Learning for Exception: Dynamic Service Caching in 5G-Enabled MECs with Bursty User Demands

Zichuan Xu†, Shengnan Wang†, Shipei Liu†, Haipeng Dai¶, Qiufen Xia*§, Weifa Liang‡, and Guowei Wu†
† School of Software, Dalian University of Technology, Dalian, China, 116621.
§ International School of Information Science & Engineering, Dalian University of Technology, Dalian, China, 116621
¶ State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu 210023, China.
‡ Research School of Computer Science, Australian National University, Canberra, ACT 2601, Australia
Emails: z.xu@dlut.edu.cn, 1608115329@qq.com, tapliu@163.com, haipengdai@nju.edu.cn, wliang@cs.anu.edu.au, qiufenxia@dlut.edu.cn, wgwdut@dlut.edu.cn.
*Corresponding author: Qiufen Xia, email: qiufenxia@dlut.edu.cn.

***Abstract*—Mobile edge computing (MEC) is envisioned as an enabling technology for extreme low-latency services in the next generation 5G access networks. In a 5G-enabled MEC, computing resources are attached to base stations. In this way, network service providers can cache their services from remote data centers to base stations in the MEC to serve user tasks in their close proximity, thereby reducing the service latency. However, mobile users usually have various dynamic hidden features, such as their locations, user group tags, and mobility patterns. Such hidden features normally lead to uncertainties of the 5G-enabled MEC, such as user demand and processing delay. This poses significant challenges for the service caching and task offloading in a 5G-enabled MEC. In this paper, we investigate the problem of dynamic service caching and task offloading in a 5G-enabled MEC with user demand and processing delay uncertainties. We first propose an online learning algorithm for the problem with given user demands by utilizing the technique of Multi-Armed Bandits (MAB), and theoretically analyze the regret bound of the algorithm. We also propose a novel architecture of Generative Adversarial Networks (GAN) to accurately predict the user demands based on small samples of hidden features of mobile users. Based on the proposed GAN model, we then devise an efficient heuristic for the problem with the uncertainties of both user demand and processing delay. We finally evaluate the performance of the proposed algorithms by simulations based on a realistic dataset of user data. Experiment results show that the performance of the proposed algorithms outperform existing algorithms by around 15%.**

***Index Terms*—Service caching; 5G-Enabled MECs; Bursty user demands; Online learning;**

I. Introduction

With the fast deployment of 5G networks, diverse 5G-enabled small-cell base stations, such as femtocells, picocells, and microcells, are providing various AI services for mobile users. By December 2017, a total of over 12 million small cells have been deployed worldwide, with forecasts as high as 70 million by 2025 [2]. Meanwhile, the technique of Mobile Edge Computing (MEC), which is one of the key pillars of 5G, promises to reduce service latency and bandwidth pressure, by deploying computing resource in the small cell base stations. One fundamental question in 5G enabled MECs is how to efficiently and effectively utilize the computing resources in base stations, such that the performance of both mobile users and network services are promoted. To this end, we can dynamically cache network services from remote data centers to base stations and offload user tasks to the cached services, so that user tasks can be implemented in their registered or near-by base stations and the performance of the 5G-enabled MEC can be promoted. In this paper, we consider the joint service caching and task offloading in a 5G-enabled MEC with both remote data centers and base stations.

Solving the problem of joint service caching and task offloading in 5G-enabled MECs faces many challenges: (1) It is vitally important to automatically answer various uncertainties in a 5G-enabled MEC, considering various time-varying features of the network and mobile users. The first uncertainty is that unexpected bursty traffic often happen in 5G-enabled MECs. For example, a sudden event can easily cause a lot of user demand on a femtocell network [3], [15]. Also, as shown in [24], the multimedia big data generated by the real world multimedia application shows a bursty pattern. In other words, it is difficult for us to grasp this rule of such burstiness in short time when mobile users would generate a lot of demands at once [40]. The second uncertainty is the time-varying processing delays of the base stations in a 5G-enabled MEC. Specifically, the delay incurred in each link (either wireless or wired) of the 5G-enabled MEC can vary depending on various situations and workloads of the link. Static or simple dynamically caching services and offloading tasks that ignore the afore-mentioned uncertainties will easily lead to performance degradation of the network services. Therefore, the design of dynamic caching and tasks offloading methods that consider such uncertainties is challenging [20]; (2) How to respond to the dynamic uncertainties of networks by given historical information of user features and network latencies. If the data sample of user features is small, how to dynamically learn uncertainties, cache services, and offload tasks in a 5G-enabled MEC is also significantly challenging. Besides, simple learning methods may not handle the dynamic uncertainties of the 5G-enabled MEC, thereby significantly reducing systems performance. How to minimize the regret in the process of online learning, and design near-optimal learning method is the third challenge; (3) Given the uncertainties of data volumes and processing latencies, predictions of such uncertainties is the usual method. However, due to the lack of large historical data of such uncertainties in typical applications, normal regression methods may not be able to predict them accurately. How to predict data volumes of users and processing latencies, considering that the data of network uncertainties is usually small (or hard-to-obtain) is another non-trivial challenge.

Although there are several studies on joint service caching and task offloading [1], [5], [6], [27], [17], [21], [22], [24], [33] in MECs, none of them focus on bursty user demands and network uncertainties (user demand and processing delay uncertainties) in a 5G-enabled MEC. To the best of our knowledge, we are the first to investigate the dynamic service caching and task offloading problem in a 5G-enabled MEC with both bursty user demands and processing uncertainties, by proposing an online-learning algorithm based on the technique of Multi-Armed Bandits (MAB). We also devise a novel Generative-Adversarial Network (GAN) guided prediction algorithm based on small samples of user information, by adopting InfoGAN and using mutual information to accurately predict time series data with the aim of avoiding model overfitting and collapse.

The main contributions of this paper are summarized as follows.

- We formulate the joint the service caching and task offloading problem in 5G-enabled MEC with both bursty user demands and processing delay uncertainties.
- We propose an online learning for a special case of the problem with given user demands, based on the MAB method, and analyze the bound of the regret of the algorithm.
- We devise a GAN-guided prediction method to accurately predict the user demands based on small amounts of historical information.
- Based on the proposed GAN model, we propose a learning-aided heuristic for the service caching and task offloading problem in a 5G-enabled MEC.
- We evaluate the performance of the proposed models and algorithms through experimental simulations based on realistic data traces, and the results show that the performance of the proposed algorithms outperform existing ones.

The remainder of the paper is arranged as follows. Section II investigates on the joint service caching and task offloading, and states the difference of this work from previous studies. Section III introduces the system model, notations and problem formulation. Section IV proposes an online learning algorithm to dynamically cache services and offload tasks in a 5G-enabled MEC. Section V devises a GAN-guided method to accurately predict the user demands and a heuristic based on the proposed novel GAN model. Section VI provides some experimental results on the performance of the proposed algorithm by extensive simulations, and Section VII concludes the paper.

II. Related Work

With the rapid development of mobile applications, users are requiring extreme low-latency and higher-quality network services. Hence, the technique of MEC has emerged to address the limited computing power of mobile devices and the huge delay caused by offloading the task to remote data centers in the core network that is far from mobile users [1], [5], [6], [19], [28], [29], [30], [31], [32].

Most existing studies in the literature focused on either task offloading or service placement in edge clouds, with the objective to optimize either latency or energy consumption of mobile devices. For instance, in [7], the authors aimed to minimize the system energy consumption and concurrently ensures the latency constraints of the computation tasks in a 5G heterogeneous network. In [9], the authors proposed a joint mobility-aware caching and base station placement scheme that considers user mobility. In [8], the authors formulated the multi-user computation offloading problem in a multi-channel wireless interference environment and proposed a game theoretic approach for it. In [12], the authors proposed an online task offloading algorithm for both a sequence of tasks or concurrent tasks, by considering both the heterogeneous tasks and dynamic channel status. It is worth noting that these works have rarely focused on jointly service caching and task offloading in 5G-enabled MECs.

The studies that are closely related to this study are [17], [27], [21], [22], [24], which considers the joint service caching and task offloading. For instance, the authors considered the problem of joint offloading and caching in wireless blockchain networks [17]. The authors in [27] devised an efficient online algorithm for the problem of joint service caching and task offloading in a dense cellular network. In addition, there are some researches focusing on tasks offloading in a 5G-enabled MEC, by focusing on either the user mobility [34] or variety of network structure [10], [16]. For example, in [35], the proposed architecture allows one user to be associated with multiple base stations for the 5G-enabled MEC. In [35], the authors considered the interference of links together with the various computational capacity of base stations in a 5G-enabled MEC. These studies however do not consider the bursty user demands and time-varying uncertain processing latencies of base stations in the 5G-enabled network. To the best of our knowledge, we are the first to consider the dynamic service caching and task offloading problem in a 5G-enabled MEC with bursty user demands and uncertain processing delays, by designing novel online-learning algorithm with a bounded regret and a GAN-guided heuristic for the problem.

III. Preliminary

In this section, we first introduce the system model, request and delay models. We then define the dynamic service caching problem with demand and processing uncertainties precisely.

A. System model

We consider a 5G-enabled heterogeneous mobile edge computing (MEC) $G = (\mathcal{BS}, E)$ that is operated by a network service provider, with a set $\mathcal{BS}$ of 5G base stations providing computing and bandwidth resources for mobile applications and a set E of links that interconnect the base stations. Since we consider a multi-tier 5G heterogeneous MEC network, each base station can be a macro base station or a small-cell base station. Each base station $bs_i \in \mathcal{BS}$ is attached with a cloudlet. Therefore, each base station bs_i has a computing resource capacity for executing tasks of mobile users. Denote by $C(bs_i)$ the capacity of base station bs_i. Such computing capacities can be virtualized using cloud technologies, such that network services and user tasks can be executed in Virtual Machines (VMs) or containers. It must be mentioned that the capacities of base stations are heterogeneous, depending on their types. For example, a macro base station can be installed with a cloudlet consisting of several AI servers for the processing of complicated tasks, while a small-cell (or even pico-cell) base station may be attached with an AI accelerator, such as Intel Movidius Neural Compute Stick [14], to speed-up AI inference. This means that macro-cell bases stations usually have higher computing capacities than small-cell and pico-cell base stations. In addition, each bs_i covers a set of users as long as they are within the transmission range of bs_i. However, different base stations have different cover ranges. Macro base stations have larger transmission ranges than small-cell and pico-cell ones. Each base station bs_i operates over the orthogonal full-duplex spectrum, and each small-cell base station is assigned with a licensed band that may overlap with the bands of other base stations. Fig. 1 illustrates the 5G heterogeneous MEC network G.

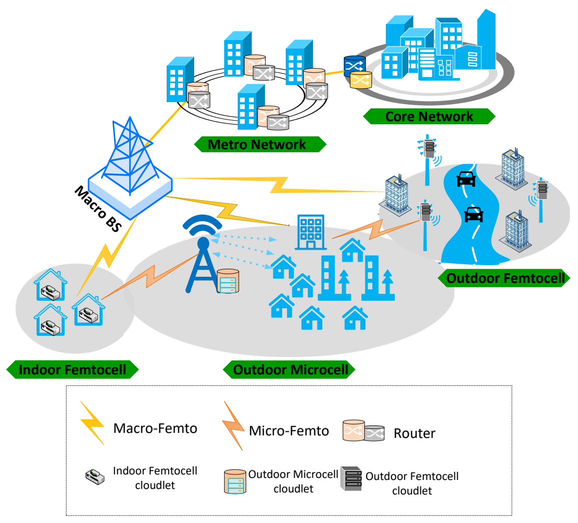


Fig. 1. An example of the 5G heterogeneous MEC network.

B. User requests with bursty demands

Mobile users access the 5G heterogeneous MEC network for various services, such as Virtual Reality (VR) services, cloud gaming, and IoT data processing, by connecting to their nearby base stations. Let r_l be a user request, and S_k its service for the processing of its data. Assuming that the time is divided into equal time slots, we use $\rho_l(t)$ to represent the amount of data that needs to be processed for request r_l in time slot t. The mobile user with request r_l has different amounts of data to be processed by its required service S_k in different time slots. Such data volumes of each request in different time slots have a bursty pattern. For example, VR services of a museum may experience a bursty amount of inference data if many people use its VR services suddenly. Nevertheless, we assume that there is basic demand of each request, which is the smallest data volume of each request during a finite-horizon monitoring period. Let ρ_l^{bsc} be the basic demand of r_l, and it is usually given as a priori. The bursty data volume of r_l basically is uncertain and not known in advance. Denote by $\rho_l^{bst}(t)$ the bursty data volume of r_l in time slot t. The data volume $\rho_l(t)$ of r_l at time slot t thus is

$$\rho_l(t) = \rho_l^{bsc} + \rho_l^{bst}(t). \tag{1}$$

The request r_l thus can be represented by $\langle \rho_l(t), S_k \rangle$. Assume that each base station assigns an amount C_{unit} of computing resource to process a unit amount of data of each request r_l. The computing resource demanded of request r_l in time slot t thus is $C_{unit} \cdot \rho_l(t)$.

C. Service caching

We focus on resource-hungry services that are originally deployed in the remote data centers in the core network, such as services for VR applications, which need to be cached into 5G base stations allowing mobile users to access them in extreme low latency. The migration of a service from remote data centers to a base station in G is referred to as *service caching*. If a service is cached into a base station of G, it can serve its requests within their proximity and handle the bursty demands of users in time. Denote by $\mathcal{S}$ a set of such services. Clearly, we have $S_k \in \mathcal{S}$, where $1 \leq k \leq |\mathcal{S}|$.

Each service $S_k \in \mathcal{S}$ has multiple users requesting to offload tasks and data to it for processing. Denote by $R(t)$ the set of requests that need to be implemented by services in $\mathcal{S}$ at the beginning of time slot t. The data $\rho_l(t)$ of each request $r_l \in R$ in time slot t needs to be served by its required service S_k. To implement request r_l, its data can be transferred to its service S_k that has already been cached into one of the base stations. In this way, the bursty data volume of request r_l in each time slot t can be processed in time.

D. Delay models

The delay experienced by implementing a request in cached service instances of service S_k in different base stations varies significantly from time to time. Denote by $d_{l,i}(t)$ the delay experienced by request r_j if its data volume is processed by an instance of service S_k in base station bs_i in time slot t. $d_{l,i}(t)$ depends on not only its data volume $\rho_l(t)$ but also many other factors, such as the congestion level of bs_i. Denote by $d_i(t)$ the delay of processing a unit amount of data in base station bs_i in time slot t, which varies in different time slots and is usually not known in advance. However, we assume that $d_i(t)$ does not change during time slot t, and can be obtained in the very beginning of time slot t.

The delay $d_{l,i}(t)$ of processing the data of request r_l in base station bs_i in time slot t thus is

$$d_{l,i}(t) = \rho_l(t) \cdot d_i(t). \tag{2}$$

Caching an instance of each service in a base station usually needs to set up its VM or container. Without loss of generality, we assume that the instantiation time of a service instance is a constant and given as a priori. However, the instantiation times of different services in different base stations may vary. Denote by $d_{i,k}^{ins}$ the instantiation delay of caching an instance of service S_k in base station bs_i.

E. Problem definition

We consider a 5G heterogeneous MEC network G with uncertain processing latencies of base stations, a set of services $\mathcal{S} = \{S_k \mid 1 \le k \le |\mathcal{S}|\}$ to be cached in the base stations of G, and a set of requests in R with uncertain bursty demands in different time slots. Assuming that the accumulative resources of all base stations is higher than the total resource demand of all requests, *the dynamic service caching problem with demand and processing delay uncertainties* is to dynamically cache services in $\mathcal{S}$ into base stations of G and assign each request r_l and its data $\rho_l(t)$ to one of the cached instance of its required service S_k, such that the average delay experienced by each request is minimized, subject to the computing resource capacity constraint of base stations.

The symbols used in this paper and their meanings are summarized in Table I.

IV. Online Learning Algorithm for the Dynamic Service Caching Problem with Given Demands

We now consider the dynamic service caching problem with given demands of user requests. Specifically, we assume that the demand of each request $\rho_l(t)$ does not change as time goes. For this problem, we propose an approximation algorithm with an approximation ratio for the service caching of each time slot. Based on the proposed approximation algorithm, we then devise an online learning algorithm with a bounded regret via leveraging the multi-armed bandit method.

A. Overview

Although the data volume of each request is assumed to be given and constant, the delay of processing a unit amount of data in each base station bs_i is still not known. To handle such uncertainty, we adopt the multi-armed bandit method to proactively learn the uncertain processing delay of each base station. Specifically, we consider each base station as a *bandit*. The bandit for each base station bs_i represents a random process of the delay of processing a unit amount of data in bs_i. Since we consider that the time is equally divided into time slots, we use $X_i(t)$ to denote the random process of the delay of processing a unit data in bs_i in time slot t. In the beginning of each time slot t, the algorithm can observe the processing delay of bs_i only when its arm is played. Based on the observed information of each arm, we are able to obtain the mean of random process X_i so far. Let θ_i be the mean of X_i, i.e., $\theta_i = E[X_i]$. Note that in the current time slot t, the mean θ_i is calculated based on the number of times that arm of bs_i is played, and let m_i be such a number of bs_i. The algorithm make decisions of service caching in future time slots, based on the observed information and the mean θ_i of each base station bs_i.

A usual approach to the problem of multi-armed bandits is to exploit good arms while exploring some random arms with probability ϵ. There thus has two important questions: (1) how to find 'good' arms according to the information observed so far, considering that it is NP-hard to cache services given the full knowledge of delay and user demand information, and (2) how to find values of ϵ, such that the regret of the obtained solution is bounded. To answer these two questions, our basic idea is to find an approximation algorithm for the dynamic service caching problem in each time slot t, such that the quality of the obtained solution is guaranteed. The algorithm then explores other arms that are not specified by the approximate solution randomly.

B. Online learning algorithm

Given the observed processing speed of each base station bs_i in the current time slot t, we now propose an approximation algorithm with an approximation ratio. The basic idea of the proposed approximation algorithm is to formulate the dynamic service caching problem with given request demands into an Integer Linear Program (ILP), based on the mean value of θ_i.

Recall that the objective of the dynamic service caching problem is to dynamically cache an instance of each service in $\mathcal{S}$, and assign each request r_l and its data $\rho_l(t)$ to its cached service, such that the average delay experienced by each request is minimized. Denote by x_{li} a binary indicator variable that shows whether request r_l is assigned to an instance of its required service S_k in base station bs_i. Let y_{ki} be an indicator variable that shows whether there is cached instance of service S_k in base station bs_i. The objective can be formulated as

$$\textbf{ILP}: \min \frac{1}{|R|}\Big(\sum_{r_l \in R}\sum_{bs_i \in \mathcal{BS}} x_{li}\cdot\rho_l(t)\cdot\theta_i + \sum_{S_k \in \mathcal{S}} y_{ki}\cdot d_{i,k}^{ins}\Big), \tag{3}$$

subject to the following constraints,

$$\sum\nolimits_{bs_i \in \mathcal{BS}} x_{li} = 1, \qquad \forall r_l \in R \tag{4}$$

$$\sum\nolimits_{r_l \in R} x_{li}\cdot\rho_l(t)\cdot C_{unit} \le C(bs_i), \quad \forall bs_i \in \mathcal{BS} \tag{5}$$

$$y_{ki} \ge x_{li}, \qquad \forall r_l \in R \text{ and its service } S_k \tag{6}$$

$$x_{li}, y_{ki} \in \{0,1\}. \tag{7}$$

Constraints (4) mean that there has to be base station to implement each request r_l. Constraints (5) say that the total resource demand of implementing these requests in each base station bs_i should not exceed its capacity. Constraints (6) mean if request r_l is assigned to base station bs_i its required service S_k has to be cached into bs_i; otherwise, its data cannot be processed.

TABLE I
SYMBOLS

Symbols	Meaning
$G = (\mathcal{BS}, E)$	a 5G-enabled heterogeneous mobile edge computing (MEC) network with a set $\mathcal{BS}$ of 5G base stations
bs_i	a base station in $\mathcal{BS}$
$C(bs_i)$	the computing capacity of base station bs_i
T and t	a finite time horizon that is equally divided into time slots and a time slot t of T
r_l and S_k	a user request and its required service
$R(t)$ and $\mathcal{S}$	a set of requests in time slot t and a set of services
R and $\|R\|$	a set of requests of all time slots in a finite time horizon and the number of requests in R
ρ_l^{bsc} and $\rho_l^{bst}(t)$	the basic data volume of request r_l and its bursty data volume in time slot t
$\rho_l(t) = \rho_l^{bsc} + \rho_l^{bst}(t)$	the amount of data that needs to be processed for request r_l in time slot t
C_{unit}	the amount of computing resource that is assigned to process a unit amount of data volume
$d_{l,i}(t)$	the delay experienced by request r_l if its data volume is processed by an instance of service S_k in base station bs_i
$d_i(t)$	the delay of processing a unit amount of data in base station bs_i in time slot t
$d_{i,k}^{ins}$	the instantiation delay of caching an instance of service S_k in base station bs_i
$X_i(t)$	the random process of the delay of processing a unit data in bs_i in time slot t
X_i	the random process of the delay of processing a unit data in bs_i
θ_i	the mean of X_i
ϵ_t	the probability of exploring some random arms of the MAB process in time slot t
m_i	the number of times that arm of bs_i is played
x_{li}	a binary indicator variable that shows whether request is assigned to an instance of its required service S_k in base station bs_i
y_{ki}	an indicator variable that shows whether there is cached instance of service S_k in base station bs_i
x^* and y^*	the obtained fractional solution to the relaxed **ILP**
γ	a threshold for x^* to make sure that base stations with higher probabilities are considered as *candidate base stations*
$\mathcal{BS}_l^{candi}$	the set of base stations for request r_l
$\mathcal{R}_t$	the regret of the MAB process
c	a constant with $0 < c < 1$
$\mathcal{G}$ and $\mathcal{D}$	the generator and discriminator of a GAN
C	latent of the GAN
c^t	the coding of user locations in time slot t
z^t	the noise vector in each time slot t
$\mathcal{G}(z^t, c^t)$	the predicted data volumes of requests in R
Q and c'^t	the predicted features of the predicted result, with each c'^t in the vector representing the coding of their predicted locations of users
$\mathcal{D}(\mathcal{G}(z^t, c^t))$	discriminator judges the predicted data volumes of generator $\mathcal{G}$
$\mathcal{D}(\rho_l(t))$	the discriminant result of true data and T is the length of a monitoring period T
$V'(\mathcal{D}, \mathcal{G})$	the loss function of the discriminator $\mathcal{D}$ and generator $\mathcal{G}$
$Q(c'^t \mid \rho_l(t))$	the distribution of hidden codes if predicted data
$P(c^t \| \rho_l(t)$	the distribution of hidden coding in true data
$I(c^t; \mathcal{G}(z^t, c^t)$	the the reduction of uncertainty in c^t when $\mathcal{G}(z^t, c^t)$ is observed
$L_1(\mathcal{G}, Q)$	the lower bound of $I(c^t; \mathcal{G}(z^t, c^t))$
$H(c^t)$	the entropy

We can solve the **ILP** to obtain the optimal solution to the problem. To proactively respond to the dynamics of networks, the length of a time slot needs to be as small as possible. However, a shorter length of each time slot may not be enough to solve the **ILP**, especially for large problem sizes. Instead, we relax the **ILP** into a Linear Program (LP), by relaxing integer variables of the **ILP** into

$$0 \leq x_{li}, y_{ki} \leq 1. \tag{8}$$

We then solve the **LP** and obtain a fractional solution. Since the fractional solution can be obtained efficiently in polynomial time, we then select base stations (arms) according to this fractional solution. Let x^* and y^* be the obtained fractional solution. It must be mentioned that the fractional solution is not a feasible solution to the dynamic service caching problem, since each service or request cannot be 'split' and assigned to different base stations.

Typically, in a solution to the MAB problem, the selection of optimal arm and random is done via the exploration and exploitation process. In contrast, our strategy is to select optimal arms according to the fractional solution, to ensure the performance guarantee of the solution. Specifically, we consider each fractional value of the fractional solution as a probability. For example, a higher value for x_{li} means that a higher probability that the selection of the arm of base station bs_i incurs a higher reward. Intuitively, we would like to select the arms with higher values of x^* and y^*. We thus define a threshold γ for x^* to make sure that base stations with higher probabilities are considered as *candidate base stations*. Denote by $\mathcal{BS}_l^{candi}$ the set of base stations for request r_l, which can be built by

$$\mathcal{BS}_l^{candi} = \{bs_i \mid x_{li}^* \geq \gamma\}. \tag{9}$$

Once obtained the candidate set of base stations for each request, it is ideal to select a base station from its candidate set. Then, we can obtain the information of the selected base stations. On the other hand, since the processing delay of each base station varies significantly, we also make some explorations to use other base stations that are not in the candidate set. That is, we adopt an ϵ_t greedy approach, in which we choose an arm from the candidate set with a high probability and pick an arm with its probability in x^*. By doing this, we exploit 'good base stations' that are considered having the potential to maximize the reward, while in the meantime, doing some explorations by picking random base stations with probability ϵ_t.

The detailed steps of the proposed algorithm for the dynamic service caching problem with given demands are shown in **Algorithm** 1.

Algorithm 1 `OL_GD`

Input: $G = (\mathcal{BS}, E)$, a set of services $\mathcal{S}$, and a set of requests R.
Output: The caching of each service $S_k \in \mathcal{S}$ and the assignment of the each request $r_k \in R$ to its cached service in a base station.

```
1: for $t \leftarrow 1, \cdots, T$ do
2:     $\epsilon_t \leftarrow \frac{1}{t}$;
3:     Relax the ILP in (3) into an LP;
4:     Obtain a fractional solution to the LP and get a candidate set $\mathcal{BS}_l^{candi}$ of base stations for each request $r_l$;
5:     Pick a random number in the range of $[0, 1]$;
6:     if the random number $< 1 - \epsilon_t$ then
7:         Asssign each request $r_l$ to base station $bs_i$ in $\mathcal{BS}_l^{candi}$ with probability $x^*_{li}$;
8:     else
9:         Randomly select a base station $bs_i \in \mathcal{BS} \setminus \mathcal{BS}_l^{candi}$ for each request $r_l$;
10:    for each base station $bs_i$ with assigned requests do
11:        Observe the delay of processing a unit amount of data and update its mean value $\theta_i$;
```

C. Algorithm analysis

Regret definition: We evaluate the quality of solutions to the problem with respect to the bound of regret, which is defined as the difference between the average delay that could be obtained by picking an optimal base station at each time and that obtained by the proposed algorithm. Regret thus can be expressed by

$$\begin{aligned}\mathcal{R}_t = & E[\sum_{r_l\in R}\sum_{bs_i\in\mathcal{BS}} X_i(t)\cdot\rho_l(t)\cdot\theta_i + y_{ki}(t)\cdot d_{i,k}^{ins}] \\ & - \min_{1\le\tau\le t}\{\sum_{r_l\in R}\sum_{bs_i\in\mathcal{BS}} x_{li}^{opt}(t)\cdot\rho_l(\tau)\cdot d_i(t) \\ & + y_{ki}^{opt}(t)\cdot d_{i,k}^{ins}\},\end{aligned} \tag{10}$$

where $x_{li}^{opt}(t)$ and $y_{ki}^{opt}(t)$ are the optimal request assignment and service caching in each time slot t.

To show the bound of regret of the proposed algorithm, we first analyze the gap between the optimal service caching and the worst service caching in the following lemma.

Lemma 1: Assuming that the distribution of $d_i(t)$ is not given while its maximum and minimum values are known in advance, the gap between the optimal service caching and the worst service caching is bounded by $\max\{|R|\cdot(d^{max} - \gamma\cdot d^{min} + \Delta^{ins}), |R|\cdot\gamma\cdot(1 - e^{-2\gamma|R|^2}) + \Delta^{ins}\}$, where $d^{max} = \arg\max_{i,t} d_i(t)$, $d^{min} = \arg\min_{i,t} d_i(t)$, and $\Delta^{ins} = \arg\max_{i,k} d_{i,k}^{ins} - \arg\min_{i,k} d_{i,k}^{ins}$.

Proof From the definition of regret in Eq. (10), the regret can be considered in the following two cases: **case 1:** the regret due to the deviation from the optimal solution if the delay of processing a unit data in each bs_i happens to be its currently observed mean value θ_i; and **case 2:** the regret due to the fact that the delay in the current delay of each bs_i deviates from θ_i.

For **case 1:** It is equivalent to calculating the worst bound of

$$\begin{aligned}& E(\mathcal{R}(T)) \\ & = E[\sum_{r_l\in R}\sum_{bs_i\in\mathcal{BS}} x_{li}(t)\cdot\rho_l(t)\cdot d_i(t) + y_{ki}(t)\cdot d_{i,k}^{ins}] \\ & - \sum_{r_l\in R}\sum_{bs_i\in\mathcal{BS}}\Big(x^*_{li}(t)\cdot\rho_l(\tau)\cdot d_i(t) + y^*_{ki}(t)\cdot d_{i,k}^{ins}\Big).\end{aligned} \tag{11}$$

Recall that in the proposed algorithm base station bs_i is considered as a candidate base station for r_l if $x^*_{li}\ge\gamma$. In the worst case, a base station with the highest delay of processing unit data with high probability, while the base station with the lowest delay is selected with probability γ. This means that

$$\begin{aligned}E(\mathcal{R}(T)) \le & \sum_{r_l\in R}\sum_{bs_i\in\mathcal{BS}}\big(\rho_l(t)\cdot d^{max} \\ & - \gamma\cdot\rho_l(t)\cdot d^{min} + \Delta^{ins}\big), \\ \le & |R|\cdot(d^{max} - \gamma\cdot d^{min} + \Delta^{ins}).\end{aligned} \tag{12}$$

For **case 2:** We can calculate the regret by

$$\begin{aligned}E(\mathcal{R}(T)) = E\big[& \sum_{r_l\in R}\sum_{bs_i\in\mathcal{BS}}\rho_l(t)\cdot\theta_i \\ & - \gamma\cdot\rho_l(t)\cdot d_i(t) + \Delta^{ins}\big]\end{aligned}$$

By Chernoff-Hoeffding bound [25], we know that

$$Pr[d_i(t) > n\theta_i + a] \le e^{-2a^2/n}, \tag{13}$$

which means that

$$Pr[\theta_i - d_i(t)\cdot\gamma < |R|\cdot\gamma] \le e^{-2\gamma|R|^2}, \tag{14}$$

if $n = \frac{1}{\gamma}$ and $a = |R|$.

We thus have

$$\begin{aligned}E(\mathcal{R}(T)) = E\big[& \sum_{r_l\in R}\sum_{bs_i\in\mathcal{BS}}\rho_l(t)\cdot\theta_i \\ & - \gamma\cdot\rho_l(t)\cdot d_i(t) + \Delta^{ins}\big] \\ \le & |R|\cdot\gamma\cdot(1 - e^{-2\gamma|R|^2}) + \Delta^{ins}.\end{aligned} \tag{15}$$

Denote by σ the gap between the optimal service caching and the worst service caching. We then have

$$\begin{aligned}\sigma = & \max\{|R|\cdot(d^{max} - \gamma\cdot d^{min} + \Delta^{ins}), \\ & |R|\cdot\gamma\cdot(1 - e^{-2\gamma|R|^2}) + \Delta^{ins}\}.\end{aligned} \tag{16}$$

We then show the bound on regret in the following theorem.

Theorem 1: The expected regret of the algorithm is upper bounded by $\sigma \log \frac{T-1}{e^{1/c}+1}$.

Proof The regret can be divided into two parts: (1) the regret due to the exploring of sub-optimal solutions that are not considered as candidate base stations, (2) the regret because of exploring to discover the optimal solution, and (3) the regret due to deviating from the optimal solution by having certain probability to explore suboptimal solutions.

For part (1), the sub-optimal solutions without the candidate set is explored with probability $(1-\gamma)$. Such base stations may be optimal solutions given their delay information. Let k be the number of time slots that an optimal base station is found, we can calculate k by

$$\sum_{t=1}^{k}(1-\gamma) \geq 1. \tag{17}$$

We thus have

$$k = \frac{1}{1-\gamma}. \tag{18}$$

For part (2), we can see that the probability of exploring solutions is $\frac{c}{t}$ in each time slot t, where $0 < c < 1$. Assuming that k' is the number of time slots that the request is assigned to its optimal base station, k' can be derived by

$$\sum_{t=1}^{k'}(\frac{c}{t}) \geq 1, \tag{19}$$

We then have

$$k' = e^{1/c}. \tag{20}$$

Assuming that $\gamma < 1 - e^{-1/c}$, the regret in parts (1) and (2) is

$$\sigma \cdot e^{1/c}. \tag{21}$$

For part (3), the expected regret till time slot t is calculated by

$$\begin{aligned} E(R(T)) &\leq \sigma \sum_{t=k+1}^{T} \frac{c}{t} \\ &\leq \sigma \int_{t=e^{1/c}+1}^{T-1} \frac{1}{x} dx = \sigma \log \frac{T-1}{e^{1/c}+1}. \end{aligned} \tag{22}$$

V. GAN-Guided Heuristic for the Dynamic Service Caching Problem

So far we assumed that the data volume of each request r_l is given, request r_l however may have time-varying data volume in different time slots in reality. In this section we remove this assumption by considering the fact that data volumes of users are not known. We propose a GAN-guided heuristic for the dynamic service caching problem with demand and processing delay uncertainties.

A. Basic idea

The basic idea of the proposed algorithm is to predict the data volumes of each request r_l in each time slot t according to the historical information. Given the predicted value, we then invoke algorithm `OL_GD` to obtain a solution for request assignment and service caching. We observe that the bursty of requests' data volumes depends on many hidden features, such as its location, its registered base station, and the current social events in its location. In particular, users in the same location may have similar distributions of their data volumes. For example, a few users may be playing the same VR game, and their data volumes can be similar with each other.

On one hand, the data samples of such hidden features are usually small and hard-to-obtain. How to predict data volumes of users based on small samples of user data is challenging. The technique of Generative Adversarial Networks (GANs) is emerging as a key enabling prediction method to deal with small samples. GANs essentially adopt the concept of a non-cooperative game in which two deep neural networks, a generator ($\mathcal{G}$) and a discriminator ($\mathcal{D}$), are trained to play against each other. The objective of $\mathcal{G}$ is to generate data volumes resembling to samples that are generated from true data distributions, while $\mathcal{D}$'s purpose is to distinguish between samples drawn from $\mathcal{G}$ and samples drawn from the true data distribution. In addition, the generator uses noise z without any restrictions in the GAN model. Our basic idea is to allow generator $\mathcal{G}$ predict the data volumes of all requests in R, and $\mathcal{D}$ judges whether the prediction is correct or wrong, until $\mathcal{G}$ predicts the correct data volumes of requests with high probabilities.

On the other hand, it is better for the prediction model to consider the variety of many different hidden features of user requests. It is known that Mutual Information GAN (InfoGANs) are capable of handling such user data with various features. We thus adopt InfoGANs with its generator and discriminator that adopt Recurrent Neural Networks (RNN). Then learnt features can be decomposed and explained by mutual information, which is denoted by I.

It must be mentioned that the novelty of this model include (1) the adoption of InfoGAN to process time series data of user information, (2) a novel feature dimension is proposed through the prior probability distribution of the latent variables. At the same time latent information can control the highly free of GAN model itself and provide an explanatory characteristic dimension, such that the model collapse is avoided.

B. GAN model

In the model of Info-RNN-GAN, we use C (latent) to represent the hidden features of users, such as the user group tag inferred from user information (location) which optimized the learn direction of generator to data volume that we wanted. In our paper, we preprocess the location of the data with one-hot encoding [39] and then treat it as the value of C, with c^t representing the coding of their locations in time slot t. It means that we can use data feature to judge data correctness. Another input of each cell is the noise vector z^t in each time slot t, as shown in Fig. 2.

Because of the continuous user activities, the prediction of their data volumes can be affected by both of their historical and future features. We thus adopt a bidirectional two-layer loop RNN (Bi-LSTM) [13] which can make decisions based on their historical and future features in the data sample, so that user behaviors can be learned from bi-directions from the current time slot. Specifically, generator ($\mathcal{G}$) adopts a Bi-LSTM to learn the features of user features. Then, softmax [18] is used to predict the data volume of the user in real time. The output is the predicted data volumes of requests in R denoted by $\mathcal{G}(z^t, c^t)$, which is also considered as *fake data*. Let $\rho_l(t)$ be the *true data* in the historical data samples of user requests. Discriminator $\mathcal{D}(\mathcal{G}(z^t, c^t))$ uses a two-layer Bi-LSTM to judge how close the fake data is from the true data. Let Q be the predicted features of the predicted result, with each c'^t in the vector representing the coding of their predicted locations of users. Q is then used to determine whether it is similar to latent C or not by the discriminator. To evaluate the closeness of the false data and true data in a monitoring period T, we use the following loss function,

$$V'(\mathcal{D}, \mathcal{G}) = \frac{1}{T}\sum_{t=1}^{T}[\log \mathcal{D}(\rho_l(t)) + \log(1 - \mathcal{D}(\mathcal{G}(z^t, c^t)))] \quad (23)$$

where $\mathcal{D}(\rho_l(t))$ represents the discriminant result of true data and T is the length of a monitoring period.

Recall that the mutual information needs to be considered, we then extend the above loss function by

$$\min_{\mathcal{G}} \max_{D} V(\mathcal{D}, \mathcal{G}) = V'(\mathcal{D}, \mathcal{G}) - \frac{\lambda}{T}\sum_{t=1}^{T} I(c^t; \mathcal{G}(z^t, c^t)), \quad (24)$$

where λ is a given constant, $V'(\mathcal{D}, \mathcal{G})$ is the loss function of the discriminator $\mathcal{D}$ and generator $\mathcal{G}$ of the Info-RNN-GAN model and $I(c^t; \mathcal{G}(z^t, c^t))$ is the the reduction of uncertainty in c^t when $\mathcal{G}(z^t, c^t)$ is observed. If the value of $I(c^t; \mathcal{G}(z^t, c^t))$ is larger, they are more relevant.

Since the distributions of the mutual information is hard to maximize directly in order to disentangled representation of result, we use a lower bound to estimate the value of the mutual information by generating the direction $Q(c'^t \mid \rho_l(t))$ to approximate $P(c^t \mid \rho_l(t))$, where the $Q(c'^t \mid \rho_l(t))$ represents the distribution of hidden codes in fake data and the $P(c^t \mid \rho_l(t))$ represents the distribution of hidden coding in true data. Denote by $L_1(\mathcal{G}, Q)$ the lower bound of $I(c^t; \mathcal{G}(z^t, c^t))$, which can be calculated by

$$\begin{aligned} L_1(\mathcal{G}, Q) = \\ &E_{\rho_l(t)\sim\mathcal{G}(z^t,c^t)}[E_{c^{t\prime}\sim P(c^t|\rho_l(t))}[\log Q(c'^t \mid \rho_l(t))]] \\ &+ H(c^t) < I(c^t; \mathcal{G}(z^t, c^t)), \end{aligned} \quad (25)$$

where the entropy $H(c^t)$ is assumed to be a given constant for simplicity.

The loss function can be approximately solved as:

$$\min_{\mathcal{G},Q} \max_{D} V(\mathcal{D}, \mathcal{G}, Q) = V'(\mathcal{D}, \mathcal{G}) - \lambda\frac{1}{m}\sum_{t=1}^{m} L_1(\mathcal{G}, Q)) \quad (26)$$

The proposed model enables us to continuously optimize the generated value based on the training model and probability

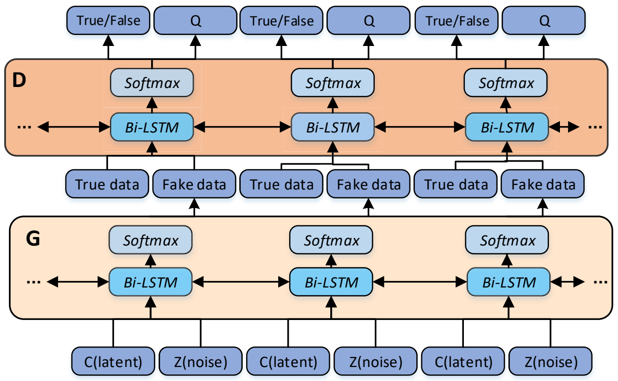


Fig. 2. Info-RNN-GAN. The generator $\mathcal{G}$ produces sequences of predicted data volumes of requests. The discriminator ($\mathcal{D}$) is trained to distinguish between real data volume and generated data volume

distribution of true data. The model has two advantages: (i) we use Bi-LSTM to learn the data features of the sequence in the generator. (ii) we control the direction of generated data for hidden features based on mutual information. Therefore, we will eventually get the predicted data volumes of request as the loss function approaches to a stable status.

C. Algorithm

We now describe the proposed algorithm. For each request r_l, its data volume may be related to the context or environment that can be extracted from its historical data. It is known that a recurrent neural network (RNN) has the ability of exploring temporal dynamic behavior, by building connections between nodes form a directed graph along a temporal sequence. We thus adopt the Info-RNN-GAN model, which is a continuous recurrent network with adversarial training [23], where the discriminator consists of a bidirectional recurrent neural network, allowing it to take context in both directions into account for its decisions. Given the predicted values of data volume of each request r_l in time slot t, the algorithm then caches the services and assign all requests, by invoking **Algorithm** 1. The discriminator $\mathcal{D}$ then evaluates the quality of the prediction and feeds the information to the generator $\mathcal{G}$. The later then re-generates its predictions for the next time slot based on the feedback. The detailed steps of the proposed algorithm are shown in **Algorithm** 2, which is referred to as `OL_GAN`.

VI. Experiments

We now evaluate the proposed algorithms by experimental simulations in both synthetic and real networks.

A. Parameter settings

We consider a 5G heterogeneous network G with its size being varied from 20 to 200 base stations, where each network topology is generated using GT-ITM [11]. In each of the generated network topology, each pair of base station has a probability of 0.1 of being connected. We mainly consider

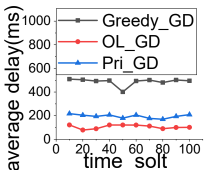
Algorithm 2 `OL_GAN`

```
Input: $G = (\mathcal{BS}, E)$, a set of services $\mathcal{S}$, and a set of requests $R$.
Output: The caching of each service $S_k \in \mathcal{S}$ and the assignment of
    the each request $r_k \in R$ to its cached service in a base station.
1:  for $t \leftarrow 1, \cdots, T$ do
2:      for each request $r_l$ do
3:          Generator $\mathcal{G}$ predicts the data volume of $r_l$;
4:      $\epsilon_t \leftarrow \frac{1}{t}$;
5:      Relax the ILP in (3) into an LP;
6:      Obtain a fractional solution to the LP and get a candidate set
        $\mathcal{BS}_l^{candi}$ of base stations for each request $r_l$;
7:      Pick a random number in the range of $[0, 1]$;
8:      if the random number $< 1 - \epsilon_t$ then
9:          Assign each request $r_l$ to base station $bs_i$ in $\mathcal{BS}_l^{candi}$ with
            probability $x_{li}^*$;
10:     else
11:         Randomly select a base station $bs_i \in \mathcal{BS} \setminus \mathcal{BS}_l^{candi}$ for
            each request $r_l$;
12:     for each base station $bs_i$ with assigned requests do
13:         Observe the delay of processing a unit amount of data and
            update its mean value $\theta_i$;
14:     for each request $r_l$ do
15:         Discriminator $\mathcal{D}$ observes the real data volume of $r_l$ and
            calculates its loss;
```

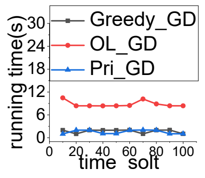
three kinds of base stations, i.e., macro, micro, and femto base stations. The transmit power of macro, micro, and femto base stations are set to 40W, 5W and 0.1W, respectively. The system bandwidth and the modulation scheme are considered as 20MHz, and 64QAM respectively as per the 3GPP standard.

The macro base station is deployed in the center while the femto and micro base stations are randomly deployed within the transmission region of the macro base station. The radius of femto, micro, and macro base stations are set to 15m, 30m, 100m, respectively. First, each macro base station has a computing capacity in the range 8,000 to 16,000 MHz, and a bandwidth capacities being varied between 500 $Mbps$ and 1,000 $Mbps$. The average delay experienced by a user in a cached instance of macro base station is a value between 30 and 50 milliseconds and the average delay experienced in a remote data center is a value between 50 and 100 milliseconds. Second, each micro base station has a computing capacity in the range from 5,000 to 10,000 MHz and a bandwidth capacity in the range from 200 $Mbps$ to 500 $Mbps$. The average delay in a micro base station is a value that is randomly withdrawn in the range of $[10, 20]$ milliseconds. Third, each femto base station has a computing and bandwidth capacities in the ranges of $[1,000,\ 2,000]$ MHz, respectively; the average delay in it is a value in the range of $[5, 10]$ milliseconds. We extract a sample of user information from the dataset of NYC Wi-Fi hotspot locations [26]. This dataset consists of many small sample data features, which satisfies the needs of our experiment. According to the extracted data of user information, we used Info-RNN-GAN network to learn the location, time, service status and other characteristics of each request. Unless otherwise specified, we will adopt these default settings in our experiments. Each value in the figures is the mean of the results by applying each mentioned algorithm on 80 different topologies of the 5G small-cell MEC.

Benchmarks: We compare the proposed algorithms with

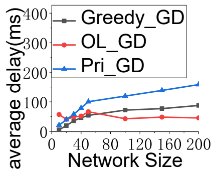


(a) Average Delay.

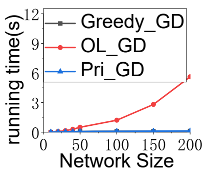


(b) Running Time.

Fig. 3. The performance of algorithms `OL_GD`, `Greedy_GD` , `Pri_GD` in a period of 100 time slots.



(a) Average Delay.



(b) Running Time.

Fig. 4. The performance of algorithms `OL_GD`, `Greedy_GD` , `Pri_GD` in networks with base stations being varied from 50 to 200.

the following three algorithms:

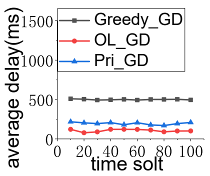
- greedy approach: each base station greedily selects a service and its tasks that could minimizes the delay of each request, assuming that the data volume of each request is given. This greedy approach with given data demands is referred to as `Greedy_GD`.
- a priority-driven service caching algorithm in [20]: the algorithm assigns each request a priority according to the number of base stations covered in its transmission range, and the base station takes priority in processing the high priority request. Similarly, this priority driven approach is referred to as `Pri_GD` with given data volumes of requests.
- an online algorithm with single an autoregression prediction, i.e., `OL_Reg`. Algorithm `OL_Reg` predicts the bursty demand following an autoregressive moving average (ARMA) model. Let $\hat{\rho}_l(t)$ be the predicted bursty data volume of request r_l in time slot t. Using the information of the previous p time slots, assuming that the value of p is given, $\hat{\rho}_l(t)$ can be predicted by the following ARMA model,

$$\begin{aligned}\hat{\rho}_l(t) &= a_1 \cdot \rho_l(t-1) + a_2 \cdot \rho_l(t-2) + \dots \\ &\quad + a_p \cdot \rho_l(t-p), \end{aligned} \tag{27}$$

where $a_{p'}$ is a constant with $0 \le a_{p'} \le 1$, $\sum_{l=1}^{p} a_l = 1$, and $a_{p_1} \ge a_{p_2}$ if $p_1 < p_2$.

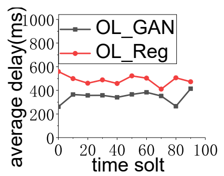
B. Performance of the algorithms with given demands

We first evaluate the performance of algorithms `OL_GD` against algorithms `Pri_GD`, and `Greedy_GD` assuming that the data volumes of user requests are given, in a network of 100 base stations of a finite time horizon of 100 time slots. Fig. 3 (a) shows the average delay obtained by algorithms `OL_GD`, `Greedy_GD`, and `Pri_GD`. It can be seen that algorithm `OL_GD` has the lowest average delay while `Greedy_GD` has the highest average delay. For example,

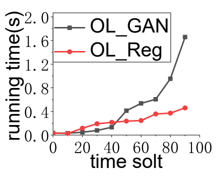


(a) Average Delay.

Fig. 5. The performance of algorithms OL_GD, Greedy_GD , Pri_GD in a period of 100 time slots.

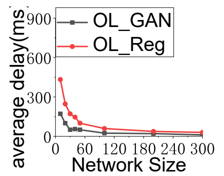


(a) Average Delay.

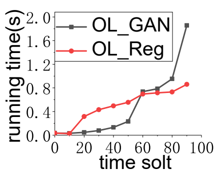


(b) Running Time.

Fig. 6. The performance of algorithms OL_GAN, OL_Reg in a period of 100 time slots.



(a) Average Delay.



(b) Running Delay.

Fig. 7. The performance of algorithms OL_GAN, OL_Reg in a period of 100 time slots and the performance of algorithms OL_GAN, OL_Reg in networks with base stations being varied from 50 to 300.

algorithm OL_GD has at least 15% lower delay than that of algorithm Pri_GD. The reason is that algorithm OL_GD handles the uncertainty of processing delay via a near-optimal online learning algorithm. Instead, algorithms Greedy_GD and Pri_GD cache services and offload user tasks according to the historical information of processing latencies of base stations in the network. The ignorance of the mentioned uncertainty may result the transmission of data volume of a request along a link with high latency. From Fig. 3 (b), we can see that algorithm OL_GD has only marginally higher running time than that of algorithms Greedy_GD and Pri_GD.

We then study the performance of the algorithms OL_GD, Greedy_GD, and Pri_GD by varying the network size from 50 to 200. The results are depicted in Fig. 4, from which we can see that the obtained average delay of algorithm OL_GD is the lowest. When the size of the base station is small in Fig. 4 (a), the average delay of algorithm OL_GD is not the lowest because the proposed algorithm OL_GD explores solutions that are suboptimal with a probability. However, the greedy approach has a higher probability of finding a better solution considering that the network size is small and the solution space is small. Furthermore, the average delays of the algorithms are varied with the growth of the number of base stations, because more services can be cached into the 5G-enabled network and increasing the probability of executing user requests in base stations with shorter execution latency. The running time is shown in Fig. 4 (b). Although the running time of algorithm OL_GD is increasing faster than the other two algorithms, the running time gap is still trivial. This means that algorithm OL_GD maintains efficiency in large scale networks.

We then study the performance of the proposed algorithms in a real network AS1755 in a time horizon of 100 time slots. The results are shown in Fig. 5. From Fig. 5 (a), we can see that algorithm OL_GD achieves a constant lower network delay than algorithms Greedy_GD and Pri_GD in real network AS1755. Furthermore, the performance gap is enlarged in real network AS1755 compared with that in a synthetic network shown in Fig. 3. This is because there is usually more bottleneck links in real network topologies than the synthetic ones.

C. Performance of the algorithms without given demands

We first evaluate the performance of algorithms OL_GAN and OL_Reg assuming that the data volumes of user requests are not given, in a network of 100 base stations of a finite time horizon of 100 time slots. Fig. 6 (a) shows the average delay, from which we can see that algorithm OL_GAN has a much lower average delay than algorithm OL_Reg. The rationale behind is that algorithm OL_Reg adopts a simple ARMA model to predict user demands, and the prediction method replies on the historical data. Usually, the accuracy of ARMA model will be higher if there is more historical information. In contrast, algorithm OL_GAN adopts a GAN-based method that works very well in small volume of historical data. The running times of both algorithms are shown in Fig. 6 (b), from which we can see that algorithm OL_GAN has around 400% higher average running time than that of algorithm OL_Reg. We can also find that the running time of algorithm OL_GAN is very low in network AS1755 in Fig. 7.

We then investigate the performance of the algorithms OL_GAN and OL_Reg by varying the network size from 50 to 300. The results are depicted in Fig. 7, from which we can see that the obtained average delay of algorithm OL_GAN is the lower than algorithm OL_Reg. Also, the average delays obtained by algorithms OL_GAN and OL_Reg are decreasing with the growth of network sizes. This is because there are more base stations with lower processing delays with the growth of base station numbers.

VII. Conclusion

In this paper we studied the problem of dynamic service caching and task offloading in 5G-enabled MECs by considering the bursty user demands and the uncertainties of processing latencies. For a special case of the problem with given user demands, we propose an online learning algorithm based on the MAB method, and analyzed the regret bound of the algorithm. We also proposed a novel prediction method to accurately predict the user demands based on the state-of-the-art GAN model. Based on the proposed prediction model, we devised an efficient heuristic for the problem with user demand and processing delay uncertainties. We finally investigated the performance of the proposed algorithms by simulations based on a realistic dataset of user data. Experiment results show that the performance of the proposed algorithms outperform the comparison algorithms by 15%.

Acknowledgments

The work of Zichuan Xu, Qiufen Xia, and Guowei Wu is partially supported by the National Natural Science Foundation of China (Grant No. 61802048, 61802047, 61772113, and 61872053), the fundamental research funds for the central universities in China (Grant No. DUT17RC(3)061, DUT17RC(3)070, DUT19RC(4)035, and DUT19GJ204), DUT-RU Co-Research Center of Advanced ICT for Active Life, and the "Xinghai Scholar Program" in Dalian University of Technology, China. The work by Haipeng Dai is supported in part by the National Natural Science Foundation of China (Grant No. 61872178) and the Natural Science Foundation of Jiangsu Province (Grant No. BK20181251). The work by Weifa Liang is supported by the Australian Research Council Discovery Project (Grant No. DP200101985).

References

[1] M. Barbera, S. Kosta, A. Mei, and J. Stefa, To offload or not to offload? The bandwidth and energy costs of mobile cloud computing. *Proc. INFOCOM*, IEEE, 2013.

[2] Small Cell Market Status Statistics Dec 2017. scf.io. Retrieved 2018-02-19.

[3] N. Bronson, *et al.*. Tao: Facebooks distributed data store for the social graph. *Proc. of ATC*, 2013

[4] W. Chen, Y. Wang, and Y. Yuan Combinatorial multi-armed bandit:General framework and applications, *Proc. of ICML*, PMLR, 2013.

[5] M. Chen, Y. Qian, Y. Hao, Y. Li, and J. Song, Data-driven computing and caching in 5G networks: Architecture and delay analysis. *IEEE Wireless Communications*, vol. 25, no. 1, pp. 70–75, 2018.

[6] M. Chen, Y. Hao, Y. Li, C.-F. Lai, and D. Wu, On the computation offloading at ad hoc cloudlet: Architecture and service modes. *IEEE Communications Magazine*, vol. 53, no. 6, pp. 18–24, 2015.

[7] C-K Chau and K. Elbassioni. Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks. *IEEE Transactions on Control of Network Systems*, vol. 5, no. 4, pp. 1809–1818, 2018.

[8] X. Chen, L. Jiao, W. Li, and X. Fu, Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2016.

[9] M. Chen, Y. Hao, L. Hu, K. Huang, and V. Lau. Green and mobility-aware caching in 5G networks. *IEEE Transactions on Wireless Communications*, vol. 16, no. 12, pp. 8347–8361, 2017.

[10] M. Deng, H. Tian, and X. Lyu, Adaptive sequential offloading game for multi-cell mobile edge computing. *Proc. of ICT*, IEEE, 2016.

[11] GT-ITM. http://www.cc.gatech.edu/projects/gtitm/.

[12] M. Jia, J. Cao, and L. Yang. Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing. *Proc. of INFOCOM WKSHPS*, IEEE, 2014.

[13] E. Kiperwasser and Y. Goldberg. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, vol. 4, pp.313–327, ACL, 2016.

[14] Intel Neural Compute Stick. https://software.intel.com/en-us/movidius-ncs, accessed Jan 2020.

[15] P. Jin, J. Guo, Y. Xiao, R. Shi, Y. Niu, F. Liu, C. Qian, and Y. Wang. PostMan: Rapidly mitigating bursty traffic by offloading packet processing. *Proc. of ATC*, USENIX, 2019.

[16] I. Ketyk, L. Kecsks, C. Nemes, and L. Farkas, Multi-user computation offloading as multiple knapsack problem for 5G mobile edge computing. *Proc. of EuCNC*, IEEE, 2016.

[17] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song. Joint computation offloading and content caching for wireless blockchain networks. *Proc. of INFOCOM*, IEEE, 2018.

[18] A. Martins and R. Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. *Proc. of ICML*, 2016.

[19] H. Ren, Z. Xu, W. Liang, Q. Xia, P. Zhou, O. F. Rana, A. Galis, and G. Wu. Efficient algorithms for delay-aware NFV-enabled multicasting in mobile edge clouds with resource sharing. To appear in *IEEE Transactions on Parallel and Distributed Systems*, 2020. DOI: 10.1109/TPDS.2020.2983918.

[20] Q. Xie, Q. Wang, N. Yu, H. Huang and X. Jia. Dynamic service caching in mobile edge networks. *Proc. of MASS*, IEEE, 2018.

[21] H. Wang, R. Li, L. Fan, and H. Zhang. Joint computation offloading and data caching with delay optimization in mobile-edge computing systems. *Proc. of WCSP*, IEEE, 2017.

[22] T. Tran, K. Chan, and D. Pompili. COSTA: Cost-aware service caching and task offloading assignment in mobile-edge computing. *Proc. of SECON*, IEEE, 2019.

[23] M. Mathieu, J. Zhao, P. Sprechmann, A. Ramesh and Y. LeCun. Disentangling factors of variation in deep representation using adversarial training. *Proc. of NIPS*, 2016.

[24] W. Miao, G. Min, Y. Wu, H. Wang, and J. Hu. Performance modelling and analysis of software-defined networking under bursty multimedia traffic. *ACM Transactions on Multimedia Computing, Communications, and Applications*, Article No.77, 19 pages, September 2016.

[25] M. Mitzenmacher and E. Upfal. Probability and Computing: Randomized Algorithms and Probabilistic Analysis. *Cambridge University Press*, 2005.

[26] NYC Wi-Fi hotspot locations. https://data.world/city-of-ny/a9we-mtpn, accessed in Jan 2020.

[27] J. Xu, L. Chen, and P. Zhou. Joint service caching and task offloading for mobile edge computing in dense networks. *Proc. of INFOCOM*, IEEE, 2018.

[28] Z. Xu, W. Gong, Q. Xia, W. Liang, O. F. Rana, and G. Wu. NFV-enabled IoT service provisioning in mobile edge clouds. To appear in *IEEE Transactions on Mobile Computing*, 2020. DOI: 10.1109/TMC.2020.2972530.

[29] Z. Xu, W. Liang, M. Jia, M. Huang, and G. Mao. Task offloading with network function services in a mobile edge-cloud network. *IEEE Transactions on Mobile Computing*, Vol.18, No.11, pp.2672 – 2685, 2019.

[30] Z. Xu, W. Liang, A. Galis, and Y. Ma. Throughput maximization and resource optimization in NFV-enabled networks. *Proc. of ICC'17*, IEEE, 2017.

[31] Z. Xu, W. Liang, M. Huang, M. Jia, S. Guo, and A. Galis. Approximation and online algorithms for NFV-enabled multicasting in SDNs. *Proc of ICDCS*, IEEE, 2017.

[32] Z. Xu, W. Liang, M. Huang, M. Jia, S. Guo, and A. Galis. Efficient NFV-enabled multicasting in SDNs. *IEEE Transactions on Communications*, Vol.67, No.3, pp. 2052 – 2070, 2019.

[33] Z. Xu, L. Zhou, S. Chau, W. Liang, Q. Xia and P. Zhou. Collaborate or separate? Distributed service caching in mobile edge clouds. To appear in *Proc of INFOCOM*, IEEE, 2020.

[34] Z. Yan, W. Zhou, S. Chen, and H. Liu. Modeling and analysis of two tier hetnets with cognitive small cells *IEEE Access*, vol. 5, pp. 2904–2912, Dec 2017.

[35] L. Yang, H. Zhang, M. Li, J. Guo, and H. Ji. Mobile edge computing empowered energy efficient task offloading in 5G. *IEEE Transactions on Vehicular Technology*, vol. 67, no. 7, pp.6398–6409, 2018.

[36] S. Bubeck and N. Cesa-Bianchi. Regret Analysis of Stochastic and Non-stochastic Multi-armed Bandit Problems *arXiv preprint arXiv:*,1204.5721, 2012.

[37] Y. Gai, B. Krishnamachari, and R. Jain. Combinatorial network optimization with unknown variables: multi-armed bandits with linear rewards and individual observations. *IEEE/ACM Transactions on Networking*, vol. 20, no. 5, pp. 1466–1478, 2012.

[38] X. Wang, C. Wu, F. Le, and F. C. Lau. Online learning-assisted VNF service chain scaling with network uncertainties. *Proc. of CLOUD*, IEEE, 2017.

[39] P. Rodrìguez, M. A. Bautista, J. Gonzàlez, and S. Escalera. Beyond One-hot Encoding: lower dimensional target embedding. `Image and Vision Computing`, vol. 75, pp. 21–31, Elsevier, 2018.

[40] X. Zhou, Z. Zhao, R. Li, Y. Zhou, T. Chen, Z. Niu and H. Zhang. Toward 5G: when explosive bursts meet soft cloud. *IEEE Network*, vol. 28, no. 6, pp 12–17, 2019.