.

A. UAVs as User Nodes

UAVs act as the user nodes that offload local tasks to nearby ground base stations (BSs), which return the executed result to themselves [16]. Existing studies include the deployment of UAVs [17], wireless transmission power distribution [18], and associations of MUs and deployment of UAVs [19]. The study in [20] focuses on a heterogeneous network enabling the uplink and downlink data transmission simultaneously, where a single UAV acts as a communicator while other UAVs act as aerial BSs gathering data from diverse sensor nodes. It jointly considers allocating communication resources, transmission power, and UAV trajectories to maximize the network throughput. The study in [21] optimizes both the wireless transmission power and 3-D trajectories of UAVs for enhancing the total aggregation rate of UAV-assisted interference channels. The study in [22] integrates cellular-connected UAVs with existing wireless networks and discusses their efficiency at the network and user levels. It analyses the performance of users and networks in a cellular network serving both ground users and UAVs in the downlink. The study in [7] investigates an UAV-assisted communication model, where rotary-wing UAVs are deployed to provide continuous coverage for the mobile nodes and optimize their trajectories while ensuring sufficient coverage for MUs.

B. UAVs as MEC Servers

UAVs can also serve as the MEC servers, assisting MUs in executing their computation tasks. This architecture is particularly suitable when an UAV has sufficient computation resources and battery capability. Alternatively, it is applicable in scenarios where the MEC servers on the ground are unavailable. In [23], an MEC network assisted by UAVs is presented, which integrates UAVs' path planning and collaboration between the air and ground to facilitate communication and edge computing. It is expanded in [24] to a broader scenario where an UAV serves as a cloudlet, providing chances for computation offloading to MUs with limited execution capabilities. In [25], an MEC-enabled UAV architecture is introduced to minimize the UAV's energy consumption by optimizing both the computing and network resources of MUs. In [26], a resource allocation problem in an UAV-enabled MEC

Authorized licensed use limited to: University of Melbourne. Downloaded on September 26,2024 at 01:07:41 UTC from IEEE Xplore. Restrictions apply.

system is explored, considering both the partial and binary computation offloading modes. This work jointly optimizes the CPU frequencies, user transmission power, task offloading, and UAVs' trajectories to maximize the weighted sum of computation rates of all MUs. The study in [27] proposes a task offloading framework in an UAV-assisted MEC network to minimize the average task delay by jointly optimizing MUs' associations and UAV deployment.

C. UAVs as Relays

UAVs can also act as aerial dynamic relays, efficiently assisting MUs in offloading their compute-intensive tasks to remote BSs or other MEC servers and receiving the computed data. In [28], a hybrid system adopts UAVs as the MEC servers connected with terrestrial BSs. UAVs navigate along a cell periphery to manage task offloading for the users at the cell edge, transmitting computation tasks to BSs on the ground. It presents an innovative framework to maximize the total throughput of MUs, considering MUs' partitioning, dynamic spectrum allocation, and UAVs' trajectories. Its system performance is significantly enhanced by considering UAVs' path planning and utilizing cyclical multiple access. In [29], an amplify-and-forward protocol for UAVs is proposed, focusing on scheduling time division relays among multiple pairs of MUs. It considers time slot allocation, wireless transmission power, and UAVs' trajectories and maximizes the minimum average information rate of pairing MUs. The study in [30] introduces a multihop UAV relaying network to maximize the network throughput. It considers computation resources and UAVs' trajectories while considering constraints of transmission power, spectrum bandwidth, information-causality of multihop relaying, and collision avoidance. The study in [31] maximizes the throughput in an UAV-assisted relaying system. It jointly optimizes the communication strategies and UAVs' trajectories while considering their maneuverability. It leverages an iterative approach that combines the block coordinate descent and successive convex approximation methods.

Unlike these studies, this work explores incorporating a ground cloud center into a multi-UAV-enabled MEC system. In this system, UAVs can not only offer MEC services as a supplement to ground BSs, particularly when the latter is damaged or overloaded, but also serve as relays to transmit the computation tasks to the cloud center and relay the executed result to terrestrial MUs. By employing partial computation offloading, MUs offload compute-intensive tasks to UAVs. In cases where UAVs have limitations in computing capacity and cannot handle all tasks, overloaded ones are transmitted to the central cloud server. Therefore, UAVs act as a combination of the MEC servers and relays. In addition, UAVs in our system also provide cache services to MUs, which cache, process, and deliver computation data. Furthermore, this work also considers static and dynamic applications supporting task offloading.

III. Problem Formulation

Fig. 1 shows the framework of an UAV-assisted hybrid cloud-edge network, which includes the cloud, the edge, and the MUs. A linked UAV (LUAV) is deployed and physically connected to the cloud. Between the LUAV and the cloud, energy transmission and data transmission are consolidated in the same cord. Because of the limited mobility of the cord, LUAV is positioned directly over the cloud. The length of this cord is in a range of [80 m, 150 m]. The cord sustains the long-duration flight of LUAV and enables a high data transmission rate and safe communication between the LUAV and the cloud. This work neglects the latency between the LUAV and the cloud due to their faster transmission rates. Table S1 in the supplementary file summarizes the main notations used in this work.

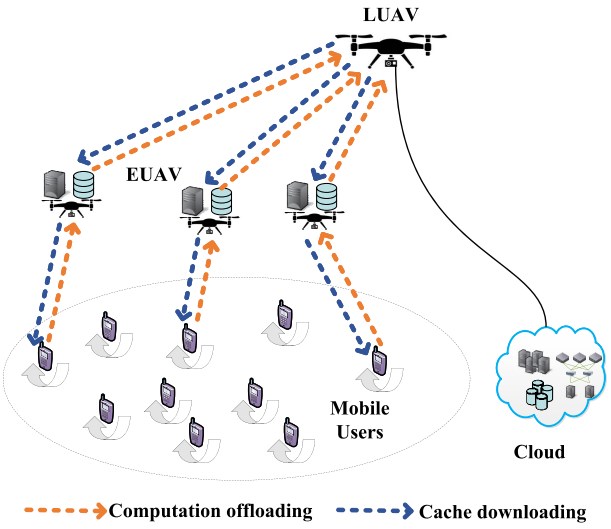

Fig. 1. Framework of an UAV-assisted hybrid cloud–edge system.

UAV networks can be operated in a distributed manner. However, distributed algorithms cannot guarantee to yield the globally optimal solution to our total system cost optimization problem. Thus, the total system cost cannot be minimized with the distributed algorithms. In addition, the distributed algorithms often require complex information communication protocols and large data transmission among MUs, edge UAVs (EUAVs), LUAV, and the cloud. Different from the existing studies [32], [33], our proposed AGSP is a typical centralized optimization algorithm, which is deployed in LUAV. LUAV periodically collects real-time information about MUs, EUAVs, and the cloud, and runs AGSP to yield the best task offloading strategy.

In the edge layer, EUAVs with limited cache and computing capabilities serve as the MEC servers around MUs. The number of EUAVs is denoted by K and a set of EUAVs is denoted by $\mathcal{K}$ and $\mathcal{K} = \{1, 2, \ldots, K\}$. $\mathbf{u}_k$ denotes a vector of 2-D coordinates of EUAV k and $\mathbf{u}_k = [\alpha_k, \beta_k]$. This work considers that all EUAVs are located in the same horizontal plane. Thus, $\tilde{h}$ denotes the fixed height of EUAVs. The deployment of EUAV k has to be in its limits, *i.e.,*

$$\check{\alpha} \le \alpha_k \le \hat{\alpha} \tag{1}$$

$$\check{\beta} \le \beta_k \le \hat{\beta} \tag{2}$$

Authorized licensed use limited to: University of Melbourne. Downloaded on September 26,2024 at 01:07:41 UTC from IEEE Xplore. Restrictions apply.

where $\hat{\alpha}$ and $\check{\alpha}$ denote the upper and lower limits of α_k. $\hat{\beta}$ and $\check{\beta}$ denote the upper and lower limits of β_k.

Similarly, the number of MUs is denoted by M, and the set of MUs is denoted by $\mathcal{M}$ and $\mathcal{M} = \{1, 2, \ldots, M\}$. The height of LUAV is denoted by $\bar{h}$. The 2-D coordinates of MU m and LUAV are denoted by $\mathbf{u}_m$ ($\mathbf{u}_m = [\alpha_m, \beta_m]$) and $\mathbf{u}_0$ ($\mathbf{u}_0 = [\alpha_0, \beta_0]$), respectively. The computation offloading can be realized only after files have been completely delivered. As illustrated in Fig. 1, we divide the transmission process into two distinct phases: 1) caching content delivery and 2) computation offloading. MUs acquire the requested files from the LUAV or EUAVs that act as the cache servers in the former. MUs determine where to perform computation tasks in the latter according to the files yielded in the former.

A. Communication Model

We assume that EUAVs establish communication with both LUAVs and terrestrial MUs. However, due to their considerable distances, terrestrial MUs are restricted from communicating solely with EUAVs. MUs and UAVs utilize the orthogonal bandwidth to communicate with others, reducing interference and improving signal quality.

1) Communication Between EUAVs and MUs: When dispatching EUAVs to provide services to ground MUs in open suburban scenarios, the likelihood of establishing LoS links between EUAVs and MUs is relatively high. This work adopts a LoS-dominant channel model, which encompasses two types of channel fading: 1) the large-scale path-loss fading and 2) the small-scale Rician fading. Both the downlink and uplink channels are assumed to undergo large-scale path loss fading and independent and identically distributed small-scale Rician fading.

Thus, the channel link between MU m and EUAV k can be mathematically modeled as

$$h_{m,k} = \Omega_{m,k}\sqrt{\xi_{m,k}} \tag{3}$$

where $\Omega_{m,k}$ is employed to characterize the small-scale fading, which is modeled by a Rician distribution following a weighted noncentral χ^2 distribution, which is modeled as

$$\Omega_{m,k} = \sqrt{\frac{\mathbb{k}_{m,k}}{\mathbb{k}_{m,k}+1}}\bar{\Omega}_{m,k} + \sqrt{\frac{1}{\mathbb{k}_{m,k}+1}}\tilde{\Omega}_{m,k} \tag{4}$$

where $\bar{\Omega}_{m,k}$ and $\tilde{\Omega}_{m,k}$ are the random scattering components. $\bar{\Omega}_{m,k}$ denotes a circularly symmetric complex Gaussian random variable with zero mean and unit variance, *i.e.,* $\bar{\Omega}_{m,k} \sim \mathcal{CN}(0, 1)$.

$\bar{\Omega}_{m,k}$ owns the real part and the imaginary part independently and identically distributed, *i.e.,* following a normal distribution. $\tilde{\Omega}_{m,k}$ denotes a circularly symmetric complex Gaussian random variable with zero mean and zero variance. Additionally, the Rician factor of the air-to-ground (AtG) channel $\mathbb{k}_{m,k}$ varies with different heights of EUAV k, which is modeled by an exponential function, *i.e.,*

$$\mathbb{k}_{m,k}(\theta_{m,k}) = k_0 \cdot e^{\tilde{k}\theta_{m,k}}. \tag{5}$$

In (5), $\theta_{m,k}$ denotes an elevation angle of the AtG path, *i.e.,* $\theta_{m,k} = \arcsin(\tilde{h}/d_{m,k})$. $d_{m,k} = \sqrt{\tilde{h}^2 + \|\mathbf{u}_m - \mathbf{u}_k\|^2}$ is the Euclidean distance between MU m and EUAV k. k_0 and $\tilde{k}$ are the constants depending on the environment and frequency, respectively. $k_0 = \mathbb{k}_{m,k}(0)$ and $\tilde{k} = (2/\pi)\ln(\mathbb{k}_{m,k}(\pi/2)/\mathbb{k}_{m,k}(0))$, which are determined based on the measurements. The LoS link is stronger when $\theta_{m,k}$ increases and smaller $\theta_{m,k}$ incurs more multipath conditions.

In (3), $\xi_{m,k}$ represents the large-scale average channel power gain, which accounts for the signal attenuation caused by the path loss, which is calculated as

$$\xi_{m,k} = A d_{m,k}^{-\epsilon_{m,k}} \tag{6}$$

where A denotes the channel gain when the reference distance d_0 is 1 m, and it depends on the antenna characteristics and the average channel attenuation. $\epsilon_{m,k}$ denotes the path loss exponent, which is given as

$$\epsilon_{m,k} = \frac{u_3}{1 + u_1 e^{-u_2(\theta_{m,k} - u_1)}} + u_4 \tag{7}$$

where u_1 and u_2 denote the AtG parameters. u_3 and u_4 denote the parameters in the path loss exponent. Equation (7) reveals the LoS-dominant channel model in (3), and it comprehensively captures both the LoS and non-LoS propagation effect in the AtG channel.

Following the Shannon theorem, EUAV k transmits the cache files and downlink data to MU m with the transmission rate of $r_{k,m}^{\downarrow}$, which is given as

$$r_{k,m}^{\downarrow} = B_{m,k}\log_2\left(1 + \frac{p_{k,m}\|h_{m,k}\|^2}{\sigma^2 + \sum_{k'=1(k'\neq k)}^{K} p_{k,m}\|h_{m,k}\|^2}\right) \tag{8}$$

where $p_{k,m}$ is the transmission power from EUAV k to MU m. $B_{m,k}$ and σ^2 are the bandwidth and additive white Gaussian noise power between EUAV k and MU m, respectively.

MU m transmits offloaded tasks to EUAV k with the transmission rate of $r_{m,k}^{\uparrow}$, which is given as

$$r_{m,k}^{\uparrow} = B_{m,k}\log_2\left(1 + \frac{q_{m,k}\|h_{m,k}\|^2}{\sigma^2 + \sum_{m'=1(m'\neq m)}^{M} q_{m,k}\|h_{m,k}\|^2}\right) \tag{9}$$

where $q_{m,k}$ is transmission power from MU m to EUAV k.

$\hat{p}_k$ and $\hat{q}_m$ denote the upper limits of $p_{k,m}$ and $q_{m,k}$, respectively. Then

$$0 \le p_{k,m} \le \hat{p}_k \tag{10}$$

$$0 \le q_{m,k} \le \hat{q}_m. \tag{11}$$

2) Communication Between LUAV and EUAVs: The wireless communication path between EUAVs and LUAV is modeled as

$$h_{k,0} = \sqrt{\xi_{k,0}} \tag{12}$$

where $\xi_{k,0}$ represents a quasistatic block fading LoS link, which is calculated as

$$\xi_{k,0} = A d_{k,0}^{-\epsilon_{k,0}} \tag{13}$$

where $d_{k,0}$ ($d_{k,0} = \sqrt{(\tilde{h} - \bar{h})^2 + \|\mathbf{u}_k - \mathbf{u}_0\|^2}$) and $\epsilon_{k,0}$ are the distance and the path loss exponent between LUAVs and EUAV k, respectively.

Authorized licensed use limited to: University of Melbourne. Downloaded on September 26,2024 at 01:07:41 UTC from IEEE Xplore. Restrictions apply.

Similarly, LUAV transmits its cache files and the result of computation tasks to EUAV k with the transmission rate of $r_{0,k}^{\downarrow}$, which is given as

$$r_{0,k}^{\downarrow} = B_{k,0}\log_2\left(1+\frac{p_{0,k}\left\|h_{k,0}\right\|^2}{{\sigma_{k,0}}^2}\right) \tag{14}$$

where $p_{0,k}$ denotes the transmission power from LUAV to EUAV k. $B_{k,0}$ and ${\sigma_{k,0}}^2$ (${\sigma_{k,0}}^2 = N_0 B_{k,0}$) represent the bandwidth and the noise power between LUAV and EUAV k, respectively, and N_0 denotes the noise power spectrum density.

Similarly, considering the interference of other EUAVs, EUAV k transmits offloaded tasks of MUs to LUAV with the transmission rate of $r_{k,0}^{\uparrow}$, which is given as

$$r_{k,0}^{\uparrow} = B_{k,0}\log_2\left(1+\frac{q_{k,0}\left\|h_{k,0}\right\|^2}{\sigma^2+\sum_{k'=1(k'\neq k)}^{K} q_{k,0}\left\|h_{k,0}\right\|^2}\right) \tag{15}$$

where $q_{k,0}$ denotes the transmission power from EUAV k to LUAV. This work assumes the transmission power between EUAV and LUAV is constant.

B. Caching and Computing Models

In this work, each EUAV offers cached data to MUs. If the requested file is not cached, the cloud must transmit it to EUAVs and MUs. Let $\lambda_{m,k}$ denote a binary caching variable, which is 1 when the requested file of MU m is cached in EUAV k and 0 otherwise, *i.e.,*

$$\lambda_{m,k}\in\{0,1\}. \tag{16}$$

Let $\mu_{m,k}^{E}$ and $\mu_{m,k}^{L}$ denote the two binary offloading variables. $\mu_{m,k}^{E}$ is 1 when the computation task of MU m is offloaded to EUAV k and it is 0 otherwise. Similarly, $\mu_{m,k}^{E}$ is 1 when a computation task of MU m is offloaded to the cloud through EUAV k, and it is 0 otherwise. Thus

$$\mu_{m,k}^{E},\ \ \mu_{m,k}^{L}\in\{0,1\}. \tag{17}$$

For each MU, its task of choosing to be offloaded to EUAVs or the cloud is mutually exclusive. Furthermore, we assume that each MU can transmit through at most one EUAV, regardless it is executed in EUAVs or the cloud, *i.e.,*

$$\sum_{k=1}^{K}\left(\mu_{m,k}^{E}+\mu_{m,k}^{L}\right)\leq 1. \tag{18}$$

According to [34], there are two kinds of computation offloading modes, *i.e.,* static offloading and dynamic offloading. As shown in Fig. 2, applications can be prepartitioned into a local part in MUs and a remote one in EUAVs or the cloud.

When a computation task requested to be offloaded is static, it has to be executed in MUs or offloaded completely to EUAVs or the cloud, as illustrated in Fig. 2(a). For example, the FLUID application of Android is a typical example, whose thin client side needs to run in MUs, and the server side demanding high-performance GPU is always executed remotely in MEC. As illustrated in Fig. 2(b), when a computation task requested to be offloaded is dynamic, it can be executed in MUs or offloaded to MEC with an arbitrary split. The Linpack of Android is a typical example, which can be executed in MUs entirely or offloaded to MEC partially or completely.

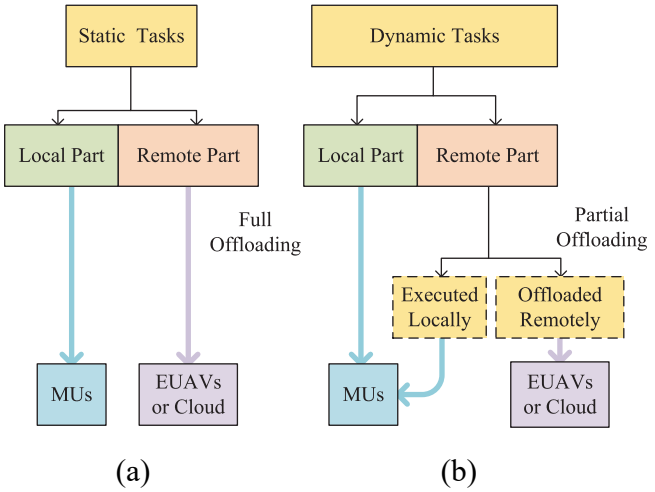

Fig. 2. Illustration of static and dynamic offloading tasks. (a) Static offloading. (b) Dynamic offloading.

We also introduce another binary variable ω_m, which is 1 when a static computation task of MU m requested to be offloaded, and it is 0 otherwise, *i.e.,*

$$\omega_m-\sum_{k=1}^{K}\left(\mu_{m,k}^{E}+\mu_{m,k}^{L}\right)\leq 0. \tag{19}$$

Besides, $a_{m,k}$ denotes an offloading ratio of MU m to EUAV k. When $\omega_m = 1$, MU m offloads all the tasks to EUAVs or the cloud, *i.e.,* $a_{m,k} = 1$. When $\omega_m = 0$, MU m decides the best amount of the offloaded task. Therefore, $a_{m,k}$ can take any value in [0,1], *i.e.,*

$$\begin{aligned} &a_{m,k}=1, \quad \omega_m=1\\ &a_{m,k}\in[0,1], \quad \omega_m=0. \end{aligned} \tag{20}$$

Additionally, D_m denotes the size of a task of MU m that needs to be executed and ζ_m denotes the number of its needed CPU cycles. This work assumes that the uplink data and the number of the CPU cycles required by the offloaded tasks are proportional to $a_{m,k}$.

C. Latency Model

MUs initiate a computation process only if they acquire the requested cache files from the cloud or EUAVs. C_m denotes the size of cache files requested by MU m. The cache capacity C_k of EUAV k cannot be exceeded by its size of stored files from MUs, *i.e.,*

$$\sum_{m=1}^{M}\lambda_{m,k}C_m\leq C_k. \tag{21}$$

$T_{m,k}^{c}$ denotes the latency of transmission of cache files from EUAV k to MU m, which is calculated as

$$T_{m,k}^{c}=\frac{C_m}{r_{k,m}^{\downarrow}}+\left(1-\lambda_{m,k}\right)\frac{C_m}{r_{0,k}^{\downarrow}}. \tag{22}$$

Authorized licensed use limited to: University of Melbourne. Downloaded on September 26,2024 at 01:07:41 UTC from IEEE Xplore. Restrictions apply.

T_m^l denotes the computation time of the local tasks in MU m. Then

$$T_m^l = \left(1 - a_{m,k}\right)\frac{\zeta_m}{f_m} \tag{23}$$

where f_m denotes the local computing ability of MU m.

$T_{m,k}^E$ denotes the computation time of the offloaded tasks in EUAV k from MU m, which is given as

$$T_{m,k}^E = \mu_{m,k}^E a_{m,k}\frac{\zeta_m}{f_{m,k}^E} \tag{24}$$

where $f_{m,k}^E$ denotes the computing ability of EUAV k.

EUAV k cannot handle tasks exceeding its computing capacity F_k, *i.e.,*

$$\sum_{m=1}^{M} \mu_{m,k}^E f_{m,k}^E \le F_k. \tag{25}$$

$T_{m,k}^L$ denotes the computation time of the offloaded tasks from MU m in the cloud, which is given as

$$T_{m,k}^L = \mu_{m,k}^L a_{m,k}\frac{\zeta_m}{f_{m,k}^L} \tag{26}$$

where $f_{m,k}^L$ denotes the allocated computing ability of the cloud during the offloading process.

The allocated computing resources of MUs, EUAVs, and the cloud have their upper limits $\hat{f}_m$, $\hat{f}^E$, and $\hat{f}^L$, *i.e.,*

$$0 \le f_m \le \hat{f}_m \tag{27}$$
$$0 \le f_{m,k}^E \le \hat{f}^E \tag{28}$$
$$0 \le f_{m,k}^L \le \hat{f}^L. \tag{29}$$

Let O_m denote the size of download data of MU m from EUAVs after the offloaded task execution on MEC. $T_{m,k}^t$ denotes the transmission latency of the whole offloading process of each task of MU m through the link of EUAV k, which is given as

$$T_{m,k}^t = \mu_{m,k}^E\left(\frac{a_{m,k}D_m}{r_{m,k}^{\uparrow}} + \frac{O_m}{r_{k,m}^{\downarrow}}\right) + \mu_{m,k}^L \left(\frac{a_{m,k}D_m}{r_{m,k}^{\uparrow}} + \frac{O_m}{r_{k,m}^{\downarrow}} + \frac{a_{m,k}D_m}{r_{k,0}^{\uparrow}} + \frac{O_m}{r_{0,k}^{\downarrow}}\right). \tag{30}$$

We consider the tasks computed locally in MU m, and ones offloaded remotely to MEC can be executed in parallel. Thus, the total latency T_m^{Δ} of executing MU m's tasks is given as

$$T_m^{\Delta} = \sum_{k=1}^{K} T_{m,k}^c + \max\left(T_m^l, \sum_{k=1}^{K}\left(T_{m,k}^E + T_{m,k}^L + T_{m,k}^t\right)\right). \tag{31}$$

The latency of each task of MU m cannot surpass its limit $\hat{T}_m$, *i.e.,*

$$T_m^{\Delta} \le \hat{T}_m. \tag{32}$$

D. Energy Consumption

The energy consumption of transmitting and executing tasks is much larger than that of UAVs' propelling and hovering. Consequently, we neglect UAVs' flying energy loss in this work. Then, the energy consumption of the whole system includes the caching file transmitting, the local computing, the transmission of the uplink and downlink data of offloaded tasks, and the computing of remotely offloaded tasks.

$\varepsilon_{m,k}^c$ denotes the energy consumption of transmitting caching files from EUAV k to MU m, *i.e.,*

$$\varepsilon_{m,k}^c = \frac{C_m}{r_{k,m}^{\downarrow}}p_{k,m} + \left(1 - \lambda_{m,k}\right)\frac{C_m}{r_{0,k}^{\downarrow}}p_{0,k}. \tag{33}$$

ε_m^l denotes the energy consumption of computing each task locally in MU m, *i.e.,*

$$\varepsilon_m^l = \kappa_m (f_m)^3 T_m^l \tag{34}$$

where κ_m denotes a capacitance coefficient of MU m, which depends on its chip architecture [35].

$\varepsilon_{m,k}^E$ and $\varepsilon_{m,k}^L$ denote the energy consumption of computing each offloaded task of MU m in EUAV k and in the cloud, respectively, which is calculated as

$$\varepsilon_{m,k}^E = T_{m,k}^E \kappa_k \left(s_1 + s_2 \left(f_{m,k}^E\right)^{\bar{e}}\right) \tag{35}$$
$$\varepsilon_{m,k}^L = T_{m,k}^L \kappa_0 \left(s_3 + s_4 \left(f_{m,k}^L\right)^{\bar{e}}\right) \tag{36}$$

where κ_k and κ_0 are both effective switched capacitance coefficients similar to κ_m. $\bar{e}$ denotes a capacitance coefficient of EUAVs and $\bar{e} \in (2.5, 3)$ following [36]. s_1, s_2, s_3, and s_4 are the four positive constants fitted by the offline power.

$\varepsilon_{m,k}^t$ denotes the transmission energy consumption of the whole offloading process of each task of MU m through the link of EUAV k, *i.e.,*

$$\varepsilon_{m,k}^t = \mu_{m,k}^E\left(\frac{a_{m,k}D_m}{r_{m,k}^{\uparrow}}q_{m,k} + \frac{O_m}{r_{k,m}^{\downarrow}}p_{k,m}\right) + \mu_{m,k}^L \left(\frac{a_{m,k}D_m}{r_{m,k}^{\uparrow}}q_{m,k} + \frac{O_m}{r_{k,m}^{\downarrow}}p_{k,m} + \frac{a_{m,k}D_m}{r_{k,0}^{\uparrow}}q_{k,0} + \frac{O_m}{r_{0,k}^{\downarrow}}p_{0,k}\right). \tag{37}$$

ε_m^{Δ} denotes the total energy consumption of executing each task of MU m, which is given as

$$\varepsilon_m^{\Delta} = \sum_{k=1}^{K}\varepsilon_{m,k}^c + \varepsilon_m^l + \sum_{k=1}^{K}\left(\varepsilon_{m,k}^E + \varepsilon_{m,k}^L + \varepsilon_{m,k}^t\right). \tag{38}$$

ε_m^{Δ} cannot exceed its limit $\hat{\varepsilon}_m$, *i.e.,*

$$\varepsilon_m^{\Delta} \le \hat{\varepsilon}_m. \tag{39}$$

E. Total Cost Optimization Problem

Application tasks of MUs with different services have varying preferences for energy consumption and latency, which are two major conflict objectives for current users. This work considers the Quality of Service (QoS) demands of MUs' tasks. For example, mobile applications with high latency requirements, such as online games, video calling, and voice recognition, prioritize T_m^{Δ}. Similarly, those sensitive to energy

Authorized licensed use limited to: University of Melbourne. Downloaded on September 26,2024 at 01:07:41 UTC from IEEE Xplore. Restrictions apply.

consumption, such as video streaming and navigation applications, prioritize ε_m^{Δ} [37]. For example, if the transmission power and computation speeds of MUs, EUAVs, and LUAV increase, the total energy consumption increases, and the latency of the tasks decreases, and vice versa.

To jointly optimize the total energy consumption of the system and the total latency of tasks, this work defines the total system cost as their weighted sum. Weight coefficients ρ_1^m and ρ_2^m are introduced to express their relative significance. They are determined based on the preferences of the MUs' tasks. Here, $\rho_1^m + \rho_2^m = 1 (0 < \rho_1^m, \rho_2^m < 1)$.

This work uses the QoS index $i (i = 1, 2)$ as MU m's preference ($i = 1$ if MU m's preference is T_m^{Δ}. Otherwise, $i = 2$). Each QoS index can be subdivided into Ψ preference levels. A higher level indicates a stronger preference and a more impact on MUs. Then, MU m's preference level for QoS index i is denoted as ψ_m^i ($\psi_m^i \in \{1, 2, \ldots, \Psi\}, i = \{1, 2\}$). To show the preference of the QoS parameter i over j of MU m, $\iota_m^{i,j} (\iota_m^{i,j} = \iota_m^i / \iota_m^j)$ is introduced. Thus, ρ_i^m can be given as

$$\rho_i^m = \frac{1}{2}\left(\frac{\iota_m^{i,1}}{\iota_m^{1,1} + \iota_m^{1,2}} + \frac{\iota_m^{i,2}}{\iota_m^{1,2} + \iota_m^{2,2}}\right). \tag{40}$$

Γ denotes the total cost of the system, which is given as

$$\Gamma = \sum_{m=1}^{M} \left(\rho_1^m \varepsilon_m^{\Delta} + \rho_2^m T_m^{\Delta}\right). \tag{41}$$

Then, the total cost minimization problem is formulated as

$$\underset{\mathbf{P},\mathbf{F},\mathbf{D},\mathbf{U},\mathbf{A}}{\mathbf{Min}} \{\Gamma\} \tag{42}$$

subject to (1), (2), (10), (11), (16)–(21), (25), (27)–(29), (32), and (39). Here, we have four types of decision variables, including transmission power allocation $\mathbf{P} = \{p_{k,m}, q_{m,k}\}$, computation capacities $\mathbf{F} = \{f_m, f_{m,k}^E, f_{m,k}^L\}$, binary variables $\mathbf{D} = \{\lambda_{m,k}, \mu_{m,k}^E, \mu_{m,k}^L\}$, EUAV coordinates $\mathbf{U} = \{\mathbf{u}_k\}$, and offloading ratio $\mathbf{A} = \{a_{m,k}\}$.

ε_m^{Δ} and T_m^{Δ} are both the nonlinear concerning integer and continuous variables. Thus, the optimization problem is a typical constrained MINLP with the NP-hard solution complexity [38]. It suffers from an exponential explosion issue, and no polynomial-time algorithms are available [39]. This work first adopts a penalty function method to convert the constrained MINLP into an unconstrained problem, *i.e.,*

$$\underset{\mathbf{P},\mathbf{F},\mathbf{D},\mathbf{U},\mathbf{A}}{\mathbf{Min}} \left\{\widetilde{\Gamma} = \overset{\infty}{\mathcal{N}} \mho + \Gamma\right\}. \tag{43}$$

In (43), $\widetilde{\Gamma}$ is a new objective function and $\overset{\infty}{\mathcal{N}}$ is a large positive number. $\mho$ is the total penalty of all the constraints, *i.e.,*

$$\mho = \sum_{\Theta=1}^{\mathbb{N}^{\neq}} (\max\{0, -\eth_\Theta(\boldsymbol{x})\})^{\gamma_1} + \sum_{\Lambda=1}^{\mathbb{N}^{=}} \left|\hbar_\Lambda(\boldsymbol{x})\right|^{\gamma_2} \tag{44}$$

where $\mathbb{N}^{\neq}$ is the number of inequality constraints and $\mathbb{N}^{=}$ is that of equality constraints. γ_1 and γ_2 are the two constants. Each inequality constraint Θ ($1 \le \Theta \le \mathbb{N}^{\neq}$) is converted into $\eth_\Theta(\boldsymbol{x}) \ge 0$ and its penalty is $(\max\{0, -\eth_\Theta(\boldsymbol{x})\})^{\gamma_1}$. Likewise, each equality constraint Λ ($1 \le \Lambda \le \mathbb{N}^{=}$) is converted into $\hbar_\Lambda(\boldsymbol{x}) = 0$ and its penalty is $|\hbar_\Lambda(\boldsymbol{x})|^{\gamma_2}$.

IV. Nonlinear Dynamic Adaptive Genetic Simulated Annealing-Based Particle Swarm Optimization

There are several deterministic algorithms, such as conjugate gradient descent and dynamic programming, to tackle the unconstrained problems. Nevertheless, they often need specific mathematical structures. For instance, some typically rely on the first-order or second-order derivatives, and their solution quality tends to be suboptimal for addressing such problems. Many studies opt for the typical meta-heuristic optimization algorithms due to their inherent strengths, including robustness, ease of implementation, ability to handle discontinuities and nonlinearities, and rapid convergence, avoiding shortcomings of the aforementioned deterministic approaches for complex real-world issues. Nevertheless, each optimization approach possesses its distinct set of strengths and weaknesses.

Traditional PSO algorithms typically rely on the individually best solutions and the globally best one to guide their evolution. Nevertheless, complex multimodal functions often lead the population to become ensnared in local optima. This work proposes a nonlinear dynamic adaptive inertial weight to help PSO yield a better balance between the linearity and nonlinearity of the sigmoid function. It considers evolutionary differences among the particles in the evolutionary process and adaptively updates inertia weight for global exploration. Moreover, this work integrates crossover and mutation operations of GA to enhance population diversity and coverage. This work adjusts the update strategy based on the Metropolis acceptance rule in SA to further mitigate the risk of converging toward local optima. Above all, this work designs an AGSP. AGSP adaptively assigns different inertia weights for global exploration and local search in different evolutionary periods. Fig. 3 shows a flowchart of AGSP. The details of its population initialization, genetic operations, and selection in AGSP are described next. Table S2 in the supplementary file summarizes the main notations of AGSP.

A. Population Initialization

$\mathcal{X}$ denotes the size of particles in the swarm. There are $\mathcal{N}$ elements in the position of each particle i ($i = 1, 2, \cdots, \mathcal{X}$). **P**, **F**, **D**, **U**, and **A** are stored in sequence, and $\widetilde{\Gamma}$ is kept in the final element. x_i and v_i denote the position and the velocity of the particle i, respectively, and their dimension is $\mathcal{N}$. ϱ_i denotes the locally best position of the particle i. ϱ^* denotes the globally best position of the whole swarm. Each particle modifies its velocity and position based on its learning experiences and that of the entire swarm. v_i and x_i are updated as

$$v_i = w^g v_i + c_1 \partial_1 (\varrho_i - x_i) + c_2 \partial_2 (\varrho^* - x_i) \tag{45}$$

$$x_i = x_i + v_i \tag{46}$$

where ∂_1 and ∂_2 are the random numbers in (0, 1). The inertia weight is denoted by w^g, which decreases as iterations with (49). c_1 and c_2 are the acceleration coefficients of each particle and the swarm, showing the impact of ϱ_i and ϱ^*, respectively.

Authorized licensed use limited to: University of Melbourne. Downloaded on September 26,2024 at 01:07:41 UTC from IEEE Xplore. Restrictions apply.

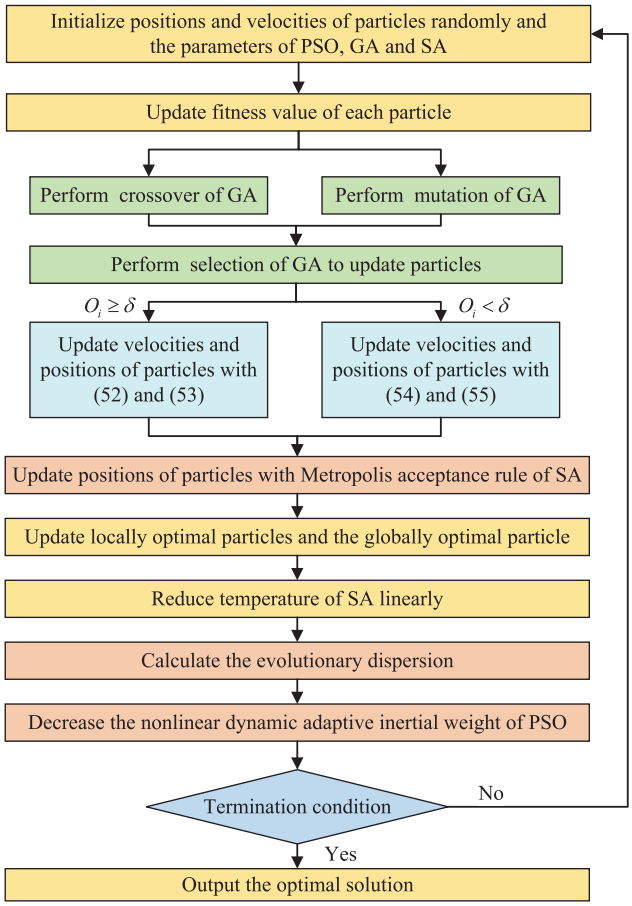

Fig. 3. Flowchart of AGSP.

B. Nonlinear Dynamic Adaptive Inertial Weight

At the beginning of iterations, the population includes more individuals with lower fitness values. The initial w is higher and the population inclines to explore the whole space. In the later evolution, the population includes more individuals with higher fitness values and the velocities of particles reduce rapidly. However, if w is always lower, premature convergence occurs, and local optima cannot be jumped out. Therefore, this work designs a dynamic adaptive inertia weight to support high exploration in the early evolution, thereby finding promising regions rapidly. Later evolution emphasizes exploiting promising regions to further converge toward the global optima quickly. The concept of evolutionary dispersion (ϑ^g) is given to describe changes in the population evolution. We define ϑ^g as the ratio of the standard deviation (Std) StdFit(g) of the fitness values of the population at the iterations g and $g-1$, *i.e.,*

$$\vartheta^g = \begin{cases} 1, & g = 1 \\ \frac{\text{StdFit}(g)}{\text{StdFit}(g-1)}, & g > 1. \end{cases} \tag{47}$$

The sigmoid function has a good balance between the linearity and nonlinearity, and therefore, it is used as the activation function, which is given as

$$S(x) = \frac{1}{1+e^{-x}}. \tag{48}$$

Thus, combining ϑ^g and the sigmoid function, a nonlinear dynamic and adaptive inertia weight, w^g, is given as

$$w^g = \hat{w} + (\check{w} - \hat{w}) \frac{1}{1 + \exp\left[-10b\left(\frac{2g}{\vartheta^g \cdot \hat{g}} - 1\right)\right]} \tag{49}$$

where $\hat{g}$ denotes the number of total iterations and b is a damping factor in [0, 1]. $\hat{w}$ and $\check{w}$ denote the upper and lower limits of w.

C. GA Operations

PSO's optimization process oscillates and converges quickly if the gap between the ϱ_i and ϱ^* is too large. Genetic operations in the GA produce superior particles for the global search ability of PSO. This work combines genetic operations into PSO to generate superior particles $y_i (i = 1, 2, \ldots, \mathcal{X})$. y_i is given as

$$y_i = \frac{c_1 \cdot \partial_1 \cdot \varrho_i + c_2 \cdot \partial_2 \cdot \varrho^*}{c_1 \cdot \partial_1 + c_2 \cdot \partial_2}. \tag{50}$$

This work also combines a mutation operation into PSO. ϱ_i and ϱ^* are encoded as a string of binary bits. The mutation possibility is denoted by ϖ. ϱ_i and ϱ^* generate their offspring z_i through a single-point crossover. Next, each bit of z_i is mutated with a possibility of ϖ, which avoids getting stuck in local optima in early evolution.

Then, we adopt a greedy criterion to choose whether y_i or z_i is chosen, *i.e.,*

$$x_i^g = \begin{cases} y_i, & \text{if } \widetilde{\Gamma}(y_i) \leq \widetilde{\Gamma}(z_i) \\ z_i, & \text{else.} \end{cases} \tag{51}$$

D. Adaptive Updating and SA-Based Selection

PSO adopts (45) and (46) to update the velocities and positions. Unlike PSO, AGSP adopts an improved adaptive velocity and position update strategy.

$\jmath_i$ denotes a ratio of the particle i's fitness value to the average one of the population, which is given as

$$\jmath_i = \frac{\exp\left(\widetilde{\Gamma}\left(x_i^g\right)\right)}{\exp\left(\frac{1}{\mathcal{X}} \sum_{i=1}^{\mathcal{X}} \widetilde{\Gamma}\left(x_i^g\right)\right)} \tag{52}$$

When $\jmath_i \geq \delta$, $\widetilde{\Gamma}(x_i^g)$ is much higher than the average fitness value of the population, showing the scattered distribution of particles. This work utilizes (53) and (54) to update the particles' velocities and positions. The individually best solutions and the globally best solution are combined linearly in the velocity update, improving the possibility of finding global optima. The position is updated with

$$v_i = w^g \cdot v_i + c_1 \cdot \partial_1 \left(\frac{\varrho_i + \varrho^*}{2} - x_i\right) + c_2 \partial_2 \left(\frac{\varrho_i - \varrho^*}{2} - x_i\right) \tag{53}$$

$$x_i^{g+1} = w^g x_i^g + (1 - w^g) v_i. \tag{54}$$

When $\jmath_i < \delta$, $\widetilde{\Gamma}(x_i^g)$ is not significantly different from the average fitness value of the population, indicating that the current particle distribution is concentrated. In such case, this work utilizes (55) and (56) to update the velocity and position

Authorized licensed use limited to: University of Melbourne. Downloaded on September 26,2024 at 01:07:41 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1 AGSP

1: Initialize x_i and v_i randomly
2: Initialize $\mathcal{X}$, ϖ, t^1, $\hat{t}$, c_1, c_2, b,$\hat{w}$, $\check{w}$ and $\hat{g}$
3: Update $\widetilde{\Gamma}$ of particles with (43)
4: Update ϱ_i and ϱ^*
5: $g \leftarrow 1$
6: **while** $g \leq \hat{g}$ **do**
7: Conduct single-point crossover of GA on ϱ_i and ϱ^* to generate y_i
8: Conduct mutation of GA on each bit of y_i with ϖ to generate z_i
9: Calculate the fitness values of y_i and z_i and adopt the selection of GA to choose the better one
10: **if** $\jmath_i \geq \delta$ **then**
11: Update v_i with (53)
12: Update x_i with (54)
13: **else**
14: Update v_i with (55)
15: Update x_i with (56)
16: **end if**
17: Update x_i^g with the Metropolis acceptance rule of SA
18: Update $\widetilde{\Gamma}$ of each particle by (43)
19: Update ϱ_i of each particle and ϱ^* of the swarm
20: $t^g \leftarrow \hat{t} \cdot t^g$
21: Calculate ϑ^g
22: $w \leftarrow \hat{w} + (\check{w} - \hat{w}) \frac{1}{1+\exp\left[-10b\left(\frac{2g}{\vartheta^g \cdot \hat{g}} - 1\right)\right]}$
23: $g \leftarrow g + 1$
24: **end while**
25: **return** ϱ^*

of each particle. Equation (55) is employed to jump out of local optima

$$v_i = w^g v_i + c_1 \partial_1 \left(\bar{\varrho}^g - x_i\right) + c_2 \partial_2 \left(\varrho^* - x_i\right) \tag{55}$$

$$x_i = x_i + v_i \tag{56}$$

where the average position ($\bar{\varrho}^g$) of particles in generation g is introduced to improve the convergence speed, which is given as

$$\bar{\varrho}^g = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} x_{i,n}^g \tag{57}$$

where $x_{i,n}^g$ denotes the value of x_i^g in dimension n.

In addition, an SA-based selection is adopted to update all the particles [40]. Particle i's position in iteration g is denoted by x_i^g. Then, x_{i+1}^g is updated according to (54) or (56).

If $\widetilde{\Gamma}(x_i^{g+1}) \leq \widetilde{\Gamma}(x_i^g)$, x_i^{g+1} is selected. Otherwise, it is conditionally selected if

$$e^{\left(\frac{\widetilde{\Gamma}\left(x_i^g\right) - \widetilde{\Gamma}\left(x_i^{g+1}\right)}{t^g}\right)} > \partial_3 \tag{58}$$

where ∂_3 is a random constant in (0,1) and t^g is the temperature in iteration g. t^1 denotes the starting temperature and $\hat{t}$ denotes the cooling rate of the temperature.

Algorithm 1 shows the pseudocodes of AGSP. Line 1 randomly initializes the particles' positions and velocities randomly. Line 2 initializes the parameters of AGSP, including $\mathcal{X}$, ϖ, t^1, $\hat{t}$, c_1, c_2, b, $\hat{w}$, $\check{w}$, and $\hat{g}$. Line 3 updates the fitness value ($\widetilde{\Gamma}$) of each particle i with (43). Line 4 updates the locally best position of the particle ϱ_i and the globally best position (ϱ^*) of the whole swarm. The loop stops while $g > \hat{g}$ in Line 6. Line 7 conducts the single-point crossover of GA on ϱ_i and ϱ^* to generate the superior particle y_i. Line 8 performs the mutation of GA on y_i's each bit with the mutation ratio ϖ to generate z_i. Line 9 calculates the fitness values of y_i and z_i to select the better one. Lines 11 and 12 update v_i and ϱ_i with (53) and (54) if $\jmath_i \geq \delta$. Lines 13 and 14 update v_i and ϱ_i with (55) and (56) if $\jmath_i > \delta$. Line 17 updates x_i^g with the Metropolis acceptance rule of SA. Line 18 updates $\widetilde{\Gamma}$ of the swarm with (43). Line 19 updates the locally best position of each particle and the globally best position of the swarm. Line 20 updates t^g by $\hat{t}$. Line 21 calculates ϑ^g. Line 22 reduces w from $\hat{w}$ to $\check{w}$ adaptively with (49). Line 25 returns ϱ^* and yields the final decision variables, including **P**, **F**, **D**, **U**, and **A**.

The time complexity analysis of AGSP is given here. In each iteration, its time complexity is $\mathcal{O}(\mathcal{X}\mathcal{N})$. The while loop is the primary computation overhead. Thus, the time complexity is $\mathcal{O}(\hat{g}\mathcal{X}\mathcal{N})$. As mentioned above, $\mathcal{N}$ stores **P**, **F**, **D**, **U**, **A**, and the fitness value. Thus, $\mathcal{N} = 11M + 1$. Then, the time complexity of AGSP is $\mathcal{O}(\hat{g}\mathcal{X}M)$.

TABLE I
PARAMETER SETTING OF SIMULATION

Notation	$B_{m,k}$	$B_{k,0}$	σ^2	$\epsilon_{k,0}$	N_0
Value	8 MHz	10 MHz	-110 dBm	3	-174 dbm/Hz

V. PERFORMANCE EVALUATION

A. Parameter Setting

This section presents numerical results to evaluate AGSP. Specifically, [0, 20] MUs are distributed randomly and uniformly within a square area of 1000 m×1000 m, *i.e.*, α_k and $\beta_k \in$ [0,1000]. EUAVs are deployed in this area and $\tilde{h} =$ 100 m. LUAV is fixed at [0, 0] m with a height of 150 m, *i.e.*, $\bar{h} = 150$ m. According to [41], $\mathbb{k}_{m,k}(0) = 5$ dB, $\mathbb{k}_{m,k}(\pi/2) =$ 15 dB, $A = -20$ dB, $u_1 = 0.136$ dB, $u_2 = 11.95$ dB, $u_3 = -1.5$, and $u_4 = 3.5$. Following [42], [43], [44], in the energy consumption model, $s_1 = 0.4$, $s_2 = 0.5$, $s_3 = 0.6$, $s_4 = 0.6$, $\kappa_m = \kappa_k = \kappa_0 = 10^{-28}$, and $\bar{e} = 2.8$. Besides, $p_{0,k} = 0.5$ W, $q_{k,0} = 1$ W, $\hat{p}_k = 0.5$ W, and $\hat{q}_m = 0.2$ W [45]. Furthermore, $\hat{f}_m = 10^9$, $\hat{f}^E = 4\times10^9$, $\hat{f}^L = 10^{10}$ [46], and $\rho_1 = \rho_2 = 0.5$. Table I gives the setting of other parameters. MUs run real-world application tasks listed in Table II, including static and dynamic offloading types [47].

According to [48] and [49], the AGSP's parameters are given as follows. Specifically, $\mathcal{X} = 100$, $c_1 = 0.5$, $c_2 = 0.5$, $\hat{w} = 0.95$, $\check{w} = 0.4$, $\hat{g} = 10^3$, $b = 0.5$, $\varpi = 0.01$, $\delta = 0.8$, $\hat{t} = 0.95$, and $t^1 = 10^8$. In addition, $\overset{\infty}{\mathcal{N}} = 10^{10}$, $\gamma_1 = 1$, and $\gamma_2 = 1.5$.

B. Results and Analysis

This work compares AGSP with its state-of-the-art peers, including PSO [50], SA-based PSO (SAP) [51], and

Authorized licensed use limited to: University of Melbourne. Downloaded on September 26,2024 at 01:07:41 UTC from IEEE Xplore. Restrictions apply.

TABLE II
CHARACTERISTICS OF REAL-WORLD APPLICATIONS

Application	Static offloading tasks			Dynamic offloading tasks		
	FACE	SPEECH	OBJECT	LINPACK	CPUBENCH	PI BENCH
ζ_m	12.3	15	44.6	50	3.36	130
D_m	62	243	73	10,240	80	10,240
O_m	60	50	50	120	80	200
$\hat{T}_m$	5	5.1	13	62.5	4.21	163

TABLE III
STATISTICAL RESULTS OF DIFFERENT ALGORITHMS OVER 30 INDEPENDENT EXECUTIONS

Algs.	Best	Worst	Mean	Std	Time
PSO	2.889306	1.192090	2.165607	0.354876	8.895063
SAP	2.908163	1.582767	2.119146	0.357336	9.025547
GLP	1.843931	1.180104	1.651803	0.116659	21.531495
GSP	1.136562	1.108736	1.122865	0.007066	21.429477
AGSP	**1.075234**	**1.043965**	**1.061926**	**0.006924**	21.301646

genetic-learning-PSO (GLP) [52]. The vanilla GSP without self-adaptive operations [53] for performing the ablation study, which emphasizes the impact of individual components (GA operations, Metropolis acceptance, and adaptive component) in AGSP. We independently run all the algorithms 30 times. PSO, SAP, GLP, and GSP have the same parameter setting as AGSP. The reasons for choosing them for comparison are given as follows.

1) *PSO:* It adopts a swarm of particles moving in the search space according to their velocities and the best positions. Yet, it often traps into local optima. PSO removes the adaptive component, GA operations, and Metropolis acceptance from AGSP.
2) *GLP:* It integrates PSO with GA to generate superior PSO exemplars and yield higher performance. GLP removes the adaptive component and Metropolis acceptance from AGSP.
3) *SAP:* SAP adaptively adjusts an inertia weight to overcome PSO's easy trapping into local optima by introducing the SA's global exploration. SAP removes the adaptive component and GA operations from AGSP.
4) *GSP:* GSP integrates GA's genetic operations and the Metropolis acceptance rule of SA to guide particles [54]. GSP removes the adaptive component from AGSP.

Table III shows the best, worst, mean, and Std results of the best fitness values for different algorithms over 30 independent executions with 1000 iterations when K =2 and M =5, and their execution times on average. It is shown that AGSP outperforms other algorithms in terms of global optimization ability and stability. Besides, the execution times of PSO and SAP are similar, and those of GLP, GSP, and AGSP are also close. We conduct the Wilcoxon sign-rank test on the execution time data from 30 independent experiments. The results have a significance level of 0.01, proving that PSO and SAP have the same time complexity as GLP, GSP, and AGSP. Thus, although the execution times of GLP, GSP, and AGSP are longer than those of PSO and SAP, they significantly improve their optimization capabilities. It is also shown that despite sharing the same time complexity as GLP and GSP, AGSP has a more robust global search ability.

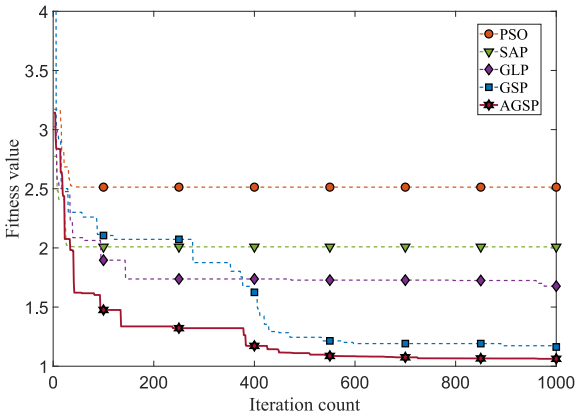

Fig. 4. Evolutionary curves of objective function values ($K = 2$ and $M = 5$).

Fig. 4 illustrates the convergence curves of s single independent execution. PSO and SAP obtain their best solutions with fewer iterations. However, PSO's final solution is poor, which shows PSO is easily trapped in local optima. SAP's final solution is better than PSO because it integrates the Metropolis acceptance rule to jump out of local optima and increase the search capability in the early stage. Yet, SAP's final solution is worse than those of GLP, GSP, and AGSP, which proves that GA's mutation, crossover, and selection operations improve the diversity of solutions in the high-dimensional space. Compared with PSO, GLP, and SAP, it is shown that GSP and AGSP which integrate genetic operations and the Metropolis acceptance rule achieve excellent performance. However, AGSP searches more quickly in the early stage than GSP because of its improved search strategy and adaptive update of weights. Consequently, AGSP's final fitness value is reduced by 136.5%, 88.99%, 57.76%, and 9.41%, respectively, compared with PSO, SAP, GLP, and GSP. Thus, AGSP finds the best solution with fewer iterations and a much shorter time.

Fig. 5(a) shows the total system cost concerning the number of MUs (M) given K =3. AGSP always yields the lowest total system cost among all the algorithms. As M increases, the total system cost rises. When $M \leq$10, the problem dimension is low, and the solutions of all the algorithms have no particular differences. Yet, when M>10, the problem dimension increases significantly and becomes more complicated. Specifically, AGSP achieves lower total system cost than PSO, SAP, GLP, and GSP by 50.99%, 50.13%, 58.51%, and 26.97% when M =20, respectively. Fig. 5(b) shows the impact of C_k on the total system cost. The total system cost of all the algorithms decreases as C_k increases because EUAVs cache more files and respond faster to the MUs' tasks. It is observed that when C_k =0, EUAVs do not possess caching capabilities, requiring

Authorized licensed use limited to: University of Melbourne. Downloaded on September 26,2024 at 01:07:41 UTC from IEEE Xplore. Restrictions apply.

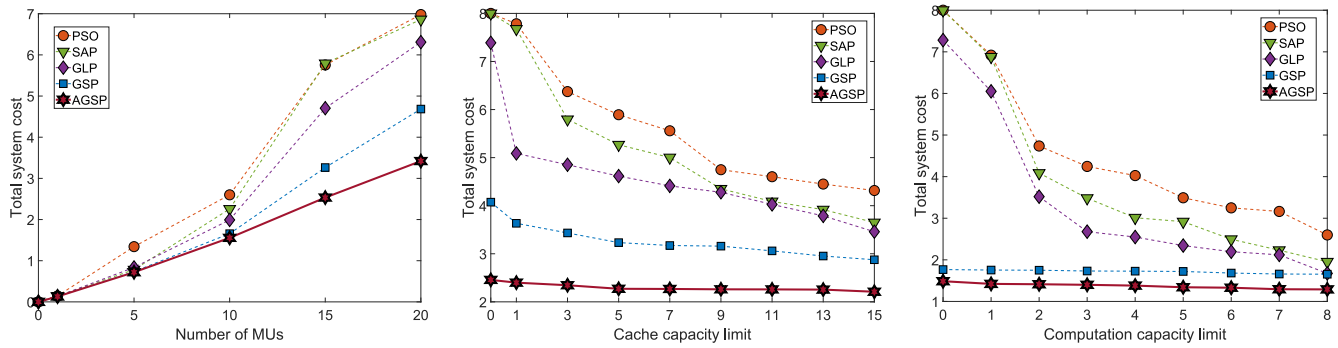

Fig. 5. Total system cost of different algorithms with different M, C_k, and F_k. (a) Total system cost versus M. (b) Total system cost versus C_k. (c) Total system cost versus F_k.

MUs to retrieve cached files from the cloud. This significantly increases the transmission latency, leading to an increase in the total system cost. In addition, it is proven that the caching decision variable significantly impacts the total system cost. When C_k is slight, many MU tasks cannot be cached, significantly penalizing the total system cost. As C_k increases, the gap between the AGSP and benchmark algorithms reduces because all the tasks are cached, and further increases in the cache capacity do not significantly affect the total system cost. Therefore, it is not economical to increase caching capability incessantly. Fig. 5(c) shows the impact of F_k on the total system cost. Similar to Fig. 5(b), as F_k increases, the total system cost decreases because higher edge capabilities lead to faster task execution. Furthermore, different from that in Fig. 5(b), the total system cost of all the algorithms (especially GSP and AGSP) in Fig. 5(c) gradually approaches specific values. This is because tasks can be executed only when their required cache files are downloaded to their MUs. As C_k increases, the transmission latency of cache files decreases. F_k is still limited, thereby leading to considerable execution latency. As F_k increases, the EUAV's computation capacity is improved significantly. Therefore, MUs prefer to offload tasks to EUAVs, which leads to lower latency and less energy consumption than the cloud.

To demonstrate the superiority of AGSP in the problem, this work compares it with the state-of-the-art strategies.

1) *A1:* UAV-enabled local and edge collaboration without the cloud. EUAVs serve as the edge servers with similar configurations in AGSP.
2) *A2:* Uniformly distributed EUAVs. EUAVs are evenly distributed in the area with the same height and remain fixed. A2 is achieved with AGSP, excluding the optimizing trajectories of EUAVs.

Fig. 6 shows that AGSP achieves the best strategy compared with A1 and A2 when M =12 and K =5. Specifically, the system total energy consumption with AGSP is reduced by 8.39% and 7.46%, respectively, and the average latency of MUs is reduced by 8.46% and 10.11%, respectively. This demonstrates that optimizing the trajectories of EUAVs and incorporating the cloud into the MEC system positively impacts saving energy consumption and reducing latency.

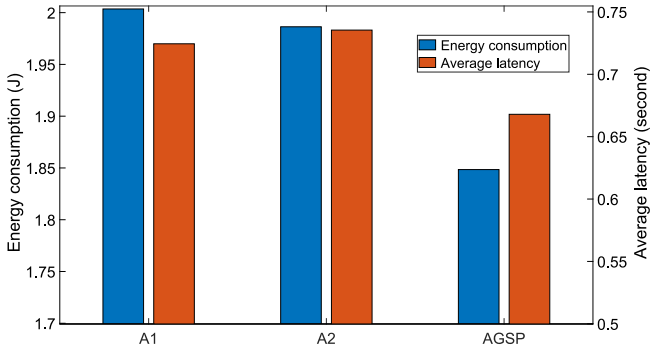

Fig. 6. Energy consumption and the average latency of MUs versus different strategies.

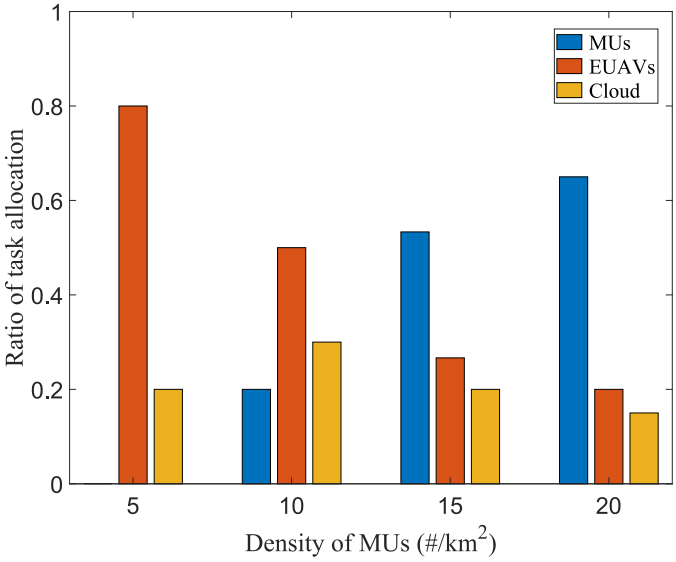

Fig. 7. Ratios of tasks executed in MUs, EUAVs, or the cloud versus MUs' density.

Fig. 7 illustrates the task allocation ratio under different densities. It is observed that the ratio of MUs increases as the density of MUs increases. This trend arises because the number of EUAVs and their computation capacities are limited. The computational capacities of EUAVs gradually

Authorized licensed use limited to: University of Melbourne. Downloaded on September 26,2024 at 01:07:41 UTC from IEEE Xplore. Restrictions apply.

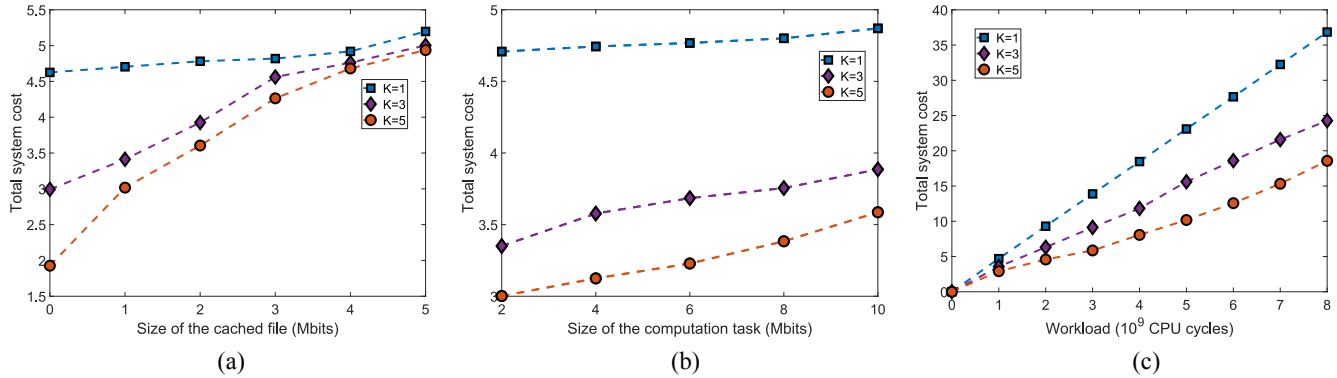

Fig. 8. Total system cost of different K with different C_m, D_m, and ζ_m. (a) Total system cost versus C_m. (b) Total system cost versus D_m. (c) Total system cost versus ζ_m.

become insufficient to meet MUs' increasing demand. Even though the cloud owns sufficient computation resources, it has to rely on EUAVs that serve as relays to transmit offloaded tasks. Thus, more MUs have to execute tasks by themselves. Additionally, more tasks are offloaded to EUAVs than to the cloud because EUAVs are closer to MUs.

Fig. 8 shows the impact of parameters, including the size of cached files C_m, the size of computational tasks D_m, and the workload ζ_m on the total system cost with M =10 and $K = \{1, 3, 5\}$. Fig. 8(a) shows that the total system cost increases with C_m because EUAVs easily handle tasks with smaller C_m. In contrast, tasks with larger C_m require computing resources in the cloud, resulting in increased total system cost. Fig. 8(b) shows that with fixed M and K, as D_m increases due to the increasing transmission latency between MUs and EUAVs, and that between EUAVs and the cloud, the total system cost also increases. Consequently, the computational capabilities of EUAVs and the cloud significantly reduce processing latency, which is much shorter than the transmission latency. Thus, all MUs choose local computing modes, and the total system cost gradually levels off at a stable value. Similarly, as shown in Fig. 8(c), as ζ_m increases, the total system cost also increases. When ζ_m is small, the local computation can meet the QoS requirements of MUs, avoiding long transmission latency among MUs, EUAVs, and the cloud. However, when ζ_m is large, local computing cannot meet the MUs' QoS requirements, necessitating assistance from more powerful EUAVs and the cloud. Furthermore, it is observed that with fixed K, the total system cost increases approximately linearly with ζ_m, consistent with the model in Section III. Furthermore, as shown in Fig. 8(a)–(c), it is shown that increasing K simultaneously improves the capabilities of data transmission, caching, and computation, thereby reducing the total system cost.

VI. Conclusion

This work proposes a layered UAV-assisted hybrid cloud-edge system. EUAVs are used as the MEC servers to collaborate with a remote cloud to provide resources to terrestrial MUs. Based on the architecture, a task offloading method among the MUs, EUAVs, and the cloud is investigated. This work minimizes the total system cost by jointly optimizing the associations between the MUs and EUAVs, transmission power and computing speeds of MUs, and offloading ratios of tasks of MUs. In addition, many real-life constraints, including user-specific latency of tasks of MUs, transmission power, computing speed limits of MUs, energy limits of MUs and EUAVs, and caching and computing capacities of EUAVs, are jointly considered. Then, a constrained MINLP is formulated, which is solved with a novel hybrid metaheuristic algorithm named AGSP. AGSP integrates the advantages of PSO, GA, and SA. Numerical results prove that compared with its four benchmark peers, AGSP reduces the total system energy consumption by at least 7.46% and the total latency of tasks by at least 8.46%, respectively.

The implicit and hidden features in yielded solutions in iterations in AGSP are not well learned with deep learning models in this work. In the future, we intend to improve AGSP by integrating more deep learning mechanisms, e.g., stacked autoencoders, to handle the higher-dimensional optimization problems, thereby realizing a more scalable system with more MUs and EUAVs in more complex scenarios.

References

[1] R. Luo, H. Jin, Q. He, S. Wu, and X. Xia, "Cost-effective edge server network design in mobile edge computing environment," *IEEE Trans. Sustain. Comput.*, vol. 7, no. 4, pp. 839–850, Oct.–Dec. 2022.

[2] L. Shi, Y. Ye, X. Chu, and G. Lu, "Computation energy efficiency maximization for a NOMA-Based WPT-MEC network," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10731–10744, Jul. 2021.

[3] J. Shi, Y. Zhou, Z. Li, Z. Zhao, Z. Chu, and P. Xiao, "Delay minimization for NOMA-mmW scheme-based MEC offloading," *IEEE Internet Things J.*, vol. 10, no. 3, pp. 2285–2296, Feb. 2023.

[4] Z. Wu, Z. Yang, C. Yang, J. Lin, Y. Liu, and X. Chen, "Joint deployment and trajectory optimization in UAV-assisted vehicular edge computing networks," *J. Commun. Netw.*, vol. 24, no. 1, pp. 47–58, Feb. 2022.

[5] X. Li, C. Zhang, R. Zhao, C. He, H. Zheng, and K. Wang, "Energy-effective offloading scheme in UAV-assisted C-RAN system," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 10821–10832, Jul. 2022.

[6] M. Huang, A. Liu, N. N. Xiong, and J. Wu, "A UAV-assisted ubiquitous trust communication system in 5G and beyond networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 11, pp. 3444–3458, Nov. 2021.

[7] J. Dandapat, N. Gupta, S. Agarwal, and S. Darshi, "Service duration maximization for continuous coverage in UAV-assisted communication system," *IEEE Commun. Lett.*, vol. 26, no. 10, pp. 2445–2449, Oct. 2022.

[8] L. Sun, L. Wan, J. Wang, L. Lin, and M. Gen, "Joint resource scheduling for UAV-enabled mobile edge computing system in Internet of Vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 12, pp. 15624–15632, Dec. 2023.

Authorized licensed use limited to: University of Melbourne. Downloaded on September 26,2024 at 01:07:41 UTC from IEEE Xplore. Restrictions apply.

[9] D. Zhai, C. Wang, R. Zhang, H. Cao, and F. R. Yu, "Energy-saving deployment optimization and resource management for UAV-assisted wireless sensor networks with NOMA," *IEEE Trans. Veh. Technol.*, vol. 71, no. 6, pp. 6609–6623, Jun. 2022.

[10] D. Lakew, W. Na, N. Dao, and S. Cho, "Aerial energy orchestration for heterogeneous UAV-assisted wireless communications," *IEEE Syst. J.*, vol. 16, no. 2, pp. 2483–2494, Jun. 2022.

[11] A. Hazarika and M. Rahmati, "A framework for information freshness analysis in UAV-based sensing and communications," in *Proc. Wireless Telecommun. Symp.*, Pomona, CA, USA, 2022, pp. 1–7.

[12] F. A. Khan, H. N. Qureshi, A. Imran, and H. Refai, "Handover probability analysis in multi-tier aerial networks at varying altitudes," in *Proc. IEEE Int. Conf. Commun.*, Rome, Italy, 2023, pp. 2890–2895.

[13] H. Mei, K. Yang, Q. Liu, and K. Wang, "Joint trajectory-resource optimization in UAV-enabled edge-cloud system with virtualized mobile clone," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5906–5921, Jul. 2020.

[14] X. Hu, K.-K. Wong, K. Yang, and Z. Zheng, "UAV-assisted relaying and edge computing: Scheduling and trajectory optimization," *IEEE Trans. Wireless Commun.*, vol. 18, no. 10, pp. 4738–4752, Oct. 2019.

[15] A. A. Nasir, "Latency optimization of UAV-enabled MEC system for virtual reality applications under Rician fading channels," *IEEE Wireless Commun. Lett.*, vol. 10, no. 8, pp. 1633–1637, Aug. 2021.

[16] Z. Yang, S. Bi, and Y. Zhang, "Online trajectory and resource optimization for stochastic UAV-enabled MEC systems," *IEEE Trans. Wireless Commun.*, vol. 21, no. 7, pp. 5629–5643, Jul. 2022.

[17] A. Merwaday, A. Tuncer, A. Kumbhar, and I. Guvenc, "Improved throughput coverage in natural disasters: Unmanned aerial base stations for public-safety communications," *IEEE Veh. Technol. Mag.*, vol. 11, no. 4, pp. 53–60, Dec. 2016.

[18] Q. Wang, Z. Chen, W. Mei, and J. Fang, "Improving physical layer security using UAV-enabled mobile relaying," *IEEE Wireless Commun. Lett.*, vol. 6, no. 3, pp. 310–313, Jun. 2017.

[19] M. Chen, M. Mozaffari, W. Saad, C. Yin, M. Debbah, and C. S. Hong, "Caching in the sky: Proactive deployment of cache-enabled unmanned aerial vehicles for optimized quality-of-experience," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 5, pp. 1046–1061, May 2017.

[20] M. Hua, L. Yang, Q. Wu, and A. L. Swindlehurst, "3D UAV trajectory and communication design for simultaneous uplink and downlink transmission," *IEEE Trans. Commun.*, vol. 68, no. 9, pp. 5908–5923, Sep. 2020.

[21] C. Shen, T.-H. Chang, J. Gong, Y. Zeng, and R. Zhang, "Multi-UAV interference coordination via joint trajectory and power control," *IEEE Trans. Signal Process.*, vol. 68, pp. 843–858, 2020.

[22] M. M. Azari, F. Rosas, and S. Pollin, "Cellular connectivity for UAVs: Network modeling, performance analysis, and design guidelines," *IEEE Trans. Wireless Commun.*, vol. 18, no. 7, pp. 3366–3381, Jul. 2019.

[23] N. Cheng et al., "Air-ground integrated mobile edge networks: Architecture, challenges, and opportunities," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 26–32, Aug. 2018.

[24] S. Jeong, O. Simeone, and J. Kang, "Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 3, pp. 2049–2063, Mar. 2018.

[25] J. Li, C. Yi, J. Chen, K. Zhu, and J. Cai, "Joint trajectory planning, application placement, and energy renewal for UAV-assisted MEC: A triple-learner-based approach," *IEEE Internet Things J.*, vol. 10, no. 15, pp. 13622–13636, Aug. 2023.

[26] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, "Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 1927–1941, Sep. 2018.

[27] Z. Han, T. Zhou, T. Xu, and H. Hu, "Joint user association and deployment optimization for delay-minimized UAV-aided MEC networks," *IEEE Wireless Commun. Lett.*, vol. 12, no. 10, pp. 1791–1795, Oct. 2023.

[28] J. Lyu, Y. Zeng, and R. Zhang, "UAV-aided offloading for cellular hotspot," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 3988–4001, Jun. 2018.

[29] X. Jiang, Z. Wu, Z. Yin, W. Yang, and Z. Yang, "Trajectory and communication design for UAV-relayed wireless networks," *IEEE Wireless Commun. Lett.*, vol. 8, no. 6, pp. 1600–1603, Dec. 2019.

[30] R. Han, Y. Wen, L. Bai, J. Liu, and J. Choi, "Rate splitting on mobile edge computing for UAV-aided IoT systems," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 4, pp. 1193–1203, Dec. 2020.

[31] Y. Zeng, R. Zhang, and T. J. Lim, "Throughput maximization for UAV-enabled mobile relaying systems," *IEEE Trans. Commun.*, vol. 64, no. 12, pp. 4983–4996, Dec. 2016.

[32] X. Fu, G. Fortino, P. Pace, G. Alo, and W. Li, "Environment-fusion multipath routing protocol for wireless sensor networks," *Inf. Fusion*, vol. 53, pp. 4–19, Jan. 2020.

[33] M. Ren, X. Fu, P. Pace, G. Alo, and G. Fortino, "Collaborative data acquisition for UAV-aided IoTs based on time-balancing scheduling," *IEEE Internet Things J.*, vol. 11, no. 8, pp. 13660–13676, Apr. 2024, doi: 10.1109/JIOT.2023.3339136.

[34] J. Bi, K. Zhang, H. Yuan, and J. Zhang, "Energy-efficient computation offloading for static and dynamic applications in hybrid mobile edge cloud system," *IEEE Trans. Sustain. Comput.*, vol. 8, no. 2, pp. 232–244, Jun. 2023.

[35] Q. Jiang, V. C. M. Leung, H. Tang, and H. S. Xi, "Adaptive scheduling of stochastic task sequence for energy-efficient mobile cloud computing," *IEEE Syst. J.*, vol. 13, no. 3, pp. 3022–3025, Sep. 2019.

[36] X. Jiao et al., "Deep reinforcement learning empowers wireless powered mobile edge computing: Towards energy-aware online offloading," *IEEE Trans. Commun.*, vol. 71, no. 9, pp. 5214–5227, Sep. 2023.

[37] Q. Xu, J. C. Davis, Y. C. Hu, and A. Jindal, "An empirical study on the impact of deep parameters on mobile app energy usage," in *Proc. IEEE Int. Conf. Softw. Anal., Evol. Reeng.*, Honolulu, HI, USA, 2022, pp. 844–855.

[38] F. Boukouvala, R. Misener, and C. A. Floudas, "Global optimization advances in mixed-integer nonlinear programming, MINLP, and constrained derivative-free optimization, CDFO," *Eur. J. Oper. Res.*, vol. 252, no. 3, pp. 701–727, Aug. 2016.

[39] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan, "Mixed-integer nonlinear optimization," *Acta Numerica*, vol. 22, pp. 1–131, Apr. 2013.

[40] Y. Koshka and M. A. Novotny, "Comparison of D-Wave quantum annealing and classical simulated annealing for local minima determination," *IEEE J. Sel. Areas Inf. Theory*, vol. 1, no. 2, pp. 515–525, Aug. 2020.

[41] Y. Liu, K. Xiong, Y. Lu, Q. Ni, P. Fan, and K. B. Letaief, "UAV-aided wireless power transfer and data collection in Rician fading," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 10, pp. 3097–3113, Oct. 2021.

[42] L. Rao, X. Liu, M. D. Ilic, and J. Liu, "Distributed coordination of Internet data centers under multiregional electricity markets," *Proc. IEEE*, vol. 100, no. 1, pp. 269–282, Jan. 2012.

[43] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, San Francisco, CA, USA, 2016, pp. 1–9.

[44] N. Cheng et al., "Space/aerial-assisted computing offloading for IoT applications: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, May 2019.

[45] B. Liu, C. Liu, and M. Peng, "Computation offloading and resource allocation in unmanned aerial vehicle networks," *IEEE Trans. Veh. Technol.*, vol. 72, no. 4, pp. 4981–4995, Apr. 2023.

[46] Y. Choi, S. Park, S. Jeon, R. Ha, and H. Cha, "Optimizing energy consumption of mobile games," *IEEE Trans. Mobile Comput.*, vol. 21, no. 10, pp. 3744–3756, Oct. 2022.

[47] H. Mazouzi, N. Achir, and K. Boussetta, "DM2-ECOP: An efficient computation offloading policy for multi-user multi-cloudlet mobile edge computing environment," *ACM Trans. Internet Technol.*, vol. 19, no. 2, pp. 1–24, Apr. 2019.

[48] J. Bi, H. Yuan, S. Duanmu, M. Zhou, and A. Abusorrah, "Energy optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3774–3785, Mar. 2021.

[49] J. Bi, H. Yuan, J. Zhai, M. Zhou, and H. V. Poor, "Self-adaptive bat algorithm with genetic operations," *IEEE/CAA J. Automatica Sinica*, vol. 9, no. 7, pp. 1284–1294, Jul. 2022.

[50] D. Wu, N. Jiang, W. Du, K. Tang, and X. Cao, "Particle swarm optimization with moving particles on scale-free networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 1, pp. 497–506, Mar. 2020.

[51] Q. Li and W. Wang, "AVO inversion in orthotropic media based on SA-PSO," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 10, pp. 8903–8912, Oct. 2021.

[52] H. Zhang, Y. Zhang, D. Jiang, J. Leng, Z. Zhang, and X. Peng, "Modeling method for wind farm based on equivalent dynamic response," in *Proc. 3rd Int. Conf. Energy Eng. Power Syst. (EEPS)*, Dali, China, 2023, pp. 377–381.

[53] S. Gao, M. Zhou, Y. Wang, J. Cheng, H. Yachi, and J. Wang, "Dendritic neuron model with effective learning algorithms for classification, approximation, and prediction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 2, pp. 601–614, Feb. 2019.

Authorized licensed use limited to: University of Melbourne. Downloaded on September 26,2024 at 01:07:41 UTC from IEEE Xplore. Restrictions apply.

[54] H. Yuan, Q. Hu, M. Wang, J. Bi, and M. Zhou, "Cost-minimized user association and partial offloading for dependent tasks in hybrid cloud–edge systems," in *Proc. IEEE 18th Int. Conf. Autom. Sci. Eng. (CASE)*, Mexico City, Mexico, 2022, pp. 1059–1064.

Haitao Yuan (Senior Member, IEEE) received the Ph.D. degree in computer engineering from New Jersey Institute of Technology, Newark, NJ, USA in 2020.

He is currently an Associate Professor with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. His research interests include cloud computing, edge computing, data centers, big data, machine learning, deep learning, and optimization algorithms.

Dr. Yuan 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 is named in the world's top 2% of Scientists List. He serves as an Associate Editor for *Expert Systems with Applications*.

Meijia Wang (Student Member, IEEE) received the B.S. degree in automation from Beihang University, Beijing, China, in 2022, where she is currently pursuing the master's degree with the School of Automation Science and Electrical Engineering.

Her research interests include cloud computing, edge computing, energy management, big data, machine learning, and deep learning.

Jing Bi (Senior Member, IEEE) received the 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 Postdoctoral Researcher with the Department of Automation, Tsinghua University, 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 of Information Technology with the School of Software Engineering, Beijing University of Technology, Beijing. 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 currently an Associate Editor of IEEE TRANSACTIONS ON SYSTEMS MAN AND CYBERNETICS: SYSTEMS.

Shuyuan Shi received the Ph.D. degree in electrical and computer engineering from the National University of Singapore, Singapore, in 2020.

She is currently a Full Professor with the Fert Beijing Institute, MIIT Key Laboratory of Spintronics, and the School of Integrated Circuit Science and Engineering, Beihang University, Beijing, China. Her research interests include novel spintronic materials, devices, and in-memory computing applications.

Prof. Shi was awarded the Institute of Microelectronics Prize 2019 in Singapore. She has been recognized as an Excellent Young Scientist (Overseas) in 2022. She serves as an Editorial Board Member for Moore and More.

Jinhong Yang received the Ph.D. degree in computer application technology from Harbin Engineering University, Harbin, China, in 2017.

She is currently a Senior Engineer with the CSSC Systems Engineering Research Institute, Beijing, China. Her research interests include machine learning, data mining, knowledge reasoning, deep learning, and intelligent optimization.

Dr. Yang serves as a reviewer for *Expert Systems With Applications*, *International Journal of Machine Learning*, and *Cybernetics*.

Jia Zhang (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Illinois at Chicago, Chicago, IL, USA, in 2000.

She is currently the Cruse C. and Marjorie F. Calahan Centennial Chair in Engineering and a Professor with the Department of Computer Science, Lyle School of Engineering, Southern Methodist University, Dallas, TX, USA. 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 graphs, and interdisciplinary applications of all of these interests in earth science.

MengChu Zhou (Fellow, IEEE) received the Ph.D. degree from Rensselaer Polytechnic Institute, Troy, NY, USA, in 1990.

He joined with New Jersey Institute of Technology, Newark, NJ, USA, where he is currently a Distinguished Professor. He has over 900 publications, including 12 books, 600+ journal papers (450+ in IEEE Transactions), 28 patents, and 29 book chapters. His interests are in Petri nets, automation, Internet of Things, and big data.

Dr. Zhou is a Fellow of IFAC, AAAS, CAA, and NAI.

Rajkumar Buyya (Fellow, IEEE) received the B.E. degree in computer science and engineering from University of Mysore, Mysuru, India, in 1992, the M.E. degree in computer science and engineering from Bangalore University, Bengaluru, India, in 1995, and the Ph.D. degree in computer science and software engineering from Monash University, Melbourne, VIC, Australia, in 2002.

He is a Redmond Barry Distinguished Professor and the Director of the Cloud Computing and Distributed Systems Laboratory, University of Melbourne, Melbourne. He has authored over 800 publications and seven textbooks.

Dr. Buyya was recognized as a "Web of Science Highly Cited Researcher" from 2016 to 2021 by Thomson Reuters. He is one of the Highly Cited Authors in Computer Science and Software Engineering worldwide, with over 146 470 citations and an H-index of 166. He was a Future Fellow of the Australian Research Council from 2012 to 2016.

Authorized licensed use limited to: University of Melbourne. Downloaded on September 26,2024 at 01:07:41 UTC from IEEE Xplore. Restrictions apply.