mathematics

MDPI

Article

Joint UAV Deployment and Task Offloading in Large-Scale UAV-Assisted MEC: A Multiobjective Evolutionary Algorithm

Qijie Qiu [1], Lingjie Li [2,*], Zhijiao Xiao [1,*], Yuhong Feng [1], Qiuzhen Lin [1] and Zhong Ming [1,2,3,*]

[1] College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China; qiuqijie2021@email.szu.edu.cn (Q.Q.); yuhongf@szu.edu.cn (Y.F.); qiuzhlin@szu.edu.cn (Q.L.)
[2] Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen 518000, China
[3] College of Big Data and Internet, Shenzhen Technology University, Shenzhen 518118, China
* Correspondence: lilingjie2017@email.szu.edu.cn (L.L.); cindyxzj@szu.edu.cn (Z.X.); mingz@szu.edu.cn (Z.M.)

Abstract: With the development of digital economy technologies, mobile edge computing (MEC) has emerged as a promising computing paradigm that provides mobile devices with closer edge computing resources. Because of high mobility, unmanned aerial vehicles (UAVs) have been extensively utilized to augment MEC to improve scalability and adaptability. However, with more UAVs or mobile devices, the search space grows exponentially, leading to the curse of dimensionality. This paper focus on the combined challenges of the deployment of UAVs and the task of offloading mobile devices in a large-scale UAV-assisted MEC. Specifically, the joint UAV deployment and task offloading problem is first modeled as a large-scale multiobjective optimization problem with the purpose of minimizing energy consumption while improving user satisfaction. Then, a large-scale UAV deployment and task offloading multiobjective optimization method based on the evolutionary algorithm, called LDOMO, is designed to address the above formulated problem. In LDOMO, a CSO-based evolutionary strategy and a MLP-based evolutionary strategy are proposed to explore solution spaces with different features for accelerating convergence and maintaining the diversity of the population, and two local search optimizers are designed to improve the quality of the solution. Finally, simulation results show that our proposed LDOMO outperforms several representative multiobjective evolutionary algorithms.

Keywords: mobile edge computing; unmanned aerial vehicle; deployment optimization; task offloading; evolutionary algorithm

MSC: 68T20

check for updates

Citation: Qiu, Q.; Li, L.; Xiao, Z.; Feng, Y.; Lin, Q.; Ming, Z. Joint UAV Deployment and Task Offloading in Large-Scale UAV-Assisted MEC: A Multiobjective Evolutionary Algorithm. *Mathematics* **2024**, *12*, 1966. https://doi.org/10.3390/math12131966

Academic Editors: Zheyi Chen, Zhengxin Yu and Wang Miao

Received: 22 May 2024
Revised: 20 June 2024
Accepted: 21 June 2024
Published: 25 June 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With advances in Internet of Things (IoT) technology, numerous digital economy applications with compute-sensitive and latency-sensitive characteristics, such as augmented reality (AR), virtual reality (VR), and autonomous driving, have emerged, significantly enhancing the user experience [1,2]. Traditional cloud computing, which relies on long-distance communication with centralized data centers, fails to meet the stringent requirements of these applications. Therefore, mobile edge computing (MEC) has emerged as an effective complementary paradigm [3]. By enabling mobile devices to offload tasks to edge base stations, MEC offers reduced latency and energy consumption, thereby improving task completion efficiency [4]. This capability positions MEC as a viable solution for executing compute-intensive and latency-critical tasks, thereby enhancing overall quality of service (QoS) for digital economy applications [5–7]. Nevertheless, the location of the edge base station in MEC is usually fixed and cannot be flexibly changed according to the dynamic needs of mobile devices, which limits the adaptability and responsiveness of MEC.

To address these above limitations, unmanned aerial vehicles (UAVs) have emerged as a promising solution to augment MEC, in both military and civilian applications [8–10].

The dynamic integration of UAC and MEC is particularly evident in scenarios such as natural disaster relief, field operations, and large-scale competition events, where UAVs act as communication relay stations or efficient edge computing platforms [11]. Unlike fixed base stations, UAV-assisted MEC leverages the high mobility and adaptability of UAVs to establish superior line-of-sight links, thereby reducing transmission distances, enhancing communication reliability, and conserving the battery life of mobile devices [12]. For example, in wildfire rescue operations, UAVs with MEC capabilities can collect and process real-time data, establish communication relays where terrestrial networks fail, and optimize resource allocation to enhance rescue coordination. Consequently, UAV-assisted MEC offers a transformative solution for improving quality of experience (QoE) across various dynamic and challenging environments.

In the realm of UAV-assisted MEC, optimizing the deployment of UAVs and the task offloading of mobile devices is essential for achieving an optimal system performance. However, existing studies primarily focus on small-scale scenarios involving single- or multi-UAV assisted MEC [13–15]. In order to obtain efficient optimization strategies, these studies predominantly rely on evolutionary algorithms to solve resource optimization problems in single- or multi-UAV-assisted MEC, such as genetic algorithm (GA) [16,17], particle swarm optimization (PSO) [18,19], and ant colony optimizer (ACO) [20,21]. However, when the number of UAVs or mobile devices increases, the search space exponentially expands as the dimensionality of the decision variables increases, which is known as the curse of dimensionality [22,23]. Therefore, for large-scale UAV-assisted MEC, existing methods encounter considerable difficulties when obtaining convergent optimization strategies.

To address the challenge of efficient UAV deployment and task offloading in a large-scale UAV-assisted MEC, this paper introduces a **l**arge-scale UAV **d**eployment and task **o**ffloading **m**ultiobjective **o**ptimization method based on the evolutionary algorithm, namely LDOMO. Specifically, LDOMO first competitively divides the entire solution space into an elite solution space with well-performing individuals and a poor solution space with bad-performing individuals, which provides important labels and guidance for population evolution. Then, the competitive swarm optimizer (CSO)-based evolutionary strategy and the multilayer perceptron (MLP)-based evolutionary strategy are proposed in LDOMO to explore the two solution spaces separately. The former accelerates the convergence of the population by guiding the poor solutions closer to the elite solutions, while the latter guides the elite solutions into more promising areas through the evolutionary directions learned by MLP to maintain the diversity of the population. Finally, to address feasibility issues and elevate solution quality, LDOMO incorporates two local search optimizers: one for UAV deployment to prevent collisions by maintaining safe distances, and another for task offloading to ensure the accomplishment of all tasks.

The main contributions of this paper are clarified as follows.

1. This paper focuses on UAV deployment and task offloading optimization in a large-scale UAV-assisted MEC and formulates it as a large-scale multiobjective optimization problem, consisting of two optimization objectives (i.e., energy consumption and user satisfaction) with several constraints.
2. This paper proposes an efficient multiobjective optimization evolutionary algorithm, called LDOMO, to address the above-formulated problem. Specifically, LDOMO proposes two evolutionary strategies to explore two solution spaces with different characteristics for accelerating convergence and maintaining the diversity of population, and and designs two local search optimizers to improve the quality of the solution.
3. Extensive simulation results show that LDOMO outperforms several representative multiobjective optimization algorithms in terms of energy consumption and user satisfaction.

The rest of the paper is organized as follows. Section 2 reviews the background of large-scale multiobjective optimization and related work. Section 3 introduces several mathematical models and the problem formulation. Section 4 presents the details of

the proposed algorithm. Section 5 provides simulation results and discussions. Finally, Section 6 provides conclusions and future work.

2. Background and Related Work

2.1. Large-Scale Multiobjective Optimization

In the realm of large-scale multiobjective optimization (LMOP), the goal is to concurrently minimize or maximize multiple objective functions that often contradict each other. Mathematically, an LMOP can be mathematically modeled as follows:

$$\begin{aligned} \min_{x\in\Omega} \quad & F(x) = (f_1(x), f_2(x), \ldots, f_m(x))^T, \\ \text{s.t.} \quad & g_i(x) \leq 0, \quad \forall i \in \Theta, \end{aligned} \tag{1}$$

where the objective vector $F(x)$ comprises m conflicting objective functions $f_i(x)$ and $x = (x_1, x_2, \ldots, x_n)$ symbolizes an n-dimensional decision vector within the search space Ω ($m \geq 2$ and $n \geq 1000$). In order to balance multiple conflicting optimization objectives, the goal of optimizing an LMOP is to obtain a set of Pareto-optimal solutions (PS) in its search space, and the mapping of PS in the objective space is called the Pareto frontier (PF), which is described as follows.

If two solutions a and b exist in the search space Ω, a dominates b ($a \prec b$) if $f_i(a) \leq f_i(b)$ for all i and $f_j(a) < f_j(b)$ for at least one j. A solution $x^* \in \Omega$ is non-dominated when no other solution in Ω dominates it. The set of non-dominated solutions constitutes the PS, and the corresponding objective values form the PF:

$$\begin{aligned} PS =& \{\forall x \in \Omega \mid \nexists y \in \Omega, y \prec x\}, \\ PF =& \{F(x^*), x^* \in PS\}. \end{aligned} \tag{2}$$

2.2. Related Work and Our Motivations

This section reviews the prior works most relevant to our work from two aspects: (1) task offloading in MEC and (2) UAV-assisted MEC networks, and clarifies the motivation of our work.

Task offloading in MEC: In recent years, task offloading in MEC has attracted great attention, as it can reduce the delay of computation-intensive tasks and energy consumption to effectively improve the system performance [24,25]. For example, Mei et al. [26] focused on the task offloading problem for MEC systems with multiple energy harvesting devices aiming to increase the throughput and extend the battery life of devices. They applied Liapunov optimization theory to design a computational task offloading algorithm to accommodate system dynamics and ensure system stability. Ren et al. [27] studied the task offloading problem in the scenario of multiple edge service providers, and proposed an ADMM-based internal load balancing algorithm and a dynamic two-stage non-cooperative game to reduce the delays of ESPs internally and externally, respectively. Deng et al. [28] investigated edge collaboration task offloading and splitting strategies in MEC networks, i.e., splitting tasks to be executed in parallel on multiple MEC servers aims to minimize the total cost of latency and energy consumption. They applied the alternating convex search algorithm to obtain optimized solutions withv a low computational complexity for the task offloading and splitting problems. Liu et al. [29] developed a task offloading model consisting of local edge computing resources, edge server resources at both macro and attached base stations, as well as cloud computing server resources. They developed an approach based on block coordinate descent technique combining convex optimization and gray wolf algorithm in order to simultaneously reduce latency and energy consumption. Zeng et al. [30] discussed task offloading in a multi-layer vehicular edge computing framework and proposed a prediction-based vehicular task offloading scheme, in which a deep learning model is designed to predict the task offloading result and service delay, and then the prediction policy with successful task offloading and minimum service delay is selected as the final offloading scheme.

UAV-assisted MEC networks: A number of research efforts have started to use UAVs as different types of wireless communication platforms, i.e., UAVs equipped with edge servers can be flexibly deployed in specific areas to provide reliable uplink and downlink communications to ground users [31,32]. Therefore, UAV deployment has become a key discussion issue in academia. In particular, a good UAV deployment can improve the quality and efficiency of task offloading for users and enhance the performance of the system [33,34]. Xu et al. [35] proposed a sky–air–ground integrated mobile edge computing system to provide high-quality computing services in areas with missing or damaged communication infrastructures and designed an algorithm based on particle swarm optimization and greedy strategies to obtain a near-optimal solution for UAV deployment and mission offloading in the system. Tian et al. [36] investigated the service satisfaction-oriented task offloading and UAV scheduling problem in UAV-assisted MEC networks, where the task priority was considered based on the latency requirement of tasks and the remaining energy status of users. To solve the proposed problem, they proposed a joint optimization algorithm for task offloading and UAV scheduling based on a genetic algorithm to maximize the total user satisfaction. To address the joint optimization of UAV deployment and offloading decisions in a three-layer UAV-assisted MEC network, Xia et al. [37] proposed a two-layer optimization algorithm to reduce the processing latency of the task, where a differential evolution learning algorithm was used in the upper layer to solve the UAV deployment problem, and a distributed deep neural network was used to generate the offloading decisions in the lower layer. Similarly, confronted with the joint optimization of UAV deployment and computational offloading problem in MEC, Chen et al. [38] designed a two-layer joint optimization approach to minimize the average task response time. First, the outer layer utilized a particle swarm optimization algorithm combined with a genetic algorithm operator to optimize UAV deployment. Second, the inner layer used a greedy algorithm to optimize task offloading.

However, none of the existing works seem to have address the specific challenges posed by large-scale UAV assistance in MEC, particularly in terms of UAV deployment and task offloading problems. In reality, many scenarios have large-scale UAV devices and users numbering more than 1000, such as city traffic management, large competition events, and natural disaster rescue. Nevertheless, existing algorithms encounter a significant challenge with the exponential growth of the search space as the number of devices and users increases, leading to the curse of dimensionality and inefficiency in search. Motivated by this gap, our work proposes a large-scale UAV deployment and task offloading multiobjective optimization method based on the evolutionary algorithm, termed LDOMO. Firstly, LDOMO divides the solution space into a poor solution space and an elite solution space to reduce the search space and improve the search efficiency. Thus, it designs a CSO-based evolutionary strategy and a MLP-based evolutionary strategy to optimize these two solution spaces, respectively. Finally, two local search optimizers are designed to prevent infeasible solutions and enhance the quality of solutions.

3. System Model and Problem Formulation

3.1. System Model and Assumption

As shown in Figure 1, there are I mobile devices, J UAVs, and a edge base station in the considered UAV-assisted MEC. Assume that the set of mobile devices is notated as $MD = \{md_1, md_2, \ldots, md_I\}$ and the set of UAVs is denoted as $UAV = \{uav_1, uav_2, \ldots, uav_J\}$. Moreover, the ith mobile device in the set MD is expressed as $md_i = \langle D_i, C_i, \psi_i \rangle$, where D_i denotes the task data size, C_i denotes the CPU cycles per bit for task computation, and ψ_i is the maximum tolerable delay for task of md_i. The jth UAV in the set UAV is denoted as $uav_j = \langle F_j, O_j \rangle$, where F_j and O_j are the computational and storage resources of uav_j, respectively.

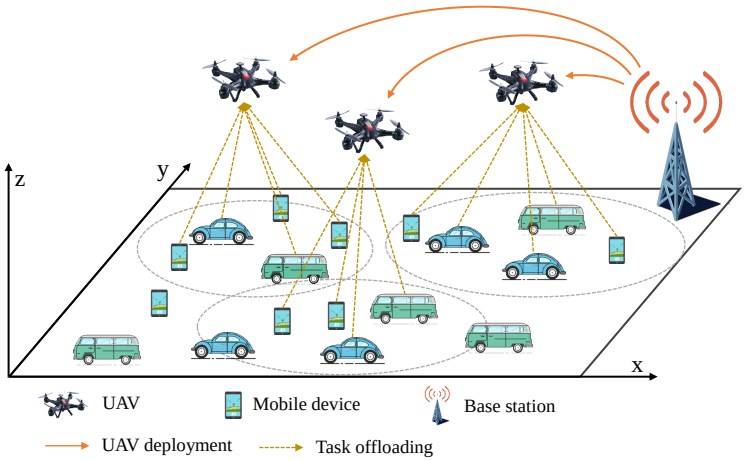


Figure 1. Schematic of UAV deployment and task offloading in a large-scale UAV-assisted MEC.

To accurately represent the positions of UAVs and mobile devices, a 3-dimensional Cartesian coordinate system was employed to model the stereospace of UAV-assisted MEC scenarios. In this system, the position of md_i was denoted as $(x_i^{md}, y_i^{md}, 0)$ and the position of UAV uav_j was denoted as $(x_j^{uav}, y_j^{uav}, H)$. Note that all mobile devices were placed on the ground and all UAVs were deployed on a plane with height H. Assume that the locations of all UAVs were determined and deployed by the edge base station in the control center of the system. Please note that obstacles in UAV flight were not considered in this article. After that, mobile devices can select the nearest or suitable UAV to offload and execute the task according to the task requirements. A binary matrix $\mathbb{L}$ of size $I \times J$ is represented as task offloading decisions for all mobile devices, where the element l_{ij} represents whether the task of md_i is offloaded to uav_j. If the task is offloaded to uav_j, $l_{ij} = 1$, otherwise $l_{ij} = 0$. For ease of description, Table 1 summarizes the main notations in this article.

Table 1. Main notations.

Symbol	Definition
MD	the set of mobile devices
I	the number of mobile devices
UAV	the set of UAVs
J	the number of UAVs
D_i	the task data size of md_i
C_i	the CPU cycles of md_i
ψ_i	the maximum admissible delay of md_i
F_j	the computational resource of uav_j
O_j	the storage resource of uav_j
$(x_i^{md}, y_i^{md}, 0)$	the spatial location of md_i
$(x_j^{uav}, y_j^{uav}, H)$	the spatial location of uav_j
l_{ij}	whether the task of md_i is offloaded to uav_j
B_{ij}	the channel bandwidth between md_i and uav_j
P^t	the transmit power
P^h	the hovering power of the UAV
θ_i	whether the task of md_i can be completed

3.2. Task Offloading Model

There are two main processes involved in the process of task offloading, i.e., the communication process and the computation process. Specifically, in the communication process, the mobile device first communicates with the selected UAV and transmits tasks to the UAV. Then, in the computation process, the tasks of the mobile device are executed on the UAV. Finally, the result of the execution is returned to the mobile device. In this paper, the result return process is ignored because the result data are too small compared with

the task data. The communication process and the computation process are discussed in detail subsequently.

Communication process: Assume that mobile devices communicate with UAVs through wireless links. In this case, the path-loss for the channel between the UAV and mobile device is given by,

$$Loss(md_i, uav_j) = \left(\frac{4\pi f}{c}\right)^2 \times \left(d(md_i, uav_j)\right)^{\alpha}, \tag{3}$$

where f represents the central frequency, c is the speed of light, α represents the path-loss exponent, and $d(md_i, uav_j)$ means the distance between md_i and uav_j, which is given as follows,

$$d(md_i, uav_j) = \sqrt{(x_i^{md} - x_j^{uav})^2 + (y_i^{md} - y_j^{uav})^2 + H^2}. \tag{4}$$

The signal-to-interference plus noise ratio (SINR) [39] of the communication link between md_i and uav_j is calculated as,

$$SINR(md_i, uav_j) = \frac{P^t \times g^2}{N_0 \times Loss(md_i, uav_j)}, \tag{5}$$

where P^t represents the transmit power from the UAV to the mobile device, g is the channel fading coefficient, and N_0 is the noise power. The wireless transmission rate between md_i and uav_j is calculated based on the Shannon's formula as follows,

$$R(md_i, uav_j) = B_{ij} \times \log_2\left(1 + SINR(md_i, uav_j)\right), \tag{6}$$

where B_{ij} is the channel bandwidth between md_i and uav_j.

When md_i chooses to offload its task to uav_j, i.e., $l_{ij} = 1$, then the communication time between them depends on the task size and transmission rate, which is calculated as follows,

$$\overrightarrow{T}(md_i, uav_j) = \frac{l_{ij} D_i}{R(md_i, uav_j)}. \tag{7}$$

Moreover, the corresponding energy consumption during communication can be given as,

$$\overrightarrow{E}(md_i, uav_j) = P^t \times \frac{l_{ij} D_i}{R(md_i, uav_j)}. \tag{8}$$

Communication process: After the mobile device has completed the task transmission, the task will be executed on the transferred UAV. In this paper, the computing power allocated to md_i from uav_j depends on the maximum allowable delay of the task aiming to equitably distribute the computational resources of the UAV. Therefore, the computing power is calculated as follows,

$$f(md_i, uav_j) = \frac{\psi_i^{-1}}{\sum_{md_k \in MD(uav_j)} \psi_k^{-1}} \times F_j, \tag{9}$$

where $MD(uav_j)$ denotes the set of all mobile devices that offloaded tasks to uav_j. According to (9), the more urgent the task of the mobile device, the more computing power is allocated to it.

Moreover, the computation time of task for md_i on uav_j is given as,

$$\hat{T}(md_i, uav_j) = \frac{l_{ij} C_i D_i}{f(md_i, uav_j)}. \tag{10}$$

Following [40], the corresponding energy consumption during computation is modeled as a function related to the computing power, which is given below,

$$\hat{E}(md_i, uav_j) = \gamma l_{ij} C_i D_i \left(f(md_i, uav_j)\right)^2, \tag{11}$$

where γ is the computing energy efficiency coefficient.

3.3. UAV Hover Model

When UAVs provide computing services for mobile devices, they need to hover at a fixed location to ensure uninterrupted communication and quality of service. In this process, the hovering time of UAV depends on the maximum time of receiving and executing tasks, which is calculated as follows,

$$\overline{T}(uav_j) = \max_{md_k \in MD(uav_j)} \left(\overrightarrow{T}(md_k, uav_j) + \hat{T}(md_k, uav_j) \right). \tag{12}$$

Moreover, the hovering energy consumption of UAV is given as follows,

$$\overline{E}(uav_j) = P^h \times \overline{T}(uav_j), \tag{13}$$

where P^h is the hovering power of the UAV.

3.4. Problem Formulation

Based on the above models, the task completion time for each mobile device is as follows,

$$\begin{aligned} T(md_i) &= \sum_{j=1}^{J} \left(\overrightarrow{T}(md_i, uav_j) + \hat{T}(md_i, uav_j) \right) \\ &= \sum_{j=1}^{J} \left(\frac{l_{ij} D_i}{R(md_i, uav_j)} + \frac{l_{ij} C_i D_i}{f(md_i, uav_j)} \right). \end{aligned} \tag{14}$$

Moreover, user satisfaction depends on whether their computational task is completed within the maximum allowable delay. Specifically, the ith user is satisfied if $T(md_i) \leq \psi_i$, denoted as $\theta_i = 1$, and $\theta_i = 0$ otherwise. In this paper, the task completion rate of all users is represented as the user satisfaction in the whole system, which is calculated as,

$$S = \frac{1}{I} \sum_{i=1}^{I} (\theta_i). \tag{15}$$

The total energy consumption of the system is denoted as,

$$E = \sum_{i=1}^{I} \sum_{j=1}^{J} \left(\overrightarrow{E}(md_i, uav_j) + \hat{E}(md_i, uav_j) \right) + \sum_{j=1}^{J} \left(\overline{E}(uav_j) \right). \tag{16}$$

In this paper, the joint optimization problem of UAV deployment and task offloading is modeled as a large-scale multiobjective optimization problem with the purposes of reducing energy consumption and user satisfaction. Mathematically, the formulated problem is defined as follows,

$$\begin{cases} \max_{\mathbf{x}^{uav}, \mathbf{y}^{uav}, \mathbb{L}} S \\ \min_{\mathbf{x}^{uav}, \mathbf{y}^{uav}, \mathbb{L}} E \end{cases} \tag{17}$$

$$s.t. \quad \sum_{j}^{J} l_{ij} = 1, \forall i \in I, \tag{18}$$

$$\sum_{i}^{I} (l_{ij} f_{ij}) \leq F_j, \forall j \in J, \tag{19}$$

$$\sum_{i}^{I} (l_{ij} d_{ij}) \leq O_j, \forall j \in J, \tag{20}$$

$$f_{ij} > 0, \forall i \in I, \forall i \in J, \tag{21}$$

$$l_{ij}, \theta_i \in \{0, 1\}, \forall i \in I, \forall i \in J. \tag{22}$$

In the formulated model, the primary goal is to maximize the user satisfaction of mobile devices while minimizing the energy consumption of system in (17) with three decision variables and several constraints. Specifically, the real decision variables $\mathbf{x}^{uav}$ and $\mathbf{y}^{uav}$ denote the horizontal position of the UAV. The binary decision variable $\mathbb{L}$ represents the task offloading scheme of the user. Constraint (18) guarantees that the task of each mobile device can be offloaded to only one UAV. Moreover, constraints (19) and (20) ensure that the computational and storage resources of each UAV cannot be exceeded, respectively. Constraints (21) and (22) define the range of values for some parameters.

4. Optimization Method

In this section, we propose a multiobjective optimization evolutionary algorithm to solve the above formulated large-scale multiobjective optimization problem in (17), called LDOMO. First, the framework of LDOMO is presented, and then the details of LDOMO are introduced subsequently.

4.1. The Overall Framework

This section presents the general framework of LDOMO proposed in this paper. The flowchart and pseudocode of LDOMO are shown in Figure 2 and Algorithm 1, respectively. As shown in Algorithm 1, LDOMO starts with two input data: a list of UAVs and a list of mobile devices. Then, the initialization process is executed in lines 1 and 2, where a population Pop and a **MLP** are initialized, as detailed in Section 4.2. Subsequently, LDOMO enters the main loop in lines 3 to 7, which consists of three core processes (i.e., competition phase, evolution phase, and optimization phase). In the competition phase (line 4), the entire population is split into two equal-sized sets, including an elite solution set $\mathbf{X}_e$ with a better performance and a poor solution set $\mathbf{X}_p$ with a worse performance. This competitive operation provides important labels and guidelines for the subsequent evolution phase, which is described in Section 4.3. In the evolution phase (line 5), two different evolutionary strategies are employed in LDOMO to evolve $\mathbf{X}_p$ and $\mathbf{X}_e$, respectively, i.e., CSO-based evolutionary strategy and MLP-based evolutionary strategy. Specifically, in the CSO-based evolutionary strategy, the poor solutions in $\mathbf{X}_p$ learn directly from the elite solutions in $\mathbf{X}_e$ to accelerate population convergence. Moreover, in the MLP-based evolutionary strategy, **MLP** is trained to learn the gradient direction of the evolutionary process by adopting the poor and elite solutions. After that, the elite solutions in $\mathbf{X}_e$ are evolved through the trained **MLP** to maintain the population diversity and enhance the population quality. Note that details of the two evolutionary strategies are presented in Section 4.4. To further improve the quality of the solution in Pop, the UAV deployment and task offloading for each solution are optimized in line 6 via two local search optimizers, i.e., a deployment optimizer and an offloading optimizer. In the deployment optimizer, redundant UAVs in each solution are eliminated based on the distance between UAVs to reduce energy consumption. Moreover, in the offloading optimizer, the failed tasks in each solution are assigned feasible UAVs as much as possible to improve user satisfaction. Note that the details of local search optimizers are presented in Section 4.5. Finally, the entire evolutionary loop is terminated when the maximum number of generations t_{max} is reached, and the final population obtained by LDOMO is output.

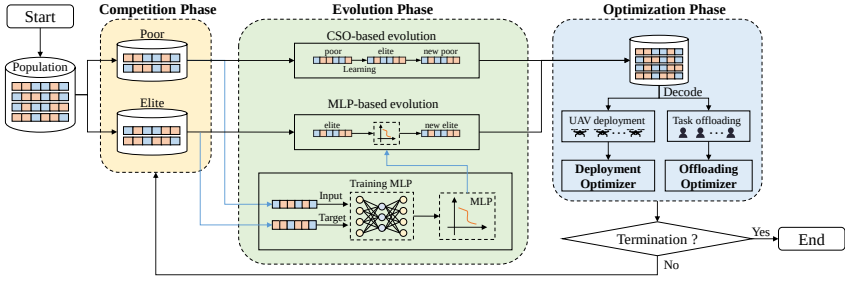


Figure 2. Framework of LDOMO.

Algorithm 1 LDOMO

Input: UAV list, MD list.
Output: The final population Pop.
1: initialize a population Pop;
2: initialize a **MLP**;
3: **for** t : 1 to t_{max} **do**
4: $\mathbf{X}_e, \mathbf{X}_p \leftarrow$ Competition Phase(Pop);\\Algorithm 2
5: $Pop \leftarrow$ Evolution Phase(**MLP**, $\mathbf{X}_e, \mathbf{X}_p$);\\Algorithm 3
6: Optimization Phase(Pop);\\Algorithm 4
7: **end for**

4.2. Population Initialization

Regarding the population representation and initialization, an equal-length integer encoding is employed to represent each solution in the population. Specifically, the locations of all UAVs and the decisions of all mobile devices are encoded in one solution. For example, in an MEC environment of 10×10 m^2 with two UAVs and three mobile devices, a solution is encoded as 2415221. The first four bits of the solution (i.e., 2415) indicate that the locations of the first and the second UAVs are (2 m, 4 m) and (1 m, 5 m), respectively. Moreover, the last three bits (i.e., 221) indicate the first two mobile devices offload the task to the second UAV, and the last one offloads to the first UAV. As for the population initialization, N solutions are randomly generated to form an initial population Pop that is evolved and optimized in subsequent sections.

4.3. Competition Phase

In order to clearly distinguish the quality of solutions, a competitive strategy in LDOMO is used to divide the population into two sets by employing fast non-dominated sort [41] and crowding distance [42], including a poor solution set with a worse performance and an elite solution set with a better performance. As for the fast non-dominated sort, if solution x^* is not worse than any solution after competition, it is called a non-dominated solution and its ranking level is $0 (Rank(x^*) = 0)$. After removing the solutions with a ranking level of 0, if solution x' is not worse than the remaining solutions, the ranking level of x' is $1 (Rank(x') = 1)$. Through analogy, the ranking level of all solutions can be calculated. Please note that the lower the ranking levels of the solution, the better the quality of the solution. However, when the solutions are in the same ranking level, the quality of them cannot be determined due to the fact that the solution has multiple optimization objectives. In this case, the crowding distance is used to further sort these solutions, which is calculated as follows,

$$\begin{cases} CD(x_i) = \infty, & i = 1 \; or \; n, \\ CD(x_i) = \sum_m^M \sum_i^I \frac{f_m(x_{i+1}) - f_m(x_{i-1})}{f_m^{max} - f_m^{min}}, & i = 2, \ldots, n-1, \end{cases} \tag{23}$$

where n represents the number of solutions and f_m^{max} and f_m^{min} are the maximum and minimum values of the mth objective function. In summary, all of the solutions in the population can be ranked from good to bad according to the following conditions,

$$\begin{cases} Rank(x_i) < Rank(x_j), \\ Rank(x_i) = Rank(x_j) \; and \; CD(x_i) > CD(x_j). \end{cases} \tag{24}$$

After the solution is sorted, the first $N/2$ solutions are labeled as elite and added to the set $\mathbf{X}_e$, while the remaining solutions are added to the set $\mathbf{X}_p$. Through this competitive strategy, LDOMO can efficiently rank solutions with multiple optimization objectives and provide critical labels and guidance for subsequent population evolution.

Algorithm 2 presents the pseudocode of the competition phase with the current population Pop as the input. In particular, two empty sets, $\mathbf{X}_e$ and $\mathbf{X}_p$, are initialized in lines 1 and 2 to store the elite and poor solutions, respectively. Subsequently, the fitness of

all solutions in Pop is calculated based on Equations (15) and (16) in line 3, which serves as a metric for solution competition and is stored in the set Fit. Then, the solutions in Pop compete with each other and are sorted in lines 4–6. Specifically, in line 4, a fast non-dominated sort is performed according to Fit and the ranking levels of all solutions are stored in set $Rank$. In line 5, the crowding distances of solutions with the same ranking level are calculated and added to set CD. In line 6, the solutions in Pop are sorted from good to bad according to Equation (24), with $Rank$ and Fit as inputs. After that, the sorted solutions are labeled in lines 7–13. The first $N/2$ solutions are labeled as elite solutions and added to set $\mathbf{X}_e$ in line 9. The remaining solutions are labeled as poor solutions and added to $\mathbf{X}_p$ in line 11. Finally, the set of elite solutions $\mathbf{X}_e$ and the set of poor solutions $\mathbf{X}_p$ are returned in line 14.

Algorithm 2 Competition Phase

Input: Pop.
Output: $\mathbf{X}_e$, $\mathbf{X}_p$.
1: initialize an empty set $\mathbf{X}_e$ to store the elite solutions;
2: initialize an empty set $\mathbf{X}_p$ to store the poor solutions;
3: $Fit \leftarrow$ calculate the fitness of solutions in Pop based on (15) and (16);
4: $Rank \leftarrow$ non-dominated sorting of solutions in Pop according to Fit;
5: $CD \leftarrow$ calculate the crowding distance of solutions based on $Rank$ and Fit;
6: $Pop_{sort} \leftarrow$ sort the solutions in Pop according to (24);
7: **for** $x \in Pop_{sort}$ **do**
8: **if** x ranks in the first $N/2$. **then**
9: $\mathbf{X}_e \leftarrow$ add x as an elite solution;
10: **else**
11: $\mathbf{X}_p \leftarrow$ add x as a poor solution;
12: **end if**
13: **end for**
14: return $\mathbf{X}_e$, $\mathbf{X}_p$.

4.4. Evolution Phase

In the process of the population evolution phase, LDOMO needs to optimize the UAV deployment and task offloading decisions in each solution. In fact, the real-life UAV deployment and task offloading problem is a large-scale optimization problem due to the dimension of its search space being larger than 1000. However, traditional evolutionary algorithms face large-scale optimization problems with rapidly decreasing efficiency due to dimensionality catastrophe. Inspired by [43,44], two different evolutionary strategies are employed in LDOMO to explore different solution spaces ($\mathbf{X}_p$ and $\mathbf{X}_e$), i.e., the CSO-based evolutionary strategy and the MLP-based evolutionary strategy. Specifically, the former strategy makes the poor solutions in $\mathbf{X}_p$ quickly approach the elite solutions in $\mathbf{X}_e$ as a way to accelerate population convergence. Meanwhile, the latter strategy exploits the gradient direction of population evolution learned by the MLP to guide elite individuals in $\mathbf{X}_e$ to more promising regions, which maintains the diversity of population and further enhances the quality of population. Employing different search strategies for different features of the solution space, LDOMO can overcome the challenges posed by dimensional catastrophe in large-scale optimization. Note that the operational procedures and specific details of population evolution phase are given subsequently.

In the CSO-based evolutionary strategy, the poor solutions in $\mathbf{X}_p$ observe and learn the behavior of the elite solutions in $\mathbf{X}_e$. This learning process facilitates knowledge transfer from the elite to the poor solutions, which promotes the convergence of the population. Assuming that the velocity and position of the poor solution are v_p and x_p, respectively, their update equations are as follows,

$$v_p(t+1) = r_1 \times v_p(t) + r_2 \times \left(x_e(t) - x_p(t)\right) + \varphi \times r_3 \times \left(\bar{x}(t) - x_p(t)\right), \tag{25}$$

$$x_p(t+1) = x_p(t) + v_p(t+1), \tag{26}$$

where r_1, r_2, and r_3 are three random numbers ranging from 0 to 1, t represents the t-th iteration, $x_e(t)$ is the position of an elite solution, $\bar{x}(t)$ is the mean position of all solutions in the t-th iteration, and φ is a control parameter of $\bar{x}(t)$. On the other hand, it is difficult for elite solutions to learn implicit knowledge from themselves for further evolution, because they are in the vicinity of a locally optimal solution. Therefore, a multilayer perceptron(MLP) in LDOMO is used to explore more promising regions to guide the evolution of elite solutions. In the MLP-based evolutionary strategy, a three-layer MLP is trained by taking $\mathbf{X}_p$ as the input, while labeling $\mathbf{X}_e$ as the target output. In this supervised learning way, the MLP can learn the possible fast convergence direction for the input solutions. Subsequently, using the trained MLP, the velocities and positions of the elite solutions in $\mathbf{X}_e$ are updated in the following way.

$$v_e(t+1) = v_e(t) + r_1(x_{MLP}(t) - x_e(t)) + r_2(x_{best}(t) - x_e(t)) + r_3(x_{rd1}(t) - x_{rd2}(t)), \tag{27}$$

$$x_e(t+1) = x_e(t) + v_e(t+1), \tag{28}$$

where x_{MLP} is the output solution by inputting x_e to the trained MLP, x_{best} is the historically best solution for the entire population, x_{rd1} and x_{rd2} are two randomly selected solutions from the current population, respectively. Obviously, the evolution of the solution x defined in (27) consists of three search components: one is a self-acceleration component provided by the knowledge of MLP (i.e., $x_{MLP}(t) - x_e(t)$), the second one is the global acceleration component guided by the global best solution (i.e., $x_{best}(t) - x_e(t)$), and the last one is the differential-regulation component guided by other solutions (i.e., $x_{rd1}(t) - x_{rd2}(t)$). In this way, the convergence of population evolution can be accelerated by following the fast convergence direction provided by MLP, while the correctness of the convergence direction can be guaranteed by the guidance of the global best solution. In addition, differential regulation prevents the evolutionary process from falling into a local optimum to some degree.

Algorithm 3 presents the pseudocode of population evolution. In particular, an empty set Qsp is firstly initialized in line 1 to store the offspring. Subsequently, two critical evolutionary strategies of LDOMO are implemented in lines 2–17. Specifically, the main loop of the CSO-based evolutionary strategy is accessed in line 2. Then, in line 3, an elite solution is randomly selected from $\mathbf{X}_e$ for guiding the current poor solution evolution. The mean position of all solutions in Pop is calculated and noted as $\overline{x}$ in line 4. In lines 5 and 6, the velocity and position of x_p are sequentially updated according to (25) and (26). The updated x_p is added to Qsp in line 7. After all of the solutions in $\mathbf{X}_p$ have been updated, the MLP-based evolutionary strategy is executed to update the elite solutions in $\mathbf{X}_e$ in lines 9–17. The **MLP** is firstly trained by inputting $\mathbf{X}_e$ and $\mathbf{X}_p$ in line 9. Then, solutions x_{MLP}, x_{best}, x_{rd1}, and x_{rd2} are generated sequentially for assisted evolution in lines 11–13. After that, the velocity and position of x_e are sequentially updated according to (27) and (28) in lines 14 and 15. In line 16, the updated x_e is added to Qsp. Finally, Qsp is returned as the next generation population in line 18.

4.5. Optimization Phase

In the process of population evolution, UAV deployments in each solution may appear aggregation and uneven distribution due to the stochastic nature of evolutionary algorithms. However, overcrowded UAVs lead to wasted computational resources or even economic losses due to crashes. Therefore, a local search optimizer is designed to optimize the deployment of UAVs by eliminating redundant UAVs that are too close together. In this way, the safety of UAVs is ensured and energy consumption can be reduced by reducing the number of UAVs. In order to further improve the quality of solutions at the same time, another local search optimizer was designed in LDOMO to optimize task offloading for each solution to enhance user satisfaction. Specifically, for each failed task that cannot be completed before the maximum allowable delay T_{max}, the first feasible UAV is selected as

the new offloading scheme for the failed task from the UAV list based on the first-come-first-served principle. Note that this process ensures that the original successful tasks are completed normally and will be stopped when a suitable UAV is found. In this way, LDOMO is able to maximize the user satisfaction, thus improving the reliability of the system and enhancing the user experience.

Algorithm 3 Evolution Phase

Input: **MLP**, $\mathbf{X}_e$, $\mathbf{X}_p$.
Output: Qsp.
1: initialize the offspring Qsp;
/* Access to CSO-based evolutionary strategy.
2: **for** each $x_p \in \mathbf{X}_p$ **do**
3: $x_e \leftarrow$ get a elite solution from $\mathbf{X}_e$;
4: $\overline{x} \leftarrow$ get the mean position of all solutions;
5: update the velocity of x_p according to (25);
6: update the position of x_p according to (26);
7: $Qsp \leftarrow$ add the updated x_p;
8: **end for**
/* Access to MLP-based evolutionary strategy.
9: **MLP** $\leftarrow$ train the **MLP** by inputting $\mathbf{X}_e$ and $\mathbf{X}_p$;
10: **for** each $x_e \in \mathbf{X}_e$ **do**
11: $x_{MLP} \leftarrow$ get a new solution by inputting x_e to **MLP**;
12: $x_{best} \leftarrow$ get the global best solution from Pop;
13: $x_{rd1}, x_{rd2} \leftarrow$ select two different random solutions;
14: update the velocity of x_e according to (27);
15: update the position of x_e according to (28);
16: $Qsp \leftarrow$ add the updated x_e;
17: **end for**
18: return Qsp as the next generation population.

Algorithm 4 provides the pseudocode of the population optimization phase, which consists of two main loops for the key optimizers—the deployment optimizer and the unloading optimizer. First, the deployment optimizer is executed in lines 1–11 to optimize the UAV deployment scheme for each solution. In particular, the UAV deployment scheme for each solution **x** in Pop is decoded and presented in set UAV in line 2. Then in lines 3 to 9, each UAV **uav** in UAV is checked and determined if it needs to be eliminated. Concretely, the Euclidean distance of the **uav** from other UAVs is calculated in line 4. In line 5, the minimum distance between **uav** and other UAVs will be evaluated if it is less than the minimum allowable distance $d_{min} = 40$ m. If it does, then **uav** is eliminated from UAV in line 6 and the tasks on **uav** are migrated to the nearest UAV in line 7. After checking all of the UAVs, the UAV deployments in **x** are updated in line 10. Subsequently, the offloading optimizer is executed in lines 12–25 to optimize the task offloading scheme for each solution. Specifically, the task offloading schemes of all of the mobile devices for each solution **x** are firstly decoded and presented in set $Task$ in line 13. Then, the completion time of each **task** in $Task$ is calculated in line 15. Moreover, if the completion time of **task** is greater than its maximum allowed delay T_{max}, LDOMO tries to find a feasible UAV for this failed task in line 17. If a feasible UAV is successfully found, the offloading decision of **task** is updated in **x**, otherwise **task** is marked as a failed task. Finally, all task offloading decisions for each solution in the population Pop are optimized and the optimized Pop is returned.

Algorithm 4 Optimization phase

Input: Pop.
Output: The optimized Pop.
/* Access to deployment optimizer.
for each **x** $\in Pop$ **do**
  $UAV \leftarrow$ decode the location of UAVs from **x**;
  for each **uav** $\in UAV$ **do**
    calculate distance $d_{\mathbf{uav}}$ between **uav** and others;
    if $\min(d_{\mathbf{uav}}) < d_{min}$ **then**
      eliminate **uav** from UAV;
      migrate tasks from **uav** to nearest UAV;
    end if
  end for
  update UAVs and tasks in **x**;
end for
/* Access to offloading optimizer.
for each **x** $\in Pop$ **do**
  $Task \leftarrow$ decode the decisions of tasks from **x**;
  for each **task** $\in Task$ **do**
    calculate the completion time $t_{\mathbf{task}}$ of **task**;
    if $t_{\mathbf{task}} > T_{max}$ **then**
      try to find a viable decision for **task**;
      if success **then**
        update **task** in **x**;
      else
        mark **task** as a failed task;
      end if
    end if
  end for
end for

5. Simulation Results

5.1. Environmental Setup

In this section, simulation experiments are performed to evaluate the performance of LDOMO, which are implemented in Python 3.11, and run on a PC with Intel (R) Core (TM) i7-12700 CPU and 32.0 GB RAM. Table 2 summarizes the environmental settings for the simulation experiments.

Table 2. Environment settings.

Parameter	Value
Distribution area of UAVs and mobile devices	1000×1000 m^2
Number of mobile devices (I)	**100, 200, . . . , 1000**
Number of UAVs (J)	**50, 100, . . . , 500**
Transmission data of one task	[0.5, 2] Mbits
CPU cycles to compute one bit	[100, 1000] CPU cycles
Maximum delay of one task	[100, 200] ms
Transmission bandwidth	100 Mbps
Computation capabilities of UAVs	[5, 10] GHz
Transmit power	1 W
Hovering power of UAVs	1 kW
Noise power	-117 dB
Central frequency	2 GHz
Minimum allowable distance of UAVs	4 m

5.2. Experimental Setup

In order to comprehensively evaluate the performance of LDOMO, several representative methods are adopted for comparisons, including NSGA-II [45], MOEA/D [46], LMOCSO [47], and MOPSO [48]. In addition, a variant of the proposed method is used for comparison, i.e., LDOMO-I, which does not use two local search optimizers to verify their effectiveness. Moreover, the parameters of all of the algorithms are summarized in Table 3. In particular, the number of neurons in the hidden layer of the MLP is set to 40 in our LDOMO. For training the MLP in each generation, the learning rate LR is set to 0.1, the number of epochs is set as $e = 20$. For the sake of fairness, all algorithms are independently run 20 times and the average results are recorded to evaluate the overall performance of the algorithms.

Table 3. Algorithm settings.

	Parameters	Common Parameters
LDOMO, LDOMO-I	LR = 0.1, e = 20, K = 40	Population Size = 100, Iteration = 200, Runtimes = 20
NSGA-II	R_c = 0.9, R_m = 0.08	
MOEA/D	T = 20, δ = 0.9, R_c = 0.8, R_m = 0.01	
LMCSO	θ = 0.1	
MOPSO	w = 0.4, $c1$ = 2.0, $c2$ = 2.0	

5.3. Simulation Results

In this section, the experimental results are discussed by comparing the performance of our method with several representative multiobjective evolutionary algorithms on different experimental scales. First, comprehensive performance comparisons of all algorithms are discussed across four different scales, each featuring varying numbers of mobile devices (UAVs): 200 (100), 400 (200), 600 (300), and 1000 (500). Subsequently, the impact of different numbers of mobile devices is analyzed based on algorithm performance, maintaining a constant number of UAVs set at 500. Finally, the effect of different numbers of UAVs is explored on the algorithm performance while keeping the number of mobile devices constant at 1000.

Performance comparison: Figure 3 shows the non-dominated solution sets obtained by all algorithms in terms of both energy consumption and user satisfaction. In a small-scale setting (i.e., 200 (100)), LDOMO-I does not dominate MOEA/D, but shows a slight performance improvement over NSGA-II, LMCSO, and MOPSO. However, LDOMO emerges as the superior performer, dominating all other algorithms due to its lowest energy consumption and highest user satisfaction. This dominance can be attributed to the effectiveness of the two local search optimizers employed by LDOMO, which contribute to reducing energy consumption and enhancing user satisfaction, thereby enhancing solution quality. In the last three large-scale experiments (i.e., 400 (200), 600 (300), and 1000 (500)), LDOMO and LDOMO-I both outperform the other algorithms and have better scalability. This illustrates that the CSO-based evolutionary strategy and MLP-based evolutionary strategy used in both LDOMO and LDOMO-I are able to learn faster convergence directions and improve convergence efficiency.

Impacts of different numbers of mobile devices: Table 4 presents the experimental results with varying numbers of mobile devices, focusing on energy consumption and user satisfaction. As shown in Table 4, the energy consumption of the system rises with the increasing number of mobile devices, attributed to the heightened computing requirements. Consequently, user satisfaction gradually diminishes as the system contends with higher computational loads. Across all scales of mobile devices, LDOMO consistently demonstrates a superior performance, maintaining the lowest energy consumption and ensuring user satisfaction of over 86%. In contrast, other algorithms exhibit lower user satisfac-

tion levels, hovering around 70%, particularly in large-scale environments. In summary, LDOMO outperforms existing representative evolutionary algorithms across different scales of mobile devices. Its ability to effectively balance energy consumption and user satisfaction underscores its suitability for addressing the complexities of UAV-assisted MEC systems, regardless of the scale of mobile device deployment.

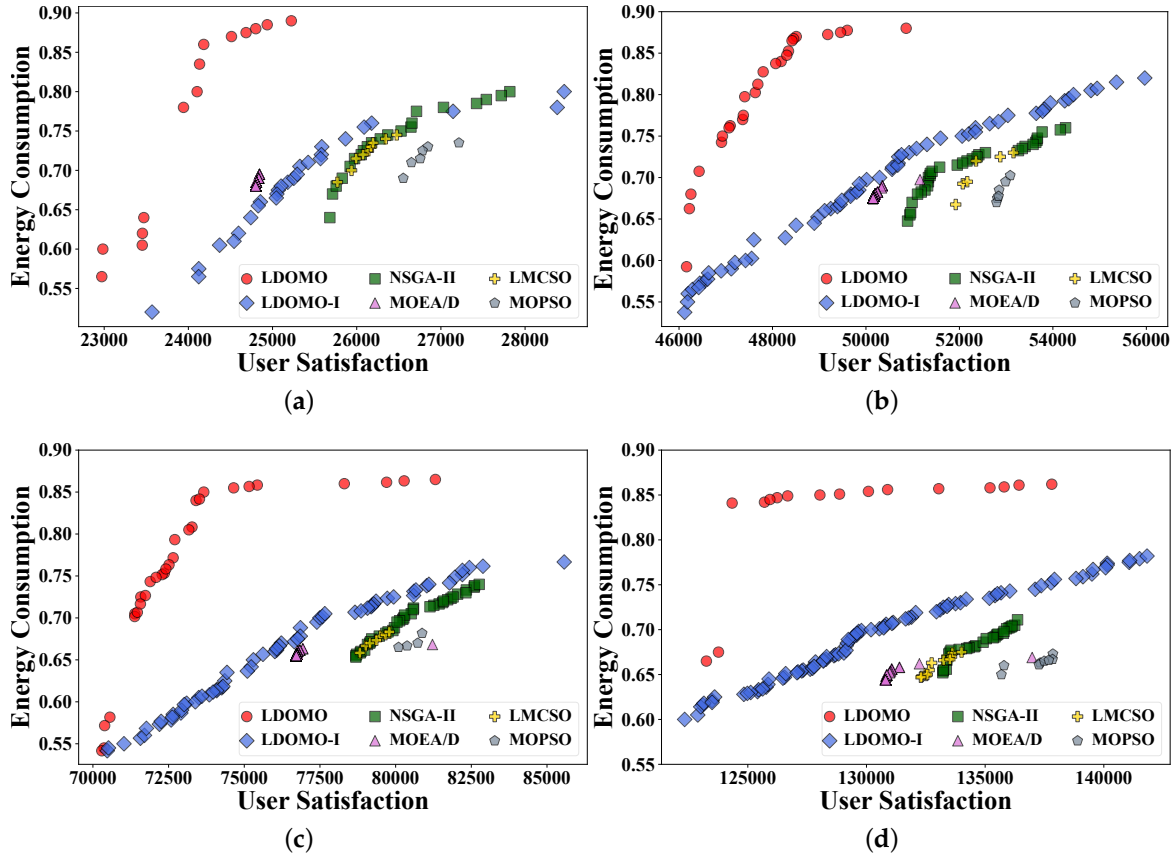


Figure 3. Performance comparisons of all algorithms on four different scales. (**a**) 200 (100); (**b**) 400 (200); (**c**) 600 (300); (**d**) 1000 (500).

Table 4. Comparison results for different numbers of mobile devices.

I (J = 500)	Energy Consumption					User Satisfaction				
	LDOMO	NSGA-II	MOEA/D	LMCSO	MOPSO	LDOMO	NSGA-II	MOEA/D	LMCSO	MOPSO
100	**56,970.1**	57,431.5	57,002.0	57,853.4	58,077.6	**0.888**	0.830	0.720	0.770	0.760
200	**63,414.9**	65,684.9	64,640.7	65,974.2	66,650.8	**0.884**	0.790	0.700	0.740	0.730
300	**71,332.8**	74,111.9	72,910.3	74,336.2	75,317.4	**0.860**	0.743	0.697	0.690	0.687
400	**77,574.7**	81,653.4	80,162.3	81,585.6	82,668.7	**0.880**	0.772	0.700	0.702	0.702
500	**84,481.2**	90,214.1	89,105.9	90,339.2	91,916.8	**0.876**	0.730	0.688	0.690	0.686
600	**92,729.7**	98,438.2	97,719.3	98,768.7	100,130.7	**0.867**	0.722	0.673	0.683	0.685
700	**102,604.7**	107,783.9	105,357.2	106,677.9	109,443.0	**0.867**	0.713	0.667	0.690	0.680
800	**103,946.7**	115,604.6	113,217.9	115,313.4	117,403.9	**0.867**	0.712	0.670	0.680	0.677
900	**112,344.3**	124,771.6	122,638.6	124,138.6	126,358.2	**0.863**	0.710	0.660	0.677	0.677
1000	**123,255.4**	133,216.1	130,804.1	132,296.6	135,684.6	**0.862**	0.711	0.669	0.675	0.673

Impacts of different numbers of UAVs: Table 5 shows the experimental results with different scales of UAVs in terms of energy consumption and user satisfaction. As observed, the energy consumption of the system exhibits a positive trend with the increasing number of UAVs, primarily due to the hovering energy consumption of UAVs. However, the user

satisfaction of mobile devices shows improvement with the escalation of UAVs, attributed to the augmented edge computing resources facilitated by the increased UAV deployment. Notably, LDOMO consistently demonstrates a superior performance across varying numbers of UAVs, maintaining the lowest energy consumption and the highest levels of user satisfaction. By effectively balancing energy consumption and user satisfaction, LDOMO emerges as a promising solution for addressing the challenges posed by different scales of UAV deployment in UAV-assisted MEC environments.

Table 5. Comparison Results for Different Numbers of UAVs.

I (J = 1000)	Energy Consumption					User Satisfaction				
	LDOMO	**NSGA-II**	**MOEA/D**	**LMCSO**	**MOPSO**	**LDOMO**	**NSGA-II**	**MOEA/D**	**LMCSO**	**MOPSO**
50	**74,540.7**	87,534.9	83,418.2	86,323.6	89,026.3	**0.844**	0.787	0.657	0.682	0.666
100	**77,093.4**	92,497.9	89,635.2	90,925.3	95,479.4	**0.857**	0.716	0.675	0.674	0.680
150	**85,530.7**	97,687.4	94,877.5	99,300.1	100,112.3	**0.857**	0.705	0.664	0.698	0.667
200	**86,474.1**	102,975.9	99,698.2	104,327.6	104,663.8	**0.859**	0.722	0.677	0.703	0.668
250	**92,963.8**	108,354.3	105,686.8	106,151.1	110,291.5	**0.861**	0.715	0.665	0.664	0.679
300	**99,043.7**	112,999.2	110,784.6	112,751.4	115,239.3	**0.862**	0.721	0.657	0.676	0.670
350	**106,903.4**	117,161.8	114,714.7	118,218.3	119,621.3	**0.861**	0.701	0.665	0.678	0.668
400	**112,741.3**	122,807.7	120,043.5	122,736.8	124,896.7	**0.860**	0.719	0.669	0.680	0.664
450	**113,291.5**	128,049.4	125,420.1	127,362.4	129,851.3	**0.862**	0.711	0.666	0.671	0.661
500	**123,255.4**	133,216.1	130,804.1	132,296.6	135,684.6	**0.862**	0.711	0.669	0.675	0.673

6. Conclusions and Future Work

In this paper, we focused on the challenges of joint UAV deployment and task offloading in a large-scale UAV-assisted MEC. First, we formulated joint UAV deployment and task offloading problem as a large-scale multiobjective optimization problem with the goal of minimizing the energy consumption while improving user satisfaction. Then, we designed a large-scale UAV deployment and task offloading multiobjective optimization method based on the evolutionary algorithm (called LDOMO) to effectively address the above formulated problem. In the proposed LDOMO, the entire population was first decomposed into the set of poor solutions and the set of elite solutions. Then, CSO-based and MLP-based evolutionary strategies were proposed to explore the two solution spaces with the aim of accelerating the convergence while improving the quality of the population. Subsequently, two local search optimizers were designed to avoid infeasible solutions and to further improve the quality of solutions. Finally, we simulated different experimental scales to compare the performance of our method with existing representative evolutionary algorithms and discussed the impact of different numbers of UAVs and mobile devices. Extensive simulation results have shown that our approach outperforms several representative evolutionary algorithms in terms of energy consumption and user satisfaction in experiments of different scales.

In future work, we will further consider the problem of UAV trajectory planning with obstacle avoidance in large-scale UAV-assisted MEC. In addition, we are interested in exploring other scenarios to extend the research work of this paper, such as partial task offloading, wireless energy harvesting, and collaboration between UAV and base stations.

Author Contributions: Conceptualization, Q.Q. and L.L.; Methodology, Q.Q. and L.L.; Software, Q.Q.; Validation, Q.Q.; Resources, Q.Q.; Data curation, Q.Q.; Writing—original draft, Q.Q.; Writing—review & editing, L.L.; Supervision, L.L., Z.X., Y.F., Q.L. and Z.M.; Project administration, Z.X., Q.L. and Z.M.; Funding acquisition, Z.X., Y.F., Q.L. and Z.M. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Natural Science Foundation of Guangdong Province under Grants 2023A1515011296 and 2023A1515011238; in part by the Stable Support Project of Shenzhen (Project No. 20231120145719001); in part by the National Natural Science Foundation of China (NSFC) under Grants 62376163 and 62272315; in part by the Shenzhen Science and Technology Foundation under Grants JCYJ20210324093212034 and JCYJ20220531101411027; in part by 2022 Guangdong Province Undergraduate University Quality Engineering Project (Shenzhen University Academic Affairs [2022] No. 7); and in part by the Guangdong Regional Joint Foundation Key Project under Grant 2022B1515120076.

Data Availability Statement: The data will be made available by the authors on request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Yin, Y.; Zheng, P.; Li, C.; Wang, L. A State-of-The-Art Survey on Augmented Reality-Assisted Digital Twin for Futuristic Human-Centric Industry Transformation. *Robot. Comput.-Integr. Manuf.* **2023**, *81*, 102515. [CrossRef]
2. Chen, Z.; Hu, J.; Min, G.; Luo, C.; El-Ghazawi, T. Adaptive and efficient resource allocation in cloud datacenters using actor-critic deep reinforcement learning. *IEEE Trans. Parallel Distrib. Syst.* **2021**, *33*, 1911–1923. [CrossRef]
3. Chen, Z.; Xiong, B.; Chen, X.; Min, G.; Li, J. Joint Computation Offloading and Resource Allocation in Multi-edge Smart Communities with Personalized Federated Deep Reinforcement Learning. *IEEE Trans. Mob. Comput.* **2024**. [CrossRef]
4. Li, L.; Qiu, Q.; Xiao, Z.; Lin, Q.; Gu, J.; Ming, Z. A Two-Stage Hybrid Multi-Objective Optimization Evolutionary Algorithm for Computing Offloading in Sustainable Edge Computing. *IEEE Trans. Consum. Electron.* **2024**, *70*, 735–746. [CrossRef]
5. Chen, Z.; Yu, Z. Intelligent offloading in blockchain-based mobile crowdsensing using deep reinforcement learning. *IEEE Commun. Mag.* **2023**, *61*, 118–123. [CrossRef]
6. Xiao, Z.; Qiu, Q.; Li, L.; Feng, Y.; Lin, Q.; Ming, Z. An Efficient Service-Aware Virtual Machine Scheduling Approach Based on Multi-Objective Evolutionary Algorithm. *IEEE Trans. Serv. Comput.* **2023**. [CrossRef]
7. Chen, Z.; Zhang, J.; Zheng, X.; Min, G.; Li, J.; Rong, C. Profit-Aware Cooperative Offloading in UAV-Enabled MEC Systems Using Lightweight Deep Reinforcement Learning. *IEEE Internet Things J.* **2023**, *11*, 21325–21336. [CrossRef]
8. Adnan, M.H.; Zukarnain, Z.A.; Amodu, O.A. Fundamental Design Aspects of UAV-Enabled MEC Systems: A Review on Models, Challenges, and Future Opportunities. *Comput. Sci. Rev.* **2024**, *51*, 100615. [CrossRef]
9. Shi, B.; Chen, Z.; Xu, Z. A Deep Reinforcement Learning Based Approach for Optimizing Trajectory and Frequency in Energy Constrained Multi-UAV Assisted MEC System. *IEEE Trans. Netw. Serv. Manag.* **2024**. [CrossRef]
10. Wang, Y.; Ru, Z.Y.; Wang, K.; Huang, P.Q. Joint Deployment and Task Scheduling Optimization for Large-scale Mobile Users in Multi-UAV-Enabled Mobile Edge Computing. *IEEE Trans. Cybern.* **2019**, *50*, 3984–3997. [CrossRef]
11. Goudarzi, S.; Soleymani, S.A.; Wang, W.; Xiao, P. UAV-Enabled Mobile Edge Computing for Resource Allocation Using Cooperative Evolutionary Computation. *IEEE Trans. Aerosp. Electron. Syst.* **2023**, *59*, 5134–5147. [CrossRef]
12. Kumbhar, F.H.; Shin, S.Y. Innovating Multi-Objective Optimal Message Routing for Unified High Mobility Networks. *IEEE Trans. Veh. Technol.* **2023**, *72*, 6571–6583. [CrossRef]
13. Guo, H.; Wang, Y.; Liu, J.; Liu, C. Multi-UAV Cooperative Task Offloading and Resource Allocation in 5G Advanced and Beyond. *IEEE Trans. Wirel. Commun.* **2024**, *23*, 347–359. [CrossRef]
14. Ning, Z.; Hu, H.; Wang, X.; Guo, L.; Guo, S.; Wang, G.; Gao, X. Mobile Edge Computing and Machine Learning in The Internet of Unmanned Aerial Vehicles: A Survey. *ACM Comput. Surv.* **2023**, *56*, 13. [CrossRef]
15. Chen, Z.; Zhang, J.; Huang, Z.; Wang, P.; Yu, Z.; Miao, W. Computation offloading in blockchain-enabled MCS systems: A scalable deep reinforcement learning approach. *Future Gener. Comput. Syst.* **2024**, *153*, 301–311. [CrossRef]
16. Asim, M.; Mashwani, W.K.; Shah, H.; Belhaouari, S.B. An Evolutionary Trajectory Planning Algorithm for Multi-UAV-Assisted MEC System. *Soft Comput.* **2022**, *26*, 7479–7492. [CrossRef]
17. Pehlivanoglu, Y.V.; Pehlivanoglu, P. An Enhanced Genetic Algorithm for Path Planning of Autonomous UAV in Target Coverage Problems. *Appl. Soft Comput.* **2021**, *112*, 107796. [CrossRef]
18. Abhishek, B.; Ranjit, S.; Shankar, T.; Eappen, G.; Sivasankar, P.; Rajesh, A. Hybrid PSO-HSA and PSO-GA Algorithm for 3D Path Planning in Autonomous UAVs. *SN Appl. Sci.* **2020**, *2*, 1805. [CrossRef]
19. Tang, Q.; Wen, S.; He, S.; Yang, K. Multi-UAV-Assisted Offloading for Joint Optimization of Energy Consumption and Latency in Mobile Edge Computing. *IEEE Syst. J.* **2024**, *18*, 1414–1425. [CrossRef]
20. Mousa, M.H.; Hussein, M.K. Efficient UAV-based Mobile Edge Computing Using Differential Evolution and Ant Colony Optimization. *PeerJ Comput. Sci.* **2022**, *8*, e870. [CrossRef]
21. Samriya, J.K.; Kumar, M.; Tiwari, R. Energy-aware ACO-DNN Optimization Model for Intrusion Detection of Unmanned Aerial Vehicle (UAVs). *J. Ambient Intell. Humaniz. Comput.* **2023**, *14*, 10947–10962. [CrossRef]
22. Shurrab, M.; Mizouni, R.; Singh, S.; Otrok, H. Reinforcement Learning Framework for UAV-based Target Localization Applications. *Internet Things* **2023**, *23*, 100867. [CrossRef]
23. Yu, Z.; Hu, J.; Min, G.; Zhao, Z.; Miao, W.; Hossain, M.S. Mobility-aware proactive edge caching for connected vehicles using federated learning. *IEEE Trans. Intell. Transp. Syst.* **2020**, *22*, 5341–5351. [CrossRef]

24. AlShathri, S.I.; Chelloug, S.A.; Hassan, D.S. Parallel Meta-Heuristics for Solving Dynamic Offloading in Fog Computing. *Mathematics* **2022**, *10*, 1258. [CrossRef]
25. Wei, D.; Wang, R.; Xia, C.; Xia, T.; Jin, X.; Xu, C. Edge Computing Offloading Method Based on Deep Reinforcement Learning for Gas Pipeline Leak Detection. *Mathematics* **2022**, *10*, 4812. [CrossRef]
26. Mei, J.; Dai, L.; Tong, Z.; Deng, X.; Li, K. Throughput-Aware Dynamic Task Offloading Under Resource Constant for MEC with Energy Harvesting Devices. *IEEE Trans. Netw. Serv. Manag.* **2023**, *20*, 3460–3473. [CrossRef]
27. Ren, J.; Liu, J.; Zhang, Y.; Li, Z.; Lyu, F.; Wang, Z.; Zhang, Y. An Efficient Two-Layer Task Offloading Scheme for MEC System with Multiple Services Providers. In Proceedings of the IEEE Conference on Computer Communications, Online, 2–5 May 2022; pp. 1519–1528.
28. Deng, T.; Chen, Y.; Chen, G.; Yang, M.; Du, L. Task Offloading Based on Edge Collaboration in MEC-Enabled IoV Networks. *J. Commun. Netw.* **2023**, *25*, 197–207. [CrossRef]
29. Liu, T.; Guo, D.; Xu, Q.; Gao, H.; Zhu, Y.; Yang, Y. Joint Task Offloading and Dispatching for MEC with Rational Mobile Devices and Edge Nodes. *IEEE Trans. Cloud Comput.* **2023**, *11*, 3262–3273. [CrossRef]
30. Zeng, F.; Tang, J.; Liu, C.; Deng, X.; Li, W. Task-Offloading Strategy Based on Performance Prediction in Vehicular Edge Computing. *Mathematics* **2022**, *10*, 1010. [CrossRef]
31. Tung, T.V.; An, T.T.; Lee, B.M. Joint Resource and Trajectory Optimization for Energy Efficiency Maximization in UAV-Based Networks. *Mathematics* **2022**, *10*, 3840. [CrossRef]
32. Arif, M.; Kim, W. Analysis of Fluctuating Antenna Beamwidth in UAV-Assisted Cellular Networks. *Mathematics* **2023**, *11*, 4706. [CrossRef]
33. Elgendy, I.A.; Meshoul, S.; Hammad, M. Joint Task Offloading, Resource Allocation, and Load-Balancing Optimization in Multi-UAV-Aided MEC Systems. *Appl. Sci.* **2023**, *13*, 2625. [CrossRef]
34. Zhu, A.; Lu, H.; Ma, M.; Zhou, Z.; Zeng, Z. DELOFF: Decentralized Learning-Based Task Offloading for Multi-UAVs in U2X-Assisted Heterogeneous Networks. *Drones* **2023**, *7*, 656. [CrossRef]
35. Xu, Y.; Deng, F.; Zhang, J. UDCO-SAGiMEC: Joint UAV Deployment and Computation Offloading for Space–Air–Ground Integrated Mobile Edge Computing. *Mathematics* **2023**, *11*, 4014. [CrossRef]
36. Tian, J.; Wang, D.; Zhang, H.; Wu, D. Service Satisfaction-Oriented Task Offloading and UAV Scheduling in UAV-Enabled MEC Networks. *IEEE Trans. Wirel. Commun.* **2023**, *22*, 8949–8964. [CrossRef]
37. Xia, J.; Wang, P.; Li, B.; Fei, Z. Intelligent Task Offloading and Collaborative Computation in Multi-UAV-Enabled Mobile Edge Computing. *China Commun.* **2022**, *19*, 244–256. [CrossRef]
38. Chen, Z.; Zheng, H.; Zhang, J.; Zheng, X.; Rong, C. Joint Computation Offloading and Deployment Optimization in Multi-UAV-Enabled MEC Systems. *Peer Netw. Appl.* **2022**, *15*, 194–205. [CrossRef]
39. Liu, B.; Liu, C.; Peng, M. Resource Allocation for Energy-Efficient MEC in NOMA-Enabled Massive IoT Networks. *IEEE J. Sel. Areas Commun.* **2020**, *39*, 1015–1027. [CrossRef]
40. Liu, Y.; Xiong, K.; Ni, Q.; Fan, P.; Letaief, K.B. UAV-Assisted Wireless Powered Cooperative Mobile Edge Computing: Joint Offloading, CPU Control, and Trajectory Optimization. *IEEE Internet Things J.* **2019**, *7*, 2777–2790. [CrossRef]
41. Li, L.; Lin, Q.; Ming, Z.; Wong, K.C.; Gong, M.; Coello, C.A.C. An Immune-Inspired Resource Allocation Strategy for Many-Objective Optimization. *IEEE Trans. Syst. Man Cybern. Syst.* **2023**, *53*, 3284–3297. [CrossRef]
42. Raquel, C.R.; Naval, P.C., Jr. An Effective Use of Crowding Distance in Multiobjective Particle Swarm Optimization. In Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, Washington, DC, USA, 25–29 June 2005; pp. 257–264.
43. Liu, S.; Li, J.; Lin, Q.; Tian, Y.; Tan, K.C. Learning to Accelerate Evolutionary Search for Large-scale Multiobjective Optimization. *IEEE Trans. Evol. Comput.* **2022**, *27*, 67–81. [CrossRef]
44. Li, L.; Li, Y.; Lin, Q.; Liu, S.; Zhou, J.; Ming, Z.; Coello, C.A.C. Neural Net-Enhanced Competitive Swarm Optimizer for Large-Scale Multiobjective Optimization. *IEEE Trans. Cybern.* **2023**, *54*, 3502–3515. [CrossRef] [PubMed]
45. Zhu, J.; Wang, X.; Huang, H.; Cheng, S.; Wu, M. A NSGA-II Algorithm for Task Scheduling in UAV-Enabled MEC System. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 9414–9429. [CrossRef]
46. Yao, Z.; Wu, H.; Chen, Y. Multi-Objective Cooperative Computation Offloading for MEC in UAVs Hybrid Networks via Integrated Optimization Framework. *Comput. Commun.* **2023**, *202*, 124–134. [CrossRef]
47. Tian, Y.; Zheng, X.; Zhang, X.; Jin, Y. Efficient Large-Scale Multiobjective Optimization Based on A Competitive Swarm Optimizer. *IEEE Trans. Cybern.* **2019**, *50*, 3696–3708. [CrossRef]
48. Zhu, X.; Zhou, M. Multiobjective Optimized Cloudlet Deployment and Task Offloading for Mobile-Edge Computing. *IEEE Internet Things J.* **2021**, *8*, 15582–15595. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.