Reinforcement Learning for Adaptive Caching with Dynamic Storage Pricing

Alireza Sadeghi, Fatemeh Sheikholeslami, Antonio G. Marques, and Georgios B. Giannakis

Abstract—Small base stations (SBs) of fifth-generation (5G) cellular networks are envisioned to have storage devices to locally serve requests for reusable and popular contents by *caching* them at the edge of the network, close to the end users. The ultimate goal is to smartly utilize a limited storage capacity to serve locally contents that are frequently requested instead of fetching them from the cloud, contributing to a better overall network performance and service experience. To enable the SBs with efficient *fetch-cache* decision-making schemes operating in dynamic settings, this paper introduces 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 per time slot influence the content availability in future slots, the novel formulation for optimal fetch-cache decisions falls into the class of dynamic programming. Under this generic formulation, first by considering stationary distributions for the costs as well as file popularities, an efficient reinforcement learning-based solver known as value iteration algorithm can be used to solve the emerging optimization problem. Later, it is shown that practical limitations on cache capacity can be handled using a particular instance of this generic dynamic pricing formulation. Under this setting, to provide a light-weight online solver for the corresponding optimization, the well-known reinforcement learning algorithm, Q-learning, is employed to find optimal fetch-cache decisions. Numerical tests corroborating the merits of the proposed approach wrap up the paper.

Index Terms—Dynamic Caching, Fetching, Dynamic Programming, Value iteration, Q-learning.

I. INTRODUCTION

In the era of data deluge, storing "popular" contents at the edge of a content delivery network (CDN) or 5G cellular network, is a promising technique to satisfy the users' demand while alleviating the congestion on the back-haul links [1], [2], [3]. To this aim, small basestations (SBs) equipped with a limited local cache capacity, must intelligently store reusable popular contents on-the-fly, such that highly requested contents can be locally served at reduced cost. To endow SBs with the required learning capability, a wide range of learning and optimization approaches have been adopted (see [1], [2]).

Alireza Sadeghi, Fatemeh Sheikholeslami, and Georgios B. Giannakis are with the Digital Technology Center and the Department of Electrical and Computer engineering, University of Minnesota, Minneapolis, USA.
Emails: {sadeghi, sheik081, georgios}@umn.edu
Antonio G. Marques is with the Department of Signal Theory and Communications, King Juan Carlos University, Madrid, Spain.
Email: antonio.garcia.marques@urjc.es

Prior works: Time-invariant (*static*) content popularity caching has been previously investigated using a multi-armed bandit approach which accounted for the demand history and under unknown popularities [4]. Coded, convexified, and distributed extensions of this problem were later studied in [5], context and trend-aware learning approaches in [6], [7], and coordinated-distributed extensions in [8]. From a learning perspective, the trade-off between the "accuracy" of learning a static popularity, and the corresponding learning "speed" is investigated [9], [10].

Nevertheless, in reality, popularities exhibit fluctuations, meaning they are *dynamic* over a time horizon. For instance, half of the top 25 requested Wikipedia articles change on a daily basis [1], [11]. This motivates recent approaches to designing adaptive caching strategies for dynamic popularity scenarios [12], [13], [14] [15], [16], [17], with competitive regret bounds via online solvers [18], [19], [20], [21]. Furthermore, by considering Markovian evolution for popularities, reinforcement learning-based approaches were studied in [22], [23], [24]. In particular, global and local popularities were modeled by different Markov processes, and a tabular Q-learning based algorithm was proposed in [22]; while a policy gradient method was pursued to optimize a parametric policy in [23]. A network of caching units was considered in [24], where a deep neural network is used to approximate the so-called Q-function, and find the optimal caching policy.

To target different objectives such as content-access latency, energy, storage or bandwidth utilization, corresponding deterministic cost parameters are defined, and the aggregated cost is minimized in [12], [25]. Deterministic cost parameters, however, may be inaccurate in modeling practical settings, as spatio-temporal popularity evolutions, network resources such as bandwidth and cache capacity are *random* and subject to change over time and space, due to e.g., time-varying data traffic over links, previous cache decisions, or channel fluctuations. Therefore, this necessitates modeling the caching problem from a *stochastic optimization* perspective, while accounting for the inherently random nature of available resources and file requests.

Contributions: This work aspires to fill this gap by relying on dual decomposition techniques which transform the limits on the available resources in the original (primal) optimization into stochastic prices in the dual problem. Building on this approach, the goal is to design more flexible caching schemes by introducing a generic dynamic pricing formulation, while enabling SBs to learn the optimal fetching-caching decisions using low-complexity techniques. Our contributions are listed as follows.

1) A general formulation of the caching problem by introducing time-varying and stochastic costs is presented, in which the fetching and caching decisions are found through a constrained optimization with the objective of reducing the overall cost, aggregated across files and time instants (Section II).

2) Since the caching decision in a given time slot not only affects the instantaneous cost, but also influences the cache availability in the future, the problem is indeed a dynamic programming (DP), and therefore can be effectively solved by reinforcement learning-based approaches. By assuming known and stationary distributions for the costs and popularities, and upon relaxing the limited cache capacity constraint, the proposed generic optimization problem is shown to become separable across files, and thus can be efficiently solved using the value-iteration algorithm (Section III).

3) Subsequently, it is shown that the particular case where the cache capacity is limited and the distribution of the pertinent parameters are unknown can be handled by the proposed generic formulation. Thus, in order to address these issues, a dual-decomposition technique is developed to cope with the coupling constraint associated with the storage limitation. Finally, an online low complexity (Q-function based) reinforcement learning solver is put forth for learning the optimal fetch-cache decisions on-the-fly (Section IV).

4) The separability of the objective across files together with the use of marginalized value functions [26] enable the decomposition of the original problem into smaller-dimension sub-problems. This in turn leads to circumventing the so-called curse of dimensionality, which commonly arises in reinforcement learning problems (Sections III and IV).

The effectiveness of the proposed scheme in terms of efficiency as well as scalability is corroborated by various numerical tests. Although our proposed approach enjoys theoretical guarantees in learning the optimal fetch-cache decisions in stationary settings, numerical tests also corroborate its merits in non-stationary scenarios.

The rest of this paper is organized as follows. Section II provides a generic formulation of the problem, where solvers adopted from reinforcement learning are developed in Section III. Limited storage and back-haul transmission rate settings are discussed in Section IV. Section V reports numerical results, and finally section VI provides concluding remarks.

II. Operating conditions and costs

Consider a memory-enabled SB responsible for serving file (content) requests denoted by $f = 1, 2, \ldots, F$ across time. The requested contents are transmitted to users either by fetching through a (costly) back-haul transmission link connecting the SB to the cloud, or, by utilizing the local storage unit in the SB where popular contents have been proactively cached 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 SB 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 SB will eliminate such expenses. However, local caching also incurs a number of (instantaneous) costs corresponding to memory or energy consumption. 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 yet 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 at hand, and user-specific requirements. The variables, constraints, and costs involved in this optimization are described in the ensuing subsections.

A. Variables and constraints

Consider the system at time slot t, where the binary 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 requests is not allowed; thus, requests must be carried out either by fetching the file from the cloud or by utilizing the content currently available in the cache. Furthermore, at the end of each slot, the SB will decide if content f should be stored in the cache for its possible reuse in a subsequent slot.

To formalize this, let us define the "fetching" *decision* variable $w_t^f \in \{0, 1\}$ along the "caching" *decision* variable $a_t^f \in \{0, 1\}$. Setting $w_t^f = 1$ 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 slot t for the next slot, 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 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 the caching decision at time $t - 1$, we have

$$\text{C1:} \quad s_t^f = a_{t-1}^f, \quad \forall f, t, \tag{1}$$

which will be incorporated into our optimization as constraints.

Moreover, since having $r_t^f = 1$ implies 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

$$\text{C2:} \quad r_t^f \leq w_t^f + s_t^f, \quad \forall f, t. \tag{2}$$

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

$$\text{C3:} \quad a_t^f \leq s_t^f + w_t^f, \quad \forall f, t. \tag{3}$$

B. 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

$$\rho_t^f = \sigma_f(\rho'_t + \rho'^f_t) + (\rho''_t + \rho''^f_t), \qquad (4)$$

where the first term is proportional to the file size σ_f, while the second one is constant. Note also that we consider file-dependent costs (via variables ρ'^f_t and ρ''^f_t), as well as cost contributions which are common across files (via ρ'_t and ρ''_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 $\rho'^f_t = \rho''^f_t = 0$. 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 (see, e.g., [26], [27], [28], [29] and Section IV). Accordingly, the corresponding form for the fetching cost is

$$\lambda_t^f = \sigma_f(\lambda'_t + \lambda'^f_t) + (\lambda''_t + \lambda''^f_t). \qquad (5)$$

As before, if the transmission link from the cloud to the SB is the same for all contents, the prices λ'_t and λ''_t are expected to dominate their file-dependent counterparts λ'^f_t and λ''^f_t.

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

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

which is the basis for the DP formulated in the next section. For future reference, Fig. 1 shows a schematic of the system model and the notation introduced in this section.

III. OPTIMAL CACHING WITH TIME-VARYING COSTS

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

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

where the expectation is taken with respect to (w.r.t.) the random variables $\boldsymbol{\theta}_t^f := \{r_t^f, \lambda_t^f, \rho_t^f\}$, and $0 < \gamma < 1$ is the discounting factor whose tuning trades off current versus more uncertain future costs [30, p.44]. To address the optimization, the following assumptions are considered:

AS1) The values of $\boldsymbol{\theta}_t^f$ come from a stationary distribution.
AS2) The distribution of $\boldsymbol{\theta}_t^f$ is known.
AS3) The drawn value of $\boldsymbol{\theta}_t^f$ is revealed at the beginning of each slot t, before fetch-cache decisions are made.
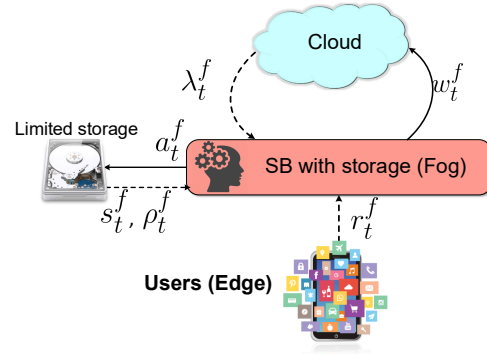

Fig. 1: System model and main notation. The state variables (dashed lines) are the storage indicator s_t^f and the content request r_t^f, as well as the dynamic caching and fetching prices ρ_t^f and λ_t^f. The optimization variables (solid lines) are the caching and fetching decisions a_t^f and w_t^f. The instantaneous per-file cost is $c_t^f = \rho_t^f a_t^f + \lambda_t^f w_t^f$. Per slot t, the SB collects the state variables $\{s_t^f, r_t^f; \rho_t^f, \lambda_t^f\}_{f=1}^F$, and decides the values of $\{a_t^f, w_t^f\}_{f=1}^F$ considering not only the cost at time t but also the cost at time instants $t' > t$.

AS1 and AS2 allow finding the expectations in this section, and will be relaxed in Section III-E to further generalize our approach to settings where the distributions are unknown. In practice, one may estimate these distributions through e.g., historical data.

The ultimate goal here is to take *real-time* fetch-cache decisions by minimizing the expected *current plus future cost* while adhering to operational constraints, giving rise to the following optimization

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

$$\text{s.t.} \quad (w_k^f, a_k^f) \in \mathcal{X}(r_k^f, a_{k-1}^f), \quad \forall f, \ k \geq t$$

where

$$\mathcal{X}(r_k^f, a_{k-1}^f) := \Big\{(w, a) \ \Big| \ w \in \{0, 1\}, \ a \in \{0, 1\},$$
$$s_k^f = a_{k-1}^f, \ r_k^f \leq w + s_k^f, \ a \leq s_k^f + w\Big\},$$

and the expectation is taken w.r.t. $\{\boldsymbol{\theta}_k^f\}_{\forall k \geq t+1}$.

The presence of the set $\mathcal{X}(r_k^f, a_{k-1}^f)$ in the constraints demonstrates that the cache state at a given time depends on previous cache decisions, thus coupling the optimization variables across time. It also implies that any instantaneous decision will influence the optimization problem in subsequent slots, having a long-standing influence on future costs. The coupling of the optimization variables across time indeed necessitates utilization of DP tools, motivating the implementation to reinforcement learning algorithms to design efficient solvers [30, p. 79].

To find the solution of the DP in (P1) we implement the following steps: a) identifying the current and expected future aggregate costs (the latter gives rise to the so-called value functions); b) expressing the corresponding Bellman

$$\left(w_t^{f*}, a_t^{f*}\right) := \underset{(w,a)\in\mathcal{X}(r_t^f, a_{t-1}^f)}{\arg\min} \left\{ \mathbb{E}_{\boldsymbol{\theta}_k^f}\left[\underset{(w_k,a_k)\in\mathcal{X}(r_k^f, a_{k-1}^f)}{\min} \left\{ \sum_{k=t}^{\infty} \gamma^{k-t}\left[c_k^f(a_k^f, w_k^f; \rho_k^f, \lambda_k^f)\big| a_t^f=a, w_t^f=w, \boldsymbol{\theta}_t^f=\boldsymbol{\theta}_0^f\right] \right\} \right] \right\} \quad (8)$$

$$= \underset{(w,a)\in\mathcal{X}(r_t^f, a_{t-1}^f)}{\arg\min} \left\{ c_t^f(a, w; \rho_t^f, \lambda_t^f) + \mathbb{E}_{\boldsymbol{\theta}_k^f}\left[\underset{(w_k,a_k)\in\mathcal{X}(r_k^f, a_{k-1}^f)}{\min} \sum_{k=t+1}^{\infty} \gamma^{k-t}\left[c_k^f(a_k^f, w_k^f; \rho_k^f, \lambda_k^f)\big| s_{t+1}^f=a \right] \right] \right\} \quad (9)$$

$$V^f\left(s^f, r^f; \rho^f, \lambda^f\right) := \underset{(w,a)\in\mathcal{X}(r_t^f, a_{t-1}^f)}{\min} \left\{ \mathbb{E}_{\boldsymbol{\theta}_k^f}\left[\underset{(w_k,a_k)\in\mathcal{X}(r_k^f, a_{k-1}^f)}{\min} \left\{ \sum_{k=t}^{\infty} \gamma^{k-t}\left[c_k^f(a_k^f, w_k^f; \rho_k^f, \lambda_k^f)\big| a_t^f=a, w_t^f=w, \boldsymbol{\theta}_t^f=\boldsymbol{\theta}^f\right] \right\} \right] \right\}$$
$$(10)$$

$$\bar{V}^f(s^f) := \mathbb{E}_{\boldsymbol{\theta}^f}\left[\underset{(w,a)\in\mathcal{X}(r_t^f, a_{t-1}^f)}{\min} \left\{ \mathbb{E}_{\boldsymbol{\theta}_k^f}\left[\underset{(w_k,a_k)\in\mathcal{X}(r_k^f, a_{k-1}^f)}{\min} \left\{ \sum_{k=t}^{\infty} \gamma^{k-t}\left[c_k^f(a_k^f, w_k^f; \rho_k^f, \lambda_k^f)\big| a_t^f=a, w_t^f=w, \boldsymbol{\theta}_t^f=\boldsymbol{\theta}^f\right] \right\} \right] \right\} \right]$$
$$= \mathbb{E}_{\boldsymbol{\theta}^f} \underset{(w,a)\in\mathcal{X}(r^f, s^f)}{\min} \left\{ c_0^f(a, w; \rho^f, \lambda^f) + \gamma\bar{V}^f(a) \right\} \quad (11)$$

equations over the value functions; and c) proposing a method to estimate the value functions accordingly. This is the subject of the ensuing subsections, which start by further exploiting the structure of our problem to reduce the complexity of the proposed solution.

A. Bellman equations for the per-content problem

Focusing on (P1), one can readily deduce 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 for a given f, do not affect the values (distribution) of $\boldsymbol{\theta}_{k'}^{f'}$ for files $f' \neq f$ and for times $t' > t$. Thus, (P1) naturally gives rise to the per-file optimization

$$(\text{P2}) \quad \underset{\{(w_k^f, a_k^f)\}_{k\geq t}}{\min} \quad \bar{\mathcal{C}}_t^f := \sum_{k=t}^{\infty} \gamma^{k-t}\mathbb{E}\left[c_k^f\left(a_k^f, w_k^f; \rho_k^f, \lambda_k^f\right)\right]$$
$$\text{s.t.} \quad (w_k^f, a_k^f) \in \mathcal{X}(r_k^f, a_{k-1}^f), \quad k \geq t$$

which must be solved for $f = 1, ..., F$. Indeed, the aggregate cost associated with (P2) will not depend on variables corresponding to files $f' \neq f$ [26]. This is the case if, for instance, the involved variables are independent of each other (which is the setup considered 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: The DP in (P2) can be solved with the corresponding Bellman equations, which require finding the associated value functions [30, p. 68]. To this end, consider the system at time t, where the cache state as well as the file requests and cost parameters are all given, so that we can write $s_t^f = s_0^f$ and $\boldsymbol{\theta}_t^f = \boldsymbol{\theta}_0^f$. Then, the optimal fetch-cache decision (w_t^{f*}, a_t^{f*}) is readily expressible as the solution to (8). The objective in (8) is 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, ρ_t^f and λ_t^f, the term accounting for the future cost can be rewritten in terms of the *stationary value function* $V^f\left(s^f, r^f; \rho^f, \lambda^f\right)$ [30, 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 $\boldsymbol{\theta}^f := [r^f, \rho^f, \lambda^f]$.

B. Marginalized 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* [26]

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

where the expectation is w.r.t $\boldsymbol{\theta}^f$. Marginalization of the value function 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 is in contrast with the original four-dimensional value function in (10), whose estimation is more difficult due to its continuous arguments. 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 equation form 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

$$(\text{P3}) \quad \underset{(w,a)}{\min} \quad c_t^f(a, w; \rho_t^f, \lambda_t^f) + \gamma\bar{V}^f(a)$$
$$\text{s.t.} \quad (w, a) \in \mathcal{X}(r_t^f, a_{t-1}^f).$$

$$\bar{V}_1 = (1-p)\Big(\mathbb{E}\min_{a\in\{0,1\}}\Big[\gamma\bar{V}_0(1-a)+(\rho+\gamma\bar{V}_1)a\ \Big|s=1,r=0\Big]\Big)+p\Big(\mathbb{E}\min_{a\in\{0,1\}}\Big[\gamma\bar{V}_0(1-a)+(\rho+\gamma\bar{V}_1)a\ \Big|s=1,r=1\Big]\Big)$$

$$= \gamma\bar{V}_0\Pr\left(\rho\geq\Delta\bar{V}_\gamma\right)+\mathbb{E}\left(\rho+\gamma\bar{V}_1\Big|\rho<\Delta\bar{V}_\gamma\right)\Pr\left(\rho<\Delta\bar{V}_\gamma\right) \tag{14}$$

$$\bar{V}_0 = (1-p)\Big(\mathbb{E}\min_{a\in\{0,1\}}\Big[\gamma\bar{V}_0(1-a)+(\lambda+\rho+\gamma\bar{V}_1)a\ \Big|s=0,r=0\Big]\Big) \tag{15}$$

$$+ \ p\Big(\mathbb{E}\min_{a\in\{0,1\}}\Big[(\lambda+\gamma\bar{V}_0)(1-a)+(\lambda+\rho+\gamma\bar{V}_1)a\ \Big|s=0,r=1\Big]\Big)$$

$$= (1-p)\Big(\gamma\bar{V}_0\Pr\left(\lambda+\rho\geq\Delta\bar{V}_\gamma\right)+\mathbb{E}\left(\lambda+\rho+\gamma\bar{V}_1\ \Big|\lambda+\rho<\Delta\bar{V}_\gamma\right)\Pr\left(\lambda+\rho<\Delta\bar{V}_\gamma\right)\Big) \tag{16}$$

$$+ \ p\Big(\mathbb{E}[\lambda]+\gamma\bar{V}_0\Pr\left(\rho\geq\Delta\bar{V}_\gamma\right)+\mathbb{E}\left(\rho+\gamma\bar{V}_1\ \Big|\rho\leq\Delta\bar{V}_\gamma\right)\Pr\left(\rho\leq\Delta\bar{V}_\gamma\right)\Big)$$

If the value-function is known, so that we have access to $\bar{V}^f(0)$ and $\bar{V}^f(1)$, the corresponding optimal (Bellman) decisions can be found as

$$w_t^f = a_t^f,\ a_t^f = \mathbb{I}_{\{\Delta\bar{V}_\gamma^f\geq\lambda_t^f+\rho_t^f\}} \quad \text{if } (r_t^f,s_t^f)=(0,0) \tag{13a}$$

$$w_t^f = 0,\ a_t^f = \mathbb{I}_{\{\Delta\bar{V}_\gamma^f\geq\rho_t^f\}} \quad \text{if } (r_t^f,s_t^f)=(0,1) \tag{13b}$$

$$w_t^f = 1,\ a_t^f = \mathbb{I}_{\{\Delta\bar{V}_\gamma^f\geq\rho_t^f\}} \quad \text{if } (r_t^f,s_t^f)=(1,0) \tag{13c}$$

$$w_t^f = 0,\ a_t^f = \mathbb{I}_{\{\Delta\bar{V}_\gamma^f\geq\rho_t^f\}} \quad \text{if } (r_t^f,s_t^f)=(1,1) \tag{13d}$$

where $\Delta\bar{V}_\gamma^f$ represents the *future* marginal cost, which is obtained as $\Delta\bar{V}_\gamma^f = \gamma(\bar{V}^f(1)-\bar{V}^f(0))$, and $\mathbb{I}_{\{\cdot\}}$ is an indicator function that yields value one if the condition in the argument holds, and zero otherwise.

The next subsection discusses how $\bar{V}^f(0)$ and $\bar{V}^f(1)$ can be calculated, but first a remark is in order.

Remark 1 (Augmented value functions). The value function $\bar{V}^f(s^f)$ can be redefined to account for extra information on r_t^f, ρ_t^f or λ_t^f, if available. For instance, consider the case where the distribution of r_t^f can be parametrized by p^f, which measures content "popularity" [31]. In such cases, the value function can incorporate the popularity parameter as an additional input to yield $\bar{V}^f(s^f,p^f)$. Consequently, the optimal decisions will depend not only on the current requests and prices, but also on the (current) popularity p^f. This indeed broadens the scope of the proposed approach, as certain types of *non-stationarity* in the distribution of r_t^f can be handled by allowing p^f to (slowly) vary with time.

C. Value function in closed form

For notational brevity, we have removed the superscript f in this subsection, and use $\bar{V}_0$ and $\bar{V}_1$ in lieu of $\bar{V}(0)$, and $\bar{V}(1)$. Denoting the *long-term* popularity of the content as $p := \mathbb{E}[r_t]$, using the expressions for the optimal actions in (13a)-(13d), and leveraging the independence among r_t, λ_t, and ρ_t, the expected cost-to-go function can be readily derived as in (14)-(16). The expectation in (14) is w.r.t. ρ, while that in (15) is w.r.t. both λ and ρ.

Solving the system of equations in (14)-(16) yields the optimal values for $\bar{V}_1$ and $\bar{V}_0$. A simple solver would be to perform exhaustive search over the range of these values since

---

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

---

1 <u>Set</u> $\bar{V}_0^0 = \bar{V}_1^0 = 0$;
  Input : $\gamma < 1$, probability density function of ρ, λ and r, precision ϵ, in order to stop
  Output: $\bar{V}_0$, $\bar{V}_1$
2 **while** $|\bar{V}_s^i - \bar{V}_s^{i+1}| < \epsilon; s \in \{0,1\}$ **do**
3   **for** $s = 0,1$ **do**
4       $\bar{V}_s^{i+1} = \mathbb{E}_{r,\rho,\lambda}\min_{(w,a)\in\mathcal{X}(r,s)}\left\{c(a,w;\rho,\lambda)+\gamma\bar{V}_a^i\right\}$
5   **end**
6     $i = i+1$
7 **end**

---

it is only a two-dimensional search space. However, a better alternative to solving the given system of equations is to rely on the well known *value iteration* algorithm [30, p. 100]. In short, this is an offline algorithm, which per iteration i updates the estimates $\{\bar{V}_0^{i+1}, \bar{V}_1^{i+1}\}$ by computing the expected cost using $\{\bar{V}_0^i, \bar{V}_1^i\}$, until the desired accuracy is achieved. This scheme is tabulated in detail in Algorithm 1, for which the distributions of r, ρ, λ are assumed to be known. We refer to [30, p.100] for a detailed discussion on the value-iteration algorithm, and its convergence guarantees.

Remark 2 (Finite-horizon approximate policies). In the proposed algorithms, namely exhaustive search as well as Algorithm 1, the solver is required to compute an expectation, which can be burdensome in setups with limited computational resources. For such scenarios, the class of finite-horizon policies emerges as a computationally affordable suboptimal alternative [30, p. 242]. The idea behind such policies is to truncate the infinite summation in the objective of (P1); thus, only considering the impact of the current decision on a few number of future time instants denoted by h, typically referred to as the *horizon*. The extreme case of a finite-horizon policy is that of a *myopic policy* with $h = 0$, which ignores any future impact of current decision, a.k.a. zero-horizon policy, thus taking the action which minimizes the instantaneous cost. This is equivalent to setting the future marginal cost to zero, hence solving (13a)-(13d) with $\Delta\bar{V}_\gamma = \Delta\bar{V}_\gamma^{h=0} = 0$. Another commonly used alternative is to consider the impact

$$Q\left(s_t, r_t, w_t, a_t; \rho_t, \lambda_t\right) := \mathbb{E}\left[\min_{\{(w_k,a_k)\in\mathcal{X}(r_k,a_{k-1})\}_{k=t+1}^{\infty}}\left\{\sum_{k=t}^{\infty}\gamma^{k-t}\left[c_k(a_k,w_k;\rho_k,\lambda_k)\Big|a_t, w_t, \boldsymbol{\theta}_t = \boldsymbol{\theta}\right]\right\}\right] \quad (19)$$

$$= \underbrace{c_t(a_t, w_t; \rho_t, \lambda_t)}_{\text{Immediate cost}} + \gamma\, \mathbb{E}\left[\underbrace{\min_{\{(w_k,a_k)\in\mathcal{X}(r_k,a_{k-1})\}_{k=t+1}^{\infty}}\left\{\sum_{k=t+1}^{\infty}\gamma^{k-(t+1)}\left[c_k(a_k,w_k;\rho_k,\lambda_k)\Big|s_{t+1} = a_t\right]\right\}}_{\text{Average minimum future cost}}\right] \quad (20)$$

$$\bar{Q}^{w_t,a_t}_{r_t,s_t} := \mathbb{E}_{\rho_t,\lambda_t}\left[Q\left(s_t, r_t, w_t, a_t; \rho_t, \lambda_t\right)\right], \quad \forall (w_t, a_t) \in \mathcal{X}(r_t, a_{t-1}) \quad (21)$$

$$= \mathbb{E}_{\rho_t,\lambda_t}\left[c_t(a_t, w_t; \rho_t, \lambda_t)\right] + \gamma\left[\mathbb{E}_{\boldsymbol{\theta}_{t+1}}\left[Q\left(s_{t+1}, r_{t+1}, w^*_{t+1}, a^*_{t+1}; \rho_{t+1}, \lambda_{t+1}\right)\Big|\boldsymbol{\theta}_{t+1}, s_{t+1} = a_t\right]\right].$$

$$\bar{Q}^{w,a}_{r,s} = \mathbb{E}[\lambda]w + \mathbb{E}[\rho]a + \gamma(1-p)\sum_{\forall(z_1,z_2)\in\mathcal{X}(0,a)}\bar{Q}^{z_1,z_2}_{0,a}\,\mathrm{Pr}\left((w^*_{t+1}, a^*_{t+1}) = (z_1, z_2)|(s_{t+1}, r_{t+1}) = (a, 0)\right)$$

$$+ \gamma p\sum_{\forall(z_1,z_2)\in\mathcal{X}(1,a)}\bar{Q}^{z_1,z_2}_{1,a}\,\mathrm{Pr}\left((w^*_{t+1}, a^*_{t+1}) = (z_1, z_2)|(s_{t+1}, r_{t+1}) = (a, 1)\right). \quad (23)$$

of the current decision for only the next time instant, which corresponds to the so-called horizon-1 policy. This entails setting the future cost at $h = 1$ as $\Delta \bar{V}^{h=1}_{\gamma} = \gamma(\bar{V}^{h=0}_1 - \bar{V}^{h=0}_0)$ with

$$\bar{V}^{h=0}_0 = (1-p)\mathbb{E}[\lambda w^{h=0} + \rho a^{h=0}|s=0, r=0]$$
$$+ p\mathbb{E}[\lambda w^{h=0} + \rho a^{h=0}|s=0, r=1] = p\mathbb{E}[\lambda] \quad (17)$$
$$\bar{V}^{h=0}_1 = (1-p)\mathbb{E}[\lambda w^{h=0} + \rho a^{h=0}|s=1, r=0]$$
$$+ p\mathbb{E}[\lambda w^{h=0} + \rho a^{h=0}|s=1, r=1] = 0, \quad (18)$$

which are then substituted into (13a)-(13d) to yield the actions $w^{h=1}$ and $a^{h=1}$. The notation $w^{h=0}$ and $a^{h=0}$ in (17) and (18) is used to denote the actions obtained when (13a)-(13d) are solved using the future marginal cost at horizon zero $\Delta \bar{V}^{h=0}_{\gamma}$, which as already mentioned, is zero; that is, under the myopic policy in lieu of the original optimal solution. Following an inductive argument, the future marginal cost at $h = 2$ is obtained as $\Delta \bar{V}^{h=2}_{\gamma} = \gamma(\bar{V}^{h=1}_1 - \bar{V}^{h=1}_0)$ with

$$\bar{V}^{h=1}_0 = (1-p)\mathbb{E}[\lambda w^{h=1} + \rho a^{h=1} + \gamma \bar{V}^{h=0}_a|s=0, r=0]$$
$$+ p\mathbb{E}[\lambda w^{h=1} + \rho a^{h=1} + \gamma \bar{V}^{h=0}_a|s=0, r=1],$$
$$\bar{V}^{h=1}_1 = (1-p)\mathbb{E}[\lambda w^{h=1} + \rho a^{h=1} + \gamma \bar{V}^{h=0}_a|s=1, r=0]$$
$$+ p\mathbb{E}[\lambda w^{h=1} + \rho a^{h=1} + \gamma \bar{V}^{h=0}_a|s=1, r=1],$$

which will allow to obtain the actions $w^{h=2}$ and $a^{h=2}$. While increasing horizons can be used, as h grows large, solving the associated equations becomes more difficult and computation of the optimal stationary policies, is preferable.

D. State-action value function (Q-function):

In many practical scenarios, knowing the underlying distributions for ρ_t, λ_t and r_t may not be possible, which motivates the introduction of online solvers that can learn the parameters on-the-fly. As clarified in the ensuing sections, in such scenarios, the so-called Q-function (or state-action value function) [30, p.69] becomes helpful, since there are rigorous theoretical guarantees on the convergence of its stochastic estimates; see [32] and [33]. Motivated by this fact, instead

of formulating our dynamic program using the value (cost-to-go) function, we can alternatively formulate it using the Q-function. Aiming at an online solver, let us tackle the DP through the estimation (learning) of the Q-function. Equation (19) defines the Q-function for a specific file under a given state (s_t, r_t), parametrized by cost parameters (ρ_t, λ_t). Under stationarity distribution assumption for $\{\rho_t, \lambda_t, r_t\}$, the Q-function $Q\left(s_t, r_t, w_t, a_t; \rho_t, \lambda_t\right)$ accounts for the minimum average aggregate cost at state (s_t, r_t), and taking specific fetch-cache decision (w_t, a_t) as for the first decision, while followed by the best possible decisions in next slots. This function is parametrized by (ρ_t, λ_t) since while making the current cache-fetch decision, the current values for these cost parameters are assumed to be known. The original Q-function in (19) needs to be learned over all values of $\{s_t, r_t, w_t, a_t, \rho_t, \lambda_t, r_t\}$, thus suffering from the curse of dimensionality, especially due to the fact that ρ_t and λ_t are continuous variables.

To alleviate this burden, we define the *marginalized Q-function* $Q(s_t, r_t, w_t, a_t)$ in (21). By changing the notation for clarity of exposition, the marginalized Q-function, $\bar{Q}^{w_t,a_t}_{r_t,s_t}$, can be rewritten in a more compact form as

$$\bar{Q}^{w_t,a_t}_{r_t,s_t} = \mathbb{E}\left[\lambda_t w_t + \rho_t a_t + \gamma \bar{Q}^{w^*_{t+1},a^*_{t+1}}_{r_{t+1},a_t}\right] \forall (w_t, a_t) \in \mathcal{X}(r_t, a_{t-1}). \quad (22)$$

Note that, while the marginalized value-function is only a function of the state, the marginalized Q-function depends on both the state (r, s) and the immediate action (w, a). The main reason one prefers to learn the value-function rather than the Q-function is that the latter is computationally more complex. To see this, note that the input space of $\bar{Q}^{w_t,a_t}_{r_t,s_t}$ is a four-dimensional binary space, hence the function has $2^4 = 16$ different inputs and one must estimate the corresponding 16 outputs. Each of these possible values are called Q-factors, and under the stationarity assumption, they can be found using (23) defined for all (r, s, w, a). In this expression, we have $(z_1, z_2) \in \{0, 1\}^2$ and the term $\mathrm{Pr}\left((w^*_{t+1}a^*_{t+1}) = (z_1, z_2)\right)$ stands for the probability of specific action (z_1, z_2) to be optimal at slot $t + 1$. This action is random because the

optimal decision at $t + 1$ depends on ρ_{t+1}, λ_{t+1} and r_{t+1}, which are not known at slot t. Although not critical for the discussion, if needed, one can show that half of the 16 Q-factors can be discarded, either for being infeasible – recall that $(w_t, a_t) \in \mathcal{X}(r_t, a_{t-1})$ – or suboptimal. This means that (23) needs to be computed only for 8 of the Q-factors.

From the point of view of offline estimation, working with the Q-function is more challenging than working with the V-function, since more parameters need to be estimated. In several realistic scenarios however, the distributions of the state variables are unknown, and one has to resort to stochastic schemes in order to learn the parameters on-the-fly. In such scenarios, the Q-function based approach is preferable, because it enables learning the optimal decisions in an online fashion even when the underlying distributions are unknown.

E. Stochastic policies: Reinforcement learning

As discussed in Section III-C, there are scenarios where obtaining the optimal value function (and, hence, the optimal stationary policy associated with it) is not computationally feasible. The closing remark in that section discussed policies which, upon replacing the optimal value function with approximations easier to compute, trade reduced complexity for loss in optimality. However, such reduced-complexity methods still require knowledge of the state distribution [cf. (17) and (18)]. In this section, we discuss stochastic schemes to approximate the value function under unknown distributions, thus relaxing assumption AS2 made earlier. The policies resulting from such stochastic methods offer a number of advantages since they: (a) incur a reduced complexity; (b) do not require knowledge of the underlying state distribution; (c) are able to handle some non-stationary environments; and in some cases, (d) they come with asymptotic optimality guarantees. To introduce this scheme, we first start by considering a simple method that updates stochastic estimates of the value function itself, and then proceed to a more advanced method which tracks the value of the Q-function. Specifically, the presented method is an instance of the celebrated Q-learning algorithm [34], which is the workhorse of stochastic approximation in DP [30, p. 68].

1) Stochastic value function estimates: The first method relies on current stochastic estimates of $\bar{V}_0$ and $\bar{V}_1$, denoted by $\hat{\bar{V}}_0(t)$ and $\hat{\bar{V}}_1(t)$ at time t (to be defined rigorously later). Given $\hat{\bar{V}}_0(t)$ and $\hat{\bar{V}}_1(t)$ at time t, the (stochastic) actions $\hat{w}_t$ and $\hat{a}_t$ are taken via solving (13a)-(13d) with $\Delta\bar{V}_\gamma = \gamma(\hat{\bar{V}}_0(t) - \hat{\bar{V}}_1(t))$. Then, stochastic estimates of the value functions $\hat{\bar{V}}_0(t)$ and $\hat{\bar{V}}_1(t)$ are updated as

- If $s_t = 0$, then $\hat{\bar{V}}_1(t + 1) = \hat{\bar{V}}_1(t)$ and $\hat{\bar{V}}_0(t + 1) = (1 - \beta_t)\hat{\bar{V}}_0(t) + \beta_t(\hat{w}_t\lambda_t + \hat{a}_t\rho_t + \gamma\hat{\bar{V}}_{\hat{a}_t}(t))$;
- If $s_t = 1$, then $\hat{\bar{V}}_0(t + 1) = \hat{\bar{V}}_0(t)$ and $\hat{\bar{V}}_1(t + 1) = (1 - \beta_t)\hat{\bar{V}}_1(t) + \beta_t(\hat{w}_t\lambda_t + \hat{a}_t\rho_t + \gamma\hat{\bar{V}}_{\hat{a}_t}(t))$;

where $\beta_t > 0$ denotes the stepsize. While easy to implement (only two recursions are required), this algorithm has no optimality guarantees.

2) Q-learning algorithm: Alternatively, one can run a stochastic approximation algorithm on the Q-function. This entails replacing the Q-factors $\bar{Q}_{r,s}^{w,a}$ with stochastic estimates

$\hat{\bar{Q}}_{r,s}^{w,a}(t)$. To describe the algorithm, suppose for now that at time t, the estimates $\hat{\bar{Q}}_{r,s}^{w,a}(t)$ are known for all (r, s, w, a). Then, in a given slot t with (r_t, s_t), action $(\hat{w}_t^*, \hat{a}_t^*)$ is obtained via either an exploration or an exploitation step. When exploring, which happens with a small probability ϵ_t, a random and feasible action $(\hat{w}_t^*, \hat{a}_t^*) \in \mathcal{X}(r_t, a_{t-1})$ is taken. In contrast, in the exploitation mode, which happens with a probability $1 - \epsilon_t$, the optimal action according to the current estimate of $\hat{\bar{Q}}_{r,s}^{w,a}(t)$ is

$$(\hat{w}_t^*, \hat{a}_t^*) := \underset{(w,a)\in\mathcal{X}(r_t,a_{t-1})}{\arg\min} \quad w\lambda_t + a\rho_t + \gamma\hat{\bar{Q}}_{r_t,s_t}^{w,a}(t). \quad (24)$$

After taking this action, going to next slot $t + 1$, and observing ρ_{t+1}, λ_{t+1}, and r_{t+1}, the Q-function estimate is updated as

$$\hat{\bar{Q}}_{r,s}^{w,a}(t + 1) =$$
$$\begin{cases} \hat{\bar{Q}}_{r,s}^{w,a}(t) & \text{if} \quad (r, s, w, a) \neq (r_t, s_t, \hat{w}_t^*, \hat{a}_t^*) \\ (1 - \beta_t)\hat{\bar{Q}}_{r_t,s_t}^{\hat{w}_t^*,\hat{a}_t^*}(t) + \beta_t\left(\hat{w}_t^*\lambda_t + \hat{a}_t^*\rho_t + \gamma\hat{\bar{Q}}_{r_{t+1},\hat{a}_t^*}^{\hat{w}_{t+1}^*,\hat{a}_{t+1}^*}(t)\right) & \text{o.w.,} \end{cases}$$
$$(25)$$

where "o.w." stands for "otherwise", $(\hat{w}_{t+1}^*, \hat{a}_{t+1}^*)$ is the optimal action for the next slot and, if needed, the stepsize β_t can be adapted for each particular state-action pair. This update rule describes one of the possible implementations of the Q-learning algorithm, which was originally introduced in [34]. This online algorithm enables making sequential decisions in an unknown environment, and is guaranteed to learn optimal decision-making rules under certain conditions specified next [30, p.148], [33].

Regarding convergence of the Q-learning algorithm, the following necessary conditions should hold [33], [35]: (c1) all feasible state (r, s) and action (w, a) pairs should be continuously updated; and, (c2) the learning rate β_t should be a diminishing step size. Under these conditions, the factors $\hat{\bar{Q}}_{r,s}^{w,a}$ converge to their optimal value $\bar{Q}_{r,s}^{*w,a}$ with probability 1; see [35] for details. To satisfy (c1), various exploration-exploitation algorithms have been proposed [36, p. 839]. Particularly, any such scheme needs to be *greedy in the limit of infinite exploration*, or GLIE [36, p. 840]. A common choice to meet this property is the ϵ-greedy approach, as considered in this work, with $\epsilon_t = 1/t$, which provides guaranteed yet slow convergence. In practice however, ϵ_t can be set to a small value for faster convergence [35], [32]. To satisfy the diminishing step size rule in (c2), let us define $t_{r,s}^{w,a}$ as the index of the t-th time when the state-action pair (r, s) and (w, a) is visited, and updated with the corresponding learning rate $\beta_{t_{r,s}^{w,a}}$. Condition (c2) requires $\sum_{t=1}^{\infty} \beta_{t_{r,s}^{w,a}} = \infty$, and $\sum_{t=1}^{\infty} \beta_{t_{r,s}^{w,a}}^2 < \infty$ to hold for all feasible state-action pairs, a typical choice for which is setting $\beta_{t_{r,s}^{w,a}} = 1/t$. Similar to ϵ_t, a constant but small learning rate is preferred in practice as it endows the algorithm to adapt to possible changes of pertinent parameters in dynamic settings.

The resultant algorithm for the problem at hand is tabulated in Algorithm 2. It is important to stress that in our particular case, we expect the algorithm to converge fast. That is the case because, under the decomposition approach followed in

---

Algorithm 2: Q-learning algorithm to estimate $\bar{Q}_{r,s}^{w,a}$ for a given file f

Input : $0 < \gamma, \beta_t < 1$
Output: $\hat{\bar{Q}}_{r,s}^{w,a}(t+1)$

1 **Initialize** $\hat{\bar{Q}}_{r,s}^{w,a}(1) = 0$, $s_1 = 0$, $\{r_0, \rho_0, \lambda_0\}$ are revealed
2 **for** $t = 1, 2, \ldots$ **do**
3   For the current state (r_t, s_t), choose $(\hat{w}_t^*, \hat{a}_t^*)$

$$(\hat{w}_t^*, \hat{a}_t^*) = \begin{cases} \text{Solve (23)} & \text{w.p. } 1 - \epsilon_t \\ \text{random } (w,a) \in \mathcal{X}_t(r_t, s_t) & \text{w.p. } \epsilon_t \end{cases}$$

4   Update state $s_{t+1} = \hat{a}_t^*$
5   Request and cost parameters, $\boldsymbol{\theta}_{t+1}$, are revealed
6   Update Q factor by (25)
7 **end**

---

this paper as well as the introduction of the marginalized Q-function, the state-action space of the resultant Q-function has very low dimension and hence, only a small number of Q-factors need to be estimated.

IV. LIMITED STORAGE AND BACK-HAUL TRANSMISSION RATE VIA DYNAMIC PRICING

So far, we have considered that the prices $\{\rho_t^f, \lambda_t^f\}$ are provided by the system, and we have not assumed any explicit limits (bounds) neither on the capacity of the local storage nor on the back-haul transmission link between the SB and the cloud. In this section, we discuss such limitations, and describe how by leveraging dual decomposition techniques, one can redefine the prices $\{\rho_t^f, \lambda_t^f\}$ to account for capacity constraints.

A. Limiting the instantaneous storage rate

In this subsection, practical limitations on the cache storage capacity are explored. Suppose that the SB is equipped with a *single* memory device that can store M files. Clearly, the cache decisions should then satisfy the following constraint per time slot

$$\text{C4:} \quad \sum_{f=1}^{F} a_t^f \sigma^f \leq M, \quad t = 1, 2, \ldots$$

In order to respect such hard capacity limits, the original optimization problem in (P1) can be simply augmented with C4, giving rise to a new optimization problem which we will refer to as (P4). Solving (P4) is more challenging than (P1), since the constraints in C4 must be enforced at each time instant, which subsequently couples the optimization across files. In order to deal with this, one can dualize C4 by augmenting the cost with the primal-dual term $\mu_t(\sum_{f=1}^{F} \sigma_f a_t^f - M)$, where μ_t denotes the Lagrange multiplier associated with the capacity constraint C4. The resultant problem is separable across files, but requires finding μ_t^*, the optimal value of the Lagrange multiplier, at each and every time instant.

If the solution to the original unconstrained problem (P1) does satisfy C4, then $\mu_t^* = 0$ due to complementary slackness.

On the other hand, if the storage limit is violated, then the constraint is active, the Lagrange multiplier satisfies $\mu_t^* > 0$, and its exact value must be found using an iterative algorithm. Once the value of the multiplier is known, the optimal actions associated with (P4) can be found using the expressions for the optimal solution to (P1) provided that the original storage price ρ_t^f is replaced with the new storage price $\rho_{t,aug}^f = \rho_t^f + \mu_t^* \sigma_f$ [cf. (4)]. The reason for this will be explained in detail in the following subsection, after introducing the ensemble counterpart of C4.

B. Limiting the long-term storage rate

Consider now the following constraint [cf. C4]

$$\text{C5:} \quad \sum_{k=t}^{\infty} \gamma^{k-t} \mathbb{E} \left[\sum_{f=1}^{F} a_k^f \sigma^f \right] \leq \sum_{k=t}^{\infty} \gamma^{k-t} M' \quad (26)$$

where the expectation is taken w.r.t. all state variables. By setting $M' = M$, one can view C5 as a relaxed version of C4. That is, while C4 enforces the limit to be respected at every time instant, C5 only requires it to be respected *on average*. From a computational perspective, dealing with C5 is easier than its instantaneous counterpart, since in the former only one constraint is enforced and, hence, only one Lagrange multiplier, denoted by μ, must be found. This comes at the price that guaranteeing C5 with $M' = M$ does not imply that C4 will always be satisfied. Alternatively, enforcing C5 with $M' < M$, will increase the probability of satisfying C4, since the solution will guarantee that "on average" there exists free space on the cache memory. A more formal discussion on this issue will be provided in the remark closing the subsection.

To describe in detail how accounting for C5 changes the optimal schemes, let (P5) be the problem obtained after augmenting (P1) with C5. Suppose now that to solve (P5) we dualize the single constraint in C5. Rearranging terms, the augmented objective associated with (P5) is given by

$$\sum_{k=t}^{\infty} \sum_{f=1}^{F} \gamma^{k-t} \mathbb{E} \left[c_k^f \left(a_k^f, w_k^f; \rho_k^f, \lambda_k^f \right) + \mu a_k^f \sigma^f \right] - \sum_{k=t}^{\infty} \gamma^{k-t} M'. \quad (27)$$

Equation (27) demonstrates that after dualization and provided that the multiplier μ is known, decisions can be optimized separately across files. To be more precise, note that the term $\sum_{k=t}^{\infty} \gamma^{k-t} M'$ in the objective is constant, so that it can be ignored, and define the modified instantaneous cost as

$$\check{c}_k^f := c_k^f \left(a_k^f, w_k^f; \rho_k^f, \lambda_k^f \right) + \mu \sigma^f a_k^f$$
$$= \left(\rho_k^f + \mu \sigma^f \right) a_k^f + \lambda_k^f w_k^f. \quad (28)$$

The last equation not only reflects that the dualization indeed facilitates separate per-file optimization, but it also reveals that term $\mu \sigma^f$ can be interpreted as an additional storage cost associated with the long-term caching constraint. More importantly, by defining the modified (augmented) prices $\rho_{t,\text{aug}}^f := \rho_t^f + \mu \sigma^f$ for all t and f, the optimization of (28) can be carried out with the schemes presented in the previous sections, provided that ρ_t^f is replaced with $\rho_{t,\text{aug}}^f$.

Note however that in order to run the optimal allocation algorithm, the value of μ needs to be known. Since the dual problem is always convex, one option is to use an iterative dual subgradient method, which computes the satisfaction/violation of the constraint C5 per iteration [37], [38, p.223]. Clearly, this requires knowledge of the state distribution, since the constraint involves an expectation. When such knowledge is not available, or when the computational complexity to carry out the expectations cannot be afforded, stochastic schemes are worth considering. For the particular case of estimating Lagrange multipliers associated with long-term constraints, a simple but powerful alternative is to resort to *stochastic dual subgradient schemes* [37], [38], which for the problem at hand, estimate the value of the multiplier μ at every time instant t using the update rule

$$\hat{\mu}_{t+1} = \left[\hat{\mu}_t + \zeta \left(\sum_{f=1}^{F} \hat{a}_t^{f*} \sigma^f - M' \right) \right]^+. \quad (29)$$

In the last expression, $\zeta > 0$ is a (small) positive constant, the update multiplied by ζ corresponds to the violation of the constraint after removing the expectation, the notation $[\cdot]^+$ stands for the $\max\{0, \cdot\}$, and $\hat{a}_t^{f*}$ denotes the optimal caching actions obtained with the policies described in Section III provided that ρ_t^f is replaced by $\hat{\rho}_{t,\text{aug}}^f = \rho_t^f + \hat{\mu}_t \sigma^f$.

We next introduce another long-term constraint that can be considered to limit the storage rate. This constraint is useful not only because it gives rise to alternative novel caching-fetching schemes, but also because it will allow us to establish connections with well-known algorithms in the area of congestion control and queue management. To start, define the variables $\alpha_{in,t}^f := [a_t^f - s_t^f]^+$ and $\alpha_{out,t}^f := [s_t^f - a_t^f]^+$ for all f and t. Clearly, if $\alpha_{in,t}^f = 1$, then content f that was not in the local cache at time $t-1$, has been stored at time t; and as a result, less storage space is available. On the other hand, if $\alpha_{out,t}^f = 1$, then content f was removed from the cache at time t, thus freeing up new storage space. With this notation at hand, we can consider the long term constraint

$$\text{C6:} \sum_{k=t}^{\infty} \gamma^{k-t} \mathbb{E} \left[\sum_{f=1}^{F} \alpha_{in,k}^f \sigma^f \right] \leq \sum_{k=t}^{\infty} \gamma^{k-t} \mathbb{E} \left[\sum_{f=1}^{F} \alpha_{out,k}^f \sigma^f \right],$$
$$(30)$$

which basically ensures the long-term stability of the local-storage. That is, the amount of data stored in the local memory is no larger than that taken out from the memory, guaranteeing that in the long term stored data does not grow unbounded.

To deal with C6 we can follow an approach similar to that of C5, under which we first dualize C6 and then use a stochastic dual method to estimate the associated dual variable. With a slight abuse of notation, supposing that the Lagrange multiplier associated with stability is by also denoted μ, the counterpart of (29) for the constraint C6 is

$$\hat{\mu}_{t+1} = \left[\hat{\mu}_t + \zeta \sum_{f=1}^{F} [\hat{a}_t^{f*} - s_t^f]^+ - [s_t^f - \hat{a}_t^{f*}]^+ \right]^+. \quad (31)$$

Note that the update term in the last iteration follows after removing the expectations in C6 and replacing $\alpha_{in,t}^f$, and

$\alpha_{out,t}^f$ with their corresponding definitions. The modifications that the expressions for the optimal policies require to account for this constraint are a bit more intricate. If $s_t^f = 0$, the problem structure is similar to that of the previous constraints, and we just need to replace ρ_t^f with $\hat{\rho}_{t,\text{aug}}^f = \rho_t^f + \hat{\mu}_t \sigma^f$. However, if $s_t^f = 1$, it turns out that: i) deciding $\hat{a}_t^{f*} = 1$ does not require modifying the caching price, but ii) deciding $\hat{a}_t^{f*} = 0$ requires considering the *negative* caching price $-\hat{\mu}_t \sigma^f$. In other words, while our formulation in Section III only considers incurring a cost when $a_t^f = 1$ (and assumes that the instantaneous cost is zero for $a_t^f = 0$), to fully account for C6, we would need to modify our original formulation so that costs can be associated with the decision $a_t^f = 0$ as well. This can be done either by considering a new cost term or, simply by replacing $\gamma \bar{V}^f(0)$ by $\gamma \bar{V}^f(0) - \hat{\mu}_t \sigma^f$ in (13a)-(13d), which are Bellman's equations describing the optimal policies.

Remark 3 (Role of the stochastic multipliers). It is well-established that the Lagrange multipliers can be interpreted as the marginal price that the system must pay to (over-)satisfy the constraint they are associated with [38, p.241]. When using stochastic methods for estimating the multipliers, further insights on the role of the multipliers can be obtained [27], [39], [28]. Consider for example the update in (29). The associated constraint C5 establishes that the long-term storage rate cannot exceed M'. To guarantee so, the stochastic scheme updates the estimated price in a way that, if the constraint for time t is oversatisfied, the price goes down, while if the constraint is violated, the price goes up. Intuitively, if the price estimate $\hat{\mu}_t$ is far from its optimal value and the constraint is violated for several consecutive time instants, the price will keep increasing, and eventually will take a value sufficiently high so that storage decisions are penalized/avoided. How quickly the system reacts to this violation can be controlled via the constant ζ. Interestingly, by tuning the values of M' and ζ, and assuming some regularity properties on the distribution of the state variables, conditions under which deterministic short-term limits as those in C4 are satisfied can be rigorously derived; see, e.g., [28] for a related problem in the context of distributed cloud networks. A similar analysis can be carried out for the update in (31) and its associated constraint C6. Every time the instantaneous version of the constraint is violated because the amount of data stored in the memory exceeds the amount exiting the memory, the corresponding price $\hat{\mu}_t$ increases, thus rendering future storage decisions more costly. In fact, if we initialize the multiplier at $\hat{\mu}_t = 0$ and set $\zeta = 1$, then the corresponding price is the total amount of information stored at time t in the local memory. In other words, the update in (31) exemplifies how the dynamic prices considered in this paper can be used to account for the actual state of the caching storage. Clearly, additional mappings from the instantaneous storage level to the instantaneous storage price can be considered. The connections between stochastic Lagrange multipliers and storing devices have been thoroughly explored in the context of demand response, queuing management and congestion control. We refer the interested readers to, e.g., [27], [39].

C. Limits on the back-haul transmission rate

The previous two subsections dealt with limited caching storage, and how some of those limitations could be accounted for by modifying the caching price ρ_t^f. This section addresses limitations on the back-haul transmission rate between the SB and the cloud as well as their impact on the fetching price λ_t^f.

While our focus has been on optimizing the decisions at the SB, contemporary networks must be designed following a holistic (cross-layer) approach that accounts for the impact of local decisions on the rest of the network. Decomposition techniques (including those presented in this paper) are essential to that end [37]. For the system at hand, suppose that $\mathbf{x}_{CD}$ includes all variables at the cloud network, $\bar{C}_{CD}(\mathbf{x}_{CD})$ denotes the associated cost, and the feasible set $\mathcal{X}_{CD}$ accounts for the constraints that cloud variables $\mathbf{x}_{CD}$ must satisfy. Similarly, let $\mathbf{x}_{SB}$, $\bar{C}_{SB}(\mathbf{x}_{SB})$, and $\mathcal{X}_{SB}$ denote the corresponding counterparts for the SB optimization analyzed in this paper. Clearly, the fetching actions w_t^f are included in $\mathbf{x}_{SB}$, while the variable b_t representing back-haul transmission rate (capacity) of the connecting link between the cloud and the SB, is included in $\mathbf{x}_{CD}$. This transmission rate will depend on the resources that the cloud chooses to allocate to that particular link, and will control the communication rate (and hence the cost of fetching requests) between the SB and the cloud. As in the previous section, one could consider two types of capacity constraints

$$C7a: \quad \sum_{f=1}^{F} w_t^f \sigma^f \le b_t, \quad t = 1, \dots, \tag{32a}$$

$$C7b: \quad \sum_{k=t}^{\infty} \gamma^{k-t} \sum_{f=1}^{F} \mathbb{E}[w_t^f \sigma^f] \le \sum_{k=t}^{\infty} \gamma^{k-t} \mathbb{E}[b_k], \tag{32b}$$

depending on whether the limit is imposed in the short term or in the long term.

With these notational conventions, one could then consider the *joint* resource allocation problem

$$\min_{\mathbf{x}_{CD}, \mathbf{x}_{SB}} \bar{C}_{CD}(\mathbf{x}_{CD}) + \bar{C}_{SB}(\mathbf{x}_{SB})$$
$$\text{s.t.} \quad \mathbf{x}_{CD} \in \mathcal{X}_{CD}, \ \mathbf{x}_{SB} \in \mathcal{X}_{SB}, \ (C7) \tag{33}$$

where the constraint C7 – either the instantaneous one in C7a or the lon-term version in C7b – couples both optimizations. It is then clear that if one dualizes C7, and the value of the Lagrange multiplier associated with C7 is known, then two separate optimizations can be run: one focusing on the cloud network and the other one on the SB. For this second optimization, consider for simplicity that the average constraint in (32b) is selected and let ν denote the Lagrange multiplier associated with such a constraint. The optimization corresponding to the SB is then

$$\min_{\mathbf{x}_{SB}} \bar{C}_{SB}(\mathbf{x}_{SB}) + \sum_{k=t}^{\infty} \gamma^{k-t} \sum_{f=1}^{F} \mathbb{E}[w_t^f \nu \sigma^f] \quad \text{s.t.} \ \mathbf{x}_{SB} \in \mathcal{X}_{SB}.$$
$$\tag{34}$$

Clearly, solving this problem is equivalent to solving the original problem in Section III, provided that the original cost is augmented with the primal-dual term associated with the

coupling constraint. To address the modified optimization, we will follow steps similar to those in Section IV-B, defining first a stochastic estimate of the Lagrange multiplier as

$$\hat{\nu}_{t+1} = \left[\hat{\nu}_t + \zeta \left(\sum_{f=1}^{F} \hat{w}_t^{f*} \sigma^f - b_t \right) \right]^+, \tag{35}$$

and then obtaining the optimal caching-fetching decisions running the schemes in Section III after replacing the original fetching cost λ_t^f with the augmented one $\lambda_{t,\text{aug}}^f = \lambda_t^f + \hat{\nu}_t \sigma_f$.

For simplicity, in this section we will limit our discussion to the case where $\hat{\nu}_t$ corresponds to the value of a Lagrange multiplier corresponding to a communication constraint. However, from a more general point of view, $\hat{\nu}_t$ represents the marginal price that the cloud network has to pay to transmit the information requested by the SB. In that sense, there exists a broad range of options to set the value of $\hat{\nu}_t$, including the congestion level at the cloud network (which is also represented by a Lagrange multiplier), or the rate (power) cost associated with the back-haul link. While a detailed discussion on those options is of interest, it goes beyond the scope of the present work.

D. Modified online solver based on Q-learning

We close this section by providing an online reinforcement-learning algorithm that modifies the one introduced in Section III to account for the multipliers introduced in Section IV. By defining per file cost $\hat{c}_k^f$ as

$$\hat{c}_k^f \left(w_k^f, a_k^f; \rho_k^f, \lambda_k^f, \hat{\mu}_k, \hat{\nu}_k \right) :=$$
$$\left(\rho_k^f + \hat{\mu}_k \sigma^f \right) a_k^f + \left(\lambda_k^f + \hat{\nu}_k \sigma^f \right) w_k^f \tag{36}$$

the problem of caching under limited cache capacity and back-haul link reduces to per file optimization as follows

$$(P8) \quad \min_{\{(w_k^f, a_k^f)\}_{k \ge t}} \sum_{k=t}^{\infty} \gamma^{k-t} \mathbb{E} \left[\hat{c}_k^f \left(a_k^f, w_k^f; \rho_k^f, \lambda_k^f, \hat{\mu}_k, \hat{\nu}_k \right) \right]$$
$$\text{s.t.} \quad (w_k^f, a_k^f) \in \mathcal{X}(r_k^f, a_{k-1}^f), \quad \forall f, \ k \ge t$$

where the updated dual variables $\hat{\mu}_k$ and $\hat{\nu}_k$ are obtained respectively by iteration (29) and (35). If we plug $\hat{c}_k^f$ instead of c_k^f into the marginalized Q-function in (21), then the solution for (P8) in current iteration k for a given file f can readily be found by solving

$$\arg\min_{(w,a) \in \mathcal{X}(r_t, a_{t-1})} \bar{Q}_{r_t, s_t}^{w, a} + w(\lambda_t + \hat{\nu}_t \sigma^f) + a(\rho_t + \hat{\mu}_t \sigma^f). \tag{37}$$

Thus, it suffices to form a marginalized Q-function for each file and solve (37), which can be easily accomplished through exhaustive search over 8 possible cache-fetch decisions $(w, a) \in \mathcal{X}(r_t, a_{t-1})$.

To simplify notation and exposition, we focus on the *limited caching capacity* constraint, and suppose that the back-haul is capable of serving any requests, thus $\hat{\nu}_t = 0$, $\forall t$. Modifications to account also for $\hat{\nu}_t \ne 0$ are straightforward.

The modified Q-learning (MQ-learning) algorithm, tabulated in Algorithm 3, essentially learns to make optimal fetch-cache decisions while accounting for the limited caching

---

Algorithm 3: Modified Q-learning for online caching

Input : $0 < \gamma,\ \beta_t < 1,\ \hat{\mu}_0, \zeta, \epsilon_t,\ M$
Output: $\hat{\check{Q}}_{r^f, s^f}^{w^f, a^f}(t+1)$

1 Initialize Set $\hat{\check{Q}}_{r^f, s^f}^{w^f, a^f}(1) = 0$ for all factors

   Set $s_0^f = 0$ and variables $\boldsymbol{\theta}_0^f = \{r_0^f, \rho_0^f, \lambda_0^f\}$ are revealed

2 for $t = 0, 1 \ldots$ **do**

3   For the current state (r_t^f, s_t^f), choose $(\check{w}_t^{f*}, \check{a}_t^{f*})$

$$(\check{w}_t^{f*}, \check{a}_t^{f*}) = \begin{cases} \text{Solve (23)} & \text{w.p. } 1 - \epsilon_t \\ \text{random } (w, a) \in \mathcal{X}_t^f(r_t^f, s_t^f) & \text{w.p. } \epsilon_t \end{cases}$$

4   Update dual variable

$$\hat{\mu}_{t+1} = \left[\hat{\mu}_t + \zeta \left(\sum_{f=1}^{F} \check{a}_t^{f*} \sigma^f - M \right) \right]^+$$

5   Incur cost $\check{c}_t^f := c_t^f(\check{a}_t^{f*}, \check{w}_t^{f*}; \rho_t^f, \lambda_t^f) + \hat{\mu}_t \check{a}_t^{f*} \sigma^f$

6   Apply $\Pi_{C4}(\cdot)$ to guarantee C4 (if required)

$$\Pi_{C4}\left[\left\{ (\check{w}_t^{f*}, \check{a}_t^{f*}) \right\}_f \right] \to \left\{ w_t^{f*}, a_t^{f*} \right\}_f$$

7   Update state $s_{t+1}^f = a_t^{f*}$

8   Request and cost parameters, $\boldsymbol{\theta}_{t+1}^f$, are revealed

9   Update all $\hat{\check{Q}}$ factors as

10

$$\hat{\check{Q}}_{r_t^f, s_t^f}^{w_t^{f*}, a_t^{f*}}(t+1) = (1 - \beta_t)\, \hat{\check{Q}}_{r_t^f, s_t^f}^{w_t^{f*}, a_t^{f*}}(t)$$
$$+ \beta_t \left[\check{c}_t^f + \gamma \min_{(w^f, a^f) \in \mathcal{X}_{t+1}^f} \hat{\check{Q}}_{r_{t+1}^f, s_{t+1}^f}^{w^f, a^f}(t) \right]$$

11 end

---

capacity constraint in C4 and/or C5. In particular, to provide a computationally efficient solver the stochastic updates corresponding to C5 are used. Subsequently, if C4 needs to be enforced, the obtained solution is projected into the feasible set through projection algorithm $\Pi_{C4}(\cdot)$. The projection $\Pi_{C4}(.)$ takes the obtained solution $\{\check{w}_t^{f*}, \check{a}_t^{f*}\}_{\forall f}$, the file sizes, as well as the marginalized Q-functions as input, and generates a feasible solution $\{w_t^{f*}, a_t^{f*}\}_{\forall f}$ satisfying C4 as follows: it sorts the files with $\check{a}_t^{f*} = 1$ in ascending Q-function order, and caches the files with the lowest Q-values until the cache capacity is reached. Overall, our modified algorithm performs a "double" learning: i) by using reinforcement schemes it learns the optimal policies that map states to actions, and ii) by using a stochastic dual approach it learns the mechanism that adapt the prices to the saturation and congestion conditions in the cache. Given the operating conditions and the design approach considered in the paper, the proposed algorithm has moderate complexity, and thanks to the reduced input dimensionality, it also converges in a moderate number of iterations.

V. NUMERICAL TESTS

In this section, we numerically assess the performance of the proposed approaches for learning optimal fetch-cache decisions. Two sets of numerical tests are provided. In the first set, summarized in Figs 2-6, the performance of the value iteration-based scheme in Alg. 1 is evaluated, and in the second set, summarized in Figs. 7-8, the performance of the Q-learning solver is investigated. In both sets, the cache and fetch cost parameters are drawn with equal probability from a finite number of values, where the mean is $\bar{\rho}^f$ and $\bar{\lambda}^f$, respectively. Furthermore, the request variable r^f is modeled as a Bernoulli random variable with mean p^f, whose value indicates the popularity of file f.

In the first set, it is assumed that p^f as well as the distribution of ρ^f, λ^f, are known a priori. 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. 2 plots the sum average cost $\bar{C}$ versus $\bar{\rho}$ for different values of $\bar{\lambda}$ and p. The fetching cost is set to $\bar{\lambda} \in \{43, 45, 50, 58\}$ for two different values of popularity $p \in \{0.3, 0.5\}$. As depicted, higher values of $\bar{\rho}, \bar{\lambda}, p$ 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 observation 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 no longer depends on $\bar{\rho}$. While this behavior occurs for the two values of p, 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. 3 depicts the sum average cost versus p for different values of $\bar{\rho}$ and $\bar{\lambda}$. The results show that for large values of $\bar{\rho}$, fetching is the optimal action, resulting in a linear increase in the total cost as p increases. In contrast, for small values of $\bar{\rho}$, caching is chosen more frequently, resulting in a sub-linear 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. 4 plots this ratio for different values of $(\bar{\rho}, \bar{\lambda})$ and fixed $p = 0.5$. 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. Similarly, the caching ratio is plotted by setting $p = 0.05$ in Fig. 5, in which fetching is mostly preferred over a wide range of storage costs due to the small value of p.
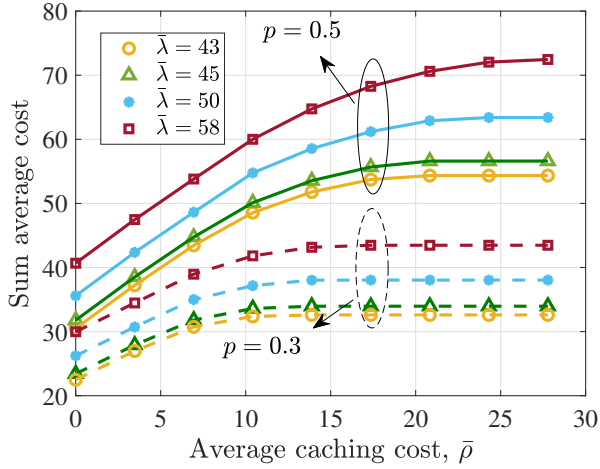
Fig. 2: Average cost versus $\bar{\rho}$ for different values of $p, \bar{\lambda}$.
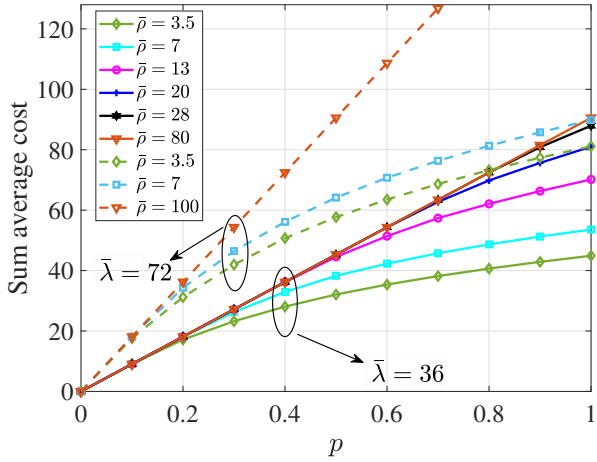

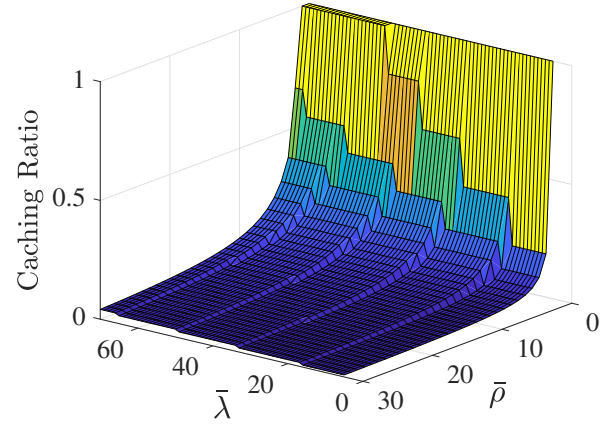Fig. 5: Caching ratio vs. $\bar{\rho}$ and $\bar{\lambda}$ for $p = 0.05$ and $s = r = 1$.

Fig. 3: Average cost versus p for different values of $\bar{\lambda}, \bar{\rho}$.
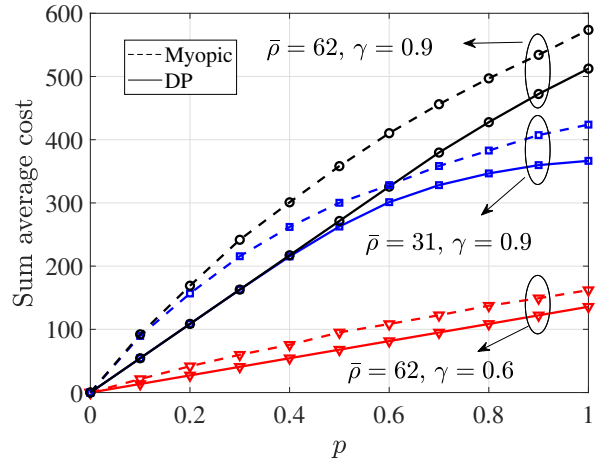
Fig. 6: Performance of DP versus myopic caching for $\bar{\lambda} = 53$.

Fig. 4: Caching ratio vs. $\bar{\rho}$ and $\bar{\lambda}$ for $p = 0.5$ and $s = r = 1$.
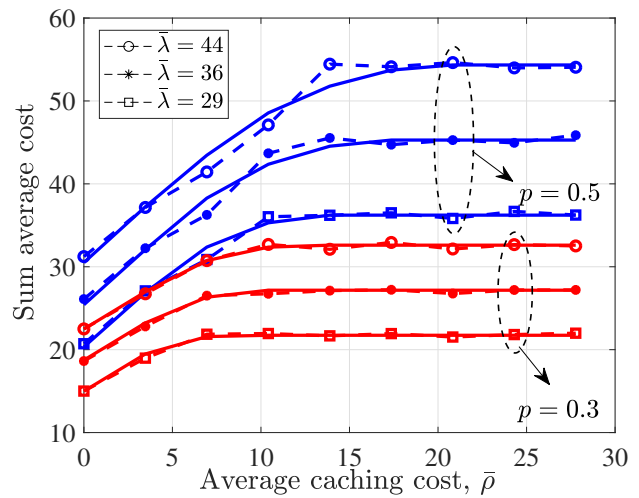

Fig. 7: Average cost versus $\bar{\rho}$ for different values of $\bar{\lambda}, p$. Solid line is for value iteration while dashed lines are for Q-learning based solver.
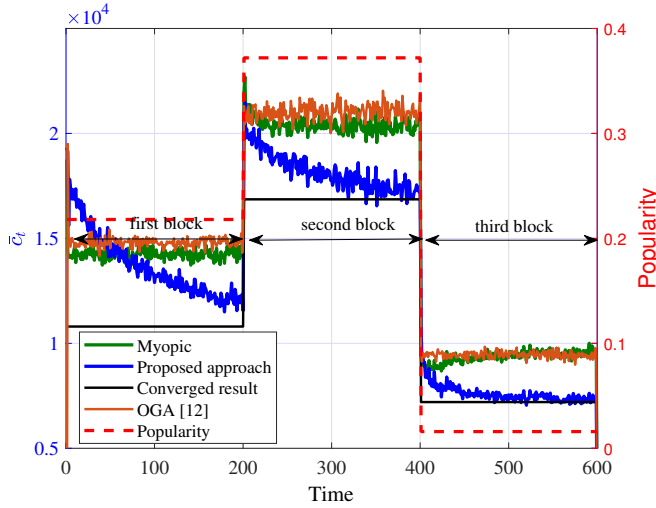
Fig. 8: Averaged immediate cost over 1000 realizations in a non-stationary setting, and a sample from popularities.

Interestingly this is true despite high fetching costs as well, and can be intuitively explained as follows: due to low popularity, deciding to cache may result in idle storing of the file in cache, thus entailing an unnecessary aggregated caching cost before the stored file can be utilized to meet user request, rendering caching suboptimal. The comparison between Fig. 4 and 5 clearly demonstrates the effect of different values of p on the performance of the cache-fetch decisions, while the proposed approach automatically adjusts to the underlying popularities.

Finally, Fig. 6 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 $\bar{\rho}, \bar{\lambda}, p$ and γ.

In the second set of tests, the performance of the online Q-learning solvers is investigated. As explained in Section III, under the assumption that the underlying distributions are stationary, the performance of the Q-learning solver should converge to the optimal one found through the value iteration algorithm. Corroborating this statement, Fig. 7 plots the sum average cost $\bar{\mathcal{C}}$ versus $\bar{\rho}$ of both the marginalized value iteration and the Q-learning solver, with $\bar{\lambda} \in \{29, 36, 44\}$ and $p \in \{0.3, 0.5\}$. The solid lines are obtained when assuming a priori knowledge of the distributions and then running the marginalized value iteration algorithm; the results and analysis are similar to the ones reported for Fig. 2. The dashed curves however, are found by assuming unknown distributions and running the Q-learning solver. Sum average cost is reported after first 1000 iterations. As the plot suggests, despite the lack of a priori knowledge on the distributions, the Q-learning solver is able to find the optimal decision making rule. As a result, it yields the same sum average cost as that of value-iteration under known distributions.

The last experiment investigates the impact of the instantaneous cache capacity constraint in C4 as well as non-stationary

distributions for popularities and costs. To this end, 1,000 different realizations (trajectories) of the random state processes are drawn, each of length $T = 600$. For every realization, the cost c_t [cf. (6)] at each and every time instant is found, and the cost trajectory is averaged across the 1,000 realizations. Specifically, let c_t^i denote the ith realization cost at time t, and define the averaged cost trajectory as $\bar{c}_t := \frac{1}{1000} \sum_{i=1}^{1000} c_t^i$. Fig. 8 reports the average trajectory of $\bar{c}_t$ in a setup where the total number of files is set to $F = 500$, the file sizes are drawn uniformly at random from the interval $[1, 100]$, and the total cache capacity is set to 40% of the aggregate file size. Adopted parameters for the MQ-learning solver are set to $\beta_t = 0.3$, and $\epsilon = 0.01$. Three blocks of iterations are shown in the figure, where in each block a specific distribution of popularities and costs are considered. For instance, the dashed line shows the popularity of a specific file in one of the realizations, where in the fist block $p = 0.23$, in the second block $p = 0.37$, and in the third one $p = 0.01$. The cost parameters have means $\bar{\lambda} = 44, \bar{\rho} = 2$, $\bar{\lambda} = 40, \bar{\rho} = 5$, and $\bar{\lambda} = 38, \bar{\rho} = 2$ in the consecutive blocks, respectively.

As Fig. 8, the proposed MQ-learning algorithm incurs large costs during the first few iterations. Then, it gradually adapts to the file popularities and cost distributions, and learns how to make optimal fetch-cache decisions, decreasing progressively the cost in each of the blocks. To better understand the behavior of the algorithm and assess its effectiveness, we compare it with that of Online Gradient Ascent (OGA) [18] as a representative state-of-the-art method among the class of online expert algorithms, the myopic policy and the stationary policy serving as the benchmark, respectively. In contrast to the OGA method, our decision variables are not continuous, but binary. Hence, caching decisions in OGA are projected into the binary feasible set for fair comparison. In general, since OGA and the myopic caching only use the current state and requests, their performance is inferior to that of our proposed method, where knowledge of the underlying request and price distributions is carefully utilized. During the first iterations however, when the MQ-learning algorithm has not adapted to the distribution of pertinent parameters, OGA and the myopic policy perform better; on the other hand, as the learning proceeds, the MQ-learning starts to make more precise decisions and, remarkably, in a couple of hundreds of iterations it is able to perform very close to the optimal policy.

Furthermore, to investigate the scalability of our proposed approach, Tables I, II, and III report the run-time (in seconds[1]) versus the number of files F as well as the storage capacity M, set as a ratio of the total aggregated file sizes. Although the proposed approach has slightly higher run-time due to the utilized dual-decomposition technique and the solution of the arising integer DP, all methods scale gracefully (linearly) as the number of files increases from 1K to 10K. Note that much faster run-times can be obtained by effectively utilizing available computation resources in parallel. However, the aim here is just to assess scalability of proposed approach compared with the OGA one. Therefore, optimally allocating

---

[1]We run these simulations in parallel with 4 pools of workers, utilizing a machine with Intel(R) Core(TM) i7-4770 CPU @ 3.4 GHz specifications.

$\%M$ ╲ F	1K	2K	3K	4K	6K	8K	10K
10 %	148	240	337	444	662	870	1089
20 %	141	229	327	435	625	858	1052
40 %	139	232	326	422	610	815	980
60 %	149	251	372	497	699	949	1086

TABLE I: Run-time of the proposed caching.

$\%M$ ╲ F	1K	2K	3K	4K	6K	8K	10K
10 %	70	120	176	241	411	622	808
20 %	73	123	183	250	389	569	796
40 %	70	122	182	272	406	585	779
60 %	92	170	254	356	551	736	908

TABLE II: Run-time of OGA caching.

$\%M$ ╲ F	1K	2K	3K	4K	6K	8K	10K
10 %	70	126	205	286	491	721	969
20 %	84	153	232	317	507	736	1025
40 %	88	157	236	328	575	822	1105
60 %	87	161	240	336	563	834	1141

TABLE III: Run-time of Myopic caching.

computation resources for obtaining best run-times is not investigated at all.

VI. CONCLUSIONS

A generic setup where a caching unit makes sequential fetch-cache decisions based on dynamic prices and user requests was investigated. Critical constraints were identified, the aggregated cost across files and time instants was formed, and the optimal adaptive caching was then formulated as a stochastic optimization problem. Due to the effects of the current cache decisions on future costs, the problem was cast as a dynamic program. To address the inherent functional estimation problem that arises in this type of programs, while leveraging the underlying problem structure, several computationally efficient algorithms were developed, including off-line (batch) approaches, as well as online (stochastic) approaches based on Q-learning. The last part of the paper was devoted to dynamic pricing mechanisms that allowed handling constraints both in the storage capacity of the cache memory, as well as on the back-haul transmission link connecting the caching unit with the cloud.

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

[3] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. C. M. Leung, "Cache in the air: exploiting content caching and delivery techniques for 5G systems," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 131–139, Feb. 2014.

[4] P. Blasco and D. Gündüz, "Learning-based optimization of cache content in a small cell base station," in *Proc. 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 Trans. 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 Trans. Multimedia*, vol. 18, no. 12, pp. 2503–2516, Dec. 2016.

[8] E. Leonardi and G. Neglia, "Implicit coordination of caches in small cell networks under unknown popularity profiles," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1276–1285, June 2018.

[9] J. Li, S. Shakkottai, J. C. S. Lui, and V. Subramanian, "Accurate learning or fast mixing? Dynamic adaptability of caching algorithms," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1314–1330, June 2018.

[10] B. N. Bharath, K. G. Nagananda, and H. V. Poor, "A learning-based approach to caching in heterogenous small cell networks," *IEEE Trans. Commun.*, vol. 64, no. 4, pp. 1674–1686, April 2016.

[11] G. Hasslinger, K. Ntougias, F. Hasslinger, and O. Hohlfeld, "Performance evaluation for new web caching strategies combining LRU with score based object selection," *Computer Networks*, vol. 125, pp. 172–186, 2017.

[12] J. Kwak, G. Paschos, and G. Iosifidis, "Dynamic cache rental and content caching in elastic wireless CDNs," in *Proc. Intl. Symp. Modeling Opt. Mobile, Ad Hoc, Wireless Netw.*, Shanghai, China, May 2018, pp. 1–8.

[13] S. Traverso, A. Mohamed, et. al., "Temporal locality in today's content caching: Why it matters and how to model it," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 5, pp. 5–12, Nov. 2013.

[14] M. Leconte, G. Paschos, L. Gkatzikis, M. Draief, S. Vassilaras, and S. Chouvardas, "Placing dynamic content in caches with small population," in *Proc. Intl. Conf. Comput. Commun.*, San Francisco, USA, April 2016, pp. 1–9.

[15] B. N. Bharath, K. G. Nagananda, D. Gündüz, and H. V. Poor, "Caching with time-varying popularity profiles: A learning-theoretic perspective," *IEEE Trans. Commun.*, vol. 66, no. 9, pp. 3837–3847, Sep. 2018.

[16] R. Pedarsani, M. A. Maddah-Ali, and U. Niesen, "Online coded caching," *IEEE/ACM Trans. Netw.*, vol. 24, no. 2, pp. 836–845, Apr. 2016.

[17] S. M. Azimi, O. Simeone, A. Sengupta, and R. Tandon, "Online edge caching and wireless delivery in fog-aided networks with dynamic content popularity," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1189–1202, June 2018.

[18] G. Paschos, A. Destounis, L. Vigneri, and G. Iosifidis, "Learning to cache with no regrets," in *Proc. of INFOCOM Conf.*, Paris, France, April 2019, pp. 545–549.

[19] T. Lykouris and S. Vassilvitskii, "Competitive caching with machine learned advice," *arXiv preprint arXiv:1802.05399*, 2018.

[20] S. Geulen, B. Vöcking, and M. Winkler, "Regret minimization for online buffering problems using the weighted majority algorithm," in *Proc. of COLT*, Haifa, Israel, June 2010, pp. 132–143.

[21] M. Englert, H. Röglin, J. Spönemann, and B. Vöcking, "Economical caching," in *Proc. of STACS*, Freiburg, Germany, Feb. 2009, pp. 385–396.

[22] 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.

[23] 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.

[24] A. Sadeghi, G. Wang, and G. B. Giannakis, "Adaptive caching via deep reinforcement learning," *arXiv preprint arXiv:1902.10301*, 2019.

[25] L. Pu, L. Jiao, X. Chen, L. Wang, Q. Xie, and J. Xu, "Online resource allocation, content placement and request routing for cost-efficient edge caching in cloud radio access networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 8, pp. 1751–1767, Aug 2018.

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

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

This is the author's version of an article that has been published in this journal. Changes were made to this version by the publisher prior to publication.

The final version of record is available at http://dx.doi.org/10.1109/JSAC.2019.2933780

15

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

[29] G. Wang, V. Kekatos, A. J. Conejo, and G. B. Giannakis, "Ergodic energy management leveraging resource variability in distribution grids," *IEEE Trans. Power Syst.*, vol. 31, no. 6, pp. 4765–4775, Nov. 2016.

[30] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, Cambridge, MA, USA: MIT Press, 2016.

[31] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proc. Intl. Conf. Comput. Commun.*, New York, USA, March 1999, pp. 126–134.

[32] J. N. Tsitsiklis, "Asynchronous stochastic approximation and Q-learning," *Mach. learn.*, vol. 16, no. 3, pp. 185–202, Sept. 1994.

[33] C. Watkins and P. Dayan, "Q-learning," *Mach. learn.*, vol. 8, no. 3-4, pp. 279–292, May 1992.

[34] C. Watkins, *Learning from delayed rewards*, Ph.D. thesis, King's College, Cambridge, 1989.

[35] V. S. Borkar and S. P. Meyn, "The ODE method for convergence of stochastic approximation and reinforcement learning," *SIAM J. Control Optim.*, vol. 38, no. 2, pp. 447–469, 2000.

[36] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*, Upper Saddle River, NJ, USA,: Prentice-Hall, 2010.

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

[38] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge, U.K.: Cambridge Univ. Press, 2004.

[39] A. G. Marques, L. M. Lopez-Ramos, G. B. Giannakis, J. Ramos, and A. J. Caamaño, "Optimal cross-layer resource allocation in cellular networks using channel- and queue-state information," *IEEE Trans. Veh. Technol.*, vol. 61, no. 6, pp. 2789–2807, July 2012.

Antonio G. Marques received the telecommunications engineering degree and the Doctorate degree, both with highest honors, from the Carlos III University of Madrid, Spain, in 2002 and 2007, respectively. In 2007, he became a faculty of the Department of Signal Theory and Communications, King Juan Carlos University, Madrid, Spain, where he currently develops his research and teaching activities as an Associate Professor. From 2005 to 2015, he held different visiting positions at the University of Minnesota, Minneapolis. In 2015, 2016 and 2017 he was a visitor scholar at the University of Pennsylvania, Philadelphia. His current research focuses on nonlinear and stochastic optimization of wireless and power networks, signal processing for graphs, and data science for networks. Dr. Marques has served the IEEE in a number of posts (currently, he is a member of the IEEE Signal Process. Theory and Methods Tech. Comm., a member of the IEEE Signal Process. Big Data Special Interest Group, the Technical Co-chair of the 2019 IEEE CAMSAP Workshop, and the General Co-chair of the 2019 IEEE Data Science Workshop). His work has been awarded in several conferences and workshops, with recent ones including IEEE SSP 2016, IEEE SAM 2016 and Asilomar 2015.

Alireza Sadeghi (S'16) received the B.Sc. degree (Hons.) from the Iran University of Science and Technology, Tehran, Iran, in 2012, and the M.Sc. degree from the University of Tehran, in 2015, both in Electrical Engineering. From February to July 2015 he was a visiting scholar at the University of Padova, Italy. Since January 2016, he is perusing the Ph.D. degree in Electrical Engineering with the Department of Electrical and Computer Engineering, University of Minnesota, MN, USA. His research interests include machine learning, optimization, and signal processing with applications in networking. He was the recipient of ADC Fellowship offered by the Digital Technology Center for 2016-2017; and the student travel awards from IEEE Communication Society and National Science Foundation (NSF).

Georgios. B. Giannakis (Fellow'97) received his Diploma in Electrical Engr. from the Ntl. Tech. Univ. of Athens, Greece, 1981. From 1982 to 1986 he was with the Univ. of Southern California (USC), where he received his MSc. in Electrical Engineering, 1983, MSc. in Mathematics, 1986, and Ph.D. in Electrical Engr., 1986. He was a faculty member with the University of Virginia from 1987 to 1998, and since 1999 he has been a professor with the Univ. of Minnesota, where he holds an ADC Endowed Chair, a University of Minnesota McKnight Presidential Chair in ECE, and serves as director of the Digital Technology Center.

His general interests span the areas of statistical learning, communications, and networking - subjects on which he has published more than 450 journal papers, 750 conference papers, 25 book chapters, two edited books and two research monographs (h-index 142). Current research focuses on Data Science, and Network Science with applications to the Internet of Things, social, brain, and power networks with renewables. He is the (co-) inventor of 32 patents issued, and the (co-) recipient of 9 best journal paper awards from the IEEE Signal Processing (SP) and Communications Societies, including the G. Marconi Prize Paper Award in Wireless Communications. He also received Technical Achievement Awards from the SP Society (2000), from EURASIP (2005), a Young Faculty Teaching Award, the G. W. Taylor Award for Distinguished Research from the University of Minnesota, and the IEEE Fourier Technical Field Award (inaugural recipient in 2015). He is a Fellow of EURASIP, and has served the IEEE in a number of posts, including that of a Distinguished Lecturer for the IEEE-SPS.

Fatemeh Sheikholeslami (S'13) received the B.Sc. and M.Sc. degrees in Electrical Engineering from the University of Tehran and Sharif University of Technology, Tehran, Iran, in 2010 and 2012, respectively. Since September 2013, she has been working toward the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Minnesota, MN, USA. Her research interests include machine learning, network science, and signal processing. She has been the recipient of the ADC Fellowship for 2013-2014 from the Digital Technology Center at the University of Minnesota.