DISTRIBUTED NETWORK CACHING VIA DYNAMIC PROGRAMMING

Alireza Sadeghi[†], Antonio G. Marques[*], and Georgios 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

Next-generation communication networks are envisioned to extensively utilize storage-enabled caching units to alleviate unfavorable surges of data traffic by pro-actively storing anticipated highly popular contents across geographically distributed storage devices during off-peak periods. This resource pre-allocation is envisioned not only to improve network efficiency, but also to increase user satisfaction. In this context, the present paper designs optimal caching schemes for *distributed caching* scenarios. In particular, we look at networks where a central node (base station) communicates with a number of "regular" nodes (users or pico base stations) equipped with *local storage* infrastructure. Given the spatio-temporal dynamics of content popularities, and the decentralized nature of our setup, the problem boils down to select what, when and *where* to cache. To address this problem, we define fetching and caching prices that vary across contents, time and space, and formulate a global optimization problem which aggregates the costs across those three domains. The resultant optimization is solved using decomposition and dynamic programming techniques, and a reduced-complexity algorithm is finally proposed. Preliminary simulations illustrating the behavior of our algorithm are finally presented.

Index Terms— Caching, Fetching, Dynamic programming, Value iteration, Dynamic pricing.

1. INTRODUCTION

The tremendous growth of data traffic over wired and wireless networks motivates the need for emerging technologies to meet the ever-increasing data demands [1]. Recognized as an appealing solution is the so-called *caching*, which refers to storing reusable popular contents across geographically distributed storage-enabled network entities [1, 2]. The rationale here is to smoothen out the unfavorable shocks of peak traffic periods by pro-actively storing anticipated highly popular contents at these storage devices and during off-peak instances. This idea of resource pre-allocation is envisioned to achieve significant network resource savings in terms of e.g., energy, bandwidth, and cost. Which already comes with user satisfaction [1, 2, 3].

Utilizing (stochastic) optimization approaches to enable content-agnostic storage-enabled network entities to pro-actively cache popular contents has quickly gained attention. Under *static popularity* settings, a multi-armed bandit formulation is considered in [4]. The coded, and convexified version is studied in [5]. Context

and trend aware learning approaches are provided in [6, 7] and a transfer learning one in [8]. Belief propagation based distributed cooperative caching is investigated in [9], and an ADMM based distributed approach in [10]. Recently, by considering *dynamic popularities*, reinforcement learning approaches are utilized to cache contents [11, 12].

Existing caching approaches accounting for dynamic popularities focus on enabling a *single cache* entity to decide (learn) what and when to cache. However, since caching is being carried out in a network level, to fully unleash its potentials a distributed approach is essential to efficiently utilize the globally available storage capacity over the entire network. Building on this observation, the present paper proposes optimal dynamic schemes to operate over a network where nodes, connected to a central controller, receive user requests and are equipped with local cache memories where they can store specific contents. In particular, the optimal cache-fetch decisions across caching entities will be found as the solution of a constrained optimization aimed at reducing an expected network cost aggregated across caching entities, contents and time instants. Since the caching decision at a given node not only affects the instantaneous cost but also the availability of the content in the next slots, the problem is indeed a dynamic programming (DP) one. DP algorithms typically involve functional estimation and, as a result, they are computationally expensive. A key aspect of this paper is to combine a marginalization and a decomposition approach to effectively reduce the dimensionality of the problem and, hence, limit the computational complexity.

2. PROBLEM FORMULATION

Consider a distributed caching system composed of a central node (CN) connected to a number of regular nodes, all devised with local storage devices. The CN can be a base station in a cellular network, or a gateway router in a content delivery network. The nodes are then user devices or access points, correspondingly. We consider $m = 1 \cdots M$ nodes, $f = 1, 2 \cdots F$ files, and $t = 1, 2, \cdots$ time instants. For convenience the CN, which is also connected to a cloud through a (congested) back-haul, is indexed by $m = 0$. With this notation, at a given slot, every node receives user file requests as indicated by the binary variable $r_{t,m}^f$, where $r_{t,m}^f = 1$ denotes that during slot t file f was requested at node m, and $r_{t,m}^f = 0$ otherwise. Upon receiving a file request, the node forwards the request to the CN unless the file is already available in the cache of the node. If file f is available at the CN, the content is served to node m. If not, the CN fetches required file either from the cloud or from one of the local nodes. Moreover, once the content has been delivered, all the nodes having the file must decide whether to keep it or to discard it. Decisions are made based on the state of the system as well as on the

prices (costs) associated with caching and fetching, with the overall goal being to minimize the long-term aggregated cost.

2.1. Optimization variables and constraints

To formalize the setting, let us define the binary *fetching* variables $\underline{w}_{t,m}^f$, and $\overline{w}_{t,m}^f$, as well as the binary *storing* decision variable $a_{t,m}^f$. For the nodes, having $\underline{w}_{t,m}^f = 1$, implies fetching file f at slot t from node m to the CN, while $\underline{w}_{t,m}^f = 0$ means "not fetching". Similarly $\overline{w}_{t,m}^f$ indicates if the CN fetches the file f from node m at time t. To have a compact notation, fetching from the cloud to the CN is denoted as $w_{t,0}^f = \underline{w}_{t,0}^f = \overline{w}_{t,0}^f$. For all the nodes, the indicator $a_{t,m}^f = 1$ means that file f will be stored at node m at the end of slot t, and the binary storage state variable $s_{t,m}^f$ accounts for availability of file f at node m at the beginning of slot t. The availability of file f directly depends on the preceding caching decision, thus the first set of constraints is

$$\text{(C1)} \quad s_{t,m}^f = a_{t-1,m}^f, \tag{1}$$

which must hold $\forall f, t, m$. Having $r_{t,m}^f = 1$ and $s_{t,m}^f = 0$ necessitates requesting the file from the CN, which in turn requires fetching it either from the cloud or from the nodes having the content. This gives rise to the second set of constraints

$$\text{(C2)} \quad r_{t,0}^f \leq w_{t,0}^f + s_{t,0}^f + \sum_{m=1}^M \left(\underline{w}_{t,m}^f \cdot s_{t,m}^f \right). \tag{2}$$

The product $\underline{w}_{t,m}^f \cdot s_{t,m}^f$ guarantees that fetching from a node is feasible only if the node has the file. Storing a file at a node necessitates to either the file being available locally or fetching it from the CN, giving rise to the following constraint

$$\text{(C3)} \quad a_{t,m}^f \leq s_{t,m}^f + \overline{w}_{t,m}^f, \ \forall m = 1 \cdots M. \tag{3}$$

Likewise, storing at the CN requires

$$\text{(C4)} \quad a_{t,0}^f \leq s_{t,0}^f + w_{t,0}^f + \sum_{m=1}^M \underline{w}_{t,m}^f \cdot s_{t,m}^f \tag{4}$$

Finally, a node can fetch only if

$$\text{(C5)} \quad \overline{w}_{t,m}^f \leq s_{t,0}^f + w_{t,0}^f + \sum_{m' \neq m} \underline{w}_{t,m'}^f \cdot s_{t,m'}^f. \tag{5}$$

2.2. Prices and aggregated costs

As assumed earlier, both caching and fetching are costly. For node m, at time t and for file f, let $\rho_{t,m}^f$, $\underline{\lambda}_{t,m}^f$, and $\overline{\lambda}_{t,m}^f$ stand for costs of storing, fetching to, and fetching from the CN, respectively. Then, the overall incurred cost is $c_t^f = \sum_{m=0}^M c_{t,m}^f$, where for regular nodes we have that

$$c_{t,m}^f := \overline{\lambda}_{t,m}^f \overline{w}_{t,m}^f + \rho_{t,m}^f a_{t,m}^f, \ \forall m = 1 \cdots M, \tag{6}$$

while for $m = 0$ (the CN) we have

$$c_{t,0}^f := \lambda_{t,0} w_{t,0}^f + \rho_{t,0}^f a_{t,0}^f + \sum_{m=1}^M \underline{\lambda}_{t,m}^f \underline{w}_{t,m}^f, \tag{7}$$

with $\lambda_{t,0}$ being the cost of fetching from the cloud and $\rho_{t,0}$ being the caching cost for the CN.

2.3. Problem Formulation

Let us collect all caching and fetching costs during slot t for file f at vectors $\boldsymbol{\rho}_t^f$ and $\boldsymbol{\lambda}_t^f = \left[\underline{\boldsymbol{\lambda}}_t^f, \overline{\boldsymbol{\lambda}}_t^f \right]$, and define similarly the action vectors $\boldsymbol{a}_t^f$ and $\boldsymbol{w}_t^f = [\underline{\boldsymbol{w}}_t, \overline{\boldsymbol{w}}_t]^\top$ as well as the request vector $\boldsymbol{r}_t^f$. The first step is to define the aggregated cost to be minimized. Since there is an inherent uncertainty in the state of the system, we consider the *discounted* long-term incurred average cost

$$\bar{C} := \mathbb{E} \left[\sum_{t=0}^\infty \sum_{f=1}^F \gamma^t c_t^f \left(\boldsymbol{a}_t^f, \boldsymbol{w}_t^f; \boldsymbol{\rho}_t^f, \boldsymbol{\lambda}_t^f \right) \right] \tag{12}$$

where γ^t is an exponentially decaying weight which reduces the impact of distant (and therefore more uncertain) costs, and the expectation is taken with respect to the random variables $\{ \boldsymbol{r}_t^f, \boldsymbol{\rho}_t^f, \boldsymbol{\lambda}_t^f \}_{f,t}$. This paper assumes these variables are stationary, so that the expectations can practically be estimated. In addition the knowledge of the state variables $\{ \boldsymbol{r}_t^f, \boldsymbol{\rho}_t^f, \boldsymbol{\lambda}_t^f \}_{f,t}$ is causal, that is, their exact values are revealed at the beginning of every time slot t, and all the fetching and caching decisions are made sequentially. The goal is then to make *sequential* fetching-caching decisions that minimize the expected current plus future costs while satisfying the constraints (C1)-(C5). This gives rise to the following optimization problem

$$\text{(P1)} \quad \min_{\{(\boldsymbol{w}_k^f, \boldsymbol{a}_k^f)\}_{f,k \geq t}} \bar{C}_t := \sum_{k=t}^\infty \sum_{f=1}^F \gamma^{k-t} \mathbb{E} \left[c_k^f \left(\boldsymbol{a}_k^f, \boldsymbol{w}_k^f; \boldsymbol{\rho}_k^f, \boldsymbol{\lambda}_k^f \right) \right]$$
$$\text{s.t.} \quad \text{(C1)} - \text{(C5)},$$

Due to (C1), (P1) is indeed a DP, because current decisions will influence not only current, but also future costs. As a result, one needs to [13, p. 10]: (i) identify the current and the expected future aggregated cost (the latter gives rise to the so-called value function), (ii) write the Bellman optimality conditions; and (iii) suggest a method to estimate the value function. In addition, note that (P1) is indeed decomposable across files, since the fetching and caching prices are different for f, t, and m. In most practical setups, those prices are in fact (stochastic) Lagrange multipliers associated with constraints that have been handled using dual decomposition techniques (see, e.g., [14]). Definition of those prices for the investigated caching setup is certainly of interest, but out of the scope of this paper. For this reason, in the sequel the focus is on solving (P1) for a single file f. Hence, the superscript f will be dropped. In addition, due to stationarity the subscript t will be dropped occasionally as well.

3. DP-BASED OPTIMAL FETCHING AND CACHING

To solve (P1), collect all the storage state variables in vector $\boldsymbol{s}_t$, define $\boldsymbol{\theta}_t = [\boldsymbol{r}_t, \boldsymbol{\rho}_t, \boldsymbol{\lambda}_t]$, and assume that their initial values $\boldsymbol{s}_0$ and $\boldsymbol{\theta}_0$ are given. Then the optimal fetch-cache decisions $(\boldsymbol{w}_t^*, \boldsymbol{a}_t^*)$ are expressible as the solution to (8). 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 (P1) is a DP and the caching decisions $\boldsymbol{a}$ influence not only the current cost $c_t(\cdot)$ 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 $\{\boldsymbol{\theta}_t\}_t$, the term accounting for the future cost can be rewritten in terms of the *stationary value function*

$$(\boldsymbol{w}_t^*, \boldsymbol{a}_t^*) := \underset{(\boldsymbol{w}, \boldsymbol{a}) \text{ s.t. (C1)-(C5)}}{\arg\min} \left\{ \mathbb{E} \left[\underset{(\boldsymbol{w}_k, \boldsymbol{a}_k) \text{ s.t. (C1)-(C5)}}{\min} \left\{ \sum_{k=t}^{\infty} \gamma^{k-t} \left[c_k\left(\boldsymbol{a}_k, \boldsymbol{w}_k; \boldsymbol{\rho}_k, \boldsymbol{\lambda}_k\right) \Big| \boldsymbol{a}_t = \boldsymbol{a}, \boldsymbol{w}_t = \boldsymbol{w}, \boldsymbol{s}_t = \boldsymbol{s}_0, \boldsymbol{\theta}_t = \boldsymbol{\theta}_0 \right] \right\} \right] \right\} \tag{8}$$

$$= \underset{(\boldsymbol{w}, \boldsymbol{a}) \text{ s.t. (C1)-(C5)}}{\arg\min} \left\{ c_t\left(\boldsymbol{a}, \boldsymbol{w}; \boldsymbol{\rho}_t, \boldsymbol{\lambda}_t\right) + \underset{\boldsymbol{\theta}_k}{\mathbb{E}} \left[\underset{(\boldsymbol{w}_k, \boldsymbol{a}_k) \text{ s.t. (C1)-(C5)}}{\min} \sum_{k=t+1}^{\infty} \gamma^{k-t} \left[c_k\left(\boldsymbol{a}_k, \boldsymbol{w}_k; \boldsymbol{\rho}_k, \boldsymbol{\lambda}_k\right) \Big| \boldsymbol{s}_{t+1} = \boldsymbol{a} \right] \right] \right\} \tag{9}$$

$$V\left(\boldsymbol{s}, \boldsymbol{\theta}\right) := \underset{(\boldsymbol{w}, \boldsymbol{a}) \text{ s.t. (C1) - (C5)}}{\min} \left\{ \underset{\boldsymbol{\theta}_k}{\mathbb{E}} \left[\underset{(\boldsymbol{w}_k, \boldsymbol{a}_k) \text{ s.t. (C1) - (C5)}}{\min} \left\{ \sum_{k=t}^{\infty} \gamma^{k-t} \left[c_k\left(\boldsymbol{a}_k, \boldsymbol{w}_k; \boldsymbol{\rho}_k, \boldsymbol{\lambda}_k\right) \Big| \boldsymbol{a}_t = \boldsymbol{a}, \boldsymbol{w}_t = \boldsymbol{w}, \boldsymbol{s}_t = \boldsymbol{s}, \boldsymbol{\theta}_t = \boldsymbol{\theta} \right] \right\} \right] \right\} \tag{10}$$

$$\bar{V}(\boldsymbol{s}) := \underset{\boldsymbol{\theta}}{\mathbb{E}} \left[V\left(\boldsymbol{s}, \boldsymbol{\theta}\right) \right] = \underset{\boldsymbol{\theta}}{\mathbb{E}} \underset{(\boldsymbol{w}, \boldsymbol{a}) \text{ s.t. (C1) - (C5)}}{\min} \left\{ c_t(\boldsymbol{a}, \boldsymbol{w}; \boldsymbol{\rho}, \boldsymbol{\lambda}) + \gamma \bar{V}(\boldsymbol{a}) \right\} \tag{11}$$

$V\left(\boldsymbol{s}, \boldsymbol{\theta}\right)$ [13, p. 68]. This function, formally defined in (10), captures the minimum aggregated cost starting from "state" $(\boldsymbol{s}, \boldsymbol{\theta})$. Assuming that prices and requests are i.i.d. across time slots and nodes, it can be shown that the solution for (P1) can also be found in terms of the *reduced value function*, $\bar{V}(\boldsymbol{s}) := \underset{\boldsymbol{\theta}}{\mathbb{E}}[V(\boldsymbol{s}, \boldsymbol{\theta})]$, which depends only on $\boldsymbol{s}$ [15]. The reduced value function is important since not only enables solving (P1), but also alleviates the computationally burden of estimating $V(\boldsymbol{s}, \boldsymbol{\theta})$, which has a much larger dimensionality.

By rewriting the reduced value function recursively as the summation of the instantaneous cost and the aggregated future values, one readily arrives at the Bellman equations provided in (11). Thus, the problem at time t reduces to

$$\text{(P2)} \quad \underset{(\boldsymbol{w}, \boldsymbol{a})}{\min} \quad c_t(\boldsymbol{a}, \boldsymbol{w}; \boldsymbol{\rho}_t, \boldsymbol{\lambda}_t) + \gamma \bar{V}(\boldsymbol{a})$$

$$\text{s.t.} \quad \text{(C1)- (C5)},$$

which requires the estimation of the reduced value function $\bar{V}(\cdot)$. This can be done using the value iteration algorithm as tabulated in Alg. 1; for a detailed discussion cf. [13, p. 100]. Alternatively, if the distributions of the random parameters are *unknown*, stochastic online solvers based on Q-learning are well motivated [13, p. 148]. This is left as future work.

3.1. Discussing the solution to (P2)

To gain insights on the optimal fetching-caching solution, suppose that at time t, we have $s_{t,0} = 0$ and the CN receives file request from node m'. In such a case, $r_{t,0} = 1$ and thus (C2) requires $w_{t,0} + \sum_{m \in \mathcal{M}_t} \underline{w}_{t,m} \geq 1$, where $\mathcal{M}_t$ denotes the set of nodes for which $s_{t,m} = 1$. Since the reduced value function does not depend on the fetching decision $\boldsymbol{w}_t$ and the cost c_t is linear in $\underline{w}_{t,m}$ [cf. (6) and (7)], we have that the optimal fetching is

$$\underline{w}_{t,m}^* = \begin{cases} 1 & \text{if } m = \arg\min_{m'' \in \mathcal{M}_t \cup \{\text{Cloud}\}} \underline{\lambda}_{t,m''} \\ 0 & \text{for all other } m \end{cases}$$

Deciding on the caching variable $\boldsymbol{a}_t$ is more complicated. Suppose first that the value function can be written as $\bar{V}(\boldsymbol{s}) = \sum_{m=0}^{M} V_m(s_m)$ and that node m' was the one requesting the content. Then, we have that, the cost of setting $a_{t,m} = 1$ is

$$c_{\{a_{t,m}=1\}} = \begin{cases} \rho_{t,m} + \gamma \bar{V}_m(1) & \text{for } m \in \mathcal{M}_t^+ \\ \rho_{t,m} + \bar{\lambda}_{t,m} + \gamma \bar{V}_m(1) & \text{for all other } m \end{cases}$$

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

1 <u>Set</u> $\bar{V}(\boldsymbol{s}) = 0$, for $\forall \boldsymbol{s} \in \mathcal{S}$; $\mathcal{S} := \left\{ \boldsymbol{v} \big| \boldsymbol{v} \in \{0,1\}^{M+1} \right\}$

  Input : $\gamma < 1$, pdf for $\boldsymbol{\rho}, \boldsymbol{\lambda}$ and $\boldsymbol{r}$

  Output: $\bar{V}(\cdot)$

2 **while** $|\bar{V}^i(\boldsymbol{s}) - \bar{V}^{i+1}(\boldsymbol{s})| < \epsilon; \forall \boldsymbol{s} \in \mathcal{S}$ **do**

3   **for** $\forall \boldsymbol{s} \in \mathcal{S}$ **do**

4         $\bar{V}^{i+1}(\boldsymbol{s}) =$

        $\underset{(\boldsymbol{\omega}, \boldsymbol{\alpha}) \text{ s.t. (C1)-(C5)}}{\mathbb{E} \min} \left\{ c(\boldsymbol{\alpha}, \boldsymbol{\omega}; \boldsymbol{\rho}, \boldsymbol{\lambda}) + \gamma \bar{V}^i(\boldsymbol{\alpha}) \right\}$

5   **end**

6     $i = i + 1$

7 **end**

---

where $\mathcal{M}_t^+ = \mathcal{M}_t \cup \{0, m'\}$. On the other hand, the cost of setting $a_{t,m} = 0$ is just $c_{\{a_{t,m}=0\}} = \gamma \bar{V}_m(0)$ for all m. Hence, to obtain the optimal $\boldsymbol{a}_t$ one only needs to solve $(M+1)$ separate tests of the form $c_{\{a_{t,m}=1\}} \gtrless c_{\{a_{t,m}=0\}}$, one per node. However, in most cases the assumption of $\bar{V}(\boldsymbol{s}) = \sum_{m=0}^{M} \bar{V}_m(s_m)$ does not hold and, as a result, caching decisions are coupled across nodes. In addition, while for a small number of users (say 10 or less) both the estimation of $\bar{V}(\boldsymbol{s})$ and the computation of the multiple decision rule is affordable, the computational burden grows exponentially as M increases. As a result, in networks with a large number of users, alternative schemes that try to impose additional structure on $\bar{V}(\boldsymbol{s})$ are well motivated.

4. NUMERICAL TESTS

Here we run numerical simulations to assess the behavior of the developed caching policy. Due to space limitations, the focus will be on the behavior of the CN, but the analysis can be generalized to nodes with $m \geq 1$ as well. The caching cost ρ_0 is considered to be uniformly drawn from $[0, 2\bar{\rho}_0]$. In addition, to model the fetching costs over the entire network, we define the "effective" fetching cost λ_{eff} as the cost of fetching from any node or cloud to the CN. λ_{eff} is also considered to be uniformly drawn from $[0, 2\bar{\lambda}_{\text{eff}}]$. To assess the CN's policy, let us define the *caching ratio* as the ratio of the number of caching decisions divided by the total number of decisions. Fig 1a. depicts this ratio for different values of $\bar{\lambda}_{\text{eff}}$ and $\bar{\rho}_0$, for $s_0 = 1$, and $r_0 = 1$. The requests from the CN are assumed
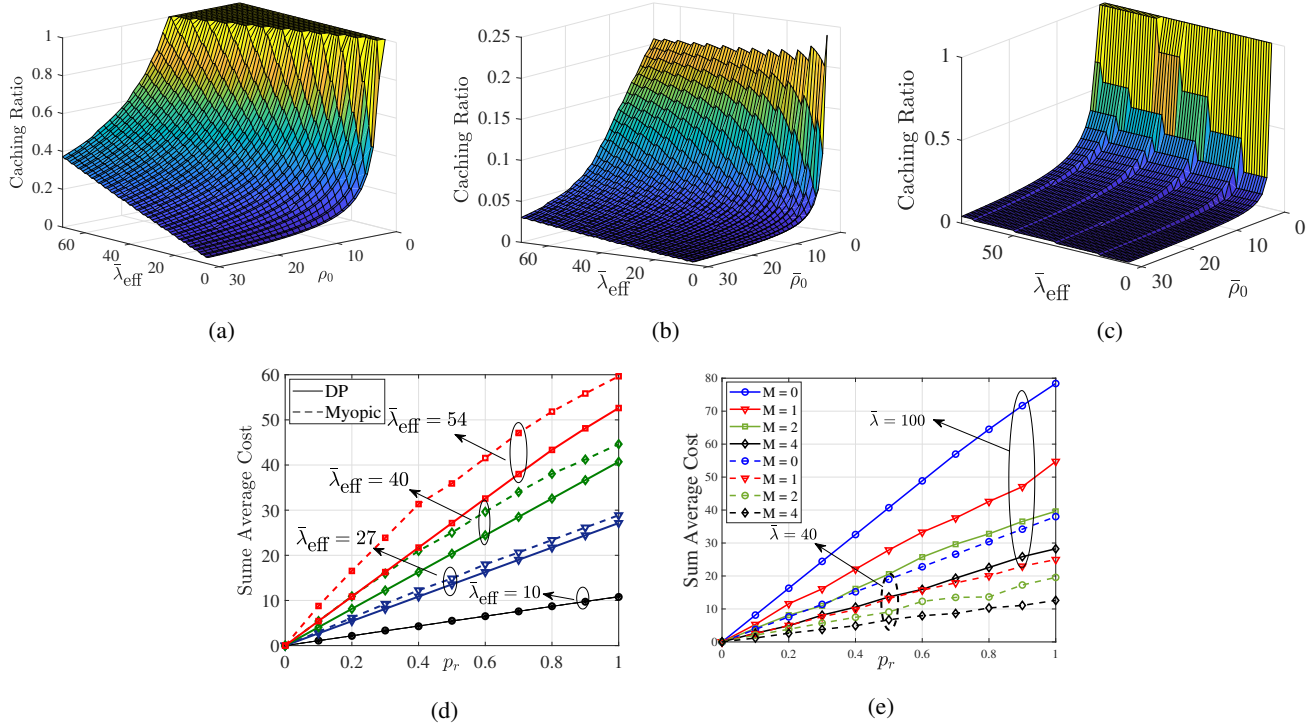
Fig. 1: Caching ratio versus $(\bar{\lambda}_{\text{eff}}, \bar{\rho}_0)$ for different initial states s_0, r_0, and request probability p_r; (a) $s_0 = 1$, $r_0 = 1$, and $p_r = 0.5$; (b) $s_0 = 0$, $r_0 = 0$, and $p_r = 0.5$; and (c) $s_0 = 0$, $r_0 = 0$, and $p_r = 0.05$. (d) Performance of DP versus myopic approach for different $\bar{\lambda}_{\text{eff}}$. Performance of of proposed algorithm for different number of regular nodes.

to follow a Bernoulli distribution with $p_r = 0.5$. The plot reveals a non-symmetric cache-versus-fetch trade off curve for the CN's policy, showing clearly how caching is preferred over fetching as the fetching cost increases. For small enough caching costs, the ratio goes to one (i.e., caching happens with probability 1). Changing the initial state to $s_0 = 0$, and $r_0 = 0$ clearly affects the CN's policy as depicted in Fig. 1b. Now, even when caching is cheap, caching decisions are rare (always below 20 %). Intuitively, this is due to the fact that the file has not already been requested ($r_0 = 0$) and, thus, caching is only made to take advantage of low fetching costs. Fig. 1c shows the caching ratio for $s_0 = 1$, $r_0 = 1$, and $p_r = 0.05$. The results confirm the relevance of the popularity of a content, since for a wide range of caching costs fetching is always decided. Interestingly, even when fetching is costly, the algorithm tends to fetch. The reason is that, due to its low popularity, the file will have to remain for a long time in the memory before being used, entailing a very high *aggregated* caching cost.

We now compare the performance of the proposed method with that of simpler schemes. Specifically, Fig. 1d reports the sum average cost of the proposed DP approach compared with that of a myopic one. The myopic policy stores a file only if the file is requested or is locally stored and the current caching cost is less than the fetching one, i.e., $\rho_{0,t} < \lambda_{\text{eff},t}$. Here, $\bar{\rho}_0 = 10$, and as this result demonstrates the DP approach outperforms the myopic one as the difference between average costs increases. This is expected because in the myopic approach the file will be stored whenever the fetching is costly; however, the DP-based approach will also consider the probability of the file being requested in future instants.

The next goal is to assess the benefits associated with distributed caching by comparing the obtained cost with the cost incurred when only the CN is equipped with a cache. To that end, Fig. 1e reports the results for different number of caching nodes $M \in \{0, 1, 2, 4\}$. The fetching costs over any link of the network are assumed to be i.i.d, and uniformly drawn from $[0 - 2\bar{\lambda}]$. The caching costs are also considered i.i.d and drawn from $[0 - 2\bar{\rho}]$. This plot depicts the sum average cost incurred by the CN for $\bar{\rho} = 62$, two values of $\bar{\lambda}$ (40 and 100), and different probabilities of requesting the files. Clearly, increasing M allows the CN to meet the demands and, hence, reduces the aggregated cost. As expected, this reduction saturates as M increases. All in all, the results corroborate the benefits of having a system equipped with distributed caching infrastructure.

5. CONCLUSION

This paper illustrated how decomposition and dynamic programming tools can be used to design optimal fetching-caching schemes for a network of caches. In particular, we focused on an architecture formed by a central controller connected to the cloud and to a set of local nodes. At each time instant the controller received different content requests and had to decide: i) if the content was going to be downloaded from the cloud or from the local nodes (provided that it was available locally), and ii) if some of the local nodes should cache the content in their local memory for future use. Upon defining suitable fetching and caching costs (that varied across time, nodes and contents), the problem was formulated as a dynamic program and its optimal solution was obtained using different strategies to reduce the associated computational burden.

6. REFERENCES

[1] G. S. Paschos, G. Iosifidis, M. Tao, D. Towsley, and G. Caire, "The role of caching in future communication systems and networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1111–1125, June 2018.

[2] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 82–89, Aug 2014.

[3] N. Golrezaei, A. F. Molisch, A. G. Dimakis, and G. Caire, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," *IEEE Commun. Mag.*, vol. 51, no. 4, pp. 142–149, April 2013.

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

[5] 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. Wireless Commun. Syst.*, Barcelona, Spain, Aug. 2014, pp. 917–921.

[6] S. Müller, O. Atan, M. van der Schaar, and A. Klein, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Tran. Wireless Commun.*, vol. 16, no. 2, pp. 1024–1036, Feb 2017.

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

[8] E. Bastug, M. Bennis, and M. Debbah, "A transfer learning approach for cache-enabled wireless networks," in *Intl. Symp. Modeling Opt. Mobile, Ad Hoc, Wireless Networks (WiOpt)*, Mumbai, India, May 2015, pp. 161–166.

[9] J. Liu, B. Bai, J. Zhang, and K. B. Letaief, "Cache placement in Fog-RANs: From centralized to distributed algorithms," *IEEE Trans. Wireless Commun.*, vol. 16, no. 11, pp. 7039–7051, Nov 2017.

[10] A. Abboud, E. Bastug, K. Hamidouche, and M. Debbah, "Distributed caching in 5G networks: An alternating direction method of multipliers approach," in *Proc. of SPAWC*, Stockholm, Sweden, June 2015, pp. 171–175.

[11] A. Sadeghi, F. Sheikholeslami, and G. B. Giannakis, "Optimal and scalable caching for 5G using reinforcement learning of space-time popularities," *IEEE J. Sel. Topics Signal Process.*, vol. 12, no. 1, pp. 180–190, Feb 2018.

[12] S. O. Somuyiwa, A. György, and D. Gündüz, "A reinforcement-learning approach to proactive caching in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1331–1344, June 2018.

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

[14] D. P. Palomar and Mung Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1439–1451, Aug 2006.

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