Article

Efficient Computation Offloading in Multi-Tier Multi-Access Edge Computing Systems: A Particle Swarm Optimization Approach

Luan N. T. Huynh ¹ , Quoc-Viet Pham ² , Xuan-Quy Pham ¹ , Tri D. T. Nguyen ¹ ,
Md Delowar Hossain ¹ and Eui-Nam Huh ^{1,*}

¹ Department of Computer Science and Engineering, Kyung Hee University, Yongin 17104, Korea; luanhnt@khu.ac.kr (L.N.T.H.); pxuanqui@khu.ac.kr (X.-Q.P.); tringuyendt@khu.ac.kr (T.D.T.N.); delowar@khu.ac.kr (M.D.H.)

² Research Institute of Computers, Information and Communication, Pusan National University, Busan 46241, Korea; vietpq@inje.ac.kr

* Correspondence: johnhuh@khu.ac.kr; Tel.: +82-31-201-3778

Received: 1 December 2019; Accepted: 22 December 2019; Published: 26 December 2019

Abstract: In recent years, multi-access edge computing (MEC) has become a promising technology used in 5G networks based on its ability to offload computational tasks from mobile devices (MDs) to edge servers in order to address MD-specific limitations. Despite considerable research on computation offloading in 5G networks, this activity in multi-tier multi-MEC server systems continues to attract attention. Here, we investigated a two-tier computation-offloading strategy for multi-user multi-MEC servers in heterogeneous networks. For this scenario, we formulated a joint resource-allocation and computation-offloading decision strategy to minimize the total computing overhead of MDs, including completion time and energy consumption. The optimization problem was formulated as a mixed-integer nonlinear program problem of NP-hard complexity. Under complex optimization and various application constraints, we divided the original problem into two subproblems: decisions of resource allocation and computation offloading. We developed an efficient, low-complexity algorithm using particle swarm optimization capable of high-quality solutions and guaranteed convergence, with a high-level heuristic (i.e., meta-heuristic) that performed well at solving a challenging optimization problem. Simulation results indicated that the proposed algorithm significantly reduced the total computing overhead of MDs relative to several baseline methods while guaranteeing to converge to stable solutions.

Keywords: computation offloading; heterogeneous networks; multi-access edge computing; particle swarm optimization; resource allocation

1. Introduction

With the tremendous growth of the Internet of Things (IoTs), massive amounts of mobile devices (MDs), such as smart devices and virtual reality (VR) glasses, connect to networks and generate large amounts of data on communications networks. Moreover, many new IoT applications, such as smart healthcare, surveillance, real-time online gaming, and augmented/VR, require increased energy efficiency and higher computing capacity [1,2]; however, MDs often have limited computing power and low-capacity batteries, with these limitations representing obstacles for the network deployment and the emergence of new applications and services.

To address this, multi-access edge computing (MEC) represents a promising solution for enhancing the computing capabilities and limited battery power of MDs. Computation offloading to

MEC servers allows them to process the computational tasks of MDs at the edge of mobile cellular networks through wireless links, which addresses latency problems between cloud servers and MDs, as well as MD-specific limitations in computational power and storage. MEC brings cloud resources (e.g., MEC servers) to the network edges, thereby enabling heavy computing tasks to be performed close to the end user [2–5]. However, due to the resource limitation of the MEC server, computation offloading remains challenging. In particular, the rapid increase in compute-intensive tasks (such as face recognition, augmented reality, and autonomous driving) from devices will cause server resource bottlenecks of the MEC server and significantly affect task execution latency. Therefore, offloading calculation strategies and resource allocation optimization (e.g., computing resources at MEC server and transmit power) are needed to improve the performance of computation offloading.

Exponential increases in data traffic have been addressed through the use of heterogeneous networks (HetNets) comprising homogeneous cell macro-base stations (MBSs) with high transmission power overlaid onto low-power small cell base stations (SBSs) (e.g., femto-cells, pico-cells, and micro-cells) to meet data-rate requirements for next-generation networks [6–8]. Therefore, integrating MEC into 5G networks in order to increase network performance, to reduce MD-related power consumption, and to utilize macro and small cells represents a research interest in academia and industry. Recent studies have focused on computation offloading in 5G networks [5], with other research focusing on single-tier MEC servers used in heterogeneous scenarios [9–16]; however, achieving effective offloading decisions related to computation-completion time and energy consumption in a multi-tier MEC server remains an optimization problem that attracts considerable attention [17–20]. Moreover, there have been several MEC-related studies associated with its use in vehicular networks [21–24] and video processing [25,26].

In Reference [7], we proposed a two-tier computation-offloading framework for multi-user multi-server MEC in 5G HetNets. Different from our previous work that only focused on optimizing the offloading decisions to minimize energy consumption, in this paper, we considered a joint resource allocation and computation-offloading problem, aiming at minimizing the total computing overhead of MDs, including completion time and energy consumption. Using complex optimization techniques and various application constraints, we developed an efficient and low-complexity algorithm capable of high-quality solutions with guaranteed convergence. We found that a high-level heuristic (i.e., meta-heuristic) achieved satisfactory performance in solving a challenging optimization problem using particle swarm optimization (PSO) [27,28]. Our main contributions include the following:

- We investigated a two-tier computation-offloading strategy using multi-user multi-server MEC in 5G HetNets.
- We addressed joint resource allocation and computation-offloading decisions in order to minimize the total computing overhead of MDs, including completion time and energy consumption. The optimization problem was formulated as a mixed-integer nonlinear program (MINLP) problem of NP-hard complexity based on the inclusion of both continuous (resource allocation) and discrete (offloading decision) variables. This problem is highly complex and difficult to solve optimally in polynomial time. To address this, we divided the original problem into two subproblems: computational resource allocation and computation-offloading decisions.
- We developed an algorithm using a meta-heuristic based on PSO to solve the optimization problem. PSO can solve large-scale NP problems with polynomial time complexity while ensuring convergence within specified limits [27–29].
- Numerical simulations showed that the proposed algorithm achieved outstanding performance in terms of total computing overhead relative to several baseline schemes as well as guaranteeing the convergence of our solution.

The remainder of the manuscript is structured as follows. In Section 2, we review relevant literature. In Section 3, we describe a model two-tier computation-offloading system in a multi-user

multi-server MEC in a 5G HetNet. In Section 4, we formulate the problem to minimize the total computing overhead of MDs, including completion time and energy consumption. Our proposed PSO-based algorithm is presented in Section 5, and the numerical simulations are described in Section 6. The conclusion is presented in Section 7.

2. Related Work

There have been numerous studies addressing offloading decisions in MEC systems, with most focusing on computation offloading from multiple users in a single- or multi-server MEC. Multi-user single-server MEC architectures involve a single MEC server integrated into an MBS or SBS. We further classify existing research as addressing either single-tier [9–12] or multi-tier [13–16] offloading of a computational task. Chen et al. [9] used a game-theoretic approach to solve the problem of computing overhead, including completion time and energy consumption, whereas another study [10] used system utility as a quality of experience measurement based on device power consumption and completion time and solved the optimization problem by jointly addressing computational resources and offloading decisions. Hao et al. [11] studied offloading and task caching to minimize the power consumption of user equipments, and another study [12] used the same scenario, given that MEC servers are only attached to one wireless access station. In Reference [12], the authors used Karush–Kuhn–Tucker (KKT) conditions to solve the problem of optimizing MD energy consumption. Moreover, there were similar studies in single MEC server such as References [30–33]. For multi-tier offloading of computational tasks [13–16], there can be one or more SBSs overlaid by an MSB, with the SBSs connected to the MBS by a wireless or wired network. In this system, only MBS is integrated with the MEC servers. Zhang et al. [13] investigated energy-efficient computation offloading in 5G HetNets comprising one MBS and one SBS by deploying an optimization problem to minimize the power consumption of the system while meeting latency limitations. The computing and communication demands on SBSs are smaller than those on an MBS. A previous study [14] optimized wireless backhaul bandwidth using resource allocation and computation-offloading decisions in order to minimize system computing overhead. Moreover, other studies [15,16] considered a HetNet scenario created by multiple small cells and one macro cell, with Wang et al. [15] assuming that each SBS was only connected with one user and proposing optimization of time and energy consumption in order to complete the computational tasks by combining offloading decisions, interference management, and resource allocation. Zhang et al. [16] proposed an algorithmic trade-off between energy consumption and completion time by combining offloading decisions and computational resource allocation in MDs. In these cases, installing a MEC at the MBS could save monetary costs but could potentially increase latency. Additionally, an IoT system involves a large number of computation-intensive tasks that can lead to resource bottlenecks at the single MEC server.

In the case of multi-user multi-server MEC architectures, most studies focus on offloading onto single-tier multi-server MEC architectures [34–40]. In SBS and MBS systems equipped with MEC servers, the SBS and MBS work independently and considered equally important (the SBS is not connected to the MBS), enabling MDs to choose either the SBS or MBS for offload. A previous study [34] investigated use of a matching game to solve the optimization problem for total computing overhead, including energy consumption and completion time, by combining resource allocation and offloading decisions, and Tuyen et al. [35] maximized the system-offloading utility by combining the computation-offloading decision and resource allocation. Chen et al. [36] proposed a creative framework for computational task offloading to a MEC server in a software-defined ultra-dense network by optimizing the completion time of the computational tasks through their execution locally and on the edge cloud. Guo et al. [37,38] investigated offloading tasks in ultra-dense networks with multiple SBSs and one MBS. In one study [37], the authors developed a game-theoretic algorithm to solve the optimization problem for overall computing overhead considering various types of computational tasks through computation-offloading decisions, whereas in the other study [38], they optimized the overall cost of the computing overhead according to power consumption and

completion time by combining the optimal offloading decision and resource allocation (CPU cycles and power transmission) by MDs. Qui et al. [39] studied task offloading to a MEC server in order to extend resource capacity by hiring resources from cloud computing resources and vehicular nodes in order to minimize the total computing overhead, including latency and monetary cost, of using computer resources. Yang et al. [40] considered a two-tier small cell network comprising a set of relay base stations connected to a set of micro-base stations, where only the micro-base stations are integrated with the MEC servers in order to perform computational tasks from the user. In that study, they used an artificial fish swarm algorithm to optimize the overall energy consumption of the system according to computation-offloading decisions. Ateya et al. [41] investigated two levels of offloading decision between two micro-cloud units to achieve high energy efficiency while satisfied with latency. Wan et al. [42] studied computation offloading strategies for the applications in vehicles using edge node considering offloading cost, offloading latency, and load balancing between edge nodes. Lee et al. [17] proposed a hierarchical MEC architecture, with MEC servers arranged according to different processing performances configured by the network operator in order to increase network performance. In that scenario, MDs that request different actions are handled by different levels of MEC servers. A previous study [18] introduced a hybrid wireless network (FiWi) to provide support between cloud computing and MEC but only considered MDs connected to a wireless base station and applied a game-theoretic approach to optimize power consumption by user equipment. Liu et al. [19] optimized IoT power consumption by IoT sensors in a system model that included a MEC server and a remote cloud server by addressing inter-task dependency and the completion time of an IoT service. However, these studies only considered multi-tier architectures that included cloud computing and one MEC. Computation offloading in such multi-tier multi-server architectures is more difficult. In the present study, we considered multi-tier multi-server MEC systems for 5G HetNets and proposed an efficient, low-complexity algorithm to obtain optimal solutions.

3. Network Setting and Computational Model

3.1. Scenario Description

We considered multi-access edge computing in 5G HetNet comprising a set of SBSs connected to one MBS (Figure 1). MDs with limited computational capacity are randomly situated, thereby allowing them to directly connect to the SBSs in order to offload their computational task(s), with the SBSs connected to the MBS through fiber-optic links [13,16]. The SBSs and the MBS are integrated with MEC servers that can service the computational task(s) of the MDs. We defined the MEC servers co-located with the MBS and the SBSs as mMECs and sMECs, respectively. In this scenario, the MDs connect to the SBSs, which enable latency reduction and increases data rates. For this reason, we did not consider the MDs directly connected to the MBS [16]. We assume that the computing capability of MEC server attached to the SBS is limited and, so, when receiving many task offloading requests, may become overloaded; therefore, the computational task of MD can first be offloaded to the sMEC, but in the event that the computation resource is exhausted, the computational task will be offloaded to the mMEC, where powerful computing resources are available [38].

We denote the set of MDs as $\mathcal{M} = \{1, 2, \dots, M\}$ and the set of SBSs as $\mathcal{N} = \{1, 2, \dots, N\}$. Each MD has a computing-intensive task to be accomplished, and here, we assume that each MD has one task, which is subsequently executed on either the MD, sMEC, or mMEC. Each MD can offload to an sMEC via wireless links or continue offloading to the mMEC via fiber-optic links. The computational task of each MD is atomic and cannot be divided. The computational task of each MD is represented as $I_m = \{D_m, C_m\}$, $m \in \mathcal{M}$, where D_m is the data size of the computational task to be distributed to the edge cloud for computation and C_m is the total number of computational requirements required to complete the computational task, as measured in CPU cycles.

Figure 1. Multi-tier computation-offloading strategy in HetNets.

3.2. Communication Model

The computational task of an MD can either be performed on the local MD or offloaded to the MEC server. Therefore, we show the communication model when the computational task is offloaded to sMEC through a wireless link or continues offloading to the mMEC via fiber-optic links. In this study, we used orthogonal frequency division multiple access for communication between MDs and SBSs. To reuse the spectrum, we assumed that the spectrum is used in an overlaid manner. The transmission rate of the computational task I_m from the MD m to the SBS n through the wireless channel is expressed as follows:

$$R_{mn} = W \log_2 \left(1 + \frac{p_m h_{mn}}{n_0 + Q_{mn}} \right), \quad (1)$$

where W is the channel bandwidth, p_m is the fixed transmit power of the MD m , h_{mn} is the channel gain between the MD m and the SBS n , n_0 is the noise power, and Q_{mn} is the inter-cell interference.

3.3. Local Execution Model

We introduce the local execution model in terms of energy consumption and completion time. We can obtain the local completion time for MD m for accomplishing task I_m as

$$T_m^l = \frac{C_m}{f_m^l}, \quad (2)$$

where f_m^l is the local computing capability of the MD m . Let α denote a coefficient associated with the chip architecture [34]. Additionally, the energy consumption E_m^l of the MD m when the task I_m is computed locally as follows:

$$E_m^l = \alpha C_m (f_m^l)^2. \quad (3)$$

3.4. Computation-Offloading Model

This section presents the model for computational task offloading. The computational task can be offloaded to the SBS for execution at the sMEC through a wireless link or to the MBS (from SBS to MBS) for execution at the mMEC via fiber-optic links. Therefore, we classified this activity into two schemes for computational task offloading: (1) through the sMEC or (2) through the mMEC. Here, the total

energy consumption for completing the computational task comprises the energy consumption for the uploading transmission, for task performance on the MEC servers, and for returning the output data. In this study, we regarded optimization from the user perspective; therefore, we ignored the energy consumption associated with execution on the MEC server and only considered that required for the upload transmission [16]. This assumption is reasonable because it has been used by existing studies [16,35,36] and allows that the MEC server is usually operated using electricity supplied from the grid. Moreover, the power required to transmit the result back to the MD is small compared with that required from transmission of the original computational task [18,34,38].

3.4.1. sMEC Offloading

For offloading to a sMEC, the completion time T_{mn}^{sbs} of task I_m comprises two aspects: transmission time T_{mn}^t and remote computation time T_{mn}^{es} . Transmission time through the wireless access link when MD m offloads its task to the sMEC n is given by

$$T_{mn}^t = \frac{D_m}{R_{mn}}. \quad (4)$$

Let f_{mn} denote the computing resources of the sMEC n allocated to the MD m . The computation time required to execute the computational task on the sMEC n can be calculated as follows:

$$T_{mn}^{es} = \frac{C_m}{f_{mn}}. \quad (5)$$

According to Equations (4) and (5), the completion time T_{mn}^{sbs} can be obtained as follows:

$$T_{mn}^{sbs} = T_{mn}^t + T_{mn}^{es} = \frac{D_m}{R_{mn}} + \frac{C_m}{f_{mn}}. \quad (6)$$

As noted, the energy consumption required to offload computational task I_m for execution at sMEC n (denoted by E_{mn}^{sbs}) is equivalent to the energy consumption required to transmit task I_m from MD m to SBS n , as follows:

$$E_{mn}^{sbs} = E_{mn}^t = p_m T_{mn}^t = p_m \frac{D_m}{R_{mn}}. \quad (7)$$

3.4.2. mMEC Offloading

Computational tasks can be executed on the mMEC following initial transmission to the SBS through the wireless access link (from MD to SBS) and then migrated to the mMEC through a fiber-optic link (from SBS to MBS). We denote the transmission data rate of the fiber-optic link between each SBS and the MBS as β . The transmission time when the computational task I_m migrates from the SBS n to the mMEC via the fiber-optic link is calculated as follows:

$$T_{mn}^{tm} = \frac{D_m}{\beta}. \quad (8)$$

Let f_0 denote the computing resources allocated to each offloading MD by the mMEC, which is assumed to be constant and identical to all offloading MDs [10,16]. The computation time for the computational task on the mMEC is given by

$$T_{mn}^{em} = \frac{C_m}{f_0}. \quad (9)$$

The completion time T_{mn}^{mbs} when the computational task m is executed on the mMEC can be computed by the sum of (1) the transmission time between MD m and SBS n according to Equation (4)

via wireless access, (2) the transmission time T_{mn}^{tm} from SBS n to the mMEC through the fiber-optic link, and (3) the computation time T_{mn}^{em} at the mMEC:

$$T_{mn}^{mbs} = T_{mn}^t + T_{mn}^{tm} + T_{mn}^{em} = \frac{D_m}{R_{mn}} + \frac{D_m}{\beta} + \frac{C_m}{f_0}. \quad (10)$$

Similarly, the energy consumption when task I_m is offloaded to the mMEC (denoted by E_{mn}^{mbs}) is equal to the transmission energy consumed E_{mn}^t according to Equation (7):

$$E_{mn}^{mbs} = E_{mn}^t = p_m \frac{D_m}{R_{mn}}. \quad (11)$$

4. Problem Formulation and Analysis

The offloading decision determined the execution location for the computational tasks (e.g., sMEC offloading or mMEC offloading). Therefore, we divided the computation-offloading decision into two phases. The first involves task offload to the MEC server via the SBS, and the second involves task execution at the sMEC or the mMEC (migration from SBS to the mMEC through the fiber-optic link). Let $Y = \{y_{mn} | m \in \mathcal{M}, n \in \mathcal{N}\}$ denote the offloading-decision matrix, where each element y_{mn} is given as follows:

$$y_{mn} = \begin{cases} 1, & \text{if the MD } m \text{ offloads its computational task to the sMEC } n, \\ 0, & \text{otherwise.} \end{cases} \quad (12)$$

Because a task is offloaded to at most one MEC server, the following constraint must be satisfied: $\sum_{n=1}^N y_{mn} \leq 1, \forall m \in \mathcal{M}$. Let $Z = \{z_{mn} | m \in \mathcal{M}, n \in \mathcal{N}\}$ represent the matrix of offloading place for task I_m :

$$z_{mn} = \begin{cases} 1, & \text{the task } I_m \text{ is executed on the sMEC } n, \\ 0, & \text{the task } I_m \text{ is executed on the mMEC.} \end{cases} \quad (13)$$

The following constraint ensures that each task can be executed by at most one MEC server: $\sum_{n=1}^N z_{mn} \leq 1, \forall m \in \mathcal{M}$. Additionally, we denote $\mathcal{M}_{\text{off}} = \{m \in \mathcal{M}, n \in \mathcal{N} | y_{mn} = 1, z_{mn} = 1\}$ as the set of MDs that offload their tasks to sMECs and $\mathcal{M}_n = \{m \in \mathcal{M} | y_{mn} = 1, z_{mn} = 1\}$ as the set of MDs that offload their tasks to the sMEC m . Finally, we denote the computational-resource matrix for the sMEC as $F = \{f_{mn} | m \in \mathcal{M}, n \in \mathcal{N}\}$, where $f_{mn} > 0$ is the resource allocation of sMEC n to computational task I_m offloaded from MD $m \in \mathcal{M}_n$. Typically, the amount of computing resources at the sMEC is limited; therefore, the following constraint must be satisfied: $\sum_{m \in \mathcal{M}_n} f_{mn} \leq f_n^{\max}, \forall n \in \mathcal{N}$, where $f_n^{\max}$ denotes the maximum computing capability of sMEC n . Moreover, $f_{mn} = 0$ if $m \notin \mathcal{M}_n$, indicating that sMEC n does not allocate any computing resources to MD m .

Here, we describe joint optimization of resource allocation and computation-offloading decision strategies in order to minimize the total computing overhead of MDs, including completion time and energy consumption. In particular, the total time necessary to accomplish computational task I_m is computed as follows:

$$T_m = \sum_{n=1}^N y_{mn} (T_{mn}^t + z_{mn} T_{mn}^{es} + (1 - z_{mn})(T_{mn}^{tm} + T_{mn}^{em})) + \left(1 - \sum_{n=1}^N y_{mn}\right) T_m^l \quad (14)$$

The total energy consumption required to complete computational task I_m is computed as follows:

$$E_m = \sum_{n=1}^N y_{mn} (z_{mn} E_{mn}^{sbs} + (1 - z_{mn}) E_{mn}^{mbs}) + \left(1 - \sum_{n=1}^N y_{mn}\right) E_m^l \quad (15)$$

Therefore, the optimization problem can be expressed as follows:

$$\begin{aligned}
 & \underset{Y, Z, F}{\text{minimize}} && \sum_{m=1}^M (\lambda_m^e E_m + \lambda_m^t T_m) \\
 & \text{subject to} && \text{C1: } y_{mn}, z_{mn} \in \{0, 1\}, \forall m \in \mathcal{M}, \forall n \in \mathcal{N}, \\
 & && \text{C2: } \sum_{n=1}^N y_{mn} \leq 1, \forall m \in \mathcal{M}, \\
 & && \text{C3: } \sum_{n=1}^N z_{mn} \leq 1, \forall m \in \mathcal{M}, \\
 & && \text{C4: } \sum_{m=1}^M y_{mn} \leq H, \forall n \in \mathcal{N}, \\
 & && \text{C5: } f_{mn} > 0, \forall m \in \mathcal{M}_{\text{off}}, \forall n \in \mathcal{N}, \\
 & && \text{C6: } \sum_{m \in \mathcal{M}_n} f_{mn} \leq f_n^{\max}, \forall n \in \mathcal{N},
 \end{aligned} \tag{16}$$

where $\lambda_m^e, \lambda_m^t \in [0, 1]$. and $\lambda_m^e + \lambda_m^t = 1$ depict the weights associated with energy consumption and completion time for MD m and H is the number of subcarriers (denoting the number of MDs allowed to offload their computational tasks simultaneously). Equation (16) shows that constraint C1 represents the binary offloading decision, and constraints C2 and C3 ensure that each MD m can select at most one SBS. Constraint C4 implies that the number of tasks to be offloaded to the MEC server must be less than the total number of subcarriers, and constraints C5 and C6 indicate that the total computing resources assigned to the offloading MDs by each sMEC cannot exceed its maximum computing capability (note that $f_{mn} = 0$ if $m \notin \mathcal{M}_n$).

Equation (16) is an MINLP problem of NP-hard complexity and difficult to solve optimally in polynomial time [10,34]. Although global optimization can be used to obtain an optimal solution to this problem, the worst-case exponential computational complexity limits its application to 5G wireless networks due to the massive level of connectivity, heterogeneous quality of service requirements, and highly dynamic wireless channels. In this study, we developed an efficient and low-complexity algorithm capable of obtaining a high-quality solution with guaranteed convergence using PSO.

5. Joint Optimization of Resource Allocation and Computation-Offloading Decisions

Here, we describe the algorithm used for joint optimization of resource allocation and computation-offloading decisions using PSO to address the problem in Equation (16). In particular, Equation (16) is an MINLP problem of NP-hard complexity [10,34] based on the presence of both continuous (resource allocation profile F) and discrete (offloading decision profiles Y and Z) variables. To address this problem, we divided it into two subproblems: (1) computational resource allocation, which is a convex problem for a given offloading decision and solved by KKT optimality conditions, and (2) computation-offloading decision, which is solved by binary PSO. PSO has the advantages of easy implementation and fast convergence, as well as the ability to escape from local optima and to converge to an approximate global optimum [27,28]. Therefore, to solve Equation (16), we developed a PSO-based algorithm to jointly optimize resource allocation and computation-offloading decisions (JROPSO) in order to minimize total computing overhead of MDs.

5.1. Computational Resource Allocation

For a given task-offloading decision, $\mathbf{Y} = \mathbf{Y}_0$, $\mathbf{Z} = \mathbf{Z}_0$, Equation (16) for MDs offloading to sMEC n can be rewritten as follows:

$$\begin{aligned} & \underset{\mathbf{F}}{\text{minimize}} && \sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{M}_n} \frac{\lambda_m^t C_m}{f_{mn}} \\ & \text{subject to} && \text{C5 and C6.} \end{aligned} \quad (17)$$

Theorem 1. *The optimization problem in Equation (17) is a convex optimization problem.*

Proof of Theorem 1. We define $G(\mathbf{F})$ as the objective function in Equation (17) and obtain the Hessian matrix $G(\mathbf{F})$ with respect to $\mathbf{F}$, as follows:

$$\frac{\partial^2 G}{\partial f_{mn}^2} = \frac{2\lambda_m^t C_m}{f_{mn}^3}, \forall m \in \mathcal{M}_{\text{off}}, \forall n \in \mathcal{N}. \quad (18)$$

$$\frac{\partial^2 h}{\partial f_{mn} \partial f_{ij}} = 0, \forall (m, n) \neq (i, j). \quad (19)$$

We can verify that $\frac{\partial^2 G}{\partial f_{mn}^2} \geq 0$; therefore, the Hessian matrix is a positive semi-definite matrix. According to previously described theorems [43], we conclude that the objective function $G(\mathbf{F})$ is convex. Note that the solution space of Equation (17) is convex, which suggests that Equation (17) is convex. $\square$

Because Equation (17) is convex, it can be solved using KKT optimality conditions. The optimal resource allocation $\mathbf{F}$ is computed as follows:

$$f_{mn}^* = \frac{f_m^{\max} \sqrt{\lambda_m^t C_m}}{\sum_{m \in \mathcal{M}_n} \sqrt{\lambda_m^t C_m}}. \quad (20)$$

5.2. Computation-Offloading Decision

When computational resource allocation is provided (i.e., $\mathbf{F} = \mathbf{F}^*$), Equation (16) can be simplified as follows:

$$\begin{aligned} & \underset{\mathbf{Y}, \mathbf{Z}}{\text{minimize}} && \sum_{m=1}^M (\lambda_m^e E_m + \lambda_m^t T_m) \\ & \text{subject to} && \text{C1, C2, C3 and C4.} \end{aligned} \quad (21)$$

Here, we applied PSO to solve the subproblem in Equation (21) in order to obtain the solution to offloading decisions $\mathbf{Y}$ and $\mathbf{Z}$. In particular, the binary version of PSO (BPSO) is applied to solve the offloading-decision problem in Equation (21). First, we introduce the standard continuous PSO algorithm and the BPSO algorithm. PSO is a natural optimization method bio-inspired by the behavior of bird flocking [44], where each particle (individual) in a swarm searches for food in a search space. An effective way to locate food is to follow the bird closest to the food. By mimicking bird-foraging behavior, a variety of optimization problems can be solved by the PSO algorithm with highly competitive performance, including joint problems of power control and channel allocation in device-to-device communications [28,29].

In the PSO algorithm, a fitness function is defined to assess solution feasibility. The algorithm is initialized with random population-candidate solutions (particles), with each candidate solution represented by the position of a particle for the optimization problem. With each iteration, the position of each particle in the swarm changes within the search space in order to obtain a good solution based on the best individual position and the best position of all particles. After numerous iterations, the best solution will be found corresponding to the optimal position of all particles. Application

of the PSO algorithm requires mapping the problem solution onto the particle space, which directly effects feasibility and performance. We considered a swarm involving K particles and assumed that the particle position k th of the swarm was represented by $\mathbf{X}_k = (X_{k1}, X_{k2}, \dots, X_{km}, \dots, X_{kM})$ with a particle velocity k th of $\mathbf{V}_k = (V_{k1}, V_{k2}, \dots, V_{kn}, \dots, V_{kM})$, where M is the total number of MDs. According to a previous study [44], the particle updates its velocity and changes to a new position via the following two equations:

$$\mathbf{V}_k = w\mathbf{V}_k + c_1r_1(\mathbf{L}_k - \mathbf{X}_k) + c_2r_2(\mathbf{G} - \mathbf{X}_k), \quad (22)$$

$$\mathbf{X}_k = \mathbf{X}_k + \mathbf{V}_k, \quad (23)$$

where $\mathbf{L}_k$ and $\mathbf{G}$ represent the best individual position of the m th particle and the global best position, respectively; w denotes a constant inertia weight; c_1 and c_2 are two acceleration coefficients; and r_1 and r_2 are two random variables in $(0, 1)$.

Because the offloading decision and location are discrete values (i.e., either 0 or 1), the problem cannot be solved using the continuous standard PSO algorithm; therefore, we used the BPSO [45]. The main differences between the PSO and BPSO involve the position-updating procedure and the transfer function [45]. The position $\mathbf{X}_k$ is based on toggling between the values 1 and 0, with the velocity of the particle used to update the position. Therefore, we converted the velocity values to probabilities within the interval $[0, 1]$ using the sigmoid function, as follows [45]:

$$Sig(V_{km}) = \frac{1}{1 + e^{-V_{km}}}, \quad (24)$$

where V_{km} is the velocity of the k th particle at the m th element, with the position of this particle updated to a binary variable according to the following rule:

$$X_{km} = \begin{cases} 0, & \text{if } rand < Sig(V_{km}), \\ 1, & \text{otherwise,} \end{cases} \quad (25)$$

where $rand$ is a uniformly random number within $[0, 1]$. Algorithm 1 shows the pseudo-code for the standard BPSO, which is similar to the standard continuous PSO algorithm, except that updating each particle position occurs using Equation (25). $Fit(\mathbf{X}_k)$ is defined as the fitness value at each particle position $\mathbf{X}_k$.

According to a previous study [27], a modified version of the BPSO can avoid local optima more effectively than the original version; therefore, we apply a new transfer function instead of that described by the rule in Equation (25), as follows [27]:

$$Sig(V_{km}) = \left| \frac{\pi}{2} \arctan\left(\frac{\pi}{2} V_{km}\right) \right|. \quad (26)$$

Each candidate solution in the subproblem in Equation (21) (i.e., the offloading decision $\mathbf{Y}$ and the offloading place $\mathbf{Z}$ for all the MDs) is constructed according to a single position of each particle and its velocity and describes how the solutions are developed. We established a search space of D -dimension and transferred $\{\mathbf{Y}, \mathbf{Z}\}$ to the position of the particles. We denote $\mathbf{X} = \{\mathbf{Y}, \mathbf{Z}\}$ as the set of computation-offloading decisions and comprising matrices containing the offloading decisions and places. Therefore, the position of the k th particle is defined by $\mathbf{X}_k$ and the velocity of the k th particle is denoted by $\mathbf{V}_k$. To solve Equation (21), the fitness function for the position can be computed as follows:

$$Fit = \sum_{m=1}^M (\lambda_m^e E_m + \lambda_m^t T_m) + P, \quad (27)$$

where $P = \sum_{n=1}^N \theta_n (\max(0, \sum_{m=1}^M y_{mn} - H))$ is the penalty function, which is an integrated form of the inequality constraint function C4 in Equation (21) and θ_n is the penalty factor. Other constraints are guaranteed during initialization and genetic activities. The penalty function plays an important role in helping particles to fly out of a given area as soon as possible or as close to the feasible area as possible.

Algorithm 1: The standard binary particle swarm optimization (BPSO) algorithm.

Randomly generate velocity v_k and position x_k for the number of particles K

repeat

for each particle k in K do

 Calculate the fitness value

if $Fit(X_k)$ is better than $Fit(L_k)$ **then**

 | Update the best individual L_k

end

if $Fit(L_k)$ is better than $Fit(G)$ **then**

 | Update the global best G

end

 Update velocity variables via:

$$V_k = wV_k + c_1r_1(L_k - X_k) + c_2r_2(G - X_k)$$

$$Sig(V_k) = \frac{1}{1+e^{-V_k}}$$

 Generate a series of uniform random number $rand$ in $[0, 1]$.

 Update the binary position X_k : $X_k = (Sig(V_k) > rand)$.

end

until Maximum iterations or stopping condition is satisfied;

5.3. The Proposed Algorithm JROPSO

As noted, the proposed algorithm JROPSO uses PSO to jointly find optimal solutions to problems concerning computation-offloading decisions in Equation (21) and computational resource allocation in Equation (20). The JROPSO algorithm is illustrated in Algorithm 2.

Algorithm 2: PSO-based joint resource allocation and computational offloading decision algorithm (JROPSO)

Input: The size (K) of the particle population ($k = \{1, \dots, K\}$), the maximum number of iterations $T_{\max}$, the convergence criteria ζ , and the parameter w, c_1 , and c_2 .

Output: The best fitness values and global best position.

Initialization: population of K particles with random positions X_k and velocities V_k . Set the iteration index as $t = 1$.

Compute the resource allocation according to Equation (20).

Calculate the fitness value of each particle according to Equation (27).

repeat

for each the number of particles K_m do

 Update the velocity V_k and position X_k according to Equations (22) and (26), respectively.

end

 Update the resource allocation according to Equation (20).

 Calculate the fitness of each particle solution according to Equation (27).

 Update the best individual value if the new fitness is better than the current value.

 Update the global best solution among K particles.

 Increase the iteration index: $t = t + 1$.

until $t > T_{\max}$ or $Fit(t) - Fit(t-1) \leq \zeta$;

6. Performance Evaluation

6.1. Simulation Settings

Here, we present the simulation results demonstrating the performance of the JROPSO algorithm. We employed the following simulation scenario for most of the simulations unless otherwise noted. We considered the HetNets as comprising of $M = 50$ MDs within a coverage area ($250 \times 250 \text{ m}^2$) including six SBSs and one MBS connected to all SBSs. The channel-gain models were assumed to follow the path-loss model according to $140.7 + 36.7\log_{10}(d)$ [35], where d (km) is the distance between the MD and the SBS. For the communication model, the noise power was set to $n_0 = -100 \text{ dBm}$ [34], and the transmit power of the MD for task offloading to the SBS was set to $p_m = 23 \text{ dBm}$. The data rate of the fiber-optic link was $\beta = 1 \text{ Gbps}$ [18], and we set the number of subcarriers at $H = 12$ and the bandwidth of each sub-channel to $W = 3 \text{ MHz}$. Additionally, there was no inter-cell interference in the wireless fronthaul. For the computational task, the input data size of the computational tasks was uniformly distributed within the range $[600, 1200] \text{ KB}$, and the number of CPU cycles required for completing the computational task was randomly generated at between $C_m = [500, 1000]$ Megacycles. In terms of computing resources, the CPU speed of the computing resource of the mMEC allocated to each MD was $f_0 = 3 \text{ GHz}$, the maximum CPU computing capacity for each sMEC was $f_m^{\max} = 6 \text{ GHz}$, and the local computing power of the MDs was randomly assigned from the set $\{0.5, 0.8, 1\} \text{ GHz}$ [34,39] and $\alpha = 5 \times 10^{-27}$ [34,35]. The weighting parameters of the computational task were set to 0.5 (i.e., $\lambda_m^t = \lambda_m^l = 0.5, \forall m \in \mathcal{M}$). For the PSO algorithm, PSO parameters were set according to a previous study [27]. Briefly, we set $w = 2, c_1 = 2, c_2 = 2$, the size of the particle population (K) to 30, and the maximum number of iterations ($T_{\max}$) to 1000. The simulation results were averaged over a number of Monte Carlo simulations, and MD and SBS locations were randomly and uniformly distributed.

6.2. Simulation Results

To evaluate the performance of the JROPSO algorithm, we compared it with baseline schemes:

- Local Computing Only (LCO): all of the MDs execute their computational tasks locally in the absence of computation offloading.
- Without MBS (WMBS): The proposed algorithm JROPSO is implemented without MBS.
- Only offloading in a 2-tier system (OF): offloading only to sMECs and mMEC, with computation resource allocation optimized for sMECs.

In the first experiment, we verified the convergence of our proposed algorithm in terms of the total computing overhead by the numerical result in Figure 2. In this experiment, we performed in 3 cases of 10 MDs, 30 MDs, and 50 MDs. From Figure 2, we see that the proposed algorithm can converge into a stable solution after a finite number of iterations. In addition, the average iteration for the convergence of the proposed algorithm is approximately linear with the number of MDs.

In the second experiment, we determined the percentage of offloading MDs according to the increase in the total number of MDs (Figure 3). Initially, when the number of MDs was small, the percentage of offloading MDs was relatively high; however, as the number of MDs increased, the percentage of offloading MDs decreased. The reason for this is that each MD tends to offload its computational task to the MEC server to save energy consumption and/or completion time. However, a larger number of MDs lead to a high competition for computing resources at the MEC server. Therefore, more MDs are not beneficial from remote computing; thus, some MDs must be executed locally to minimize the total computing overhead.

Figure 2. JROPSO convergence in terms of total computing overhead according to the number of iterations.

Figure 3. Percentage of offloading mobile devices (MDs) according to the number of MDs.

In the third experiment, we evaluated proposed algorithm JROPSO in terms of the total computing overhead of MDs with respect to increases in the number of MDs (Figure 4). The results indicated that total computing overhead calculated by all of the schemes increased along with the number of MDs. The results indicated that total computing overhead calculated by all of the schemes increased along with the number of MDs. The reason is that the total computing overhead of MDs is linearly related to the number of MDs. We also saw that the proposed algorithm JROPSO helped reduce the total computing overhead of MDs. More specifically, the proposed algorithm JROPSO achieved the lowest total computing overhead relative to other schemes. This is because the proposed algorithm JROPSO was optimized for the joint resource-allocation and computation-offloading decision such that computation offloading is beneficial for offloading tasks.

Figure 4. Comparison of the total computing overhead according to the number of MDs.

In the fourth experiment, we evaluated algorithm performance according to different weighting parameters (Figure 5). Specifically, we increased energy consumption λ_m^e from 0.1 to 0.9 with a step of 0.1, which altered completion time λ_m^t accordingly to $\lambda_m^t = 1 - \lambda_m^e$. The number of MDs is fixed at 50. The results indicated that the proposed algorithm JROPSO was superior to other solutions, although the weighting parameters vary. The total computing overhead according to the LCO algorithm consistently increased along with the λ_m^e parameter because local execution of computational tasks by MDs results in large energy consumption as compared with the execution time. For example, MD m with $f_l = 1$ GHz and $C_m = 800$ Megacycles returned $T_m^l = (800 \times 10^6) / (1 \times 10^9) = 0.8$ (seconds) and $E_m^l = 5 \times 10^{-27} \times 800 \times 10^6 \times (1 \times 10^9)^2 = 4$ (Joules). This suggests that, for local computing, the completion time was 5-fold smaller than the energy consumption. By contrast, the FO algorithm offloaded the computational task to the MEC server in order to save the energy consumption of the MDs. Therefore, increases in λ_m^e decreased the total computing overhead and suggested that a large enough λ_m^e would result in decreases in total computing overhead according to the FO, WMBS, and JROPSO algorithms. This is a consequence of the benefits of offloading to allow energy savings versus computing time. These results indicated that optimal performance requires choosing the appropriate weighting parameters.

Figure 5. Performance evaluation under different weighting parameters.

Figure 6 shows the effect of data size on the JROPSO algorithm. The results indicated that increased data size did not change the total computing overhead according to the LCO algorithm based on it having no impact on completion time or energy consumption when the tasks are executed locally. By contrast, the computing overhead calculated by the OF algorithm increased rapidly due to high completion time to offloading task on MEC server. The large data sizes resulted in computing overheads for the JROPSO algorithm close to the WMBS algorithm. The reason is that some tasks will not be executed at mMEC due to the considerable time required to transfer the task to mMEC. These findings suggest that computing offloaded tasks involving small amounts of data achieve better results.

Figure 6. The effect of data size on task offloading.

7. Conclusions

In this study, we described the development of an efficient optimization method to determining computation-offloading decisions in multi-tier multi-MEC-server architectures within 5G HetNets. We addressed resource-allocation and computation-offloading decision strategies to minimize the total computing overhead, including completion time and energy consumption, and developed the JROPSO algorithm to obtain optimal solutions. Our simulations confirmed the efficiency of the algorithm and its superior performance relative to other methods. In addition, this paper showed that the PSO algorithm is used to solve the optimization problem of computation offloading with highly competitive performance. In future work, the comparison of our proposed algorithm with recent meta-heuristic algorithms like Ant Colony Optimization (ACO), Whale Optimization Algorithm (WOA), and Harris Hawks Optimization (HHO) is also promising. Moreover, we will involve investigation of PSO applications in non-orthogonal multiple-access-assisted MEC systems.

Author Contributions: E.-N.H. supervised the study; L.N.T.H. proposed the conceptualization; L.N.T.H. developed the methodology; L.N.T.H. and Q.-V.P. analyzed the algorithm; L.N.T.H. programmed the software; L.N.T.H. wrote the original draft of the manuscript; L.N.T.H., Q.-V.P., X.-Q.P., T.D.T.N., M.D.H., and E.-N.H. participated in manuscript review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: This work was supported by Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2017-0-00294, Service mobility support distributed cloud technology).

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

References

1. Mach, P.; Becvar, Z. Mobile Edge Computing: A Survey on Architecture and Computation Offloading. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 1628–1656. [\[CrossRef\]](#)
2. Mao, Y.; You, C.; Zhang, J.; Huang, K.; Letaief, K.B. A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2322–2358. [\[CrossRef\]](#)
3. Y.3508: Cloud Computing—Overview and High-Level Requirements of Distributed Cloud; Recommendation Y.3508; ITU Publications: Geneva, Switzerland, 2019.
4. Pham, Q.; Nguyen, T.H.; Han, Z.; Hwang, W. Coalitional Games for Computation Offloading in NOMA-Enabled Multi-Access Edge Computing. *IEEE Trans. Veh. Technol.* **2019**. [\[CrossRef\]](#)
5. Pham, Q.V.; Fang, F.; Vu, H.; Le, M.; Ding, Z.; Le, L.B.; Hwang, W. A Survey of Multi-Access Edge Computing in 5G and Beyond: Fundamentals, Technology Integration, and State-of-the-Art. *arXiv* **2019**, arXiv:1906.08452.
6. Ramazanali, H.; Mesodiakaki, A.; Vinel, A.; Verikoukis, C. Survey of user association in 5G HetNets. In Proceedings of the 2016 8th IEEE Latin-American Conference on Communications (LATINCOM), Medellin, Colombia, 15–17 November 2016; pp. 1–6.
7. Huynh, L.N.T.; Pham, Q.V.; Nguyen, Q.D.; Pham, X.Q.; Nguyen, V.; Huh, E.N. Energy-Efficient Computation Offloading with Multi-MEC Servers in 5G Two-Tier Heterogeneous Networks. In Proceedings of the 13th International Conference on Ubiquitous Information Management and Communication (IMCOM), Phuket, Thailand, 4–6 January 2019; Springer International Publishing: Cham, Switzerland, 2019; pp. 120–129.
8. Hu, Y.C.; Patel, M.; Sabella, D.; Sprecher, N.; Young, V. Mobile edge computing—A key technology towards 5G. *ETSI White Pap.* **2015**, *11*, 1–16.
9. Chen, X.; Jiao, L.; Li, W.; Fu, X. Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing. *IEEE/ACM Trans. Netw.* **2016**, *24*, 2795–2808. [\[CrossRef\]](#)
10. Lyu, X.; Tian, H.; Sengul, C.; Zhang, P. Multiuser Joint Task Offloading and Resource Optimization in Proximate Clouds. *IEEE Trans. Veh. Technol.* **2017**, *66*, 3435–3447. [\[CrossRef\]](#)
11. Hao, Y.; Chen, M.; Hu, L.; Hossain, M.S.; Ghoneim, A. Energy Efficient Task Caching and Offloading for Mobile Edge Computing. *IEEE Access* **2018**, *6*, 11365–11373. [\[CrossRef\]](#)
12. Tao, X.; Ota, K.; Dong, M.; Qi, H.; Li, K. Performance Guaranteed Computation Offloading for Mobile-Edge Cloud Computing. *IEEE Wirel. Commun. Lett.* **2017**, *6*, 774–777. [\[CrossRef\]](#)
13. Zhang, K.; Mao, Y.; Leng, S.; Zhao, Q.; Li, L.; Peng, X.; Pan, L.; Maharjan, S.; Zhang, Y. Energy-Efficient Offloading for Mobile Edge Computing in 5G Heterogeneous Networks. *IEEE Access* **2016**, *4*, 5896–5907. [\[CrossRef\]](#)
14. Pham, Q.V.; Le, L.B.; Chung, S.H.; Hwang, W.J. Mobile Edge Computing With Wireless Backhaul: Joint Task Offloading and Resource Allocation. *IEEE Access* **2019**, *7*, 16444–16459. [\[CrossRef\]](#)
15. Wang, C.; Yu, F.R.; Liang, C.; Chen, Q.; Tang, L. Joint Computation Offloading and Interference Management in Wireless Cellular Networks with Mobile Edge Computing. *IEEE Trans. Veh. Technol.* **2017**, *66*, 7432–7445. [\[CrossRef\]](#)
16. Zhang, J.; Hu, X.; Ning, Z.; Ngai, E.C.; Zhou, L.; Wei, J.; Cheng, J.; Hu, B. Energy-Latency Tradeoff for Energy-Aware Offloading in Mobile Edge Computing Networks. *IEEE Internet Things J.* **2018**, *5*, 2633–2645. [\[CrossRef\]](#)
17. Lee, J.; Lee, J. Hierarchical Mobile Edge Computing Architecture Based on Context Awareness. *Appl. Sci.* **2018**, *8*, 1160. [\[CrossRef\]](#)
18. Guo, H.; Liu, J. Collaborative Computation Offloading for Multiaccess Edge Computing Over Fiber–Wireless Networks. *IEEE Trans. Veh. Technol.* **2018**, *67*, 4514–4526. [\[CrossRef\]](#)
19. Liu, F.; Huang, Z.; Wang, L. Energy-Efficient Collaborative Task Computation Offloading in Cloud-Assisted Edge Computing for IoT Sensors. *Sensors* **2019**, *19*, 1105. [\[CrossRef\]](#)
20. Ryu, J.W.; Pham, Q.V.; Luan, H.N.T.; Hwang, W.J.; Kim, J.D.; Lee, J.T. Multi-Access Edge Computing Empowered Heterogeneous Networks: A Novel Architecture and Potential Works. *Symmetry* **2019**, *11*, 842. [\[CrossRef\]](#)
21. Zhao, J.; Li, Q.; Gong, Y.; Zhang, K. Computation Offloading and Resource Allocation for Cloud Assisted Mobile Edge Computing in Vehicular Networks. *IEEE Trans. Veh. Technol.* **2019**, *68*, 7944–7956. [\[CrossRef\]](#)
22. Wang, J.; Feng, D.; Zhang, S.; Tang, J.; Quek, T.Q.S. Computation Offloading for Mobile Edge Computing Enabled Vehicular Networks. *IEEE Access* **2019**, *7*, 62624–62632. [\[CrossRef\]](#)
23. Fan, X.; Cui, T.; Cao, C.; Chen, Q.; Kwak, K.S. Minimum-Cost Offloading for Collaborative Task Execution of MEC-Assisted Platooning. *Sensors* **2019**, *19*, 847. [\[CrossRef\]](#)

24. Lamb, Z.W.; Agrawal, D.P. Analysis of Mobile Edge Computing for Vehicular Networks. *Sensors* **2019**, *19*, 1303. [CrossRef] [PubMed]
25. Tran, T.X.; Pompili, D. Adaptive Bitrate Video Caching and Processing in Mobile-Edge Computing Networks. *IEEE Trans. Mob. Comput.* **2019**, *18*, 1965–1978. [CrossRef]
26. Long, C.; Cao, Y.; Jiang, T.; Zhang, Q. Edge Computing Framework for Cooperative Video Processing in Multimedia IoT Systems. *IEEE Trans. Multimed.* **2018**, *20*, 1126–1139. [CrossRef]
27. Mirjalili, S.; Lewis, A. S-shaped versus V-shaped transfer functions for binary Particle Swarm Optimization. *Swarm Evol. Comput.* **2013**, *9*, 1–14. [CrossRef]
28. Xu, J.; Guo, C.; Zhang, H. Joint channel allocation and power control based on PSO for cellular networks with D2D communications. *Comput. Netw.* **2018**, *133*, 104–119. [CrossRef]
29. Girmay, G.G.; Pham, Q.; Hwang, W. Joint channel and Power Allocation for Device-to-Device Communication on Licensed and Unlicensed Band. *IEEE Access* **2019**, *7*, 22196–22205. [CrossRef]
30. Mao, Y.; Zhang, J.; Song, S.H.; Letaief, K.B. Power-Delay Tradeoff in Multi-User Mobile-Edge Computing Systems. In Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, 4–8 December 2016; pp. 1–6.
31. Deng, M.; Tian, H.; Lyu, X. Adaptive sequential offloading game for multi-cell Mobile Edge Computing. In Proceedings of the 2016 23rd International Conference on Telecommunications (ICT), Thessaloniki, Greece, 16–18 May 2016; pp. 1–5.
32. Al-Shuwaili, A.; Simeone, O. Energy-Efficient Resource Allocation for Mobile Edge Computing-Based Augmented Reality Applications. *IEEE Wirel. Commun. Lett.* **2017**, *6*, 398–401. [CrossRef]
33. Kan, T.; Chiang, Y.; Wei, H. Task offloading and resource allocation in mobile-edge computing system. In Proceedings of the 2018 27th Wireless and Optical Communication Conference (WOCC), Hualien, Taiwan, 30 April–1 May 2018; pp. 1–4.
34. Pham, Q.V.; Anh, T.L.; Tran, N.H.; Park, B.J.; Hong, C.S. Decentralized Computation Offloading and Resource Allocation for Mobile-Edge Computing: A Matching Game Approach. *IEEE Access* **2018**, *6*, 75868–75885. [CrossRef]
35. Tran, T.X.; Pompili, D. Joint Task Offloading and Resource Allocation for Multi-Server Mobile-Edge Computing Networks. *IEEE Trans. Veh. Technol.* **2019**, *68*, 856–868. [CrossRef]
36. Chen, M.; Hao, Y. Task Offloading for Mobile Edge Computing in Software Defined Ultra-Dense Network. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 587–597. [CrossRef]
37. Guo, H.; Liu, J.; Zhang, J.; Sun, W.; Kato, N. Mobile-Edge Computation Offloading for Ultradense IoT Networks. *IEEE Internet Things J.* **2018**, *5*, 4977–4988. [CrossRef]
38. Guo, H.; Zhang, J.; Liu, J.; Zhang, H. Energy-Aware Computation Offloading and Transmit Power Allocation in Ultradense IoT Networks. *IEEE Internet Things J.* **2019**, *6*, 4317–4329. [CrossRef]
39. Pham, X.Q.; Nguyen, T.D.; Nguyen, V.; Huh, E.N. Joint Node Selection and Resource Allocation for Task Offloading in Scalable Vehicle-Assisted Multi-Access Edge Computing. *Symmetry* **2019**, *11*, 58. [CrossRef]
40. Yang, L.; Zhang, H.; Li, M.; Guo, J.; Ji, H. Mobile Edge Computing Empowered Energy Efficient Task Offloading in 5G. *IEEE Trans. Veh. Technol.* **2018**, *67*, 6398–6409. [CrossRef]
41. Ateya, A.A.; Muthanna, A.; Vybornova, A.; Darya, P.; Koucheryavy, A. Energy—Aware Offloading Algorithm for Multi-level Cloud Based 5G System. In *Internet of Things, Smart Spaces, and Next Generation Networks and Systems*; Springer International Publishing: Cham, Switzerland, 2018; pp. 355–370.
42. Wan, S.; Li, X.; Xue, Y.; Lin, W.; Xu, X. Efficient computation offloading for Internet of Vehicles in edge computing-assisted 5G networks. *J. Supercomput.* **2019**, 1–30. [CrossRef]
43. Boyd, S.; Vandenberghe, L. *Convex Optimization*; Cambridge University Press: Cambridge, UK, 2004.
44. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the International Conference on Neural Networks (ICNN'95), Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
45. Kennedy, J.; Eberhart, R. A discrete binary version of the particle swarm algorithm. In Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, Orlando, FL, USA, 12–15 October 1997; Volume 5, pp. 4104–4108.