An Online Learning-Based Task Offloading Framework for 5G Small Cell Networks

Xueying Zhang
School of Cyber Science and Engineering
Wuhan University
snowyzhang@whu.edu.cn

Ruiting Zhou
Wuhan University
The Chinese University of Hong Kong
ruitingzhou@whu.edu.cn

Zhi Zhou
School of Data and Computer Science
Sun Yat-sen University
hustzhouzhi@gmail.com

John C.S. Lui
Department of Computer Science and Engineering
The Chinese University of Hong Kong
cslui@cse.cuhk.edu.hk

Zongpeng Li
Huawei
Wuhan University
zongpeng@whu.edu.cn

ABSTRACT

Small cells are deployed in 5G networks to complement the macro cells for improving coverage and capacity. Small cells and edge computing are natural partners which can improve users' experience. Small cell nodes (SCNs) equipped with edge servers can support emerging computing services such as virtual reality which impose low-latency and precise contextual requirements. With the proliferation of wireless devices, there is an increasing demand for offloading tasks to SCNs. Given limited computation and communication resources, the fundamental problem for a small cell network is how to select computing tasks to maximize effective rewards in an uncertain and stochastic environment. To this end, we propose an online learning framework, LFSC, which has the performance guarantee to guide task offloading in a small cell network. LFSC balances between reward and constraint violations, and it consists of three subroutines: i) a randomized algorithm which calculates selection probability of each task based on task weights; ii) a greedy assignment algorithm which cooperatively allocates tasks among different SCNs based on the selection probability; iii) an update algorithm which exploits the multi-armed bandit (MAB) technique to update task weights according to the feedback. Our theoretical analysis shows that both the regret and violations metrics of LFSC have the sub-linear property. Extensive simulation studies based on real world data confirm that LFSC achieves a close-to-optimal reward with low violations, and outperforms many state-of-the-art algorithms.

CCS CONCEPTS

• **Network** → **Algorithms**; • **Theory of computation** → *Theory and algorithms for application domains.*

KEYWORDS

5G Network, Task Offloading, Online Learning

ACM Reference Format:
Xueying Zhang, Ruiting Zhou, Zhi Zhou, John C.S. Lui, and Zongpeng Li. 2020. An Online Learning-Based Task Offloading Framework for 5G Small Cell Networks. In *49th International Conference on Parallel Processing - ICPP (ICPP '20), August 17–20, 2020, Edmonton, AB, Canada.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3404397.3404417

1 INTRODUCTION

Explosive growth in mobile data traffic brings severe challenges to existing macrocell coverage. Small cells are introduced in 5G as a fundamental element of network densification. Small cell nodes (SCNs) operate in high frequencies, covering a range of 10 meters to 2 kilometers each [15]. SCNs are often attached to existing structures (*e.g.,* streetlights or utility poles) and connected to the core network (or the macrocell base station) through fiber optic cables. In particular, SCNs are close to wireless devices (WDs) and are able to process larger amount of data at faster speeds. A small cell network also increases the macrocell's capacity, and provides wireless users with better and faster connectivity [29]. The number of small cell installation in the US increased 550% in 2018, and is predicted to exceed $800,000$ by 2026 [2].

5G is expected to support transmission speed as high as 10 Gb/s [19], which boosts the demand for new services such as security surveillance, virtual reality, and automatic driving [32]. These applications usually generate a huge amount of data and are often delay sensitive, and thus such services are often prioritized to process at the edge rather than at the remote cloud due to strict delay and high computing requirements [3]. Therefore, edge computing and small cells are natural partners that may work in concert. Small cells equipped with edge servers represent a competitive solution for mobile task offloading. Since these servers are near to tasks' origin, so they can better meet the strict latency requirements. A major operator survey [1] shows that over 79% of operators will deploy small cells with edge computing before 2020 to support differentiated services for the potential market worth.

In this work, we consider how a small cell network can accept offloaded tasks from wireless devices (WDs). Operating such a system

ICPP '20, August 17–20, 2020, Edmonton, AB, Canada

ACM ISBN 978-1-4503-8816-0/20/08…$15.00
https://doi.org/10.1145/3404397.3404417

is very challenging. First, different types of tasks have different features, *e.g.,* input and output data size, latency requirement, etc. The above information is usually summarized as a task's context and leads to different allocation strategies. However, naively considering such large amount of contexts incurs high computation complexity. Second, both communication and computation resources at a SCN is limited. Due to the physical limitation of 5G high frequency bands, *e.g.,* Millimeter-Wave (mmWave) channel sparsity, beamforming technique, and number of radio frequency (RF) chains [23], each SCN can only establish a fixed number of connections to accept offloaded tasks. Furthermore, a lower-powered SCN may support only a small edge server with finite computation resource. Third, there exist significant uncertainties in the task offloading process. For example, 5G mmWave signals are prone to blockage due to weak diffraction capabilities [15]. Once blockage happens, the execution of a task is interrupted. Therefore, an efficient system needs to guarantee its quality of offloading service. In addition, the reward (*e.g.,* task value or computation rate) and resource consumption of a task may be time varying, depending on the quality of returned result. Worse yet, in practice, the aforementioned information can only be learned after the system offloads and processes tasks from WDs. Last but not least, a WD may be covered by multiple small cells. Collaborative task offloading between SCNs is non-trivial, because maximizing the reward at a single SCN does not always imply a global reward maximization. Therefore, *how to select offloading tasks in a small cell network such that the effective reward, i.e., total reward per unit resource, is maximized* is a fundamental and challenging problem.

Multi-armed bandit (MAB) is an online optimization method for learning an effective strategy in an unknown environment. Existing efforts on MAB are not directly applicable to this problem. They either focus on a single agent [24][20] or ignore system constraints [17][14]. Meanwhile, previous research on task offloading in edge computing [28] cannot capture all features in small cell networks, such as channel instability and collaborative offloading. We will discuss this in detail in Sec. 2. In this work, we propose an online learning-based framework, LFSC, to address the above challenges and guide the task offloading process in small cell networks. Our contributions are summarized as follows:

First, we formulate the task offloading problem in 5G small cell networks as an integer linear program (ILP). In the online setting, there is no prior knowledge on the relevant system parameters. We exploit MAB theory to deal with the unknown environment. We first relax the integral constraint, and consider it as the task selection probability. Rather than pursue reward maximization, our design aims to minimize the regret, which is the difference between the optimal reward and the average of our task offloading algorithm's reward. Our online learning algorithm, LFSC, makes a good balance between maximizing the overall effective reward (*i.e.,* minimizing the *regret*) and satisfying resource capacity constraint as well as QoS requirement (*i.e.,* keeping low *violations*).

Second, to tackle other challenges in the algorithm design, we leverage the following techniques: i) we introduce a series of adjustable penalty coefficients, using the Lagrangian method in constrained optimization [13], to balance between maximizing objective values and curbing constraint violations; ii) we divide the task context space into small hypercubes of similar contexts, and estimate the relevant parameters of each task. In addition, each hypercube maintains a weight, which is used to calculate the probability that each task will be offloaded in each time slot. Hence, the combined stage explosion and high computing complexity can be masterly avoided; iii) in view of the computational difficulty of coordinating all SCNs, a greedy algorithm is designed to conduct task offloading. While SCNs make a collaborative offloading decision, which can prevent a task from being repeatedly offloaded to multiple SCNs at the same time. LFSC consists of three subroutines: i) a randomized algorithm, trading off between *exploration* and *exploitation*, computes the selection probability of each task being offloaded to SCNs; ii) the greedy algorithm coordinates multiple SCNs for task offloading, based on the selection probability; iii) an update algorithm updates auxiliary variables based on feedback from current decision, which will help in calculating the selection probability in the next time slot.

Third, we prove the sub-linear upper bounds on the *regret* and *violations* of LFSC through rigorous theoretical analysis. We prove that LFSC converges to the optimal task offloading decision. Comparing with existing state-of-the-art, we further demonstrate LFSC's effectiveness by extensive simulations. The results show that LFSC significantly outperforms other benchmark algorithms. Under the same system settings, our algorithm's effective reward almost coincides with the optimal value. Furthermore, in the early stage of exploration, the total violations of LFSC are only 30%, 32% and 20% of the vUCB [5], FML [4] and random algorithm, respectively. Moreover, these percentages decrease over time.

The rest of the paper is organized as follows. Sec.2 reviews related literature. The small cell network is modeled in Sec. 3. The task offloading framework is presented in Sec. 4 and evaluated in Sec. 5. Sec. 6 concludes the paper.

2 RELATED WORK

Task Offloading. Previous studies on computation offloading focus on when/how/what to offload from user devices to the cloud or edge servers [28][30][12][25][27]. Sundar *et al.* [28] study the dependent task offloading problem. They make the assumption that the current state of the server and its performance of processing different tasks are known in advance. Xu *et al.* [30] study task offloading in an unknown dynamic system. Eshraghi *et al.* [12] propose an algorithm to jointly optimize the offloading decisions for minimizing a weighted sum of expected cost. They only consider one computing access point and a remote cloud center. Online learning algorithms based on MAB are proposed in [25][27] to help making task offloading desicions in the MEC environment.

Unfortunately, the above schemes are not well-suited for 5G networks with its special small cell architecture and unique properties such as the collaboration among multiple SCNs, and the communication limit of a SCN. Cheng *et al.* [10] investigate joint task offloading in 5G radio accessing networks, but not for small cell networks. An artificial fish swarm policy is developed in [31], which involves minimizing the overall energy consumption while offloading tasks in 5G. But tasks are offloaded to the macrocell base station rather than SCNs. In this paper, we consider offload workload to the 5G small cell networks in an unknown environment, and aim at maximizing effective reward under system constraints.

Multi-armed Bandit (MAB) Schemes. To address the uncertainty in 5G environments, we propose an online learning algorithm based on MAB, which was proven effective to balance between exploration and exploitation in sequential decisions [18]. The basic MAB framework learns to choose a single optimal arm among a set of candidate arms of *a priori* unknown rewards [6], without any constraints. Li *et al.* [20] take context-dependent rewards into account. Gai *et al.* [14] study multiple-play each time. Furthermore, Kim *et al.* [17] propose a contextual MAB algorithm for a relaxed, semi-parametric reward model. The above studies neglect system constraints, which are crucial to guarantee system QoS. Mahdavi *et al.* [21] extend the study of MAB where the learner aims to maximize the total reward, given that some additional constraints need to be satisfied. The arm's reward and cost in [21] are independent, while the objective value in our work is a compound reward, which involves learning multiple parameters. Cai *et al.* [7] propose an online learning framework using stochastic constrained bandit model with time-varying multi-level rewards based on MAB.

Note that offloading tasks in 5G small cell networks needs the collaboration of multiple SCNs, which implies there are multiple agents in the MAB framework. The above algorithms only work on the single agent and cannot apply to multi-agents case. Shahrampour *et al.* [26] address the MAB problem in a multi-agents framework, where all agents explore the same finite set of arms. While, in our work, each agent has a different set of arms and the sets change over time. An online distributed experts problem is studied to minimize the regret at time horizon in [16], where the system involves multiple sites (agents). However, the bandits are considered to be non-stochastic in this work. Nicol *et al.* [8] propose a global recommendation algorithm among multiple network nodes (*i.e.,* multi-agents) to improve the quality of recommendations, where only one node reveals its payoff (reward) in each round.

3 SYSTEM MODEL

3.1 System Overview

As shown in Fig. 1, we consider a small cell network where M small cell nodes (SCNs) are connected to a macrocell base station (MBS) via fiber optic cables. A set of wireless devices (WDs) are distributed in this small cell network, and each may request to offload a computing task. Each SCN is equipped with a computing server, which can process tasks from WDs. Let $\mathcal{T} = \{1, 2, \dots, T\}$ denote a large time span. M SCNs are denoted as $\mathcal{M} = \{1, 2, \dots, M\}$. Let $\mathcal{D}_t$ denote all tasks in time slot t and $\mathcal{D}_{m,t}$ denote the set of tasks that are within the coverage of SCN m in time slot t. We assume the maximum number of WDs that appear in SCN m's coverage area to be K_m, *i.e.,* $K_m = \max_{t \in \mathcal{T}} |D_{m,t}|$. Note that a WD may be covered by multiple small cells, and WDs are free to move from one cell to another in different time slots.

3.2 Task Offloading Problem

Task Context Information. In order to provide better performance, we consider the scenario where the MBS controls and prioritizes offloading task to SCNs. A computing task is characterized by the following meta information: i) the size of input data that needs to be transmitted from a WD to a SCN; ii) the size of output data that is to be fed back from a SCN to a WD; iii) the type of latency requirement (*e.g.,* tasks can be roughly classified into two categories: latency-sensitive, latency-insensitive), and so on. The above meta information of task i ($i \in \mathcal{D}_t$) is represented by its context ϕ_i.

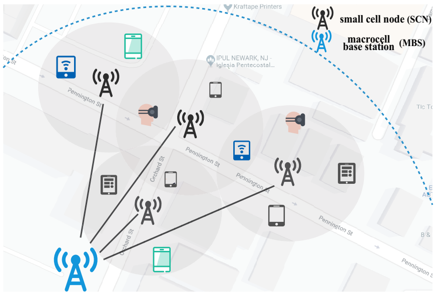

small cell node (SCN)
macrocell base station (MBS)
Kraftape Printers
IPUL NEWARK, NJ - Iglesia Pentecostal...
Pennington St
Orchard St

Figure 1: An illustration of a small cell network.

Random Process in Task Offloading. Consider three unknown random processes that capture the offloading scenario at SCN m, $U^m_\phi(t)$, $V^m_\phi(t)$ and $Q^m_\phi(t)$, where ϕ is the task's context. As a realization of $U^m_\phi(t)$, $u^{m,t}_{\phi_i}$ characterizes the reward for SCN m to complete task i with context ϕ_i at time t (*e.g.,* the value or computation rate of processing task i). Latency-sensitive tasks have higher rewards since they are eager to be completed as soon as possible. Let $v^{m,t}_{\phi_i}$ be the likelihood for SCN m to complete task i at t. This captures the unstable communication link between SCN m and the WD caused by weak penetration of 5G millimeter-Wave (mmWave). The variable $q^{m,t}_{\phi_i}$ characterizes resource consumption at SCN m while processing task i at t. Given that the environment and the resource consumption of a task are relatively stable in the long run, we assume that $V^m_\phi(t)$ and $Q^m_\phi(t)$ are stationary across contexts. The other random processes $U^m_\phi(t)$ are not necessarily stationary. They are all independent across ϕ and independent of each other. Without loss of generality, we normalize $U^m_\phi(t) \in [0, 1]$, $V^m_\phi(t) \in [0, 1]$ and $1/Q^m_\phi(t) \in [0, 1]$. Therefore, the effective reward per unit resource for SCN m to complete the task with context ϕ at time t is $G^m_\phi(t) = U^m_\phi(t)V^m_\phi(t)/Q^m_\phi(t)$. For the convenience of description, we use *compound reward* to refer to $G^m_\phi(t)$'s realization $g^{m,t}_{\phi_i}$. Let $\boldsymbol{g}_{m,t} = \{g^{m,t}_{\phi_i}\}_{i \in \mathcal{D}_{m,t}}$, similarly, we have $\boldsymbol{v}_{m,t}$ and $\boldsymbol{q}_{m,t}$.

System Constraints. 5G utilizes beamforming for mmWave communications between SCNs and WDs. Due to physical limitations such as RF chains, the number of beams emitted by each SCN is limited. Hence, SCN m cannot support all tasks in $\mathcal{D}_{m,t}$ if the number of requests is beyond its capacity. Let c denote the maximum number of tasks that each SCN can support at a time slot. Similarly, the computing resources (*i.e.,* RAM, CPU, GPU) at each SCN are also limited. The total resources utilized by all tasks at each SCN cannot exceed its resource capacity β at each time slot. Last but not least, in order to provide QoS guarantee, we also need to impose a requirement that the number of successfully processed tasks by each SCN at a time slot is at least α.

Decision Variable. Let a binary variable $p^{m,t}_i$ indicate whether SCN m executes task i at time t. Let $I_{m,t} \subseteq \mathcal{D}_{m,t}$ be the set of tasks selected by SCN m at t. Table I lists all notations.

Problem Formulation. We aim to maximize the total compound reward under system constraints. The optimization problem can be formulated as the following integer linear program (ILP):

$$\text{maximize} \sum_{t\in\mathcal{T}}\sum_{m\in\mathcal{M}}\sum_{i\in\mathcal{D}_{m,t}} g_{\phi_i}^{m,t} p_i^{m,t} \quad (1)$$

$$\text{subject to:} \quad \sum_{i\in\mathcal{D}_{m,t}} p_i^{m,t} \le c, \forall t\in\mathcal{T}, \forall m\in\mathcal{M}, \quad (1a)$$

$$\sum_{m\in\mathcal{M}} p_i^{m,t} \le 1, \forall t\in\mathcal{T}, \forall i\in\mathcal{D}_t, \quad (1b)$$

$$\sum_{i\in\mathcal{D}_{m,t}} v_{\phi_i}^{m,t} p_i^{m,t} \ge \alpha, \forall t\in\mathcal{T}, \forall m\in\mathcal{M}, \quad (1c)$$

$$\sum_{i\in\mathcal{D}_{m,t}} q_{\phi_i}^{m,t} p_i^{m,t} \le \beta, \forall t\in\mathcal{T}, \forall m\in\mathcal{M}, \quad (1d)$$

$$p_i^{m,t} \in \{0,1\}, \forall t\in\mathcal{T}, \forall m\in\mathcal{M}, \forall i\in\mathcal{D}_{m,t}. \quad (1e)$$

Constraint (1a) implies that the number of tasks accepted by each SCN does not exceed its communication capacity. Constraint (1b) guarantees that each task is not repetitively offloaded by multiple SCNs, which avoids wasting resources and improves the efficiency of overall system. System QoS requirement and resource capacity are modeled by (1c) and (1d), respectively.

Table 1. Summary of Notations

$\mathcal{M}$	set of M SCNs	$\mathcal{T}$	set of T time slots
$\mathcal{D}_t$	set of tasks at t	ϕ_i	the context of task i
Φ	context space	D_Φ	# of context dimensions
$I_{m,t}$	set of the tasks offloaded to SCN m at t		
$\mathcal{D}_{m,t}$	set of tasks in SCN m' coverage area at t		
K_m	maximum # of tasks in m's coverage		
F_T	set of hypercubes		
h_T	# of parts each dimension can be divided into		
$p_i^{m,t}$	selection probability for m to execute task i at t		
$u_{\phi_i}^{m,t}$	reward for SCN m to complete task i at t		
$v_{\phi_i}^{m,t}$	likelihood for SCN m to complete task i at t		
$q_{\phi_i}^{m,t}$	resource consumption while m processing i at t		
$g_{\phi_i}^{m,t}$	compound reward for m complete task i at t		
c	maximum # of tasks that each SCN can support		
α	minimum completed task threshold of each SCN		
β	computation resource capacity of each SCN		

Challenges. Existing online optimization literature assumes that the information of time slot t is known at the *beginning* of t. We consider a more practical scenario, where such information is not available. In particular, $u_{\phi_i}^{m,t}$, $v_{\phi_i}^{m,t}$ and $q_{\phi_i}^{m,t}$ can only be observed after a SCN processes a task. In this paper, we design a learning-based framework to allocate at most c tasks to each SCN. It is still challenging to maximize the compound reward without any prior knowledge. Hence, we relax the constraint of $p_i^{m,t}$. Let $\boldsymbol{p}_{m,t} = (p_1^{m,t}, p_2^{m,t}, \dots, p_{K_m}^{m,t})$ represent the task *selection probability vector* of SCN m at time t, where $p_i^{m,t} \in [0,1]$, $i \in \mathcal{D}_{m,t}$.[1] Let $\{\boldsymbol{p}_{m,t}^*\}_{m\in\mathcal{M}}$ denote the optimal solution. We quantify the performance of our algorithm by its *regret* value, which is defined as the difference between the omniscient *oracle*'s compound reward and the expectation of the LFSC's compound rewards. We assume that the oracle makes the best selection and achieves the optimal compound reward. The regret is defined as,

$$R(T) = \sum_{m\in\mathcal{M}} \left[\sum_{t\in\mathcal{T}} \boldsymbol{g}_{m,t}\boldsymbol{p}_{m,t}^* - \mathbb{E}\left(\sum_{t\in\mathcal{T}} \boldsymbol{g}_{m,t}\boldsymbol{p}_{m,t}\right)\right]. \quad (2)$$

In addition to maximizing the total compound rewards (minimizing the regret), (1c) and (1d) should also be guaranteed. To measure the overall violations of the constraints until time T, we define two *violations* under the LFSC framework,

$$V_1(T) = \sum_{m\in\mathcal{M}} \mathbb{E}\left[\sum_{t\in\mathcal{T}} (\alpha - \boldsymbol{v}_{m,t}\boldsymbol{p}_{m,t})\right]_+, \quad (3)$$

$$V_2(T) = \sum_{m\in\mathcal{M}} \mathbb{E}\left[\sum_{t\in\mathcal{T}} (\boldsymbol{q}_{m,t}\boldsymbol{p}_{m,t} - \beta)\right]_+, \quad (4)$$

where $[\cdot]_+ = \max(\cdot, 0)$. $V_1(T)$ shows the overall difference between the total expected number of completed tasks and the minimum completed task threshold. Similarly, $V_2(T)$ measures how much the overall expected resource consumption exceeds the resource capacity. The regret and the above two violations are important metrics to measure the performance of offloading tasks selection. A good algorithm should reduce both the regret and violations, and it learns more information about the environment.

3.3 Discussion

Since SCNs are deployed closer to WDs than MBS, they can provide low-lantency services and have higher priority in task offloading. For those tasks that are not selected by SCNs, they can be offloaded and processed by MBS. In addition, we assume all tasks can be processed in one time slot. If some tasks need to execute over multiple slots, they can keep submitting offloading requests in the subsequent time slots. In future work, we will consider the case where a task's reward is obtained only after full execution.

4 ALGORITHM DESIGN AND ANALYSIS

4.1 Challenges and Solutions

In the general MAB framework, the learner simply aims to maximize the total reward (*i.e.*, or minimize the regret) without taking any constraints into account, which does not apply to our model. Enlightened by [7], we leverage the theory of Lagrangian method in constrained combinatorial optimization to **balance** between maximizing the total compound reward and satisfying the system constraints. Specifically, we introduce a set of adjustable Lagrangian multipliers $\lambda_1^m(T)$ and $\lambda_2^m(T)$ for each SCN m, and combine the regret with violations to construct a new regret function:

$$Y = \sum_{m\in\mathcal{M}} [R_m(T) + \lambda_1^m(T)(V_{m,1}(T))^2 + \lambda_2^m(T)(V_{m,2}(T))^2],$$

where the Lagrangian multipliers play a regulatory role and $R_m(T)$, $V_{m,1}(T)$ and $V_{m,2}(T)$ denote the regret and violations of SCN m at time slot t, respectively. If constraints are being violated a lot, LFSC places more weight on the violations controlled by $\lambda_1^m(T)$ ($\lambda_2^m(T)$); it decreases the weight on violations when constraints are satisfied reasonably. Note that, our algorithm allows violations to happen in some time slots, but the constraints must hold in the long term. Now our goal is to obtain a sub-linear bound for Y, *i.e.*, $Y \le T^{1-\theta}(0 < \theta < 1)$, and thus we can further derive sub-linear bounds for both regret and violations in long terms.

[1] With the exception of $\boldsymbol{p}_{m,t}$ being a column vector, all other vectors in this paper are row vectors.

Algorithm 1 An Online Learning Framework (**LFSC**)

```
Initialize context partition: divide context space $\Phi$ into $(h_T)^{D_\Phi}$ hypercubes of identical size
Initialize partitions weight: $w_f^{m,1} = 1$ for $f \in \mathcal{F}_T$ for $m \in \mathcal{M}$
Initialize auxiliary variables: $\lambda_{m,1}^1 = 0$, $\lambda_{m,2}^1 = 0$, $\alpha > 0$, $\beta > 0$, $\gamma_m \in (0, 1]$, $\delta_m = \frac{8\gamma_m c}{1-\gamma_m}$, $\eta_m = \frac{\gamma_m \delta_m c}{(\delta_m + c)K_m}$ for $m \in \mathcal{M}$
for $t = 1, \cdots, T$ do
    for $m \in \mathcal{M}$ do
        $\tilde{\boldsymbol{p}}_m^t$ = Calculating($\{w_f^{m,t}\}_{f \in \mathcal{F}_T}$)
    end for
    $\{\mathcal{I}_m^t\}_{m \in \mathcal{M}}$ = GreedySelect($c$, $\{\tilde{\boldsymbol{p}}_m^t\}_{m \in \mathcal{M}}$)
    for $m \in \mathcal{M}$ do
        $\{w_f^{m,t+1}\}_{f \in \mathcal{F}_T}$ = Updating($\{w_f^{m,t}\}_{f \in \mathcal{F}_T}$, $\mathcal{I}_m^t$, $\tilde{\boldsymbol{p}}_m^t$)
    end for
end for
```

Algorithm 2 Calculate Chosen Probability Vectors ***Calculating***

```
Input: $\{w_f^{m,t}\}_{f \in \mathcal{F}_T}$
1:  Observe SCN $m$'s current neighbor tasks $\mathcal{D}_{m,t}$
2:  Observe tasks' contexts $\boldsymbol{\phi}_{m,t} = \{\phi_{i,t}\}_{i \in \mathcal{D}_{m,t}}$
3:  for $i \in \mathcal{D}_{m,t}$ do
4:      Find $\boldsymbol{f}_t = \{f_{i,t}\}_{i \in \mathcal{D}_{m,t}}$ such that $\phi_{i,t} \in f_{i,t} \in \mathcal{F}_T$
5:  end for
6:  if $\arg\max_{j \in \mathcal{F}_T} w_j^{m,t} \geq (\frac{1}{c} - \frac{\gamma}{|\mathcal{F}_T|})/(1-\gamma)\sum_{f \in \mathcal{F}_T} w_f^{m,t}$ then
7:      Decide $\epsilon_t$ so as to satisfy
8:      $\frac{\epsilon_t}{\sum_{w_f^{m,t} \geq \epsilon_t} \epsilon_t + \sum_{w_f^{m,t} < \epsilon_t} w_f^{m,t}} = (\frac{1}{c} - \frac{\gamma}{|\mathcal{F}_T|})/(1-\gamma)$
9:      Set $S^t = \{f : w_f^{m,t} \geq \epsilon_t\}$ and $\tilde{w}_f^{m,t} = \epsilon_t$ for $f \in S^t$
10: else
11:     Set $S^t = \emptyset$
12: end if
13: Set $\tilde{w}_f^{m,t} = w_f^{m,t}$ for $f \in \mathcal{F}_T \backslash S^t$
14: Set $\tilde{w}_i^{m,t} = \tilde{w}_{f_i}^{m,t}$
15: for $i \in \mathcal{D}_{m,t}$ do
16:     $\tilde{p}_i^{m,t} = c[(1-\gamma)\frac{\tilde{w}_i^{m,t}}{\sum_{i \in \mathcal{D}_{m,t}} \tilde{w}_i^{m,t}} + \frac{\gamma}{|\mathcal{D}_{m,t}|}]$
17: end for
18: Return: $\tilde{\boldsymbol{p}}_m^t$
```

The **core** issue to be solved is how the MBS selects c offloading tasks for each SCN based on historical knowledge. A straightforward approach is to enumerate all possible sets and select the optimal one. Unfortunately, this leads to a very large search space. Furthermore, the compound reward can only be observed after task completion. In order to avoid this combinatorial explosion, the traditional way is to keep a series of weights for each SCN's all task contexts [9]. According to these weights, the selection probability vectors are calculated in each time slot. Nevertheless, each task comes with its context, which means there are massive contexts to be learned. If we maintain a weight for each context, it will have high computational complexity. Hence, we use a basic hypothesis that for similar task contexts, their feedback by a particular SCN will be similar. Under this hypothesis, our algorithm uniformly partitions the context space into small hypercubes of similar task contexts and maintains a weight for each hypercube. Meanwhile, the algorithm learns about the parameters of different hypercubes, which can be considered as approximate estimates of the parameters for contexts belonged to it.

Finally, since a task can be assigned to multiple SCNs, we need to consider the collaboration between different SCNs. If we extend the traditional single agent MAB approach to our setting, there are two key obstacles: i) the tasks covered by multiple SCNs may be repeatedly offloaded, which causes unnecessary waste of computing resources; ii) *cascade sub-optimality* will occur when a SCN selects a sub-optimal task i since its optimal task i^* has already been offloaded to another SCN. This sub-optimal selection has the potential to bring cascade effect. In other words, local optimality at a single SCN does not always result in the global optimum. Therefore, we design a greedy algorithm that maps a task to a SCN with the maximum reward to solve this challenge.

4.2 Algorithm Details

The framework of our algorithm, LFSC, is shown in Alg. 1. At a high level, the algorithm consists of two parts: i) a tailored contextual MAB algorithm, which balances between *exploration* and *exploitation* in each time slot to learn parameters such that a close-to-optimal performance can be achieved; ii) a greedy assignment algorithm, which gives a collaborative task offloading solution among all SCNs.

Let Φ be the D_Φ-dimensional context space, where D_Φ is the number of context dimensions per task. We assume it is bounded and can hence be set to $\Phi := [0, 1]^{D_\Phi}$ without loss of generality. First, during initialization, LFSC uniformly partitions the context space Φ into $(h_T)^{D_\Phi}$ hypercubes (*i.e.,* each hypercube has the same size $(\frac{1}{h_T})^{D_\Phi}$), where h_T is an input to our algorithm. Let $\mathcal{F}_T$ be the resulting partition. Then LFSC initializes a set of weights for the hypercubes of each SCN, which will be updated according to historical observations. In each time slot, LFSC calculates the selection probability vector for each SCN towards current tasks within its coverage, as shown in Alg. 2. Next, the greedy assignment algorithm in Alg. 4 assigns tasks to SCNs based on selection probability vectors. Finally, in Alg. 3, each SCN accepts tasks according to the assignment. After receiving the feedback ($i.e.$, $u_{\phi_i}^{m,t}$, $v_{\phi_i}^{m,t}$ and $q_{\phi_i}^{m,t}$) of the processed tasks, LFSC updates estimated parameters and hypercubes' weights as well as some auxiliary variables for each SCN, which will be used in the next time slot to help learning.

Tailored Contextual MAB Algorithm. The algorithm consists of Alg. 2 and Alg. 3. In each iteration, LFSC first gets the contexts of all tasks within SCN m's coverage and classifies them into corresponding hypercubes (Lines 1-5 in Alg. 2). Then, our algorithm preprocesses the weights of hypercubes and determines each task's weight (Lines 6-14 in Alg. 2), which are used to calculate the selection probability vector $\tilde{\boldsymbol{p}}_m^t$ (Lines 15-17 in Alg. 2). Note that the first and second terms in the RHS of Line 16 reflect the trade-off between exploitation and exploration. After each SCN has processed tasks according to the greedy assignment approach, MBS receives their feedback and calculates compound rewards for them (Line 1 in Alg. 3). In Lines 2-5, Alg. 3 calculates the unbiased estimates $\hat{g}_i^{m,t}$, $\hat{v}_i^{m,t}$, $\hat{q}_i^{m,t}$ for each task, where $\mathbb{1}(A)$ is an indicator function, $i.e.$, $\mathbb{1}(A) = 1$ if the event A happens and $\mathbb{1}(A) = 0$ otherwise. Next, in Line 8, the estimated compound reward $\hat{g}_f^{m,t}$ of the hypercube f

Algorithm 3 Update Auxiliary Variables ***Updating***

```
Input: $\{w_f^{m,t}\}_{f\in\mathcal{F}_T}$, $\mathcal{I}_m^t$, $\tilde{\boldsymbol{p}}_m^t$
1:  Receive feedback $u_i^{m,t}$, $v_i^{m,t}$, $q_i^{m,t}$ and calculate compound reward $g_i^{m,t}$ for $i \in \mathcal{I}_m^t$
2:  for $i \in \mathcal{D}_{m,t}$ do
3:      $\hat{g}_i^{m,t} = g_i^{m,t}/\tilde{p}_i^{m,t}\mathbb{1}(i \in \mathcal{I}_m^t)$
4:      $\hat{v}_i^{m,t} = v_i^{m,t}/\tilde{p}_i^{m,t}\mathbb{1}(i \in \mathcal{I}_m^t)$
5:      $\hat{q}_i^{m,t} = q_i^{m,t}/\tilde{p}_i^{m,t}\mathbb{1}(i \in \mathcal{I}_m^t)$
6:  end for
7:  for $f \in \mathcal{F}$ do
8:      Calculate hypercubes' compound reward and parameters $\hat{g}_f^{m,t}$, $\hat{v}_f^{m,t}$ and $\hat{q}_f^{m,t}$
9:      if $f \notin \mathcal{S}^t$ then
10:         $w_f^{m,t+1} = w_f^{m,t} \exp[\eta_m(\hat{g}_f^{m,t} + \lambda_{m,1}^t\hat{v}_f^{m,t} + \lambda_{m,2}^t\hat{q}_f^{m,t})]$
11:     else
12:         $w_f^{m,t+1} = w_f^{m,t}$
13:     end if
14: end for
15: Update Lagrange multipliers:
16: $\lambda_{m,1}^{t+1} = [(1-\delta_m\eta_m)\lambda_{m,1}^t - \eta_m(\frac{\hat{\boldsymbol{v}}_t\tilde{\boldsymbol{p}}^t}{1-\gamma_m} - \alpha)]_+$
17: $\lambda_{m,2}^{t+1} = [(1-\delta_m\eta_m)\lambda_{m,2}^t - \eta_m(\beta - \frac{\hat{\boldsymbol{q}}_t\tilde{\boldsymbol{p}}^t}{1-\gamma_m})]_+$
18: Return: $\{w_f^{m,t+1}\}_{f\in\mathcal{F}_T}$
```

is calculated according to $\hat{g}_f^{m,t} = \sum_{i:\phi_{i,t}\in f}\hat{g}_i^{m,t}/\sum_{i:\phi_{i,t}\in f}$. Similarly, our algorithm computes $\hat{v}_f^{m,t}$ and $\hat{q}_f^{m,t}$. Finally, Lines 9-17 in Alg. 3 update the weights of all hypercubes and Lagrange multipliers of each SCN at the end of each iteration.

Greedy Assignment Algorithm. We design a greedy assignment algorithm to give a collaborative task offloading solution among all SCNs. According to our system model, we abstract a weighted bipartite graph $\mathcal{G} = (\mathcal{M}, \mathcal{D}_t, E)$, where $\mathcal{M}$, $\mathcal{D}_t$ and E represent left vertices (*i.e.,* all SCNs), right vertices (*i.e.,* all tasks at time slot t) and edges, respectively. If task i is within the coverage of SCN m at time slot t ($i.e.$, $i \in \mathcal{D}_{m,t}$), there is a weighted edge between them, which is denoted as $w(m, i)$. Hence, $E \triangleq \{w(m, i)\}$. In particular, we set $w(m, i) = \tilde{p}_i^{m,t}$ after all SCNs' selection probability vectors are computed in Alg. 2.

Algorithm 4 Greedy Assignment Algorithm ***GreedySelect***

```
Input: $c$, $\{w(m, i)\}$
    Initialize: $\Omega = \emptyset$, $E' = \{w(m, i)\}$, $C(m) = 0$ for $m \in \mathcal{M}$
1:  while $E' \neq \emptyset$ do
2:      select $(m, i) = \arg\max_{(m',i')\in E'} w(m', i')$
3:      if $C(m) < c$ then
4:          $\Omega = \Omega \cup \{(m, i)\}$
5:          $C(m) = C(m) + 1$
6:          $E' = E'\backslash\{(m', i')\}\forall(m', i') : i' = i$
7:      else
8:          $E' = E'\backslash\{(m, i)\}$
9:      end if
10: end while
11: Return: $\Omega$
```

The greedy algorithm operates in an iterative fashion. Let $C(m)$ denote the number of tasks that will be offloaded to SCN m until now, which is initialized to zero. In each iteration, the highest weight edge (m, i) is selected until there is no edge in E' (Line 2 in Alg. 4). If the number of selected tasks for SCN m is less than c, task i will be selected and offloaded to SCN m. Meanwhile, we update $C(m)$ and the set of available edges E' (Lines 3-6). Otherwise, this edge is deleted from E'. Finally, we get the task offloading scheme Ω after all iterations.

4.3 Regret and Violation Analysis

Now we establish the upper bounds on the regret $R(T)$ and violations $V_1(T)$, $V_2(T)$ of our online learning algorithm LFSC. The theorem below states that the regret and violations of our algorithm are all *sub-linear* with respect to T, which means LFSC converges to the optimal task offloading decisions over time, and has an asymptotically optimal performance when T is sufficiently large.

Theorem 1. *Let* $\eta_m = \frac{\gamma_m\delta_m c}{(\delta_m+c)K_m}$, $\delta_m = \frac{8\gamma_m c}{1-\gamma_m}$ *and* $\gamma = \min(1, \sqrt{\frac{2K_m(1+c)}{c\ln(K_m/c)T^{2/3}}})$, *we achieve its sub-linear bounds for the regret* $R(T)$ *and violations* $V_1(T)$, $V_2(T)$ *as follows*:

$$R(T) \leq O[T^{\frac{2}{3}-\frac{\sigma}{D_\Phi}}LD_\Phi^{\frac{\sigma}{2}}c(c+1)\sum_{m\in\mathcal{M}}(K_m\ln K_m)],$$

$$V_1(T)(V_2(T)) \leq O(T^{\frac{5}{6}}L^{\frac{1}{2}}D_\Phi^{\frac{\sigma}{4}}c^{\frac{3}{2}}\sum_{m\in\mathcal{M}}K_m^{\frac{1}{2}}).$$

To prove Theorem 1, we first decompose the multi-agent problem into multiple single agent problems and analyze the sub-problems. Let $R^m(T)$, $V_1^m(T)$, $V_2^m(T)$ respectively denote the regret and two violations of SCN m when the all tasks within SCN m's coverage are available to it, and there exist no conflicts with other SCNs.

Lemma 1. *For a SCN m, we can establish its sub-linear bounds for its regret* $R^m(T)$ *and violations* $V_1^m(T)$, $V_2^m(T)$ *as follows:*

$$R^m(T) \leq O(LD_\Phi^{\frac{\sigma}{2}}cK_m\ln K_mT^{\frac{2}{3}-\frac{\sigma}{D_\Phi}}), \tag{5}$$

$$V_1^m(T)(V_2^m(T)) \leq O(L^{\frac{1}{2}}D_\Phi^{\frac{\sigma}{4}}c^{\frac{1}{2}}K_m^{\frac{1}{2}}T^{\frac{5}{6}}). \tag{6}$$

Proof. Please refer to Appendix. A.1. □

Next, we analyze the performance of our greedy algorithm in Alg. 4.

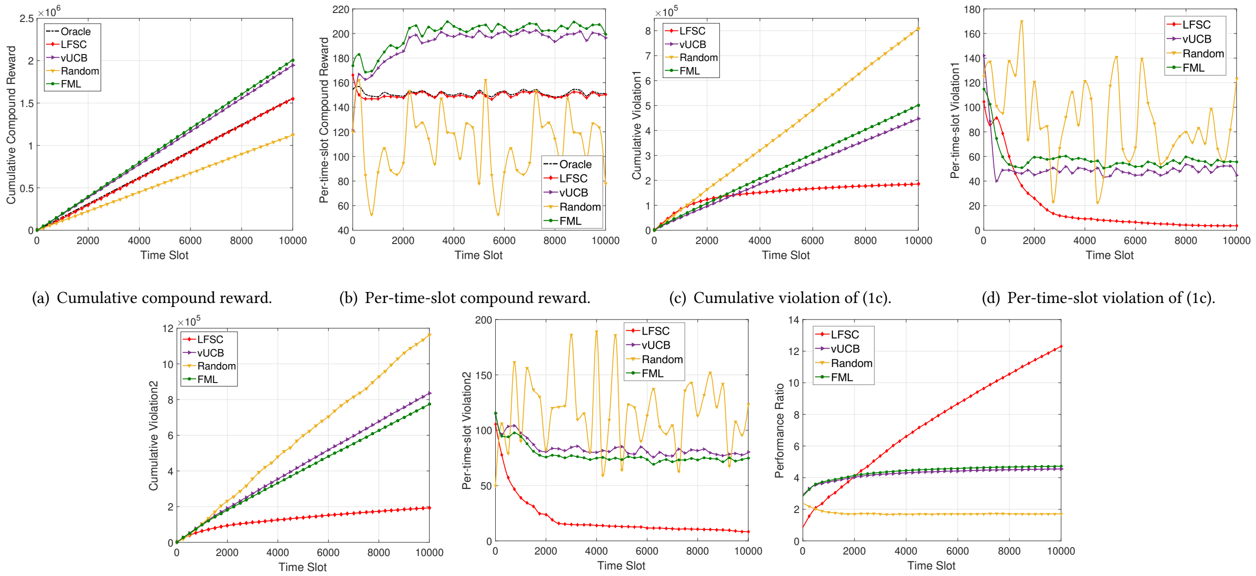
Lemma 2. *For any given weighted bipartite graph instance* $\mathcal{G} = (\mathcal{M}, \mathcal{D}_t, \{w(m, i)\})$, *let* Ω^* *and* Ω *denote the optimal solution and the output of our greedy algorithm, respectively. Then,* $\sum_{(m,i)\in\Omega} g_{\phi_i}^{m,t}w(m, i) \geq \frac{1}{c+1}\sum_{(m,i)\in\Omega^*} g_{\phi_i}^{m,t}w(m, i)$, *where* $w(m, i) = \tilde{p}_i^{m,t}$ *in our setting and* $g_{\phi_i}^{m,t}w(m, i)$ *correspondingly represents the objective value in ILP (1).*

Proof. Please refer to Appendix. 1.2. □

Finally, combining Lemma 1 and Lemma 2, we derive the upper bounds for the regret and violations of LFSC. Let $\boldsymbol{p}_{m,t}^{**}$ denote the optimal solution for SCN m without considering conflicts. Note that $\boldsymbol{p}_{m,t}^{*}$ is the optimal solution for SCN m, which takes into account the conflicts among multiple SCNs. Since the conflicts among SCNs are ignored in Lemma 1, we have $\sum_{t\in\mathcal{T}}\boldsymbol{g}_{m,t}\boldsymbol{p}_{m,t}^{*} \leq \sum_{t\in\mathcal{T}}\boldsymbol{g}_{m,t}\boldsymbol{p}_{m,t}^{**}$ for $m \in \mathcal{M}$, and we obtain,

$$\begin{aligned} R(T) &= \sum_{m\in\mathcal{M}}[\sum_{t\in\mathcal{T}}\boldsymbol{g}_{m,t}\boldsymbol{p}_{m,t}^{*} - \mathbb{E}(\sum_{t\in\mathcal{T}}\boldsymbol{g}_{m,t}\tilde{\boldsymbol{p}}_{m,t})] \\ &\leq (c+1)[\sum_{m\in\mathcal{M}}\sum_{t\in\mathcal{T}}\boldsymbol{g}_{m,t}\boldsymbol{p}_{m,t}^{**} - \mathbb{E}\sum_{t\in\mathcal{T}}\boldsymbol{g}_{m,t}\tilde{\boldsymbol{p}}_{m,t}]. \end{aligned}$$

Therefore, we obtain:



(a) Cumulative compound reward.

(b) Per-time-slot compound reward.

(c) Cumulative violation of (1c).

(d) Per-time-slot violation of (1c).

(e) Cumulative violation of (1d).

(f) Per-time-slot violation of (1d).

(g) Performance ratio

Figure 2: Compound rewards, violations and performance ratio of LFSC, Oracle, vUCB, FML and Random.

$$R(T) \le O[T^{\frac{2}{3}-\frac{\sigma}{D_\Phi}} L D_\Phi^{\frac{\sigma}{2}} c(c+1) \sum_{m\in\mathcal{M}} (K_m \ln K_m)],$$

$$V_1(T)(V_2(T)) \le O(T^{\frac{5}{6}} L^{\frac{1}{2}} D_\Phi^{\frac{\sigma}{4}} c^{\frac{3}{2}} \sum_{m\in\mathcal{M}} K_m^{\frac{1}{2}}),$$

which shows that both regret and violations are sub-linear in the time horizon T. □

5 PERFORMANCE EVALUATION

We next demonstrate the performance of LFSC via numerical simulations. We first describe the simulation settings. Then the benchmark algorithms and numerical results are presented.

Simulation Setup. We consider a scenario where there are 30 SCNs connected to a MBS. Suppose the number of WDs appearing in each SCN's coverage area varies randomly in interval [35, 100] in each time slot, *i.e.,* $|\mathcal{D}_{m,t}| \in [35, 100]$. In a time slot, each SCN can simultaneously support up to 20 WDs. For simplicity, we only consider the input and output data size of tasks, as well as the type of computation resources they depend on (*i.e.,* CPU, GPU, or both CPU and GPU). The input data size of tasks is randomly distributed between 5 Mbit and 20 Mbit [22]. Similarly, the output is between 1 Mbit and 4 Mbit. Then we divide the input/output data size into three categories by default. The reward and likelihood of a SCN completing a task are normalized and uniformly distributed in [0, 1]. And the resource consumption that a SCN processes a task is uniformly distributed in [1, 2] [25]. To guarantee the performance of the network, the minimum completed task threshold α and computation resource limit β of each SCN are set to 15 and 27, respectively.

Benchmark Algorithms. To evaluate the performance of our algorithm, we provide a thorough analysis by comparing LFSC with the following benchmark schemes:

- *Oracle*: Oracle has *a priori* knowledge of the entire system. In each time slot, Oracle makes the best task offloading policy under the system constraints, and it constitutes a performance upper bound to the other algorithms.
- *Variant-UCB (vUCB)*: This is a variant of the classic learning algorithm UCB [5], which we adapt to our use-case. To fit our model, vUCB maintains a series indices $\bar{g}_f^t + \sqrt{2\ln(t)/(N_f(t))}$ for our hypercubes for each SCN, where $\bar{g}_f^t$ is the estimated compound reward of hypercube f, and $N_f(t)$ is the total number of times that tasks with context in hypercube f has been selected before time t. Then our greedy algorithm Alg. 4 is used to guide task offloading among multiple SCNs based on the indices.
- *FML*: Fast Machine Learning (FML) [4] is an efficient context-aware online learning algorithm. Since FML only considers a single agent, it is slightly modified to fit our system model. Specifically, our greedy algorithm is added to handle multi-agents problem (*i.e.*, multiple SCNs in our paper).
- *Random*: This algorithm randomly picks c tasks for each SCN in each time slot, and each task cannot be repeatedly offloaded.

Performance Metrics. Our performance metrics consist of cumulative (per-time-slot) compound reward, cumulative (per-time-slot) violations of (1c) and (1d) in ILP (1) and *performance ratio*. The cumulative compound reward (violation) is the overall compound reward of all SCNs in the system up to time slot t. And the per-time-slot compound reward (violation) at t is the compound reward (violation) of all SCNs in time slot t. In particular, we define the performance ratio as: *performance ratio=cumulative compound reward / (cumulative violation1 + cumulative violation2)*, which shows the ratio between total reward and violations.

Result Analysis. We run simulations with $T = 10,000$. As can be seen in Fig. 2(a), the cumulative compound reward of LFSC is almost identical to that of the Oracle at each time slot. To get more details, as shown in Fig. 2(b), the per-time-slot compound reward of LFSC is slightly larger than that of the Oracle in the first few time slots ($t \leq 74$). It is because LFSC is in unknown environment at the beginning. It may offload the tasks that have large compound rewards but violate the system constraints. As t increases, LFSC is in the exploration stage and learns from history observation. Thus, its per-time-slot compound reward is decreasing and smaller than Oracle's. After that, the compound reward becomes closer to the value of Oracle, which means LFSC becomes more accurate in estimating the system parameters. Note the per-time-slot compound reward of the Oracle is varying since the number of incoming tasks and their contexts may be different in each time slot. However, the cumulative compound rewards and per-time slot compound rewards of vUCB and FML are always larger than the values of our LFSC and the Oracle. This is because these two algorithm select tasks with large compound reward regardless of the minimum completed task threshold and computation resource limit.

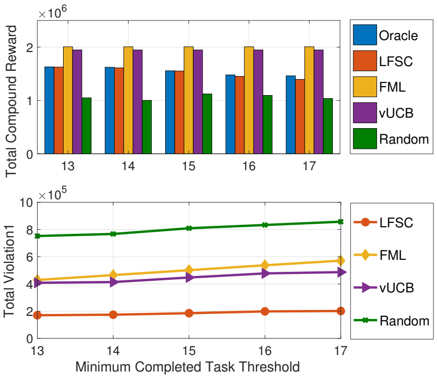


Figure 3: Total compound reward and violation of (1c) under different value of α.

Fig. 2(c) shows that for each time slot, LFSC's cumulative violation of minimum completed task threshold is always the lowest. LFSC selects and processes tasks that are less likely to violate the minimum completed task threshold. Moreover, the violations of LFSC occur in the exploration stage, and the growth rate of later cumulative violations tends to zero. This can also be verified in Fig. 2(d). After $t = 1200$, the per-time-slot violation-1 of vUCB and FML fluctuates within a certain range. In contrast, the value of LFSC gradually decreases and asymptotically approaches zero. Since the Oracle can make task offloading decisions without any violation, it is not depicted in Fig. 2(c) and Fig. 2(d). Similarly, the cumulative/per-time-slot violation of computation resource limit are shown in Fig. 2(e) and Fig. 2(f). They show the superiority of LFSC in meeting resource capacity constraints. In Fig. 2(g), after $t = 2750$, LFSC has a significantly better performance ratio compared to vUB and FML algorithms. It strikes a balance between gleaning reward and curbing violations.

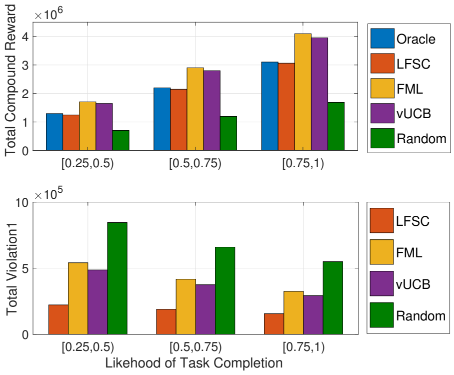


Figure 4: The impact of the likelihood of task completion

Next, we investigate the impact of the minimum completed task threshold α on the total compound reward and violation of (1c). As before, we run the simulation with $T = 10,000$. Fig. 3 shows the total compound rewards and violations of algorithms for different $\alpha \in \{13, 14, 15, 16, 17\}$. As the value of α increases, the total compound reward of LFSC decreases. Nonetheless, it is still the closest to the value of Oracle. Note the total compound rewards of vUCB and FML have not changed because the way they make task offloading decisions is not affected by the value of α. The total violations of all algorithms increase with the increase of α, yet, the value of LFSC increases more slowly.

Finally, we observe the performance of LFSC in different environments where the range of the likelihood that a task is successfully offloaded is different. The likelihood of task completion is divided into three intervals, *i.e.*, $[0.25, 0.5)$, $[0.5, 0.75)$ and $[0.75, 1]$. From Fig. 4, we can find that it is easier to satisfy the minimum completed task threshold, and all algorithms' violations of constraint (1c) in ILP (1) are decreasing as the likelihood increases, when the value of α is fixed. The likelihood is smaller, *i.e.,* the minimum threshold is more difficult to satisfy, the superiority of our LFSC algorithm over vUCB and FML is more prominent.

6 CONCLUSION

We studied task offloading in 5G small cell networks, and proposed an online learning-based solution framework. Our algorithm, LFSC, leverages the MAB technique to learn the best task selection strategy in a small cell network, while considering resource capacity constraints and QoS requirement. The efficiency of LFSC is verified by both theoretical analysis and simulation studies. We proved that LFSC achieves sub-linear bounds for both regret and violations. Our algorithm's superiority over other benchmark algorithms is also confirmed by large-scale evaluations based on real-world data.

For future work, it is interesting to jointly consider offloading tasks to MBS and SCNs. Tasks that do not restrict the latency but consume large amount of computing resources will be offloaded to MBS. As mentioned in Sec. 3, another question is how to guarantee full execution of tasks, in which the execution time extends to multiple time slots. A possible solution is to assign an extra reward for processed tasks, such that they have the priority in future offloading decisions.

ACKNOWLEDGEMENTS

The corresponding author is Ruiting Zhou. This work was supported by the Fundamental Research Funds for the Central Universities (2042019kf0016), National Science Foundation of China under Grants (61802449) and the GRF 14201819.

REFERENCES

[1] [n.d.]. *Small cells and edge compute make a natural partnership.* https://www.smallcellforum.org/blog/small-cells-and-edge-compute-make-a-natural-partnership/.

[2] [n.d.]. *What is a Small Cell? A Brief Explainer.* https://www.ctia.org/news/what-is-a-small-cell.

[3] [n.d.]. *Why Edge Computing is Key to a 5G Future.* https://datamakespossible.westerndigital.com/edge-computing-key-5g-future/.

[4] A. Asadi, S. Müller, G. H. Sim, A. Klein, and M. Hollick. 2018. FML: Fast Machine Learning for 5G mmWave Vehicular Communications. In *Proc. of IEEE INFOCOM.*

[5] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. 2002. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning* 47, 2 (2002), 235–256.

[6] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. 2002. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning* 47, 2-3 (2002), 235–256.

[7] K. Cai, X. Liu, Y. J. Chen, and J. C. S. Lui. 2018. An Online Learning Approach to Network Application Optimization with Guarantee. In *Proc. of IEEE INFOCOM.*

[8] Nicolò Cesa-Bianchi, Claudio Gentile, and Giovanni Zappella. 2013. A Gang of Bandits. In *Proc. of NIPS.*

[9] Lixing Chen and Jie Xu. 2019. Task Replication for Vehicular Cloud: Contextual Combinatorial Bandit with Delayed Feedback. In *Proc. of IEEE INFOCOM.*

[10] Zhipeng Cheng, Yuliang Tang, and Haijie Wu. 2019. Joint Task Offloading and Flexible Functional Split in 5G Radio Access Network. In *Proc. of IEEE ICOIN.*

[11] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2016. *Introduction to Algorithms* (3rd ed.). The MIT Press.

[12] N. Eshraghi and B. Liang. 2019. Joint Offloading Decision and Resource Allocation with Uncertain Task Computing Requirement. In *Proc. of IEEE INFOCOM.*

[13] Marshall L. Fisher. 1981. The Lagrangian Relaxation Method for Solving Integer Programming Problems. *Management Science* 27, 1 (1981), 1–18.

[14] Yi Gai, Bhaskar Krishnamachari, and Rahul Jain. 2012. Combinatorial Network Optimization With Unknown Variables: Multi-Armed Bandits With Linear Rewards and Individual Observations. *IEEE/ACM Trans. Netw.* 20, 5 (2012), 1466–1478.

[15] L.T. Hwang and T.S.J. Horng. 2018. *3D IC and RF SiPs: Advanced Stacking and Planar Solutions for 5G Mobility.* Wiley.

[16] Varun Kanade, Zhenming Liu, and Bozidar Radunovic. 2012. Distributed Non-Stochastic Experts. In *Proc. of NIPS.*

[17] Gi-Soo Kim and Myunghee Cho Paik. 2019. Contextual Multi-armed Bandit Algorithm for Semiparametric Reward Model. In *Pro. of ACM ICML.*

[18] T.L Lai and Herbert Robbins. 1985. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics* 6, 1 (1985), 4 – 22.

[19] B. Lannoo, A. Dixit, D. Colle, J. Bauwelinck, B. Dhoedt, B. Jooris, I. Moerman, M. Pickavet, H. Rogier, P. Simoens, G. Torfs, D. Vande Ginste, and P. Demeester. 2015. Radio-over-fibre for ultra-small 5G cells. In *Proc. of IEEE ICTON.*

[20] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proc. of ACM WWW.*

[21] Mehrdad Mahdavi, Tianbao Yang, and Rong Jin. 2012. Efficient Constrained Regret Minimization. *CoRR* abs/1205.2265 (2012).

[22] M. M. Mowla, I. Ahmad, D. Habibi, and Q. V. Phung. 2017. An energy efficient resource management and planning system for 5G networks. In *Proc. of IEEE CCNC.*

[23] Yong Niu, Yong Li, Depeng Jin, Li Su, and Athanasios V. Vasilakos. 2015. A survey of millimeter wave communications (mmWave) for 5G: opportunities and challenges. *Wireless Networks* 21, 8 (2015), 2657–2676.

[24] Sadegh Nobari. 2019. DBA: Dynamic Multi-Armed Bandit Algorithm. In *Proc. of AAAI.*

[25] Tao Ouyang, Rui Li, Xu Chen, Zhi Zhou, and Xin Tang. 2019. Adaptive User-managed Service Placement for Mobile Edge Computing: An Online Learning Approach. In *Proc. of IEEE INFOCOM.*

[26] S. Shahrampour, A. Rakhlin, and A. Jadbabaie. 2017. Multi-armed bandits in multi-agent networks. In *Proc. of IEEE ICASSP.*

[27] Yuxuan Sun, Sheng Zhou, and Jie Xu. 2017. EMM: Energy-Aware Mobility Management for Mobile Edge Computing in Ultra Dense Networks. *IEEE Journal on Selected Areas in Communications* 35, 11 (2017), 2637–2646.

[28] S. Sundar and B. Liang. 2018. Offloading Dependent Tasks with Communication Delay and Deadline Constraint. In *Proc. of IEEE INFOCOM.*

[29] William Webb. 2017. Modelling Small Cell Deployments within a Macrocell. *Digital Policy, Regulation and Governance* 20 (2017).

[30] J. Xu, L. Chen, and P. Zhou. 2018. Joint Service Caching and Task Offloading for Mobile Edge Computing in Dense Networks. In *Proc. of IEEE INFOCOM.*

[31] L. Yang, H. Zhang, M. Li, J. Guo, and H. Ji. 2018. Mobile Edge Computing Empowered Energy Efficient Task Offloading in 5G. *IEEE Transactions on Vehicular Technology* 67, 7 (2018).

[32] Qixia Zhang, Fangming Liu, and Chaobing Zeng. 2019. Adaptive Interference-Aware VNF Placement for Service-Customized 5G Network Slices. In *Proc. IEEE INFOCOM.*

A APPENDIX

A.1 Proof of Lemma 1

Assumption 1. *We assume there exists constant $L > 0$, $\sigma > 0$ such that for all contexts $\phi_x, \phi_y \in \Phi$, it holds that $|l_{\phi_x} - l_{\phi_y}| \le L||\phi_x - \phi_y||^{\sigma}$, where $||\cdot||$ denotes the Euclidean norm in $\mathbb{R}^{D_\Phi}$.*

This assumption is needed for the analysis of the regret, but it should be noted that our algorithm can also be applied if this assumption does not hold.[2]

Proof of Lemma 1: Let $W_t = \sum_{i\in\mathcal{D}_{m,t}} w_{f_i}^{m,t}$ and $\tilde{W}_t = \sum_{i\in\mathcal{D}_{m,t}} \tilde{w}_{f_i}^{m,t}$. Define $\hat{\boldsymbol{r}}_{m,t} = \hat{\boldsymbol{g}}_{m,t} + \lambda_1^t \hat{\boldsymbol{v}}_{m,t} + \lambda_2^t \hat{\boldsymbol{q}}_{m,t}$ and $\boldsymbol{r}_{m,t} = \boldsymbol{g}_{m,t} + \lambda_1^t \boldsymbol{v}_{m,t} + \lambda_2^t \boldsymbol{q}_{m,t}$.[3] For the sequence of selected offloading tasks $\mathcal{I}_m^t$ at $t \in \mathcal{T}$,

$$\begin{aligned}
&\sum_{t\in\mathcal{T}} \ln \frac{W_{t+1}}{W_t} = \ln \frac{W_{T+1}}{W_1} = \ln(\sum_{i\in\mathcal{D}_{m,T+1}} w_{f_i}^{T+1}) - \ln|\mathcal{D}_{m,1}| \\
&\ge \ln \sum_{i\in\mathcal{D}_{m,T+1}} p_i^{m,t} w_{f_i}^{m,T+1} - \ln K_m \\
&= \ln[\sum_{i\in\mathcal{D}_{m,T+1}} \frac{p_i^{m,t}}{c} (\sum_{t:f_i\notin S^t} \exp(\eta \hat{r}_{f_i}^{m,t}))] - \ln \frac{K_m}{c} \\
&\ge \sum_{i\in\mathcal{D}_{m,T+1}} \frac{p_i^{m,t}}{c} \sum_{t:f_i\notin S^t} \eta \hat{r}_{f_i}^{m,t} - \ln \frac{K_m}{c} && (7) \\
&= \frac{\eta}{c} \sum_{i\in\mathcal{D}_{m,T+1}} p_i^{m,t} \sum_{t:f_i\notin S^t} \hat{r}_{f_i}^{m,t} - \ln \frac{K_m}{c} && (8)
\end{aligned}$$

Inequality (7) follows from the concavity of the log function. Then, we have,

$$\begin{aligned}
&\frac{W_{t+1}}{W_t} = \sum_{i:f_i\notin S^t} \frac{w_{f_i}^{m,t+1}}{W_t} + \sum_{i:f\in S^t} \frac{w_{f_i}^{m,t+1}}{W_t} \\
&\le \sum_{i:f_i\notin S^t} \frac{w_{f_i}^{m,t}}{W_t} [1 + \eta \hat{r}_{f_i}^{m,t} + (\eta \hat{r}_{f_i}^{m,t})^2] + \sum_{i:f_i\in S^t} \frac{w_{f_i}^{m,t}}{W_t} && (9) \\
&= 1 + \frac{\tilde{W}_t}{W_t} \sum_{i:f_i\notin S^t} \frac{\frac{\tilde{p}_i^{m,t}}{c} - \frac{\gamma}{K_m}}{1-\gamma} [\eta \hat{r}_{f_i}^{m,t} + (\eta \hat{r}_{f_i}^{m,t})^2] \\
&\le 1 + \frac{\eta}{c(1-\gamma)} \sum_{i:f_i\notin S^t} \tilde{p}_i^{m,t} \hat{r}_{f_i}^{m,t} + \frac{\eta^2}{c(1-\gamma)} \sum_{i:f_i\notin S^t} \tilde{p}_i^{m,t} (\hat{r}_{f_i}^{m,t})^2 && (10) \\
&\le 1 + \frac{\eta^2}{c(1-\gamma)} \sum_{i:f_i\notin S^t} (1 + \lambda_1^t + \lambda_2^t) \hat{r}_{f_i}^{m,t} + \frac{\eta}{c(1-\gamma)} \sum_{i:f_i\notin S^t} \tilde{p}_i^{m,t} \hat{r}_{f_i}^{m,t}. && (11)
\end{aligned}$$

Inequality (9) uses $e^a \le 1 + a + a^2$ for $a \le 1$, inequality (10) holds because $\tilde{W}_t/W_t \le 1$, and inequality (11) uses the fact that $\tilde{p}_i^{m,t} \hat{r}_{f_i}^{m,t} = r_{f_i}^{m,t} \le 1+\lambda_1^t+\lambda_2^t$ for $f_i \in S^t$ and $\tilde{p}_i^{m,t} \hat{r}_{f_i}^{m,t} = 0$ for $f_i \notin S^t$. Due to $1 + x \le e^x$, we have $\ln \frac{W_{t+1}}{W_t} \le \frac{\eta}{c(1-\gamma)} \sum_{i:f_i\notin S^t} \tilde{p}_i^{m,t} \hat{r}_{f_i}^{m,t} + \frac{\eta^2}{c(1-\gamma)} \sum_{i:f_i\notin S^t} (1 + \lambda_1^t + \lambda_2^t) \hat{r}_{f_i}^{m,t}$. By summing over t, we obtain

[2] However, a regret bound might not be guaranteed in this case.

[3] For convenience, we denote $\lambda_{m,1}^t(\lambda_{m,2}^t)$ as $\lambda_1^t(\lambda_2^t)$ later.

$$\ln\frac{W_{T+1}}{W_1} \le \frac{\eta}{c(1-\gamma)}\sum_{t\in\mathcal{T}}\sum_{i:f_i\notin S^t}\tilde{p}_i^{m,t}\hat{r}_{f_i}^{m,t} + \frac{\eta^2}{c(1-\gamma)}\sum_{t\in\mathcal{T}}\sum_{i:f_i\notin S^t}(1+\lambda_1^t+\lambda_2^t)\hat{r}_{f_i}^{m,t}. \tag{12}$$

From (8) and (12), we get

$$\sum_{i\in\mathcal{D}_{m,T+1}} p_i^{m,t}\sum_{t:f_i\notin S^t}\hat{r}_{f_i}^{m,t} - \frac{c}{\eta}\ln\frac{K_m}{c} \le \frac{1}{1-\gamma}\sum_{t\in\mathcal{T}}\sum_{i:f_i\notin S^t}\tilde{p}_i^{m,t}\hat{r}_{f_i}^{m,t} + \frac{\eta}{1-\gamma}\sum_{t\in\mathcal{T}}\sum_{i:f_i\notin S^t}(1+\lambda_1^t+\lambda_2^t)\hat{r}_{f_i}^t.$$

Since $\tilde{p}_i^{m,t}=1$ for $f_{i,t}\in S^t$, we obtain $\sum_{i\in\mathcal{D}_{m,T+1}} p_i^{m,t}\sum_{t:f_i\in S^t}\hat{r}_{f_i}^{m,t} \le \frac{1}{1-\gamma}\sum_{t\in\mathcal{T}}\sum_{i:f_i\in S^t}\tilde{p}_i^{m,t}\hat{r}_{f_i}^{m,t}$. Then we get $\sum_{t\in\mathcal{T}}\hat{\boldsymbol{r}}_{m,t}\boldsymbol{p}_{m,t} - \frac{\sum_{t\in\mathcal{T}}\hat{\boldsymbol{r}}_{m,t}\tilde{\boldsymbol{p}}_{m,t}}{1-\gamma} \le \frac{c}{\eta}\ln\frac{K_m}{c} + \frac{\eta}{1-\gamma}\sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{D}_{m,t}}(1+\lambda_1^t+\lambda_2^t)\hat{r}_{f_i}^{m,t}$. Taking expectation on both sides, we have $\mathbb{E}[\sum_{t\in\mathcal{T}}\hat{\boldsymbol{r}}_{m,t}\boldsymbol{p}_{m,t} - \frac{\sum_{t\in\mathcal{T}}\hat{\boldsymbol{r}}_{m,t}\tilde{\boldsymbol{p}}_{m,t}}{1-\gamma}]$

$$\begin{aligned}
&\le \frac{\eta}{1-\gamma}\mathbb{E}[\sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{D}_{m,t}}(1+\lambda_1^t+\lambda_2^t)\hat{r}_{f_i}^{m,t}] + \frac{c}{\eta}\mathbb{E}[\ln\frac{K_m}{c}]\\
&\le \frac{\eta}{1-\gamma}\mathbb{E}[\sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{D}_{m,t}}(1+\lambda_1^t+\lambda_2^t)^2] + \frac{c}{\eta}\mathbb{E}[\ln\frac{K_m}{c}]
\end{aligned} \tag{13}$$

Inequality (13) holds because $g_{f_i}^{m,t}, v_{f_i}^{m,t} \le 1$. Given that $(a+b)^2 \le 2a^2+2b^2$, we can derive

$$\begin{aligned}
&\mathbb{E}[\sum_{t\in\mathcal{T}}\hat{\boldsymbol{r}}_{m,t}\boldsymbol{p}_{m,t} - \frac{\sum_{t\in\mathcal{T}}\hat{\boldsymbol{r}}_{m,t}\tilde{\boldsymbol{p}}_{m,t}}{1-\gamma}]\\
&\le \frac{4K_m\eta}{1-\gamma}\sum_{t\in\mathcal{T}}[(\lambda_1^t)^2+(\lambda_2^t)^2] + \frac{2K_m\eta}{1-\gamma} + \frac{c}{\eta}\ln\frac{K_m}{c}.
\end{aligned} \tag{14}$$

From Line 16 of the Alg. 3, we denote that $\lambda_1^{t+1} \le [(1-\delta\eta)\lambda_1^t+\eta\alpha]_+$. By induction on λ_1^t, we can obtain $\lambda_1^t \le \frac{\alpha}{\delta}$. Let $y_t(\lambda_1) = \frac{\delta}{2}\lambda_1^2 + \lambda_1(\frac{\hat{\boldsymbol{v}}_{m,t}\tilde{\boldsymbol{p}}_{m,t}}{1-\gamma}-\alpha)$, $t\in\mathcal{T}$. Then we have $\lambda_1^{t+1} = [\lambda_1^t - \eta\nabla y_t(\lambda_1^t)]_+$. Applying the standard analysis of online gradient descent yields

$$\begin{aligned}
&|\lambda_1^{t+1}-\lambda_1|^2 = |[\lambda_1^t-\eta\nabla y_t(\lambda_1^t)]_+ - \lambda_1|^2\\
&\le 2\eta^2\alpha^2 + |\lambda_1^t-\lambda_1|^2 + 2\eta^2(\frac{\boldsymbol{v}\hat{_{m,t}}\tilde{\boldsymbol{p}}_{m,t}}{1-\gamma})^2 + 2\eta(y_t(\lambda_1)-y_t(\lambda_1^t)).
\end{aligned}$$

By rearranging the terms we get $y_t(\lambda_1^t) - y_t(\lambda_1) \le$

$$\begin{aligned}
&\frac{1}{2\eta}(|\lambda_1^{t+1}-\lambda_1|^2 - |\lambda_1^t-\lambda_1|^2) + \eta[(\frac{\boldsymbol{v}\hat{_{m,t}}\tilde{\boldsymbol{p}}_{m,t}}{1-\gamma})^2+\alpha^2]\\
&\le \frac{1}{2\eta}(|\lambda_1^{t+1}-\lambda_1|^2 - |\lambda_1^t-\lambda_1|^2) + c^2\eta + \frac{c\eta}{(1-\gamma)^2}\sum_{i\in\mathcal{D}_{m,t}}\hat{u}_{f_i}^{m,t}.
\end{aligned}$$

Then, taking expectation over $\sum_{t\in\mathcal{T}}[y_t(\lambda_1^t)-y_t(\lambda_1)]$, we obtain

$$\begin{aligned}
&\mathbb{E}[\frac{2}{\delta}\sum_{t\in\mathcal{T}}((\lambda_1^t)^2-\lambda_1^2) + \sum_{t\in\mathcal{T}}(\lambda_1^t-\lambda_1)(\frac{\hat{\boldsymbol{v}}_{m,t}\tilde{\boldsymbol{p}}_{m,t}}{1-\gamma}-\alpha)]\\
&\le \frac{\lambda_1^2}{2\eta} + c^2\eta T + \frac{cK_m\eta}{(1-\gamma)^2}T.
\end{aligned} \tag{15}$$

Similarly, we get $\mathbb{E}[\frac{2}{\delta}\sum_{t\in\mathcal{T}}((\lambda_2^t)^2-\lambda_2^2) + \sum_{t\in\mathcal{T}}(\lambda_2^t-\lambda_2)(\beta-\frac{\boldsymbol{q}\hat{_{m,t}}\tilde{\boldsymbol{p}}_{m,t}}{1-\gamma})]$

$$\le \frac{\lambda_2^2}{2\eta} + c^2\eta T + \frac{cK_m\eta}{(1-\gamma)^2}T. \tag{16}$$

Combining (15), (16) with (14), we have

$$\begin{aligned}
&\sum_{t\in\mathcal{T}}\boldsymbol{g}_{m,t}\boldsymbol{p}_{m,t} - \frac{1}{1-\gamma}\mathbb{E}[\sum_{t\in\mathcal{T}}\boldsymbol{g}_{m,t}\tilde{\boldsymbol{p}}_{m,t}] - \mathbb{E}[(\frac{\delta T}{2}+\frac{1}{2\eta})(\lambda_1^2+\lambda_2^2)\\
&-\lambda_2\sum_{t\in\mathcal{T}}(\frac{\hat{\boldsymbol{q}}_{m,t}\tilde{\boldsymbol{p}}_{m,t}}{1-\gamma}-\beta) - \lambda_1\sum_{t\in\mathcal{T}}(\alpha-\frac{\hat{\boldsymbol{v}}_{m,t}\tilde{\boldsymbol{p}}_t}{1-\gamma})] \le \frac{2K_m\eta}{1-\gamma}T\\
&+2c^2\eta T + \frac{2cK_m\eta}{(1-\gamma)^2}T + (\frac{4K_m\eta}{1-\gamma}-\frac{\delta}{2})\sum_{t\in\mathcal{T}}[(\lambda_1^t)^2+(\lambda_2^t)^2]\\
&+\frac{c}{\eta}\ln\frac{K_m}{c} + \mathbb{E}[\sum_{t\in\mathcal{T}}(\lambda_1^t(\alpha-\frac{\boldsymbol{v}_{m,t}\boldsymbol{p}_{m,t}}{1-\gamma}) + \lambda_2^t(\frac{\boldsymbol{q}_t\boldsymbol{p}_{m,t}}{1-\gamma})-\beta)].
\end{aligned}$$

Since $\frac{4K_m\eta}{1-\gamma} \le \frac{\delta}{2}$, when $\boldsymbol{v}_{m,t}\boldsymbol{p}_{m,t} \ge \alpha$ and $\boldsymbol{q}_{m,t}\boldsymbol{p}_{m,t} \le \beta$,

$$\begin{aligned}
&(1-\gamma)\sum_{t\in\mathcal{T}}\boldsymbol{g}_{m,t}\boldsymbol{p}_{m,t} - \mathbb{E}[\sum_{t\in\mathcal{T}}\boldsymbol{g}_{m,t}\tilde{\boldsymbol{p}}_{m,t}] + \mathbb{E}[-(\frac{\delta T}{2}+\frac{1}{2\eta})(\lambda_1^2+\lambda_2^2)\\
&+\lambda_1\sum_{t\in\mathcal{T}}(\alpha(1-\gamma)-\boldsymbol{v}_{m,t}\tilde{\boldsymbol{p}}_{m,t}) + \lambda_2\sum_{t\in\mathcal{T}}(\boldsymbol{q}_{m,t}\tilde{\boldsymbol{p}}_{m,t}-\beta(1-\gamma))]\\
&\le \frac{c(1-\gamma)}{\eta}\ln\frac{K_m}{c} + 2\eta K_mT + 2\eta c^2T + \frac{2cK_m\eta}{1-\gamma}T.
\end{aligned}$$

By rearranging the terms we get

$$\begin{aligned}
&\sum_{t\in\mathcal{T}}\boldsymbol{g}_{m,t}\boldsymbol{p}_{m,t} - \mathbb{E}[\sum_{t\in\mathcal{T}}\boldsymbol{g}_{m,t}\tilde{\boldsymbol{p}}_{m,t}] + \mathbb{E}\{\lambda_1\sum_{t\in\mathcal{T}}[\alpha(1-\gamma)-\boldsymbol{v}_{m,t}\tilde{\boldsymbol{p}}_{m,t}]-\\
&\lambda_1^2(\frac{\delta T}{2}+\frac{1}{2\eta})\} + \mathbb{E}\{\lambda_2\sum_{t\in\mathcal{T}}[\boldsymbol{q}_{m,t}\tilde{\boldsymbol{p}}_{m,t}-\beta(1-\gamma)] - \lambda_2^2(\frac{\delta T}{2}+\frac{1}{2\eta})\} \le F(T),
\end{aligned}$$

where $F(T) = \frac{c(1-\gamma)}{\eta}\ln\frac{K_m}{c} + 2\eta K_mT + 2\eta c^2T + \frac{2cK_m\eta}{1-\gamma}T + \gamma K_mT$. Let $\lambda_1 = \frac{\sum_{t\in\mathcal{T}}[\alpha(1-\gamma)-\boldsymbol{v}_{m,t}\tilde{\boldsymbol{p}}_{m,t}]}{\delta T+1/\eta}$ and $\lambda_{m,2} = \frac{\sum_{t\in\mathcal{T}}[\boldsymbol{q}_{m,t}\tilde{\boldsymbol{p}}_{m,t}-\beta(1-\gamma)]}{\delta T+1/\eta}$. By maximizing over $\boldsymbol{p}_{m,t}$,

$$\begin{aligned}
&\sum_{t\in\mathcal{T}}\boldsymbol{g}_{m,t}\boldsymbol{p}_{m,t}^{**} - \mathbb{E}[\sum_{t\in\mathcal{T}}\boldsymbol{g}_{m,t}\tilde{\boldsymbol{p}}_{m,t}] + \mathbb{E}[\frac{[\sum_{t\in\mathcal{T}}(\alpha(1-\gamma)-\boldsymbol{v}_{m,t}\tilde{\boldsymbol{p}}_{m,t})]_+^2}{2\delta T+2/\eta}]\\
&+\mathbb{E}[\frac{[\sum_{t\in\mathcal{T}}(\boldsymbol{q}_{m,t}\tilde{\boldsymbol{p}}_{m,t}-\beta(1-\gamma))]_+^2}{2\delta T+2/\eta}] \le F(T),
\end{aligned} \tag{17}$$

where $\boldsymbol{p}_{m,t}^{**}$ is the optimal solution for SCN m without considering conflicts.

So far, we have been substituting task i's relevant parameters ($g_{\phi_i}^{m,t}$ and so on) with its corresponding hypercube $f_{i,t}$'s parameters while analyzing the regret and violations. Next, we take into account the deviation between them. According to Assumption 1, we define $\max\{|l_{\phi_i}-l_{\phi_j}|\} = \max\{L||\phi_i-\phi_j||^{\sigma}\} = L(D_\Phi^{1/2}/h_T)^{\sigma}$ as the gap between any two contexts in the same hypercube. Let $h_T = \lceil T^{\varepsilon}\rceil \le 2T^{\varepsilon}$, $0<\varepsilon<\frac{1}{h_T}$. After considering the *context gap*, we finally obtain LHS of (17) $\le LD_\Phi^{\frac{\sigma}{2}}T^{-\sigma\varepsilon}F(T)$.

As $\gamma = \min(1,\sqrt{\frac{2K_m(1+c)}{c\ln(K_m/c)T^{2/3}}}) = \Theta(T^{-\frac{1}{3}})$, $\delta = \Theta(T^{-\frac{1}{3}})$ and $\eta = \Theta(\frac{1}{K_m}T^{-\frac{2}{3}})$, we have $F(T) = O(cK_m\ln K_mT^{\frac{2}{3}})$. It is easy to show $R^m(T) \le LD_\Phi^{\frac{\sigma}{2}}T^{-\sigma\varepsilon}F(T)$. Furthermore, we get $V_1^m(T) \le \sqrt{2(F(T)+cT)(\delta T+1/\eta)}$ and $V_2^m(T) \le \sqrt{2(F(T)+cT)(\delta T+1/\eta)}$. Finally, we obtain $R^m(T) \le O(LD_\Phi^{\frac{\sigma}{2}}cK_m\ln K_mT^{\frac{2}{3}-\frac{\sigma}{D_\Phi}})$ and $V_1^m(T)(V_2^m(T)) \le O(L^{\frac{1}{2}}D_\Phi^{\frac{\sigma}{4}}c^{\frac{1}{2}}K_m^{\frac{1}{2}}T^{\frac{5}{6}})$. □

1.2 Proof of Lemma 2

The proof is based on a *charging argument*, which is often used in the matching literature [11]. Specifically, we prove that each edge belonging to Ω^* can be charged to an edge in Ω by constructing an injective function. During the charging process, we ensure that no more than $c + 1$ edges in Ω^* are charged to the same edge in Ω. It shows the approximation factor of our greedy algorithm is $c + 1$.

First, we construct an injective function $h : \Omega^* \to \Omega$. For any edge $(m, i) \in \Omega^*$, $h[(m, i)]$ is defined as the edge $(m', i') \in \Omega$ with the largest weight that is not less than $w(m, i)$, where $m' = m$ or $i' = i$.

To show that h is indeed a function mapping Ω^* to Ω, we assume that there is no such edge $h[(m, i)]$ for edge $(m, i) \in \Omega^*$. This means that edge (m, i) is not in Ω, which implies that the edge (m, i) was removed from the set E' at some iterations during the course of Alg. 4. In particular, as per the algorithm, the removal can occur in only two ways: i) via Line 6, there exists an edge (m', i) that has been selected to Ω ahead of (m, i); ii) via Line 8, the SCN m has been selected c tasks, *i.e*, the communication capacity constraint was met. On this basis, we divide the analysis into two cases.

Case 1: Removal via Line 6 in Alg 4. Without loss of generality, we suppose that $(m', i))$ is the edge added to Ω during the iteration in which edge (m, i) is removed. According to the definition of Line 2, $(m', i) = \arg\max_{(m'', i'') \in E'} w(m'', i'')$ before the removal of (m, i) from E'. Hence, we can infer that $w(m, i) \le w(m', i)$.

Case 2: Removal via Line 8 in Alg 4. In this case, because the number of tasks selected by SCN m has reached the capacity limit and our greedy algorithm selects tasks in the decreasing order of the weight of the corresponding edge, it must be the case that for every $(m, i') \in \Omega$, we can obtain that $w(m, i) \le w(m, i')$.

Now we find that there exists an edge in Ω that satisfies the definition of h in both cases, which is contrary to our assumption. This means that for each edge in Ω^*, there is such an edge $h(m, i)$ in Ω. Meanwhile, such an edge is unique according to h's definition. Therefore, h proves to be a function mapping Ω^* to Ω.

Next, we show that h is a $(c + 1)$*-to-one* function. According to our greedy algorithm and h's definition, the only edges that can be charged to (m, i) must contain either the node m or the node i. Hence, for any given edge in Ω, at most $(c + 1)$ edges in Ω^* can be charged to it, which means h is $(c+1)$*-to-one.* Since $w(m, i)$ $(i.e, \tilde{p}_i^{m,t}$ in our setting) is proportional to $g_i^{m,t}$ and the auxiliary variables are updated in the same way at the same time slot, by the charging argument, our greedy algorithm is a $(c + 1)$-approximation algorithm, namely, $\sum_{(m,i)\in\Omega} g_i^{m,t} w(m, i) \ge \frac{1}{c+1} \sum_{(m,i)\in\Omega^*} g_i^{m,t} w(m, i)$. Note that the performance of our greedy algorithm in practice appears to be much closer to the optimum solution than the $1/(c + 1)$ approximation. □