

Intelligent Caching in Dense Small-Cell Networks with Limited External Resources

Bingshan Hu, Maryam Tanha, Dawood Sajjadi, and Jianping Pan

Department of Computer Science, University of Victoria, Victoria, British Columbia, Canada

Abstract—A promising solution to alleviate the mobile traffic burden on the Internet is to cache the most popular content at the heterogeneous wireless network edge. However, due to the vast content stored at the remote server, and to cache effectively, it concerns the file popularity profile that may not be known by the network operators in advance. Therefore, online learning techniques are used to tackle the challenges brought by the unknown knowledge. We present an effective and efficient algorithm based on the stochastic combinatorial multi-armed bandits with locked-up slots to address the content caching problem. Our work particularly addresses the scenario where dense small cells with diverse user populations are deployed. Additionally, this network is only given limited external resources such as computational resource to learn the caching policies and wireless backhaul resource to refresh the caches. Our algorithm learns the caching policies online which is to decide which files to be cached sequentially. Despite sharing the limited external resources, the proposed algorithm guarantees the performance of each small cell to approach the optimum. Experiments are conducted to cross-validate the theorem presented in this work.

Index Terms—Cellular Content Caching, Multi-armed Bandits.

I. INTRODUCTION

A growing number of wireless devices are being connected to the Internet such as smart phones, tablets, and laptops, and with the advent of Internet of Things (IoT), this number is expected to surge in the near future [1]. Additionally, Cisco points out mobile data traffic has been continuously increasing over the past few years, and it is predicted to grow almost sevenfold by 2021 [2]. Consequently, a prohibitively high network backbone bandwidth will be required to backhaul all the resultant traffic [1]. Take the Netflix subscribers for example, user statistics show heavy traffic bursts during the holiday season, which imposes a sudden burden on the Internet [3]. Whilst on the one hand, users do not want their Quality of Service (QoS) to be impacted by such traffic bursts, network operators on the other hand may not be able to incur the vast investments necessary to guarantee the QoS. Therefore, how to alleviate the heavy traffic burden imposed on the Internet in a cost effective way is an urgent problem to be tackled.

One promising solution is to *cache* the most popular content at the network edge, thereby bringing the content closer to the end users [4]. In some of the prior works [5], [6], a Small-Cell Base Station (SBS) is used to cache files. However, the cost to deploy SBSs, particularly the dense deployment with a wired backhaul, is significant to the network operators. Instead, a cost-effective way is to deploy them with a wireless backhaul,

which is also more flexible to manage. Additionally, due to the vast files stored at the remote server and the limited size of the cache, we need to decide which files to be cached carefully. Typically, the selection of the to-be-cached files is termed as the *content caching problem*, and the decision of the selected files is termed as the *caching policy*. To cache effectively, one of the most commonly used indicators is the content popularity profile which describes the request probability towards each content. Basically, the more popular the content is, the higher probability the user requests it.

The existing works [4]–[6] addressed the content caching problems assuming a priori knowledge of the content popularity profile. However, this assumption may never be practical as users themselves may not know it, not to mention the network operators. Some other works [7]–[11] dealt with more practical cases where a priori knowledge was not assumed but online learning techniques were used to learn the caching policies sequentially. The key idea is to keep the users informed of what has been cached already and then observe whether the cached files are requested by the users or not. They only investigated the performance in a single small-cell scenario. However, practically dense small cells are often deployed to maximize the coverage and the spectrum utilization, and typically different small cell serves different user population. Therefore, we are motivated to address the content caching problem in dense small-cell networks with a diverse user population. Additionally, we make a more practical assumption that all small cells share the limited external resources such as computational resource *to learn the caching policies* and wireless backhaul resource *to refresh the caches*.

To achieve the aforementioned goals, we take a similar approach to the existing works [7]–[9] that formulated the content caching problem as a stochastic combinatorial multi-armed bandit (MAB) problem, i.e., the content popularity profile are independent and identically distributed (i.i.d.) from unknown distributions. However, to tackle the challenge brought by the limited external resources, we additionally introduce the concept of locked-up slots in the classic stochastic combinatorial MAB problem, which is inspired by [12].

To the best of our knowledge, it is the first work that content caching problem is formulated as a stochastic combinatorial MAB problem with locked-up slots. The following summarizes the key contributions in this paper. (i) We formulate the content caching problem in dense small-cell networks with limited external resources and a diverse user population as a

stochastic combinatorial MAB problem with locked-up slots. (ii) We propose a novel online algorithm called Aggregated Upper Confidence Bound with Locked-Up Slots (LA-UCB) to solve the aforementioned problem. (iii) We derive a sub-linear regret and analyze the external resource required of LA-UCB to show the effectiveness and efficiency. (iv) We conduct experiments to cross-validate the theoretical analysis.

The remainder of the paper is organized as follows. In Section II, we present system model and problem formulation. In Section III, we present a novel online algorithm LA-UCB, and analyze the regret and external resources required. In Section IV, we conduct experiments to cross-validate the theoretical analysis presented. Section V concludes the paper.

II. SYSTEM MODEL AND PROBLEM FORMULATION

In this paper, we focus on a heterogeneous network with dense small cells deployed within the coverage area of a macro cell. Each SBS works independently and has a *cache entity* to store files while the Macro-Cell Base Station (MBS) has a *learning entity* to learn caching policies, fetch corresponding files from the remote server, and refresh the caches. We assume there is a wireless backhaul between MBS and SBS, and no direct communications among SBSs. Additionally, we further assume there is no learning capability at SBSs due to the limited computational resource.

Set $\mathbb{K}$ collects all the small cells and k indicates the k -th element in $\mathbb{K}$. Set $\mathbb{U}_k$ collects all the users covered by the k -th small cell and $i^{(k)}$ indicates the i -th element in $\mathbb{U}_k$. Set $\mathbb{F}_k$ collects all the files that are potentially requested by $\mathbb{U}_k$ and $j^{(k)}$ indicates the j -th element in $\mathbb{F}_k$. Without loss of generality, the size of all files in $\mathbb{F}_k$ is assumed to be identical. Event $e(j^{(k)}, i^{(k)})$ represents an independent Bernoulli event that user $i^{(k)} \in \mathbb{U}_k$ requests file $j^{(k)} \in \mathbb{F}_k$ with success probability $p(j^{(k)}, i^{(k)})$. Set $\mathbb{E}_k = \{e(j^{(k)}, i^{(k)}) : j^{(k)} \in \mathbb{F}_k, i^{(k)} \in \mathbb{U}_k\}$ collects all the aforementioned events while set $\mathbb{P}_k = \{p(j^{(k)}, i^{(k)}) : j^{(k)} \in \mathbb{F}_k, i^{(k)} \in \mathbb{U}_k\}$ collects all the corresponding success probabilities. Let c_k represent the cache size at SBS k . Considering the fact that a caching policy is to select c_k files out of $\mathbb{F}_k$, set $\mathbb{S}_k \subseteq \mathbb{F}_k$ represents the policy played at SBS k . Define that if a user requests an already cached file, we call a *unit of reward* is achieved due to caching.

To introduce the problem formulation, define $r_{\mathbb{S}_k}$ as the *reward*, which is the expected number of reward units that can be achieved when playing $\mathbb{S}_k$. From the property of independent Bernoulli event and the linearity of expectation, $r_{\mathbb{S}_k}$ only depends on the corresponding success probabilities concerning $\mathbb{S}_k$ and $\mathbb{U}_k$, i.e., $r_{\mathbb{S}_k} = \sum_{j^{(k)} \in \mathbb{S}_k} \sum_{i^{(k)} \in \mathbb{U}_k} p(j^{(k)}, i^{(k)})$. Therefore, to find the best policy, we formulate it as optimization problem (1) and use $\mathbb{S}_k^*$ to indicate the policy yielded by (1).

$$\begin{aligned} \max \quad & \sum_{j^{(k)} \in \mathbb{S}_k} \sum_{i^{(k)} \in \mathbb{U}_k} p(j^{(k)}, i^{(k)}) \quad , \\ \text{s.t.} \quad & \mathbb{S}_k \subseteq \mathbb{F}_k, \quad |\mathbb{S}_k| = c_k \quad . \end{aligned} \quad (1)$$

From (1) it is worth to mention an important observation as follows. Given k and $j^{(k)}$, the probability distribution of the sum over all $e(j^{(k)}, i^{(k)})$, $\forall i^{(k)} \in \mathbb{U}_k$ forms a Poisson binomial

distribution with mean $\theta_{j^{(k)}}$ equal to the sum over all the corresponding success probabilities, which is

$$\theta_{j^{(k)}} = \sum_{i^{(k)} \in \mathbb{U}_k} p(j^{(k)}, i^{(k)}) \quad , \quad (2)$$

and the finite bounded support $\{0, |\mathbb{U}_k|\}$ [13], [14]. Then by applying (2) to (1), (1) could be simplified as

$$\begin{aligned} \max \quad & \sum_{j^{(k)} \in \mathbb{S}_k} \theta_{j^{(k)}} \quad , \\ \text{s.t.} \quad & \mathbb{S}_k \subseteq \mathbb{F}_k, \quad |\mathbb{S}_k| = c_k \quad . \end{aligned} \quad (3)$$

If $p(j^{(k)}, i^{(k)})$ were assumed a priori knowledge, it would be trivial to solve since we can calculate $\theta_{j^{(k)}}$ first and then input it to an optimization problem solver. However, practically we would never acquire all $p(j^{(k)}, i^{(k)})$ perfectly, and thus online learning techniques are used to cope with the unknown knowledge. Additionally, due to the limited external resources, not all the caching policies can be learned simultaneously or not all the caches can be refreshed simultaneously. The external resource limitation makes the problem more challenging.

In our model, time is divided into slots and the network works in a discrete manner with slot index $t = 1, 2, \dots, T$. For the sake of simplicity, we assume MBS can only serve one SBS in each t and uses a round robin approach to decide which SBS to serve. Note that “serve” means MBS needs to consume computational resource to learn the caching policy and wireless backhaul resource to refresh the cache.

In each t , the following happens sequentially. (i) MBS selects the to-be-served SBS in a round robin approach; (ii) MBS learns the caching policy of the selected SBS; (iii) MBS refreshes the cache based on the just learned policy; (iv) All SBSs play the policies. Note that “play policy” means SBS broadcasts what has been cached to the users, users request files, SBS serves the request, and SBS reports the observed reward of each cached file to MBS.

Let $\mathbb{S}_k(t)$ indicate the to-be-played policy at t and $m(t) \in \mathbb{K}$ indicate the to-be-served SBS at t . Then $\mathbb{S}_{m(t)}(t)$ indicates the policy that can be updated at t , and to find the best $\mathbb{S}_{m(t)}(t)$, it can be formulated as

$$\begin{aligned} \max \quad & \sum_{j^{(m(t))} \in \mathbb{S}_{m(t)}(t)} \hat{\theta}_{j^{(m(t))}, t-1} \quad , \\ \text{s.t.} \quad & \mathbb{S}_{m(t)}(t) \subseteq \mathbb{F}_{m(t)}, \quad |\mathbb{S}_{m(t)}(t)| = c_{m(t)} \quad , \end{aligned} \quad (4)$$

where $\hat{\theta}_{j^{(m(t))}, t-1}$ is the empirical mean observed at the end of $t-1$. Recall that true mean $\theta_{j^{(m(t))}}$ is not known in advance, and thus we can only use the information observed at the end of $t-1$, the empirical mean $\hat{\theta}_{j^{(m(t))}, t-1}$, to find the best $\mathbb{S}_{m(t)}(t)$. Note that $\mathbb{S}_{m(t)}(t)$ may deviate from $\mathbb{S}_{m(t)}^*$ significantly since $\mathbb{S}_{m(t)}^*$ is obtained based on the true mean $\theta_{j^{(m(t))}}$ while $\mathbb{S}_{m(t)}(t)$ is obtained based on the empirical mean up to $t-1$. In contrast, for all the remaining SBSs, $\mathbb{S}_k(t)$ cannot be updated at t , i.e., $\mathbb{S}_k(t) = \mathbb{S}_k(t-1)$ for all $k \in \mathbb{K} \setminus m(t)$.

Remark 1. In our design, we choose to decouple the functionalities of learning and fetching (by MBS), and caching (by SBS). The benefit of the decoupling is to exploit the potential time-space multiplexing gain. Practically, MBS can use an

additional buffer to store the fetched files for a while. Since a popular file for SBS 1 at t may also be a popular file for SBS 2 at $t + 1$, storing the file in a buffer eliminates the need to fetch the same file again. Additionally, after the system reaches stability, the policies learned in adjacent slots for a given SBS do not vary too much over time. By using the additional buffer at MBS, the cost to fetch files can be significantly reduced.

III. INTELLIGENT CACHING ALGORITHM

In this section, we will first introduce our proposed algorithm: Aggregated Upper Confidence Bound with Locked-Up Slots (LA-UCB), and then analyze the regret and the external resources required of LA-UCB.

Before presenting LA-UCB, it is worth to mention, actually, MBS does not simply use empirical mean $\hat{\theta}_{j^{(m(t))},t-1}$ to find the best $\mathbb{S}_{m(t)}(t)$ as it needs to make a trade-off between exploration (to estimate true mean) and exploitation (to achieve the best instantaneous reward). Instead, it inputs $\hat{\theta}_{j^{(m(t))},t}$, the Upper Confidence Bound (UCB) of $\hat{\theta}_{j^{(m(t))},t-1}$, to the problem solver. The UCB is computed based on (5) for all $j^{(m(t))} \in \mathbb{F}_{m(t)}$ [15].

$$\bar{\theta}_{j^{(m(t))},t} = \hat{\theta}_{j^{(m(t))},t-1} + |\mathbb{U}_{m(t)}| \sqrt{\frac{2 \cdot \ln t}{T_{j^{(m(t))},t-1}}}, \quad (5)$$

where $T_{j^{(m(t))},t-1}$ is the number of times file $j^{(m(t))}$ has been cached at the SBS $m(t)$ at the end of $t - 1$. Basically, the principle behind UCB is we always need to be optimistic in face of uncertainty. More specifically, if the learned policy is equal to $\mathbb{S}_{m(t)}^*$, SBS $m(t)$ acts optimally at t . Whilst if $\mathbb{S}_{m(t)}(t)$ is inferior to $\mathbb{S}_{m(t)}^*$, SBS $m(t)$ can still learn some information to guide to make fewer mistakes about the future policy.

Algorithm 1 describes LA-UCB in detail. For each $k \in \mathbb{K}$, MBS uses two $|\mathbb{F}_k| \times 1$ dimensional vectors to record the observations during the learning. The first vector $T_{j^{(k)},t}$ records the number of times that file $j^{(k)}$ has been cached at the end of t . The second vector $\hat{\theta}_{j^{(k)},t}$ records the empirical mean after $j^{(k)}$ has been cached $T_{j^{(k)},t}$ times. Initially, $\mathbb{S}_k(0) = \emptyset$, and $T_{j^{(k)},0} = 0$ and $\hat{\theta}_{j^{(k)},0} = 0$ for all $j^{(k)} \in \mathbb{F}_k$. How to update these two vectors are summarized in (6).

$$T_{j^{(k)},t} = \begin{cases} T_{j^{(k)},t-1} + 1, & j^{(k)} \in \mathbb{S}_k(t) \\ T_{j^{(k)},t-1}, & j^{(k)} \notin \mathbb{S}_k(t) \end{cases}, \quad (6)$$

$$\hat{\theta}_{j^{(k)},t} = \frac{1}{T_{j^{(k)},t}} \cdot \sum_{s=1}^t X_{j^{(k)},s},$$

where $X_{j^{(k)},s}$ is the observed reward of file $j^{(k)}$ at slot s . Note that $X_{j^{(k)},s} = 0$, if $j^{(k)} \notin \mathbb{S}_k(s)$. This is due to the fact that we can only observe the rewards of the already cached files.

Just like other UCB-based algorithms, LA-UCB also suffers performance loss due to the unknown knowledge $\theta_{j^{(k)}}$, which is termed as the *regret*. The regret describes the expected cumulative performance gap between always playing $\mathbb{S}_k^*$ and those yielded by LA-UCB. Additionally, the locked-up feature makes the regret even higher due to the fact that if the policy yielded by LA-UCB is inferior to $\mathbb{S}_k^*$, this policy will continue to being played in the next consecutive slots until a new policy

Algorithm 1 LA-UCB Algorithm

- 1: $t \leftarrow 0$
 - 2: **while** $t \leq T$ **do**
 - 3: $t \leftarrow t + 1$
 - 4: MBS selects $m(t)$ in a round robin approach.
 - 5: MBS computes UCB $\bar{\theta}_{j^{(m(t))},t}$ for all $j^{(m(t))} \in \mathbb{F}_{m(t)}$.
 - 6: To find the best $\mathbb{S}_{m(t)}(t)$, MBS solves problem (4) with $\bar{\theta}_{j^{(m(t))},t}$ replacing $\hat{\theta}_{j^{(m(t))},t-1}$ as the input.
 - 7: MBS refreshes the cache of SBS $m(t)$ based on the new $\mathbb{S}_{m(t)}(t)$.
 - 8: All SBSs play $\mathbb{S}_k(t)$.
 - 9: MBS updates $T_{j^{(k)},t}$ and $\hat{\theta}_{j^{(k)},t}, \forall k \in \mathbb{K}$ based on (6).
 - 10: **end while**
-

is yielded. Based on the explanation of the regret, given k , the regret $\mathcal{R}_k^T$ can be expressed as

$$\mathcal{R}_k^T = \mathbb{E} \left[\sum_{t=1}^T (r_{\mathbb{S}_k^*} - r_{\mathbb{S}_k(t)}) \right], \quad (7)$$

where $r_{\mathbb{S}_k^*}$ is the reward achieved when playing $\mathbb{S}_k^*$. We call $r_{\mathbb{S}_k^*}$ the *optimum* since playing $\mathbb{S}_k^*$ achieves the best performance.

To introduce the theorem, define $\bar{\mathbb{S}}_k^*$ as the complement of $\mathbb{S}_k^*$ with respect to $\mathbb{F}_k$, i.e., $\bar{\mathbb{S}}_k^* = \mathbb{F}_k \setminus \mathbb{S}_k^*$ and $\Delta_{i^{(k)},j^{(k)}} = \theta_{j^{(k)}} - \theta_{i^{(k)}}$ as the loss of reward brought by caching $i^{(k)} \in \bar{\mathbb{S}}_k^*$ instead of $j^{(k)} \in \mathbb{S}_k^*$.

Theorem 1. *Given k , the regret $\mathcal{R}_k^T$ of LA-UCB is at most*

$$\sum_{i^{(k)} \in \bar{\mathbb{S}}_k^*} \left(\frac{16|\mathbb{U}_k|^2 \ln T}{\Delta_{i^{(k)},c_k^{(k)}}} - \frac{8|\mathbb{U}_k|^2 \ln T}{\Delta_{i^{(k)},1^{(k)}}} + \sum_{j^{(k)} \in \mathbb{S}_k^*} |\mathbb{K}| \frac{\pi^2+3}{3} \Delta_{i^{(k)},j^{(k)}} \right),$$

where $\Delta_{i^{(k)},1^{(k)}}$ and $\Delta_{i^{(k)},c_k^{(k)}}$ represent the maximum and the minimum loss of the reward by caching $i^{(k)} \in \bar{\mathbb{S}}_k^*$, respectively.

Due to the space limitation, the proof can be found in online appendix [16]. Theorem 1 shows the effectiveness of LA-UCB as the time-averaged regret will decay to zero. This tendency implies the time-averaged reward can approach $r_{\mathbb{S}_k^*}$ eventually.

Now we start analyzing the external resources required by LA-UCB. Firstly, since exactly one SBS is served in each slot, the total amount of computation needed for all SBSs to learn the policies is still T . This means even if MBS increases the number of the to-be-served SBSs, it does not increase the total number of computation. Secondly, it is enough to reserve the wireless backhaul resource that can refresh the largest cache only. These two properties are particularly beneficial for the dense small-cell networks where each SBS can be activated or deactivated dynamically as we do not need to vary these external resources along with the activation or deactivation. Next, MBS can learn the policy efficiently since it just needs to sort the UCBs and select the $c_{m(t)}$ highest-ranked files to be cached. The time complexity to sort is $O(|\mathbb{F}_{m(t)}| \log(|\mathbb{F}_{m(t)}|))$. Additionally, the memory resource required to record the observations during the learning process is linear in $|\mathbb{F}_k|$. These two properties imply that user population per small cell has no impact on the time complexity and

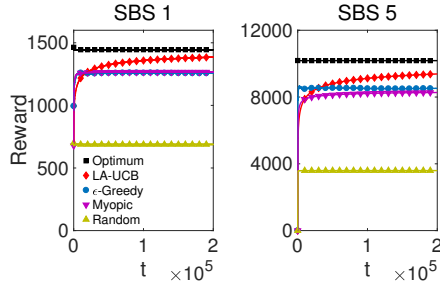


Fig. 1. The time-averaged reward vs time (SBS 1 and SBS 5)

memory resource requirement. This is particularly beneficial for the networks with a diverse population as we do not need to purposely reserve more resources for the small cells having denser populations.

IV. NUMERICAL RESULTS

We conduct experiments to cross-validate Theorem 1 and use the time-averaged reward and regret as performance metrics. We use a dataset from MovieLens where 671 users rated 8944 movies [17]. To reflect the dense deployment with a diverse user population, we deploy 8 small cells and distribute users per small cell biased with some having denser population. Then we can get all $\mathbb{U}_k$. To construct $\mathbb{F}_k$, we take the strategy that if one particular file has received at least one rating from the users in $\mathbb{U}_k$, we add this file into $\mathbb{F}_k$. Let triplet $(k, |\mathbb{U}_k|, |\mathbb{F}_k|)$ be the parameters after construction. Then we can get 8 triplets: (1, 20, 2462), (2, 20, 1090), (3, 45, 2879), (4, 46, 2890), (5, 120, 5401), (6, 120, 4209), (7, 150, 5266), and (8, 150, 4827). We set $T = 200,000$ and the cache size $c_k = 0.25 \times |\mathbb{F}_k|$. We compare LA-UCB with 4 reference algorithms. The first one is the *Optimum* which always plays $\mathbb{S}_k^*, \forall k \in \mathbb{K}$. The second one is the ϵ -Greedy which with probability ϵ selects $c_{m(t)}$ files out of $\mathbb{F}_{m(t)}$ randomly while with probability $1 - \epsilon$ selects the $c_{m(t)}$ highest-ranked files based on empirical mean. We set $\epsilon = 0.25$ in experiments. The third one is the *Myopic*, a modified version of the Least Recently Used and also a reference algorithm in [7], [10]. The *Myopic* keeps all the requested files in last slot in the cache while removing not requested files from the cache. Then the empty space is filled by selecting files randomly until full. The fourth one is the *Random* which selects files randomly.

Fig. 1 shows the time-averaged reward of SBS 1 (20 users) and SBS 5 (120 users) while Fig. 2 shows the regret. The reason why we present these two is they represent the typical setting of a diverse user population, i.e., one SBS has denser population. From the results we can see LA-UCB approaches to the Optimum w.r.t. the time-averaged reward and increases logarithmically at most w.r.t. the regret over time. These observations cross-validate Theorem 1. In contrast, w.r.t. the time-averaged reward, all the three sub-optimal algorithms can only climb to some certain values and then stay there without increasing too much over time anymore while w.r.t. the regret, they increase linearly over time but at different rates.

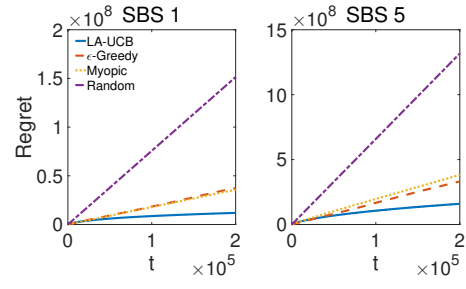


Fig. 2. The regret vs time (SBS 1 and SBS 5)

V. CONCLUSIONS

In this paper, we studied the content caching problem in dense small-cell networks with limited external resources and a diverse user population. We proposed a new algorithm LA-UCB. The algorithm guaranteed the performance of each small cell to approach the optimum even sharing the limited external resources. It also allowed the diversity of user population. The regret analysis and the external resources required of LA-UCB were also given. Experiments confirmed the advantages of LA-UCB and cross-validated the theoretical analysis of the regret.

REFERENCES

- [1] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, 2016.
- [2] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021 White Paper, <http://goo.gl/yITuVx>, 2017.
- [3] Caching Content for Holiday Streaming, <https://goo.gl/sghwEv>, 2015.
- [4] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proc. INFOCOM*, 2010, pp. 1–9.
- [5] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: Wireless video content delivery through distributed caching helpers," in *Proc. INFOCOM*, 2012, pp. 1107–1115.
- [6] K. Poularakis, G. Iosifidis, V. Sourlas, and L. Tassiulas, "Exploiting caching and multicast for 5G wireless networks," *IEEE Transactions on Wireless Communications*, vol. 15, no. 4, pp. 2995–3007, 2016.
- [7] P. Blasco and D. Gunduz, "Learning-based optimization of cache content in a small cell base station," in *Proc. ICC*, 2014, pp. 1897–1903.
- [8] P. Blasco and D. Gunduz, "Multi-armed bandit optimization of cache content in wireless infostation networks," in *Proc. ISIT*, 2014, pp. 51–55.
- [9] P. Blasco and D. Gündüz, "Content-level selective offloading in heterogeneous networks: Multi-armed bandit optimization and regret bounds," *arXiv preprint arXiv:1407.6154*, 2014.
- [10] S. Müller, O. Atan, M. van der Schaar, and A. Klein, "Smart caching in wireless small cell networks via contextual multi-armed bandits," in *Proc. ICC*, 2016, pp. 1–7.
- [11] S. Müller, O. Atan, M. van der Schaar, and A. Klein, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 2, pp. 1024–1036, 2017.
- [12] J. Komiyama, I. Sato, and H. Nakagawa, "Multi-armed bandit problem with lock-up periods," in *Proc. AACL*, 2013, pp. 100–115.
- [13] C. Daskalakis, I. Diakonikolas, R. O'Donnell, R. A. Servedio, and L. Tan, "Learning sums of independent integer random variables," in *Proc. FOCS*, 2013, pp. 217–226.
- [14] Y. H. Wang, "On the number of successes in independent trials," *Statistica Sinica*, pp. 295–312, 1993.
- [15] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2-3, pp. 235–256, 2002.
- [16] [Online]. Available: <https://goo.gl/gZFFGm>.
- [17] F. M. Harper and J. A. Konstan, "The MovieLens Datasets: History and Context," *ACM Transactions on Interactive Intelligent Systems*, vol. 5, no. 4, p. 19, 2016.