sensors

MDPI

Article

JUTAR: Joint User-Association, Task-Partition, and Resource-Allocation Algorithm for MEC Networks

Ling Kang [1,2], Yi Wang [1,2,*], Yanjun Hu [1], Fang Jiang [1], Na Bai [1] and Yu Deng [1,2]

[1] Information Materials and Intelligent Sensing Laboratory of Anhui Province, Anhui University, Hefei 230601, China
[2] National Engineering Research Center for Agro-Ecological Big Data Analysis & Application, Anhui University, Hefei 230601, China
* Correspondence: yiwang@ahu.edu.cn

Abstract: Mobile edge computing (MEC) is a promising technique to support the emerging delay-sensitive and compute-intensive applications for user equipment (UE) by means of computation offloading. However, designing a computation offloading algorithm for the MEC network to meet the restrictive requirements towards system latency and energy consumption remains challenging. In this paper, we propose a joint user-association, task-partition, and resource-allocation (JUTAR) algorithm to solve the computation offloading problem. In particular, we first build an optimization function for the computation offloading problem. Then, we utilize the user association and smooth approximation to simplify the objective function. Finally, we employ the particle swarm algorithm (PSA) to find the optimal solution. The proposed JUTAR algorithm achieves a better system performance compared with the state-of-the-art (SOA) computation offloading algorithm due to the joint optimization of the user association, task partition, and resource allocation for computation offloading. Numerical results show that, compared with the SOA algorithm, the proposed JUTAR achieves about 21% system performance gain in the MEC network with 100 pieces of UE.

Keywords: mobile edge computing (MEC); computation offloading; joint optimization; system overhead

Citation: Kang, L.; Wang, Y.; Hu, Y.; Jiang, F.; Bai, N.; Deng, Y. JUTAR: Joint User-Association, Task-Partition, and Resource-Allocation Algorithm for MEC Networks. *Sensors* **2023**, *23*, 1601. https://doi.org/10.3390/s23031601

Academic Editor: Anfeng Liu

Received: 18 December 2022
Revised: 19 January 2023
Accepted: 19 January 2023
Published: 1 February 2023

1. Introduction

Various mobile applications facilitate people's lives but bring very high requirements for mobile user equipment (UE). Admittedly, UE enjoys more and more powerful computing process units (CPU) nowadays, but they still can not handle the numerous computation tasks given a strict low-latency restriction due to their inherent hardware resource limits. Since the base station (BS) holds hundreds of times the computation capability of the UE, it can help UE solve heavy computation tasks. In particular, UE can offload all or part of the computation-intensive tasks to the BS, relieving its local computation burden. Based on this idea, several techniques have recently been proposed to solve the computation overload problem with respect to UE [1]. The mobile cloud computing (MCC) [2] technique works by uploading the tasks of UE to the cloud servers at the macro BS (MBS), which possesses very powerful computation capabilities and can finish the uploaded tasks within a short time. However, the transmission latency for the UE may be unacceptable if the UE suffers from a far distance from the MBS. To address this bottleneck, a mobile edge computing (MEC) [3,4] technique is further proposed as a promising approach for this issue. By deploying the small BS (SBS) involving servers at the edge of the network, each UE can upload its computation tasks to the nearest SBS, thus not only mitigating its computation load but also reducing the transmission latency.

1.1. Related Works

Drawing insights from MEC literature, the computation offloading strategy ranks as a dominant issue in the MEC network. Refs. [5–7] considered the computation offloading

issue with the binary offloading strategy, in which all computation tasks for each UE are either computed locally or uploaded to servers. The binary offloading strategy enjoys low computational complexity but causes a high awaiting latency for each UE due to its principle that the computation tasks can not be processed until all of the tasks are uploaded. To overcome this drawback, the partial computation offloading strategy was proposed in [8–11], in which some computation tasks for each UE are computed locally, and other computation tasks are uploaded to servers. The partial computation offloading strategy allows performing the computation offloading and task processing in parallel, thus reducing the system latency. In particular, Refs. [8–11] merely aimed at optimizing a single key parameter indicator (KPI), e.g., energy consumption or system latency, whereas [11] investigated the joint optimization of several KPIs to achieve the best system overhead. However, Refs. [8–11] only considered MEC networks with one edge server, which bottlenecks the system's computational capability. Fortunately, this problem can be solved by deploying densely distributed servers [12,13] in MEC networks. The main issues considered in the MEC network with densely distributed servers are user association, task partition, and resource allocation [14–17]. In particular, Ref. [14] solved the user association problem according to the bandwidth, power, and interference. Ref. [15] considered the user association issue according to the data size of the uploaded tasks. Refs. [16,17] jointly considered the task partition and resource allocation issues to not only balance the overload of each server but also exploit the computational capability of each server more efficiently. Recently, deep learning techniques were extensively employed to optimize the computation offloading problem [18–22]. Ref. [18] used the genetic algorithm (GA) to decide whether the tasks of each UE are offloaded to SBS or MBS. Ref. [19] enhanced the decision strategy by comprehensively considering the GA and each UE's offloading prior probability. Ref. [20] relied on deep reinforcement learning (DRL) to solve the resource allocation issue. Ref. [21] employed the GA to save energy consumption and further considered each UE's mobility. Ref. [22] exploited the LSTM network to further improve the system performance by considering each UE's direction.

1.2. Motivation and Contributions

Thus far, the existing computation offloading schemes still suffer from several concerns. First, the user association should not only consider the bandwidth, power, and interference for each UE but also evaluate the data size of the tasks and the channel quality [14,15]. Second, the traditional algorithms employed the task partition and resource allocation shows low convergence speeds, which bottleneck the system performance [16,17]. Finally, the existing computation offloading schemes usually optimize the user association, the task partition and the resource allocation problem, respectively, which could miss the optimal solution for the overall MEC network [9,12]. The performance of the computation offloading scheme is dominated by the user-association together with the task-partition and the resource-allocation strategy, and this knowledge motivates us to think of an approach to jointly optimize the user-association, task-partition, and resource-allocation issues. In this paper, we propose a joint user-association, task-partition, and resource-allocation (JUTAR) algorithm to solve the computation offloading problem in the MEC network with densely distributed servers. The main contributions of this work are listed as follows:

- We build up the optimization function with respect to the joint user-association, task-partition, and resource-allocation issues given an MEC network with massive servers. With the joint optimization of these problems, the optimization function could explore better results for the realistic MEC network.
- We define a user-association metric, which comprehensively considers the distance and overload of each UE and the target SBS, to indicate the user-association for each UE. In addition, we employed the smooth approximation to further simplify the optimization function.

- We propose using the particle swarm algorithm (PSA) to find the optimal results of the optimization function. The PSA can heuristically find the optimal solutions for the function and contribute to better system performance.

Numerical results also demonstrate that the proposed JUTAR algorithm enjoys the lowest system overhead compared with the existing computation offloading schemes given different parameters (e.g., number of users, data size of tasks).

The rest of this paper is organized as follows. Section 2 introduces the system model and the problem formulation of MEC networks. Section 3 discusses the details of the proposed JUTAR algorithm. Section 4 provides numerical results to validate the proposed algorithm. Section 5 concludes this work.

2. System Model and Problem Formulation

Figure 1 shows the model of the MEC network, involving an MBS, several SBSs, and various types of UE. Each BS is equipped with an MEC server. The SBSs and UE are randomly distributed within the coverage of the MBS. In addition, all of the SBSs are connected to the MBS via wired links. Supposing there are $M+1$ BSs (including the MBS), and N pieces of UE, let $\boldsymbol{S} = \{S_0, S_1, \cdots, S_M\}$ and $\boldsymbol{U} = \{u_1, u_2, \cdots, u_N\}$ denote the set of the BSs and the UE, respectively, where S_0 represents the MBS, S_m ($m \in \{1, 2, ..., M\}$) denotes the m-th SBS, and u_n ($n \in \{1, 2, ..., N\}$) denotes the n-th UE. Although each user could detect several SBSs in this scenario, we only consider the case that each user associates with one SBS for simplicity. $a_{m,n}$ indicates that UE n is associated with SBS m, and we have

$$a_{m,n} = \begin{cases} 1, & \text{if } n\text{-th UE} \in m\text{-th SBS}, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

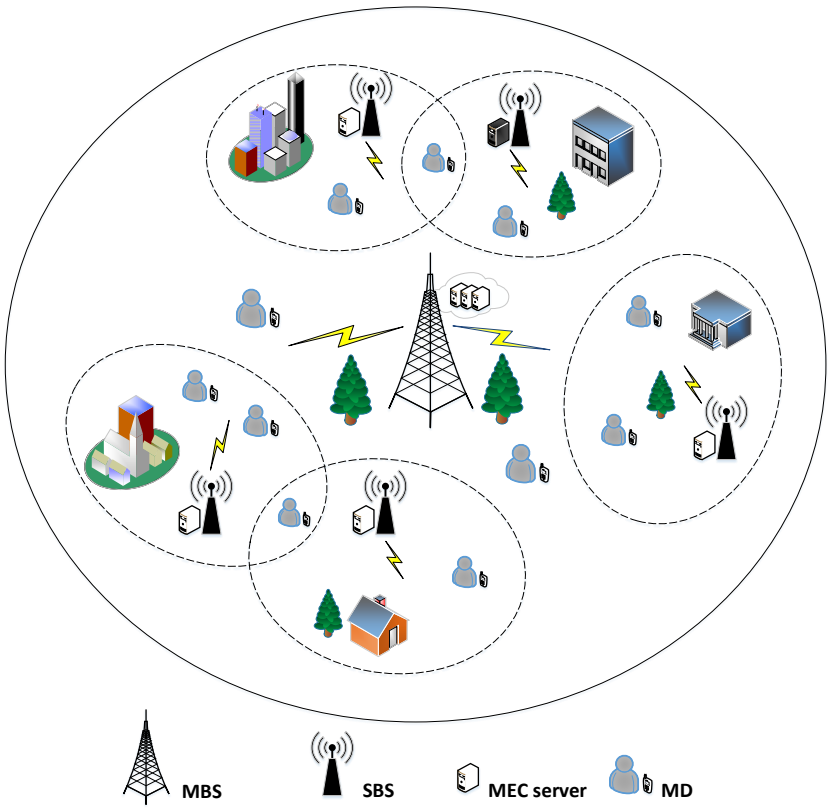

Figure 1. System model of the MEC network.

In the MEC network system, a computational task can be divided into several independent subtasks, which are addressed by the MEC servers and the local devices, respectively. The task requirement of the n-th UE is denoted as $I_n =< d_n, c_n, t_n^{\max} >$, where d_n is the

data size of the task, c_n is the CPU clock cycles required to process a bit of data in the task, $t_n^{\max}$ is the maximum latency of this task. Therefore, the task I_n requires $c_n d_n$ clock cycles for the n-th UE.

Generally, computational offloading has two different solutions. On the one hand, as Figure 2a shows, if the UE is directly connected to the MBS, its computational task is divided into two parts: (1) local computational task, denoted as x_n^{l}, and (2) remote computational task, denoted as x_n^{c}, where $x_n^{\mathrm{l}} + x_n^{\mathrm{c}} = 1$. On the other hand, if the UE is connected into an SBS, its computational tasks are processed in part by the local device (local computation, denoted as x_n^{l}), in part by the SBS (computational offloading, denoted as $x_{m,n}^{\mathrm{e}}$), and in part by the MBS (further computational offloading, denoted as x_n^{c}). Figure 2b shows the offloading strategy of the UE corresponding to an SBS, and the processed tasks satisfy $x_n^{\mathrm{l}} + x_{m,n}^{\mathrm{e}} + x_n^{\mathrm{c}} = 1$. Since the local tasks and the offloading tasks are processed in parallel, the system latency is dominated by the part with the highest latency. However, energy consumption is the accumulation of energy consumption for different parts.

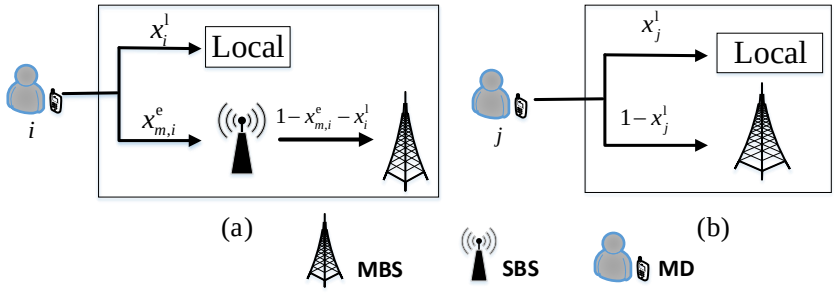

Figure 2. Computational offloading strategies for the UE in different scenarios: (**a**) only associated with an MBS, (**b**) associated with the SBS and MBS.

2.1. Local Computing Model

Let t_n^{l} denote the latency of the local tasks, which can be written as

$$t_n^{\mathrm{l}} = \frac{x_n^{\mathrm{l}} c_n d_n}{f_n^{\mathrm{l}}}, \tag{2}$$

where f_n^{l} is the computational capability of the n-th UE, measured in CPU clock cycles per second. Let ς denote the energy coefficient depending on CPU chip architecture, then the power consumption of the CPU is expressed as $\varsigma\left(f_n^{\mathrm{l}}\right)^3$. Hence, the execution energy for this task (denoted as E_n^{l}).

$$E_n^{\mathrm{l}} = x_n^{\mathrm{l}} c_n d_n \varsigma \left(f_n^{\mathrm{l}}\right)^2. \tag{3}$$

2.2. Computing Model in SBS

The tasks with respect to the n-th UE and processed in the m-th SBS are divided into two parts. In particular, the first part will be offloaded into a corresponding MEC server, and the second part is processed by the SBS itself. Let the notation B, σ^2, p_n and $h_{m,n}$ denote the system bandwidth, noise power, transmit power of the n-th UE, and channel coefficient between the n-th UE and m-th SBS, respectively, then the transmission rate of the n-th UE is

$$r_{m,n} = B \log_2\left(1 + \frac{P_n h_{m,n}}{\sigma^2}\right) \text{(bits/s)}. \tag{4}$$

In addition, let the notation $f_{m,n}$ represent the computational capacity (clock cycles per second) of the m-th SBS, and the maximum computational capacity of the m-th SBS is

$f_{m,n}^{\max}$. With the notation defined above, the latency $t_{m,n}^{\mathrm{e}}$ and energy consumption $E_{m,n}^{\mathrm{e}}$ of the tasks with respect to the n-th UE and processed in the m-th SBS can be written as

$$t_{m,n}^{\mathrm{e}} = a_{m,n} x_n^{\mathrm{e}} \left(\frac{d_n}{r_{m,n}} + \frac{c_n d_n}{f_{m,n}} \right), \tag{5}$$

$$E_{m,n}^{\mathrm{e}} = a_{m,n} x_n^{\mathrm{e}} \left(p_n \frac{d_n}{r_{m,n}} + c_n d_n e_{\mathrm{s}} \right), \tag{6}$$

where e_s is the energy consumption of m-th SBS per CPU cycle clock. It is noted that the latency and energy consumption of the feedback operation from the MEC sever are neglected since the data size of the feedback is negligible compared with the transmitted data [7].

2.3. Computing Model for the MBS

The latency of the tasks associated with x_n^c consists of the transmission latency and the processing latency. In detail, if the n-th UE is directly connected to the MBS, the transmission rate is

$$r_{0,n} = B \log_2 \left(1 + \frac{P_n h_{0,n}}{\sigma^2} \right) \text{ (bits/s)}, \tag{7}$$

where $h_{0,n}$ is the channel gain between the n-th UE and the MBS. On the contrary, if the n-th UE is linked with an SBS, the transmission rate is $r_{m,n}$ defined above. In addition to the latency transmitting the data from the n-th UE to the m-th SBS, the overall transmission latency also involves the latency offloading the data from the m-th SBS to the MBS, which is represented as $\gamma x_n^{\mathrm{c}} a_{m,n} d_n$. Here, γ is a factor of the transmission via the wired line. Let the notation κ be the indicator about whether the n-th UE is directly connected to the MBS (1 for YES and 0 for NO), then the overall latency about the tasks x_n^c is

$$t_n^{\mathrm{c}} = \frac{x_n^{\mathrm{c}} d_n}{r_n} + (1-\kappa)\gamma x_n^{\mathrm{c}} a_{m,n} d_n + \frac{x_n^{\mathrm{c}} c_n d_n}{F_{\mathrm{c}}}, \tag{8}$$

where F_{c} [clock cycles/s] is the computing capacity of the MEC server corresponding to the MBS [10]. Following the same principle as the latency analysis, the overall energy consumption of the tasks associated with x_n^c is

$$E_n^{\mathrm{c}} = P_n \frac{x_n^{\mathrm{c}} d_n}{r_n} + (1-\kappa)\beta\gamma x_n^{\mathrm{c}} a_{m,n} d_n + x_n^{\mathrm{c}} c_n d_n e_{\mathrm{c}}, \tag{9}$$

where β is the transmission power consumption via a wired line e_{c} is the energy consumption per CPU clock cycle of the MBS [16].

Thus far, the overall latency of the tasks for the n-th UE is

$$t_n = \max\left(t_n^{\mathrm{l}}, t_{m,n}^{\mathrm{e}}, t_n^{\mathrm{c}} \right), \tag{10}$$

and the overall energy consumption for processing the tasks of the n-th UE is expressed as

$$E_n = \left(E_n^{\mathrm{l}} + E_{m,n}^{\mathrm{e}} + E_n^{\mathrm{c}} \right). \tag{11}$$

Table 1 concludes the notations used in this paper.

Table 1. Notations in this work.

Notations	Description
u_n	n-th UE
S_m	m-th SBS
$a_{m,n}$	Indicator of if the n-th UE is associated with the m-th SBS
d_n	Data size of the task
c_n	Required CPU clock cycles to process a bit of data in the task
$t_n^{\max}$	Maximum tolerance latency of this task
t_n^{l}	Latency of the local tasks
E_n^{l}	Energy consumption of the local tasks
f_n^{l}	Computational capability of the n-th UE
$r_{m,n}$	Transmission rate of the n-th UE
$t_{m,n}^{\mathrm{e}}$	Latency of the tasks of the n-th UE and processed in the m-th SBS
$E_{m,n}^{\mathrm{e}}$	Energy consumption of the tasks of the n-th UE and processed in the m-th SBS
t_n^{c}	Latency of the tasks of the n-th UE and processed in the MBS
E_n^{c}	Energy consumption of the tasks of the n-th UE and processed in the MBS
t_n	Overall latency of the tasks for the n-th UE
E_n	Overall energy consumption of the tasks of the n-th UE
x_n	Task offloading ratio
$b_{\phi_k,n}$	UE association metric
load_{ϕ_k}	Overload of the ϕ_k-th SBS

3. Methodology

Since the latency and the energy consumption serve as the key parameter indicator (KPI) for the MEC network, designing an offloading strategy that can balance those two KPIs becomes demanding. Considering the trade-off of the latency and the energy consumption, the optimization problem of the computation offloading is formulated as

$$\begin{aligned}
\mathrm{P1}: \min_{a_{m,n},\mathbf{x}_n,f_{m,n}} & \sum_{n=1}^{N}(t_n+\lambda E_n), \\
\text{s.t.} \quad & t_n \le t_n^{\max}, && \text{(C1)} \\
& \sum_{m=1}^{M} a_{m,n} \le 1, a_{m,n} \in \{0,1\}, && \text{(C2)} \\
& x_n^{\mathrm{l}} + x_{m,n}^{\mathrm{e}} + x_n^{\mathrm{c}} = 1, \forall m \in M, n \in N, && \text{(C3)} \\
& f_{m,n} > 0, \forall m \in M, n \in N, && \text{(C4)} \\
& \sum_{n \in N} f_{m,n} \le f_m^{\max}, \forall m \in M, n \in N && \text{(C5)}
\end{aligned} \tag{12}$$

Here, λ denotes the weight coefficients corresponding to the latency and energy consumption, and (C1) limits the maximum latency. (C2) guarantees that each UE can

be associated with at most one SBS. (C3) promises that the overall tasks of the n-th UE are normalized as one. (C4) and (C5) circumvent the scenarios in that the required computational resource of the connected UE exceeds the overall computational capacity for the n-th SBS. Unfortunately, the optimal solution of the P1 problem can not be computed directly since there are several strongly coupled variables, and the problem is built by a non-convex nonlinear function. To this end, heuristic methods are employed in this section. In particular, the P1 problem is first simplified by associating each UE with the appropriate SBS and then approximated by utilizing a smooth approximation equation. Finally, a PSA algorithm is developed to find the optimal solution to the problem.

3.1. User Association

Supposing the network involves M SBSs and N users, let $\mathbf{A}$ denote the user-association matrix with dimensions of $M \times N$, written as

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,N} \\ \vdots & \vdots & \vdots & \vdots \\ a_{M,1} & a_{M,2} & \cdots & a_{M,N} \end{bmatrix} \tag{13}$$

where $a_{i,j}$ represents whether the j-th user is associated with the i-th SBS. Since the i-th user may be covered by several SBSs at the same time, the i-th column of the $\mathbf{A}$ could involve K, $1 \leq K \leq M$ unitary elements. Let $\mathbf{\Phi}_n = \{\phi_1, \phi_2, ..., \phi_K\}$ denote a vector consisting of the index of the unitary element in the n-th column of the matrix $\mathbf{A}$. The user association for each UE should comprehensively consider: (1) the distance between the UE and the target SBS, and (2) the overload of the SBS. Based on this, we denote $b_{\phi_k,n}$ as the association metric with respect to each SBS. The $b_{\phi_k,n}$ is illustrated as

$$b_{\phi_k,n} = \frac{h_{\phi_k,n}}{1 + \frac{\text{load}_{\phi_k}}{\sum_k \text{load}_{\phi_k}}}, k = \{1, 2, \cdots, K\}. \tag{14}$$

where load_{ϕ_k} denotes the overload of the ϕ_k-th SBS and $h_{\phi_k,n}$ is the channel coefficient between the n-th UE and the ϕ_k-th SBS. The channel coefficient $h_{\phi_k,n}$ follows the Rayleigh fading channel model, and is computed by

$$h_{\phi_k,n} = 140.7 + 36.7 \ \log d, \tag{15}$$

where d is the distance between each pair of UE and the target SBS. It is observed that the larger the $b_{\phi_k,n}$, the more reliable the SBS. Hence, each UE will be associated with the corresponding SBS with the largest association metric. Since the UE may also be covered by the MBS, they can be categorized into two groups. Let $N_{m,0}$ denote the first group of UE, in which the offloading tasks for each UE can only be processed by the MBS. Denote $N^c_{m,0}$ as the second group of UE, which is the complementary set of $N_{m,0}$. The offloading tasks for each UE in $N^c_{m,0}$ are addressed in part by the SBS and in part by the MBS. Hence, the P1 optimization problem can be simplified as

$$\begin{aligned} \text{P2}: & \min_{\mathbf{x}_n} \sum_{n \in N_{m,0}} \left(\max\left(t^{\text{l}}_n, t^{\text{c}}_n\right) + \lambda \left(E^{\text{l}}_n + E^{\text{c}}_n\right)\right) + \\ & \min_{\mathbf{x_n}, f_{m,n}} \sum_{n \in N^c_{m,0}} \left(\max\left(t^{\text{l}}_n, t^{\text{e}}_{m,n}, t^{\text{c}}_n\right) + \lambda E_n\right), \\ & \text{s.t.} \quad \text{(C1)} \ - \ \text{(C5)} \ . \end{aligned} \tag{16}$$

For simplicity, we use $\mathcal{O}_M$ to denote the optimization function corresponding to the computation offloading tasks processed by the MBS and $\mathcal{O}_S$ to denote the optimization func-

tion corresponding to the computation offloading tasks addressed by the SBS. Therefore, we have

$$\begin{cases} \mathcal{O}_M = \min\limits_{\mathbf{x}_n} \sum\limits_{n \in N_{m,0}} \left(\max\left(t_n^{\mathrm{l}}, t_n^{\mathrm{c}} \right) + \lambda \left(E_n^{\mathrm{l}} + E_n^{\mathrm{c}} \right) \right), \\ \mathcal{O}_S = \min\limits_{\mathbf{x}_n, f_{m,n}} \sum\limits_{n \in N_{m,0}^c} \left(\max\left(t_n^{\mathrm{l}}, t_{m,n}^{\mathrm{e}}, t_n^{\mathrm{c}} \right) + \lambda E_n \right). \end{cases} \tag{17}$$

3.2. Smooth Approximation

Although the function $\mathcal{O}_M$ can be solved by employing the linear optimization toolbox, the solution of the P2 problem is still unavailable due to the nonlinear optimization function $\mathcal{O}_S$ involving the variables $\mathbf{x}_n$ and $f_{m,n}$. Hence, heuristic approaches (e.g., PSA algorithm) can be employed to find optimal the solutions to $\mathcal{O}_S$. However, the max function in $\mathcal{O}_S$ involves tree variables, which makes it hard to derive the optimal results theoretically, hindering the application of the heuristic approaches. Fortunately, the max function can be eliminated by the smooth approximation function for the mathematical uniform norm [23]. Let $\mathcal{O}_S' = \min\limits_{\mathbf{x}_n, f_{m,n}} \sum\limits_{n \in N_{m,0}^c} \max\left(t_n^{\mathrm{l}}, t_{m,n}^{\mathrm{e}}, t_n^{\mathrm{c}} \right)$, and then it can be rewritten as

$$\mathcal{O}_S' = \min_{\boldsymbol{u} \in \Phi^n} \max(\boldsymbol{u}) = \min_{\boldsymbol{u} \in \Phi^n} \|\boldsymbol{u}\|_\infty, \tag{18}$$

where $\boldsymbol{u}$ is the vector involving all of the possible choices with respect to $(t_n^{\mathrm{l}}, t_{m,n}^{\mathrm{e}}, t_n^{\mathrm{c}})$. According to the property of the uniform norm, by introducing the entropy function, Equation (18) can be approximated by

$$\min_{\boldsymbol{u} \in \Phi^n} F_\mu(\boldsymbol{u}) = \mu \ln \left[\left(\sum_{i=1}^{n} \left(e^{\frac{u_i}{\mu}} + e^{-\frac{u_i}{\mu}} \right) \right) \right], \tag{19}$$

where μ is the smoothing factor. Therefore, the $\max\left(t_n^{\mathrm{l}}, t_{m,n}^{\mathrm{e}}, t_n^{\mathrm{c}}\right)$ can be rewritten as

$$\mu \ln \left[e^{\frac{t_n^{\mathrm{l}}}{\mu}} + e^{-\frac{t_n^{\mathrm{l}}}{\mu}} + e^{\frac{t_{mn}^{e}}{\mu}} + e^{-\frac{t_{mn}^{e}}{\mu}} + e^{\frac{t_n^{c}}{\mu}} + e^{-\frac{t_n^{c}}{\mu}} \right]. \tag{20}$$

Replacing the $\max\left(t_n^{\mathrm{l}}, t_{m,n}^{\mathrm{e}}, t_n^{\mathrm{c}}\right)$ with (20), the P2 problem can be converted into P3 problem, illustrated as

$$\begin{aligned} \mathrm{P3}: & \min_{\mathbf{x}_n} \sum_{n \in N_{m,0}} \left(\max\left(t_n^{\mathrm{l}}, t_n^{\mathrm{c}} \right) + \lambda \left(E_n^{\mathrm{l}} + E_n^{\mathrm{c}} \right) \right) + \\ & \min_{\mathbf{x}_n, f_{m,n}} \sum_{n \in N_{m,0}^c} \left(\mu \ln \left[e^{\frac{t_n^{\mathrm{l}}}{\mu}} + e^{-\frac{t_n^{\mathrm{l}}}{\mu}} + e^{\frac{t_{m,n}^{\mathrm{e}}}{\mu}} + \right. \right. \\ & \left. \left. e^{-\frac{t_{m,n}^{\mathrm{e}}}{\mu}} + e^{\frac{t_n^{\mathrm{c}}}{\mu}} + e^{-\frac{t_n^{\mathrm{c}}}{\mu}} \right] + \lambda E_n \right) \\ & \text{s.t.} \quad \text{(C1)} - \text{(C5)}. \end{aligned} \tag{21}$$

3.3. Optimized with PSA Algorithm

With the derivations above, the joint optimization problem for the computation offloading in a single SBS scenario is simplified as the P3 function, which can be solved with the PSA algorithm [24]. The PSA algorithm enjoys the advantages of rapid convergence and low complexity. In addition, it can circumvent the local optimal results and achieve the approximated global optimal point. The details of employing the PSA algorithm to solve the P3 problem is listed as follows:

(1) Initialization : Let $\Theta_m = \{\theta_1, \theta_2, ..., \theta_{|N_{m,0}^c|}\}$ denote a particle for the m-th single SBS, where θ_k represents the computation offloading strategy for the k-th user involved in set

$N^c_{m,0}$. For the k-th user, recall that x^l_k and x^e_k denote the tasks processed locally and the tasks offloaded from the user to the single SBS, respectively. Given $\boldsymbol{x}_k$ and $f_{m,k}$, θ_k can be denoted as $\theta_k = (x^l_k, x^e_k, f_{m,k})$. Figure 3 shows a toy example for Θ_m with $|N^c_{m,0}| = 3$. For initialization, it is assumed that there are Q particles. In addition, let $V_m = \{v_1, v_2, ..., v_{|N^c_{m,0}|}\}$ represent the velocity towards to the particle Θ_m. In particular, the velocity with respect to the k-th user is $v_k = (v^l_k, v^e_k, v^f_k)$, where v^l_k, v^e_k, and v^f_k are initialized with random values. It is noted that all of the initial values must satisfy the constraints in the P1 problem.

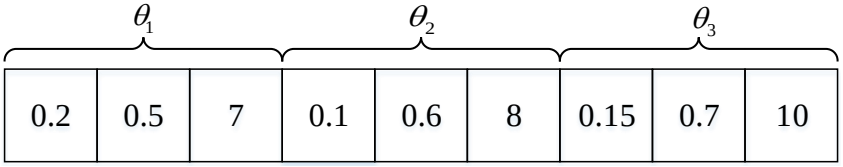

Figure 3. The position of the k-th particle.

(2) Update Fitness function: According to the formula of the P3 problem, the *Fitness function* is denoted as

$$\begin{aligned} \text{Fit} = \sum_{n \in N^c_{m,0}} \Bigg(\mu \ln \Bigg[& e^{\frac{t^l_n}{\mu}} + e^{-\frac{t^l_n}{\mu}} + e^{\frac{t^e_{m,n}}{\mu}} + e^{-\frac{t^e_{m,n}}{\mu}} + \\ & e^{\frac{t^c_n}{\mu}} + e^{-\frac{t^c_n}{\mu}} \Bigg] + \lambda E_n \Bigg) + \eta. \end{aligned} \tag{22}$$

where η is the penalty function for the particles. Once the particle violates the constraints, it will be penalized by the penalty function. According to the constraint in P1 problem, η is computed by

$$\eta = \sum_{n=1}^{N^c_{m,0}} \theta_n \left(\max \left(0, \sum_{n \in N} f_{m,n} - f^{\max}_m \right) \right). \tag{23}$$

where θ_n is the penalty factor. It is emphasized that the better the offloading strategy, the smaller the fitness function.

(3) Update particle: In the t-th iteration, Θ_m and V_m will be updated as

$$\begin{cases} V_m(t+1) = & W \cdot V_m(t) + C_1 \cdot R_1 \cdot (E_{\text{opt}} - \Theta_m(t)) \\ & + C_2 \cdot R_2 \cdot (G_{\text{opt}} - \Theta_m(t)), \\ \Theta_m(t+1) = & \Theta_m(t) + V_m(t+1), \end{cases} \tag{24}$$

where:

- C_1 and C_2 are two acceleration factors,
- W is a constant weight of the inertia,
- R_1 and R_2 are random factors chosen from $[0, 1]$,
- E_{opt} is the optimal position for the m-th particle in the single SBS scenario,
- G_{opt} is the optimal position for m-th particle in the global MEC network.

(4) Terminate algorithm: If the maximum number of the iterations is achieved, the algorithm will be terminated. On the contrary, the algorithm step (2) and step (3) are repeated.

3.4. Proposed JUTAR Algorithm

Thus far, the computation offloading problem in the single SBS-based scenario has been solved. However, there are multiple SBSs and massive users in the realistic MEC network. Therefore, the JUTAR algorithm is proposed in this subsection to solve the computation offloading problem in the overall MEC network.

Algorithm 1 shows the details of the JUTAR algorithm. In particular, first, each user in the network is associated with a single SBS according to the association metric computed

by (14). Second, the problem corresponding to computation offloading tasks directly processed by the MBS is solved by employing the linear toolbox. Third, the computation offloading problem for each user in the single SBS scenario is transformed from the P2 problem to the P3 problem by means of smooth approximation. Finally, the results of the problem associated with the tasks offloaded to the SBS are acquired by utilizing the PSA algorithm. With the JUTAR algorithm, the P1 problem in the overall MEC network is converted into the P4 problem, written as

$$\begin{aligned} \text{P4}: \min_{\mathbf{x}_n, f_{m,n}} & \sum_{m} \sum_{n \in N_m^{\text{rec}}} \Big(\max\Big(t_n^{\text{l}}, t_{m,n}^{\text{e}}, t_n^{\text{c}}\Big) + \\ & \lambda \Big(E_n^{\text{l}} + E_{m,n}^{\text{e}} + E_n^{\text{c}} \Big) \Big) + \\ & \min_{\mathbf{x}_n} \sum_{n=1}^{N_1} \Big(\max\Big(t_n^{\text{l}}, t_n^{\text{c}}\Big) + \lambda \Big(E_n^{\text{l}} + E_n^{\text{c}} \Big) \Big), \\ \text{s.t.} \quad & \text{(C1)} - \text{(C5)}. \end{aligned} \tag{25}$$

Note that although the P4 problem is derived from the P1 problem by simplifying the scenario of the network, it converges to the same results as the P1 problem [25].

Algorithm 1: JUTAR Algorithm

Input: the network parameters: N, M, $f^l = \left\{ f_1^l, f_2^l, \cdots, f_N^l \right\}$, $f = \{f_1, f_2 \cdots, f_M\}$. the algorithm parameters: Q, T, C_1, C_2, W, R_1 and R_2.
Output: $< \boldsymbol{A}_{M \times N}, \boldsymbol{x}_n, F >$.

for $u_i \in U$ **do**
  $\boldsymbol{\Phi}_i = \{\phi_1, \phi_2, \cdots, \phi_K\}$; //$u_i$ selects all reachable SBSs
  for $\Phi_j \in \boldsymbol{\Phi}_i$ **do**
    compute $b_{\Phi_j,i}$ according to Equation (14);
  $y = \arg\max_{\boldsymbol{\Phi}_i} \left(b_{\Phi_j,i} \right)$, $a_{y,i} = 1, \boldsymbol{\Phi}_i / \{a_{y,i}\} = 0$;
if $a_{\Phi_j,i} = 0$ **then**
  use the optimization toolbox for solving;
  else **for** $S_i \in \boldsymbol{S} / \{S_0\}$ **do**
    transform from the P2 problem to the P3 problem by means of smooth approximation.
    Initialization: Θ_m, $\boldsymbol{V}_k$, $t = 0$;
    while $t \leq T$ **do**
      for $q \in Q$ **do**
        Update Θ_m, $\boldsymbol{V}_q$, compute the fitness of the particle according to Equation (23);
      Update the individual optimal value and the global optimal value if $\text{Fit}_q^t \ll \text{Fit}_q^{t-1}$;
      t = t + 1;

4. Numerical Results

In this section, we provide numerical results to demonstrate the performance of the proposed JUTAR algorithm for the MEC network. We consider a network with coverage of 1.2 km × 1.2 km, involving one MBS, multiple SBSs, and multiple users. The MBS locates on the central point of the network, and the SBSs, together with the users, are randomly distributed. We follow the setup of the parameters in [18]. In particular, the computation input data size of subtask is $d_n \in [10, 12]$ Kbits , and the number of CPU cycles to execute one bit of the task is $c_n \in [1000, 1200]$ cycles/bit. The computational capabilities of the local device is $f_n^{\mathrm{l}} \in [4, 5] \times 10^8$ cycles/s, the maximum computational capabilities of the SBS is $f_m^{\mathrm{l}} \in [8, 10] \times 10^9$ cycles/s, and the maximum computational capabilities of MBS is $F_{\mathrm{c}} = 15 \times 10^9$ cycles/s. The transmit power for each user is set as 0.1 W. A Rayleigh fading channel is employed, and the channel coefficients are computed by (15). In addition, we set the energy coefficient with respect to the CPU as $\varsigma = 10^{-28}$. The energy consumption of both the SBS and the MBS are set as $e_s = e_c = 0.02$ W/GHz. Since the latency and the energy consumption use the dominant roles to measure the computation offloading algorithms, we comprehensively consider these two KPIs and propose the system overhead Ω as a new metric. The system overhead Ω is computed by

$$\Omega = \sum_{n=1}^{N} (t_n + \lambda E_n). \tag{26}$$

Without special explanation, the parameter λ is set as $\lambda = 0.5$. Consequently, the smaller the system overhead Ω, the better the computation offloading algorithm. To show the performance advantages of the proposed JUTAR algorithm, we provide comparison results of the existing algorithms as follows:

- All local processing (ALP) algorithm [8]: all tasks of the user are processed locally.
- Partial offloading strategy (POS) algorithm [8]: partial tasks of each UE are offloaded into either SBS or MBS based on a possible user association.
- All MBS processing (AMP) algorithm [10]: the overall tasks of each UE can only be uploaded to the MBS.
- GA [18]: partial tasks of each UE are uploaded to either SBS or MBS, which is decided according to the genetic algorithm.
- Priority offloading mechanism with joint offloading proportion and transmission (PROMOT) algorithm [19]: partial tasks of each UE are offloaded into SBS or MBS according to not only the GA algorithm but also its offloading prior probability.

Figure 4 shows the comparison results of the system overhead for the proposed JUTAR algorithm and other algorithms with different amounts of UE and date size of tasks. In particular, Figure 4a shows the system performance of the proposed JUTAR algorithm with various amounts of UE. It is observed that compared with existing algorithms, the proposed JUTAR algorithm shows comparable performance with a relatively small amount of UE but achieves the best results with a relatively large amount of UE. In particular, compared with the SOA GA algorithm, the proposed JUTAR algorithm saves about 21% system overhead given 100 pieces of UE in the MEC network. Figure 4b reports the system performance of the proposed JUTAR algorithm with various data sizes of tasks. It can be seen that the curves in Figure 4b have a consistent trend with that in Figure 4a. Compared with the GA algorithm, the proposed JUTAR algorithm saves about 33% system overhead when the offloaded tasks hold the size of 10,000 bits.

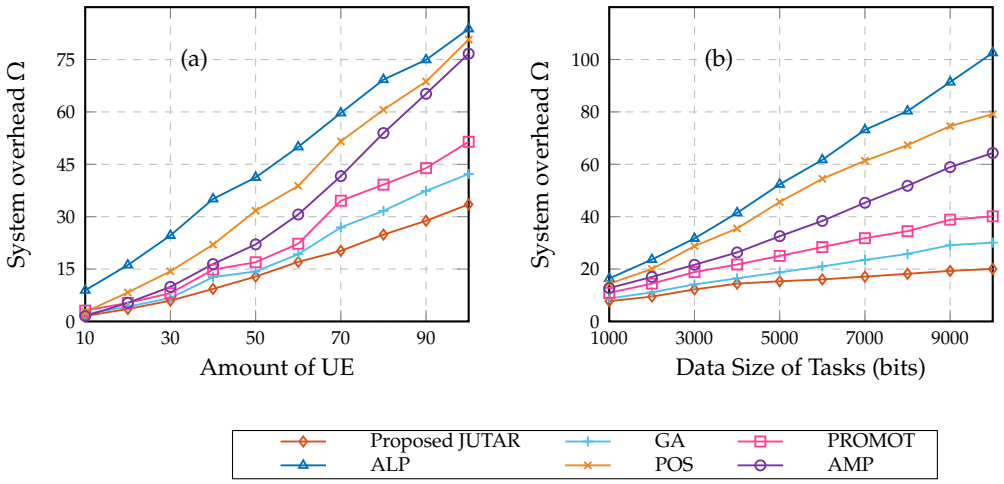

Figure 4. Comparison results of the system overhead vs. (**a**) amount of UE, (**b**) data size for the proposed JUTAR algorithm [8,10,18,19].

Figure 5 illustrates the system performance of the proposed JUTAR algorithm with 10–100 servers. From Figure 5, we can draw two observations. First, the system overhead of the proposed JUTAR algorithm degrades rapidly by increasing servers in the region with a relatively small number of servers but achieves a performance floor in the region with a relatively large number of servers. Second, although the proposed JUTAR algorithm suffers from a performance floor with a large number of servers, it still enjoys the best performance compared with the existing algorithms. From the numerical results reported in Figures 4 and 5, we can conclude that given a different amounts of UE, servers, and data sizes of tasks, the proposed JUTAR algorithm holds the best system performance compared with the existing algorithms.

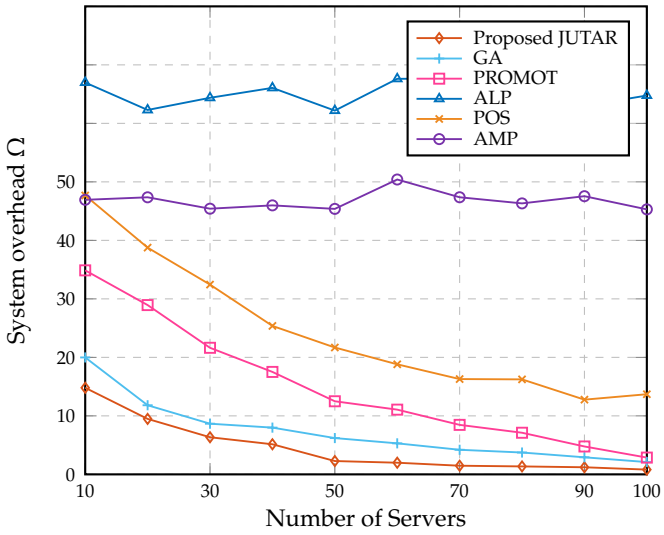

Figure 5. Comparison results of the system overhead vs. number of servers for the proposed JUTAR algorithm.

5. Conclusions

In this paper, we investigated the computation offloading problem in MEC networks with multiple SBSs and multiple pieces of UE. We proposed a JUTAR algorithm to solve the problem by jointly optimizing the user-association, task-partition, and resource-allocation issues. In particular, first, we simplified the network from a multiple SBS-based scenario to a single SBS-based scenario by employing the proposed user-association scheme. Second, for the single SBS-based scenario, we transformed the nonlinear optimization problem

into the linear problem by means of the smooth approximation. Finally, we solved the linear problem in the single SBS-based scenario by utilizing the PSA algorithm. Given a different amount of UE, servers, and data size of tasks, our algorithm shows the smallest system overhead compared with the existing algorithms. In future work, we will optimize the complexity of the algorithm and further consider the mobility of each UE for realistic MEC scenarios.

Author Contributions: L.K., Y.W. and Y.H. drafted the original manuscript and designed the experiments. L.K., Y.W. and Y.H. provided ideas and suggestions. F.J., N.B. and Y.D. provided a critical review and helped to draft the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Natural Science Foundation of China (No. 62071002) and the Open Research Fund of the National Engineering Research Center for Agro-Ecological Big Data Analysis and Application, Anhui University (No. AE202214).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Zhang, Y.; Dong, X.; Zhao, Y. Decentralized computation offloading over wireless-powered mobile-edge computing networks. In Proceedings of the IEEE International Conference on Artificial Intelligence and Information Systems (ICAIIS), Dalian, China, 20–22 March 2020; pp. 137–140.
2. Abbas, N.; Zhang, Y.; Taherkordi, A.; Skeie, T. Mobile edge computing: A survey. *IEEE Internet Things J.* **2017**, *5*, 450–465. [CrossRef]
3. Wei, X.; Wang, S.; Zhou, A.; Xu, J.; Su, S.; Kumar, S.; Yang, F. MVR: An architecture for computation offloading in mobile edge computing. In Proceedings of the 2017 IEEE International Conference on Edge Computing (EDGE), Honolulu, HI, USA, 25–30 June 2017; pp. 232–235.
4. Ren, J.; Yu, G.; He, Y.; Li, G.Y. Collaborative cloud and edge computing for latency minimization. *IEEE Trans. Veh. Technol.* **2019**, *68*, 5031–5044. [CrossRef]
5. Bi, S.; Zhang, Y.J. Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading. *IEEE Trans. Wirel. Commun.* **2018**, *17*, 4177–4190. [CrossRef]
6. Chen, L.; Li, X.; Ji, H.; Leung, V. Computation offloading balance in small cell networks with mobile edge computing. *Wirel. Netw.* **2019**, *25*, 4133–4145. [CrossRef]
7. Wu, Y.H.; Wang, Y.H.; Zhou, F.H.; Hu, R.O. Computation efficiency maximization in OFDMA-based mobile edge computing networks. *IEEE Commun. Lett.* **2019**, *24*, 159–163. [CrossRef]
8. Ning, Z.L.; Dong, P.R.; Kong, X.J.; Xia, F. A cooperative partial computation offloading scheme for mobile edge computing enabled internet of things. *IEEE Internet Things J.* **2018**, *6*, 4804–4814. [CrossRef]
9. Zhang, Y.; Fu, J.Q. Energy-efficient computation offloading strategy with tasks scheduling in edge computing. *Wirel. Netw.* **2021**, *27*, 609–620. [CrossRef]
10. Tao, X.Y.; Ota, K.; Dong, M.X.; Qi, H.; Li, K.Q. Performance guaranteed computation offloading for mobile-edge cloud computing. *IEEE Wirel. Commun. Lett.* **2017**, *6*, 774–777. [CrossRef]
11. Zhou, S.C.; Jadoon, W. The partial computation offloading strategy based on game theory for multi-user in mobile edge computing environment. *Comput. Netw.* **2020**, *178*, 107334. [CrossRef]
12. Feng, M.; Krunz, M.; Zhang, W. Joint task partitioning and user association for latency minimization in mobile edge computing networks. *IEEE Trans. Veh. Technol.* **2021**, *70*, 8108–8121. [CrossRef]
13. Xiao, Y.; Krunz, M. Distributed optimization for energy-efficient fog computing in the tactile internet. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 2390–2400. [CrossRef]
14. Han, Q.; Yang, B.; Miao, G.; Chen, C.; Wang, X.; Guan, X. Backhaul-aware user association and resource allocation for energy-constrained hetnets. *IEEE Trans. Veh. Technol.* **2016**, *66*, 580–593. [CrossRef]
15. Zhou, Y.; Yeoh, P.L.; Pan, C.; Wang, K.; Elkashlan, M.; Wang, Z.; Vucetic, B.; Li, Y. Offloading optimization for low-latency secure mobile edge computing systems. *IEEE Wirel. Commun. Lett.* **2019**, *9*, 480–484. [CrossRef]
16. Li, H.L.; Xu, H.T.; Zhou, C.C.; Lu, X.; Han, Z. Joint optimization strategy of computation offloading and resource allocation in multi-access edge computing environment. *IEEE Trans. Veh. Technol.* **2020**, *69*, 10214–10226. [CrossRef]
17. Tran, T.X.; Pompili, D. Joint task offloading and resource allocation for multi-server mobile-edge computing networks. *IEEE Trans. Veh. Technol.* **2018**, *68*, 856–868. [CrossRef]

18. Liao, Z.F.; Peng, J.; Xiong, B.; Huang, J.W. Adaptive offloading in mobile-edge computing for ultra-dense cellular networks based on genetic algorithm. *J. Cloud Comput.* **2021**, *10*, 15. [CrossRef]
19. Wang, J.; Wu, W.; Liao, Z.; Sherratt, R.S.; Kim, G.-J.; Alfarraj, O.; Alzubi, A.; Tolba, A. A probability preferred priori offloading mechanism in mobile edge computing. *IEEE Access* **2020**, *8*, 39758–39767. [CrossRef]
20. Li, J.; Gao, H.; Lv, T.; Lu, Y. Deep reinforcement learning based computation offloading and resource allocation for MEC. In Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), Barcelona, Spain, 15–18 April 2018; pp. 1–6.
21. Jehangiri, A.I.; Maqsood, T.; Umar, A.I.; Shuja, J.; Ahmad, Z.; Dhaou, I.B.; Alsharekh, M.F. LIMPO: Lightweight mobility prediction and offloading framework using machine learning for mobile edge computing. *Clust. Comput.* **2022**, 1–19.
22. Zaman, S.K.U.; Jehangiri, A.I.; Maqsood, T.; Umar, A.I.; Khan, M.A.; Jhanjhi, N.Z.; Shorfuzzaman, M.; Masud, M. COME-UP: Computation offloading in mobile edge computing with LSTM based user direction prediction. *Appl. Sci.* **2022**, *12*, 3312. [CrossRef]
23. Wang, R. Adjustable entropy method for solving convex inequality problem. *J. Syst. Eng. Electr.* **2009**, *20*, 1111–1114.
24. Huynh, L.N.T.; Pham, Q.V.; Pham, X.; Nguyen, T.D.T.; Hossain, M.D.; Huh, E.N. Efficient computation offloading in multi-tier multi-access edge computing systems: A particle swarm optimization approach. *Appl. Sci.* **2019**, *10*, 203. [CrossRef]
25. Tammer, K. The application of parametric optimization and imbedding to the foundation and realization of a generalized primal decomposition approach. *Math. Res.* **1987**, *35*, 376–386.