Joint Pushing and Caching for Bandwidth Utilization Maximization in Wireless Networks

Yaping Sun, Ying Cui, and Hui Liu
Shanghai Jiao Tong University, China

Abstract—Joint pushing and caching is recognized as an efficient remedy to the problem of spectrum scarcity incurred by the tremendous mobile data traffic. In this paper, by exploiting storage resource at end-users and predictability of users’ demand processes, we would like to design the optimal joint pushing and caching to maximize bandwidth utilization, which is one of the most important concerns for network operators. In particular, we formulate the stochastic optimization problem as an infinite horizon average cost Markov Decision Process (MDP), for which there generally exist only numerical solutions without many insights. By structural analysis, we show how the optimal policy achieves a balance between the current transmission cost and the future average transmission cost. In addition, we show that the optimal average transmission cost decreases with the cache size, revealing a tradeoff between the cache size and the bandwidth utilization. Moreover, due to the fact that obtaining a numerical optimal solution suffers the curse of dimensionality and implementing it requires a centralized controller and global system information, we develop a decentralized policy with polynomial complexity w.r.t. the numbers of users and files as well as cache size, by a linear approximation of the value function and relaxing the original policy optimization problem. The results in this paper offer useful guidelines for designing practical cache-enabled content-centric wireless networks.

I. INTRODUCTION

The rapid proliferation of smart mobile devices has triggered an unprecedented growth of the global mobile data traffic, which exhibits the following features. (i) Requests for common popular contents may arrive at the server from multiple users asynchronously. (ii) The demand process of a user may be temporally correlated and predictable to certain extent based on priori information. (iii) There exist large traffic variations between the peak and off-peak hours as well as small or medium traffic fluctuations over a smaller timescale, leading to the underutilization of wireless spectrum. Given these features, joint pushing and caching is proposed as a promising approach to better support the exponential growth of wireless data traffic. In particular, in view of features (i) and (ii), traffic load can be greatly reduced by placing contents at end-users. On the other hand, leveraging features (ii) and (iii), bandwidth utilization can be significantly improved by proactively transmitting (i.e., pushing) contents to user caches.

Many recent works have studied joint pushing and caching to improve the performance of wireless networks. For instance, [1] considers offline joint pushing and caching to minimize the energy consumption, assuming perfect knowledge of future content requests. In most cases, the assumption cannot be satisfied, and hence the proposed offline joint design has limited applications. To address this problem, in [2]–[6], the authors consider joint pushing and caching based on statistical information of content requests. In particular, [2] and [3] exploit temporal correlation of content demands in the joint designs. However, they consider a single user setup only, without reflecting asynchronous demands for common contents from multiple users, and thus the proposed joint designs cannot be directly applied to practical networks with multiple users. Moreover, the demand correlation in [3] lies in one time slot only and there is no future reuse for the requested files. In [4]–[6], temporal correlation of content demands is not captured, and hence the benefits of joint pushing and caching cannot be fully unleashed. Furthermore, the joint designs in [5] and [6] do not consider future reuse of requested files, and thus cannot be applied to certain applications, such as music and video streaming. The last not the least, [1]–[6] do not address the problem of the optimal joint pushing and caching design to maximize the bandwidth utilization by exploiting storage resource at end-users and the aforementioned traffic features.

In this paper, we shed light on the above mentioned issue. Specifically, we consider a cache-enabled content-centric wireless network consisting of a single server connected to multiple users via a shared and errorless link. Each user is equipped with a cache of limited size and generates inelastic file requests. We model the demand process of each user as a Markov chain, capturing the asynchronous feature and temporal correlation of file requests as well as traffic fluctuations. The last not the least, [1]–[6] do not address the problem of the optimal joint pushing and caching design to maximize the bandwidth utilization by exploiting storage resource at end-users and the aforementioned traffic features.

In this paper, we shed light on the above mentioned issue. Specifically, we consider a cache-enabled content-centric wireless network consisting of a single server connected to multiple users via a shared and errorless link. Each user is equipped with a cache of limited size and generates inelastic file requests. We model the demand process of each user as a Markov chain, capturing the asynchronous feature and temporal correlation of file requests as well as traffic fluctuations. The last not the least, [1]–[6] do not address the problem of the optimal joint pushing and caching design to maximize the bandwidth utilization by exploiting storage resource at end-users and the aforementioned traffic features.

In this paper, we shed light on the above mentioned issue. Specifically, we consider a cache-enabled content-centric wireless network consisting of a single server connected to multiple users via a shared and errorless link. Each user is equipped with a cache of limited size and generates inelastic file requests. We model the demand process of each user as a Markov chain, capturing the asynchronous feature and temporal correlation of file requests as well as traffic fluctuations. The last not the least, [1]–[6] do not address the problem of the optimal joint pushing and caching design to maximize the bandwidth utilization by exploiting storage resource at end-users and the aforementioned traffic features.

In this paper, we shed light on the above mentioned issue. Specifically, we consider a cache-enabled content-centric wireless network consisting of a single server connected to multiple users via a shared and errorless link. Each user is equipped with a cache of limited size and generates inelastic file requests. We model the demand process of each user as a Markov chain, capturing the asynchronous feature and temporal correlation of file requests as well as traffic fluctuations. The last not the least, [1]–[6] do not address the problem of the optimal joint pushing and caching design to maximize the bandwidth utilization by exploiting storage resource at end-users and the aforementioned traffic features.

In this paper, we shed light on the above mentioned issue. Specifically, we consider a cache-enabled content-centric wireless network consisting of a single server connected to multiple users via a shared and errorless link. Each user is equipped with a cache of limited size and generates inelastic file requests. We model the demand process of each user as a Markov chain, capturing the asynchronous feature and temporal correlation of file requests as well as traffic fluctuations. The last not the least, [1]–[6] do not address the problem of the optimal joint pushing and caching design to maximize the bandwidth utilization by exploiting storage resource at end-users and the aforementioned traffic features.

In this paper, we shed light on the above mentioned issue. Specifically, we consider a cache-enabled content-centric wireless network consisting of a single server connected to multiple users via a shared and errorless link. Each user is equipped with a cache of limited size and generates inelastic file requests. We model the demand process of each user as a Markov chain, capturing the asynchronous feature and temporal correlation of file requests as well as traffic fluctuations. The last not the least, [1]–[6] do not address the problem of the optimal joint pushing and caching design to maximize the bandwidth utilization by exploiting storage resource at end-users and the aforementioned traffic features.
II. System Model

A. Network Architecture

We consider a cache-enabled content-centric wireless network, which consists of a single server connected with \( K \) users, denoted as \( K = \{1, 2, \ldots, K\} \), via a shared link. That is, all the users have the same channel condition. In addition, we assume that the shared link is errorless. The server is accessible to a database of \( F \) files, denoted as \( F = \{1, 2, \ldots, F\} \). All the files are of the same size. Each user is equipped with a cache of size \( C \) (in files). The system operates over an infinite time horizon and time is slotted, indexed by \( t = 0, 1, 2, \ldots \). At the beginning of each time slot, each user submits at most one file request, which is assumed to be delay intolerant and must be served before the end of the slot, either by its own cache if the requested file has been stored locally, or by the server via the shared link. At each slot, the server can reactively transmit a file requested by some users at the slot or proactively transmit (i.e., push) a file which has not been requested by any user at the slot. Each transmitted file can be received by all the users concurrently before the end of the time slot. After being received, a file could be stored into some user caches.

B. System State

1) Demand State: At the beginning of time slot \( t \), each user \( k \) generates at most one file request. Let \( A_k(t) \in F \triangleq F \cup \{0\} \) denote the demand state of user \( k \) at the beginning of time slot \( t \), where \( A_k(t) = 0 \) indicates that user \( k \) requests nothing, and \( A_k(t) = f \in F \) indicates that user \( k \) requests file \( f \). Here, \( F \) denotes the demand state space for each user which is of cardinality \( F + 1 \). Let \( A(t) \triangleq (A_k(t))_{k \in K} \in F^K \) denote the system demand state (of the \( K \) users) at the beginning of time slot \( t \), where \( F^K \) represents the system demand state space. Note that the cardinality of \( F^K \) is \((F + 1)^K\), which increases exponentially with \( K \). Furthermore, we assume that each file request is delay-intolerant, i.e., it must get served within one time slot.

For user \( k \), we assume that \( A_k(t) \) evolves according to a first-order \((F + 1)-\)state Markov chain, denoted as \( \{A_k(t) : t = 0, 1, 2, \ldots\} \), which captures temporal correlation of order one of user \( k \)'s demand process and is a widely adopted traffic model \( 3 \). Let \( \Pr[A_k(t + 1) = j | A_k(t) = i] \) represent the transition probability of going to state \( j \in F \) at time slot \( t + 1 \) given that the demand state at time slot \( t \) is \( i \in F \) for user \( k \)'s demand process. Assume that \( \{A_k(t)\} \) is time-homogeneous and denote \( q_{i,j}^{(k)} \triangleq \Pr[A_k(t + 1) = j | A_k(t) = i] \). Furthermore, we restrict our attention to an irreducible Markov chain. Denote with \( Q_k \triangleq (q_{i,j}^{(k)})_{i,j \in F, j \in F} \) the transition probability matrix of \( \{A_k(t)\} \). We assume that the \( K \) time-homogeneous Markov chains, i.e., \( \{A_k(t)\} \), \( k \in K \), are independent of each other. Thus, we have \( \Pr[A(t + 1) = j | A(t) = i] = \prod_{k=1}^{K} q_{i,k}^{(k)} \), where \( i \triangleq (i_k)_{k \in K} \in F^K \) and \( j \triangleq (j_k)_{k \in K} \in F^K \).

2) Cache State: Let \( S_{k,f}(t) \in \{0, 1\} \) denote the cache state of file \( f \) in the storage of user \( k \) at time slot \( t \), where \( S_{k,f}(t) = 1 \) means that file \( f \) is cached in user \( k \)'s storage and \( S_{k,f}(t) = 0 \) otherwise. Under the cache size constraint, we have

\[
\sum_{f \in F} S_{k,f}(t) \leq C, \quad k \in K.
\]

Let \( S(t) \triangleq (S_{k,f}(t))_{f \in F, k \in K} \in S^K \) denote the cache state for user \( k \) at time slot \( t \), where \( S \triangleq \{(S_f)_{f \in F} : \sum_{f \in F} S_f \leq C\} \) represents the cache state space for each user. Here, the user index is suppressed considering that the cache state space is the same across all the users. Let \( S(t) \triangleq (S_{k,f}(t))_{k \in K, f \in F} \in S^K \) denote the system cache state at time slot \( t \), where \( S^K \) represents the system cache state space. The cardinality of \( S^K \) is \((\sum_{i=0}^{C} (\binom{F}{i}) \cdot )^K \), which also increases with the number of users \( K \) exponentially.

3) System State: At time slot \( t \), the system state consists of the system demand state and the system cache state, denoted as \( \chi(t) \triangleq (A(t), S(t)) \in F^K \times S^K \), where \( F^K \times S^K \) represents the system state space.

C. System Action

1) Pushing Action: A file transmission could be reactive or proactive at each time slot. Denote with \( R(t) \in \{0, 1\} \) the reactive transmission action for file \( f \) at time slot \( t \), where \( R(t) = 1 \) when there exists at least one user who requests file \( f \) but could not find it in its local cache and \( R(t) = 0 \) otherwise. Thus, we have

\[
R(t) = 1 \quad \text{min}_{k \in K, A_k(t) = f} S_{k,f}(t), \quad f \in F,
\]

which is determined directly by \( \chi(t) \). Denote with \( R(t) \triangleq (R_i(t))_{f \in F} \) the system reactive transmission action at time slot \( t \) given by \( 2 \). On the other hand, denote with \( P_f(t) \in \{0, 1\} \) the pushing action for file \( f \) at time slot \( t \), where \( P_f(t) = 1 \) denotes that file \( f \) is pushed (i.e., transmitted proactively) and \( P_f(t) = 0 \) otherwise. Considering that file \( f \) is transmitted at most once at time slot \( t \), we have

\[
P_f(t) + R_f(t) \leq 1, \quad f \in F.
\]

By \( 2 \) and \( 3 \), we obtain the pushing action constraint as:

\[
P_f(t) \leq \min_{k \in K, A_k(t) = f} S_{k,f}(t), \quad f \in F.
\]

Denote with \( P(t) \triangleq (P_f(t))_{f \in F} \in U_P(\chi(t)) \) the system pushing action at time slot \( t \), where \( U_P(\chi(t)) \triangleq \{(P_f)_{f \in F} : P_f \leq \min_{k \in K, A_k(t) = f} S_{k,f}(t)\} \) represents the system pushing action space under system state \( \chi \).

System pushing action \( P \) together with reactive transmission action \( R \) incurs a certain transmission cost. We assume that the transmission cost is an increasing and continuously convex function of the corresponding traffic load (i.e., \( \sum_{f \in F} (R_f + P_f) \)), denoted by \( \phi \). In accordance with the practice, we further assume that \( \phi(0) = 0 \). For example, we can choose \( \phi(x) = \)

1We assume that the duration of each time slot is long enough to average the small-scale channel fading process, and hence the ergodic capacity can be achieved using channel coding.

2Note that we do not need to design the reactive transmission action \( R \).
implies that the cache state of file following cache update constraint: we have the following two more cache update constraints: Let
\begin{align*}
\sum_{t=1}^{\infty} x(t) &= 1, \\
\sum_{t=1}^{\infty} y(t) &= 1, \\
\sum_{t=1}^{\infty} \Delta x(t) &= 1.
\end{align*}

\[ a^x - 1 \text{ with } a > 1 \text{ or } \phi(x) = x^d \text{ with } d \geq 2. \]

Fig. 1. An illustration of the relationship between the average cost and bandwidth utilization. Note that \( \sum_{t=1}^{\infty} x(t) = \frac{1}{2} \sum_{t=1}^{\infty} y(t) \), while \( \sum_{t=1}^{\infty} x(t) > \sum_{t=1}^{\infty} y(t) \).

2) Caching Action: After the transmitted files being received by the users, the system cache state may be updated. Let \( \Delta S_{k,f}(t) \in \{-1, 0, 1\} \) denote the caching action for file \( f \) at user \( k \) at the end of time slot \( t \), where \( \Delta S_{k,f}(t) = 1 \) means that file \( f \) is stored into the cache of user \( k \), \( \Delta S_{k,f}(t) = 0 \) implies that the cache state of file \( f \) at user \( k \) does not change, and \( \Delta S_{k,f}(t) = -1 \) indicates that file \( f \) is removed from the cache of user \( k \). Accordingly, the caching action satisfies the following cache update constraint:

\[-S_{k,f}(t+1) = S_{k,f}(t) + \Delta S_{k,f}(t), \quad f \in \mathcal{F}, k \in \mathcal{K}.\tag{5}\]

where \( R_f(t) \) is given by \( \phi(\Delta S(t)) \). Here, the first inequality is to guarantee that file \( f \) can be removed from the cache of user \( k \) only when it has been stored at user \( k \), and the second inequality is to guarantee that file \( f \) can be stored into the cache of user \( k \) only when it has been transmitted from the server. The cache state evolves according to:

\[ S_{k,f}(t+1) = S_{k,f}(t) + \Delta S_{k,f}(t), \quad f \in \mathcal{F}, k \in \mathcal{K}. \tag{6}\]

Since \( S_{k,f}(t+1) \in \{0,1\} \) and \( S_{k,f}(t+1) \) also satisfies \( \phi(\Delta S(t)) \), we have the following two more cache update constraints:

\[ S_{k,f}(t) + \Delta S_{k,f}(t) \in \{0,1\}, \quad f \in \mathcal{F}, k \in \mathcal{K}, \tag{7}\]

\[ \sum_{f \in \mathcal{F}} (S_{k,f}(t) + \Delta S_{k,f}(t)) \leq C, \quad f \in \mathcal{F}, k \in \mathcal{K}. \tag{8}\]

From \( \phi(\Delta S(t)) \), \( \phi(\Delta S(t)) \) and \( \phi(\Delta S(t)) \), we denote with \( \Delta S_k(t) = (\Delta S_{k,f}(t))_{f \in \mathcal{F}} \in U_{\Delta S}(\chi(t), P(t)) \) the caching action at user \( k \) at the end of time slot \( t \), where \( U_{\Delta S,k}(\chi(t), P(t)) \) denotes the system caching action space under system state \( \chi \) and pushing action \( P \).

3) System Action: At time slot \( t \), the system action consists of both the pushing action and caching action, denoted with \( (P(t), \Delta S(t)) \), \( (P(t), \Delta S(t)) \in U(\chi(t)) \) the system action at time slot \( t \), where \( U(\chi(t)) \) denotes the system action space under system state \( \chi \).

III. PROBLEM FORMULATION

Given an observed system state \( \chi \), the joint pushing and caching action denoted as \( (P, \Delta S) \) is determined according to a policy defined as below.

Definition 1 (Stationary Joint Pushing and Caching Policy): A stationary joint pushing and caching policy \( \mu \) is a mapping from the system state \( \chi \) to the system action \( (P, \Delta S) \), i.e., \( (P, \Delta S) = \mu(\chi) \in U(\chi) \).

Specifically, we have \( P = \mu_P(\chi) \) and \( \Delta S = \mu_{\Delta S}(\chi, P) \in U_{\Delta S}(\chi, P) \).

From the properties of \( \{A(t)\} \) and \( \{S(t)\} \), we see that the induced system state process \( \{\chi(t)\} \) under policy \( \mu \) is a controlled Markov chain. The average transmission cost under policy \( \mu \) is given by

\[ \phi(\mu) = \limsup_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E} \left[ \phi \left( \sum_{f \in \mathcal{F}} R_f(t) + P_f(t) \right) \right], \tag{9}\]

where \( R_f(t) \) is obtained from \( \phi(\Delta S(t)) \) and the expectation is taken w.r.t. the measure induced by the \( K \) Markov chains. As illustrated in Fig. 1, \( \phi(\mu) \) could reflect the bandwidth utilization.

In this paper, we expect to obtain an optimal joint pushing and caching policy \( \mu \) to minimize the average transmission cost \( \phi(\mu) \) defined in \( \phi(\Delta S(t)) \), i.e., maximizing the bandwidth utilization.

Problem 1 (Joint Pushing and Caching Optimization):

\[ \tilde{\phi}^* \triangleq \min_{\mu} \tilde{\phi}(\mu) \]

s.t. \( \phi(\Delta S(t)) \), \( \phi(\Delta S(t)) \), \( \phi(\Delta S(t)) \), \( \phi(\Delta S(t)) \), \( \phi(\Delta S(t)) \), \( \phi(\Delta S(t)) \).

where \( \tilde{\phi}^* \) denotes the minimum average transmission cost under the optimal policy \( \mu^* \triangleq (\mu_P^*, \mu_{\Delta S}^*) \), i.e., \( \tilde{\phi}^* = \phi(\mu^*) \).

Problem 1 is an infinite horizon average cost MDP. According to Definition 4.2.2 and Proposition 4.2.6 in [2], we know that there exists an optimal policy that is unichain. Hence, in this paper, we restrict our attention to stationary unichain policies. Since the MDP we consider is a unichain infinite horizon average cost MDP with finite state and action spaces as well as a bounded per-stage cost, there always exists a deterministic stationary unichain policy that is optimal. Thus, it is sufficient to revolve around the deterministic stationary unichain policy space. In the sequel, we use policy \( \mu \) to represent a deterministic stationary unichain policy.
IV. OPTIMAL POLICY

A. Optimality Equation

We can obtain the optimal joint pushing and caching policy \( \mu^* \) through solving the equivalent Bellman equation.

**Lemma 1 (Equivalent Bellman Equation):** There exist a scalar \( \theta \) and a value function \( V(\cdot) \) satisfying

\[
\theta + V(\chi) = \min_{(P, \Delta S) \in U(\chi)} \left\{ \phi \left( \sum_{f \in \mathcal{F}} (R_f + P_f) \right) + \sum_{A' \in \mathcal{F}^K} \Pr[A'|A]V(A', S + \Delta S), \right. \\
\left. \quad \chi \in \mathcal{F}^K \times \mathcal{S}^K \right\}, 
\]

where \( R_f \) is given by \( \mathcal{S}_f \). Then, \( \theta = \hat{\phi}^* \) is the optimal value to Problem 1 for all initial system states \( \chi \in \mathcal{F}^K \times \mathcal{S}^K \), and the optimal policy \( \mu^* \) can be obtained from

\[
\mu^*(\chi) = \arg \min_{(P, \Delta S) \in U(\chi)} \left\{ \phi \left( \sum_{f \in \mathcal{F}} (R_f + P_f) \right) + \sum_{A' \in \mathcal{F}^K} \Pr[A'|A]V(A', S + \Delta S) \right\}, \quad \chi \in \mathcal{F}^K \times \mathcal{S}^K. 
\]

From (11), we see that the optimal policy \( \mu^* \) achieves a balance between the current transmission cost (i.e., the first term) and the future average transmission cost (i.e., the second term). Moreover, how \( \mu^* \) achieves the balance is illustrated in the following corollary.

**Corollary 1:** The optimal pushing policy \( \mu^*_P \) is given by

\[
\mu^*_P(\chi) = \arg \min_{P \in U_P(\chi)} \left\{ \phi \left( \sum_{f \in \mathcal{F}} (R_f + P_f) \right) + W(\chi, P) \right\}, \\
\quad \chi \in \mathcal{F}^K \times \mathcal{S}^K, 
\]

where \( W(\chi, P) \equiv \Delta S_{\in \mathcal{U}_{\Delta S}(\chi, P)} \sum_{A' \in \mathcal{F}^K} P[A'|A]V(A', S + \Delta S) \) is a decreasing function of \( P \). Furthermore, the optimal caching policy \( \mu^*_C \) is given by

\[
\mu^*_C(\chi; \mu^*_P(\chi)) = \arg \min_{\Delta S \in \mathcal{U}_{\Delta S}(\chi, \mu^*_P(\chi))} \sum_{A' \in \mathcal{F}^K} P[A'|A]V(A', S + \Delta S), \\
\quad \chi \in \mathcal{F}^K \times \mathcal{S}^K, 
\]

where \( \mu^*_P \) is obtained from (12).

**Remark 1 (Balance between Current Transmission Cost and Future Average Transmission Cost):** From (12), note that the current transmission cost \( \phi \left( \sum_{f \in \mathcal{F}} (R_f + P_f) \right) \) increases with \( P \) and the future average transmission cost \( W(\chi, P) \) decreases with \( P \). Thus, we conclude that the optimal pushing policy \( \mu^*_P \) achieves the perfect balance between the current transmission cost and the future average transmission cost for any system state \( \chi \). On the other hand, from (13), we learn that the optimal caching policy \( \mu^*_C \) achieves the lowest future average transmission cost under the optimal pushing policy \( \mu^*_P \).

From Lemma 1 and Corollary 1, we note that \( \mu^* \) depends on system state \( \chi \) via the value function \( V(\cdot) \). Obtaining \( V(\cdot) \) involves solving the equivalent Bellman equation for all \( \chi \), and there is generally no closed-form solution offering any design insights \( \mathcal{S}_f \). In addition, obtaining numerical solutions through brute-force search such as value iteration and policy iteration is usually infeasible for practical implementation, due to the curse of dimensionality \( \mathcal{S}_f \). Therefore, it is desirable to study optimality properties of \( \mu^* \) and exploit these properties to design low-complexity near-optimal policies.

B. Optimality Properties

First, we analyze the impact of cache size \( C \) on the optimal average transmission cost. For ease of exposition, we rewrite the optimal average transmission cost \( \theta \) as a function of cache size \( C \), i.e., \( \theta(C) \). Based on the coupling and interchange arguments, we obtain the following lemma.

**Lemma 2 (Impact of Cache Size):** \( \theta(C) \) decreases with cache size \( C \).

**Remark 2 (Tradeoff between Cache Size and Bandwidth Utilization):** As illustrated in Fig. 1 a lower average transmission cost always corresponds to a higher bandwidth utilization. Hence, Lemma 2 reveals the tradeoff between the cache size and the bandwidth utilization.

Next, by analyzing the partial monotonicity of value function \( V(\cdot) \), we obtain the following lemma.

**Lemma 3 (Transient System States):** Any \( \chi = (A, S) \) satisfying \( S \notin \mathcal{S}^K \), where \( \mathcal{S} \equiv \{(S_f)_{f \in \mathcal{F}} : \sum_{f=1}^K S_f = C\} \), is transient under \( \mu^* \).

**Remark 3 (Reduction of System State Space):** Lemma 3 reveals that the optimal policy will make full use of the available storage resource. Also, considering the expected sum cost over the infinite horizon incurred by a transient state is finite and negligible w.r.t. the average transmission cost, we shall restrict our attention to the reduced system state space \( \mathcal{F}^K \times \mathcal{S}^K \) without loss of optimality.

V. LOW COMPLEXITY DECENTRALIZED POLICY

To obtain the optimal policy \( \mu^* \) from (11) under the reduced system state space given in Lemma 3 we need to compute \( \{V(\chi) : \chi \in \mathcal{F}^K \times \mathcal{S}^K \} \) by solving \( (F+1)^K \) equations, the number of which increases exponentially w.r.t. the number of users \( K \) and combinatorially w.r.t. the number of files \( F \) as well as the cache size \( C \). However, in practice, \( K, F \) and \( C \) are relatively large, and hence solving \( \mu^* \) suffers from the curse of dimensionality. Besides, the implementation of \( \mu^* \) requires a centralized controller and system state information, resulting in large signaling overhead. Therefore, it is of significant importance to develop a low-complexity decentralized policy for practical implementation.

A. Value Approximation

To alleviate the curse of dimensionality, we approximate the value function \( V(\cdot) \) in (10) as follows:

\[
V(\chi) \approx \hat{V}(\chi) = \sum_{k \in K} \sum_{f \in \mathcal{F}} \hat{V}_k^1(A_k, f), 
\]

(14)
where for all \( k \in K, \hat{V}_k^i(i, f) \), \( i \in \bar{F}, f \in F \), satisfy:

\[
\begin{align*}
\theta_k(1) + V_k^i(i, f) &= \frac{1}{K} \left( \frac{\phi(1)}{C} - \phi(1(A_k = f)) \right) \\
+ \min_{f' \in (s, f)} \sum_{j \in F} q_{i,j}^{(k)} \hat{V}_k^j(j, f') \quad i \in \bar{F}, f \in F.
\end{align*}
\]

(15)

In the following, we demonstrate the performance of the value approximation from the perspective of the average transmission cost and that of the complexity reduction, respectively. First, by analyzing structural properties of Problem 1 we obtain the following lemma.

**Lemma 4:** We have \( \theta(C) \geq C \sum_{k \in K} \theta_k(1) \).

Next, from (15), we note that the computation of \( \{\hat{V}_k^i(i, f) : i \in \bar{F}, f \in F\} \) is at per-user and per-file level, requiring to solve a system of \( KF(F + 1) \) equations. Compared with the computation of \( \{V(\chi) : \chi \in \mathcal{F}_K \times \mathcal{S}_K\} \) which requires to solve a system of \( (F + 1)^K \) equations, the value approximation in (14) eliminates both the exponential and combinatorial computational complexity.

Substituting (14) into (11), we have:

**Problem 2 (Approximate Joint Pushing and Caching Optimization):** For any system state \( \chi \in \mathcal{F}_K \times \mathcal{S}_K \), we have:

\[
\hat{\phi}^*(\chi) \equiv \min_{(P, \Delta S)} \left\{ \phi \left( \sum_{f \in F} (R_f + \hat{P}_f) \right) \right\}
\]

\[
+ \min_{k \in K, f, S_k, f + \Delta S_k, f + 1} \left\{ \sum_{j \in F} g_k(A_k, f) \right\}
\]

\[
st. \quad (2), (4), (7), (8), (9).
\]

where \( R_f \) is given by (2), and \( g_k(i, f) \equiv \sum_{j \in F} q_{i,j}^{(k)} \hat{V}_k^j(j, f) \).

Note that solving Problem 2 calls for combinatorial complexity of \( \mathcal{O}(\mathcal{P}_2) \) and centralized implementation with system state information. Thus, we are motivated to develop a low-complexity decentralized policy. For all \( k \in K \), by replacing \( P_f \) with \( P_k.f \) and adding constraint \( \hat{P}_f = P_k.f \), we obtain an equivalent problem to Problem 2. Then, after omitting the constraints \( P_f = P_k.f \) for all \( k \in K \), we attain a relaxed optimization problem to Problem 2, based on which a low-complexity decentralized policy is proposed.

Given \( \chi \in \mathcal{F}_K \times \mathcal{S}_K \), for all \( k \in K \), sort the elements in \( \mathcal{G}_K(\chi) \equiv \{g_k(A_k, f) : S_k, f + R_f = 0, f \in F\} \) in ascending order, where \( R_f \) is given by (2), let \( f_{k,i} \) denote the index of the file with the \( i \)-th minimum in \( \mathcal{G}_K(\chi) \) and define:

\[
y_k.f(p) \equiv \begin{cases} 1, & f = f_{k,i}, \quad i \leq p, \\ 0, & \text{otherwise}, \\ f \in F, \quad p \in \{0, 1, \ldots, |\mathcal{G}_K(\chi)|\}. \end{cases}
\]

(16)

In addition, denote with \( \chi_k \equiv (A_k, S_k) \in \bar{F} \times \bar{S} \) the state of user \( k \), \( T \equiv (T_f)_{f \in F} \) the transmission action and \( \hat{U}_{\Delta S}(\chi_k, T) \equiv \{(\Delta S_k, f) : f \in F, -S_k.f \leq \Delta S_k, f \leq T_f, f + \Delta S_k, f \in \{0, 1\} \}\) the cache action space for user \( k \) and transmission action \( T \). Then, we obtain the following low-complexity decentralized policy denoted as \( \hat{\mu}^* \equiv (\hat{\mu}_T^*, \hat{\mu}_{\Delta S}) \).

**Lemma 5 (Low-Complexity Decentralized Joint Pushing and Caching):** For any \( \chi \in \mathcal{F}_K \times \mathcal{S}_K \), we have \( \hat{\mu}_T^*(\chi) \equiv (\hat{P}_f^*)_{f \in F} \) with \( \hat{P}_f^* = \max_{k \in K} y_k.f(p_k^*) \), where \( y_k.f(p_k^*) \) is given by (16) and \( p_k^* \equiv \arg \min \{p : \sum_{f \in F} R_f + p + \min_{\Delta S_k \in \mathcal{G}_K(\chi_k, T) \times \bar{S}} \Delta S_k \leq g_k(A_k, f). \}

In addition, we have \( \hat{\mu}_{\Delta S}^*(\chi) \equiv (\Delta S_k^*)_{k \in K} \) with \( \Delta S_k^* = \arg \min_{\Delta S_k \in \mathcal{G}_K(\chi_k, T) \times \bar{S}} \Delta S_k \).

From Lemma 5, given \( \{V_k^i(i, f) : i \in \bar{F}, f \in F\} \), \( k \in K \), the complexity of the proposed policy is \( \mathcal{O}(KF\log(F)) \), which is much lower than that of the conventional numerical solutions. Furthermore, we note that it can be implemented in a decentralized manner. In particular, firstly, each user submits its request which could not be served by its local cache. Then the server broadcasts the corresponding file indexes \( \{f : \min_{k \in K} A_k = f, S_k, f = 1\} \), based on which each user constructs \( R \). Next, based on \( \chi_k \) and \( R \), user \( k \) computes \( (y_k.f(p^*_k))_{f \in F} \) and sends it to the server. Finally, the server obtains \( \hat{P}_f^* \) and transmits the files in \( \{f \in F : R_f + \hat{P}_f^* \geq 1\} \), based on which user \( k \) obtains \( \Delta S_k^* \).

Next, we characterize the relationship between the optimal value of Problem 2, i.e., \( \phi^*(\chi) \), and the value under \( \hat{\mu}^* \).

**Lemma 6:** For any \( \chi \in \mathcal{F}_K \times \mathcal{S}_K \), we have:

\[
\frac{1}{K} \sum_{k \in K} \phi \left( \sum_{f \in F} R_f + y_k.f(p_k^*) \right) \leq \hat{\phi}^*(\chi) \leq \phi \left( \sum_{f \in F} R_f + \hat{P}_f^* \right) + \sum_{k \in K} \sum_{f \in F} S_k, f + \Delta S_k, f \geq 1 \cdot g_k(A_k, f), \quad \text{where} \quad \Delta S_k^* = \min_{\Delta S_k \in \mathcal{G}_K(\chi_k, T) \times \bar{S}} \Delta S_k.
\]

VI. CONCLUSIONS

In this paper, we formulate the bandwidth utilization maximization via joint pushing and caching as an infinite horizon average cost MDP. By structural analysis, we show how the optimal policy balances the current transmission cost with the future average transmission cost. In addition, we show that the optimal policy achieves a tradeoff between the cache size and the bandwidth utilization. The optimal policy is of high computational and implementational complexity. By a linear approximation of the value function and relaxation techniques, we develop a decentralized policy with polynomial complexity, which provides useful guidelines for network design.

REFERENCES

[1] M. Gregori, J. Gmez-Vilardeb, J. Matamoros, and D. Gndz, “Wireless content caching for small cell and d2d networks,” IEEE J. Select. Areas Commun., vol. 34, no. 5, pp. 1222–1234, May 2016.
[2] S. Gittenis and N. Bambos, “Joint transmitter power control and mobile cache management in wireless computing,” IEEE Trans. Mobile Computing, vol. 7, no. 4, pp. 498–512, April 2008.
[3] L. Huang, “System intelligence: Model, bounds and algorithms,” in Proc. 17th ACM MobiHoc, New York, NY, USA, pp. 171–180, 2016.
[4] H. Feng, Z. Chen and H. Liu, “Performance Analysis of Push-Based Converged Networks With Limited Storage,” IEEE Trans. Commun., vol. 15, no. 12, pp. 8154-8168, Dec. 2016.
[5] W. Chen and H. V. Poor, “Content pushing with request delay information,” IEEE Trans. Commun., vol. PP, no. 99, pp. 1–1, 2017.
[6] J. Tadrous and A. Eryilmaz, “On Optimal Proactive Caching for Mobile Networks With Demand Uncertainties,” IEEE Trans. Networking, vol. 24, no. 5, pp. 2715-2727, 2016.
[7] D. P. Bertsekas, “Dynamic programming and optimal control 3rd edition, volume II,” Belmont, MA: Athena Scientific, 2011.