Learning Distributed Caching Strategies in Small Cell Networks

Avik Sengupta[†], SaiDhiraj Amuru[‡], Ravi Tandon[†], R. Michael Buehrer[‡], T. Charles Clancy[†]
[†]Hume Center for National Security and Technology, [‡]Wireless@VT,
Virginia Tech, Blacksburg, VA USA
Email: {aviksg, adhiraj, tandonr, rbuehrer, tcc}@vt.edu

***Abstract*—Caching has emerged as a vital tool in modern communication systems for reducing peak data rates by allowing popular files to be pre-fetched and stored locally at end users' devices. With the shift in paradigm from homogeneous cellular networks to the heterogeneous ones, the concept of data offloading to small cell base stations (sBS) has garnered significant attention. Caching at these small cell base stations has recently been proposed, where popular files are pre-fetched and stored locally in order to avoid bottlenecks in the limited capacity backhaul connection link to the core network. In this paper, we study distributed caching strategies in such a heterogeneous small cell wireless network from a reinforcement learning perspective. Using state of the art results, it can be shown that the optimal joint cache content placement in the sBSs turns out to be a NP-hard problem even when the sBS's are aware of the popularity profile of the files that are to be cached. To address this problem, we propose a coded caching framework, where the sBSs learn the popularity profile of the files (based on their demand history) via a combinatorial multi-armed bandit framework. The sBSs then pre-fetch segments of the Fountain-encoded versions of the popular files at regular intervals to serve users' requests. We show that the proposed coded caching framework can be modeled as a linear program that takes into account the network connectivity and thereby jointly designs the caching strategies. Numerical results are presented to show the benefits of the joint coded caching technique over naive decentralized cache placement strategies.**

I. Introduction

Caching has emerged as an important tool in wireless content distribution networks and has garnered significant attention in the paradigm of video distribution and streaming. Caching has the potential to reduce network load by storing content at local caches during times of low network load so that users can be served directly from these caches when the traffic volume is high. Different aspects of caching networks have recently been studied in literature [1]–[4]. Optimal cache placement which accounts for the network topology has been studied in [5], [6] and references therein. In [5], the authors addressed a distributed cache placement problem with an aim to reduce the delay in delivering the files to the end users. A single sBS cache content placement problem was addressed in [6] from a reinforcement learning perspective.

In this work, we study distributed caching strategies in a small cell wireless network from a reinforcement learning perspective. Specifically we use a distributed caching model similar to [5] and present novel caching strategies by using a multi-armed bandit (MAB)-based framework which was earlier used in [6]. The files to be cached are modeled as the arms of the MAB problem. Since, multiple files can be cached at any given time, a more relevant framework is the combinatorial MAB framework [7]. The goal of the caching strategy is to pick the best set of files at any time t based on their popularity so that the requests from users can be directly served by the caches without accessing the core network. Even if the file popularity is known a priori, we show that the optimal distributed cache content placement turns out to be NP-hard. However, in reality the file popularity profile could change over time and hence it is necessary to learn it dynamically during the caching procedure. As it is well known in the MAB literature, there is a tradeoff between the exploration of new arms (i.e., caching new files to estimate their popularity) versus the exploitation of the known arms (caching files that are known to have high popularity and hence give higher rewards). We thus use a UCB-type algorithm [8] to learn the file popularity, where we also take into consideration the fact that the file popularity is modeled by a Zipf distribution [6].

We then present a novel caching strategy that uses the learned file popularity profile to optimize the cache content placement by taking into account the users' connectivity to the small cell base stations (sBS). Specifically, we present an optimal joint cache placement policy by storing segments of the files in the caches and show that it is a convex relaxation of the previously mentioned uncoded problem. This is realized in practice by using rateless codes (for example, fountain codes) [9], that can produce endless streams of coded symbols and ensure complete decoding of the original files when a fixed fraction of these coded symbols are collected. Using numerical simulations, we show that the proposed distributed caching strategy holds an advantage over naive strategies that locally optimize their cache contents without accounting for the network topology. We also show that in a realistic scenario where each user is only connected to few sBSs in the network, the proposed coded cache placement outperforms both the naive coded and uncoded solutions.

The paper is organized as follows: the system model is detailed in Section II, followed by the formulation of the uncoded cache placement as an integer assignment problem in Section III. A learning-based coded caching framework is described in Section IV, the numerical results are presented in Section V and finally Section VI concludes the paper.

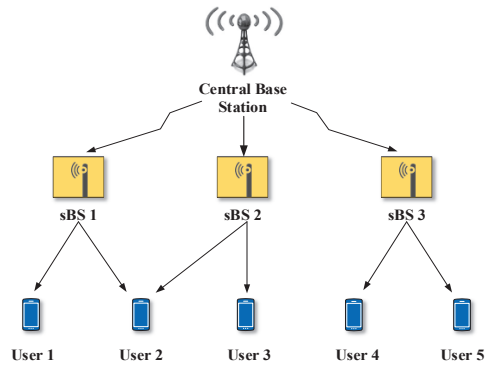

Fig. 1. System Model for Distributed Caching with number of sBSs, N = 3 and the number of users, U = 5.

II. System Model

We consider a set of users $u \in \mathcal{U} = \{1, 2, \ldots, U\}$ in an area served by a central base station (BS) and a set, $\mathcal{S} = \{1, 2, \ldots, N\}$, of sBS's. Users can make random requests from a directory of files $\mathcal{F} = \{1, 2, \ldots, F\}$ where each file f has size S_f bits. The central BS is considered to have a cache memory large enough to store the entire file directory. Each sBS has a cache memory that can store M bits of data. Each user can be served from the cache of either the cellular BS or one or more sBSs in the network. If a user's request cannot be serviced from the caches of the sBS's to which it is connected, then it can be downloaded directly from the central BS. This ensures that all the users' requests are serviced. However, downloading content from a sBS is much more cost effective than downloading the content directly from the central BS and thus the optimal cache placement should minimize the need for servicing user requests directly from the central BS. An example small cell wireless network architecture is shown in Fig. 1. The link from the central BS to the sBSs represents the backhaul links in the wireless network.

The connectivity of the users and sBS can be modeled as a bipartite graph $\mathcal{G} = (\mathcal{S}, \mathcal{U}, \mathcal{E})$, where the edges $(\mathcal{S}_n, u) \in \mathcal{E}$ if there exists a communication link between user u and the n-th sBS. $\mathcal{N}(u) \subseteq \mathcal{S}$ denotes the neighborhood of user u i.e., the sBSs that can serve the user. For example, from Fig. 1, $\mathcal{N}(1) = \{\mathcal{S}_1\}$ while $\mathcal{N}(2) = \{\mathcal{S}_1, \mathcal{S}_2\}$. On the other hand, the neighborhood of the nth sBS i.e., the number of users connected to sBS n is denoted by $\mathcal{U}(\mathcal{S}_n)$. For example, in Fig. 1, $\mathcal{U}(\mathcal{S}_1) = \{1, 2\}$ and $\mathcal{U}(\mathcal{S}_2) = \{2, 3\}$. Let $d^t_{f,n}$ denote the number of requests for file f from the users $u \in \mathcal{U}(\mathcal{S}_n)$ at a time t. The instantaneous demand $d^t_{f,n}$ is an independent and identically distributed random variable with a mean $\theta_{f,n} \in \left[0, |\mathcal{U}(\mathcal{S}_n)|\right]$, where $|\mathcal{U}(\mathcal{S}_n)|$ indicates the cardinality of the set $\mathcal{U}(\mathcal{S}_n)$. In other words, the mean $\theta_{f,n}$ signifies the popularity of the f-th file at the n-th sBS i.e., the expected number of requests for the file f at $\mathcal{S}_n$ at time t. Thus, based on the user requests up to time $(t-1)$, the files have a popularity distribution $\Theta_n = \{\theta_{1,n}, \theta_{2,n}, \ldots, \theta_{F,n}\}$. The true popularity distribution of files in a multimedia network is generally modeled as a ZipF distribution [10], [11]. Thus the mean demand is:

$$\theta_{f,n} = \frac{|\mathcal{U}(\mathcal{S}_n)|}{f^{\gamma} \sum_{k=1}^{F} k^{-\gamma}}, \tag{1}$$

where γ models the skewness of the popularity profile. For example, $\gamma = 0$ models a uniform file popularity profile and as γ increases the skewness increases. We ask the following question in this work: *Given a network topology, sBS cache memory capacity and random file requests from users, what is optimal caching strategy such that a maximum number of requests can be served directly from these caches without accessing the BS?* To answer this question, we formulate the joint cache placement as a reward maximization problem and discuss a reinforcement learning-based coded caching framework.

III. Uncoded Joint Cache Placement

Based on the problem definition mentioned earlier, we model the cache placement problem as follows. When the popularity profile Θ_n of all files $f \in \mathcal{F}$ is known at the sBSs $\mathcal{S}_n$, the average instantaneous reward for each user u in the network is then defined as:

$$\overline{R}_u = \sum_{n=1}^{|\mathcal{N}(u)|} \sum_{f=1}^{F} \theta_{f,n} \cdot S_f \cdot \left[\prod_{\substack{i=1 \\ i \neq n}}^{|\mathcal{N}(u)|} (1 - x^i_f) \cdot x^n_f \right], \tag{2}$$

where x^n_f are the elements of the $F \times N$ cache assignment matrix $\mathbf{X}$ and $x^n_f = 1$ denotes that the file f is cached at the n-th sBS. The expression $\prod_{i=1, i\neq n}^{|\mathcal{N}(u)|}(1 - x^i_f) \cdot x^n_f$ is an indicator function (defined over the set of feasible assignment matrices $\mathbf{X}$) for the condition that the file f is placed in the cache of the n-th sBS of the neighborhood $\mathcal{N}(u)$ of user u. This condition ensures that the file that may be requested by a user connected to several sBS's needs to be placed at only one sBS among the user's neighbor set, so that the remaining cache memory can be utilized to serve other users' requests. In other words, the reward formulation ensures that there is minimal repetitive placement of a file f while concurrently maximizing the number of requests served from the sBS caches. Thus, the optimal joint cache placement maximizes the average instantaneous reward of the users in the network. The cache placement is formulated as the following integer programming problem:

$$\max_{\mathbf{X}} \quad \sum_{u=1}^{U} \overline{R_u} \qquad \text{s.t.} \quad \sum_{f=1}^{F} x^n_f \cdot S_f \leq M, \quad \forall n, \quad \mathbf{X} \in \{0,1\}^{F \times N}. \tag{3}$$

The optimal cache placement in (3) encourages placement of a file f only once in a given user's neighborhood of sBSs so that remaining space in the other caches can be used to service more requests. However, this formulation also allows replication of files in cases where the average instantaneous reward for system, i.e., the amount of data downloaded from the caches on an average is maximized by storing the file at multiple caches. Following [5], it is easy to show that even with knowledge of the popularity profile Θ_n at each sBS, the

Algorithm 1 Estimating File Popularity

1: **for** sBS $\mathcal{S}_n \in \{1, \ldots, N\}$ **do**
2: **if** file f is cached **then**
3: Update $\bar{\theta}_{f,n} = \frac{\sum_{i=1}^{t} d_{f,n}^{i}}{t}$, where $\sum_{i=1}^{t} d_{f,n}^{i}$ is the cumulative sum of requests for file f from users $u \in \mathcal{U}(\mathcal{S}_n)$ until time t.
4: Compute $\hat{\theta}_{f,n} = \bar{\theta}_{f,n} + \frac{|\mathcal{U}(\mathcal{S}_n)| \cdot S_f}{F^{\gamma}} \sqrt{\frac{3\log(|\mathcal{U}(\mathcal{S}_n)|t)}{2|\mathcal{U}(\mathcal{S}_n)|T_{f,n}}}$ where $T_{f,n}$ is the number of times a fraction of file f is cached in $\mathcal{S}_n$ up to time t, $|\mathcal{U}(\mathcal{S}_n)|$ indicates cardinality of the $\mathcal{U}(\mathcal{S}_n)$.
5: **end if**
6: **end for**

integer assignment problem in (3) is a NP-hard problem. In practice, the popularity profile can change with time and may not be known at the sBS, i.e., it must be learnt over time. Also, allowing only a $\{0, 1\}$ file assignment is an assumption which can be too restrictive. To this end, we next propose a formulation which addresses these issues. We model this problem from a reinforcement learning perspective using a coded caching strategy to learn file popularity profile at each sBS and the corresponding optimal cache placement strategy.

IV. Learning the Coded Joint Cache Placement

In order to solve the optimal cache placement problem, we introduce a convex relaxation of the uncoded cache placement wherein file segments can be stored in the caches instead of the entire files. As discussed in [5], this can be achieved in practice by encoding files using a rateless code like Raptor Codes [9]. The sBSs can then store the resulting coded outputs (in other words, the file segments) which enable the original file recovery when a user gathers enough number of segments/fractions. The coded cache placement matrix is denoted by an $F \times N$ matrix $\mathbf{L}$ with $[\mathbf{L}]_{f,n} = [\lambda_{f,n}]$ for $f \in \mathcal{F}$ and $n = 1, \ldots, N$. The fraction $\lambda_{f,n}$ (normalized by file size S_f) is the fraction of coded outputs stored in the cache of $\mathcal{S}_n$.

Since the file popularity profile is *unknown* at each sBS, we introduce a reinforcement learning-based framework to learn it jointly at each sBS, while accounting for the network topology. To this end, we formulate a MAB problem [7], where the files $f \in \mathcal{F}$ are treated as the arms of the MAB framework. Since each cache can choose several files at any time t (assuming $S_f \leq M$), it is appropriate to model it as a CMAB problem [6]. At time t, sBS $\mathcal{S}_n$ learns the file popularity distribution $\bar{\Theta}_n = \{\bar{\theta}_{1,n}, \bar{\theta}_{2,n}, \ldots, \bar{\theta}_{F,n}\}$ based on the history of instantaneous demands i.e., $d_{f,n}^{t}$ as detailed in Algorithm 1. An UCB-based algorithm [8] is used to perturb $\bar{\Theta}_n$ to obtain $\widehat{\Theta}_n$ that are used to determine the optimal cache placement. The perturbation factor in Algorithm 1 is $\frac{|\mathcal{U}(\mathcal{S}_n)| \cdot S_f}{F^{\gamma}} \sqrt{\frac{3\log(|\mathcal{U}(\mathcal{S}_n)|t)}{2|\mathcal{U}(\mathcal{S}_n)|T_{f,n}}}$, where $|\mathcal{U}(\mathcal{S}_n)|$ is the number of users connected to $\mathcal{S}_n$ and accounts of the network topology as opposed to the formulation in [6] which only accounts for a single BS. This factor is based on a modified combinatorial UCB algorithm that promotes exploitation-exploration based on the Zipf distribution i.e., it places files in the caches by artificially increasing the popularity measure of files that are less often cached. The factor $T_{f,n}$ denotes the number of times a file $f \in \mathcal{F}$ has been cached in $\mathcal{S}_n$ until time t and is updated if at time t, a fraction $\lambda_{f,n} > 0$ is stored in $\mathcal{S}_n$.

The optimal cache placement problem is again modeled as a reward maximization problem. Given that a user requests a file f at time t, we first order the sBSs, $n \in \mathcal{N}(u)$, in descending order of the file popularity profile estimate upto time $t-1$, with the first sBS being the one with the highest $\hat{\theta}_{f,n},\ n \in \mathcal{N}(u)$. The fraction of file f that a user is able to download from $\mathcal{S}_n$ is denoted by $\lambda_{f,n} S_f$. Similar to (3), a reward measure for this download is given by $\lambda_{f,n} \hat{\theta}_{f,n} S_f$. The instantaneous reward for a user u that can download file f from the first k sBS's in the ordered list is given by,

$$\begin{aligned} \overline{R}_u^{f,k} &= \sum_{i=1}^{k-1} \lambda_{f,i}\, \widehat{\theta}_{f,i} S_f + \left(1 - \sum_{i=1}^{k-1} \lambda_{f,i}\right) \widehat{\theta}_{f,k} S_f \\ &= \left[\widehat{\theta}_{f,k} - \sum_{i=1}^{k-1} \lambda_{f,i} (\widehat{\theta}_{f,k} - \widehat{\theta}_{f,i})\right] S_f \\ &\qquad\qquad \forall k \in \{1, 2, \ldots, |\mathcal{N}(u)|\}. \end{aligned} \tag{4}$$

The reward function is linear(affine) in terms of the placement variables $\lambda_{f,n}$. The file f can be fully downloaded from the k best caches in the list if and only if $\sum_{i=1}^{k} \lambda_{f,i} \geq 1$. If the whole file cannot be downloaded from all the caches in the list i.e., $\sum_{i=1}^{|\mathcal{N}(u)|} \lambda_{f,i} < 1$, then the remaining fraction can be downloaded from the central BS. The reward for a user u for downloading the file f is thus a piecewise-defined affine function of the placement variables $\lambda_{f,n}$:

$$\overline{R}_u^{f} = \begin{cases} \overline{R}_u^{f,1} & \text{if } \lambda_{f,1} \geq 1 \\ \vdots & \\ \overline{R}_u^{f,j} & \text{if } \begin{array}{l} \sum_{i=1}^{j-1} \lambda_{f,i} < 1 \\ \sum_{i=1}^{j} \lambda_{f,i} \geq 1 \end{array} \\ \vdots & \\ \overline{R}_u^{f,|\mathcal{N}(u)|} & \text{if } \sum_{i=1}^{|\mathcal{N}(u)|-1} \lambda_{f,i} < 1 \end{cases} \tag{5}$$

We aim to maximize the sum of minimum average rewards for all users and files to determine the optimal cache placement. Thus the average instantaneous reward is defined as the pointwise minimum of the piecewise-defined affine function:

$$\bar{R}_u^{f} = \min_{k \in \{1,2,\ldots,|\mathcal{N}(u)|\}} \bar{R}_u^{f,k}, \tag{6}$$

which is a concave function [12] of the placement matrix $\mathbf{L}$.

Next we prove that the expression in (6) is a concave function of $\mathbf{L}$. Suppose that the user downloads the entire file from the first j caches in the list i.e., $\bar{R}_u^{f} = \bar{R}_u^{f,j}$. We have to show that $\bar{R}_u^{f,j} \leq \bar{R}_u^{f,j'}\ \ \forall j \neq j'$. Using (5), the condition is given by:

$$\widehat{\theta}_{f,j} - \sum_{i=1}^{j-1} \lambda_{f,i} (\widehat{\theta}_{f,j} - \widehat{\theta}_{f,i}) \leq \widehat{\theta}_{f,j'} - \sum_{i=1}^{j'-1} \lambda_{f,i} (\widehat{\theta}_{f,j'} - \widehat{\theta}_{f,i}) \tag{7}$$

Considering the case for $j' > j$, we can re-write (7) as:

$$\left(\sum_{i=1}^{j}\lambda_{f,i}-1\right)\left(\hat{\theta}_{f,j}-\hat{\theta}_{f,j'}\right)+\sum_{i=j+1}^{j'-1}\lambda_{f,i}\left(\hat{\theta}_{f,i}-\hat{\theta}_{f,j'}\right)\geq 0 \quad (8)$$

This is easily verified as $\sum_{i=1}^{j}\lambda_{f,i}\geq 1$ and for all $j^{'}>j$, we have $(\hat{\theta}_{f,j}-\hat{\theta}_{f,j'})\geq 0$. Also since $i<j^{'}$ and $(\hat{\theta}_{f,i}-\hat{\theta}_{f,j'})\geq 0$. Therefore (8) is satisfied. The proof for $j^{'}<j$ is similar. Thus, we have shown that $\bar{R}_u^f$ is a concave function of $\mathbf{L}$. The optimal coded cache placement problem can then be formulated as the following convex optimization[1] problem:

$$\begin{aligned}\max_{\mathbf{L}} \quad & \sum_{u=1}^{U}\sum_{f=1}^{F}\min_{k\in\{1,2,\ldots,|\mathcal{N}(u)|\}}\left\{\overline{R}_u^{f,k}\right\}\\ \text{s.t.} \quad & \sum_{f=1}^{F}\lambda_f^n\cdot S_f\leq M, \quad \forall n,\\ & \mathbf{L}\in[0,1]^{F\times N}. \end{aligned} \quad (9)$$

Similar to [5], the optimization in (9) can be reduced to the following LP by introducing the auxiliary variable y_u^f and the resulting optimization problem is given by,

$$\begin{aligned}\max_{\mathbf{L}} \quad & \sum_{u=1}^{U}\sum_{f=1}^{F}\mathrm{y}_u^f\\ \text{s.t.} \quad & \mathrm{y}_u^f\leq\overline{R}_u^{f,k} \qquad \forall k\in\{1,2,\ldots,|\mathcal{N}(u)|\}\\ & \sum_{f=1}^{F}\lambda_f^n\cdot S_f\leq M, \qquad \forall n,\\ & \mathbf{L}\in[0,1]^{F\times N}. \end{aligned} \quad (10)$$

The average instantaneous reward obtained with the coded cache placement is an improvement over the uncoded case as the $\{0,1\}$ binary assignment of the uncoded scheme is also a feasible solution to the coded caching problem [5]. Thus the coded caching is a convex relaxation of the uncoded case. The reward value is however also dependent on the learning problem, the UCB perturbation and also the network connectivity. In the next section, numerical results are presented for the proposed coded caching framework.

V. Numerical Results

The performance of the proposed joint learning and cache placement strategy is compared against two naive schemes which give further insight into the behavior of the small cell caching network. The two naive schemes are as follows:

- *Uncoded Local Learning Scheme:* In this scheme, uncoded cache placement is performed locally at each sBS. In this procedure, every sBS learns the file popularity profile from its own set of connected users without taking into account the overall network connectivity. In other words, the exploration-exploitation based perturbation in this naive scheme is based only on the local cache placement. This scheme entails the solving the following knapsack problem at each sBS:

$$\max_{\mathbf{X}} \quad \sum_{f=1}^{F}\hat{\theta}_f S_f x_f \quad \text{s.t.} \quad \sum_{f=1}^{F}x_f\cdot S_f\leq M, \quad \forall n, \quad \mathbf{X}\in\{0,1\}^{F\times 1}. \quad (11)$$

- *Coded Local Learning Scheme:* In this case the learning process is similar to the previous case. However, the file placement is relaxed to store fractions of the rateless encoded files entailing the solution of the following linear optimization problem at each sBS:

$$\max_{\mathbf{L}} \quad \sum_{f=1}^{F}\hat{\theta}_f S_f \lambda_f \quad \text{s.t.} \quad \sum_{f=1}^{F}\lambda_f\cdot S_f\leq M, \quad \forall n, \quad \mathbf{L}\in[0,1]^{F\times 1}. \quad (12)$$

For the simulations, a network of 5 sBSs, 30 users, 50 files and a cache size of $M=30$ was used. The files are chosen to have sizes $S_f\in\{1,3,5,7,9\}$. The users' requests were randomly generated by sampling a Zipf file popularity distribution with $\gamma=2$. Each sBS then used the learning procedure pertinent to the caching scheme in order to estimate γ [6] and subsequently the popularity profile. The coded distributed (joint learning) cache placement scheme is compared against the naive coded and uncoded local learning based schemes in Fig. 2. The metric which is observed is the instantaneous reward i.e., the amount of data downloaded from the caches at a given time instant. It can be seen that the solutions converge quickly

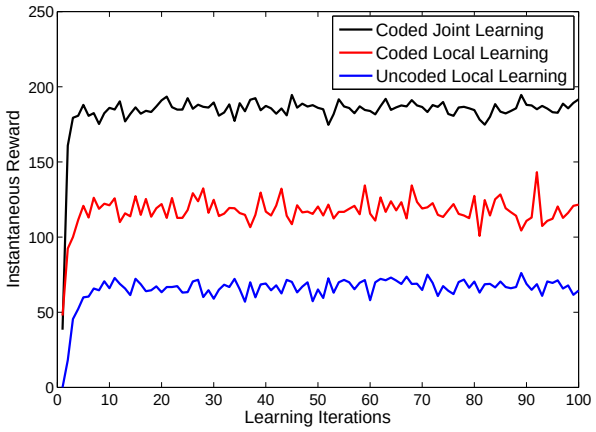

Fig. 2. Instantaneous rewards for the Coded Caching schemes . The parameters used in the simulations are $N=5, U=30, F=30, M=30, S_f\in\{1,3,5,7,9\}, \gamma=2$.

and the coded joint placement outperforms the coded and uncoded local learning based schemes. This is expected as the joint learning accounts for the connectivity and topology of the network while optimally placing file segments in the sBS caches. This policy ensures that minimal replication of file fragments occur in those caches which have common users in the underlying wireless network. The connectivity

[1]Maximizing a concave function subject to constraints is equivalent to minimizing a convex function subject to the same constraints [12].

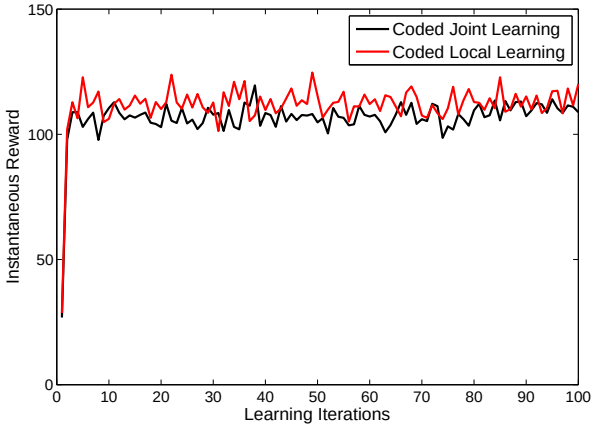

Fig. 3. Instantaneous rewards for the case of sparsely connected network. The parameters used in the simulations are $N = 5, U = 30, F = 30, M = 30, S_f \in \{1, 3, 5, 7, 9\}, \gamma = 2$.

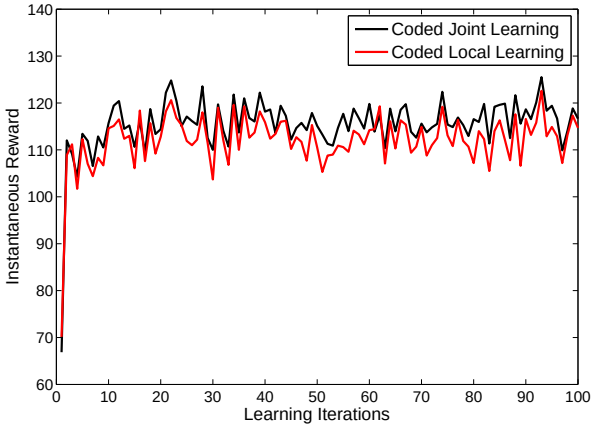

Fig. 4. Instantaneous rewards for the case of densely connected network. The parameters used in the simulations are $N = 5, U = 30, F = 30, M = 30, S_f \in \{1, 3, 5, 7, 9\}, \gamma = 2$.

matrix of the network, $\mathcal{C} \in \{0, 1\}^{U \times N}$, defines the wireless communication links in $\mathcal{G} = (\mathcal{S}, \mathcal{U}, \mathcal{E})$. The sparsity of the connectivity matrix has interesting implications on the instantaneous reward. For the simulation in Fig. 2, a relatively non-sparse connectivity matrix was used. This is the most realistic scenario where each user is connected to a small subset of sBSs in the network. Under such a network topology, the joint learning holds a clear advantage.

In a sparsely connected network, the advantage of joint learning diminishes as the joint optimization problem represents a decoupled situation i.e. it reduces to the local learning problem. It can be seen from Fig. 3 the rewards for the two cases under this network topology are almost identical. In fact the local learning holds an edge in this case as the exploration-exploitation in the joint learning case leads to falsely optimistic estimates of popularity and the few users with multiple connectivity tend to adversely influence the cache placement.

The other extreme is the case of a very densely connected network, where almost all the users are connected to almost all the sBSs in the network. In such a scenario, each sBS sees almost an identical set of requests and the popularity estimates across the sBSs are almost identical. In this case the ordering of caches for file placement based on $\hat{\theta}_{f,n}$ as discussed in Section IV becomes close to uniform. Thus it is expected that the performance of the joint and the local scheme will again be close to each other. This behavior is observed in Fig. 4. However in the densely connected case, the joint learning holds an advantage as the joint exploration-exploitation procedure yields higher dividends as more caches share common users and hence the joint learning better optimizes the placement (for example, by minimizing the file replication).

VI. Conclusions

In this paper, we addressed the problem of distributed cache placement in a small cell network from a reinforcement learning perspective. The optimal uncoded cache placement problem was shown to be NP hard even with the knowledge of the file popularity distribution. A convex relaxation to the uncoded problem, namely, the distributed coded cache placement problem was formulated. It was shown to be reducible to a linear program form yielding efficient solutions. Through numerical simulations, distributed coded cache placement was shown to outperform the naive local learning based schemes for relatively densely connected network topologies. The network topology, for example sparse networks and dense networks, was also shown to have an effect on the MAB based learning which directly effects the cache placement strategies.

References

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

[2] M. A. Maddah-Ali and U. Niesen, "Decentralized Coded Caching Attains Order-Optimal Memory-Rate Tradeoff," *to appear in IEEE/ACM Trans. on Networking*.

[3] M. Ji, G. Caire, and A. F. Molisch, "Optimal throughput-outage trade-off in wireless one-hop caching networks," *arXiv : 1302.2168*, Feb 2013.

[4] M. Ji, G. Caire, and Molisch, "Wireless device-to-device caching networks: Basic principles and system performance," *arXiv : 1305.5216*, May 2013.

[5] N. Golrezaei, K. Shanmugam, A. Dimakis, A. Molisch, and G. Caire, "FemtoCaching: Wireless Video Content Delivery through Distributed Caching Helpers," *IEEE Transactions on Information Theory*, vol. 59(12), pp. 8402–8413, Dec. 2013.

[6] P. Blasco and D. Gündüz, "Learning-Based Optimization of Cache Content in a Small Cell Base Station," in *IEEE ICC*, June 2014.

[7] W. Chen, Y. Wang, and Y. Yuan, "Combinatorial multi-armed bandit: General framework and applications," in *ICML (1)*, 2013, pp. 151–159.

[8] 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. [Online]. Available: http://dx.doi.org/10.1023/A%3A1013689704352

[9] A. Shokrollahi, "Raptor codes," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, June 2006.

[10] M. Hefeeda and O. Saleh, "Traffic modeling and proportional partial caching for peer-to-peer systems," *IEEE/ACM Transactions on Networking*, vol. 16, no. 6, pp. 1447–1460, Dec. 2008.

[11] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: evidence and implications," in *IEEE INFOCOM*, vol. 1, 1999, pp. 126–134 vol.1.

[12] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.