REINFORCEMENT LEARNING FOR 5G CACHING WITH DYNAMIC COST

Alireza Sadeghi[†], *Fatemeh Sheikholeslami*[†], *Antonio G. Marques*[*], *and Gergios B. Giannakis*[†]

[†]Digital Technology Center and Dept. of ECE, University of Minnesota, Minneapolis, USA
[*]Dept. of Signal Theory and Comms., King Juan Carlos University, Madrid, Spain

ABSTRACT

In next generation cellular networks (5G) the access points (APs) are anticipated to be equipped with storage devices to serve locally requests for reusable popular contents by *caching* them at the edge of the network. The ultimate goal is to shift part of the load on the back-haul links from on-peak to off-peak periods, contributing to a better overall network performance and service experience. In order to enable the APs with efficient (optimal) fetch-cache decision making schemes able to work in dynamic settings, we introduce simple but flexible generic time-varying fetching and caching costs, which are then used to formulate a constrained minimization of the aggregate cost across files and time. Since caching decisions in every time slot influence the content availability in future instants, the novel formulation for optimal fetch-cache decisions falls into the class of dynamic programming, for which efficient reinforcement-learning-based solvers are proposed. The performance of our algorithms is assessed via numerical tests, and discussions on the inherent fetching-versus-caching trade-off are provided.

Index Terms— Caching, Fetching, Dynamic Programming, Resource allocation, Dynamic pricing.

1. INTRODUCTION

In the era of date deluge, storing "popular" contents at the edge of 5G cellular networks is a promising technique to satisfy the users' demands while alleviating the congestion on the back-haul links. To this aim, access points (APs) equipped with a local cache must intelligently store reusable popular contents during off-peak periods, and utilize the stored data during on-peak hours [1]. Due to its practical relevance, design of optimal caching schemes is quickly gaining traction. In order to enable content-agnostic APs with the required learning capability, a multi-armed bandit formulation of the problem in a static "popularity" scenario was investigated in [2]. The coded, convexified, and distributed extensions of this problem were later studied in [3]. Context and trend-aware learning approaches were considered in [4] and [5]. From a similar perspective, an asynchronous approach based on reinforcement learning was adopted in [6], where Markov chains were used for modeling *dynamic* user demands.

However, most existing efforts for 5G caching have focused on enabling the APs for optimal caching in *static* scenarios *with fixed costs*. By considering generic time-varying fetching and caching costs, this paper aims at designing more flexible chaching schemes, while enabling APs to learn the optimal fetching-caching decisions. In particular, the caching and fetching schemes will be found as the solution of a constrained optimization aimed at reducing the cost aggregated across files and time instants. Since the caching decision in a given time slot not only affects the instantaneous cost, but also will influence the cache availability in future, the problem must be handled using dynamic programming (DP) tools. The optimization is shown to be separable across files, and thus it can be efficiently solved by decomposing the value function associated with the original DP into a summation of smaller-dimension value functions. To reduce the computational complexity, a reduced (marginalized) version of the so-called value iteration algorithm [7] is introduced, and its performance is assessed via numerical tests.

The work in this paper has been supported by USA NSF grants 1423316, 1508993, 1514056, 1711471, and by the Spanish MINECO grant OMICROM (TEC2013-41604-R).

2. OPERATING CONDITIONS AND COSTS

Consider a memory-enabled access point (AP) responsible for serving user file requests over $f = 1, 2, \ldots, F$ contents across time. The requested contents are transmitted to users either by fetching through a (expensive) back-haul link connecting the AP to the cloud, or by utilizing the local storage unit in the AP to pro-actively cache the popular contents ahead of time. The system is considered to operate in a slotted fashion with $t = 1, 2, \ldots$ denoting time.

During slot t and given the available cache contents, the AP receives a number of file requests whose provision incurs certain costs. Specifically for a requested file f, fetching it from the cloud through the back-haul link gives rise to scheduling, routing and transmission costs, whereas its availability at the cache storage in the AP will eliminate such expenses. However, local caching also incurs a number of costs, such as storage, memory, or energy consumption, for each time instant. This gives rise to an inherent caching-versus-fetching trade-off, where one is promoted over the other depending on their relative costs. The objective here is to propose a simple but sufficiently general framework to minimize the sum-average cost over time by optimizing fetch-cache decisions while adhering to the constraints inherent to the operation of the system and the requirements from the users. The variables, constraints and costs involved in this optimization are described in the ensuing subsections.

2.1. Variables and constraints

Consider the system at time slot t, where the binary state variable r_t^f represents the incoming request for file f; that is, $r_t^f = 1$ if the file f is requested during slot t, and $r_t^f = 0$ otherwise. Here, we assume that $r_t^f = 1$ necessitates serving the file to the user and dropping is not allowed, thus requests must be carried out either by fetching the file from the cloud or by utilizing cache capacity. Furthermore, at the end of each time slot, the AP will decide if content f should be stored in cache for possible use in subsequent slot.

To formalize this, let us define the "fetching" decision variable $w_t^f \in \{0,1\}$ along the "caching" variable $a_t^f \in \{0,1\}$. Having $w_t^f = 1$ implies "fetching" file f at time t, while $w_t^f = 0$ means "no-fetching". Similarly, $a_t^f = 1$ implies that content f will be stored in cache at the end of the slot t, while $a_t^f = 0$ implies that it will not. Furthermore, let the storage state variable $s_t^f \in \{0,1\}$ account for the availability of the files at the local cache. In particular, $s_t^f = 1$ if file f is available in the cache at the beginning of slot t, and $s_t^f = 0$ otherwise. Since the availability of file f directly depends on caching decision at time $t-1$, we have that

$$s_t^f = a_{t-1}^f, \quad \forall f, t, \tag{1}$$

which will be incorporated to our optimization as constraints.

Moreover, since having $r_t^f = 1$ necessitates the transmission of file f to the user(s), it requires either having the file in cache ($s_t^f = 1$) or fetching it from the cloud ($w_t^f = 1$), giving rise to the second set of constraints

$$r_t^f \leq w_t^f + s_t^f, \quad \forall f, t. \tag{2}$$

Finally, the caching decision a_t^f can be set to 1 only when the content f is available at time t; that is, only if either fetching is carried out $w_t^f = 1$ or the current cache state is $s_t^f = 1$. This in turn implies the third set of constraints as

$$a_t^f \leq s_t^f + w_t^f, \quad \forall f, t. \tag{3}$$

2.2. Prices and aggregated costs

To account for the caching and fetching costs, let ρ_t^f and λ_t^f denote the (generic) costs associated with $a_t^f = 1$ and $w_t^f = 1$, respectively. Focusing for now on the caching cost and with σ_f denoting the size of content f, a simple form for ρ_t^f is

$$\rho_t^f = \sigma_f({\rho'}_t + {\rho'}_t^f) + ({\rho''}_t + {\rho''}_t^f), \tag{4}$$

where the first term is proportional to the file size σ_f and the second one is constant. Note also that we consider file-dependent costs (via variables ${\rho'}_t^f$ and ${\rho''}_t^f$) as well as cost contributions which are common across files (via ${\rho'}_t$ and ${\rho''}_t$). In most practical setups the latter will dominate over the former. For example, the caching cost per bit is likely to be the same regardless of the particular type of content, so that ${\rho'}_t^f = {\rho''}_t^f = 0$. From a modeling perspective, variables ρ_t^f can correspond to actual prices paid to an external entity (e.g., if associated with energy consumption costs), marginal utility or cost functions, congestion indicators, Lagrange multipliers associated with constraints, or linear combinations of those [7, 8, 9]. Analogously, the corresponding form for the fetching cost is

$$\lambda_t^f = \sigma_f({\lambda'}_t + {\lambda'}_t^f) + ({\lambda''}_t + {\lambda''}_t^f), \tag{5}$$

As before, if the transmission link from cloud to the AP is the same for all contents, the prices ${\lambda'}_t$ and ${\lambda''}_t$ are expected to dominate over the file-dependent counterparts ${\lambda'}_t^f$ and ${\lambda''}_t^f$.

Upon defining $c_t^f(a_t^f, w_t^f; \rho_t^f, \lambda_t^f) = \rho_t^f a_t^f + \lambda_t^f w_t^f$, the aggregate cost at time t is given by

$$c_t := \sum_{f=1}^{F} c_t^f(a_t^f, w_t^f; \rho_t^f, \lambda_t^f) = \sum_{f=1}^{F} \rho_t^f a_t^f + \lambda_t^f w_t^f, \tag{6}$$

which is the basis for the DP formulated in the next section.

3. FETCH-CACHE DECISION MAKING VIA DP

Since decisions are coupled across time [cf. constraint (1)], and the future value of prices as well as state variables are inherently random, our goal is to optimize the long-term average aggregate cost

$$\bar{\mathcal{C}} := \mathbb{E}\left[\sum_{t=0}^{\infty}\sum_{f=1}^{F}\gamma^t c_t^f\left(a_t^f, w_t^f; \rho_t^f, \lambda_t^f\right)\right] \tag{7}$$

where the expectation is taken with respect to (w.r.t.) the random variables $\{r_t^f, \lambda_t^f, \rho_t^f\}$, and $0 < \gamma < 1$ is the discounting factor whose tuning trades off current versus future costs. In the rest of the paper, the prices and state variables $\{r_t^f, \lambda_t^f, \rho_t^f\}$ are assumed to be stationary, therefore the expectations can be practically estimated.

We investigate a setup where knowledge of the state information is causal, that is, the exact values of $\{r_t^f, \rho_t^f, \lambda_t^f\}$ are revealed at the beginning of each slot t, and fetch-cache decisions are made sequentially per slot. Hence, the goal is to make *real-time* fetch-cache decisions that minimize the expected *current plus future cost* while adhering to operational constraints, giving rise to the following optimization

$$\text{(P1)} \quad \min_{\{(w_k^f, a_k^f)\}_{f, k\geq t}} \bar{\mathcal{C}}_t := \sum_{k=t}^{\infty}\sum_{f=1}^{F}\gamma^{k-t}\mathbb{E}\left[c_k^f\left(a_k^f, w_k^f; \rho_k^f, \lambda_k^f\right)\right]$$
$$\text{s.t.} \quad (w_k^f, a_k^f) \in \mathcal{X}(r_k^f, s_k^f), \quad \forall f, \; k \geq t$$

where $\mathcal{X}(r_t^f, s_t^f) := \{(w,a) \mid w \in \{0,1\}, a \in \{0,1\}, s_t^f = a_{t-1}^f, \; r_t^f \leq w + s_t^f, \quad a \leq s_t^f + w\}$ and the expectation is taken w.r.t. $\{r_k^f, \rho_k^f, \lambda_k^f\}_{\forall k \geq t+1}$. In contrast to many resource allocation problems where, after introducing pertinent prices (Lagrange multipliers), the optimization decouples across time [8, 9], the presence of constraint (1) entails that current caching decisions impact future costs and therefore such costs must be taken into account. This ultimately implies that (P1) is a DP [10, p. 79] and, therefore, to solve it we need to: a) identify the current and the expected future aggregate cost (this second term will give rise to the so-called value function); b) write the corresponding Bellman equations; and c) propose a method to estimate the value function. This is the subject of the ensuing subsections, which start by further exploiting the problem structure to reduce complexity.

3.1. Bellman equations for the per-content problem

Focusing on (P1) one can readily realize that: (i) consideration of the content-dependent prices renders the objective in (P1) separable across f, and (ii) the constraints in (P1) are also separable across f. Furthermore, the decisions a_t^f and w_t^f do not affect the values (distribution) of $\{r_{t'}^{f'}, \rho_{t'}^{f'}, \lambda_{t'}^{f'}\}$ for files $f' \neq f$ and for times $t' > t$. Thus (P1) naturally gives rise to the per-file optimization

$$\text{(P2)} \quad \min_{\{(w_k^f, a_k^f)\}_{k\geq t}} \bar{\mathcal{C}}_t^f := \sum_{k=t}^{\infty}\gamma^{k-t}\mathbb{E}\left[c_k^f\left(a_k^f, w_k^f; \rho_k^f, \lambda_k^f\right)\right]$$
$$\text{s.t.} \quad (w_k^f, a_k^f) \in \mathcal{X}(r_k^f, s_k^f), \quad k \geq t$$

which must be solved for $f = 1, ..., F$. Indeed, the aggregate cost associated with (P2) will not depend on variables associated with files $f' \neq f$ [7]. This is the case if, for example, the involved variables are independent of each other (which is the setup considered

$$\left(w_t^{f*}, a_t^{f*}\right) := \arg\min_{(w,a)\in\mathcal{X}(r_t^f, s_t^f)} \left\{ \mathbb{E}_{r_k^f,\rho_k^f,\lambda_k^f} \left[\min_{(w_k,a_k)\in\mathcal{X}(r_k^f,s_k^f)} \left\{ \sum_{k=t}^{\infty} \gamma^{k-t} \left[c_k^f(a_k^f, w_k^f; \rho_k^f, \lambda_k^f) \middle| a_t^f = a, w_t^f = w, \boldsymbol{\theta}_t^f = \boldsymbol{\theta}_0^f \right] \right\} \right] \right\} \tag{8}$$

$$= \arg\min_{(w,a)\in\mathcal{X}(r_t^f, s_t^f)} \left\{ c_t^f(a, w; \rho_t^f, \lambda_t^f) + \mathbb{E}_{r_k^f,\rho_k^f,\lambda_k^f} \left[\min_{(w_k,a_k)\in\mathcal{X}(r_k^f,s_k^f)} \sum_{k=t+1}^{\infty} \gamma^{k-t} \left[c_k^f(a_k^f, w_k^f; \rho_k^f, \lambda_k^f) \middle| s_{t+1}^f = a \right] \right] \right\} \tag{9}$$

$$V^f\left(s^f, r^f; \rho^f, \lambda^f\right) := \min_{(w,a)\in\mathcal{X}(r_t^f, s_t^f)} \left\{ \mathbb{E}_{r_k^f,\rho_k^f,\lambda_k^f} \left[\min_{(w_k,a_k)\in\mathcal{X}(r_k^f,s_k^f)} \left\{ \sum_{k=t}^{\infty} \gamma^{k-t} \left[c_k^f(a_k^f, w_k^f; \rho_k^f, \lambda_k^f) \middle| a_t^f = a, w_t^f = w, \boldsymbol{\theta}_t^f = \boldsymbol{\theta}^f \right] \right\} \right] \right\} \tag{10}$$

$$\bar{V}^f(s^f) := \mathbb{E}_{r^f,\rho^f,\lambda^f} \left[\min_{(w,a)\in\mathcal{X}(r_t^f, s_t^f)} \left\{ \mathbb{E}_{r_k^f,\rho_k^f,\lambda_k^f} \left[\min_{(w_k,a_k)\in\mathcal{X}(r_k^f,s_k^f)} \left\{ \sum_{k=t}^{\infty} \gamma^{k-t} \left[c_k^f(a_k^f, w_k^f; \rho_k^f, \lambda_k^f) \middle| a_t^f = a, w_t^f = w, \boldsymbol{\theta}_t^f = \boldsymbol{\theta}^f \right] \right\} \right] \right\} \right]$$

$$= \mathbb{E}_{r^f,\rho^f,\lambda^f} \min_{(w,a)\in\mathcal{X}(r^f,s^f)} \left\{ c_0^f(a, w; \rho^f, \lambda^f) + \gamma \bar{V}^f(a) \right\} \tag{11}$$

here) or when the focus is on a large system where the contribution of an individual variable to the aggregate network behavior is practically negligible.

Bellman equations and value function: Finding the solution to the DP in (P2) requires writing the corresponding Bellman equations and associated value function [10, p. 68]. To this end, consider the system at time t where the cache state $s_t^f = s_0^f$, file request $r_t^f = r_0^f$ as well as cost parameters $\lambda_t^f = \lambda_0^f$ and $\rho_t^f = \rho_0^f$ are all given. For brevity consider the vector $\boldsymbol{\theta}_t^f := [r_t^f, \rho_t^f, \lambda_t^f]$ collecting the state variables at time t for file f, then, the optimal fetch-cache decision (w_t^{f*}, a_t^{f*}) is readily expressible as the solution to (8)[1]. The objective in (8) is then rewritten in (9) as the summation of current and discounted average future costs. The form of (9) is testament to the fact that problem (P2) is a DP and the caching decision a influences not only the current cost $c_t^f(\cdot)$, but also future costs through the second term as well. Bellman equations can be leveraged for tackling such a DP. Under the stationarity assumption for variables $\{r_t^f, \rho_t^f, \lambda_t^f\}$, the term accounting for the future cost can be rewritten in terms of the *stationary value function* $V^f\left(s^f, r^f; \rho^f, \lambda^f\right)$ [10, p. 68]. This function, formally defined in (10), captures the minimum sum average cost for the "state" (s^f, r^f), parametrized by (λ^f, ρ^f), where for notational convenience, we define $\boldsymbol{\theta}^f := [r^f, \rho^f, \lambda^f]$.

3.2. Reduced value function

If one further assumes that price parameters and requests are i.i.d. across time, it can be shown that the optimal solution to (P2) can be expressed in terms of the *reduced value function* [7]

$$\bar{V}^f\left(s^f\right) := \mathbb{E}_{r^f,\lambda^f,\rho^f}\left[V^f\left(s^f, r^f; \rho^f, \lambda^f\right)\right], \tag{12}$$

where the expectation is w.r.t $\{r^f, \rho^f, \lambda^f\}$. This is important not only because it captures the average future cost of file f for cache state $s^f \in \{0, 1\}$, but also because $\bar{V}^f(\cdot)$ is a function of a binary variable, and therefore, its estimation requires only estimating two values. This contrasts with the original four-dimensional value function in (10), which is much harder to estimate.

[1]In equations (8)-(11), the expectation (or minimization) w.r.t. variables that have subscript k, the corresponding expectation (or minimization) is w.r.t. $\forall k \geq t + 1$.

Algorithm 1: Value iteration for finding $\bar{V}^f(\cdot)$

```
1 Set $\bar{V}_0^f(s) = 0$, for $s \in \{0, 1\}$ ;
  Input  : $\gamma < 1$, probability density function of $\rho_t^f$, $\lambda_t^f$ and $r^f$
  Output: $\bar{V}^f(\cdot)$
2 while $|\bar{V}_k^f(s) - \bar{V}_{k+1}^f(s)| < \epsilon; s \in \{0, 1\}$ do
3     for $s = 0, 1$ do
4         $\bar{V}_{k+1}^f(s) =$
          $\mathbb{E}_{r,\rho,\lambda} \min_{(w,a)\in\mathcal{X}(r,s)} \left\{ c_t^f(a, w; \rho, \lambda) + \gamma \bar{V}_k^f(a) \right\}$
5     end
6     $k = k + 1$
7 end
```

By rewriting the proposed alternative value function $\bar{V}^f(\cdot)$ in a recursive fashion as the summation of instantaneous cost and discounted future values $\bar{V}^f(\cdot)$, one readily arrives at the Bellman equations provided in (11). Thus, the problem reduces to finding $\bar{V}^f(0)$ and $\bar{V}^f(1)$ for all f, after which the optimal fetch-cache decisions (w_t^{f*}, a_t^{f*}) are easily found as the solution to

$$\text{(P3)} \quad \min_{(w,a)} \quad c_t^f(a, w; \rho_t^f, \lambda_t^f) + \gamma \bar{V}^f(a)$$
$$\text{s.t.} \quad (w, a) \in \mathcal{X}(r_t^f, s_t^f).$$

The well-known *value iteration* algorithm is leveraged to find $\bar{V}^f(\cdot)$ as tabulated in Algorithm 1; c.f. [10, p. 100] for a detailed discussion on value iteration. Alternatively, if the distributions of the random parameters are unknown, stochastic Q-learning schemes that estimate the *proposed* value function *online* while making decisions can be developed [10, p. 148].

Remark (Augmented value functions). The value function $\bar{V}^f(s^f)$ can be redefined to account for additional information on r_t^f, ρ_t^f or

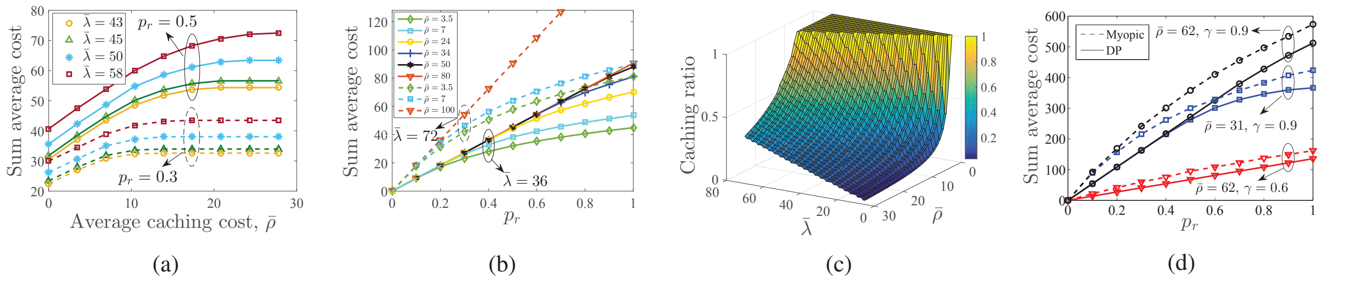

Fig. 1: Average cost versus $\bar{\rho}$ for different values of $\bar{\lambda}$ and p_r (a). Average cost versus p_r for different values of $\bar{\rho}$, $\bar{\lambda}$ (b). Caching ratio vs. $\bar{\rho}$ and $\bar{\lambda}$ for $p_r = 0.5$ and $s = r = 1$ (c). Performance of DP versus myopic caching for $\bar{\lambda} = 53$ (d).

λ_t^f upon existence. Consider for example that the distribution of r_t^f can be parametrized by p_r^f which measures the "popularity" of the content [11]. In such cases, the value function can incorporate the popularity parameter as an additional input to yield $\bar{V}^f(s^f, p_r^f)$. Consequently, the optimal decisions will depend not only on the current requests and prices but also on the (current) popularity p_r^f which will affect future requests. This indeed broadens the scope of the proposed approach, since certain types of *non-stationary* distributions for r_t^f can be handled by enabling the parameter p_r^f to (slowly) vary with time.

4. NUMERICAL TESTS

In this section, the performance of the proposed approach for learning optimal fetch-cache decisions is assessed via numerical tests. Cache and fetch cost parameters are drawn from uniform distributions with mean $\bar{\rho}^f$ and $\bar{\lambda}^f$, respectively. Furthermore, the request variable r^f is considered to be Bernoulli with mean p_r^f, whose value indicates the popularity of file f. The simulations are carried out for a content of unit size, and can be readily extended to files of different sizes. To help readability, we drop the superscript f in this section.

Fig. 1a plots the sum average cost $\bar{C}$ versus $\bar{\rho}$ for different values of $\bar{\lambda}$ and p_r. Parameter $\bar{\lambda}$ is varied over $\{43, 45, 50, 58\}$ for two different values of popularity $p_r \in \{0.3, 0.5\}$. As depicted, higher values of $\bar{\rho}, \bar{\lambda}, p_r$ generally lead to a higher average cost. In particular, when $\bar{\rho} \ll \bar{\lambda}$, caching is considerably cheaper than fetching, thus setting $a_t = 1$ is optimal for most t. As a consequence, the total cost linearly increases with $\bar{\rho}$ as most requests are met via cached contents rather than fetching. Interestingly, if $\bar{\rho}$ keeps increasing, the aggregate cost gradually saturates and does not grow anymore. The reason behind this phenomenon is the fact that, for very high values of $\bar{\rho}$, fetching becomes the optimal decision for meeting most file requests and, hence, the aggregate cost does not depend on $\bar{\rho}$ anymore. While this behavior occurs for the two values of p_r, we observe that for the smallest one, the saturation is more abrupt and takes place at a lower $\bar{\rho}$. The intuition in this case is that for lower popularity values, the file is requested less frequently, thus the caching cost aggregated over a (long) period of time often exceeds the "reward" obtained when (infrequent) requests are served by the local cache. As a consequence, fetching in the infrequent case of $r_t = 1$ incurs less cost than the caching cost aggregated over time. To corroborate these findings, Fig. 1b depicts the sum average cost versus p_r for different values of $\bar{\rho}$ and $\bar{\lambda}$. The results in the figure show that for large values of $\bar{\rho}$ fetching is the optimal action, resulting in a linear increase of the total cost as p_r increases. In contrast, for small values of $\bar{\rho}$ caching is chosen more frequently, resulting in a sublinear cost growth.

To investigate the caching-versus-fetching trade-off for a broader range of $\bar{\rho}$ and $\bar{\lambda}$, let us define the *caching ratio* as the aggregated number of positive caching decisions (those for which $a_t = 1$) divided by the total number of decisions. Fig. 1c plots this ratio for different values of $(\bar{\rho}, \bar{\lambda})$ and fixed $p_r = 0.5$. As the plot demonstrates, when $\bar{\rho}$ is small and $\bar{\lambda}$ is large, files are cached almost all the time, with the caching ratio decreasing (non-symmetrically) as $\bar{\rho}$ increases and $\bar{\lambda}$ decreases.

Finally, Fig. 1d compares the performance of the proposed DP-based strategy with that of a myopic one. The myopic policy sets $a_t = 1$ if $\lambda_t > \rho_t$ and the content is locally available (either because $w_t = 1$ or because $s_t = 1$); and sets $a_t = 0$ otherwise. The results indicate that the proposed strategy outperforms the myopic one for all values of $\bar{\rho}, \bar{\lambda}, p_r$ and γ.

5. CONCLUSIONS AND FUTURE WORK

A wireless setup where an AP makes sequential fetch-cache decisions based on dynamic user requests as well as costs was investigated. Critical constraints were identified, generic time-varying prices were considered, and the cost aggregated across flows and time instants was formed. The proposed DP problem was solved via reinforcement learning where a value iteration algorithm (operating on a reduced-version of the value function) was put forth. Preliminary simulations demonstrated the benefits of the proposed approach and motivate additional work. Future research includes: design of enhanced suboptimal schemes; particularization of the form of the dynamic prices to account for specific relevant operating conditions (including finite-size limited storage); and consideration of scenarios where the distribution of the state variables is tracked and the parameters describing such distributions are incorporated as inputs of the value function.

6. REFERENCES

[1] G. Paschos, E. Bastug, I. Land, G. Caire, and M. Debbah, "Wireless caching: technical misconceptions and business barriers," *IEEE Communications Magazine*, vol. 54, no. 8, pp. 16–22, Aug. 2016.

[2] P. Blasco and D. Gündüz, "Learning-based optimization of cache content in a small cell base station," in *Intl. Conf. on Communications*, Sydney, Australia, June 2014, pp. 1897–1903.

[3] A. Sengupta, S. Amuru, R. Tandon, R. M. Buehrer, and T. C. Clancy, "Learning distributed caching strategies in small cell networks," in *Proc. Intl. Symp. on Wireless Communications Systems*, Barcelona, Spain, Aug. 2014, pp. 917–921.

[4] 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, Feb. 2017.

[5] S. Li, J. Xu, M. van der Schaar, and W. Li, "Trend-aware video caching through online learning," *IEEE Transactions on Multimedia*, vol. 18, no. 12, pp. 2503–2516, Dec. 2016.

[6] A. Sadeghi, F. Sheikholeslami, and G. B. Giannakis, "Optimal and scalable caching for 5G using reinforcement learning of space-time popularities," *arXiv preprint arXiv:1708.06698*, 2017.

[7] L. M. Lopez-Ramos, A. G. Marques, and J. Ramos, "Jointly optimal sensing and resource allocation for multiuser interweave cognitive radios," *IEEE Transactions on Wireless Communications*, vol. 13, no. 11, pp. 5954–5967, Nov. 2014.

[8] L. Georgiadis, M. J. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Foundations and Trends in Networking*, vol. 1, no. 1, pp. 1–144, 2006.

[9] T. Chen, A. G. Marques, and G. B. Giannakis, "DGLB: Distributed stochastic geographical load balancing over cloud networks," *IEEE Transactions on Parallel Distributed Systems*, vol. 28, no. 7, pp. 1866–1880, July 2017.

[10] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, MIT press Cambridge, 1998.

[11] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *Intl. Conf. on Computer Communications*, New York, USA, March 1999, pp. 126–134.