Learning-aided Content Placement in Caching-enabled Fog Computing Systems using Thompson Sampling

Junge Zhu, Xi Huang, Ziyu Shao
School of Information Science and Technology, ShanghaiTech University
Email: {zhujg, huangxi shaozy}@shanghaitech.edu.cn

***Abstract*—In this paper, we focus on the problem of online content placement with an unknown content popularity in caching-enabled fog computing systems, *i.e.*, to decide and update the cached content on resource-limited edge fog nodes with maximum cache hit rate and minimum switching costs of content update. Faced with such uncertainties, the decision-making must be well integrated with effective online learning while ensuring minimum performance loss (*a.k.a.* regret) due to improper content update decision-making. To overcome such difficulties, we formulate the problem as a multi-play multi-armed bandit problem. By adopting Thompson sampling methods, we propose a learning-aided content placement scheme which continuously improves its online decision-making through proactively learning from successive hit-or-miss feedback. Our theoretical and simulation results demonstrate the effectiveness of LACP against baseline schemes with an $O(\log T)$ cache regret over time horizon T.**

***Index Terms*—Content placement, wireless caching, Thompson sampling, fog computing, learning-aided control.**

I. Introduction

Wireless caching techniques are nowadays widely adopted in fog computing systems to improve the quality-of-service at the network edge [1]–[3]. Typically, in a hierarchical caching-enabled fog computing system, each edge fog node (EFN) has limited caching capacity and is only able to cache a subset of available files (often popular files with high access frequency) to serve users that stay within its service range. The selection of such files are usually known as the *content placement*. Each time when a user requests to download a file, the request is called a *hit* if the requested file is cached on the EFN. In this case, the user can download the file directly from the EFN. If the file is not on the EFN (*a.k.a.* a *miss*), then extra bandwidth and longer latency are required to download the file from the EFN's upper-level fog node. To improve quality-of-service, the EFN can proactively update its cached content over time to maximize the total hit number. However, such updates would induce extra costs (*a.k.a. switching cost*) in terms of bandwidth consumption or latencies for fetching files onto the EFN, thus they should not be performed too frequently and must be carefully taken into account of the decision-making procedure.

By far, the design of optimal content placement scheme is still an open problem. The challenges are twofold. One is concerning learning from uncertainties. Often times fog computing systems manifest various uncertainties in practice. For example, content popularity usually remains unknown *a priori*; the lack of such information can prevent the EFN from conducting effective decision-making to achieve minimum performance loss (*a.k.a. regret*) due to decision-making under uncertainties. To this end, the EFN must leverage available information such as historical user profiles or hit-or-miss signals to learn the statistics of such unknown dynamics proactively. A number of existing work have proposed various solutions based on popularity prediction [4]–[6]. Such solutions basically assume a fixed (usually imperfect) popularity prediction upon deciding the content placement. However, acquiring such predictions often requires dedicated offline training based on historical data that may be outdated by the time of decision-making. A more promising alternative is to *integrate* learning into the online decision-making procedure, *i.e.*, having each EFN update its content placement through continuous learning from hit-or-miss feedback in real-time [7]–[9]. To this end, the content placement scheme design must properly handle the *exploration-exploitation* trade-off during such an integrated process. The decision-making, if overly relying on temporally learned estimates, *i.e.*, over-exploitation, may induce constantly enhanced bias towards some sub-optimal placement decisions and result in increasing hit-number loss; if switching among different content placement too frequently, it can lead to more request misses and higher switching costs. Under such uncertainties, it is even more challenging to decide the content placement in the face of the non-trivial trade-off between maximizing the total number of hits and minimizing the switching cost.

In this paper, we focus on the online content placement problem in caching-enabled fog systems with unknown content popularity. Our key results and contributions are summarized as follows.

- ◇ **Problem Formulation:** We formulate the problem as a multi-armed bandit problem with multiple plays, with the aim to maximize the expectation of the total number of hits and reduce the switching cost due to proactive cache updates by the EFN over a finite time horizon.
- ◇ **Algorithm Design:** By employing multi-play Thompson sampling (TS) methods [10], we propose *LACP*, a learning-aided content placement scheme. By proactively learning from successively collected hit-or-miss feedback, LACP continuously refines its online decision-making to achieve a decent tradeoff between exploration and exploitation, as well as a proper balance between hit number maximization and switching cost minimization. Compared to existing solutions that adopt upper-bound confidence (UCB) methods [7]–[9], our TS-based scheme takes the advantage of Bayesian control rule which handles decision-making under uncertainties in a more adaptive manner and outperformed UCB methods in various scenarios [11].

- ◇ **Theoretical Analysis:** Our theoretical analysis shows that LACP can achieve an $O(\log T)$ regret in terms of cache hit over time horizon T.
- ◇ **Numerical simulations:** We conduct extensive simulations to demonstrate the effectiveness of LACP compared to baseline schemes in terms of notable regret reduction over time. Meanwhile, we observe the relationship between the loss of hit number and the switching cost by adjusting the weight of switching cost minimization. Generally, the decrease in the loss of hit number is often accompanied by an increase in the switching cost, and vice versa.

The rest of the paper is organized as follows. Section II presents the system model and problem formulation. Section III elaborates the algorithm design and its corresponding performance analysis. Section IV shows the simulation results while Section V concludes the paper. We relegate all proofs and more results to our technical report [12]

II. System Model and Problem Formulation

In this section, we first introduce our system model, then present the problem formulation of content placement problem in wireless caching networks under multiple-play multi-armed bandit (MP-MAB) settings. Figure 1 shows an example of caching-enabled fog computing system.

A. System Model

We focus on the interaction between each edge fog node (EFN) and the set of m users within its service range in a caching-enabled fog computing system over a finite horizon of T time slots. The EFN's upper-level fog node *i.e.*, central fog node(CFN), stores the set of all n files that could be requested by users, denoted by $\mathcal{F} \triangleq \{f_1, \ldots, f_n\}$. We define the popularity of each file i as p_i which is assumed constant within the time horizon; meanwhile, we denote the content popularity profile by vector $\mathbf{p} \triangleq \{p_1, \ldots, p_n\}$. For notational simplicity, each file is assumed to have unit size, though our model can also be extended to files with different sizes. Considering the caching capacity limit of the EFN, we assume that it can cache at most C files in every time slot. Particularly, we denote the set of cached files on the EFN in time slot t by I_t such that $I_t \in \{I \subseteq \mathcal{F} \,|\, |I| \leq C\}$.

Based on the above model, the interaction between the EFN and its users during each time slot t proceeds as follows. At the beginning of time slot t, the EFN first decides whether to update its cached file set I_{t-1}, *i.e.*, whether to download new files and cache them locally. If it decides not to update its cached content, then we have $I_t = I_{t-1}$; otherwise, we use I_t to denote its updated cached file set.[1] After the content update phase, each user will request files according to the popularity profile $\mathbf{p}$. For each user, its requested file hits the cache (counted as one hit) only when it is in set I_t. Such a process is repeated over the time horizon.

B. Problem Formulation

Two optimization objectives are typically considered when it comes to finding the optimal content placement. One is to maximize the expected total number of average hits for the cached files on the EFN over the time horizon, which is defined as $\sum_{t=1}^{T} \sum_{f_i \in I_t} mp_i$ with respect to $\{I_t\}_{t=1}^{T}$. The greater the total number of average cache hits are, the more effective the content placement is. The other is to minimize the switching cost due to cache content update (in terms of bandwidth consumption or induced delay to fetch files) between successive time slots. In our model, we define it as the number of different files between I_{t-1} and I_t, *i.e.*, $g(I_{t-1}, I_t) \triangleq \sum_{f \in \mathcal{F}} |\mathbb{1}_{f \in I_t} - \mathbb{1}_{f \in I_{t-1}}|$ where $\mathbb{1}_{\mathrm{A}} = 1$ if event A occurs and zero otherwise. Taking switching cost into account prevents the EFN from too frequent content update which may result in not only considerable bandwidth consumption or longer delay but also excessive cache misses.

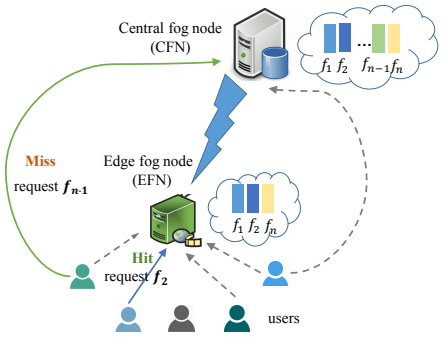

Fig. 1: A caching-enabled fog computing system

By combining the above objectives, we have the following problem formulation.

$$\underset{I_t \in \mathcal{I}}{\text{Maximize}} \quad \sum_{t=1}^{T} \left[\left(\sum_{f_i \in I_t} mp_i \right) - w * g(I_{t-1}, I_t) \right], \qquad (1)$$

where w is a constant parameter which weighs the importance of switching cost minimization. In fact, parameter w can also be viewed as the cost of updating one file on the EFN.

III. Algorithm Design & Performance Analysis

For problem (1), if the content popularity profile $\mathbf{p}$ is known *a priori*, then the optimal content placement can be decided by directly caching the set of most popular C files, denoted by set I^*. Then the EFN can fix the cache content over T time slots (hence the switching cost is always zero). However, the popularity profile $\mathbf{p}$ is usually unattainable in practice, making the problem extremely challenging to solve. Faced with such difficulties, starting from some initial content placement, the EFN needs to continuously update its cache by learning the file popularity through the hit-or-miss feedback from users. To this end, the content update procedure must be conducted in a dynamic fashion while interacting with online learning procedure, aiming to achieve not only a decent tradeoff between exploration and exploitation but also a proper balance between cache hit number maximization and switching cost minimization.

To this end, in this section, we first reformulate problem (1) under the settings of multiple-play multi-armed bandits (MP-MAB) model. Then by adopting multi-play Thompson sampling methods, we develop an online learning-aided content placement (LACP) scheme to solve problem (1) with an unknown popularity profile $\mathbf{p}$.

A. Problem Reformulation

A typical MAB problem considers the situation in which an agent makes a series of interactions with its residing environment. In each round, the agent picks one action (arm)

[1]Note that if, in some time slot t, the EFN has cached C files but it decides to add a new file $f \notin I_t$, then the EFN can adopt any existing cache replacement scheme to update its content, *e.g.*, recently least used (RLU).

from a given set of actions (arm set); according to the chosen action, the environment reveals some reward to the agent which is sampled from some unknown distribution. With rounds of interaction, the agent aims to find an optimal policy to maximize the expected cumulative rewards. MP-MAB extends the settings of MAB by allowing the agent to play multiple arms in the same round, with the rewards of each selected arm being sampled independently from their own distributions.

To recast problem (1) into the settings of MP-MAB, we view the EFN as the agent and the available file set $\mathcal{F}$ as the arm set. Then in each time slot t, the agent selects a subset of C arms (files), denoted by $I_t \in \mathcal{I}$. Next, the EFN receives requests of the selected files from users. The corresponding reward is defined as the weighted difference between total hit number and the switching cost, *i.e.*, $\sum_{f_i \in I_t} mp_i - w \cdot g(I_{t-1}, I_t)$. By defining $X_i(t)$ as the observed number of hits for requesting file f_i, $\forall f_i \in I_t$, such that $X_i(t) \in [0, m]$, $\forall f_i \in I_t$, we can rewrite problem (1) as

$$\underset{[I_t^\pi]_t}{\text{Maximize}} \sum_{t=1}^{T} \left[\left(\sum_{f_i \in I_t^\pi} \mathbb{E}[X_i(t)] \right) - w * g(I_t^\pi, I_{t-1}^\pi) \right], \tag{2}$$

where π denotes some particular content placement policy. To define the regret, *i.e.*, the performance loss due to successive content update under policy π, we consider two types of regrets.

One is the hit-regret, defined as the difference between the numbers of hits given the optimal content replacement I^* and policy π, respectively, *i.e.*,

$$R_h^\pi(T) \triangleq \sum_{t=1}^{T} \sum_{f_j \in I^*} mp_j - \sum_{t=1}^{T} \sum_{f_i \in I_t^\pi} \mathbb{E}[X_i(t)]. \tag{3}$$

The other one is switching-regret. Since the switching cost under optimal content placement policy equals zero, thus we define the switching-regret with respect to policy π as

$$R_s^\pi(T) \triangleq \sum_{t=1}^{T} w * g(I_t^\pi, I_{t-1}^\pi). \tag{4}$$

As a result, the *regret* with respect to policy π is defined as

$$R^\pi(T) \triangleq R_h^\pi(T) + R_s^\pi(T). \tag{5}$$

Note that minimizing the regret is equivalent to maximizing the expected reward in (2).

B. Algorithm Design

To solve problem (2), we adopt Thompson sampling methods [13] and propose *LACP*, a learning-aided content placement scheme. We show LACP's pseudocode in Algorithm 1.

First, notice that the hit number of each selected file f_i, denoted by $X_i(t)$, follows some Binomial distribution with an unknown mean p_i. Then to employ Thompson sampling methods, for each file f_i, we assume the popularity probability p_i to be a random variable that follows a Beta distribution with parameters a_i and b_i, *i.e.*, $p_i \sim \text{Beta}(a_i, b_i)$. By Beta-Binomial conjugacy [14], the posterior distribution of p_i follows a beta distribution with parameters $a_i + H_i(t)$ and $b_i + m - H_i(t)$, *i.e.*, $\text{Beta}(a_i + H_i(t), b_i + m - H_i(t))$, where $H_i(t)$ denotes the total number of hits by requesting file f_i in the first t time slots.

Algorithm 1 Learning-aided Content Placement (LACP)

Input: Beta parameter $(a_i, b_i) \leftarrow (1, 1), \forall f_i \in \mathcal{F}$, $|\mathcal{F}| = n$
Output: Content placement decision sequence $\{I_t\}_{t=1}^T$

1: **for** t in $\{1, ..., T\}$ **do**
2: **for** i in $\{1, ..., n\}$ **do**
3: Draw $\theta_i(t) \sim \text{Beta}(a_i, b_i)$ for file f_i.
4: **end for**
5: **for** i in $\{1, ..., n\}$ **do**
6: **if** $i \in I_{t-1}$ **then**
7: $\text{cost}(i) \leftarrow 0$.
8: **else**
9: $\text{cost}(i) \leftarrow 1$.
10: **end if**
11: **end for**
12: $I_t \leftarrow$ top C files with the largest $m\theta_i(t) - w \cdot \text{cost}(i)$.
13: Observe the number of hits $X_i(t)$ for each f_i in I_t.
14: **for** $f_i \in I_t$ **do**
15: $a_i \leftarrow a_i + X_i(t)$.
16: $b_i \leftarrow b_i + m - X_i(t)$.
17: **end for**
18: **end for**

In Algorithm 1, the prior distribution of p_i for each $f_i \in \mathcal{F}$ is set as $\text{Beta}(1, 1)$ with $a_i = b_i = 1$. In other words, without further prior information, the unknown mean p_i is assumed to follow a uniform distribution over interval $(0, 1)$. Such parameters a_i and b_i would be updated later after the EFN observes the number of hits for each requested file f_i at the end of each time slot (lines 14-17). In the following, we elaborate how LACP proceeds in each time slot t.

At the beginning of each time slot t, the EFN first acquires an estimate (denoted by $\theta_i(t)$) of the unknown mean p_i by drawing a sample from distribution $\text{Beta}(a_i, b_i)$ (lines 2-4). Then the EFN computes the switching cost of selecting each file by checking whether the file has already been cached on the EFN (lines 5-10). Next, the EFN updates its cached content I_t by including the top C files (I_t) with the largest estimated rewards. Then users send requests for files, the EFN observes and collects the number of hits $X_i(t)$ for each $f_i \in I_t$ (line 13). Finally, at the end of time slot t, the EFN updates the parameters of the posterior distribution with respect to each popularity probability p_i (lines 14-17).

In addition, we note that LACP can be run in a computationally-efficient manner in each time slot t. Particularly in Algorithm 1, both of the two for-loops (lines 2-4 and lines 5-10) require n iteration and the last for-loop (lines 14-17) requires C ($C < n$) iteration. Each iteration requires a constant complexity. Hence, the computational complexity for LACP is $O(n)$, linear in the number of files, implying LACP can make content placement decisions with low overheads.

Remark: LACP strikes a decent balance between exploration and exploitation by effectively exploiting the collected hit-or-miss feedback information from user requests. On the one hand, with more and more feedback being collected, LACP continuously updates its posterior knowledge about the popularity of different files. Particularly, the EFN tends to keep cached files with more cache hits in the past time slots, since the posterior distribution of such files tend to skew towards larger values (hence the sampled estimated

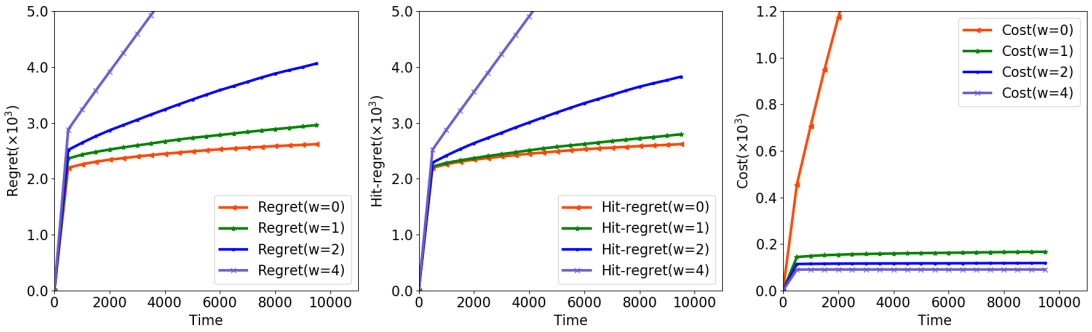

Fig. 2: Comparison of regret of LACP with various w

would be large with a higher probability). In this way, the selection of empirically popular files will be reinforced timely which conduces to effective exploitation and thus maximizing the cumulative rewards. On the other hand, unlike existing solutions [7]–[9] which generally update the cached content based on upper-confidence bound estimate in a deterministic manner, LACP generates the estimate for the unknown mean of each file's popularity in a probabilistic manner. Such a design conduces to continuous exploration.

C. Performance Analysis

We show that the regret induced by LACP (with $w = 0$) over finite time horizon is guaranteed within an $O(\log T)$ upper bound, which is specified by the following theorem.

Theorem 1. *Over T time slots, the expected regret incurred by LACP (with $w = 0$) is upper bounded by*

$$\mathbb{E}(R_{\widetilde{\pi}}(T)) \leq m \sum_{f_i \in \mathcal{F} \setminus I^*} \frac{\Delta_{i,l} \log T}{D\left(p_i, p_l\right)} + O((\log T)^{2/3}), \quad (6)$$

where 1) $l \triangleq \arg\min_{f_j \in I^*} p_j$, i.e., the index of the file whose p_j is the minimum in the optimal cache content placement set I^*; 2)$\Delta_{i,l} = p_l - p_i$; and 3) $D\left(x, y\right)$ is the Kullback-Leibler divergence [15] *between two Bernoulli distributions Bern(x) and Bern(y), i.e., $D\left(x, y\right) \triangleq x \log \frac{x}{y} + (1-x) \log \frac{1-x}{1-y}$.*

IV. Simulation

In this section, we conduct simulations to verify the effectiveness of LACP by comparing it with the state-of-the-art multiple-play multi-armed bandit algorithms: CUCB [16] and ϵ-greedy [17]. Meanwhile, we explore the relationship between the loss of hit number and the switching cost by adjusting the weight of switching cost in LACP scheme. We refer more details about the two baseline schemes to our technical report [12].

A. Basic Settings

We consider an edge fog node (EFN) in a caching-enabled fog computing system. There are 50 users in its service range, *i.e.*, $m = 50$. The EFN can cache at most 10 files ($C = 10$) in every time slot and the total number of files stored in a central fog node (CFN) equals to 100 ($|\mathcal{F}| = 100$). The popularity probability of each file is sampled from the uniform distribution Uni$(0, 1)$ randomly. Note that all simulation results are acquired by averaging the results from 50 runs of simulations.

B. Simulation Results

Recall that the regret with respect to a policy is the gap between the cumulative reward incurred by the policy and the optimal cumulative reward. Therefore, a smaller regret implies a higher reward. In Figures 2-3, we evaluate the performance of algorithms by comparing their regret.

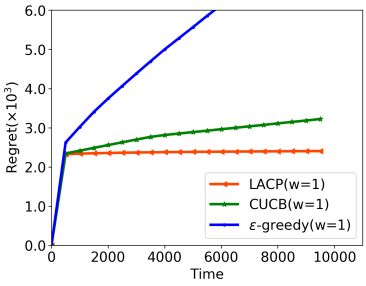

Fig. 3: Comparison of regret of LACP, CUCB and ϵ-greedy

From Figure 2, we see that the weight of switching cost influences the regret, hit-regret and switching cost greatly. As the weight increases, the hit-regret increases as well. The reason is that when the EFN chooses the cache content, it should not only consider the expected hit number of the files by now, but also consider whether the file with the largest expected hit number is already on the EFN and the switching cost of adding it to the EFN. In particular, $w = 0$ means that the EFN does not take switching cost into account, and selects files with largest expected hit number. The larger the weight is, the bigger the probability the content placement remains the same. Therefore, the switching cost will decrease as the weight increases. When $w = 0$, the switching cost is quite high. Generally, the decrease in the loss of hit number is often accompanied by an increase in the switching cost, and vice versa. Besides, we also see that the hit-regret increases rapidly at the beginning. As t increases, the growth of the hit-number becomes slower. This means that LACP can learn the knowledge and have a more accurate estimation of files' popularities. Figure 3 shows the comparison of LACP, CUCB and ϵ-greedy under the same settings with weight $w = 1$. We see that LACP has a better performance by achieving a notable regret reduction.

V. Conclusion

In this paper, we studied the problem of online content placement with unknown content popularity. By formulating the problem as an MP-MAB problem with binary hit-or-miss feedback, we proposed a content placement scheme based on multi-play Thompson sampling techniques. Our proposed scheme effectively integrates online learning into the decision-making procedure for content placement update on each EFN. We conducted both theoretical analysis and simulations to demonstrate the effectiveness of LACP.

REFERENCES

[1] M. Ji, G. Caire, and A. F. Molisch, "Optimal throughput-outage trade-off in wireless one-hop caching networks," in *Proceedings of IEEE ISIT*, 2013.

[2] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2856–2867, 2014.

[3] A. Sengupta, S. Amuru, R. Tandon, R. M. Buehrer, and T. C. Clancy, "Learning distributed caching strategies in small cell networks." in *Proceedings of ISWCS*, 2014.

[4] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proceedings of IEEE INFOCOM*, 2010.

[5] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless video content delivery through distributed caching helpers," *arXiv preprint arXiv:1109.4179*, 2011.

[6] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "Wireless video content delivery through coded distributed caching," in *Proceedings of IEEE ICC*, 2012.

[7] P. Blasco and D. Gündüz, "Learning-based optimization of cache content in a small cell base station," in *Proceedings of IEEE ICC*, 2014.

[8] ——, "Content-level selective offloading in heterogeneous networks: Multi-armed bandit optimization and regret bounds," *arXiv preprint arXiv:1407.6154*, 2014.

[9] S. Müller, O. Atan, M. van der Schaar, and A. Klein, "Smart caching in wireless small cell networks via contextual multi-armed bandits," in *Proceedings of IEEE ICC*, 2016.

[10] Y. Xia, T. Qin, W. Ma, N. Yu, and T.-Y. Liu, "Budgeted multi-armed bandits with multiple plays." in *Proceedings of IJCAI*, 2016.

[11] O. Chapelle and L. Li, "An empirical evaluation of thompson sampling," in *NeurIPS*, 2011.

[12] J. Zhu, X. Huang, and Z. Shao, "Tech. report," https://faculty.sist.shanghaitech.edu.cn/faculty/shaozy/zhu20-lacp.pdf.

[13] S. Agrawal and N. Goyal, "Further optimal regret bounds for thompson sampling," in *Proceedings of Artificial intelligence and statistics*, 2013.

[14] P. D. Hoff, *A First Course in Bayesian Statistical Methods*. Springer, 2009, vol. 580.

[15] A. Garivier and O. Cappé, "The kl-ucb algorithm for bounded stochastic bandits and beyond," in *Proceedings of COLT*, 2011.

[16] W. Chen, Y. Wang, and Y. Yuan, "Combinatorial multi-armed bandit: General framework and applications," in *Proceedings of ICML*, 2013.

[17] V. Kuleshov and D. Precup, "Algorithms for multi-armed bandit problems," *arXiv preprint arXiv:1402.6028*, 2014.