electronics

MDPI

Article

Probabilistic Task Offloading with Uncertain Processing Times in Device-to-Device Edge Networks

Chang Shu, Yinhui Luo * and Fang Liu

School of Computer Science, Civil Aviation Flight University of China, Guanghan 618307, China; shuchang0530@163.com (C.S.); fangliu@cafuc.edu.cn (F.L.)
* Correspondence: loyinhv@163.com

Abstract: D2D edge computing is a promising solution to address the conflict between limited network capacity and increasing application demands, where mobile devices can offload their tasks to other peer devices/servers for better performance. Task offloading is critical to the performance of D2D edge computing. Most existing works on task offloading assume the task processing time is known or can be accurately estimated. However, the processing time is often uncertain until it is finished. Moreover, the same task can have largely different execution times under different scenarios, which leads to inaccurate offloading decisions and degraded performance. To address this problem, we propose a game-based probabilistic task offloading scheme with an uncertain processing time in D2D edge networks. First, we characterize the uncertainty of the task processing time using a probabilistic model. Second, we incorporate the proposed probabilistic model into an offloading decision game. We also analyze the structural properties of the game and prove that it can reach a Nash equilibrium. We evaluate the proposed work using real-world applications and datasets. The experimental results show that the proposed probabilistic model can accurately characterize the uncertainty of completion time, and the offloading algorithm can effectively improve the overall task completion rate in D2D networks.

Keywords: device to device; edge computing; Internet of Things; probabilistic model; task offloading

Citation: Shu, C.; Luo, Y.; Liu, F. Probabilistic Task Offloading with Uncertain Processing Times in Device-to-Device Edge Networks. *Electronics* **2024**, *13*, 1889. https://doi.org/10.3390/electronics13101889

Academic Editor: Martin Reisslein

Received: 3 April 2024
Revised: 6 May 2024
Accepted: 9 May 2024
Published: 11 May 2024

1. Introduction

Recent years have witnessed the increasing proliferation and popularity of mobile devices (MDs) in day-to-day life. It is envisioned that by the year 2022, around 12.3 billion interconnected MDs will surge at the network edge [1]. Meanwhile, novel applications are rapidly growing along with MDs, such as augmented reality (AR) [2], autonomous driving [3], and online games [4]. However, the unprecedented growth of these latency-critical and computationally intensive applications is hardly manageable on the resource-limited MDs. To tackle this issue, in recent years, the emergence of mobile edge computing has provided a ray of hope in addressing the challenge posed by computationally intensive applications for MDs.

The combination of device-to-device (D2D) communications and edge computing, denoted as D2D edge computing, has emerged as a promising research direction to unleash the full potential of edge computing [5–9]. With D2D edge computing, each MD offloads its tasks to peer MDs to utilize spare computation/communication resources, which enlarges network capacity and improves resource utilization.

In D2D edge computing, each device often has multiple candidate devices for offloading, and different offloading choices can lead to largely different offloading efficiencies. Therefore, the offloading decision is a critical problem in D2D edge computing. Many research efforts have been devoted to task offloading in D2D edge computing [6,10,11]. Most existing works study the optimal computation allocation policy to minimize the end-to-end delay. For example, Gong et al. [10] studied a delay-optimal distributed task offloading strategy that exploits wireless connected edge devices to perform distributed computing.

One of the fundamental assumptions in these works is that the processing times of all tasks are known or can be accurately estimated [6,10]. However, in practice, a task's processing time (TPT) is generally uncertain until it is processed to completion. Existing offloading decisions rely on knowledge of the TPT, i.e., the goal is to minimize the overall TPT [10] or maximize task completion rates while guaranteeing the TPT within deadlines [12]. Considering the variations in TPTs, existing works inevitably result in inaccurate offloading decisions. Dealing with uncertain TPTs is a non-trivial task due to the two challenges outlined below.

How to characterize varying task processing times? Existing efforts mainly employ averaged mathematical expectations to estimate uncertain TPTs [13,14]. However, for the same task, different input data will lead to a different number of iterations and different selections of judgment statements triggered, resulting in different execution times for the same program. For example, in visual relationship detection [15], the execution time depends on the number of detected targets included in the input image. The TPT with one target can be hundreds of times smaller than the TPT with twenty targets [15].

We can see that general average TPT values can hardly reflect the specific TPT in given scenarios. As a result, characterizing the TPT is essential to achieve accurate offloading decisions in various scenarios.

How to incorporate uncertain TPT variations into task offloading? Due to the lack of TPT distributions, existing works [13,14] use deterministic models for task offloading, where each task is determined to be offloaded or not according to its performance expectations. However, it is entirely possible that some tasks with poor expectations can achieve high offloading performance with certain probabilities. Such an impact is overlooked by deterministic model-based offloading schemes and will thus lead to inaccurate offloading decisions (as analyzed in Section 2). To exploit such potential opportunities, we need to (1) build a connection between the TPT model and optimization performance metrics (e.g., task completion rate), which is non-trivial since it is affected by multi-task contentions and queueing processes, and (2) support and incorporate the probabilistic TPT models and metrics into the offloading process.

To address the above challenges, we propose *Mature*, a ga*m*e-based probabilistic t*a*sk offloading scheme wi*t*h *u*ncertain p*r*ocessing tim*e*s in D2D edge networks. *Mature* is aimed at maximizing task completion rates within deadlines. It has two salient features. First, the TPT uncertainty is characterized by a probabilistic model based on real-world measurements. Second, we propose a game-based probabilistic offloading approach and incorporate the probabilistic TPT model into the approach. As a result, the offloading decision can effectively accommodate probabilistic TPT variations. We implement *Mature* and conduct extensive experiments using five datasets and four applications. The results show that the proposed fine-grained TPT distributions can accurately characterize the completion time, and the offloading algorithm can effectively improve task completion rates by 23%. It is worth noting that this improvement can reach 36% when the devices are resource-limited. Overall, the contributions of this paper can be summarized as follows:

1. **Innovative Probabilistic Model for TPT Uncertainty:** We develop a probabilistic model based on real-world data to effectively characterize the uncertainties in task processing times (TPTs). This model enhances the predictability and reliability of task offloading decisions within device-to-device (D2D) edge networks.
2. **Game-Based Offloading Strategy:** We introduce a novel game-based offloading strategy that seamlessly integrates our probabilistic TPT model. This approach optimizes task offloading decisions, allowing for dynamic adaptation to the inherent variability in TPTs and enhancing task completion rates under various network conditions and device capabilities.
3. **Validation Through Comprehensive Experiments:** The effectiveness of our model and offloading strategy is demonstrated through extensive experimental evaluations involving five datasets and four applications. The results confirm that our approach

significantly improves the accuracy of completion time predictions and increases task completion rates, particularly in resource-constrained environments.

The rest of this paper is organized as follows. Section 2 presents the related works and the motivation for this work. Section 3 introduces the measurement-based probabilistic model for TPTs and task completion rates. Section 4 proposes the main design of the game-based probabilistic task offloading algorithm with uncertain processing times. Section 5 presents the evaluation results.

2. Related Works and a Motivating Example

In this section, we summarize the existing works on task offloading in D2D edge computing. We use an example to show the limitations of existing works and explore the potential benefits of *probabilistic* task processing times.

2.1. Related Works

Recent research has highlighted edge computing as a pivotal emerging technology for 5G networks, as acknowledged by the European 5G PPP (5G Infrastructure Public-Private Partnership) research body. Extensive research has been conducted on the computation offloading problem, resulting in a plethora of offloading policies. These policies can be broadly categorized into two distinct stages.

In the first stage of investigation into computation offloading, researchers typically treat applications as indivisible tasks that are wholly transferred to edge servers. The authors of [16], examined the multi-user offloading challenge in an edge computing environment characterized by multi-channel wireless interference. They established the problem's NP-hard nature concerning achieving the optimal solution. Subsequently, a heuristic method was utilized to enable efficient computation offloading across a distributed framework. The efficacy of the approach was demonstrated through simulations, which indicated that the algorithm not only enhances offloading performance but also maintains scalability with an increase in the number of users. Moreover, in [17], the focus shifted to a distributed offloading strategy designed for multi-user and multi-server settings, employing orthogonal frequency-division multiple access within small-cell networks. The authors formulated a distributed overhead minimization challenge and applied potential game theory to demonstrate that the decision-making process aligns with a potential game scenario. The simulation results further affirmed that the newly proposed algorithm can significantly conserve overhead compared to other existing computation offloading algorithms. Other inquiries documented in [18–20] addressed similar multi-user and multi-server environments. These studies introduced heuristic decision-making algorithms for offloading, which strategically assigned different tasks to optimal channels and servers with the objective of reducing average completion times. These collective efforts underscore a critical engagement with the complexities of task offloading in contemporary edge computing scenarios.

As applications grow increasingly complex, the field of computation offloading is advancing into the second stage, characterized by the use of Directed Acyclic Graphs (DAGs) to pinpoint precise offloading opportunities. In this stage, as detailed in [21], applications are broken down into sequential modules. The decision-making process involves selecting specific modules for offloading and determining their execution context—either in cloud or edge computing environments. This is facilitated by the Iterative Heuristic MEC Resource Allocation (IHRA) algorithm, which dynamically adjusts offloading decisions based on current conditions. Further exploration of this nuanced approach was presented in [22,23], where the focus shifted to fine-grained task offloading within low-power IoT systems. The authors aimed to strategically map subtasks across diverse servers while arranging their execution order to meet task precedence demands and minimize the overall completion time. The notable distinction in this stage of computation offloading research is the interaction between subtasks—they are not merely sequential but exhibit interdependency. To enhance decision-making efficiency, a distributed computation offloading algorithm,

termed DEFO, has been developed using game theory principles. This algorithm optimizes offloading decisions by considering the competitive and cooperative interactions among the distributed components involved in the computation offloading process.

As the number of smart devices and the demand for real-time applications increase, MEC architectures alone cannot fully meet the stringent latency requirements, especially in scenarios with high device densities and unpredictable dynamic environments. This realization has led to the emergence of D2D computing, which extends the concept of edge computing by enabling direct communication between nearby devices without routing traffic through a base station or central server [5,7,11,24–29].

Although device-to-device (D2D) edge computing can more effectively exploit the computational resources of all devices in specific scenarios, the computational power of these devices is generally inferior to that of traditional edge servers. Consequently, the fluctuation in task processing times is more pronounced, which, in turn, directly influences the strategies for task offloading. The offloading decisions determine when and where to offload the tasks and play a critical role in the performance of D2D edge computing. Existing works on offloading decisions can be classified into two categories according to the different assumptions on TPTs.

Offloading decisions with given task processing times (TPTs). The offloading decision in D2D networks with known TPTs was studied in [6–8,10,11,30]. For example, Gone et al. [10] delved into the integration of D2D edge computing with parallel computing within the framework of distributed edge computing (DEC). By leveraging distributed devices in D2D edge networks, DEC is poised to significantly reduce service delays by enabling parallel processing of computation tasks offloaded from an end device to edge devices connected via a wireless network. The study focused on the crucial challenge of minimizing execution delays for distributed algorithms applied to these tasks. Cao et al. [30] presented a novel approach to enhance energy efficiency in D2D edge computing systems through cooperative computation and communication. They examined a three-node MEC setup involving a user node, a helper node, and an access point (AP) with an MEC server, exploring two offloading models: partial and binary. A four-slot transmission protocol was developed to enable efficient cooperation, optimizing resource allocation to minimize energy consumption while meeting latency constraints. Efficient algorithms tackled non-convex optimization problems, and the numerical results demonstrated substantial improvements in computation capacity and energy efficiency compared to conventional methods. The study set a new standard for integrated resource management in D2D edge network environments. In the aforementioned work, the TPT was assumed to be given in advance and known during the offloading decision. These works may not be feasible in practical scenarios considering that the TPT is often unavailable before the tasks are executed [13].

Offloading decisions with uncertain TPTs. Other works studied the offloading problem with uncertain TPTs [12–14]. Chang et al. [14] addressed the challenge of making offloading decisions in edge computing (EC) for continuous applications with uncertain characteristics such as user behavior and application duration. A novel algorithm, the Response Time Improved Offloading algorithm with Energy Constraint (RTIOEC), was proposed to optimize these decisions under energy consumption constraints. The study focused on minimizing the average response time while maintaining energy efficiency. The evaluation results confirmed that the RTIOEC effectively achieves these goals, demonstrating its practicality for EC environments where application properties are not fully known. In [12], Zhan et al. studied a D2D multimedia data offloading architecture to improve the QoS of multimedia and meet the expected deadline. In these works, the unknown TPT is estimated using the average TPT value. While the average TPT is reasonable in scenarios with stable inputs and system environments, as analyzed in Section 3, the averaged expectation of the TPT cannot reflect the actual task processing in general cases and thus can significantly mislead the offloading decisions.

Unlike the above works, *Mature* utilizes a probabilistic model to characterize the TPT more accurately, thereby enhancing the precision of task processing time estimates. This improvement enables the formulation of more appropriate offloading strategies, subsequently increasing the computational efficiency in D2D network edge computing environments. Moreover, considering the distributed nature of D2D networks, the probabilistic model is then incorporated into the proposed game-based offloading scheme. As a result, the temporal variations of task processing are considered and utilized to achieve better offloading performance in D2D edge computing.

2.2. A Motivating Example

Next, we use an example to illustrate the impact of a *probabilistic* TPT on the offloading process and how it can be exploited to improve offloading performance. We consider a typical scenario where the apps AR [2] and Beautycam continuously invoke the image processing module on a smartphone. The average TPT for an image processing task (from either app) is 1.2 s. As shown in the left part of Figure 1a, the deadlines for AR and Beautycam are 600 ms and 1.3 s, respectively. Evidently, the average processing time (1.2 s) does not meet the deadline requirement of the AR application (0.6 s). In addition, the processing of AR tasks may also incur unexpected delays for Beautycam tasks. As a result, the smartphone will seek to offload the AR tasks to other devices if possible. Although there is a smartphone nearby that can provide extra computing capability, its average TPT for image processing is 1.2 s, which also exceeds the AR deadline. In this case, following the policy of existing works [10,12,30], the AR tasks will be dropped or only processed with the best effort when there are no other tasks.

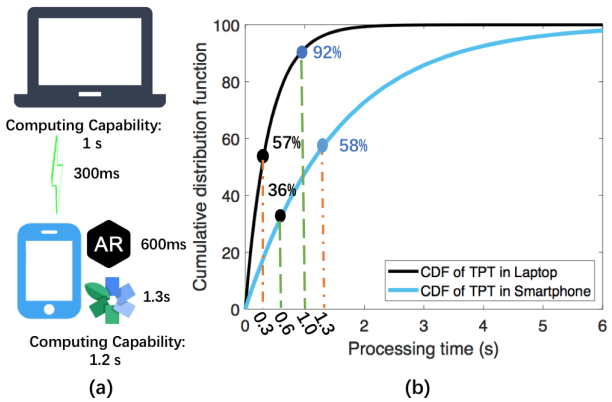

Figure 1. An example of task offloading with uncertain processing times. (**a**) Offloading decision in D2D networks. (**b**) CDF of processing time.

However, as demonstrated in our measurement study in Section 3, a task's processing time usually fluctuates within a certain range (due to variations in input data, environment, task contentions, etc.), and the *average* processing time can be highly misleading in offloading decisions. Figure 1b depicts the cumulative distribution function (CDF) plots of the processing times of the image processing module on the laptop and smartphone, respectively. The black/blue lines denote the TPTs of image processing on the laptop/smartphone. With the CDFs plotted for both devices, we observe the following:

1. Although the average TPT (1.2 s) is smaller than the deadline of Beautycam (1.3 s), as shown in the figure, the actual task completion rate (the probability that the TPT is under 1.3 s) is only 58%.
2. The offloading decisions made by existing works significantly underutilize the devices' resources and yield an unsatisfactory overall completion rate (58%). The reason for this is analyzed below.

We consider three offloading strategies for comparison: (1) Existing work: AR's tasks are dropped, and BC's tasks are executed on the smartphone (as analyzed above).

(2) OffloadAR: AR's tasks are offloaded to the laptop, and BC's tasks are executed on the smartphone. (3) OffloadBC: BC's tasks are offloaded to the laptop, and AR's tasks are executed on the smartphone. The round-trip communication delay for offloading is 300 ms. The deadlines and corresponding cumulative probabilities are denoted using dashed red lines (for OffloadAR) and green lines (for OffloadBC) in Figure 1b. For example, with OffloadAR, BC's tasks (1.3 s) are executed locally on the smartphone (58% completion rate), and AR's tasks (0.3 s) are executed on the laptop (57% completion rate). It is worth noting that the deadline for AR tasks on the laptop becomes $0.6 - 0.3 = 0.3$ s, as the communication delay takes 0.3 s. Table 1 shows the task completion rates for the different offloading strategies. We can see that OffloadBC achieves the highest overall task completion rate of 36% + 92%. We use the sum as the overall task completion rate since the two apps are treated as equally important. If the apps have different priorities, we could add weights to each completion rate and use the weighted sum as the overall completion rate.

Table 1. The completion rate comparison.

	Augmented Reality	**Beautycam**
Existing work	0%	58%
OffloadAR	57%	58%
OffloadBC	**36%**	**92%**

BC: Beautycam; AR: Augmented Reality; The bold numbers represent the task completion rates under the optimal offloading scheme.

From the example, we can infer that (1) the average task processing time cannot reflect the actual offloading performance, and (2) the probabilistic distribution of the TPT can be used to improve the overall task completion rate. However, exploiting TPT distributions to improve offloading performance is a non-trivial task due to the following two challenges. First, we need to obtain the probabilistic TPT models before making offloading decisions. Moreover, considering the multi-task contention and queueing processes, the task completion rate cannot be obtained directly from the TPT model. Second, existing offloading schemes use deterministic models and cannot effectively support the probabilistic TPT model. In the next sections, we describe how we overcome the above challenges and exploit probabilistic TPTs for better offloading performance.

3. Measurement-Based Probabilistic Model for TPTs and Task Completion Rates

Considering that the application types are known, we first use offline measurements and curve fitting to obtain the rough TPT model for an application. Then, for specific tasks and scenarios of the application, we use a small amount of initial data to obtain the parameters for the TPT model.

While the TPT model with measurement-based curve fitting is relatively straightforward, unlike the example in Figure 1, it is a highly challenging task to further infer the task completion rate using the TPT model and task deadlines. The reason is that besides the TPT, the multi-task contention and scheduling processes also affect the whole task completion period from input to output. As a result, there exists a gap between the TPT and the task completion rate of a task.

In this section, we first use the image processing task as a case study to explain how the probabilistic model is obtained. After that, we derive a model for the task completion rate, which incorporates both the TPT model and the queueing process of multiple tasks.

3.1. Measurement-Based Model for TPTs

Now, we use the image processing task as a case study to explain the TPT modeling process. Image processing is one of the most popular services in mobile offloading [31]. Specifically, we investigate the image processing service of visual relationship detection (VRD), which has been widely used in a variety of image understanding tasks, e.g., object

categorization [32], object detection [33], image segmentation [34], and human–object interactions [15]. VRD involves detecting and localizing objects, as well as the relationship between these detected objects. For the example in Figure 2, the VRD output could be {*mouse-eat-biscuit*} or {*mouse-in-cup*}.

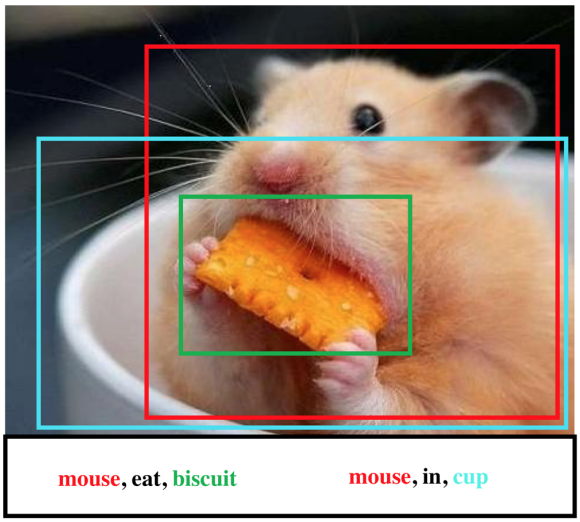

Figure 2. Visual relationship detection.

To obtain the TPT model for the VRD application (the MF-URLN algorithm—multi-modal feature-based undetermined relationship learning network [15]), we first conduct offline measurements using existing VRD datasets with 5000 images and 37,993 relationships. The measured probability distribution function (PDF) of the TPT for VRD is shown in Figure 3. Each bar represents the proportion of images for which the measured TPT is within the corresponding time intervals. Then, we employ curve fitting to determine the appropriate model formulation for the TPT model. The red line denotes the fitted curve of the PDF, which follows an exponential distribution with $\mu = 0.588$.

$$f(t) = \mu Exp(-\mu t). \tag{1}$$

Although the offline measurements and datasets do not exactly match the specific task scenarios, we could use the obtained exponential distribution as an initial TPT model for the VRD service. When encountering analogous tasks in diverse scenarios, we can expediently calibrate the critical parameters of the distribution using minimal input data. Comprehensive experimental validation of this method is detailed in Section 5.1.

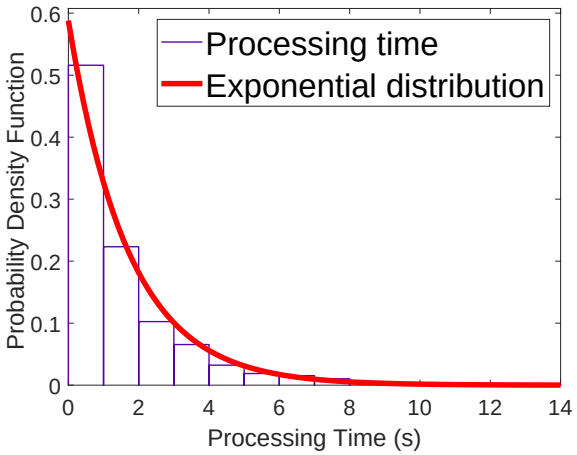

Figure 3. PDF of processing times.

3.2. Model for Task Completion Rates

After characterizing the uncertainty of the TPT using a probabilistic model, in this section, we derive the task completion rate model based on the TPT.

In D2D edge networks, mobile devices (MDs) will generate new tasks at regular intervals (e.g., AR devices continuously collect and process images to cope with the change in the user's view). Usually, we assume that inter-arrival times are independent and have a common distribution. In many practical situations, tasks arrive according to a Poisson stream. Therefore, we hypothesize that task processing follows a basic queueing model, denoted as M/G/1 (queueing theory is widely used in literature to model the service procedure [35]), with exponential inter-arrival time and arbitrary service time. In this basic model, tasks arrive one by one, and they are always allowed to enter the system. There is always room, there are no priority rules, and tasks are served in the order of arrival. In queueing theory, the sojourn time is the waiting time plus the processing time. In D2D edge computing, the completion time equals the sojourn time plus the transmission time if the task is offloaded to other devices. The distribution of the completion time in VRD obeys

$$F(t - t_r) = 1 - Exp[-(\mu_j - \lambda_k)(t - t_r)], \tag{2}$$

where the parameter of processing time μ has been obtained by the TPT model in Equation (1) and the parameter λ_k denotes the task arrival rate of a certain type of task, k. If the task is executed locally, $t_r = 0$. Figure 4 plots the cumulative density function (CDF) of completion times with one type of task under different given deadlines, where we assume its λ_k is 0.15. The red point depicts that 89% of the tasks can be finished before a given deadline (4 s). If the task is offloaded to other devices with the same computing capability, considering the transmission delay ($t_r = 2.1$ s), the tasks should be completed before 1.9 s, and the completion rate will be reduced to 60%. Therefore, by investigating the CDF curves of completion times and transmission times, we can see whether the device makes the appropriate offloading decision (e.g., to execute locally or offload to other devices). How to make an optimal offloading decision in D2D networks is discussed in Section 4. Subsequently, in Section 5.1, we further evaluate the fitting accuracy of the task completion rate model using real data.

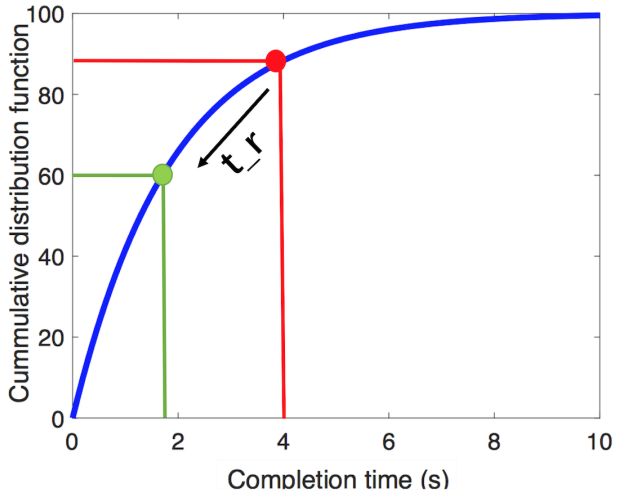

Figure 4. CDF of completion times.

Queueing theory is also suitable for modeling the completion rate with various TPT models, the theoretical completion time follows the M/G/1 model, which can be formulated as

$$F(t) = L^{-1}\{\mathcal{S}(\alpha)\} = L^{-1}\left\{\frac{(1 - \lambda/\mu)\mathcal{B}(\alpha)\alpha}{\lambda\mathcal{B}(\alpha) + \alpha - \lambda}\right\}, \tag{3}$$

where $\mathcal{B}(\alpha)$ can be obtained by

$$\mathcal{B}(\alpha) = \int_0^\infty e^{-\alpha t} \, d\, f(t) \tag{4}$$

where $f(t)$ denotes the TPT model, which is capable of accommodating any probabilistic distribution model. In Section 5.1, we empirically validate this hypothesis through experiments involving typical applications.

4. Game-Based Probabilistic Task Offloading with Uncertain Processing Times

In this section, we incorporate the TPT model and completion model into task offloading to achieve better offloading performance with uncertain processing times in D2D networks. We first describe the system model and discuss how the TPT model is incorporated into the offloading process. Then, we formulate the task completion rate maximization problem as a strategy game. After that, we resort to the potential game to prove the existence of the Nash equilibrium of our strategy game (multi-device task offloading game). Based on the characteristics of the potential game, we design a game-based distributed task offloading decision with uncertain processing times (*Mature*) in D2D edge networks and then analyze its convergence. For ease of understanding, the notations utilized in this paper are summarized in Table 2.

Table 2. Notations used in this paper.

Notation	Description
$\mathcal{N}$	The set of MDs
$\mathcal{K}$	The arrival rate of tasks
c_i	The computation capability of device i
δ_i	Energy consumption per CPU cycle of the device
τ_k	The delay constraints of device i
λ^k	The parameter of the Poisson distribution
λ_j	The arrival rate of workloads on device j
$t_{i,j}$	The transmission time from device i to device j
$F_{i,j}^k$	The completion rate of tasks
$f(t)$	The probability distribution of all tasks
$\mathbf{a}$	An appropriate task offloading decision
$a_{i,k}^k$	Whether task k of device i is offloaded to device j
a_{-i}	Task offloading decision of all devices except for device i
$\mathcal{A}_i$	The offloading decisions of device i
$D_{i,j}^k$	The completion rate decline of tasks

4.1. System Model

In this subsection, the system model adopted in this work is described. We consider a set of $\mathcal{N} = [1, 2, ..i, j, ..N]$ mobile devices (MDs). Each MD has several tasks to be completed with the help of other peer devices, and the computation capability of devices i can be represented by c_i. In the D2D networks, there are K types of computationally intensive tasks permitted to be offloaded to other devices with delay constraints τ_k. As stated earlier in Section 3, the arrival rate of task $\mathcal{K} = [1, 2, ..k]$ follows a Poisson distribution with parameter λ^k, and the processing times required by this task follow a certain probability distribution. Let $a_{i,j}^k$ be an indicator to denote whether task k of device i is offloaded to device j:

$$a_{i,j}^k = \begin{cases} 1 & \text{if task } k \text{ of device } i \text{ is offloaded to device } j \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Significantly, we adopt $a_i = \{a_i^1, a_i^2, a_i^3, ...a_i^k\}$ to represent a possible offloading decision for each task of device i, where $a_i^k = \{0\} \cup \mathcal{N}$ denotes the offloading decision for task k of device i (e.g., $a_i^k = i$ means that device i chooses to execute task k locally, and $a_i^k = 0$ denotes that device i has no demand for task k). According to our measurement study of VRD in Section 3, the completion rate for task k of device i can be obtained as

$$F_{i,j}^k(t_{i,j}^k) = 1 - Exp[-(\mu_k/c_j - (\lambda_j + \lambda^k))t_{i,j}^k] \tag{6}$$

$$t_{i,j}^k = \tau_k - t_{i,j} \tag{7}$$

where $t_{i,j}$ denotes the transmission time from device i to device j ($t_{i,i} = 0$). The transmission delay is fundamentally determined by the size of the task, denoted as S, and the bandwidth of the communication network, represented as R. This delay, calculated by the formula L/R, encapsulates the duration required for the entire task to be transmitted over the D2D edge network. λ_j denotes the arrival rate of the existing workload on device j. From this equation, we can observe that with an increasing number of tasks, the completion rate of each task will decrease exponentially. It should be stressed here that Equation (6) is only suitable for the processing time of the task, which follows an exponential distribution. A more general expression can be described as $F_{i,j}^k(\lambda_j, \lambda^k, f(t), c_j, t_{i,j}^k)$, where ($f(t)$ denotes the probability distribution of the TPTs of all tasks. More details can be seen in Equation (3)). Our goal is to find an appropriate task offloading decision $\mathbf{a} = (a_1, a_2, ...a_N)$ for all tasks from different devices to maximize the completion rates of all tasks. The task offloading problem can be formulated as follows:

$$\textbf{Opt} : \max_{\mathbf{a}} \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{K} a_{i,j}^k F_{i,j}^k(\lambda_j, \lambda^k, f(t), c_j, t_{i,j}^k) \ \forall i, j \in \mathcal{N} \tag{8}$$

$$s.t \quad a_i \in \{0, 1, 2.., N\} \quad \forall i \in \mathcal{N} \tag{9}$$

$$\sum_{i=1}^{N} \sum_{k=1}^{K} a_{i,j}^k \delta_j / \mu_k < B_j \quad \forall i \in \mathcal{N}, \tag{10}$$

where Equation (10) represents the capacity constraint of each device; δ_i denotes the energy consumption per CPU cycle in user device i, which can be obtained using the method in [36]; and $1/\mu_k$ denotes the average CPU cycle required to accomplish task k. We can easily observe that finding the optimal task offloading decision is impractical (a special case of our problem can be reduced to 0–1 knapsack problem), which inspires us to design a distributed computation offloading scheme.

4.2. Potential Game Formulation

In this section, we adopt a game-theoretic approach to coordinate competition among multiple users. In a game theory-based task offloading decisions among multiple users, at each step, a rational user reacts to other users' actions in the previous step and makes an optimal local decision. After a finite number of steps, all users can self-organize into a mutual equilibrium state: the Nash equilibrium. Let $a_{-i} = \{a_1, ..., a_{i-1}, a_{i+1}, ..a_n\}$ denote the task offloading decision for all devices except for device i. By analyzing the strategies of other devices, device i aims to choose an optimal local offloading decision for all its tasks, maximizing the overall completion rate:

$$\max_{a_i^k = 0,1,...,N} \mathcal{P}(a_i, a_{-i}) \quad \forall i \in \mathcal{N} \tag{11}$$

$$\mathcal{P}_i = \sum_{j=1}^{N} \sum_{k=1}^{K} a_{i,j}^k F_{i,j}^k \quad \forall j \in \mathcal{N} \quad \forall k \in \mathcal{K} \tag{12}$$

Now we can formulate our task offloading problem as a strategic game, which can be described by $\Gamma = (\mathcal{N}, \{\mathcal{A}_i\}_{i\in\mathcal{N}}, \{\mathcal{P}_i\}_{i\in\mathcal{N}})$, where we regard each device as one player, so $\mathcal{N}$ indicates the set of all players. $\mathcal{A}_i$ denotes the offloading decisions of device i. It is worth noting that if a strategy $a' = (a'_1, a'_2, ..., a'_n)$ is a Nash equilibrium, no user can further increase its completion rate by changing its strategy, i.e,

$$\mathcal{P}(a'_i, a'_{-i}) \geq \mathcal{P}(a_i, a'_{-i}) \quad \forall i \in \mathcal{N} \quad \forall a_i \in \mathcal{A}_i \tag{13}$$

Our aim is to explore the strategy $a' = (a'_1, a'_2, ..., a'_n)$ after finite iterations. First, we need to study the existence of the Nash equilibrium by proving that this strategy game is a potential game.

Theorem 1. *A strategy game is defined as a potential game if there is a potential function $\phi(\mathbf{a})$ such that for every device i, $a'_i, a_i \in \mathcal{A}_i$. If*

$$\mathcal{P}(a'_i, a_{-i}) \geq \mathcal{P}(a_i, a_{-i}) \tag{14}$$

we have

$$\phi(a'_i, a_{-i}) \geq \phi(a_i, a_{-i}) \tag{15}$$

The obvious property of a potential game is that it can always achieve a Nash equilibrium if only one device updates its strategy at any given time. Thus, we construct such a potential function for our task offloading game to prove that it has at least one convergent decision. The salient property of potential games, where they are guaranteed to reach a Nash equilibrium when strategies are updated sequentially by individual players, underscores their utility in modeling strategic interactions. Given this attribute, constructing a potential function for our task offloading scenario is both strategic and beneficial. Consider a scenario where devices decide whether to offload computational tasks to other devices or handle them locally. The potential function is designed to increase when a device efficiently offloads a task, enhancing system performance, and decrease when a device inefficiently opts for local processing, reflecting suboptimal decisions. By devising such a potential function, we establish a framework where each device's decision to offload tasks can be analyzed as part of a holistic system that inherently moves toward equilibrium. This methodological approach not only enhances the predictability and stability of the decision-making process but also ensures that the system admits at least one convergent decision, aligning individual incentives with the collective optimal. Thus, integrating a potential function into our game-theoretic model substantiates its efficacy by proving the existence of at least one stable strategy configuration. In our work, the potential function can be defined as follows:

$$\begin{aligned} \phi(a) = &-1/2 \sum_{i=1}^{N} \sum_{l\neq i}^{N} \sum_{q_1=1}^{k} \sum_{q_2=1}^{k} D_{ij}^{q_1} D_{lj}^{q_2} I\{a_i^{q_1} = j, i \neq j\} \\ &- 1/2 \sum_{i=1}^{N} \sum_{l\neq i}^{N} \sum_{q_1=1}^{k} \sum_{q_2=1}^{k} D_{ii}^{q_1} D_{lj}^{q_2} I\{a_i^{q_1} = i\}, \quad \forall j \in \mathcal{N} \end{aligned} \tag{16}$$

where q_1 and q_2 denote the tasks belonging to device i and device l ($l \neq i$). The indicator function $I\{a_i^{q_1} = j\} = 1$ means that task q_1 of device i will be offloaded to device j; otherwise, $I\{a_i^{q_1} = j\} = 0$. $D_{i,j}^k$ shows the completion rate decline of tasks that belong to device i if task k of device i offloads to device j. In this strategy game, each device only considers the completion rate of its own tasks and is only responsible for updating the offloading decision for tasks that are generated by it (e.g., task k of device j is offloaded to

device i; device i has no right to change the offloading decision for this task, even if this task is executed on it). $D_{i,j}^k$ can be described as

$$\begin{aligned} D_{i,j}^k = \sum_{q=1}^{k} a_{i,j}^q [F_{i,j}^q(\lambda_j, f(t), c_j, t_{i,j}^k) \\ - F_{i,j}^q(\lambda_j, \lambda^k, f'(t), c_j, t_{i,j}^k)] \end{aligned} \tag{17}$$

Equation (17) shows the completion rate decline of all tasks q (the tasks are generated by device i) that are executed on device j if task k of device i is offloaded to device j. In the offloading strategy game, device i will choose the peer device with a minimum $D_{i,j}^k$ to offload its task k.

Theorem 2. *The task offloading decision with uncertain processing times is a potential game with the potential function, as given in Equation (16), and thus it always can reach a Nash equilibrium.*

Proof. Suppose that an MD wants to update the offloading decision a_i to $a_i{}'$, as this update will lead to an increase in the completion rate of its tasks, i.e., $\mathcal{P}(a_i, a_{-i}) < \mathcal{P}(a_i', a_{-i})$. According to the definition of the potential game, our aim is to prove that this update will also lead to a decrease in the potential function, i,e., $\phi(a_i, a_{-i}) < \phi(a_i', a_{-i})$. Based on the different offloading strategies, we aim to prove that the strategy game is a potential game using the following three cases: (1) $a_i^k = i$, $(a_i^k)' = j$; (2) $a_i^k = j$, $(a_i^k)' = i$; and (3) $a_i^k = j_1$, $(a_i^k)' = j_2$.

(1) Case 1: $a_i^k = i$, $(a_i^k)' = j$.

As device i selects device j to offload its task k rather than computing it locally, we know that $\mathcal{P}(a_i, a_{-i}) < \mathcal{P}(a_i', a_{-i})$, which can further derive that the completion rate decline is smaller when task k is offloaded to device j, $D_{ii}^k > D_{ij}^k$.

$$\begin{aligned} &\phi(a_i{}', a_{-i}) - \phi(a_i, a_{-i}) \\ &= D_{ii}^k \sum_{l \neq i}^{N} \sum_{q=1}^{k} D_{lj}^q - D_{ij}^k \sum_{l \neq i}^{N} \sum_{q=1}^{k} D_{lj}^q > 0 \end{aligned} \tag{18}$$

(2) Case 2: $a_i^k = j$, $(a_i^k)' = i$. Using a similar argument to case (1), we can conclude that $\phi(a_i{}', a_{-i}) - \phi(a_i, a_{-i}) > 0$.

(3) Case 3: $a_i^k = j_1$, $(a_i^k)' = j_2$.

Since $a_i^k = j_1$, $(a_i^k)' = j_2$, we know that $\mathcal{P}(a_i, a_{-i}) < \mathcal{P}(a_i', a_{-i})$, which further implies that $D_{ij_1}^k > D_{ij_2}^k$ and $\sum_{l \neq i}^{N} \sum_{q \neq k}^{k} D_{lj_1}^q > \sum_{l \neq i}^{N} \sum_{q \neq k}^{k} D_{lj_2}^q$

$$\begin{aligned} &\phi(a_i{}', a_{-i}) - \phi(a_i, a_{-i}) \\ &= 1/2 D_{ij_1}^k \sum_{l \neq i}^{N} \sum_{q \neq k}^{k} D_{lj_1}^q I\{a_i^k = j, i \neq j\} \\ &+ 1/2 \sum_{l \neq i}^{N} \sum_{q \neq k}^{k} D_{lj_1}^q D_{ij_1}^k I\{a_i^k = j, i \neq j\} \\ &- 1/2 D_{ij_2}^k \sum_{l \neq i}^{N} \sum_{q \neq k}^{k} D_{lj_2}^q I\{a_i^k = j, i \neq j\} \\ &- 1/2 \sum_{l \neq i}^{N} \sum_{q \neq k}^{k} D_{lj_2}^q D_{ij_2}^k I\{a_i^k = j, i \neq j\} \\ &= D_{ij_1}^k \sum_{l \neq i}^{N} \sum_{q \neq k}^{k} D_{lj_1}^q - D_{ij_2}^k \sum_{l \neq i}^{N} \sum_{q \neq k}^{k} D_{lj_2}^q > 0 \end{aligned} \tag{19}$$

□

From Theorem 2, we have proven that our task offloading strategy can be formulated as a potential game by constructing a potential function in Equation (16). This implies that the completion rates of all tasks from different devices are guaranteed to reach a Nash equilibrium within a finite number of iterations. The *Mature* algorithm is elaborated in Algorithm 1.

Algorithm 1 The *Mature* Algorithm

1. **Input**: $\mathcal{N}$, $\mathcal{K}$, c_i, λ^k,$f_k(t)$,$t^k_{i,j}$
2. **Output**: $a^k_{i,j}$ and average completion rate.
3. **Initialize**: a_i = {0, 0, 0,...0}, all tasks execute locally,
study the PDF of TPT, broadcast λ^k and the c_i
4. **for** each decision slot t do
5. **for all** $i \in$ **N** do
6. **for all** $k \in$ **K** do
7. compute $\mathcal{P}_i = \sum_{j=1}^{N} \sum_{k=1}^{K} a^k_{i,j} F^k_{i,j}$, base on other
users' decisions in previous step, i.e., a_{-i}
8. **end for**
9. **end for**
10. Find only one MD i to update its offloading decision, which could achieve the maximum completion rate increment
11. **if** Gain = 1 **or** $\Delta\mathcal{P}_i \leq \delta$ **then**
12. **return a**
13. **end if**
14. update a_i and the workload on device i and peer devices
16. The updated devices will broadcast its existing workload
17. **end for**

4.3. The Game-Based Distributed Task Offloading Decision

In the initialization step of *Mature* (line 3), each MD executes its tasks locally to study the probability distribution of each task's processing time. Then, the MDs broadcast information about the workload (probability distribution of TPT) and their computation capability to other MDs in D2D networks. This broadcasting not only happens at initialization but also periodically, which ensures that any new device joining the network can receive up-to-date information about the network's state and seamlessly integrate into the decision-making process. During every iteration, each MD computes the $\mathcal{P}_i = \sum_{j=1}^{N} \sum_{k=1}^{K} a^k_{i,j} F^k_{i,j}$ based on the offloading decisions of other MDs in the previous step. By utilizing the coordination and agreement algorithms in [37], the devices in D2D networks vote on only one MD with the greatest gain and update its offloading decision. In our proposed algorithm, each device iteratively updates its offloading decisions at every decision slot based on the most current available information. This iterative decision-making process is pivotal for the system's adaptability to dynamic network environments. Specifically, it enables the system to swiftly respond to real-time changes, such as devices joining or exiting the network. The flexibility of this approach ensures that the decisions made in each slot accurately reflect the existing network configuration and conditions, thereby continually optimizing the overall system performance. After that, the corresponding devices broadcast their updated workloads to other devices. This mechanism ensures that when a device exits the network, its tasks are redistributed among the remaining devices during the next decision slot, maintaining balance and optimizing network performance. Moreover, devices that update their workloads broadcast this information, guaranteeing that all devices in the network are consistently informed about the current workload distribution. This is crucial for making informed decisions in subsequent slots, particularly under changing network conditions, and ensures the network's adaptability to dynamically changing scenarios.

After a finite number of iterations, the system finally achieves the Nash Equilibrium in lines 11–12. In such a state, no user can further increase the completion rate (or the growth

is less than the threshold) by changing its strategy unilaterally. The threshold is introduced to improve the performance of convergence, which makes the algorithm more suitable for the dynamic scenario in the D2D edge network. Next, we analyze the convergence time of the *Mature* algorithm.

Theorem 3. *The game-based probabilistic task offloading decision will terminate within most* $NKD_{max}{}^2/D_{min}\Delta D$ *decision slots.*

Proof. First of all, we assume that

$$D_{max} = \max_{i,j\in N,k\in K}\{D_{i,j}^k\} \quad D_{min} = \min_{i,j\in N,k\in K}\{D_{i,j}^k\} \tag{20}$$

$$\Delta D = min\{D_{i,j}^k - D_{i,j'}^k\} \tag{21}$$

(1) According to the potential function in Equation (16),

$$\begin{aligned}\phi(a) =& -1/2\sum_{i=1}^{N}\sum_{l\neq i}^{N}\sum_{q_1=1}^{k}\sum_{q_2=1}^{k} D_{ij}^{q_1}D_{lj}^{q_2}\\ &-1/2\sum_{i=1}^{N}\sum_{l\neq i}^{N}\sum_{q_1=1}^{k}\sum_{q_2=1}^{k} D_{ii}^{q_1}D_{lj}^{q_2}\\ &\geq -\sum_{i=1}^{N}\sum_{l\neq i}^{N}\sum_{q_1=1}^{k}\sum_{q_2=1}^{k} D_{max}\\ &= -N^2K^2D_{max}{}^2,\end{aligned} \tag{22}$$

and we can derive that $-N^2K^2D_{max}{}^2 \leq \phi(a) \leq 0$.

(2) In each iteration, if device i updates its offloading strategy from a_i to $a_i{}'$, the completion rate will be increased. As we have proven, the strategy game is a potential game:

$$\prec(a_i', a_{-i}) \geq \prec(a_i, a_{-i}) \tag{23}$$

For case (1), $a_i^k = i$, $a_i^{k'} = j$,

$$\begin{aligned}&\phi(a_i{}', a_{-i}) - \phi(a_i, a_{-i})\\ &= D_{ii}^k\sum_{l\neq i}^{N}\sum_{q=1}^{k} D_{lj}^q - D_{ij}^k\sum_{l\neq i}^{N}\sum_{q=1}^{k} D_{lj}^q\\ &= (D_{ii}^k - D_{ij}^k)\sum_{l\neq i}^{N}\sum_{q=1}^{k} D_{lj}^q\end{aligned} \tag{24}$$

as $\sum_{l\neq i}^{N}\sum_{q=1}^{k} D_{lj}^q \geq NKD_{min}$, $D_{ii}^k > D_{ij}^k$:

$$\phi(a_i{}', a_{-i}) - \phi(a_i, a_{-i}) \geq NKD_{min}\Delta D \tag{25}$$

For case (2), $a_i^k = i$, $a_i^{k'} = j$, using a similar argument to case (1), we can conclude that $\phi(a_i{}', a_{-i}) - \phi(a_i, a_{-i}) \geq NKD_{min}\Delta D$.

For case (3), $a_i^k = j_1$, $a_i^{k\prime} = j_2$, as $\sum_{l\neq i}^{N}\sum_{q\neq k}^{k} D_{lj_1}^q > \sum_{l\neq i}^{N}\sum_{q\neq k}^{k} D_{lj_2}^q$,

$$\begin{aligned}
&\phi(a_i{}', a_{-i}) - \phi(a_i, a_{-i}) \\
&= D_{ij_1}^k \sum_{l\neq i}^{N}\sum_{q\neq k}^{k} D_{lj_1}^q - D_{ij_2}^k \sum_{l\neq i}^{N}\sum_{q\neq k}^{k} D_{lj_2}^q \\
&\geq D_{ij_1}^k \sum_{l\neq i}^{N}\sum_{q\neq k}^{k} D_{lj_1}^q - D_{ij_2}^k \sum_{l\neq i}^{N}\sum_{q\neq k}^{k} D_{lj_1}^q \\
&\geq NKD_{min}\Delta D.
\end{aligned} \tag{26}$$

From Equations (25) and (26), we know that the minimum increment of ϕ is $NKD_{min}\Delta D$. According to Equation (22), we can see that the algorithm will terminate within at most ${NKD_{max}}^2/D_{min}\Delta D$ decision slots. □

Although the Mature algorithm may achieve a higher convergence time in some conditions, it has much higher computation efficiency and provides a workable solution in ultra-dense networks. It should be noted that (1) we can further reduce the convergence time by adjusting the threshold δ in line 11, and (2) the convergence time only indicates the duration until the game system becomes stable, not the time for task executions. Before convergence, the task may achieve an unappealing completion rate, but it will improve with the increasing number of iterations. In Section 5, we show the convergence of the *Mature* algorithm through simulations.

5. Evaluation

In this section, we first conduct experiments to validate the accuracy of the task completion rate estimation model presented in Section 3.2. We also assess its applicability to a variety of tasks. Subsequently, experiments are performed to evaluate the performance of our proposed offloading algorithm, Mature. Initially, we investigate the convergence properties of Mature. Following this, a comparative analysis is conducted between Mature and state-of-the-art algorithms [12].

5.1. Validation of Task Completion Rate Accuracy

In this section, we evaluate the accuracy and applicability of the task completion rate model. However, prior to this assessment, it is essential to evaluate the estimation accuracy and the time required for the TPT model, as these elements are critical factors affecting task completion rates. To this end, we conduct an evaluation of TPT estimation across various datasets.

5.1.1. Validation of TPT Model

In Section 3.1, offline measurements were conducted using established VRD datasets. The task processing time (TPT) model was developed by training on a comprehensive dataset of 5000 instances. Although these offline measurements introduced a measurable degree of latency, the utility of the model in practical applications is considerable. By integrating minimal data inputs with the model, it is possible to swiftly ascertain the critical parameters of the TPT model, thereby markedly enhancing the efficiency of estimating task completion rates. When our TPT model encounters VRD tasks with new datasets [38], we can use a small portion of the input data to find satisfactory parameters. Parameter training with different amounts of data is shown in Figure 5. We can see that only 30 input images were enough to obtain a TPT model with an error rate of <4%. The error rate only increased by 1.3% compared to fitting the TPT model with the whole dataset. Based on the measurements, the curve fitting-based modeling approach can be applied to a number of different scenarios. We applied our general TPT model (exponential distribution) of VRD to characterize the processing time of VRD in a new dataset (which contains 4000 images) by updating the parameter μ in the exponential distribution.

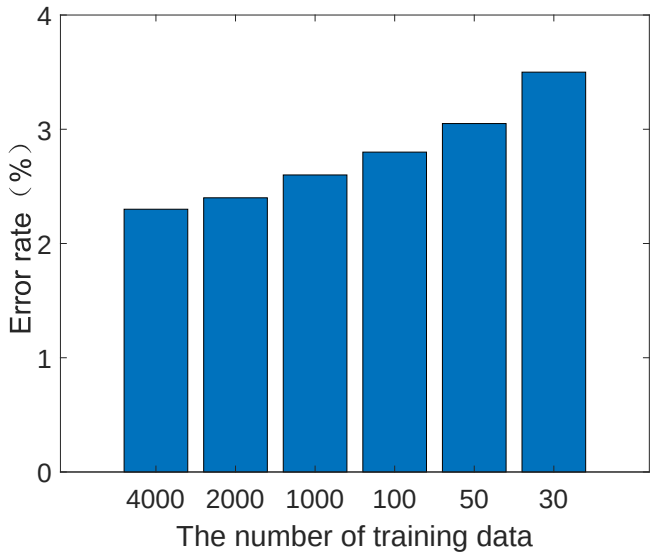

Figure 5. Error rate of fitting.

Next, we study the TPT model for more types of applications—automatic speech recognition [24], forced alignment [25], and part-of-speech tagging [26]—using three real-world datasets [27–29]. As different lengths of speech at a given time may lead to different processing times in automatic speech recognition, Figure 6a plots the PDF of recognition times, which follows a gamma distribution. In Figures 7a and 8a, the general TPT models follow Gaussian and generalized extreme value distributions. Figure 9a represents the CDF of specific applications, where the TPT fluctuates within a relatively narrow range. It can be considered an extension of the exponential distribution. With these general TPT models for various applications, we are able to update the TPT parameters using a small amount of data and further obtain the task completion rates.

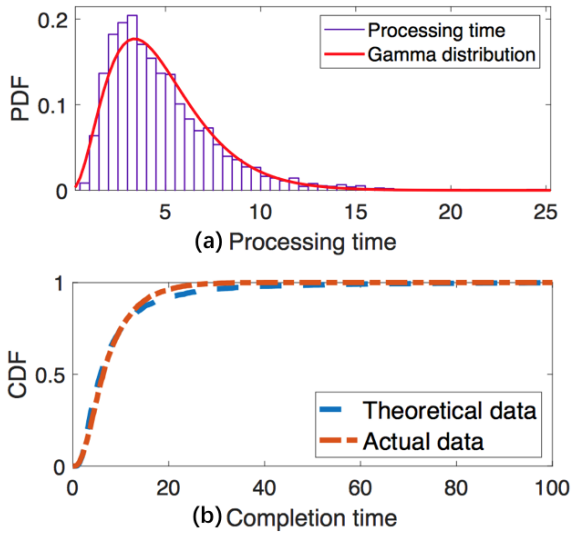

Figure 6. Automatic speech recognition.

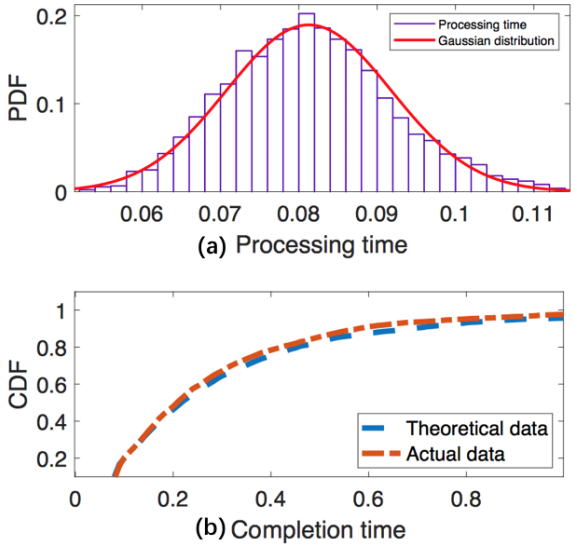

Figure 7. Forced alignment.

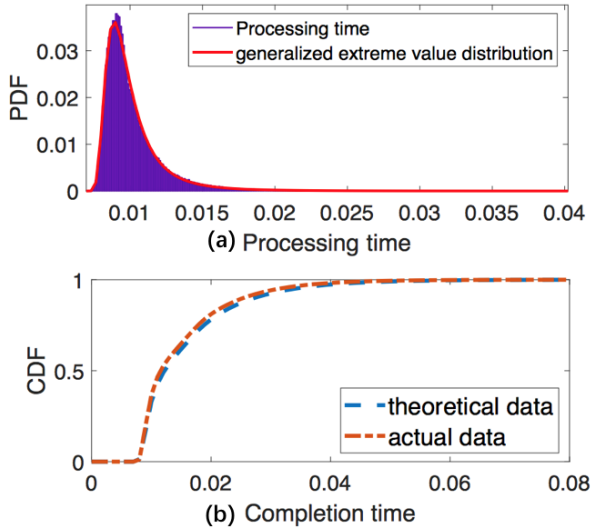

Figure 8. Part-of-speech tagging.

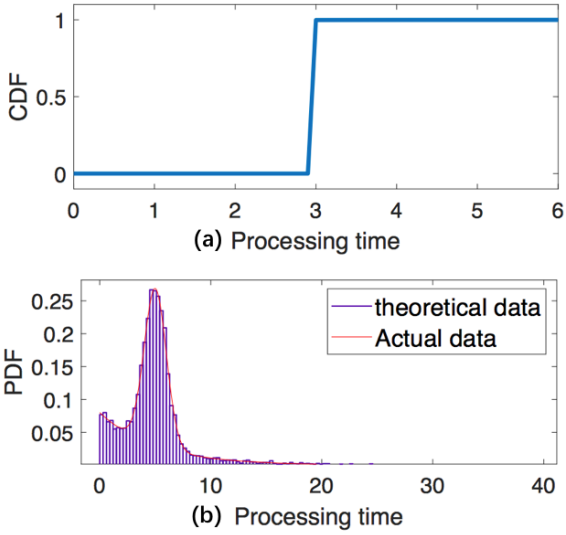

Figure 9. Specific distribution.

5.1.2. Validation of Completion Rate Model

After confirming the high efficiency of the TPT model's fitting process, we proceed to evaluate the task completion rate model proposed in Equation (2) using actual data. Figure 10 shows the CDF of completion times based on the actual completion rate running on the processor (i.e., the red line) and the CDF of predicted completion times based on the distribution fitting result (i.e., the blue line). In this experiment, the images from the visual relationship detection dataset [38] are executed on a processor with an Intel(R) Xeon(R) @1.60HZ, and the images arrive following a Poisson stream ($\lambda = 0.15$). In Figure 10, we can see that (1) the proposed probabilistic model in Equation (1) can accurately characterize the completion time uncertainty in Figure 3, and (2) queueing theory is suitable for modeling the completion rate with an accuracy of 97.7%.

To ensure that the completion rate model is applicable to real networks, we extend our model to various devices. Figure 11 presents the CDF of completion rates on different processors. We first evaluate the completion rate on a processor with an Intel Xeon @ 1.6GHz and obtain the parameter $\mu = 0.588$ in Equation (2), as shown by the solid blue line. Then, the tasks are offloaded to other devices with an Intel Core @3.20GHZ. As the computation capability is doubled, the parameter μ in the complete rate model is increased to 1.176, and the blue dashed line depicts our estimated completion rate on the offloading device. The red dashed line represents the actual data running on the offloading device. By comparing the red line to the blue line, it can be inferred that there is no need to fit the data on the offloading device to obtain the completion rate model. Instead, we can obtain it by updating the parameters in the completion rate model (e.g., Equation (2)) with an accuracy of 98.2%.

To ensure that queueing theory is suitable for modeling the completion rate with various TPT models, as shown in Equation (3), we implement the typical applications (speech recognition, force alignment, and speech tagging) on a running processor with an Intel i7 @1.8GHZ. The completion rate model is represented by the blue lines in Figures 6b–8b, and the orange lines denote the theoretical completion time following the M/G/1 model, which can be formulated as Equation (3) (e.g., Gamma distribution and Gaussian distribution). From these figures, it can be inferred that queueing theory (M/G/1) is suitable for modeling the completion rate with various TPT models, and the error rate is less than 5%. When applications with different TPT models ($f_k(t)$) run on the same device, $f(t)$ is given as:

$$f(t) = \sum_{k=1}^{n} \frac{\lambda_k}{\lambda_1 + \lambda_1 + ..\lambda_n} f_k(t) \tag{27}$$

We also conduct an experiment to validate this assumption, as shown in Figure 9b. The speech recognition application and visual relationship detection are executed on the same processor simultaneously, and the red line depicts the PDF of the processing times. The purple lines denote the results derived from Equations (3) and (27). We can observe that these two lines are perfectly matched, and the assumption is valid.

Generally, the TPTs of different applications may follow different probability distributions and one measurement does not fit all. However, we argue that as long as the application types are known, we could conduct an offline measurement study to obtain the corresponding TPT model. With the model, the parameters for specific task scenarios can be further trained using a small amount of data, which is feasible and scalable for practical scenarios.

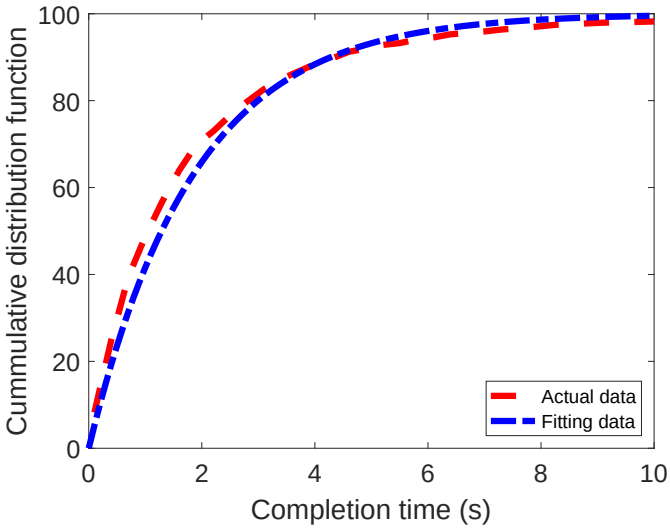

Figure 10. The CDF of completion times.

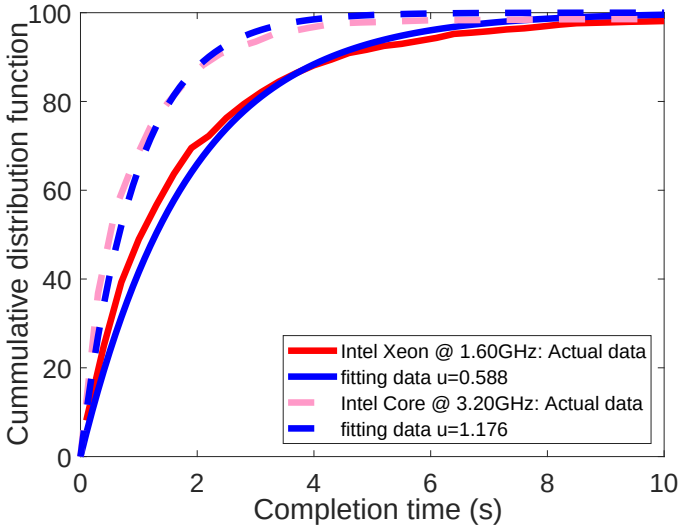

Figure 11. The CDF of completion times.

5.2. Experimental Settings for D2D Edge Network

We consider a D2D edge network, where multiple MDs are randomly distributed in an area. Each MD has several tasks to be executed, belonging to different types of applications. Some MDs are occupied by computationally intensive applications, so it may be desirable for them to offload their tasks to devices with underutilized resources. As there are multiple choices of offloading destinations for each MD to offload their tasks, our goal is to offload the tasks of each MD to the most appropriate peer devices to maximize the overall task completion rate. The overall completion rate is calculated as the sum of the completion rates of all tasks. When there are given priorities for the tasks, we could use a weighted sum as the overall completion rate. The experimental settings are as follows:

- Each MD is equipped with one CPU. The CPU frequency of the MD is in the range of 1.6∼4.8 GHZ. In our experimental setup, a D2D edge network scenario is simulated, where 20 mobile devices (MDs) are randomly distributed. Each node possesses a defined communication range, facilitating direct data exchange among proximate devices.
- The applications executed on the MDs include (1) image recognition: visual relationship detection [15] and object detection [39]; (2) speech recognition: forced alignment [40] and part-of-speech tagging [41]; and (3) randomly generated applications with TPT models following different probability distributions, where the CPU cy-

cles required by each application vary from 20 Megacycles to 100 Megacycles. We adopt open-source codes to implement these applications [42–44] and use the datasets from [38,45–47].

- In the offloading game, MDs continuously select the "best" device to offload their tasks to achieve the maximum completion rate.
- The transmission times of MDs depend on the distance and the number of hops between them.

The parameters for the simulation are shown in Table 3.

Table 3. Simulation parameter settings.

Parameter	Value
The CPU frequency of an MD	1.6∼4.8 GHZ
The number of MDs in the network	20∼25
The offloaded app	image recognition; speech recognition;
The CPU cycle of a random app	20∼100 Megacycles.
The transmission power	100 mWatts
The background noise	−50 dBm
The wireless channel bandwidth	20 MHZ

5.3. The Convergence of Mature

Mature is a game-based algorithm, and its convergence must be guaranteed. Following the analysis in Section 4, at each step, only one MD can update its offloading decision. As a result, the convergence time should increase with more MDs, and the convergence performance will become a bottleneck factor for the system to meet the delay-sensitive requirement. To solve this problem, we propose a complementary policy. With more MDs, during each iteration, we select multiple MDs to update offloading decisions simultaneously, as long as these MDs select different peer MDs. Figure 12 shows the convergence behavior of Mature with different numbers of MDs, which are very similar. The applications are requested by the MDs following a uniform distribution, and the tasks are generated on the MDs following a Poisson distribution. Figure 12a illustrates the average completion rate at a given iteration. Specifically, one hundred experiments are repeated in each iteration number. In this figure, we can observe that Mature reached a Nash equilibrium after a finite number of iterations. We can also observe that the increased number of MDs increased the average completion rate because (1) each user had more opportunities to offload their applications, and (2) the distance between MDs was shorter, so the transmission times decreased.

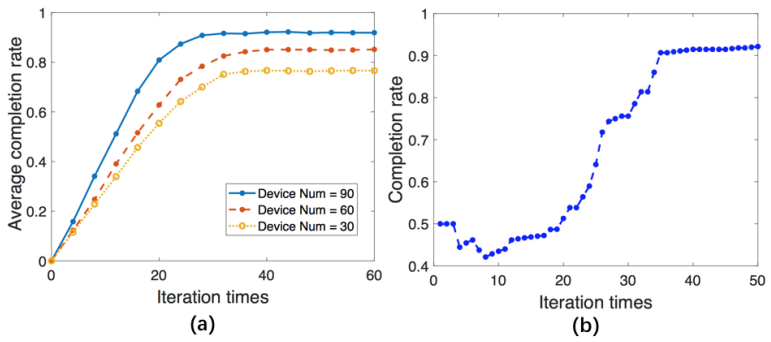

Figure 12. The convergence of Mature. (**a**) The average completion rate of all the devices. (**b**) The completion rate of a single device.

Note that the convergence time only indicates the time when the system became stable, not the time when the task started to be executed. Figure 12b depicts the task completion rates across the iterations in a single experiment. We can observe that before convergence (the 35th iteration), the task achieved an unappealing completion rate. There was even a decline during the 2nd–12th iterations because a large number of requests arrived in the queue before the processor could execute them or offload them. However, the completion rate improved with more iterations.

5.4. Performance of Mature

In this section, we mainly compare Mature with an existing work on D2D edge computing, and the parameter settings are referenced from Table 3. Zhang et al. [12] studied D2D multimedia data offloading architectures to satisfy statistical delay-bounded QoS requirements. Unlike Mature, it uses deterministic models for task offloading, where each application is determined to be offloaded or not based on its performance expectation. However, there is a probability that some applications with poor expectations can also achieve high offloading performance. Such an impact is overlooked by deterministic model-based offloading schemes and may lead to inaccurate offloading decisions, as shown in Figure 13.

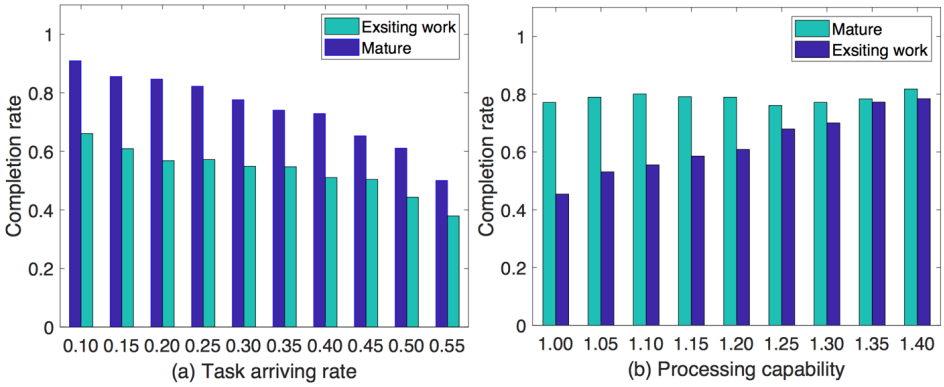

Figure 13. Comparison with an existing work. (**a**) The completion rates for different task arrival rates. (**b**) The completion rate of different processing capability.

Figure 13a shows the completion rates for different task arrival rates. The X-axis denotes the average task arrival rate (the parameter λ in Equation (3)). We can observe that Mature effectively improved the task completion rates by 18%∼23% compared to the existing work. However, as the arrival rate increased, the completion rate of Mature declined more than that of the existing work. The main reason for this is that the offloading scheme in the existing work abandons some applications if they cannot meet the deadline constraint. With the increasing task arrival rate, more and more applications exceeding the deadline are given up. As a result, there are more computation resources available for other applications. Figure 13b shows the completion rates for MDs with different processing capabilities (the parameter μ in Equation (3)). We can observe that Mature always achieved a higher completion rate compared to the existing work. As the processing capability of MDs increased, a large proportion of applications were executed on their local processors without the need for offloading. In conclusion, Mature is more suitable for resource-constrained MDs.

Figure 14 presents the performance of the extended Mature algorithm. Considering the variability of edge network scenarios, it is challenging to pre-train data to obtain the TPT model. To improve the efficiency of data depiction, we utilized only a small amount of data to characterize the TPT model, which resulted in an error rate of 3.5%. Figure 14a shows the difference between the completion rate with the actual TPT model and that with the fitting TPT model (with an error rate of 4%). We can observe that the predicted TPT model, even with certain errors, did not significantly influence the completion rate.

Thus, utilizing a small amount of data to characterize the TPT model can improve the efficiency of D2D edge networks. Figure 14b shows the completion rates of the improved Mature algorithm. If the sojourn time of the task exceeds the delay constraint, it either stops executing in the processor or quits the waiting queue. In this figure, we can observe that the improved Mature always achieved a higher completion rate. However, with this approach, MDs could not process tasks that exceeded the delay limitation.

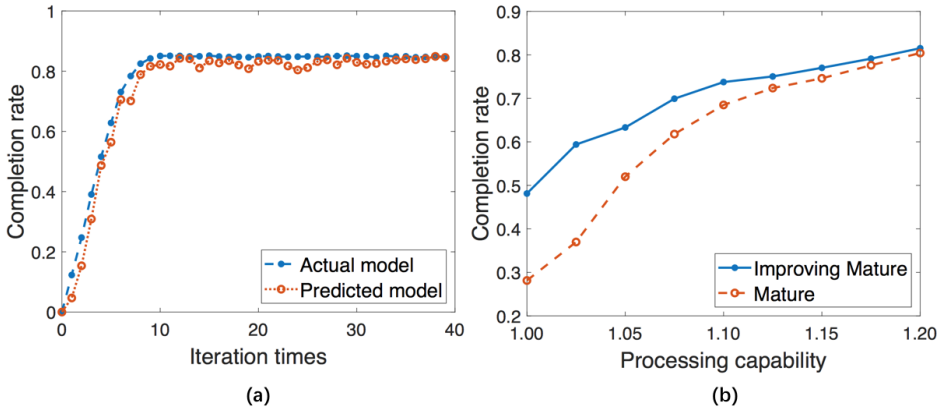

Figure 14. Comparison with improved Mature. (**a**) The completion rate of actual model and predicted model. (**b**) The processing capability of improving Mature and Mature.

6. Conclusions and Future Work

In this paper, we investigate the task offloading problem in D2D edge computing with uncertain task processing times (TPTs). We propose Mature, a game-based task offloading approach. By employing a probabilistic TPT model and a queueing-based estimation scheme for task completion rates, Mature successfully exploits the benefits of probabilistic TPT, as analyzed in Section 1. Moreover, we also analyze the structural properties of the Mature game, prove that it can reach a Nash equilibrium, and propose a distributed offloading decision scheme. We implement and evaluate Mature using real-world applications and datasets. The experimental results show that compared to existing works, Mature accurately characterizes the uncertainty of TPTs/task completion rates, improving the offloading performance by 23% in terms of the overall task completion rate. In future research, we aim to significantly expand the scope of our probabilistic models by integrating a broader spectrum of statistical distributions and employing cutting-edge machine learning algorithms. This augmentation is pivotal as it would substantially enhance the predictive precision of our model regarding task processing times under diverse operational scenarios. Such enhancements are critical for not only capturing the inherent variability in task execution but also for adapting predictive frameworks to real-time changes and uncertainties within networked environments.

Author Contributions: Conceptualization, C.S. and Y.L.; methodology, C.S. and Y.L.; software, F.L.; validation, C.S.; formal analysis, C.S.; data curation, F.L.; writing—original draft preparation, C.S.; writing—review and editing, C.S. and Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Fundamental Research Funds for the Central Universities (No. 24CAFUC04014).

Data Availability Statement: Data is contained within the article.

Acknowledgments: We acknowledge the support from the Fundamental Research Funds for the Central Universities.

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

References

1. Chiang, M.; Tao, Z. Fog and IoT: An Overview of Research Opportunities. *IEEE Internet Things J.* **2017**, *3*, 854–864. [CrossRef]
2. Zhang, W.; Li, S.; Liu, L.; Jia, Z.; Raychaudhuri, D. Hetero-Edge: Orchestration of Real-time Vision Applications on Heterogeneous Edge Clouds. In Proceedings of the IEEE International Conference on Computer Communications (INFOCOM) 2019, Paris, France, 29 April–2 May 2019.
3. Liu, S.; Liu, L.; Tang, J.; Yu, B.; Wang, Y.; Shi, W. Edge computing for autonomous driving: Opportunities and challenges. *Proc. IEEE* **2019**, *107*, 1697–1716. [CrossRef]
4. Wang, N.; Varghese, B.; Matthaiou, M.; Nikolopoulos, D.S. ENORM: A framework for edge node resource management. *IEEE Trans. Serv. Comput.* **2017**, *13*, 1086–1099. [CrossRef]
5. Dai, X.; Xiao, Z.; Jiang, H.; Alazab, M.; Lui, J.C.; Dustdar, S.; Liu, J. Task co-offloading for D2D-assisted mobile edge computing in industrial internet of things. *IEEE Trans. Ind. Informat.* **2022**, *19*, 480–490. [CrossRef]
6. Kim, J.; Kim, T.; Hashemi, M.; Brinton, C.G.; Love, D.J. Joint Optimization of Signal Design and Resource Allocation in Wireless D2D Edge Computing. *arXiv* **2020**, arXiv:2002.11850.
7. He, Y.; Ren, J.; Yu, G.; Cai, Y. D2D communications meet mobile edge computing for enhanced computation capacity in cellular networks. *IEEE Trans. Wirel. Commun.* **2019**, *18*, 1750–1763. [CrossRef]
8. Abbas, N.; Zhang, Y.; Taherkordi, A.; Skeie, T. Mobile edge computing: A survey. *IEEE Internet Things J.* **2017**, *5*, 450–465. [CrossRef]
9. Saeik, F.; Avgeris, M.; Spatharakis, D.; Santi, N.; Dechouniotis, D.; Violos, J.; Leivadeas, A.; Athanasopoulos, N.; Mitton, N.; Papavassiliou, S. Task offloading in Edge and Cloud Computing: A survey on mathematical, artificial intelligence and control theory solutions. *Comput. Netw.* **2021**, *195*, 108177. [CrossRef]
10. Gong, X. Delay-Optimal Distributed Edge Computing in Wireless Edge Networks. *arXiv* **2020**, arXiv:2002.02596.
11. Xing, H.; Liu, L.; Xu, J.; Nallanathan, A. Joint task assignment and resource allocation for D2D-enabled mobile-edge computing. *IEEE Trans. Commun.* **2019**, *67*, 4193–4207. [CrossRef]
12. Zhang, X.; Zhu, Q. D2D offloading for statistical QoS provisionings over 5G multimedia mobile wireless networks. In Proceedings of the IEEE INFOCOM 2019-IEEE Conference on Computer Communications, Paris, France, 29 April–2 May 2019; pp. 82–90.
13. Eshraghi, N.; Liang, B. Joint offloading decision and resource allocation with uncertain task computing requirement. In Proceedings of the IEEE INFOCOM 2019-IEEE Conference on Computer Communications, Paris, France, 29 April–2 May 2019; pp. 1414–1422.
14. Chang, W.; Xiao, Y.; Lou, W.; Shou, G. Offloading Decision in Edge Computing for Continuous Applications under Uncertainty. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 6196–6209. [CrossRef]
15. Zhan, Y.; Yu, J.; Yu, T.; Tao, D. On Exploring Undetermined Relationships for Visual Relationship Detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15 June–20 June 2019; pp. 5128–5137.
16. 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]
17. Yang, L.; Zhang, H.; Li, X.; Ji, H.; Leung, V. A Distributed Computation Offloading Strategy in Small-Cell Networks Integrated With Mobile Edge Computing. *IEEE/ACM Trans. Netw. (TON)* **2018**, *26*, 2762–2773. [CrossRef]
18. Wang, X.; Ye, J.; Lui, J.C. Decentralized task offloading in edge computing: A multi-user multi-armed bandit approach. In Proceedings of the IEEE INFOCOM 2022-IEEE Conference on Computer Communications, Virtual, 2–5 May 2022; pp. 1199–1208.
19. Zhang, Z.; Li, C.; Peng, S.; Pei, X. A new task offloading algorithm in edge computing. *EURASIP J. Wirel. Commun. Netw.* **2021**, *2021*, 17. [CrossRef]
20. Tang, L.; Tang, B.; Zhang, L.; Guo, F.; He, H. Joint optimization of network selection and task offloading for vehicular edge computing. *J. Cloud Comput.* **2021**, *10*, 23. [CrossRef]
21. Ning, Z.; Dong, P.; Kong, X.; Xia, F. A cooperative partial computation offloading scheme for mobile edge computing enabled Internet of Things. *IEEE Internet Things J.* **2018**, *6*, 4804–4814. [CrossRef]
22. Shu, C.; Luo, Y.; Liu, F. Exploiting Duplications for Efficient Task Offloading in Multi-User Edge Computing. *Electronics* **2022**, *11*, 2244. [CrossRef]
23. Shu, C.; Zhao, Z.; Han, Y.; Min, G.; Duan, H. Multi-user offloading for edge computing networks: A dependency-aware and latency-optimal approach. *IEEE Internet Things J.* **2019**, *7*, 1678–1689. [CrossRef]
24. Tam, P.; Math, S.; Kim, S. Optimized multi-service tasks offloading for federated learning in edge virtualization. *IEEE Trans. Netw. Sci. Eng.* **2022**, *9*, 4363–4378. [CrossRef]
25. Zhang, X.; Liu, Y.; Liu, J.; Argyriou, A.; Han, Y. D2D-assisted federated learning in mobile edge computing networks. In Proceedings of the 2021 IEEE Wireless Communications and Networking Conference (WCNC), Nanjing, China, 29 March–1 April 2021; pp. 1–7.
26. Wang, F.; Lau, V.K. Multi-level over-the-air aggregation of mobile edge computing over D2D wireless networks. *IEEE Trans. Wirel. Commun.* **2022**, *21*, 8337–8353. [CrossRef]
27. Long, D.; Wu, Q.; Fan, Q.; Fan, P.; Li, Z.; Fan, J. A power allocation scheme for MIMO-NOMA and D2D vehicular edge computing based on decentralized DRL. *Sensors* **2023**, *23*, 3449. [CrossRef]
28. Jiang, W.; Feng, D.; Sun, Y.; Feng, G.; Wang, Z.; Xia, X.G. Joint computation offloading and resource allocation for D2D-Assisted mobile edge computing. *IEEE Trans. Serv. Comput.* **2022**, *16*, 1949–1963. [CrossRef]

29. Wang, X.; Ye, J.; Lui, J.C. Mean Field Graph Based D2D Collaboration and Offloading Pricing in Mobile Edge Computing. *IEEE/ACM Trans. Netw.* **2023**, *32*, 491–505. [CrossRef]
30. Cao, X.; Wang, F.; Xu, J.; Zhang, R.; Cui, S. Joint Computation and Communication Cooperation for Energy-Efficient Mobile Edge Computing. *IEEE Internet Things J.* **2019**, *6*, 4188–4200. [CrossRef]
31. Khan, W.Z.; Ahmed, E.; Hakak, S.; Yaqoob, I.; Ahmed, A. Edge computing: A survey. *Future Gener. Comput. Syst.* **2019**, *97*, 219–235. [CrossRef]
32. Winn, J.; Criminisi, A.; Minka, T. Object categorization by learned universal visual dictionary. In Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1, Beijing, China, 17–21 October 2005; Volume 2, pp. 1800–1807.
33. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2117–2125.
34. Haralick, R.M.; Shapiro, L.G. Image segmentation techniques. *Comput. Vis. Graph. Image Process.* **1985**, *29*, 100–132. [CrossRef]
35. Cooper, R.B. Queueing theory. In *Proceedings of the ACM'81 Conference*; ACM: New York, NY, USA, 1981; pp. 119–122.
36. Wen, Y.; Zhang, W.; Luo, H. Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones. In Proceedings of the Infocom, Orlando, FL, USA, 25–30 March 2012.
37. Coulouris, G.; Dollimore, J.; Kindberg, T. *Distributed Systems—Concepts and Designs*, 3rd ed.; Elsevier: Amsterdam, The Netherlands, 2002.
38. Lu, C. Visual relationship detection with language priors. In Proceedings of the Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016.
39. Viola, P.; Jones, M. Robust real-time object detection. *Int. J. Comput. Vis.* **2001**, *4*, 4.
40. Moreno, P.J.; Joerg, C.; Thong, J.M.V.; Glickman, O. A recursive algorithm for the forced alignment of very long audio segments. In Proceedings of the Fifth International Conference on Spoken Language Processing, Sydney, Australia, 30 November–4 December 1998.
41. Toutanova, K.; Klein, D.; Manning, C.D.; Singer, Y. Feature-rich part-of-speech tagging with a cyclic dependency network. In Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology–Volume 1, Edmonton, AB, Canada, 27 May–1 June 2023; Association for Computational Linguistics: Stroudsburg, PA, USA, 2003; pp. 173–180.
42. Uberi. Automatic Speech Recognition. Available online: https://github.com/RevoSpeechTech/speech-datasets-collection (accessed on 9 June 2018).
43. Tilusnet. Part-of-Speech Tagging. Available online: https://paperswithcode.com/dataset/penn-treebank (accessed on 6 June 2018).
44. Aeneas. Forced Alignment. Available online: https://github.com/RevoSpeechTech/speech-datasets-collection (accessed on 5 June 2018).
45. CMU. Automatic Speech Recognition Datasets. Available online: http://festvox.org/cmuarctic/ (accessed on 21 August 2018).
46. DARPA. Forced Alignment Datasets. Available online: https://paperswithcode.com/dataset/timit (accessed on 5 January 2022).
47. Cristian. Part-of-Speech Tagging Datasets. Available online: https://consonni.dev/datasets/wikilinkgraphs-rawwikilinks-snapshots/ (accessed on 1 March 2021).