Reinforcement Learning based Matching for Decentralized Task Offloading in Fog Computing Networks

Hoa Tran-Dang, Dong-Seong Kim

***Abstract*—This paper proposes an algorithm called RL-MATCH for performing the distributed task offloading in the fog computing networks based on the matching theory and reinforcement learning. Fundamentally, RL-MATCH aims to match each task node (TN) having computation needs with each helper node (HN) with available computing resource for task offloading realization. Given the dynamic of IoT and fog computing environment featured by the time varying of task requirement and resource states, online learning is needed to estimate the network context as well as construct preference relations for the two sides of matching game. We, therefore propose to use Thompson Sampling (TS) method for this bandit learning problem to acquire better exploitation and exploration trade-off, therefore allowing TNs to achieve the informed preference relations of HNs quickly. Extensive simulation results demonstrate the potential advantages of the TS based learning over the ϵ-greedy and UCB based baselines.**

***Index Terms*—Fog computing network, multi-armed bandit, Thompson sampling, decentralized offloading, stable matching.**

I. Introduction

Fog computing has been introduced and integrated widely in the practical IoT and cyber-physical systems (CPS) [1], [2], [3]. As an extension of cloud computing, fog computing platforms placed between the cloud layer and user equipment (UE) layer in the systems also provide the cloud-like services (i.e., IaaS, PaaS, SaaS) to the UEs. In this context, devices with integrated fog computing platforms called fog nodes (FNs) can process and offload most tasks requested by the UEs on behalf of the cloud servers [4], thereby allowing the systems to achieve the improved performances in terms of service delay, energy saving, and service cost [5].

However, to realize these benefits of fog computing paradigm, there requires efficient task offloading operations to overcome inherent challenges in the fog computing environment such as the heterogeneity of fog computing devices, various types of computational tasks with different quality of service (QoS) and quality of experience (QoE) requirements [6].

There are a large number of centralized optimization techniques and algorithms proposed in the literature to provide optimal offloading solutions [7]. These approaches require a centralized control to gather the global system information, thus incurring a significant overhead and computation complexity of algorithms especially in case of density and complicated heterogeneity of fog computing environment [8].

The aforementioned limitations of optimization have lead to a second class of game theory based offloading solutions that can avoid the cost-intensive centralized resource management as well as substantially reduce the complexity of algorithms [9]. However, classical game theoretical algorithms such as best response require some information regarding actions of other players [10]. Correspondingly, many assumptions are introduced in the game theory based algorithms to simplify the system models that, in some case, are impractical.

Recently, matching theory has emerged as a promising technique applied to derive distributed task offloading algorithms [11] that significantly can reduce the service latency in the fog-based systems. More importantly, the matching-based approaches have potential advantages over the optimization and game theory based solution owing to the distributed and low computational complexity algorithm [12], [13].

However, most of aforementioned solutions assume the information regarding the resource states of fog computing nodes are known $a\ priori$, which is not realistic in many practical applications. For example, the resource demand-side FNs called task nodes (TNs) are likely to be uncertain about the computing capability (i.e., CPU frequency, queuing delay) of resource supply-side FNs (i.e., helper nodes (HNs)) at time of offloading requesting since it is varying over time. Therefore, to efficiently offload the tasks in an online manner, the TNs must interact constantly with the HNs to learn their unknown computing resource status.

Multi-armed bandit (MAB) is a common approach to modeling this type of learning process, which aims to solve the exploitation and exploration dilemma [14]. Several algorithms including ϵ-greedy, upper confidence bound (UCB), and Thompson sampling (TS) are proposed for the player (i.e., learner) to select the optimal arm, which offers the highest cumulative reward [15]. Basically, the ϵ-greedy algorithm is simple to implement with low complexity but heavily relying on the random choice, thus occurring the long convergence. TS is Bayesian method [16] in which the player keeps a posterior distribution over the expected arm rewards, and at each round takes a sample from each arm's posterior, and then, plays the arm with the largest sample. Reward observed from the played arm is then used to update its posterior. This sampling strategy allows the arm to frequently select the arms whose probabilities of being optimal are the highest based on their posteriors and to occasionally explore inferior arms to

refine their posteriors. Meanwhile, the UCB strategy is to play the arm with the highest UCB index to trade-off exploration and exploitation, which is usually composed of sample mean reward of an arm plus an exploration bonus that accounts for the uncertainty in the arm's reward estimates [17]. Unlike TS, performance of this type of policies heavily rely on the confidence sets used to compute the exploration bonus. This together with the superior performance of TS evaluated in numerous applications [18] motivate us to investigate a TS based learning for our problem.

In these regards, this paper aims to develop an online task offloading algorithm abbreviated as RL-MATCH for the dynamic fog computing-enabled systems based on MAB learning, which particularly use the TS technique to efficiently learn the uncertainty of fog computing environment.

II. Related Works

Task offloading has recently gained popularity as a potential approach to efficiently use distributed computing resources, and several studies conducted in this area have proposed efficient offloading solutions using different methodologies.

The work presented in [19] developed a code offloading architecture with on-demand computing resource allocation and concurrent task execution. To enhance the total energy efficiency in homogeneous fog networks, the authors in [20] specifically made the assumption that they had complete knowledge of the system status. The offloading decision of which node to offload optimally becomes an integer programming problem and is challenging to solve under the supposition that the tasks could not be divided arbitrarily [21]. Real-time information collection and response are essential for making intelligent offloading decision in a fog-enabled network [22]. Using the real-time statuses of the users and the servers, such as the sizes of the compute queues, one efficient task offloading technique should be able to quickly adapt to the demanding dynamics in the environments. As a result, task offloading is frequently a stochastic programming issue, and the above-mentioned traditional optimization techniques with deterministic parameters are no longer appropriate. Utilizing the Lyapunov optimization approach is one option to resolve this conundrum. Additionally, a game-theoretic decentralized strategy was offered in [23] to provide independent offloading decisions from each user. Accordingly, the task offloading is structured as a game and followed the Nash equilibrium rather than attempting to solve a challenging integer programming issue.

All of the computation offloading strategies listed above presupposed complete knowledge of the system characteristics. In practical, these factors are sometimes unknown or just partially known to the user. As an example, some values (also known as bandit feedback) are only disclosed for the nodes that are queried. The authors in [24] specifically considered the calculation and communication delays of each task as a posterior. Each device's movement was thought to be unpredictable. Offloading feed-backs are used in the reinforcement learning-based offloading methods [25] to learn characteristics like latency and energy cost. There will be a trade-off between using the empirically best node as frequently as possible and researching other nodes to uncover more lucrative activities when the number of nodes that can be queried is constrained owing to the limiting amount of resources that are available. To tackle this trade-off, the ϵ-greedy approach, which is commonly used in reinforcement learning [25] should be used. This strategy, nevertheless, converges slowly and is non-optimal.

The authors in [26] introduced D2CIT - a two-tier distributed strategy for offloading computing in an IoT context with fog. They took into account SNs with time-sensitive tasks needed to be computed within the predefined deadlines. The high-level tasks will be divided into more manageable subtasks by D2CIT, which will then create a DATG. For the FN selection, they provided a greedy solution that separates the job from the appropriate SNs. In a dynamic setting, they develop an ϵ-greedy nonstationary MAB-based strategy for automated subtask redistribution.Additionally, they contrasted D2CIT with already existing solutions and demonstrated how the suggested algorithm performs better in terms of latency and speedup.

The study [27] examined a performance guarantee for an effective online task offloading approach in a fog-enabled network. A stochastic programming with delayed bandit feedback was defined and formulated since the expectations for processing speeds vary rapidly at unknown time instants and the system information is only accessible after completing the related tasks. BLOT, an effective online task offloading method based on the UCB policy, is developed to address this issue. The simulation results show that the proposed BLOT algorithm is capable of learning and selecting the appropriate node to offload tasks in an online manner under non-stationary conditions.

The work [28] presents the combination of learning and matching to design a learning-matching algorithm for task offloading and resource allocation in the vehicular fog computing (VFC) systems. Considering the uncertainty of information of systems, the algorithm proposes a pricing based model iterated over time slot to learn the uncertainty as well as handle the matching conflict. Evaluated by extensive simulations, the proposed algorithm can achieve bounded deviation from the optimal performance without the availability of global information.

III. System Model

A. Fog Computing Network

In the general system architecture, a FCN consists of multiple TNs and multiple HNs co-existing in an area to support computing various types of tasks as shown in Fig. 1. Without the loss of generality, this paper considers the FCN with an equal number M of TNs and HNs. Define the set of TNs and HNs as $\mathcal{T} = \{T_1, \ldots, T_i, \ldots, T_M\}$ and $\mathcal{H} = \{H_1, \ldots, H_j, \ldots, H_M\}$, respectively. FNs are heterogeneous in terms of computing capability, storage, and connectivity

technology. In addition, we assume that all the FNs can connect together through wireless links.

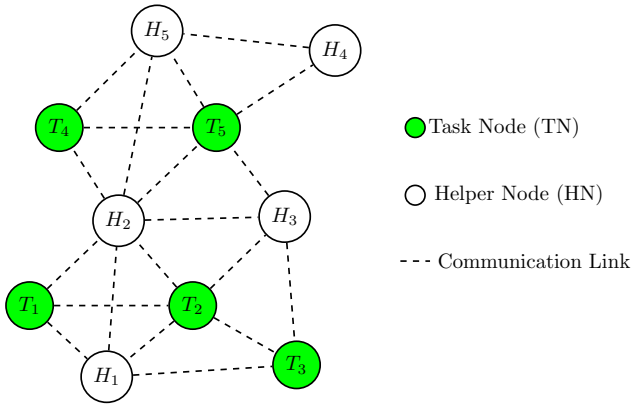

Fig. 1: An illustrative model of FCN with two types of FNs including TNs and HNs.

We adopt a time-slot model where the total time period is divided into K discrete intervals, the set of which is denoted as $\mathcal{S} = \{1, \ldots, t, \ldots, K\}$. At the t^{th} slot, each TN i generates a corresponding task $T_i(t)$, which is represented by a tuple $T_i(t) = \{L_i(t), \Gamma_i(t), D_i^{max}(t)\}$, where $L_i(t)$ is the task size (bits), $\Gamma_i(t)$ is the computation complexity of task (CPU cycles/bit), and $D_i^{max}(t)$ is the maximum permitted latency for the task T_i at time slot t. The specification of task varies across different slots. In addition, the tasks are assumed to be split arbitrarily, thus each task T_i should be either allocated as one whole piece to one neighboring HN or processed locally.

In the dynamic of fog computing environment, the computing capability of a HN H_j is represented through CPU frequency $f_j(t)$ (cycles/s), and CPU processing density $\rho_j(t)$ (cycles/bit), which are assumed to vary over different time slots.

B. Problem Formulation

Denote the set of task offloading decisions between M TNs and M HNs as $\alpha(t) = \{\alpha_{ij}(t)\}$, where each element is a binary variable, i.e., $\alpha_{ij}(t) \in \{0, 1\}$. $\alpha_{ij}(t) = 1$ means that the task T_i is decided to be offloaded by H_j in the time slot t; $\alpha_{ij}(t) = 0$, otherwise. $D_{ij}(t)$ is the total delay when H_j is assigned to process T_i and is calculated as follow:

$$D_{ji}(t) = \delta_{ij}^{tx}(t) + \delta_j^w(t) + \delta_{ji}^p(t), \tag{1}$$

where $\delta_{ij}^{tx}(t)$ is the transmission delay, $\delta_j^w(t)$ is the waiting delay in queue of $H_j(t)$, and $\delta_{ji}^p(t)$ is the processing delay by $H_j(t)$. Given the data rate $R_{ij}(t)$ (bits/s) between T_i and H_j at time slot t, we can achieve $\delta_{ij}^{tx}(t) = L_i(t)/R_{ij}(t)$. The processing delay is derived as:

$$\delta_{ji}^p(t) = \frac{L_i(t)\Gamma_i(t)}{f_i(t)}. \tag{2}$$

The waiting delay is measured by H_j as follow:

$$\delta_j^w(t) = \frac{Q_j(t)\rho_j(t)}{f_j(t)}, \tag{3}$$

where $Q_j(t)$ is the queue length (bits) of H_j in the time slot t.

For TNs, the objective of offloading their tasks to HNs is to minimize the long-term average delay $\overline{D}$, which is defined as follow:

$$\overline{D} = \lim_{K\to\infty} \frac{1}{K} \sum_{t=1}^{K} \sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_{ij}(t) D_{ij}(t). \tag{4}$$

Practically, the FNs are managed by different service providers which aim at maximizing the revenue by providing the best services (PaaS, IaaS, and SaaS) for task offloading operations. Therefore, the preference relation of HN is based on the pay-off that it receives to process the tasks. For each $H_j \in \mathcal{H}$, the cost to process a bit of task T_i is denoted as c_{ji}. The total pay-off received by H_j when offloading T_i is expressed as $C_{ji}(t) = \frac{D_{ij}(t)}{D_i^{max}} c_{ji} L_i(t)$. For HNs, the objective of offloading their tasks from TNs is to maximize the average pay-off $\overline{C}$ over time slots, which is defined as follow:

$$\overline{C} = \lim_{K\to\infty} \frac{1}{K} \sum_{t=1}^{K} \sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_{ij}(t) C_{ji}(t). \tag{5}$$

Definitely, the task offloading optimization problem is formulated as follows:

$$\begin{aligned} \textbf{P:} \quad & \min_{\alpha_{ij}(t)} \overline{D} \quad \& \quad \max_{\alpha_{ij}(t)} \overline{C} \\ \text{s. t.} \quad & C_1 : \alpha_{ij}(t) \in \{0,1\}, \forall \{i,j,t\} \in \{\mathcal{T}, \mathcal{H}, \mathcal{S}\}, \\ & C_2 : \sum_{i=1}^{\mathcal{T}} \alpha_{ij}(t) \leq 1, \forall j \in \mathcal{H}, \forall t \in \mathcal{S} \\ & C_3 : \sum_{j=1}^{\mathcal{H}} \alpha_{ij}(t) \leq 1, \forall i \in \mathcal{T}, \forall t \in \mathcal{S}. \end{aligned} \tag{6}$$

Here, the constraints C_1, C_2, and C_3 guarantee that at each time slot t, a TN's task can be offloaded to only one HN, and a HN can process at most one task of TN.

There are two difficulties in solve the above problem. First, it is a stochastic programming problem. The exact information about the delay $D_{ij}(t)$ is not available before the task $T_i(t)$ is completed. In addition, event if $D_{ij}(t)$ is estimated a priori, this problem is still a combinatorial optimization problem and the complexity is in the order of $\mathcal{O}(M^{2K})$. This is due to the fact that the previous offloading decisions determine the queue length in each HN and further affect the decisions of future tasks.

IV. RL-MATCH ALGORITHM

A. OTO Matching Model for Offloading Problem

Given the constraints C_2 and C_3 of problem **P**, the task offloading problem can be modeled as an one-to-one (OTO) matching game between players of two sets: $\mathcal{T}$ and $\mathcal{H}$, which is defined as follow.

Definition 4.1: The OTO matching is a *function* $\mathcal{M}$: $\mathcal{T} \cup \mathcal{H} \mapsto \mathcal{T} \cup \mathcal{H}$ such that the three following constraints are satisfied:

- For any $T_i \in \mathcal{T}$, $\mathcal{M}(T_i) \in \mathcal{H} \cup \{T_i\}$,
- For any $H_j \in \mathcal{H}$, $\mathcal{M}(H_j) \in \mathcal{T} \cup \{H_j\}$,
- For any $T_i \in \mathcal{T}$ and $H_j \in \mathcal{H}$, $T_i = \mathcal{M}(H_j)$ if and only if $H_j = \mathcal{M}(T_i)$.

In the OTO matching model, each agent T_i can only be matched with one agent H_j, and T_i remains unmatched if $\mathcal{M}(T_i) = T_i$. The objective of matching is to reach the stable status for all pairs.

Definition 4.2: A matching $\mathcal{M}$ is pairwise stable if there is no block pair (T_i, H_j).

Definition 4.3: (T_i, H_j) is a block pair for a matching $\mathcal{M}$ if three following conditions are satisfied:

- $\mathcal{M}(T_i) \neq H_j$,
- $H_j \succ_{T_i} \mathcal{M}(T_i)$,
- $T_i \succ_{H_j} \mathcal{M}(H_j)$,

where $\succ_{T_i}$ and $\succ_{H_j}$ represent two preference relations allowing the TNs and HNs from each of the two sets to express preferences over the opposite nodes.

Each node builds a ranking of other nodes in the opposite side individually using this preference relation, and then creating its own preference list (PL). With the full connected FCNs, each node has a complete PL over the nodes on the opposite side. Assume that each $T_i \in \mathcal{T}$ has a PL denoted as $\mathcal{P}(T_i) = (H_2, H_4, \cdots, H_j, \cdots, H_M, T_i)$. This means that T_i prefers agent H_2 to H_4 (i.e., $H_2 \succ_{T_i} H_4$).

To achieve the PL, each node is based on its preference values when matching with all the nodes of opposite set. These values are usually determined by utility functions taking account the objective of matching game. In this paper, for each T_i, its preference value for H_j is quantified by an unknown value $x_{ij} \in [0, 1]$. For two different H_j and $H_{j'}$, $x_{ij} > x_{ij'}$ implies that $H_j \succ_{T_i} H_{j'}$. For the objective of delay minimization, the preference value taking into account the capability of HNs to processing the tasks serve as the reward offered by HN. Furthermore, to limit the rewards to the range (0,1), we use the sigmoid function and is calculated as follow:

$$x_{ij}(t) = \frac{1}{1 + e^{-(D_i^{max}(t) - D_{ij}(t))}}. \tag{7}$$

Similarly, denote y_{ji} as the preference value of H_j for T_i. For two different player T_i and $T_{i'}$, $y_{ji} > y_{ji'}$ implies that H_i prefers T_i to $T_{i'}$. The ranking for the reference of each H_j is known after it receives all the offloading requests sent from players (TNs). We use $C_{ji}(t)$ to express the preference values of T_i for H_j (i.e., $y_{ji}(t) = C_{ji}(t)$). For any two tasks T_i and $T_{i'}$ requested to be offloaded by a HN H_j, the preference relation is as follow: $T_i \succ_{H_j} T_{i'}$ if and only if $y_{ji} > y_{ji'}$.

B. Algorithm Description

Due to the dynamic nature of fog computing environment characterized by the time-varying capabilities of HNs and QoS requirements of tasks, $x_{ij}(t)$ are unknown a prior and must be learned by MAB learning. In this context, TNs and HNs plays roles as players and arms, respectively.

At each round $r = 1, 2, ..., R$ of time slot $t = 1, 2, ..., K$, each player T_i attempts to pull an arm H_j. When multiple player attempt to pull the same arm, a matching conflict occurs and only the player preferred most by this arm is accepted. If a player T_i wins the matching conflict, it will receive a random reward $x_{ij}(t, r)$ derived from Equation 7. Otherwise, if failing the conflict, T_i is unmatched in this round and receive no reward, i.e., $x_{ij}(t, r) = 0$. Over R round, the average reward $\overline{X}_{ij}(t)$ received by T_i when attempting an arm H_j is estimated as follow:

$$\overline{X}_{ij}(t) = \frac{\sum_{r=0}^{R} \gamma^r x_{ij}(t, r)}{\sum_{r=0}^{R} \gamma^r}, \tag{8}$$

where $\gamma \in (0, 1)$ serves as the discount factor.

Algorithm 1 presents the procedures to implement the preference learning and matching conflict handling.

The algorithm takes the player set $\mathcal{T}$ and arm set $\mathcal{H}$ as input. For each player T_i and arm H_j, the algorithm maintains a Beta distribution $Beta(a_{ij}, b_{ij})$ for the preference value. Initially, the distribution is $Beta(1, 1)$ corresponding to the uniform distribution on $[0, 1]$. It will be later updated based on observed feedback and tend to concentrate on the mean value $\overline{X}_{ij}(t)$. In round r of slot t, the algorithm samples an index $\theta_{ij}(r)$ from $Beta(a_{ij}, b_{ij})$ to represent the current estimation.

To avoid the frequent conflicts, each player constructs a potential matching set to exclude arms that already reject it in the previous round. We assume that the successful matched players are public at the end of each round. However, players can still simultaneously pull same arms for next round. To address this issues, we incorporate a random delay mechanism with hyper-parameter λ. Accordingly, each player first draws a Bernoulli random variable $\pi_i(t)$ with expectation λ. If $\pi_i(t) = 0$, T_i still attempts to pull the arms with the largest index in the potential set; otherwise, it follows the last-round choice.

When all players decide which arm to pull in this round, arms will determine which player to accept according o their rankings. If player T_i wins the conflict, it updates the average reward and the corresponding Beta distribution.

V. SIMULATION RESULTS AND EVALUATION ANALYSIS

A. Simulation Environment Configuration

We evaluate the proposed algorithms in the fog environments where the network size ($2 \times M$) is 10 (5 TNs and 5 HNs). Table I summarizes the important parameters and values for the simulation scenario, where U[x,y] indicates the uniform distribution on interval [x, y]. In each scenario, we run all

Algorithm 1: RL-MATCH Algorithm

Input: Player set $\mathcal{T}$, arm set $\mathcal{H}$, parameter $\lambda \in (0,1)$.
Output: A stable matching $\mathcal{M} = \{(T_i(K), H_j(K))\}$

Initialization: $\forall\{T_i, H_j\} \in \{\mathcal{T}, \mathcal{H}\}$, random matching at $t = 0$: $\mathcal{M}(T_i(0)) = H_j(0)$, $a_{ij} = b_{ij} = 1$.
for $t = 1, 2, ..., K$ **do**
  for $r = 1, 2, ..., R$ **do**
    for $T_i \in \mathcal{T}$ **do**
      $\forall H_j \in \mathcal{H}$, sample $\theta_{ij}(t) \sim Beta(a_{ij}, b_{ij})$
      Independently draw $\pi_i(t) \sim Bernoulli(\lambda)$
      if $\pi_i(t) = 0$ **then**
        Construct potential matching set of T_i
        $P_i(t) := \{H_j : y_{ji}(t) \geq y_{ji'}(t)\}$ where $\mathcal{M}(T_{i'}(t-1)) = H_j$
        Pull $H_j(t) \in \arg\max_{H_j \in P_i(t)} \theta_{ij}(t)$
      else
        Pull $H_j(t) = \mathcal{M}(T_i(t-1))$
      if p_i *wins matching confict* **then**
        $H_j(t) = \mathcal{M}(T_i(t))$
        T_i received an instant reward $x_{ij}(t,r)$ dervied from Equation (7)
        Update $\overline{X}_{ij}(t)$ as Equation (8)
        Draw $\omega(t) \sim Bernoulli(\overline{X}_{ij}(t))$
        Update $a_{ij}(t) \leftarrow a_{ij}(t) + \omega(t)$
        Update $b_{ij}(t) \leftarrow b_{ij}(t) + (1 - \omega(t))$
      else
        $\mathcal{M}(T_i(t)) = T_i(t)$
        T_i received a reward $x_{ij}(t,r) = 0$
        Update $\overline{X}_{ij}(t)$ as Equation (8)

algorithms for $K = 1000$ time slots and all results averaged over 100 independent runs. Given the network configuration, we performed a large number of trial processes (10000 trials) to measure the response delays (offloading delays) offered by HNs. With the range of obtained offloading delays (0.087 to 0.45 s), we accordingly selected the maximum permitted latency for all tasks as in the set {0.1, 0.2, 0.3, 0.4}(s) to evaluate the proposed algorithms.

TABLE I: Parameters for simulation

Parameters	Values
Network size $(2 \times M)$	10
Task size, $L(t)$	U[1,15](KB)
$D^{max}(t)$	{0.1, 0.2, 0.3, 0.4} (s)
Data rate r_j	U[0.5,1](MBps)
Processing density of FNs, $\gamma(t)$	U[500,1000] (cycles/bit)
CPU frequency of FNs, $f(t)$	{1.0, 1.5, 2.0, 2.5} (GHz)

B. Evaluation Analysis

Since the most of the existing solutions focus on bandit learning algorithm use ϵ-greedy and UCB, we compare our results with these two techniques.

Fig. 2 depicts the average offloading delay achieved by different bandit learning approaches (i.e., ϵ-greedy, UCB, and proposed TS) over time slots. Notably, our TS-based BLM algorithm can achieve the sub-optimal result compared to the optimal solutions obtained when the information of network is available. This presents the efficiency of the proposed bandit learning using TS to deal with the dynamic environment of fog computing networks.

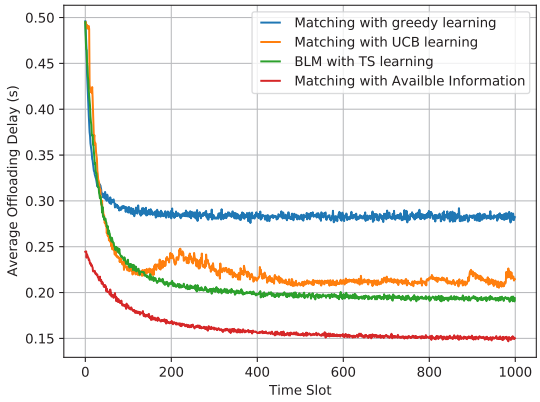

Fig. 2: The average delay offered by three different bandit learning based task offloading algorithms.

The TS-based learning technique presents its out-performance over ϵ-greedy and UCB due to the principle of posterior distribution estimation. Meanwhile, ϵ-greedy and UCB tend to assign the best HNs to tasks in a greedy manner. When the network experiences a large number of time slots, the performance gap between the comparative algorithms are smaller because the network achieves more stable states. In addition, more observations collected over time slots allow TN to select the optimal HNs efficiently.

Fig. 3 presents the probability of selecting the optimal HNs.

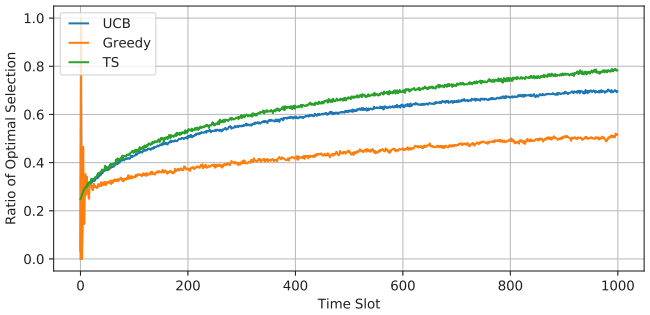

Fig. 3: Ratio of optimal selection

The results show that the matching conflicts occurred during the resource competition have a severe impact on the ratio

of optimal selection for the greedy and UCB algorithms. Meanwhile, our proposed algorithm using TS technique and especially the developed matching conflict avoidance mechanism can efficiently handle the issues. It naturally incorporates the uncertainty or stochasticity in the reward distribution by sampling from a probability distribution. This helps balance exploration and exploitation effectively. While the greedy method tends to be overly exploitative, always choosing the action with the highest estimated value, which may lead to suboptimal decisions if the estimates are inaccurate. Finally, UCB can be overly optimistic and might converge to suboptimal actions.

VI. Conclusions

This paper introduces the TS-based bandit learning for distributed task offloading in the dynamic fog computing-based system. In principle, the algorithm applies the MAB learning empowered by Thompson sampling method to efficiently learn the uncertainty of fog computing environment, thus allowing TN to select the optimal HN for task offloading over time slots. Extensive simulation results demonstrate the the proposed algorithm outperform the benchmark algorithms using ϵ-greedy and UCB learning methods. The advantage of TS method in the RL-MATCH algorithm opens potential direction for extending the current works. For example, matching models such as many-to-one and many-to-many matching can be investigated in large-scale networks.

Acknowledgments

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the Grand Information Technology Research Center support program(IITP-2023-2020-0-01612) supervised by the IITP (Institute for Information & communications Technology Planning & Evaluation)", and Korea Research Fellowship Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (RS-2023-00249687).

References

[1] D. A. Chekired, L. Khoukhi, and H. T. Mouftah, "Industrial iot data scheduling based on hierarchical fog computing: A key for enabling smart factory," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4590–4602, Oct 2018.

[2] H. Tran-Dang, N. Krommenacker, P. Charpentier, and D.-S. Kim, "The internet of things for logistics: Perspectives, application review, and challenges," *IETE Technical Review*, pp. 1–29.

[3] H. Tran-Dang, N. Krommenacker, P. Charpentier, and D. Kim, "Toward the internet of things for physical internet: Perspectives and challenges," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 4711–4736, 2020.

[4] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing - MCC 2012*. ACM Press, 2012.

[5] H. Tran-Dang and D.-S. Kim, "FRATO: Fog resource based adaptive task offloading for delay-minimizing IoT service provisioning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 10, pp. 2491–2508.

[6] M. Aazam, S. Zeadally, and K. A. Harras, "Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities," *Future Generation Computer Systems*, vol. 87, pp. 278–289, Oct. 2018.

[7] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 283–294, 2018.

[8] G. Lee, W. Saad, and M. Bennis, "An online optimization framework for distributed fog network formation with minimal latency," *IEEE Transactions on Wireless Communications*, vol. 18, no. 4, pp. 2244–2258, 2019.

[9] Y. Yang, Z. Liu, X. Yang, K. Wang, X. Hong, and X. Ge, "Pomt: Paired offloading of multiple tasks in heterogeneous fog networks," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8658–8669, 2019.

[10] S. Durand and B. Gaujal, "Complexity and Optimality of the Best Response Algorithm in Random Potential Games," in *Symposium on Algorithmic Game Theory (SAGT) 2016*, pp. 40–51.

[11] H. Tran-Dang and D.-S. Kim, "A survey on matching theory for distributed computation offloading in iot-fog-cloud systems: Perspectives and open issues," *IEEE Access*, vol. 10, pp. 118 353–118 369.

[12] Tran-Dang, Hoa and Kim, Dong-Seong, "Disco: Distributed computation offloading framework for fog computing networks," *Journal of Communications and Networks*, pp. 1–11.

[13] H. Tran-Dang and D.-S. Kim, *Cooperative and Distributed Intelligent Computation in Fog Computing: Concepts, Architectures, and Frameworks*. Springer Nature Switzerland.

[14] H. Tran-Dang, S. Bhardwaj, T. Rahim, A. Musaddiq, and D.-S. Kim, "Reinforcement learning based resource management for fog computing environment: Literature review, challenges, and open issues," *Journal of Communications and Networks*, pp. 1–16.

[15] J. Vermorel and M. Mohri, "Multi-armed bandit algorithms and empirical evaluation," in *Machine Learning: ECML 2005: 16th European Conference on Machine Learning, Porto, Portugal, October 3-7, 2005. Proceedings 16*. Springer, pp. 437–448.

[16] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3-4, pp. 285–294.

[17] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2, pp. 235–256.

[18] O. Chapelle and L. Li, "An empirical evaluation of thompson sampling," *Advances in neural information processing systems*, vol. 24.

[19] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *2012 Proceedings IEEE INFOCOM*, pp. 945–953.

[20] Y. Yang, K. Wang, G. Zhang, X. Chen, X. Luo, and M.-T. Zhou, "Meets: Maximal energy efficient task scheduling in homogeneous fog networks," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 4076–4087.

[21] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Transactions on Communications*, vol. 65, no. 8, pp. 3571–3584.

[22] P. Yang, N. Zhang, Y. Bi, L. Yu, and X. S. Shen, "Catalyzing cloud-fog interoperation in 5g wireless networks: An sdn approach," *IEEE Network*, vol. 31, no. 5, pp. 14–20.

[23] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983.

[24] T. Chen and G. B. Giannakis, "Bandit convex optimization for scalable and dynamic iot management," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 1276–1286.

[25] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for iot devices with energy harvesting," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1930–1941.

[26] S. Misra, S. P. Rachuri, P. K. Deb, and A. Mukherjee, "Multiarmed-bandit-based decentralized computation offloading in fog-enabled iot," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 10 010–10 017.

[27] Z. Zhu, T. Liu, Y. Yang, and X. Luo, "Blot: Bandit learning-based offloading of tasks in fog-enabled networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 12, pp. 2636–2649.

[28] H. Liao, Z. Zhou, X. Zhao, B. Ai, and S. Mumtaz, "Task offloading for vehicular fog computing under information uncertainty: A matching-learning approach," in *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC)*, pp. 2001–2006.