Digital Twin-aided Distributed Contextual Bandit Learning for Computation Offloading in Dynamic Fog Computing Networks

Hoa Tran-Dang and Dong-Seong Kim

***Abstract*—In this paper, we propose a novel approach for parallel task computation in dynamic fog computing networks, leveraging digital twin technology and distributed contextual bandit learning. Our method, digital twin-aided distributed contextual bandit learning (DT-DCBL), utilizes digital twins to create real-time virtual representations of fog network components, capturing their dynamic states and behaviors. This enables accurate modeling of network conditions, device capabilities, and task workloads. We employ Thompson Sampling within the contextual bandit framework to address the exploration-exploitation dilemma, efficiently learning the uncertain environment of the fog network. To optimize task offloading, we incorporate matching theory, ensuring effective pairing between task nodes and helper nodes based on their individual preferences and capabilities. By combining these advanced techniques, DT-DCBL facilitates adaptive and efficient task allocation, significantly reducing task execution latency and enhancing resource utilization in fog computing networks. Our simulation results demonstrate that the proposed approach outperforms traditional methods, achieving superior performance in terms of delay reduction and computational efficiency in dynamic fog environments.**

***Index Terms*—Decentralized task offloading, exploitation and exploration dilemma, fog computing network, multi-armed bandit, stable matching, Thompson sampling.**

I. Introduction

A. Context and Motivations

THE Internet of Things (IoT) has evolved into a critical enabler of intelligent systems, including smart cities [1], smart factories [2], and smart logistics and supply chains [3], [4]. These systems rely on pervasive connectivity among distributed devices to generate and exchange large volumes of data, which are processed to provide intelligent services. However, resource limitations in computation, storage, bandwidth, and energy prevent IoT devices from locally processing complex tasks and large-scale data efficiently.

Manuscript received November 29, 2024; revised August 4, 2025; approved for publication by Lorenz, Pascal, Division 3 Editor, October 10, 2025.

This work was supported in part by the Ministry of Science and ICT (MSIT), Korea, under the Innovative Human Resource Development for Local Intellectualization program, supervised by the Institute for Information and Communications Technology Planning and Evaluation (IITP) (IITP-2025-RS-2020-II201612, IITP-2025-RS-2024-00438430) and Korea Research Fellowship Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (2018R1A6A1A03024003, RS-2023-00249687).

H. Tran-Dang and D.-S. Kim are with department of IT Convergence Engineering, Kumoh National Institute of Technology, Korea, email: {hoa.trandang, dskim}@kumoh.ac.kr.

D.-S. Kim is the corresponding author.

Digital Object Identifier: 10.23919/JCN.2025.000081

Cloud computing has traditionally addressed this limitation by offering elastic resources and services through paradigms such as PaaS, IaaS, and SaaS [5], [6]. Yet, the centralized nature of cloud-based solutions introduces latency and reliability concerns, especially for real-time, latency-sensitive applications, due to physical distance to cloud servers and network fluctuations.

To mitigate these issues, *fog computing networks (FCNs)* have emerged as a decentralized computing paradigm that brings computation closer to the data source [7], [8]. FCNs reduce service latency [9], improve energy efficiency [10], and enhance quality of service/quality of experience (QoS/QoE) by performing computation at intermediate fog nodes. However, realizing the benefits of FCNs necessitates efficient *task offloading strategies*, particularly in *multi-task multi-helper (MTMH)* scenarios where multiple task nodes (TNs) offload tasks to multiple helper nodes (HNs) [11], [12]. The challenge lies in dynamically matching TNs with suitable HNs while minimizing latency and meeting diverse QoS constraints. This challenge is further compounded by the heterogeneity of node capabilities, time-varying resources, and non-uniform task requirements [13].

While centralized optimization-based solutions [14], [15] have been extensively studied, they rely on complete system knowledge, introduce significant computational and communication overhead, and are difficult to scale in dense, heterogeneous FCNs [16]. As a result, research attention has shifted toward decentralized task offloading frameworks.

Game-theoretic methods have been proposed to reduce coordination overhead in decentralized settings [17], [18], yet they often assume partial knowledge of other agents' strategies and rely on idealized equilibrium concepts [19], such as Nash equilibrium [20], which may not effectively capture the mutual preferences and selfish behavior of TNs and HNs.

To address these limitations, *matching theory (MT)* has emerged as a promising tool for modeling and solving decentralized, stable, and preference-aware task offloading problems [21], [22]. Grounded in the deferred acceptance (DA) algorithm [23], MT-based frameworks enable tractable and distributed matching of TNs to HNs based on mutual preferences. However, existing MT-based approaches generally assume that the preferences of TNs and HNs are known in advance, which is often unrealistic in dynamic environments where computing capabilities and task characteristics evolve unpredictably.

To manage this uncertainty, recent work has introduced learning-augmented matching approaches. Digital twin (DT)

1229-2370/26/$10.00 © 2026 KICS

technology has gained traction as a way to alleviate preference uncertainty by constructing virtual replicas of fog nodes to estimate their current resource states (e.g., CPU availability, queueing delays) [24]. Nevertheless, the stochastic nature of task arrivals and their dynamic requirements still demands adaptive learning mechanisms from the TN side to infer which HNs are most suitable for offloading.

This need aligns well with online learning paradigms, particularly *multi-armed bandits (MABs)*, which model the exploration-exploitation trade-off in uncertain decision-making environments [25]. Compared to reinforcement learning (RL) and deep reinforcement learning (DRL) methods [26]–[28], MAB-based approaches offer faster convergence and lower complexity—qualities particularly suitable for resource-constrained fog environments. Among MAB algorithms, Thompson sampling (TS) [29], a Bayesian method that maintains posterior distributions over expected rewards, has shown strong empirical performance and theoretical guarantees [30], [31].

Building on these foundations, this work investigates the following core research problem: *How can we design an efficient, distributed task offloading framework in dynamic, resource-constrained fog computing networks that accounts for mutual preference uncertainties of task nodes and helper nodes?* To address this question, we propose a novel framework–digital twin-aided distributed contextual bandit learning (DT-DCBL)—-which integrates matching theory, TS-based contextual bandit learning, and DT-assisted resource estimation for adaptive and scalable task offloading in dynamic FCNs.

We hypothesize that integrating TS into contextual bandit learning will accelerate convergence and improve decision accuracy in the presence of unknown and evolving preferences between TNs and HNs. Additionally, we postulate that leveraging DT models to estimate HN resource states in real time will substantially reduce preference uncertainty and improve offloading outcomes. Lastly, we expect that the proposed DT-DCBL framework will outperform centralized optimization and game-theoretic baselines in terms of average task latency and offloading success rates, particularly in multi-task multi-helper fog environments characterized by high variability and limited observability.

B. Paper Contributions

The primary contributions of this paper are as follows. First, we develop a DT-enabled architectural framework that provides real-time, fine-grained visibility into the dynamic states and resource conditions of fog nodes. This facilitates accurate estimation of HN availability and responsiveness, which is crucial for adaptive task offloading in highly dynamic environments. Second, we design a distributed contextual bandit learning algorithm that leverages TS to enable task nodes to learn optimal offloading strategies under uncertainty. The algorithm is specifically tailored to the decentralized and resource-constrained nature of fog computing networks, with emphasis on minimizing computational overhead while maximizing resource utilization. Third, we conduct comprehensive simulations to evaluate the proposed DT-DCBL framework, demonstrating substantial improvements in offloading efficiency, average task latency, and overall network performance when compared to state-of-the-art baseline approaches.

C. Paper Organization

The rest of this paper is organized as follows. Section 2 highlights the typical related works regarding the bandit learning-based algorithms in different fog computing environments. Section 3 introduces the system model and formulates the task offloading problem for the heterogeneous dynamic FCNs. Section 4 proposes and describes the design of proposed model. Section 5 shows the simulation results and comparative analysis. Section 6 concludes the paper and discusses the open directions to extend the current work.

II. Related Work

This section reviews existing task offloading approaches in fog computing based on bandit learning, with a focus on how these methods address dynamic system conditions, decentralized decision-making, and uncertain feedback environments.

Bandit learning has been extended to address bandit convex optimization (BCO), which deals with time-varying objectives and constraints under limited feedback. The foundational works of Flaxman et al. [32] and Kumagai et al. [33] were adapted to fog computing in BanSaP algorithms [34], [35], designed for online offloading in dynamic IoT networks. BanSaP algorithms maintain sublinear dynamic regret and support long-term constraint satisfaction without requiring full gradient information. Through simulation, they demonstrated superior adaptability over traditional online convex optimization in fluctuating environments.

D2CIT [36] applies a non-stationary ε-greedy MAB algorithm to offload subtasks over fog and cloud layers based on directed acyclic task graphs (DATGs). Each fog node is modeled as an arm, and subtasks are scheduled in a way that adapts to time-varying system loads and processing delays. Compared to benchmark policies, D2CIT achieved up to 17% latency reduction and 59% speedup.

The BLOT algorithm [37] is designed for fog networks with abrupt state changes and delayed feedback. It minimizes long-term cost (latency, energy, switching) by modeling the task offloading as a non-stationary bandit problem. A sliding-window estimator detects changes and adapts to the environment. Despite its simplicity and theoretical robustness, BLOT was only validated on a small-scale network.

Wang et al. [38] employed a combinatorial MAB (CMAB) framework for joint fog node and spectrum scheduling, treating each (node, channel) pair as an arm. The learning algorithm estimates delay and reliability using upper confidence bounds. The model supports multi-arm selection and shows better performance in dynamic network topologies than classic UCB-based schemes.

To handle uncertainty and adversarial threats in vehicular fog computing (VFC), Cho et al. [39] used an adversarial MAB model. It employs input-dependent exploration bonuses

and implicit score patching to minimize offloading regret in environments with privacy risks, attacks, or node failures. The method adapts without strong stationarity assumptions and remains robust under volatile conditions.

Zhu et al. [40] proposed a CMAB framework under stochastic and non-stationary channel conditions. Their WFCUCB algorithm balances exploration-exploitation under mixed arm reward distributions and achieves rapid convergence and low latency without assuming prior knowledge.

Sun et al. [41] introduced task replication in VFC using CMAB to minimize delay. Each task is offloaded to multiple fog nodes, and redundant replicas are terminated upon the first successful execution. The learning algorithm estimates delay distributions and adaptively decides the replication count. Results showed a 30% reduction in average delay and higher task completion rates.

Shabir et al. [42] extended task replication with resource-awareness. Their algorithm selectively replicates tasks based on node reliability and system load, improving the task delivery ratio and reducing system overhead compared to the method in [41].

In [43], contextual MAB is used to exploit sensor heterogeneity and mobility in urban fog-assisted wireless sensor networks. The algorithm learns context-dependent delay profiles of fog nodes and adapts offloading policies accordingly. This model achieved significant improvements in delay and reliability in dynamic WSN environments.

Zhang et al. [44] developed calibrated contextual bandit learning (CCBL) for user-side task offloading in ultra-dense MEC systems. Users independently learn delay models of base stations using local context and predict neighboring users' actions to avoid conflicts. The method relies on approachability theory for convergence and supports a scalable decentralized architecture.

In [45], the learning-matching algorithm combines online learning and matching theory to solve a one-to-many matching problem in fog networks. Preferences are estimated based on observed feedback, and a pricing mechanism resolves conflicts among competing task nodes. LMA achieves bounded regret without requiring centralized coordination or global system knowledge.

While the above studies present significant progress, they often assume either full observability of fog node states, rely on stationary environments, or focus on centralized or single-node task offloading strategies. In contrast, our proposed DT-DCBL approach integrates DT-based state estimation with contextual bandit earning in a decentralized setting. This allows for real-time task offloading under uncertainty and matching conflicts in large-scale, dynamic fog networks. To further clarify the contribution, Table I compares DT-DCBL with representative approaches.

TABLE I
COMPARISON OF DT-DCBL WITH RELATED METHODS.

Method	Context	DT	Decentralized	Matching	Dynamic
UCB / ϵ-Greedy	✗	✗	✗	✗	✗
CMAB [38], [41]	✗	✗	✗	✗	✓
CCBL [44]	✓	✗	✓	✗	✓
BLOT [37]	✗	✗	✗	✗	✓
DT-DCBL	✓	✓	✓	✓	✓

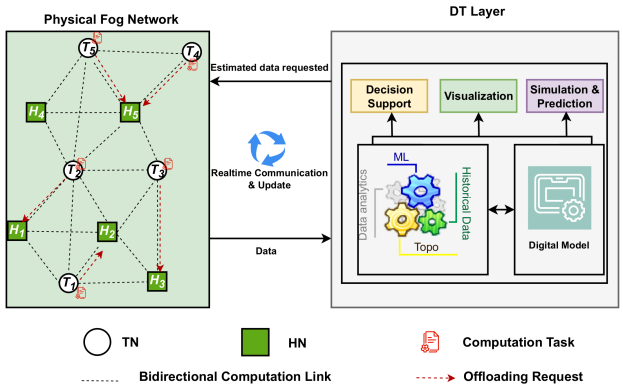


Fig. 1. Task offloading in FCN with the assistance of DT.

III. System Model

A. Fog Computing Network

In the general system architecture, a FCN consists of M TNs and N 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 to investigate the performance of distributed task offloading in form of one-to-one matching with bandit learning integration. Notably, scenarios with unequal number of TNs and HNs can be deduced or extended from the current network configuration. For example, as $M > N$, a part of tasks that are unable to be processed by HNs are managed by the cloud. Indeed, as $M < N$, there exist M HNs with the most ample computing capability selected for offloading the tasks from M TNs. Furthermore, as a HN is able to process multiple tasks via, for example, parallel virtual machines, it is worth to investigate the many-to-one matching model for task offloading as extension works.

In this work, we define the set of TNs and HNs as $\mathcal{T} = \{T_1, \cdots, T_i, \cdots, T_M\}$ and $\mathcal{H} = \{H_1, \cdots, H_j, \cdots, H_M\}$, 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. We adopt a time-slot model where the total time period is divided into K discrete intervals, the set of which is denoted as $\mathcal{S} = \{1, \cdots, t, \cdots, K\}$. At the t^{th} slot, each TN i generates a corresponding task $T_i(t)$, which is represented by a tuple $T_i(t) = \{s_i(t), \gamma_i(t), D_i^{\max}(t)\}$, where $s_i(t)$ 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 not split arbitrarily, thus each task T_i should be either allocated as one whole piece to one neighboring HN or processed locally.

The unfinished tasks are assumed to be cached in a first-in first-out (FIFO) queue at each FN. Due to the limited computation and storage resource within one individual FN, the tasks processed locally usually experience high latency, which

degrades QoS and QoE. To enable low-latency processing, TNs may offload some of its computation burdens to the nearby HNs. These HNs typically posses more computation and storage resources and are deployed to help other TNs on demand.

Meanwhile, the computing resource state of H_j is represented by CPU frequency $f_j(t)$ (cycles/s), mean computing service rate μ_j (bits/s) and waiting in queue $q_j(t)$ (bits), which are assumed to vary over different time slots.

Table II summarizes the key variables and notations used in this paper.

TABLE II
NOTATIONS AND VARIABLES.

Symbols	Definitions
FN, TN, HN	Fog node, Task node, Helper node
$\mathcal{T}$, $\mathcal{H}$	Set of TNs (tasks), set of HNs
M	Number of TNs, Number of HNs in FCN
T_i, H_j	TN i, HN j
t	The t^{th} time slot
$T_i(t)$	The task generated by TN i in time slot t
$f_j(t)$	CPU frequency of H_j in time slot t (cycles/s)
$\hat{f}_j(t)$	CPU frequency deviation of H_j in time slot t
$s_i(t)$	The task size of $T_i(t)$ (bits)
$\gamma_i(t)$	Computation complexity of $T_i(t)$ (cycles/bit)
$D_i^{\max}(t)$	Maximum permitted latency for $T_i(t)$
$D_{ij}(t)$	Latency for executing $T_i(t)$ by HN H_j
$\delta_{ij}^{tx}(t)$	Transmission delay from T_i to H_j
$R_{ij}(t)$	Data rate between T_i and H_j
$\delta_j^w(t)$	Waiting delay in the queue of H_j
$q_j(t)$	Queue length of H_j in time slot t (bits)
$\hat{q}_j(t)$	Queue length deviation of H_j in time slot t (bits)
$\mu_j(t)$	Processing rate of H_j in time slot t (bits/s)
$\hat{\mu}_j(t)$	Processing rate deviation of H_j in time slot t (bits/s)
$\delta_{ji}^p(t)$	Delay to process T_i by H_j

B. Digital Twin Model

We consider a centralized DT layer hosted at a fog server, which provides services to replicate and simulate the behavior of physical fog nodes (both TNs and HNs). The DT services maintain a virtual replica for each physical fog node, encapsulating its hardware configuration, historical performance data, and real-time operational state. Through a real-time update and control mechanism, the DT layer synchronizes with the physical system and supports estimation and decision-making functionalities [24].

In particular, each HN H_j has a corresponding digital counterpart DT_j, represented as:

$$DT_j = \{(f_j, \hat{f}_j), (\mu_j, \hat{\mu}_j), (q_j, \hat{q}_j)\},$$

where:

- f_j, μ_j, and q_j are the estimated (simulated) values of CPU frequency, mean computing service rate, and queue length, respectively, as predicted by the DT model.
- $\hat{f}_j$, $\hat{\mu}_j$, and $\hat{q}_j$ denote the deviations due to modeling and simulation inaccuracies between the DT and the physical world.

In our work, we assume that deviations are negative, i.e., the DT tends to overestimate the capabilities of the physical HNs. Therefore, the actual (ground-truth) values on the physical HNs are calculated as:

$$f_j^{\text{actual}} = f_j - \hat{f}_j, \quad \mu_j^{\text{actual}} = \mu_j - \hat{\mu}_j, \quad q_j^{\text{actual}} = q_j - \hat{q}_j.$$

TNs query the DT layer to retrieve the estimated system states of HNs along with the associated deviation values. This enables TNs to make informed offloading decisions based on a predictive model of system behavior under uncertainty. Notably, the DT layer operates as a shared service accessible to all TNs, and the physical HNs do not directly share their internal parameters with TNs.

C. 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 by $D_{ji}(t) = \delta_{ij}^{tx}(t) + \delta_j^w(t) + \delta_{ji}^p(t)$, where $\delta_{ij}^{tx}(t)$ is the transmission latency, $\delta_j^w(t)$ is the waiting delay in queue of H_j, and $\delta_{ji}^p(t)$ is the execution latency by H_j. Given the data rate $r_{ij}(t)$ (bits/s) between T_i and H_j at time slot t, we can achieve $\delta_{ij}^{tx}(t) = (\alpha_{ij}(t)s_i(t))/r_{ij}(t)$.

The estimated latency of H_j to execute T_i is given by $\tilde{\delta}_{ji}^p(t) = \frac{\alpha_{ij}(t)\gamma_i(t)}{f_j(t)}$. The latency gap $\Delta_{ji}^p(t)$ between real value and DT estimation can be expressed as:

$$\Delta_{ji}^p(t) = \frac{\alpha_{ij}(t)\gamma_i(t)\hat{f}_j(t)}{f_j(t)(f_j(t) - \hat{f}_j(t))}. \tag{1}$$

As a result the actual latency for executing at H_j can be expressed as $\delta_{ij}^p(t) = \tilde{\delta}_{ji}^p(t) + \Delta_{ji}^p(t)$. The estimated waiting delay is measured by H_j as $\delta_j^w(t) = \frac{q_j(t) - \hat{q}_j(t)}{\mu_j(t) - \hat{\mu}_j(t)}$.

For TNs, the objective of offloading their tasks to HNs is to minimize the long-term average delay $\overline{D}$, which is defined as $\overline{D} = \lim_{K\to\infty} \frac{1}{K} \sum_{t=1}^{K} \sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_{ij}(t) D_{ij}(t)$.

We also define Ω to be the set of task suffering from outage and can be expressed as per $\Omega = \{T_i \in \mathcal{T} | \alpha_{ij}(t) = 1 \ \& \ D_{ij}(t) > D_i^{\max}; \forall j \in \mathcal{H}\}$.

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

$$\begin{aligned} \textbf{P:} \quad & \min_{\alpha_{ij}(t)} \overline{D} \quad \& \quad \min_{\alpha_{ij}(t)} |\Omega| \\ \text{s. t.} \quad & C_1 : \alpha_{ij}(t) \in \{0, 1\}, \forall \{i, j, t\} \in \{\mathcal{T}, \mathcal{H}, \mathcal{S}\}, \\ & C_2 : \sum_{i=1}^{\mathcal{T}} \alpha_{ij}(t) \leq 1, \forall j \in \mathcal{H}, \forall t \in \mathcal{S}, \\ & C_3 : \sum_{j=1}^{\mathcal{H}} \alpha_{ij}(t) \leq j, \forall i \in \mathcal{T}, \forall t \in \mathcal{S}. \end{aligned} \tag{2}$$

The objective is to minimize the total offloading delay and the number of outages. Here, the constraints C_1, C_2, and C_3 guarantee that at each time slot t, a task can be offloaded to only one HN, and a HN can process at one task from TNs.

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. Proposed Model Description

A. OTO Matching Model for MTMH Problems

Given the constraints C_2 and C_3 of problem **P**, the task offloading problem can be modeled as a 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 constructing 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 a stable matching, each node forms a PL over its potential partners based on utility functions that reflect the optimization objective—minimizing task delay and outage. For each TN T_i, the utility of offloading to HN H_j is given by an unknown preference value $u_{ij}(t)$, and vice versa, $v_{ji}(t)$ denotes the preference of H_j for T_i. These are derived via the utility functions $f_{ij}(\cdot)$ and $g_{ij}(\cdot)$, respectively, and are expressed as:

$$u_{ij}(t) = v_{ji}(t) = \frac{1}{1 + e^{-(D_i^{\max}(t) - D_{ij}(t))}}. \tag{3}$$

Intuitively, this sigmoid-based function prioritizes matches where the expected offloading delay $D_{ij}(t)$ is significantly below the deadline $D_i^{\max}(t)$, resulting in a higher utility value. A larger $u_{ij}(t)$ implies a higher preference of T_i for H_j, and similarly for $v_{ji}(t)$. These utilities are initially unknown to the nodes and are learned iteratively using contextual bandit learning. This symmetric utility modeling assumes that both TNs and HNs evaluate each other based on the same offloading delay conditions, promoting fairness and stability in the matching process.

B. Contextual Bandit Learning Model and Algorithm

In the multi-armed bandit (MAB) framework, TNs act as players and HNs serve as arms. The objective is for each task node T_i to learn the acceptance probabilities $u_{ij}(t)$ and $v_{ij}(t)$, which represent the likelihood of successfully offloading a task to helper node H_j at time slot t. We propose a contextual bandit learning method that extends TS to our multi-stage, matching-constrained environment as illustrated in Fig.2.

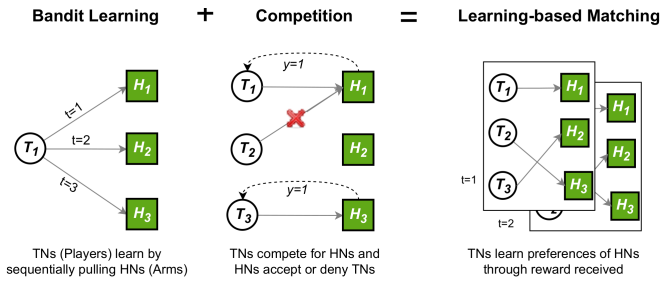


Fig. 2. Contextual bandit learning based two-sided matching.

In each time slot $t = 1, 2, \cdots, K$, the offloading decision-making is repeated over multiple learning rounds $r = 1, 2, \cdots, R$. In each round, a task node T_i selects a helper node H_j to offload its task to. If multiple task nodes attempt to offload to the same helper, a matching conflict occurs. Only the task node most preferred by the helper is accepted and receives a reward $y_{ij}(t)[r] = 1$; others receive $y_{ij}(t)[r] = 0$.

Let $\theta_{ij}(t)[r]$ represent the probability that T_i's offloading request to H_j is accepted in round r of time slot t. We define:

$$P(\theta_{ij}(t)[r]) = P(y_{ij}(t)[r] = 1 \mid D_{ij}(t)[r-1]),$$

where $D_{ij}(t)[r-1]$ is the history of contextual data and feedback collected up to round $r-1$.

To estimate $\theta_{ij}(t)[r]$, we use a logistic regression model with the contextual features $D_{ij}(t)$ (reflecting task-helper dynamics) and $D_i^{\max}(t)$ (reflecting the task's local processing difficulty). The probability is modeled as:

$$\theta_{ij}(t)[r] = \sigma(w_{ij0} + w_{ij1} D_{ij}(t) + w_{ij2} D_i^{\max}(t)),$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function. The regression weights $w_{ij} = \{w_{ij0}, w_{ij1}, w_{ij2}\}$ are treated as random variables with Gaussian posterior distributions $w_{ijk} \sim \mathcal{N}(m_{ijk}, q_{ijk}^{-1})$. This probabilistic modeling enables posterior sampling of reward parameters, conforming to the contextual TS principle [46]. The learning is implemented using a batch logistic regression algorithm (Algorithm 1) with Laplace approximation and maximum likelihood estimation (MLE) to update the posterior mean and precision of the

weights. This sampling-based method allows each task node to make adaptive, uncertainty-aware offloading decisions over time and learning rounds. For theoretical analysis and regret bounds, refer to Appendix B.

Algorithm 1: Logistics regression with batch updates

Input: Regularization parameter $\lambda \in (0, 1)$.
Output: Distribution of weights $w_k \sim \mathcal{N}(m_k, q_k^{-1})$
Initialization: $m_k[0] = 0$, $q_k[0] = \lambda$, $\forall k = 0, 1, 2$
for $t = 1, 2, \cdots, T$ **do**
  Get a new batch of n training data samples $(\mathbf{x}_j; y_j)$, where $\mathbf{x} = \{D, D^{\max}\}$, $\forall j = 1, \cdots, n$.
  Find $\mathbf{w}$ to minimize (or MLE): $\frac{1}{2}\sum_{k=0}^{2} q_k(w_k - m_k)^2 + \sum_{j=1}^{n} \log(1 + \exp(-y_j \mathbf{w}^T \mathbf{x}_j))$.
  Update:
  $m_k[t] \leftarrow w_k[t]$;
  $q_k[t] \leftarrow q_k[t-1] + \sum_{j=1}^{n} x_{kj}^2 p_j(1 - p_j)$, where x_{kj} is x_k of the j^{th} sample, and $p_j = (1 + \exp(-y_j \mathbf{w}^T \mathbf{x}_j)^{-1}$

C. DT-DCBL Algorithm

Algorithm 2 details the execution of DT-CBLM procedure.

Algorithm 2: DT-CBLM Algorithm (for every T_i)

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 $t = 0$: $\mathcal{M}(T_i(0)) = H_j(0)$.
for $t = 1, 2, \cdots, K$ **do**
  for $\tau = 1, 2, \cdots, R$ **do**
    Get $\{(f_j, \hat{f}_j), (\mu_j, \hat{\mu}_j), (q_j, \hat{q}_j)\} \leftarrow DT_{ij}(t)[\tau]$, $\forall H_j \in \mathcal{H}$;
    Get context: $\mathbf{x}_{ij}(t)[\tau] := \{D_{ij}(t)[\tau], D_i^{\max}[t]\}$;
    Sample the weights $\mathbf{w}_{ij}(t)[\tau]$ by Algorithm 1;
    Get $\theta_{ij}(t)[\tau] = \sigma(w_{ij0}(t)[\tau] + w_{ij1}(t)[\tau] D_{ij}(t)[\tau] + w_{ij2}(t)[\tau] D_i^{\max}(t))$;
    Draw $\pi_i(t)[\tau] \sim Bernoulli(\lambda)$;
    if $\pi_i[t][\tau] = 0$ **then** // Exploration
      Construct potential matching set of T_i
      $\mathcal{P}_i(t)[\tau] := \{H_j : v_{ji}(t)[\tau] \geq v_{ji'}(t)[\tau]\}$
      where $\mathcal{M}(T_{i'}(t)[\tau - 1]) = H_j(t)[\tau]$;
      Pull $H_j \in \arg\max_{H_j \in \mathcal{P}_i(t)[\tau]} \theta_{ij}(t)[\tau]$;
    else // Exploitation
      Pull $H_j(t)[\tau] = \mathcal{M}(T_i(t)[\tau - 1])$;
    if p_i *wins matching confict* **then** // If accepted
      $H_j(t)[\tau] = \mathcal{M}(T_i(t)[\tau])$;
      $y_{ij}(t)[\tau] = 1$;
    else // If rejected
      $\mathcal{M}(T_i(t)) = T_i(t)$;
      $y_{ij}(t)[\tau] = 0$;
    Update
    $\mathcal{D}_{ij}(t) \leftarrow \{D_{ij}(t)[\tau], D_i^{\max}(t); y_{ij}(t)[\tau]\}$;

For every task–helper pair (T_i, H_j) the algorithm maintains a posterior $P(\theta_{ij})$ that quantifies the probability of a successful offload. At the beginning of each time slot, each task node T_i queries the DT layer to obtain the predicted delay $\hat{D}_{ij}(t)$ and the corresponding uncertainty bounds $\delta_{ij}(t)$. These are used to guide the offloading decision, acknowledging the discrepancy between prediction and the eventual true delay. Once a minimal amount of data is gathered, T_i samples the logistic-regression weights via Algorithm 1 and evaluates $\theta_{ij}(t)$ to guide arm selection. To mitigate repeated collisions, T_i keeps a potential match set that excludes any helper that rejected it in the previous round. A Bernoulli exploration variable $\pi_i(t) \sim \text{Bernoulli}(\lambda)$ balances exploration and exploitation: if $\pi_i(t) = 0$, T_i probes the highest-valued helper in its current match set; if $\pi_i(t) = 1$, it re-uses the helper chosen in the preceding round. After all TNs issue proposals, every helper H_j accepts the most preferred request according to its current ranking. The realised reward $y_{ij}(t)$ is then appended to the history $\mathcal{D}_{ij}(t)$, and the posterior over θ_{ij} is updated for the next learning round.

D. Complexity Analysis

The proposed BLM algorithm is based on iteration learning mechanism for obtain the optimal solution within each time slot, thus incurring an additional delay to reach the global optimum (i.e., stable matching). Regarding the computational complexity of the algorithm, in each iteration, the preference list of TNs and temporary fairing between TNs and HNs is updated and their stable matching is evaluated based on the DA algorithm. The number of rounds R and the network dimension $M * N$ determine the number of calculations required to update the TN-HN matching outcome. Given that $M = N$ in our proposition, the proposed algorithm has an overall complexity of order $O(R * M^2)$ per time slot.

E. Analysis of Stable and Pareto-Optimal Matching

Definition 4.4 (Stable Matching): A one-to-one matching μ between the set of Task Nodes $\mathcal{T}$ and Helper Nodes $\mathcal{H}$ is said to be *stable* if there exists no blocking pair (i, j) such that:

1) i prefers j to their current match $\mu(i)$, i.e., $u_{ij} > u_{i\mu(i)}$,
2) and j prefers i to their current match $\mu(j)$, i.e., $u_{ij} > u_{j\mu(j)}$.

Definition 4.5 (Pareto-Optimal Matching): A matching μ is said to be *Pareto-optimal* if there does not exist another matching μ' such that:

- For all TNs and HNs, $u_i(\mu'(i)) \geq u_i(\mu(i))$, and
- There exists at least one agent i such that $u_i(\mu'(i)) > u_i(\mu(i))$.

Proposition 4.1: The stable matching outcome μ^* produced by the proposed distributed contextual bandit matching algorithm is Pareto-optimal within the domain of one-to-one matchings.

Proof We proceed by contradiction. Suppose that the matching μ^* is stable but not Pareto-optimal. Then, there exists another matching μ' such that:

- For all $i \in \mathcal{T} \cup \mathcal{H}$: $u_i(\mu'(i)) \geq u_i(\mu^*(i))$, and

- For at least one i: $u_i(\mu'(i)) > u_i(\mu^*(i))$.

This implies that at least one pair (i, j) is matched in μ' but not in μ^*, and both i and j strictly prefer each other over their partners in μ^*. That is:

$$u_{ij} > u_{i\mu^*(i)} \quad \text{and} \quad u_{ij} > u_{j\mu^*(j)},$$

which by the definition of stability, constitutes a blocking pair in μ^*. This contradicts the assumption that μ^* is stable.

Therefore, no such matching μ' can exist, and hence, the matching μ^* is Pareto-optimal within the set of all stable one-to-one matchings.

V. Simulation Results and Evaluation Analysis

A. Simulation Environment Configuration

We evaluate the proposed algorithms in fog environments where the network size ($2 \times M$) includes 10 and 20 nodes. In particular, to justify the impact of DT, we conduct a comparative analysis under two different deviation levels: 1% and 5%. These deviations reflect the error due to modeling and simulation in the DT layer and are assumed to remain fixed during each simulation run. Table III summarizes the key parameters and values used in the simulation setup, where U[x, y] indicates the uniform distribution over the interval $[x, y]$. For each scenario, all algorithms are executed over $K = 10000$ time slots, and the results are averaged over 100 independent runs to ensure statistical robustness. Given the network configuration, we performed extensive trial processes (10000 trials) to measure the offloading delays offered by the helper nodes (HNs). Based on the observed range of offloading delays (0.087 to 0.45 seconds), we selected the set $\{0.1, 0.2, 0.3, 0.4\}$ (seconds) as the maximum permitted latency values for tasks to evaluate the effectiveness of the proposed algorithms under different latency constraints.

TABLE III
Parameters for Simulation.

Parameters	Values
Network size ($2 \times M$)	{10, 20}
Task size, $L(t)$	U[1,15] (KB)
Maximum permitted latency, $D^{\max}(t)$	{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)
Deviations of f, μ, q	{1%, 5%}

B. Comparative Algorithms

Since most existing solutions for task offloading in dynamic networks rely on bandit learning algorithms such as ϵ-greedy and UCB, we compare our proposed algorithm, DT-DCBL, which leverages contextual TS—against these two strategies used in prior works [36], [37], and [38]. Notably, the algorithm in [38] employs a UCB-based CMAB approach to select efficient HNs. In contrast, our TS-learning approach incorporates digital twin feedback with deviation modeling, enabling more accurate and adaptive learning under uncertainty.

C. Evaluation Metrics

We evaluate the performance of proposed algorithms in four key metrics including average offloading delay, cumulative learning regret, successful offloading ratio of tasks, and cumulative matching unstability under the impact of network size and error deviation of DT.

D. Evaluation Analysis

Fig. 3 illustrates the average offloading delay achieved by different bandit learning approaches—including ϵ-Greedy, UCB, CMAB, and our proposed TS-based learning with digital twin support—across 10,000 time slots and under varying network sizes (10 and 20 nodes). Across all algorithms, the

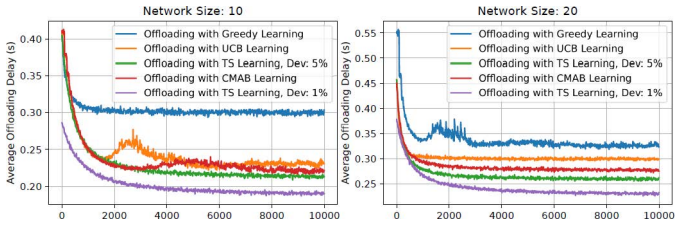


Fig. 3. Average offloading delay with different network size.

offloading delay decreases steadily as time progresses, reflecting the adaptive learning behavior of each method. Among them, the proposed TS-based learning approach consistently achieves the lowest delay, particularly in early time slots, due to its efficient exploration-exploitation balance using posterior sampling.

In contrast, CMAB exhibits competitive performance but remains slightly behind TS due to its reliance on contextual estimation without direct uncertainty modeling. UCB and ϵ-Greedy, while simpler, tend to converge more slowly and suffer from suboptimal selections, as they often exploit prematurely or lack refined contextual understanding.

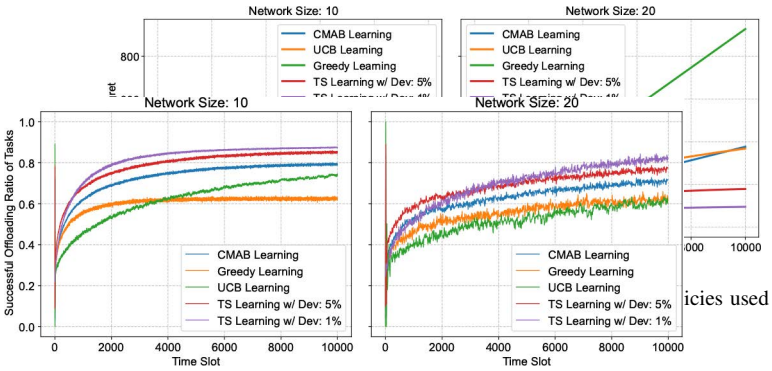
As shown in the right subfigure of Fig. 3, increasing the network size from 10 to 20 nodes leads to a noticeable rise in average offloading delay for all strategies. This increase is attributed to the heightened complexity and contention in larger networks. Notably, the proposed TS-based approach maintains its advantage even in these more challenging scenarios, demonstrating strong scalability and robustness.

Furthermore, the inclusion of deviation levels (1% and 5%) for TS Learning provides additional insights into performance sensitivity. Lower deviation leads to more accurate reward estimation and hence, lower delay, reaffirming the benefit of tighter confidence in contextual observations.

We also analyze the performance of the proposed learning-based offloading algorithms using the cumulative learning regret (LR) metric, which quantifies the performance gap between the learned offloading decision and an ideal offloading decision with full information. Specifically, the regret is defined as the expected cumulative difference in offloading delay between the decisions made with incomplete information and those made with perfect knowledge:

$$LR = \mathbb{E}\left\{\sum_{t=1}^{K}\sum_{i=1}^{M}\left(\alpha_{ij}(t)D_{ij}(t) - \alpha_{ij^*}(t)D_{ij^*}(t)\right)\right\}, \quad (4)$$

where $T_i = \mathcal{M}(H_j)$ denotes the task assignment to helper H_j under learned matching, while $T_i = \mathcal{M}(H_{j^*})$ represents the optimal matching based on complete information.



icies used

As shown in Fig. 4, the proposed TS-based learning approach (with DT assistance) achieves the lowest cumulative regret across both small and large network sizes. This confirms its strong learning efficiency and adaptability. The Greedy algorithm incurs the highest cumulative regret in both settings due to its exploitation-only nature, which fails to adequately explore better offloading opportunities. UCB performs moderately better than Greedy by incorporating uncertainty-based exploration, but still suffers from limited contextual modeling. CMAB learning achieves improved regret performance over UCB and Greedy by leveraging contextual information across multiple arms. However, it still falls short of the TS-based approach, especially in the larger network (size 20), due to its slower convergence over high-dimensional action spaces and increased likelihood of action collisions. The proposed TS-based learning method excels due to its Bayesian sampling framework, where decisions are made according to the posterior probability of each action being optimal. This results in efficient exploration that prioritizes promising actions, while quickly eliminating suboptimal ones. In contrast to UCB and CMAB—which either overly explore or face scalability issues—TS efficiently balances exploration and exploitation based on real-time uncertainty estimation.

Moreover, the impact of DT modeling is evident in the results: TS learning with a 1% deviation consistently outperforms that with 5% deviation, highlighting that more accurate system state predictions lead to better matching decisions and lower regret. This confirms the critical role of precise DT support in accelerating convergence and enhancing offloading effectiveness in dynamic fog environments. The performance gap between the algorithms also widens with network size. For instance, the regret difference between CMAB and TS becomes significantly larger in the 20-node network, indicating that the TS method is more robust to network scaling and can effectively manage increased contention and complexity. This scalability advantage is vital for real-world deployment in dense edge computing systems.

Fig. 5 illustrates the successful offloading ratio, defined as the proportion of tasks successfully offloaded and completed within their respective deadlines. This metric directly reflects the system's ability to make timely and resource-aware offloading decisions under dynamic and partially observable environments.

Fig. 5. Successful offloading ratio.

The results reveal that the proposed TS-based learning approach, particularly when integrated with DT models, consistently achieves the highest successful offloading ratio across both network sizes. This indicates its superior capability in dynamically learning optimal task-helper mappings while accounting for uncertain and time-varying system states. In contrast, the Greedy algorithm exhibits the lowest offloading success due to its purely exploitative strategy. By always selecting the seemingly best current option without exploring alternatives, it suffers from repeated task collisions and resource contention, especially in the larger network. As a result, many tasks fail to meet their deadlines due to oversubscription at popular nodes. The UCB algorithm performs slightly better by incorporating an exploration term in its decision-making. However, its upper-confidence bounds can become overly optimistic, leading to repeated selections of suboptimal nodes in early stages. This inefficiency becomes more pronounced as the network scales, resulting in a flatter learning curve and a relatively modest offloading ratio. CMAB learning outperforms both UCB and Greedy by leveraging contextual features to model arm utilities more precisely. However, it remains sensitive to high-dimensional context spaces and exhibits fluctuations in the success ratio due to instability in estimating context-action mappings. Its performance plateaus earlier than TS-based methods and remains inferior across both small and large networks. The TS learning algorithm, particularly when combined with DT assistance, demonstrates robust and stable improvements. Its success stems from probabilistic sampling of actions based on posterior distributions over rewards, enabling a balanced and informed exploration-exploitation tradeoff. This approach inherently mitigates overconfidence in early decisions and reduces the likelihood of persistent conflicts or suboptimal mappings.

Furthermore, the impact of digital twin model accuracy is evident: the variant with 1% deviation outperforms the 5% deviation variant throughout the simulation horizon. More accurate DT predictions enable the TS learner to better anticipate the availability and capacity of helper nodes, thus reducing the frequency of resource collisions and improving task delivery timeliness. Notably, as the network size increases (from 10 to

20), the performance gap between algorithms widens. This underscores the scalability and adaptability of the TS+DT approach in high-density edge environments, where learning efficient offloading strategies amidst frequent contention and dynamic load is significantly more challenging.

Regarding the stable matching performance, Fig. 6 reports the cumulative matching unstability, which is defined as the number of unstable matchings over time slots.

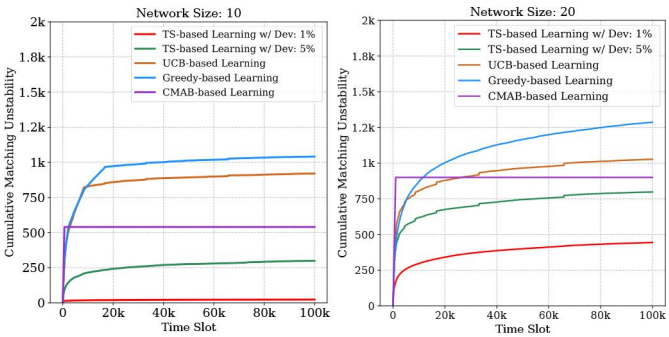


Fig. 6. Cumulative matching unstability with different network size

As shown in the simulation results, our TS-based learning algorithm shows the least matching unstability. Meanwhile, the greedy and UCB based learning solutions converge much slower and explore more to find a stable matching. The CMAB learning enables better performance than the greedy and UCB approach due to learning multiple arm distributions simultaneously. Particularly, the TS algorithm explicitly models uncertainty by sampling from a distribution of possible values for each arm. This allows it to adapt to changes in the environment and handle situations where the true reward distributions are not well-known. In the presence of uncertainty, the greedy and UCB based learning approaches rely on point estimates of the mean or upper confidence bound, respectively, which may lead to suboptimal decisions if the estimates are inaccurate. Increasing the network size from 10 to 20 nodes results in overall higher cumulative matching unstability for all algorithms. However, TS-based learning methods remain the most stable and successful.

Our algorithm is coupled by DT for estimating the resource states and handling system uncertainty, while TS component assists TNs in efficiently selecting suitable HNs through sequential learning. In order to disentangle the impact of each module and verify the synergy of combining both DT and TS, we conducted an ablation study by comparing three variants: (i) Full TS + DT (our proposed method), (ii) TS without DT, and (iii) DT with UCB. As illustrated in Fig. 7, the proposed TS + DT algorithm consistently achieves superior performance across all metrics, including the lowest task offloading delay, the lowest cumulative regret, and the highest successful offloading rate over 10,000 time slots. The variant using TS without DT demonstrates higher regret and slower adaptation due to its lack of predictive modeling of resource dynamics. Meanwhile, DT with UCB benefits from DT's estimation capability but is limited by UCB's less efficient exploration strategy compared to TS. These findings confirm that both DT and TS contribute significantly and complement one another effectively in enhancing offloading performance.

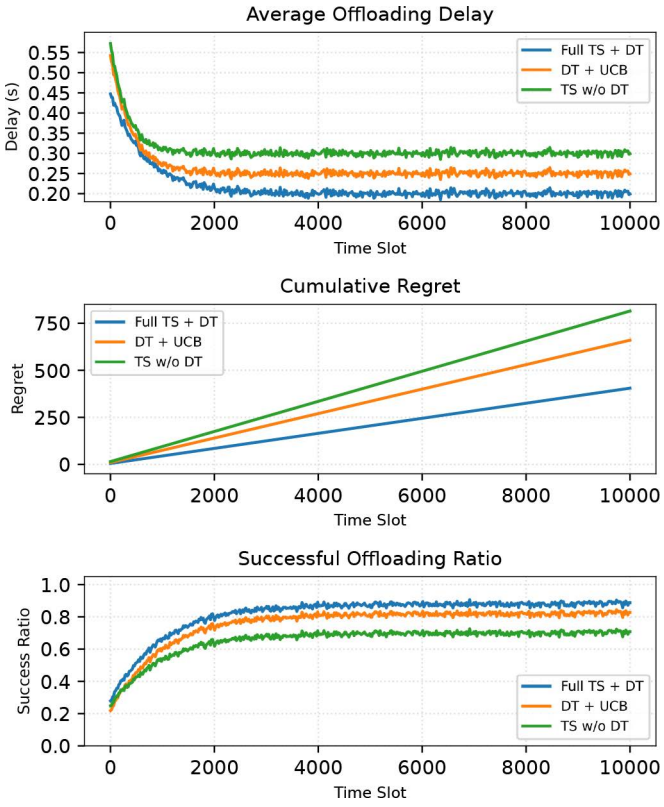


Fig. 7. Performance comparison of DT-TS, TS-only, and DT-UCB algorithms.

VI. Conclusions

This paper presents an innovative approach for computation offloading in dynamic fog computing networks, leveraging DT technology and distributed contextual bandit learning. The integration of DT enables more accurate and dynamic decision-making processes for task offloading. The proposed distributed contextual bandit learning framework addresses the exploration-exploitation trade-off effectively, ensuring optimal resource utilization and reduced latency in fog networks.

Our extensive simulations and experiments validate the efficacy of the proposed method, demonstrating significant improvements in task completion time and overall system performance. The DT-assisted approach not only enhances the adaptability and scalability of fog networks but also provides a robust solution to the challenges posed by the heterogeneity and dynamic nature of fog computing environments.

VII. Limitations and Future Works

The superior performance of TS-based bandit learning demonstrated in the simulation results open many opportunities for future works. Firstly, it is worthy to investigate the proposed algorithms in a general setting of FCNs, in which the set of TNs and HNs has different number of nodes and the network is not fully connected. Secondly, several

matching models would be notably investigated such as many-to-many (MTM) and many-to-one (MTO), in which multiple TNs can be matched with multiple HNs and single HN, respectively. Furthermore, other dimension of dynamic fog networks also need to be investigated such as the mobility of fog nodes as well as the volatile of HNs. The other research issue is to evaluate the performance of the proposed algorithm when the hard deadlines of tasks are concerned.

APPENDIX

APPENDIX A: THEORETICAL REGRET ANALYSIS AND LOWER BOUND

Our proposed algorithm DT-DCBL fits within the CMAB framework with partial feedback and matching constraints. Under standard assumptions such as bounded context vectors, Lipschitz continuity of reward functions, and sub-Gaussian noise in the reward observations, the regret of DT-DCBL can be shown to grow as $\tilde{O}(Md\sqrt{TR})$, where M is the number of TNs, d is the feature dimension, T is the number of time slots, and R is the number of learning rounds per slot. This regret scaling is consistent with theoretical results from contextual bandit literature with linear payoffs using Thompson sampling [46] and UCB-based methods [47]. Furthermore, the extension to multi-agent and matching-based bandit systems under decentralized and partial feedback settings has been studied in [48], supporting the sub-linear regret performance in more complex interaction settings such as ours. This result demonstrates that the DT-DCBL algorithm achieves sub-linear regret over time, ensuring improved decision performance as learning progresses. Our algorithm follows a contextual combinatorial multi-armed bandit setting, where each TN independently selects a HN in a distributed and decentralized manner. After selection, it receives a binary reward feedback (i.e., $y = 1$ for acceptance, $y = 0$ for rejection). Each TN applies a contextual Bayesian Thompson Sampling-based update rule to adjust its belief over the expected reward. This learning structure aligns with prior work such as [46], [49], where it has been shown that Thompson Sampling in CMAB settings with Bernoulli feedback yields a regret bound of $O(\sqrt{T\log T})$ over T time slots, assuming a bounded context space and prior regularity.

Furthermore, theoretical lower bounds for contextual MABs have been established in [50] and [51], showing that any algorithm must suffer a regret of at least $\Omega(\sqrt{T})$ in the worst case, depending on the dimensionality of the context vector. In matching markets with limited feedback, Liu $et.\ al$ [52] proved that a regret of $\Omega(\log T)$ is unavoidable under decentralized partial information settings. Our algorithm respects these bounds and performs close to the empirical upper limit observed in similar decentralized context multi-armed bandit systems.

APPENDIX B: LAPLACE APPROXIMATION

We are interested in approximating a posterior distribution of a weight w_k ($\forall k = 0, 1, 2$) for a logistic regression denoted by $p(w_i|\mathcal{D}) = 1/Cf(w_i)$, where $\mathcal{D} = \{\mathbf{x}; y\}$ is the set of historical observables with n samples ($j = 1, \cdots, n$), $f(\cdot)$ is some unnormalized density and $C = \int f(w_i)dw_k$ is the normalizing constant. $p(w_k|\mathcal{D})$ can be approximated by a second-order Taylor expansion of $f(w_k)$ around the mode $\hat{w}_i$ of the target $p(w_i|\mathcal{D}_i)$. Let $g(w_k) = \log f(w_k)$ and consider the second-order Taylor expansion of $g(w_k)$ around $\hat{w}_k$.

$$g(w_k) \approx g(\hat{w}_k) + g'(\hat{w}_k)(w_k - \hat{w}_k) + \frac{1}{2}g"(\hat{w}_k)(w_k - \hat{w}_k)^2. \tag{5}$$

Note that $g'(\hat{w}_k) = \frac{d\log f(w_k)}{dw_k}|_{w_k=\hat{w}_k} = \frac{f'(\hat{w}_k)}{f(\hat{w}_k)} = 0$, since $f'(\hat{w}_k) = 0$, so we have $g(w_k) \approx g(\hat{w}_k) + \frac{1}{2}g"(\hat{w}_k)(w_k - \hat{w}_k)^2$. Applying the exponential function on both sides yields:

$$f(w_k) \approx f(\hat{w}_k)\exp(\frac{1}{2}g"(\hat{w}_k)(w_k - \hat{w}_k)^2). \tag{6}$$

We can now approximate the normalizing constant:

$$\begin{aligned} C &= \int f(w_k)dw_i \\ &\approx f(\hat{w}_k)\int \exp(-\frac{(-g"(\hat{w}_k))}{2}(w_k - \hat{w}_k)^2)dw_k. \end{aligned} \tag{7}$$

Consider a random variable $z \sim \mathcal{N}(\mu, \sigma^2)$, let $\gamma = \sigma^{-2}$, the pdf of z is $p(z) = \frac{1}{\sigma\sqrt{2\pi}}\exp(-\frac{1}{2}\frac{(z-\mu)^2}{\sigma^2}) = \frac{\sqrt{\gamma}}{\sqrt{2\pi}}\exp(-\frac{\gamma}{2}(z-\mu)^2)$. Since $\int p(z)dz = 1$, thus we have $\int \exp(-\frac{\gamma}{2}(z-\mu)^2)dz = \sqrt{2\pi}/\sqrt{\gamma}$. Referencing to (7), we have $C \approx f(\hat{w}_k)\frac{\sqrt{2\pi}}{\sqrt{-g"(\hat{w}_k)}}$.

Finally, using the Laplace approximation, we have:

$$p(w_k|\mathcal{D}) = \frac{1}{\sqrt{2\pi}\sqrt{(-g"(\hat{w}_k))^{-1}}}\exp(-\frac{1}{2}\frac{(w_k - \hat{w}_k)^2}{(-g"(\hat{w}_k))^{-1}}). \tag{8}$$

Definitely,

$$p(w_k \mid \mathcal{D}) = \mathcal{N}\left(\hat{w}_k, (-g''(\hat{w}_k))^{-1}\right),$$

where $g"(\hat{w}_k) = \frac{d^2(\log f(w_k))}{dw_i^2}|_{w_k=\hat{w}_k}$.

On the other hand, the posterior distribution over the weights is given by Bayes' rule:

$$p(w_k|\mathcal{D}) = \frac{P(\mathcal{D}|w_k)p(w_k)}{P(\mathcal{D})}. \tag{9}$$

Applying log function to the two side yields:

$$-\log p(w_k|\mathcal{D}) = -\log p(\mathcal{D}|w_k) - \log p(w_k) + \log p(\mathcal{D}). \tag{10}$$

The negative log likelihood (NLL) is given by

$$-\log p(\mathcal{D}|w_k) = \sum_{j=1}^{n}\log(1 + e^{-y_j\mathbf{w}^T\mathbf{x}_j}), \tag{11}$$

where $y_j = \{-1, +1\}$.

The negative log prior is given by

$$-\log p(w_k) = -\sum_i \frac{q_k(w_k - m_k)^2}{2} + constant \tag{12}$$

since $w_k \sim \mathcal{N}(m_k, q_k^{-1})$.

Consequently, for MLE,

$$\max p(w_k|\mathcal{D}) = \min(-\log p(w_k|\mathcal{D})) \tag{13}$$

$$= \min[\sum_{k=0}^{2} \frac{q_k(w_k - m_k)^2}{2} \tag{14}$$

$$+ \sum_{j=1}^{n} \log(1 + e^{-y_j \mathbf{w}^T \mathbf{x}_j})].$$

Finally, referencing to 8, we have

$$q_k \leftarrow -g"(w_k)|_{w_i=\hat{w}_k} = q_k + \sum_{j=1}^{n} x_{kj}^2 p_j(1-p_j), \tag{15}$$

where x_{kj} is x_k of the j^{th} sample and $p_j = (1 + \exp(-\mathbf{w}^T \mathbf{x}_j))^{-1}$.

REFERENCES

[1] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, 2014.

[2] 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 Trans. Ind. Inf.*, vol. 14, no. 10, pp. 4590–4602, 2018.

[3] 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.

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

[5] B. P. Rimal, E. Choi, and I. Lumb, "A taxonomy and survey of cloud computing systems," in *Proc. INC, IMS and IDC*, 2009.

[6] A. Botta, W. de Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and internet of things: A survey," *Future Generation Comput. Syst.*, vol. 56, pp. 684–700, Mar. 2016.

[7] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proc. MCC*, 2012.

[8] S. Sarkar, S. Chatterjee, and S. Misra, "Assessment of the suitability of fog computing in the context of internet of things," *IEEE Trans. Cloud Comput.*, vol. 6, no. 1, pp. 46–59, 2018.

[9] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On reducing IoT service delay via fog offloading," *IEEE Internet Things J*, vol. 5, no. 2, pp. 998–1010, 2018.

[10] S. Sarkar and S. Misra, "Theoretical modelling of fog computing: A green computing paradigm to support iot applications," *IET Netw.*, vol. 5, no. 2, pp. 23–29, 2016.

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

[12] Z. Liu, X. Yang, K. Wang, Y. Yang, and Z. Shao, "Computation offloading game for multi-task multi-helper fog networks," in *Proc. GLOBECOM*, 2019.

[13] K. H. Abdulkareem *et al.*, "A review of fog computing and machine learning: Concepts, applications, challenges, and open issues," *IEEE Access*, vol. 7, pp. 153123–153140.

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

[15] K. Guo, M. Sheng, T. Q. S. Quek, and Z. Qiu, "Task offloading and scheduling in fog RAN: A parallel communication and computation perspective," *IEEE Wireless Commun. Lett.*, vol. 9, no. 2, pp. 215–218, 2020.

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

[17] Y. Yang *et al.*, "Pomt: Paired offloading of multiple tasks in heterogeneous fog networks," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8658–8669, 2019.

[18] Z. Liu, Y. Yang, K. Wang, Z. Shao, and J. Zhang, "Post: Parallel offloading of splittable tasks in heterogeneous fog networks," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3170–3183, 2020.

[19] S. Durand and B. Gaujal, "Complexity and optimality of the best response algorithm in random potential games," in *Proc. SAGT*, 2016.

[20] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distributed Syst.*, vol. 26, no. 4, pp. 974–983, 2015.

[21] Z. Liu, X. Yang, Y. Yang, K. Wang, and G. Mao, "Dats: Dispersive stable task scheduling in heterogeneous fog networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3423–3436, 2019.

[22] Tran-Dang, Hoa and Kim, and Dong-Seong, "Disco: Distributed computation offloading framework for fog computing networks," *J. Commun. Netw.*, pp. 1–11.

[23] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *The American Mathematical Monthly*, vol. 69, no. 1, pp. 9–15.

[24] W. Sun, H. Zhang, R. Wang, and Y. Zhang, "Reducing offloading latency for digital twin edge networks in 6G," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 12240–12251, 2020.

[25] T. Lattimore and C. Szepesvári, *Bandit Algorithms*, 2020.

[26] R. Sutton and A. Barto, "Reinforcement learning: An introduction," *IEEE Trans. Neural Netw.*, vol. 9, no. 5, pp. 1054–1054, 1998.

[27] Z. Yang and W. Bai, "Distributed computation offloading in mobile fog computing: A deep neural network approach," *IEEE Commun. Lett.*, vol. 26, no. 3, pp. 696–700, 2022.

[28] M. Goudarzi, M. Palaniswami, and R. Buyya, "A distributed deep reinforcement learning technique for application placement in edge and fog computing environments," *IEEE Trans. Mobile Comput.*, vol. 22, no. 5, pp. 2491–2505, 2023.

[29] 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, 1933.

[30] O. Chapelle and L. Li, "An empirical evaluation of Thompson sampling," in Proc. *NeurIPS*, 2011.

[31] D. Russo and B. Van Roy, "Learning to optimize via posterior sampling," *Mathematics Operations Research*, vol. 39, no. 4, pp. 1221–1243, 2014.

[32] A. D. Flaxman, A. T. Kalai, and H. B. McMahan, "Online convex optimization in the bandit setting: Gradient descent without a gradient," in *Proc. SODA*, 2005.

[33] W. Kumagai, "Introduction to bandit convex optimization algorithms," in *Proc. ISITA*, 2018.

[34] T. Chen and G. B. Giannakis, "Bandit convex optimization for scalable and dynamic IoT management," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 1276–1286, 2018.

[35] ——, "Harnessing bandit online learning to low-latency fog computing," in *Proc. ICASSP*, 2018.

[36] S. Misra, S. P. Rachuri, P. K. Deb, and A. Mukherjee, "Multiarmed-bandit-based decentralized computation offloading in fog-enabled IoT," *IEEE Internet Things J*, vol. 8, no. 12, pp. 10010–10017, 2020.

[37] Z. Zhu, T. Liu, Y. Yang, and X. Luo, "Blot: Bandit learning-based offloading of tasks in fog-enabled networks," *IEEE Trans. Parallel Distributed Syst.*, vol. 30, no. 12, pp. 2636–2649, 2019.

[38] K. Wang, Y. Tan, Z. Shao, S. Ci, and Y. Yang, "Learning-based task offloading for delay-sensitive applications in dynamic fog networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11399–11403, 2019.

[39] B. Cho and Y. Xiao, "Learning-based decentralized offloading decision making in an adversarial environment," *IEEE Trans. Veh. Technol.*, vol. 70, no. 11, pp. 11308–11323, 2021.

[40] H. Zhu, K. Kang, X. Luo, and H. Qian, "Online learning for computation peer offloading with semi-bandit feedback," in *Proc. ICASSP*, 2019.

[41] Y. Sun, S. Zhou, and Z. Niu, "Distributed task replication for vehicular edge computing: Performance analysis and learning-based algorithm," *IEEE Trans. Wireless Commun.*, vol. 20, no. 2, pp. 1138–1151, 2021.

[42] B. Shabir, A. W. Malik, A. U. Rahman, M. A. Khan, and Z. Anwar, "A reliable learning based task offloading framework for vehicular edge computing," in *Proc. ICoDT2*, 2022.

[43] Y. Shan, H. Wang, Z. Cao, and K. Yury, "Data offloading in heterogeneous dynamic fog computing network: A contextual bandit approach," in *Proc. ICCCI*, 2021.

[44] R. Zhang *et al.*, "Calibrated bandit learning for decentralized task offloading in ultra-dense networks," *IEEE Trans. Commun.*, vol. 70, no. 4, pp. 2547–2560, 2022.

[45] 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 *Proc. IWCMC*, 2019.

[46] S. Agrawal and N. Goyal, "Thompson sampling for contextual bandits with linear payoffs," in *Proc. ICML*, 2014.

[47] W. Chu, L. Li, L. Reyzin, and R. Schapire, "Contextual bandits with linear payoff functions," in *Proc. ICAIS*, 2011.

[48] S. Basu, K. Sankararaman, and A. Sankararaman, "Beyond $\log^2(t)$ regret for decentralized bandits in matching markets," in *Proc. ICML*, 2021.
[49] D. Russo and B. Van Roy, "Learning to optimize via information-directed sampling," in *Proc. NeurIPS*, 2014.
[50] V. Dani, T. P. Hayes, and S. M. Kakade, "Stochastic linear optimization under bandit feedback," in *Proc. COLT*, 2008.
[51] Y. Abbasi-Yadkori, D. Pál, and C. Szepesvári, "Improved algorithms for linear stochastic bandits," in *Proc. NeurIPS*, 2011.
[52] L. T. Liu, F. Ruan, H. Mania, and M. I. Jordan, "Bandit learning in decentralized matching markets," *J. Machine Learning Research*, vol. 22, no. 1, pp. 9612–9645, 2021.

Hoa Tran-Dang (M'17, SM'25) received the B.E. degree in Electrical and Electronics Engineering from Hanoi University of Science and Technology (HUST), Vietnam and the M.S. degree in Electronics Engineering from Kumoh National Institute of Technology (KIT), South of Korea in 2010 and 2012, respectively. He pursued the Ph.D. degree with University of Lorraine, France during 2013-2017. He currently works in th Department of IT Convergence Engineering at Kumoh National Institute of Technology, South of Korea as a Research Professor. His research interests include Internet of Things (IoT), edge/fog computing, distributed intelligent computing, physical Internet, and AI.

Dong-Seong Kim (S'98-M'03-SM'14) received his Ph.D. degree in Electrical and Computer Engineering from the Seoul National University, Seoul, Korea, in 2003. From 1994 to 2003, he worked as a Full-time Researcher in ERC-ACI at Seoul National University, Seoul, Korea. From March 2003 to February 2005, he worked as a Postdoctoral Researcher at the Wireless Network Laboratory in the School of Electrical and Computer Engineering at Cornell University, NY. From 2007 to 2009, he was a Visiting Professor with Department of Computer Science, University of California, Davis, CA. He is currently a Director of Kit Convergence Research Institute and ICT Convergence Research Center (ITRC program) supported by Korean government at Kumoh National Institute of Technology. He is IEEE and ACM Senior Member. His current main research interests are real-time IoT, industrial wireless control network, networked embedded system and Fieldbus.