A Parallelizable Task Offloading Model with Trajectory-Prediction for Mobile Edge Networks

Pu Han 1,2,3, Lin Han 2, Bo Yuan 4, Jeng-Shyang Pan 5,6 and Jiandong Shang 2,*

1 School of Computer and Artificial Intelligence, Zhengzhou University, Zhengzhou 450001, China
2 National Supercomputing Center in Zhengzhou, Zhengzhou University, Zhengzhou 450000, China
3 Nanyang Institute of Technology, No.80, Changjiang Road, Nanyang 473000, China
4 School of Informatics, University of Leicester, Leicester LE1 7RH, UK
5 College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China
6 Department of Information Management, Chaoyang University of Technology, Taichung 413310, Taiwan

* Correspondence: shangjiandong@zzu.edu.cn

Abstract: As an emerging computing model, edge computing greatly expands the collaboration capabilities of the servers. It makes full use of the available resources around the users to quickly complete the task request coming from the terminal devices. Task offloading is a common solution for improving the efficiency of task execution on edge networks. However, the peculiarities of the edge networks, especially the random access of mobile devices, brings unpredictable challenges to the task offloading in a mobile edge network. In this paper, we propose a trajectory prediction model for moving targets in edge networks without users’ historical paths which represents their habitual movement trajectory. We also put forward a mobility-aware parallelizable task offloading strategy based on a trajectory prediction model and parallel mechanisms of tasks. In our experiments, we compared the hit ratio of the prediction model, network bandwidth and task execution efficiency of the edge networks by using the EUA data set. Experimental results showed that our model is much better than random, non-position prediction parallel, non-parallel strategy-based position prediction. Where the task offloading hit rate is closed to the user’s moving speed, when the speed is less 12.96 m/s, the hit rate can reach more than 80%. Meanwhile, we we also find that the bandwidth occupancy is significantly related to the degree of task parallelism and the number of services running on servers in the network. The parallel strategy can boost network bandwidth utilization by more than eight times when compared to a non-parallel policy as the number of parallel activities grows.

Keywords: edge computing; mobile edge network; parallelization; task offloading; trajectory prediction

1. Introduction

With the popularization and widespread promotion of handheld mobile devices in the Internet of Things (IoT), the mobile computing model has become one of the most important forms of computing under the current hybrid Internet communication model [1]. In the mobile network, people can submit various computing tasks at any time with the nomadic devices in their hands [2]. Unlike the traditional resource-centering cloud architecture model [3,4], mobile edge computing is a kind of distributed computing model centered on edge services [5]. It could resolve the problems, such as edge network delay [6], task overload [7], and network congestion [8], by maximizing the utilization of edge computing resources [9].

Due to the limitation of edge network resources in mobile edge network(MEN), how to optimize the offloading of services and the allocation of resources has always been a challenging problem [10]. Although the promotion of 5G technology and devices have greatly improved the data transmission and service delay of the MEN, the random access
of network devices, the variety of available network resources and the users mobility may limit the advantages of MEN. Many researchers proposed that providing services to users in the nearest place can reduce the network delay and improve the Quality of Service (QoS) [7,11,12]. J.Xu et al. [13] proposed a service proximity caching strategy, which responds to the user’s service request on the nearest edge server, and shortens the waiting time for service requests. Another approaches split the multiple user’s tasks into many parts, and distribute them to the terminal devices, edge servers, and cloud to share computing resources of MEN [14–16].

The user task request and offloading scheme in the MEN is closely related to the access location of the user equipment (UE), thus it is becoming a trend to introduce user’s mobility into the task offloading strategy of edge computing services [17]. Because of the randomness and unpredictability of the user behavior in MEN [18], as shown in Figure 1, the demand for the resources provided by the network is not determined when the UEs pass through the area covered by the edge servers. It is the real-time changes in network resources caused by this random behavior that affect the performance of task load strategies. Nadembega et al. discovered that the movement of users in an edge network has a great impact on the user experience and QoS of network in [19]. They also proposed a method to improve the efficiency of data transmission for edge network tasks. Therefore, it is worth considering the behavioral factors jointly when exploring task offloading solutions in MEN.

![Figure 1. Moving targets in the MEN.](image)

Figure 1 represents some moving UEs request services from edge networks (ENs). If no policy intervention is added during the tasks assignment, the terminal users have to wait around for their expected results [20]. It can be improved by the effective utilization of edge devices to guess the user’s location before task offloading [21,22]. This kind of method, by introducing the trajectory prediction based on a prior data involving users’ previous location into the tasks offloading in MEN, indeed improved the effective utilization of resources near the user. However, it is challenging to obtain the user’s personal location information.

In this paper, we proposed an approach to distribute the tasks in MEN based on a non-prior trajectory prediction model (NPTP) which can enhance the predictive ability of task offloading decision. In other words, we construct a dynamic user-centering edge network (UCEN) based on user’s movement trajectory, and surmise the next position of the users just following their current information in the coverage area of MEN. Then, a tasks offloading model can automatically adapt to the topology changes of UCENs based on the user’s movement. In addition, the offloading model distributes the sub-tasks of one service to different computing devices for concurrent execution, and the simulation experiments of services paralleling showed that the model has obvious advantages in
the task’s execution efficiency and equipment occupation rate. In this article, we analyze current trajectory prediction methods and tasks assignment strategies in MEN, and make some improvements in how to reduce the dependence of trajectory prediction on historical information and sub-tasks offloading among the parallelizable tasks. In short, our contributions are the following:

1. A non-prior trajectory prediction model (NPTP).
   NPTP is based on random walking model, which can forecast the user’s next position through their current location and the moving direction. It provides an important reference for edge servers to optimize service offloading.

2. A dynamic construction mechanism for mobile edge networks.
   According to the user’s trajectory, UEs construct a mobile cloud dynamically to serve themselves, which is a mobile user-centering edge network (MUCEN) architecture. The MUCEN would be constructed along the user’s movement path provisionally. Therefore, the available computing devices, prepared for the user during their movement, are dynamically adjusted in the domain of MUCEN.

3. Parallelizable Task Offloading Strategy Based Position Prediction (PTOSBPP) in MEN.
   Due to the concurrent and asynchronous execution of the sub-tasks, we proposed a service offloading strategy for MEN that can adapt to dynamic user’s location.

The rest of this paper is organized as follows: Section 2 introduces the related methods about task offloading in MEN. It illustrates that how NPTP model to guess the next position of the UEs, and formulate the cost of task offloading in Section 3. Then, the algorithms of NPTP and PTOSBPP in detail are described in Section 4. Finally, we show the simulation results and analyze the performance of the algorithms, and conclude this study in Section 5.

2. Related Works

Edge computing is a new computing model after cloud computing and fog computing [9,11]. Its purpose is to assuage the computing pressure of the cloud with the idle and scattered resources around the UEs. Task offloading is a well-established approach to improve the edge computing performance [1,23]. However, with the increasing of the portable devices in MEN, it makes the task unloading and resource apportionment more complicated. The deployment of MEN services is a challenging issue during the process of tracking user’s mobility [24]. In order to achieve a better QoS and Quality of Experience (QoE) in MEN, lots of task offloading strategies are proposed by many researchers who are studying how to distribute the limited network resources to end users in a rational way. The traditional optimization techniques for these distribution schemes can be broadly divided into two categories: decision optimization and architecture or devices optimization.

Decision optimization mainly analyzes the parameters, bandwidths, server speeds, available memory etc., evaluates the cost function, and then determines a resource allocation strategy [25]. For instance, Chen, Xu et al. [26] designed a novel D2D Crowd framework to assist in the sharing of edge network resources. The authors of [27] proposed an adaptive computation offloading approach for saving the power of portable computer. Wolski et al. [28] presented a framework for making computation offloading decisions based on the networks bandwidth. The methods, proposed in [29,30], tried to explore as many computing resources as possible around the base station (BS) to meet the computing needs of edge services. As mentioned in [1], discovering and using the potential resources as much as possible is theoretically conducive to task execution efficiency in MEN. In most cases, these solutions are restricted by the number of free resources in UCENs, and weak in adapting to real-time changes of the network topology caused by random accessing. When the edge resources are insufficient, the cloud servers still need to provide a large amount of computing resources to assist in completing the computing tasks. Optimizing the network service architecture also address the issue of deal distribution of edge services. The UEs are responeded from the MEN in a short time though a novel network device [31,32] or virtualization technologies [25]. Femtolet proposed by Mukherjee et al. [32] could offer
communication and computation offloading facilities simultaneously at low power and low latency. As well as, Chun et al. [33] designed an architecture to extend the edge server by cloud. There are lots of articles studying the methods mentioned above. We summarize them in Table 1, and analyze the shortcomings among various approaches.

### Table 1. Categories of Traditional Task Offloading Research in Edge Networks.

| Optimization Goals | Methodology Description                                                                 | References                      | Shortcoming                                                                 |
|--------------------|----------------------------------------------------------------------------------------|--------------------------------|----------------------------------------------------------------------------|
| Decision          | Assigning computational tasks to user’s neighbouring devices based on the resources in the current area | [1,4,13,16,23,26–30,34]       | The offloading strategy is limited by the number of available resources.         |
| Offloading based on constant resources | Expanding the available resources set based on the user’s current area and the predicted target area, and then offloading the tasks | [19,21,22,35–40]               | Predictive modeling needs historical movement trajectory, it’s difficult to obtain these personal data. |
| Architecture or Devices | Optimizing the architecture of edge network or update devices to improve the performance of task offloading | [3,5,18,20,31–33]             | It requires additional investment to upgrade and renovate the network.          |

Arbitrary accessing computational resource is a relatively prevalent occurrence in MEN, which causes complications in task offloading. Thus, if the destination server is sensed in advance for a moving user, it is beneficial for task offloading. As predicting the target edge sever is related to the user’s spatio-temporal location, it is necessary to study the mobile behavior of users. There are large numbers of mobile devices with random access in the MEN. It is the way of connection that brings huge challenges to the task offloading. Currently, this issue has attracted much attention in authors in [9,14,19,21,22,41]. A deep learning method in [37] is used to predict the trajectory of mobile edge network users for the task distribution among the ESs. O.Tao et al. [24] used the Markov model to predict the user’s movement trajectory, and achieved the optimal goal of long-term service in the application and released of network resources. Aissioui et al. [42] also proposed a FMeC-based framework for automated driving, which addresses the increased cost of network service migration caused by vehicle location updates. The methods proposed in [38,39] speculated the UE’s next position based on the terminal devices’ track. In some special scenario, the service offloading decision is built on a Lagrange trajectory interpolation method [40]. These studies indicate that mobility of the users has a certain impact on the efficiency of task execution in MEN, and the prediction of the user’s movement trajectory is a suitable for improving task offloading performance. However, all of methods have a common feature that they needed historical data for position prediction. We call them as prior displacement prediction model (PDPM). In Section 1, we have already stated that it’s difficult and even illegal to acquire the past privacy data. Thus, PDPM seems to have some limitations in practice.

To overcome the shortage of PDPM in historical data collection, we propose a non-prior trajectory prediction model to predict the path of moving users in this paper. In addition, considering the techniques of paralleling and fragmentation could expand the number of edge devices that participate in task offloading, and reduce the network latency, the methods in [15,43] show that parallelization is one of the effective and efficient ways to improve the offloading of edge tasks in MEN. Therefore, our experiment also import a sub-task offloading policy.

### 3. Problem Statement and System Model

In this section, we formulate the mobile user-centering edge network architecture, introduce the model of the NPTP, and then discuss the optimal cost calculation method of parallel unloading strategy.
3.1. Mobile User-Centering Edge Network Architecture

Generally, the terminal device with users are centered on the edge server which dominates the task assignment. All of the service request are triggered by them. The hardware resource, such as processor, storage etc., can also participate in the scheduling and allocation of resources. However, these resources are monopolized by the user device for security.

Different from the traditional MEN, the user-centering mobile edge network is a dynamic network architecture along with the path of moving UEs. As shown in Figure 2, the wireless mobile devices carried by users are randomly connected to the edge network, that leads to the rebuilding of a new network topology. Thus, the edge server needs to constantly adjust the assignment of tasks.

![Figure 2. Architecture of User-centered edge mobile network.](image)

As the user’s location changes in ENs, so does the serving ES (Figure 2). That may result in available resources providing to users to be different at a certain moment. In mobile user-centric edge network, the resources available around the user can be adjusted timely to cope with random changes by the user’s location.

Given $U = \{U_i | i = 1, 2, 3, \ldots, M\}$ represents the set of users in the network, where $M$ is the number of users; $Dev = \{Dev_i | i = 1, 2, 3, \ldots, D\}$ represents the set of devices in the network, $D$ is the number of devices; $ES = \{ES_i | i = 1, 2, 3, \ldots, S\}$ represents the set of ES, where $S$ is the number of edge servers in the network. The user-centered edge service network can be expressed as a triple $(U, D, ES)$.

$EN = \{EN_i | i = 1, 2, 3, \ldots, EC\}$ represents the set of ENs, and $EC$ is the number of ENs in the network. Each edge network can be represented by a quintuple $EN_i = (UQ_i, RQ_i, TQ_i, RS_{cur_i}, RS_{max_i})$, where $UQ$ represents the user queue requesting the computing devices in MEN, $RQ$ represents the available resource queue, $TQ$ represents the task request queue in the EN, $RS_{cur}$ represents the current usage of edge network resources, and $RS_{max}$ represents the maximum load capacity of the EN (including the upper limit of connection number, CPU occupancy rate, bandwidth, etc.). All sets above must obey the constraints in Equation (1).

$$\begin{align*}
\sum_{i=1}^{S} |UQ_i| &\leq M \\
\forall |RQ_i| &\leq |Dev_i|, 1 \leq i \leq S \\
\sum_{i=1}^{S} RS_{cur_i} &\leq \sum_{i=1}^{S} RS_{max_i}
\end{align*}$$

(1)
Since the number of resources and network performance varies over time, we set $EN_k(t) = (UQ_i(t), RQ_i(t), TQ, RS_{\text{cur}}(t), RS_{\text{max}})$ to represent the $EN_k$ at time $t$, where $RS_{\text{max}}$ is a constant independent of time; $Dev^T_1(t)$ represents the set of devices in $EN_k$ at $t$.

### 3.2. User Trajectory Prediction Model

Compared with the conventional prediction method, the user trajectory prediction model proposed in this paper is a non-priori prediction solution. Just according to the current motion state of the user, we guess the possible position of the user in the next time slice. The user’s movement refers to the random walk model which can change the direction and speed of the users randomly.

Assuming that the speed of user movement varies with time, which is denoted by $v(t)$ at time $t$. For every fixed time slice $T$, the edge server needs to update the location of the user. Then, the moving distance $d$ of the user in a single $T$ can be expressed as $d = T * v(t)$, which directly affects the selection of the target device for the current request task offloading.

When the $T$ infinitely approaches a very small value, we regard that the UEs movement in a straight line with a constant direction and speed. Therefore, we can deduce a circular region with radius $r$ according to Equation (2) (red notation in Figure 3), which is the user’s activity range in the next time period.

Equation (2) represents that the UEs move in a straight line with a constant direction and speed, when the $\epsilon$ is a small value. Therefore, we can deduce a circular region with radius $r$ according to Equation (2) (red notation in Figure 3), which is the user’s activity range in the next time period.

$$r(t) = \lim_{T \to \epsilon} T \cdot v(t),$$

where $\epsilon$ is extremely small values, and $T$ is the time interval for system polling.

According to the radius $r$ determined by Equation (2), there is an unambiguous circular region for the moving UEs. In Figure 3, $\alpha$, $\beta$, $\gamma$, and $\theta$ are the vector angles between the velocity vector $v$ and different adjacent communication ESs, which represent the range of service areas that the user is about to reach for different base stations. These angles divide the entire circular region into several fan-shaped regions. We choose the sector area corresponding to the smallest two angles as the range with the greatest probability of user’s next location, which is the region corresponding to $\angle \alpha$ and $\angle \beta$ in Figure 3. Therefore, the position of the moving user at next time is most likely to fall within the domain delimited by Equation (3).

$$\text{Area} = \text{Area}(\beta, r) + \text{Area}(\alpha, r)$$

where

$$\begin{align*}
\text{Area}(\beta, r) &= \pi / 360 \cdot \beta \cdot r^2 \\
\text{Area}(\alpha, r) &= \pi / 360 \cdot \alpha \cdot r^2
\end{align*}$$

The user activity range prediction model described in Figure 3 can outline the next location based on the current user’s status, which is an important basis for the target device selection when the subsequent terminal device task is unloaded. This model can effectively narrow down the search scope for available devices in the offloading algorithm.

### 3.3. Parallelizable Task Offloading Cost Model

In Figure 3, the candidate ES is determined by the size of the overlap between the sector area and the coverage area of the BS. The resources of current EN and alternative ES will participate in the offloading of the network task. In this section, it mainly introduces the offloading cost evaluation model in parallelizable task offloading strategy.

$BS = \{BS_i|j = 1, 2, 3, \ldots, BN\}$ represents the set of edge server BSs, $BN$ is the number of BSs, $Task = \{\text{task}_i|i = 1, 2, 3, \ldots, N\}$ represents the set of tasks in the MEN, and $N$ is the total number of tasks that can be requested in the network. Furthermore, each task has a parallelizable subtask sequence $\text{sub}_\text{task}_{ij}$, which is represented by a $k$-tuple $\text{task}_i = \{\text{sub}_\text{task}^1_i, \text{sub}_\text{task}^2_i, \ldots, \text{sub}_\text{task}^k_i\}$.
For simplicity, each BS is represented by a quadruple $BS_j = \{EN, CU, BW, TQ\}$, where $EN$ is the edge network to which the BS belongs, it contains the available resources in the network; $CU$ represents CPU occupancy; $BW$ represents the bandwidth usage of edge servers; $TQ$ represents the task queue on the BS. Assuming that all BS servers in MEN possess same performance, so that $CPU_{MAX}$, $Bandwidth_{MAX}$ and $BS_{Task\_MAX}$ respectively represent the upper limit of the server’s computing power, bandwidth, and the number of tasks carried.

Each sub-task is represented by $\text{sub\_task}_i^k = (CPU_i^k, Data_i^k, Tolerance_{maxi}^k)$, where $CPU$ represents the number of CPU clock cycles of task; $Data$ is the amount of data generated by each task in the execution process; $Tolerance_{max}$ is the maximum delay of execution time of each sub-task. In this paper, we mainly considers the consumption of the task in the CPU clock cycle on the computing device and the data exchange in the network to evaluate the offloading strategy.

3.3.1. Energy Consumption Calculation Model of Task Offloading

All resources around moving UEs will be used as the targets to unload each task, and the feasibility of this solution has been demonstrated in [44–46]. In Equation (4), $\xi_c, \xi_1$ are CPU clock cycle rated energy consumption on remote cloud servers (CS), edge servers (ES), and terminal devices (TD), $\text{task}_i$ can be executed in parallel with $K$ sub-tasks which have been divided by the designers of the task in advance, where $K_1$, $K_2$ and $K_3$ represent the number of sub-tasks assigned on CS, ES and TD. According to the offloading strategy proposed in the article, the energy consumption required for the execution of $\text{task}_i$ is shown in Equation (4).

$$
\begin{align*}
\text{Energy\_CPU\_task}_i &= \xi_c \cdot \sum_{\delta=1}^{K_1} CPU_{\text{sub\_task}}^{\delta} \\
&+ \xi_e \cdot \sum_{\omega=1}^{K_2} CPU_{\text{sub\_task}}^{\omega} + \xi_t \cdot \sum_{\chi=1}^{K_3} CPU_{\text{sub\_task}}^{\chi}; \\
\text{where } K_1 + K_2 + K_3 &= |\text{task}_i| 
\end{align*}
$$

In Equation (4), $\xi_c \cdot \sum_{\delta=1}^{K_1} CPU_{\text{sub\_task}}^{\delta}$, $\xi_e \cdot \sum_{\omega=1}^{K_2} CPU_{\text{sub\_task}}^{\omega}$, $\xi_t \cdot \sum_{\chi=1}^{K_3} CPU_{\text{sub\_task}}^{\chi}$ represent the energy consumption of CPU cycles of tasks in cloud, edge and end device respectively.
Generality, the edge network described in this paper is a random network with multi-task and multi-user access, the same task may be requested by multiple users at the same point in time. So, we assume that $cnt^t_i$ represents the total number of requests for task at $t$ time, and then the CPU energy consumption for the entire network is shown in Equation (5).

$$\text{Energy}_{CPU}(t) = \sum_{i=1}^{\text{Task}} \text{Energy}_{CPU_{task_i}} * cnt^t_i \quad (5)$$

### 3.3.2. Network Delay Calculation Model

Network delay is another important indicator to measure the offloading strategy. It includes the execution time of the task itself, the synchronization waiting time between subtasks, and the time for data transmission. However, the transmission is the main reason for excessive network latency [11]. Equation (6) is the total data amount of task$i$, where $\text{sub}_{D}^k_i$ is the data amount of all sub-tasks of task$i$.

$$D_{task_i} = \sum_{k=1}^{K} \text{sub}_{D}^k_{task_i}, \quad \text{where } K = |\text{task}_i| \quad (6)$$

Commonly, the data in the network can be divided into upload data and download data, and $\text{sub}_{D}^k_{task_i}$ in Equation (6) can be expressed in Equation (7).

$$\text{sub}_{D}^k_{task_i} = \text{up}_{D}^k_{task_i} + \text{down}_{D}^k_{task_i} \quad (7)$$

In Equation (7), $\text{up}_{D}^k_{task_i}$ and $\text{down}_{D}^k_{task_i}$ represent sub-task upload and download data; If the communication rate is represented by $B$, the execution time of task$i$ in the network can be represented as:

$$T_{\_delay_{task_i}} = \sum_{k=1}^{T_{\text{task}_i}} (\text{up}_{D}^k_{task_i} / B + \text{down}_{D}^k_{task_i} / B) + \text{sys}_{\_delay_{task_i}} \quad (8)$$

In Equation (8), $\text{sys}_{\_delay_{task_i}}$ is the time delay caused by EN itself. $cnt^t_i$ represents the number of requests for task$i$, the network delay of the whole network at $t$ can refer to Equation (9).

$$T_{\_delay}(t) = \sum_{i=1}^{\text{Task}} T_{\_delay_{task_i}} * cnt^t_i = \sum_{i=1}^{\text{Task}} (\sum_{k=1}^{T_{\text{task}_i}} (\text{up}_{D}^k_{task_i} / B + \text{down}_{D}^k_{task_i} / B + \text{sys}_{\_delay_{task_i}})) * cnt^t_i \quad (9)$$

### 3.4. Problem Statement

In this section, we study the impact of user mobility behavior on task offloading in mobile edge network, and propose a task offloading solution for parallel tasks based on the user trajectory prediction model. We transform this issue into the problem of finding an optimal set of devices with the least cost for task offloading in a mobile user-centering network, which is represented as multi-objective optimization function in Equation (10).

$$\text{Cost}_{task_i} = T_{\_delay_{task_i}} + \text{Energy}_{CPU_{task_i}} \quad (10)$$

where $\text{Cost}_{task_i}$ is the cost function of offloading task$i$. Assuming $A$ is the set of all assignments of task at $t$, the objective function of offloading cost (Equation (10)) can be transformed into Equation (11) which is regarded as a function of $t$ and $A$. Thus, the
execution time of the task and the computational resources consumed become the optimization targets.

\[ \text{Cost}(A,t) = T_{\text{delay}}(A,t) + \text{Energy}_{\text{CPU}}(A,t) \]  

(11)

To put it clearly, we need to convert the \( \text{Energy}_{\text{CPU}}(A,t) \) into computer clock cycles which is measured in time. Therefore, the units of \( T_{\text{delay}}(A,t) \) and \( \text{Energy}_{\text{CPU}}(A,t) \) are then unified. The assumptions and conversion rules are listed in the experimental section. Considering the parallelization of tasks, the each terms in Equation (11) are non-negative. The minimum of offloading cost is obtained when both of \( T_{\text{delay}} \) \( \text{task}_i \) and \( \text{Energy}_{\text{CPU}} \) \( \text{task}_i \) reach minimum values. Therefore, the process of finding the optimal solution of \( \text{Cost}(A,t) \) can be converted to two minimization functions (Equation (12)).

\[
\begin{cases}
\text{Min}\{T_{\text{delay}}(A,t)\} \\
\text{Min}\{\text{Energy}_{\text{CPU}}(A,t)\}
\end{cases}
\]  

(12)

Based on the constraint of Equation (12), we try to seek an optimal \( A \) all tasks at \( t \). However, it is hard to search a solution of \( \text{Cost}(A,t) \) to satisfy both of the conditions in Equation (12). We assume that \( A \) is the subset of \( A \) which makes the \( T_{\text{delay}}(A,t) \), and \( B \) is the subset of \( A \) which makes the \( \text{Energy}_{\text{CPU}}(A,t) \). Therefore, the optimal solution Equation (11) is divided into two cases. If \( A \cap B \neq \Phi \), there is a optimal solution for \( \text{Cost}(A,t) \). Otherwise, \( \text{Cost}(A,t) \) only has a suboptimal solution.

4. Mobility-Aware Parallelizable Task Offloading Strategy (MPTOS)

Based on the models discussed in the previous section, this section describes the process of NPTP and PTOSBPP in detail. A mobility-aware parallelizable task offloading strategy is proposed here, the task offloading decision is based on the NPTP.

4.1. Overview of the MPTOS

As the UEs shuttle in the EN, the task requesting of moving UEs is irregular. The ESs perceive the user’s location according to task application from the end devices and passively provide service for them. In order to solve the NP-hard problem \([30,47,48]\), task unloading strategy must be able to adapt to the changes of EN caused by the user’s movement. It is MPTOS that can dynamically modify the offloading policy as the ENs topology changes. MPTOS also includes user trajectory prediction algorithm without prior information and task offloading algorithm for parallelizable task. To demonstrate more clearly the task offloading of MPTOS, we present a complete description of the task offloading and decision-making process in Algorithms 1 and 2 in this section.

4.2. Non-Prior Trajectory Prediction Algorithm

Non-prior information trajectory prediction algorithm is the basis for the implementation of following service offloading strategy. The algorithm predicts the location of the UEs in the next slice by the instantaneous position and current speed of the UEs. Then, PTOSBPP unloads the request service received by the current device to the target edge device in advance according to the feedback results of NPTP at \( t \), which is convenient for terminal user service requests at \( t+1 \).

According to the model in Figure 3, it quantifies the position angle between the motion vector \( \vec{v} \) and the edge servers, which determines the next hop ES. Then, the offloading tactics works on the current available device queue including the next target ES. The algorithm of NPTP is shown in Algorithm 1.
Algorithm 1 Non-prior Trajectory Prediction algorithm (NPTP)

Require:
- cu_dev_coordinate: Current coordinate of the user’s device
- cur_BS_coordinate: Current coordinate of the servicing BS
- speed: Current user’s speed that includes a direction and a value of mobile rate
- R: Radius of BS covered area

Ensure:
- Next_BS: Candidate device for the task offloading

1: Mobile_r ← |speed| * T
2: Neighbor_BS ← get_neighborBS()
3: dis ← distance(cur_dev_coordinate, cur_BS_coordinate)
4: Mobile_area ← pi * Mobile_r * Mobile_r
5: if dis ≤ R * Mobile_r then
6:     Next_BS ← cur_BS_coordinate
7: else
8:     //Get the angle between the velocity vector and different adjacent BS, and save the result in Neighbor_angle
9:     for i ∈ (0, len(Neighbor_BS)) do
10:        temp_v ← make_vector(Neighbor_BS [i], cur_dev_coordinate)
11:        Neighbor_angle[i] ← get_angle(speed, temp_v)
12:     end for
13:     min_angle ← Min(Neighbor_angle)
14:     Next_BS ← getBS_from_angle(min_angle)
15: end if
16: return Next_BS

Algorithm 2 explains the scheme of PTOSBPP in detail. The US sends different types of task requests to different edge servers by the user’s trajectory at each time slice, the PTOSBPP will be running on each ESs to offload the tasks to different candidate equipment.

4.3. Parallelizable Task Offloading Strategy Based Position Prediction

In order to adapt to the movement of UEs, the algorithm of PTOSBPP, combined with the characteristics of parallel task in ENs, has to select the optimal unloading target device from available device queues in each time slice according to Equation (12) in Section 3.4. Thus, the strategy of PTOSBPP conform the following principles to distribute the tasks in ENs:

1) MEN does its best to satisfy users’ resource requests. In addition, the time dependence of edge task on UEs’ displacement should be considered comprehensively in the provision of services by MEN.

2) Large tasks that take up more CPU cycles are preferentially distributed to the devices with better bandwidth and sufficient-resources ESs. That can shorten the execution time of a single task.

3) The number of tasks offloaded on ESs is restricted by increasing the parallelization of subtasks. In other words, when network delay caused by mobility cannot be avoided, cloud server and end device collaboration ought to be utilized to its full potentials.

Algorithm 2 explains the scheme of PTOSBPP in detail. The US sends different types of task requests to different edge servers by the user’s trajectory at each time slice, the PTOSBPP will be running on each ESs to offload the tasks to different candidate equipment.
Algorithm 2 Parallelizable Task Offloading Strategy Based Position Prediction (PTOSBPP)

Require:
M,N: Number of users and subtask of per task
subtask[M][N]: Subtasks list of task
cur_dev,cur_BS,cloud: Current user’s device, servicing BS and cloud servers
pre_BS[M]: Candidate BSs for M users selected in the Algorithm 1

Ensure:
A[M][N]: Device assignment list of each subtask contained in subtask[M][N]
1. Initialization of the information of UEs, edge servers, cloud and tasks
2. Make array of subtask randomly
3. for i ∈ (0, M ) do
4. k ∈ {et|exe_Time(subtask[i][et]) > T}
5. t ∈ {et|exe_Time(subtask[i][et]) ≤ T}
6. if pre_BS[i] == cur_BS then
7. if len(cur_dev.TQ) ≤ MAX then
8. A[i][t] ← cur_dev
9. else
10. A[i][t] ← cur_BS
11. end if
12. if len(cur_BS.TQ) ≤ BS_Task_MAX then
13. A[i][k] ← cur_BS
14. else
15. A[i][k] ← cloud
16. end if
17. end if
18. for j ∈ (0, N) do
19. if len(cur_BS.TQ) ≤ BS_Task_MAX then
20. A[i][t] ← cur_BS
21. else
22. A[i][t] ← cur_dev
23. end if
24. if len(pre_BS[i].RQ) ≤ BS_Task_MAX then
25. A[i][k] ← pre_BS
26. else
27. A[i][k] ← cloud
28. end if
29. end for
30. end for
31. return A

5. Experiment and Discussion

In this section, we evaluate the proposed approaches by a series of experiments. All the experiments are conducted on a Windows OS equipped with Intel Core i5-8500 and 16 GB RAM. All of the algorithms are completed with python3.7 and pycharm IDE. We take a part of the EUA data set in these experiments, which contains the location information of 125 base station sites and 816 users in a Central Business District(CBD). EUA data set is a publicly released to facilitate research in Edge Computing, we can get it from the internet freely (https://www.opensourceagenda.com/projects/eua-dataset) (accessed on 12 October 2022). In addition, we also construct and use a task data set with 8 types, and each task is divided into several sub-tasks. According to the random walk model in Section 3, the users move in the area of a Central Business District, and send different task requests to the edge server in time slices stochastically.

5.1. Benchmark Approaches

As there is no completely similar experimental scenario to the issue discussed in this paper, we abstract three experimental schemes from the previous literature [9,14,19,21,24,35]
to compare the performance of our methods. The tasks offloading method, proposed in this paper, is based on user trajectory prediction and introduces the parallel execution strategy of sub-tasks. By combining the random walk model with the EUA data set, we have designed three experimental scenarios to verify the effectiveness of our method. The following benchmark experiments are different combinations of NPTP and parallelizable task offloading strategy, and the detail are shown in Table 2.

Table 2. Detail of Different Experiments.

| Experimental Scheme                                      | Position Prediction | Parallel Strategy |
|----------------------------------------------------------|---------------------|-------------------|
| Random strategy (RS)                                     | ×                   | ×                 |
| Non-position Prediction Parallel Strategy (NPPS)         | ×                   | ✓                 |
| Non-parallel Strategy Based Position Prediction (NPSBPP)  | ✓                   | ×                 |
| Parallelizable Task Offloading Strategy Based Position Prediction (PTOSBPP) | ✓                 | ✓                 |

(1) Random strategy (RS)
RS randomly assigns the undivided task to the devices surrounding the UEs. In RS, since the tasks are not split into multiple sub-tasks, the UEs, ESs and cloud servers cannot execute in parallel. If a task can be completed in one time slice, it will be unloaded to the neighbor ES. Otherwise, the UE requesting the task is responsible for completing the task.

(2) Non-position Prediction Parallel Strategy (NPPS)
In non-position prediction parallel strategy, the task is split into several parallel sub-tasks. That is the same task at the same time that can be divided into multiple sub-tasks and run in parallel between the UEs, ESs and cloud servers. This strategy may improve the execution efficiency of tasks within a single time slice. However, in order to avoid repeated requests during UEs moving and the failure to feedback the results to the terminal devices, the sub-tasks whose execution time exceeds one time slice are unloaded to the UEs for execution firstly.

(3) Non-parallel Strategy Based Position Prediction (NPSBPP)
In NPSBPP, it introduces the user trajectory prediction model mentioned in Section 3, and the edge resources predicted by model are also considered as the candidate devices for unloading the current task. Meanwhile, the task that runs more than one time slice is offloaded beforehand to the cloud or the ES at the next slice.

5.2. Experimental Setting and Evaluation Standard
To improve the universality of our approaches, our experiment Section 5 groups of fixed number of users from 816 users in the EUA data set. These users move at different speed based on the random walk model proposed in Section 3, and request for different types of tasks randomly at each time slice. The simulation experiment is carried out under four different task offloading strategies. In order to ensure the consistency of the experiment, the parameters in the simulation environment are set as follows in Table 3:

To comprehensively analyze the performance of our approaches, we choose a series of comparison indicators to evaluate our method, such as task offloading predicting hit rate, network bandwidth, and task execution efficiency. Where task offloading predicting hit rate is equal to the accuracy rate of user’s position prediction, bandwidth is mainly used to measure the burden of task parallel strategy, and the task execution efficiency is quantified the task acceleration ratio between parallel mode and serial mode.

5.3. Results and Discussion
In this section, we display the experimental results from the predicted hit rate of the mobile users, bandwidth, and task execution efficiency and compare them with RS, NPPS and NPSBPP.
Table 3. Experimental Parameters Setting.

| Object                | Parameter Setting                                                                 |
|-----------------------|-----------------------------------------------------------------------------------|
| End-users             | Select 80 users as a group randomly from 816 users in EUA dataset, that might ensure the users are distributed within the CDB randomly; each experimental scenario contains 5 groups users. |
| Task                  | 8 different types of task, each task is represented by a subtask set \( \text{subtask}_{i} \) \( \forall i \leq N \), where \( N \) is the total number of sub-tasks. We also assume that each task contains 2 to 5 sub-tasks. |
| Cloud/MEN Servers     | There are 125 edge servers and one cloud server in the CBD area, and we assuming the resource of cloud is sufficient. |
| Coverage Radius of BS (m) | 150 \( \leq R \leq 400 \) |
| Bandwidth             | We assume that the bandwidth between the end, the edge and the cloud is adequate. \( BW_{end \_to \_edge} = 100 \text{ MB/s}; BW_{edge \_to \_cloud} = 1000 \text{ MB/s} \) |
| Time Slot             | \( T \) is a constant, we set \( T = 2s \); and the number of task requests generated by each user in each time slice is in the range \([0,3]\) |
| Rated Cycle of CPU    | The cycle of CPU of Cloud \( \delta_{1} = 10 \) |
|                       | The cycle of CPU of MEN servers \( \delta_{2} = 40 \) and the number of task requests generated by each user in each time slice is in the range \([0,3]\) |
| Moving speed (m/s)    | \( \delta_{1} : 2 \leq \delta_{1} \leq 6 \) and \( \delta_{2} : 6 \leq \delta_{2} \leq 10 \) |
|                       | \( \delta_{3} : 10 \leq \delta_{3} \leq 13 \) and \( \delta_{4} : 13 \leq \delta_{4} \leq 16 \) |

5.3.1. Task Offloading Hit Rate

Among the four strategies, NPPS and PTOSBPP adopt the position prediction method proposed in Section 3 to offloading the tasks, so that the task offloading hit rate and its comparison will be obtained from these two experiments. In each experimental scenario, we simulate the users’ trajectory by the random walk model, and collect the track information of 400 randomly selected users in 5 groups. According to prediction model in Figure 3, we compare the location of each user at time slice of \( t \) and \( t + 1 \) by counting the number of times that the front and back positions are the same. Furthermore, we make statistics of these data according to 4 ranges of speed to observe the impact of rate on the prediction accuracy. Figure 4a,b present the predicting hit rate of NPPS and PTOSBPP among different user sets. Since no prediction strategy is used in RS and NPPS, there is no need to compare this item in these two cases.

Each line of Figure 4 displays the hit ratio trend for different user data sets. As shown in Figure 4, with the increase of user’s moving speed, the accuracy rate gradually decreases. However, they still remain above 75%. In other words, when there is no sufficient additional information, the faster the user moves, the harder it is to predict the user location.

5.3.2. Bandwidth

Any task unloading solution, especially collaboration strategy of UEs, ESs and clouds, has a certain impact on the EN. NPPS and PTOSBPP are offloading strategies for parallel tasks. In the simulation experiments, we have calculated the bandwidth occupancy of different offloading strategies at each time. Since more sub-tasks in the parallel offloading
strategy may be assigned to different edge devices for execution shown in Figure 5a, it can be seen from Figure 5b that the average bandwidth utilization of NPSBPP and PTOSBPP is higher than RS and NPPS. Nevertheless, all of the curves in Figure 5b do not reach half of the total bandwidth.

![Image](image1.png)

(a) The Number of Tasks Running on ESs or Cloud; (b) Bandwidth Occupancy Rate at Different Time.

5.3.3. Execution Efficiency of Task

Eight different types of task requests are randomly generated at each time slice in the simulation experiments. Each task can be divided into several sub-tasks. Under different strategies, we collected the task execution time of 400 random users, calculated the average ratio of the actual execution time to the theoretical time of all users’ requesting tasks. Figure 6a presents the average number of tasks or sub-tasks that needs to be unloaded in MEN at each point in time, and the Figure 6b shows that the average ratio of the actual task execution time to the theoretical practice time under different unloading scheme whose horizontal axis is the UEs’ moving speed. The actual task execution time of RS and NPPS is over four times more than theoretical value, while the NPSBPP and PTOSBPP are basically close to the theoretical time.

![Image](image2.png)

(a) Task Requests at Different Time Slice; (b) Average Execution Efficiency of Tasks among Different Cases.

As can be seen from Figure 6, the execution efficiency of programs in PTOSBPP and NPSBPP have obvious advantages over RS and NPPS. Since the parallel mechanism, the actual execution time of all tasks is close to the theoretical time in PTOSBPP, while the execution efficiency of NPSBPP is slightly higher than that of PTOSBPP. These results suggest that the prediction of UEs’ position in task offloading can effectively improve the execution efficiency of tasks in MEN, and the parallel division of tasks can shorten the execution time of tasks as well.

6. Conclusions and Future Work

In this paper, we identified the importance of prediction of user movement trajectory in task offloading of MEN network. We proposed a task offloading strategy for paralleling edge network based on a prediction model of non-prioritized trajectory. The time complexity of two algorithms has certain advantages to some extent. For example, the time complexity of Algorithm 1 seems to be $O(n)$, but the $n$ is the number of neighbor edge servers which is not more excessive. Meanwhile, the time complexity of Algorithm 2 is
O(M × N), where M and N are the number of UEs and subtask. As the data structure of device assignment list, the temporal consumption of this algorithm needs to improve in the future.

Experimental results show that the prediction of user location not only improves the hit rate of target edge server for task offloading in parallel networks, but also effectively improves the execution efficiency of tasks and the utilization rate of network resources. Meanwhile, it demonstrated that the user mobility rate is the key factor affecting the above performance. However, we found that the hit rate of task offloading prediction decreased as the user moved faster. Therefore, how to improve the hit rate of the task offloading in fast moving scenarios is one of our research challenges and key points in the future.

Additionally, mobile user-centered architecture is a type of dynamic networking approach in MEN, it is appropriate for the accessing way of MEN. In these networks, it may provide a personalized list of available devices for any end user, and increase the equipment utilization of the edge network. Since the user device participates in managing the resource of edge networks, the security of the network is an inevitable issue, which also is a potential area for research in our next works.

Author Contributions: Conceptualization, P.H. and B.Y.; methodology and experiment, P.H. and L.H.; software and data curation, J.-S.P. and J.S.; writing—original draft preparation, P.H.; writing—review and editing, J.-S.P. and J.S.; funding acquisition, J.S. and L.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the project of Research on Key Technologies for water and sediment simulation and intelligent decision of Yellow River (Grant No.201400211000).

Institutional Review Board Statement: Not applicable for studies not involving humans or animals.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author upon reasonable request.

Acknowledgments: The corresponding author (J. Shang) was supported by the project of Research on Key Technologies for the Construction and Service of Yellow River Simulators for Suppercomputing funded by Henan Provincial Department of Science and Technology (Grant No.201400210900).

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Abbreviations
The following abbreviations are used in this manuscript:

| Abbreviation | Description |
|--------------|-------------|
| IoT | Internet of Things |
| MEN | Mobile edge network |
| UE | User equipment |
| QoS | Quality of service |
| QoE | Quality of Experience |
| EN | Edge network |
| UCEN | User-centering edge network |
| NPTP | Non-prior trajectory prediction model |
| MUCEN | Mobile user-centering edge network |
| CS | Cloud server |
| BS | Base station |
| ES | Edge server |
| TD | Terminal device |
| PDPM | prior displacement prediction model |
| MPTOS | Mobility-aware Parallelizable Task Offloading Strategy |
PTOSBPP  Parallelizable Task Offloading Strategy Based Position Prediction
RS  Random strategy
NPPS  Non-position Prediction Parallel Strategy
NPSBPP  Non-parallel Strategy Based Position Prediction

References
1. Akherfi, K.; Gerndt, M.; Harroud, H. Mobile cloud computing for computation offloading: Issues and challenges. Appl. Comput. Inform. 2018, 14, 1–16. [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. Odun-Ayo, I.; Ananya, M.; Agono, F.; Goddy-Worlu, R. Cloud computing architecture: A critical analysis. In Proceedings of the 2018 18th International Conference on Computational Science and Applications (ICCSA), Melbourne, VIC, Australia, 2–5 July 2018; pp. 1–7.
4. Zhang, A.N.; Chu, S.C.; Song, P.C.; Wang, H.; Pan, J.S. Task Scheduling in Cloud Computing Environment Using Advanced Phasmatodea Population Evolution Algorithms. Electronics 2022, 11, 1451. [CrossRef]
5. Sabella, D.; Vaillant, A.; Kuure, P.; Rauschenbach, U.; Giust, F. Mobile-edge computing architecture: The role of MEC in the Internet of Things. IEEE Consum. Electron. Mag. 2016, 5, 84–91. [CrossRef]
6. Nguyen, T.T.; Pan, J.S.; Dao, T.K.; Chu, S.C. Load balancing for mitigating hotspot problem in wireless sensor network based on enhanced diversity pollen. J. Inf. Telecommun. 2018, 2, 91–106. [CrossRef]
7. 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.
8. Pan, J.S.; Li, G.C.; Li, J.; Gao, M.; Chu, S.C. Application of the Novel Parallel QUasi-Affine TRansformation Evolution in WSN Coverage Optimization. In Advances in Intelligent Systems and Computing; Springer: Berlin/Heidelberg, Germany, 2022; pp. 241–251.
9. Shakarami, A.; Ghobaei-Arani, M.; Shahidinejad, A. A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective. Comput. Netw. 2020, 182, 107496. [CrossRef]
10. Mach, P.; Beecvar, Z. Mobile edge computing: A survey on architecture and computation offloading. IEEE Commun. Surv. Tutor. 2017, 19, 1628–1656. [CrossRef]
11. Khan, W.Z.; Ahmed, E.; Hakak, S.; Yaqoob, I.; Ahmed, A. Edge computing: A survey. Future Gener. Comput. Syst. 2019, 97, 219–235. [CrossRef]
12. Shahzadi, S.; Iqbal, M.; Dagiuclas, T.; Qayyum, Z.U. Multi-access edge computing: Open issues, challenges and future perspectives. J. Cloud Comput. 2017, 6, 30. [CrossRef]
13. Xu, J.; Chen, L.; Zhou, P. Joint service caching and task offloading for mobile edge computing in dense networks. In Proceedings of the IEEE INFOCOM 2018-IEEE Conference on Computer Communications, Honolulu, HI, USA, 15–19 April 2018; pp. 207–215.
14. Atanasov, I.; Pencheva, E.; Namekov, A.; Trfonov, V. Provisioning of UE Behavior Prognostic by Multiaccess Edge Computing In Proceedings of the 2019 International Symposium on Networks, Computers and Communications (ISNCC), Istanbul, Turkey, 18–20 June 2019; pp. 1–6.
15. Dong, L.; Satpute, M.N.; Shan, J.; Liu, B.; Yu, Y.; Yan, T. Computation offloading for mobile-edge computing with multi-user. In Proceedings of the 2019 IEEE 39th international conference on distributed computing systems (ICDCS), Dallas, TX, USA, 7–9 July 2019; pp. 841–850.
16. Ou, S.; Yang, K.; Liotta, A. An adaptive multi-constraint partitioning algorithm for offloading in pervasive systems. In Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications (PERCOM’06), Pisa, Italy, 13–17 March 2006; pp. 10–125.
17. Pan, J.S.; Fan, F.; Chu, S.C.; Du, Z.; Zhao, H. A node location method in wireless sensor networks based on a hybrid optimization algorithm. Wirel. Commun. Mob. Comput. 2020, 1–14. [CrossRef]
18. Nasrín, W.; Xie, J. SharedMEC: Sharing clouds to support user mobility in mobile edge computing. In Proceedings of the 2018 IEEE International Conference on Communications (ICC), Kansas City, MO, USA, 20–24 May 2018; pp. 1–6.
19. Nadembeaga, A.; Hafid, A.S.; Brisebois, R. Mobility prediction model-based service migration procedure for follow me cloud to support QoS and QoE. In Proceedings of the 2016 IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, 22–27 May 2016; pp. 1–6.
20. Elgazzar, K.; Martin, P.; Hassanein, H.S. Cloud-assisted computation offloading to support mobile services. IEEE Trans. Cloud Comput. 2014, 4, 279–292. [CrossRef]
21. Nadembeaga, A.; Taleb, T.; Hafid, A. A destination prediction model based on historical data, contextual knowledge and spatial conceptual maps. In Proceedings of the 2012 IEEE International Conference on Communications (ICC), Ottawa, ON, Canada, 10–15 June 2012; pp. 1416–1420.
22. Nadembeaga, A.; Hafid, A.; Taleb, T. A path prediction model to support mobile multimedia streaming. In Proceedings of the 2012 IEEE International Conference on Communications (ICC), Ottawa, ON, Canada, 10–15 June 2012; pp. 2001–2005.
23. Li, W.; Wang, F.; Pan, Y.; Zhang, L.; Liu, J. Computing Cost Optimization for Multi-BS in MEC by Offloading. Mob. Netw. Appl. 2020, 27, 1–13. [CrossRef]
24. Ouyang, T.; Zhou, Z.; Chen, X. Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing. *IEEE J. Sel. Areas Commun.* 2018, 36, 2333–2345. [CrossRef]
25. Kumar, K.; Liu, J.; Lu, Y.H.; Bhargava, B. A survey of computation offloading for mobile systems. *Mob. Netw. Appl.* 2013, 18, 129–140. [CrossRef]
26. Chen, X.; Pu, L.; Gao, L.; Wu, W.; Wu, D. Exploiting massive D2D collaboration for energy-efficient mobile edge computing. *IEEE Wirel. Commun.* 2017, 24, 64–71. [CrossRef]
27. Xian, C.; Lu, Y.H.; Li, Z. Adaptive computation offloading for energy conservation on battery-powered systems. In Proceedings of the 2007 International Conference on Parallel and Distributed Systems, Hsinchu, Taiwan, 5–7 December 2007; pp. 1–8.
28. Wolski, R.; Gurun, S.; Krintz, C.; Nurmi, D. Using bandwidth data to make computation offloading decisions. In Proceedings of the 2008 IEEE International Symposium on Parallel and Distributed Processing, Miami, FL, USA, 14–18 April 2008; pp. 1–8.
29. 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:Berlin/Heidelberg, Germany, 2018; pp. 355–370.
30. Chen, X.; Jiao, L.; Li, W.; Fu, X. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Trans. Netw.* 2015, 24, 2795–2808. [CrossRef]
31. Hirsch, M.; Mateos, C.; Zunino, A. Augmenting computing capabilities at the edge by jointly exploiting mobile devices: A survey. *Future Gener. Comput. Syst.* 2018, 88, 644–662. [CrossRef]
32. Mukherjee, A.; De, D. Femtolet: A novel fifth generation network device for green mobile cloud computing. *Simul. Model. Pract. Theory* 2016, 62, 68–87. [CrossRef]
33. Chun, B.G.; Maniatis, P. Augmented smartphone applications through cloud edge execution. In Proceedings of the HotOS, Ann Arbor, MI, USA, 1–3 June 2009; Volume 9, pp. 8–11.
34. ur Rehman, M.H.; Chee, S.L.; Wah, T.Y.; Iqbal, A.; Jayaraman, P.P. Opportunistic computation offloading in mobile edge cloud computing environments. In Proceedings of the 2016 17th IEEE International Conference on Mobile Data Management (MDM), Porto, Portugal, 13–16 June 2016; Volume 1, pp. 208–213.
35. Liu, L.; Zhao, M.; Yu, M.; Jan, M.A.; Lan, D.; Taherkordi, A. Mobility-aware multi-hop task offloading for autonomous driving in vehicular edge computing and networks. *IEEE Trans. Intell. Transp. Syst.* 2022, 1–14 [CrossRef]
36. Long, T.; Ma, Y.; Xia, Y.; Xiao, X.; Peng, Q.; Zhao, J. A Mobility-Aware and Fault-Tolerant Service Offloading Method in Mobile Edge Computing. In Proceedings of the 2022 IEEE International Conference on Web Services (ICWS), Barcelona, Spain, 10–16 July 2022; pp. 67–72.
37. Wu, C.; Peng, Q.; Xia, Y.; Lee, J. Mobility-aware tasks offloading in mobile edge computing environment. In Proceedings of the 2019 Seventh International Symposium on Computing and Networking (CANDAR), Nagasaki, Japan, 25–28 November 2019; pp. 204–210.
38. Prabhala, B.; La Porta, T. Spatial and temporal considerations in next place predictions. In Proceedings of the 2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Hong Kong, China, 26 April–1 May 2015; pp. 390–395.
39. Prabhala, B.; La Porta, T. Next place predictions based on user mobility traces. In Proceedings of the 2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Hong Kong, China, 26 April–1 May 2015; pp. 93–94.
40. Wang, Z.; Zhao, Z.; Min, G.; Huang, X.; Ni, Q.; Wang, R. User mobility aware task assignment for mobile edge computing. *Future Gener. Comput. Syst.* 2018, 85, 1–8. [CrossRef]
41. Hui, G.; Rui, L.L.; Gao, Z.P. V2V Task Offloading Algorithm with LSTM-based Spatiotemporal Trajectory Prediction Model in SVCNs. *IEEE Trans. Veh. Technol.* 2022, 1–16. [CrossRef]
42. Aissioui, A.; Ksentini, A.; Gueroui, A.M.; Taleb, T. On enabling 5G automotive systems using follow me edge-cloud concept. *IEEE Trans. Veh. Technol.* 2018, 67, 5302–5316. [CrossRef]
43. Sun, M.; Zhou, Z.; Wang, J.; Du, C.; Gaaloul, W. Energy-efficient IoT service composition for concurrent timed applications. *Future Gener. Comput. Syst.* 2019, 100, 1017–1030. [CrossRef]
44. Yang, L.; Cao, J.; Cheng, H.; Ji, Y. Multi-user computation partitioning for latency sensitive mobile cloud applications. *IEEE Trans. Comput.* 2014, 64, 2253–2266. [CrossRef]
45. Deng, S.; Huang, L.; Taheri, J.; Zomaya, A.Y. Computation offloading for service workflow in mobile cloud computing. *IEEE Trans. Parallel Distrib. Syst.* 2014, 26, 3317–3329. [CrossRef]
46. Du, J.; Zhao, L.; Feng, J.; Chu, X. Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee. *IEEE Trans. Commun.* 2017, 66, 1594–1608. [CrossRef]
47. Mehrabi, M.; You, D.; Latzko, V.; Salah, H.; Reisslein, M.; Fitzek, F.H. Device-enhanced MEC: Multi-access edge computing (MEC) aided by end device computation and caching: A survey. *IEEE Access* 2019, 7, 166079–166108. [CrossRef]
48. Huang, M.; Liu, W.; Wang, T.; Liu, A.; Zhang, S. A cloud–MEC collaborative task offloading scheme with service orchestration. *IEEE Internet Things J.* 2019, 7, 5792–5805. [CrossRef]