An Optimal Peak Hour Content Server Cache Update Scheduling Algorithm for 5G HetNets

Manyou Ma and Vincent W.S. Wong
Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, Canada
email: {manyoum, vincentw}@ece.ubc.ca

***Abstract*—Most of the existing caching schemes assume that the pushing of popular contents from the macro base station (MBS) to content servers (CSs) is performed during off-peak hours when the network traffic is low. However, since popular files, such as breaking news, may also be generated during peak hours, performing CS content update during peak hours is necessary. In this paper, we propose an optimal cache content update scheduling algorithm for heterogeneous networks (HetNets). The decision-making module is located in the MBS. The action set includes performing CS content update, letting the CSs simultaneously serve user requests, and using the MBS to directly serve user requests. The MBS aims to maximize the total throughput of the system within the duration of the peak hour under the uncertainty of the arrival of new user requests and the addition of new files. We formulate the peak hour CS cache content update scheduling problem as a Markov decision process and propose an optimal cache content update scheduling algorithm based on dynamic programming. We perform simulations and compare our proposed optimal scheduling algorithm with the periodic update and greedy scheduling heuristics. Simulation results show that our proposed algorithm outperforms those two heuristics under different scenarios.**

I. Introduction

The fifth generation (5G) wireless communication networks are expected to face a significant increase in data traffic compared to the volume in the current networks [1]. To address this challenge, caching in heterogeneous networks (HetNets) [2] has been proposed. Under the HetNet architecture, a macrocell consists one macro base station (MBS) and multiple small-cell base stations with storage capacity acting as content servers (CSs). During off-peak hours, when the data traffic is low, the MBS can push popular contents to the CSs via a wireless backhaul. During peak hours, when the network traffic is high, multiple CSs can simultaneously serve user requests with lower power consumption and higher throughput.

Previous research has studied different aspects of HetNet caching. For instance, to predict future file popularity, Blasco and Gunduz in [3] introduced a method based on the multi-armed bandit model, and in [4], Hou *et al.* applied a transfer learning-based approach. To achieve optimal cache content placement throughout the CSs in the network, Sadeghi *et al.* in [5] introduced a reinforcement learning-based algorithm for the CS to choose a set of files to cache based on the file popularity time-series. Maddah-Ali and Niesen in [6] studied the optimal cache placement problem to reduce the backhaul rate during peak hours. To optimize the content delivery phase, Zhou *et al.* in [7] studied the optimal scheduling and power allocation problem. The aforementioned works assume that CS content placement and update can be performed during off-peak hours, when the data traffic in the network is low and the cost related to updating the CSs is negligible. In [8], Ding *et al.* advocated the need for performing peak hour caching, where CS cache contents are updated during peak hours, because popular files, such as breaking news, are inevitably generated during peak hours. Thus, spectral resources need to be allocated periodically for CS content update. However, during peak hours, when the user request rate is high, efficient use of spectral resources is crucial for maximizing the system throughput. The periodic CS content update with a fixed period proposed in [8] may not be the optimal solution, and the joint design of cache content updating and user-request scheduling is necessary. In [9], Wang *et al.* studied the joint design of user scheduling and content caching in a HetNet using the CS as a relay. However, their solution assumed service of the current user request finishes before the next request arrives, and hence no buffering is required. Since many user requests may be dropped without having queues or buffers to store incoming user requests when the system user request rate is high, their algorithm may not be adequate for peak hour scheduling.

A peak hour CS content update strategy should consider real-time network status, such as the number of user requests in the network buffer, to make scheduling decisions. As discussed in [7], the challenge of scheduling in HetNets is due to the heterogeneous structure of the network, which involves choosing a coupled action between the MBS and CSs. In the peak hour CS content update problem, another challenge is to balance between the improvement in throughput in future time slots, brought by pushing the new file to the CSs, and the decrease in throughput in the current time slot, since no user request is being served. To tackle these challenges, in this paper, we propose an optimal peak hour CS content update scheduling algorithm that aims to maximize the total throughput of the system during peak hours based on an Markov decision process (MDP) framework. The contributions of our work are as follows:

- We formulate the peak hour CS content update problem as a finite horizon MDP. We introduce multiple queues at the CSs and the MBS to track the user requests that can be handled either by CSs or MBS. In this model, the arrival rates of different queues are dependent on the scheduling decisions, and hence are not constant. Therefore, it cannot

be solved by conventional tools from queuing theory.
- We obtain the optimal scheduling policy for the MDP problem and propose an optimal peak hour CS content update algorithm based on dynamic programming.
- We perform simulation studies and show that our proposed algorithm achieves a higher total throughput compared to two heuristics under different scenarios.

The rest of this paper is organized as follows. The system model is presented in Section II. An MDP-based problem formulation for finding the optimal CS content update strategy is described in Section III. The dynamic programming-based solution and optimal scheduling algorithm are presented in Section IV. Performance evaluations are presented in Section V, and Section VI concludes the paper.

II. System Model

Consider a network topology with one MBS and N small cells, where each small cell $n \in \{1, \ldots, N\}$ has one CS and U_n users. The MBS is connected to the core network via a high-speed backhaul wired link. The CSs are connected to the MBS via a wireless backhaul. Similar to [7], [8], we assume that the files in the system have the same size, and the transmission of one file can be finished within one time slot[1]. During off-peak hours or specified periods within the peak hours, those files which are likely to become popular in future time slots are pushed to the CSs using multicasting through the wireless backhaul. We assume error-free transmission can be achieved, hence each CS has an identical set of cached files. We adopt the HetNet model in [7], where the CSs have disjoint coverage areas. Hence, they can transmit simultaneously during a time slot with appropriate frequency reuse schemes. However, since the MBS's coverage area overlaps with those of the CSs, the MBS and CSs cannot transmit simultaneously without causing interference. At the beginning of a time slot, the decision making module in the MBS makes a decision by choosing one of the following actions: (a) stay idle, (b) MBS performs CS content update via multicasting, (c) all N CSs simultaneously serve user requests, and (d) MBS serves a user request. A sample network topology with $N = 3$, $U_1 = 2$, $U_2 = 4$, and $U_3 = 3$ is shown in Fig. 1. The four aforementioned actions are also depicted.

We use the following scheme to model the dynamics of file popularity when a new file is generated. Before a new file is generated, suppose that the files requested by users in the HetNet belong to a finite set $\mathcal{F} = \{f_1, f_2, \ldots, f_F\}$. The probability that a user request corresponds to file f_i is denoted by $P(f_i)$, where $\sum_{f_i \in \mathcal{F}} P(f_i) = 1$. We also call $P(f_i)$ the popularity of file f_i. We introduce set $\mathcal{X} = \{f_1, f_2, \ldots, f_X\}$, where $X \leq F$, to denote the set of files that are cached at the CSs. Let C^l denote the probability that a user request corresponds to a file that is cached in the CS. We have

$$C^l = \sum_{f_i \in \mathcal{X}} P(f_i). \tag{1}$$

[1]This assumption can be relaxed by dividing a large file into multiple sub-files, where each sub-file can be sent in one time slot.

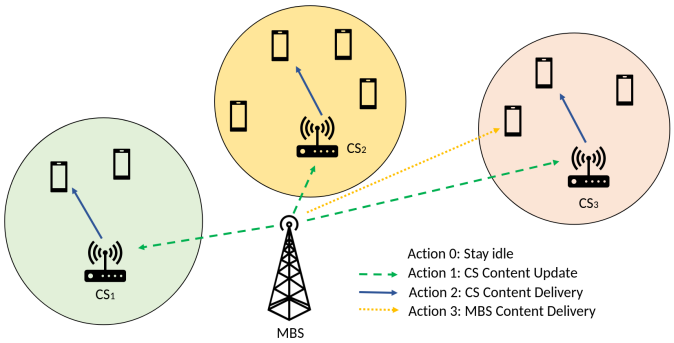


Fig. 1: An example of a HetNet with one MBS and three CSs. The four sample actions of the MBS are also shown. The green dashed lines represent the MBS updates the CSs via multicasting. The blue solid lines represent the CSs simultaneously serve user requests. The yellow dotted line represents the MBS directly serves a user request.

This quantity is also called the *cache hit probability* in the literature [8], which is the portion of user requests that can be handled by CSs locally. After a new file $f_g \notin \mathcal{F}$ is generated and has popularity $G = P_{\text{new}}(f_g)$, the set $\mathcal{F}_{\text{new}} = \mathcal{F} \bigcup \{f_g\}$ represents the complete set of files requested by users. After f_g is generated, we set the popularity of f_i as $P_{\text{new}}(f_i) = P(f_i)(1-G), \forall\, f_i \in \mathcal{F}$, so that $\sum_{f_i \in \mathcal{F}_{\text{new}}} P(f_i) = \sum_{i=1}^{F} P_{\text{new}}(f_i) + G = 1$. Therefore, the cache hit probability of the HetNet before file f_g is being pushed to the CSs is

$$C^l_{\text{before}} = \frac{\sum_{f_i \in \mathcal{X}} P_{\text{new}}(f_i)}{\sum_{i=1}^{F} P_{\text{new}}(f_i) + G} = C^l(1-G). \tag{2}$$

After file f_g is pushed to the CSs, the set of cached files becomes $\mathcal{X}_{\text{new}} = \mathcal{X} \bigcup \{f_g\}$. The cache hit probability becomes

$$C^l_{\text{after}} = \frac{\sum_{f_i \in \mathcal{X}_{\text{new}}} P_{\text{new}}(f_i)}{\sum_{i=1}^{F} P_{\text{new}}(f_i) + G} = C^l_{\text{before}} + G. \tag{3}$$

We ensure that the CSs have space to store the newly generated file, by letting the CSs to set aside enough storage space to accommodate the maximum possible amount of newly generated files during the finite duration of the peak hour. In the following sections, we will use equations (2) and (3) to model the dynamics of the cache hit probability in a HetNet[2].

III. MDP Problem Formulation

In this section, we formulate the peak hour CS content update scheduling problem as a finite-horizon MDP. We define the decision epochs, states, actions, state transition probabilities, and the rewards corresponding to the MDP problem. In each decision epoch, the MBS makes a decision based on its current state, with the goal of maximizing the total throughput for the finite duration of the network's peak hour.

A. Decision Epochs and States

1) Decision Epochs: We use $\mathcal{T} = \{0, 1, 2, \ldots, T-1\}$ to represent the set of decision epochs of the system. In each decision epoch, each user requests one of the available files

[2]We acknowledge this only models a simplified version of file popularity dynamics. In practice, more realistic file popularity models can also be used in place of (2) and (3) without modifications to other parts of our algorithm.

in the system independently, following a Bernoulli random process with rate λ_u. New popular files are also generated following a Bernoulli random process with rate λ_f.

2) File Popularity: We use a state variable C_t^l to denote the cache hit probability in decision epoch $t \in \mathcal{T}$. We discretize C_t^l into N_c levels, as multiples of $1/N_c$, to achieve a discrete state space, where $C_t^l \in \mathcal{C} = \{0, 1/N_c, 2/N_c, \ldots, 1\}$. In decision epoch t, let G_t denote the popularity of a newly generated file, which will be added into the system in decision epoch $t+1$. Let $\mathcal{L} = \{0, L_1, L_2, \ldots, L_{N_\mathrm{f}}\}$ denote the set of possible popularity of new files in the system. $G_t = 0$ corresponds to the case where the newly generated file will not be requested by any user. Therefore, we also use $G_t = 0$ to represent the case where no new file has been generated since the last content update. For example, the set $\mathcal{L} = \{0, 0.1, 0.2\}$ represents that the newly generated file in the system can have a popularity of either 0.1 or 0.2. Since the arrival process of newly generated files follows a Bernoulli process with rate λ_f, G_t is independent from other state variables and has the probability distribution

$$\mathbb{P}(G_t) = (1-\lambda_f)\mathbf{I}(G_t = 0) + \frac{\lambda_f}{N_\mathrm{f}}\mathbf{I}(G_t \in \mathcal{L} \setminus \{0\}), \quad (4)$$

where $\mathbf{I}(\cdot)$ denotes the indicator function. Let $C_t^g \in \mathcal{L}$ denote the popularity of the latest generated uncached popular file in decision epoch t.

3) Queues: For each CS, we introduce a request queue to store user requests that can be served by that CS locally. We also introduce a queue for the MBS to store the user requests that must be served by the MBS directly. Therefore, we have $N_q = N+1$ queues in total to track user requests. Let $Q_{n,t}^{\mathrm{local}} \in \mathcal{Q} = \{0, 1, \ldots, Q_{\max}\}$ denote the queue length corresponding to the n-th CS in decision epoch t. The user requests are stored in a buffer with finite size $Q_{\max}$. When the finite buffer is full, the newly arrived user request will be dropped. Let $Y_{n,t}^{\mathrm{local}} \in \mathcal{Y}_n = \{0, 1, \ldots, U_n\}$ denote the number of new user requests for the n-th CS, which is added into $Q_{n,t}^{\mathrm{local}}$ at the beginning of decision epoch t. Let $Q_t^{\mathrm{MBS}} \in \mathcal{Q}$ denote the queue status that corresponds to the number of user requests of contents which have not been cached by the CSs in decision epoch t, hence need to be served by the MBS directly. Let $Y_t^{\mathrm{MBS}} \in \mathcal{Y} = \{0, 1, \ldots, U\}$ denote the number of new user requests for Q_t^{MBS}, where $U = \sum_{n=1}^{N} U_n$ and it represents the total number of users in the system.

4) Overall system state: The system state space is the countable set $\mathcal{S} = \mathcal{C} \times \mathcal{L}^2 \times \mathcal{Q}^{N+1} \times \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_N \times \mathcal{Y}$. The state vector $\mathbf{s}_t$ represents the overall state of the system in decision epoch t, where

$$\begin{aligned}\mathbf{s}_t = \big(C_t^l, G_t, C_t^g,\, Q_{1,t}^{\mathrm{local}}, \ldots,\, Q_{N,t}^{\mathrm{local}},\, Q_t^{\mathrm{MBS}},\\ Y_{1,t}^{\mathrm{local}}, \ldots,\, Y_{N,t}^{\mathrm{local}},\, Y_t^{\mathrm{MBS}}\big),\ \forall\, t \in \mathcal{T}.\end{aligned} \quad (5)$$

B. Actions and State Transition Probability

1) Actions: Given the state vector $\mathbf{s}_t$, the decision making module at the MBS chooses an action $a_t \in \mathcal{A}_t(\mathbf{s}_t)$, where $\mathcal{A}_t(\mathbf{s}_t)$ represents the available action set in decision epoch t. We have $\mathcal{A}_t(\mathbf{s}_t) \subseteq \mathcal{A} = \{0, 1, 2, 3\}$, for $t \in \mathcal{T}$, where the MBS stays idle when $a_t = 0$, updates the newly generated file to the CSs when $a_t = 1$, schedules the CSs to serve the user requests when $a_t = 2$, and serves a user request from Q_t^{MBS} when $a_t = 3$. The available action set in decision epoch t depends on $\mathbf{s}_t$, where action $1 \in \mathcal{A}_t(\mathbf{s}_t)$ if $C_t^g > 0$, action $2 \in \mathcal{A}_t(\mathbf{s}_t)$ if $\sum_{n=1}^{N} Q_{n,t}^{\mathrm{local}} > 0$, and action $3 \in \mathcal{A}_t(\mathbf{s}_t)$ if $Q_t^{\mathrm{MBS}} > 0$.

2) State Transition of $Y_{n,t}^{\mathrm{local}}$ and Y_t^{MBS}: We use $\mathcal{B}(\cdot)$ to denote the probability mass function of the binomial distribution

$$\mathcal{B}(K, i, \lambda) = \binom{K}{i}\lambda^i(1-\lambda)^{(K-i)}, \quad i = 0, 1, \ldots, K. \quad (6)$$

Since C_t^l corresponds to the portion of user requests that can be served by the CSs locally in decision epoch t, the number of new user requests for the n-th CS $Y_{n,t}^{\mathrm{local}}$ follows

$$\mathbb{P}(Y_{n,t+1}^{\mathrm{local}} \mid C_t^l) = \mathcal{B}(U_n, Y_{n,t}^{\mathrm{local}}, \lambda_u C_t^l)\mathbf{I}(Y_{n,t}^{\mathrm{local}} \in \mathcal{Y}_n). \quad (7)$$

Similarly, the number of new user requests for the MBS, Y_t^{MBS}, can be expressed as

$$\mathbb{P}(Y_{t+1}^{\mathrm{MBS}} \mid C_t^l) = \mathcal{B}(U, Y_t^{\mathrm{MBS}}, \lambda_u(1-C_t^l))\mathbf{I}(Y_t^{\mathrm{MBS}} \in \mathcal{Y}). \quad (8)$$

3) State Transition of C_t^g: Given state variables G_t, C_t^g and action a_t, the popularity of the most recently generated uncached file in decision epoch $t+1$ is deterministic. That is,

$$\begin{aligned}\mathbb{P}(C_{t+1}^g \mid \mathbf{s}_t, a_t) = {}& \mathbf{I}(a_t \neq 1)\mathbf{I}(G_t = 0)\mathbf{I}(C_{t+1}^g = C_t^g)\\ &+ \mathbf{I}(a_t = 1)\mathbf{I}(G_t = 0)\mathbf{I}(C_{t+1}^g = 0)\\ &+ \mathbf{I}(G_t > 0)\mathbf{I}(C_{t+1}^g = G_t).\end{aligned} \quad (9)$$

The first term considers the case when CS content update was not performed in decision epoch t, and no new file is generated in decision epoch $t+1$. The second term considers the case when CS content update was performed in decision epoch t and no new file is generated in decision epoch $t+1$. The third term considers the case when a new file is generated in decision epoch $t+1$, and becomes the most recently generated popular file that has not been cached.

4) State Transition of C_t^l: Given state variables G_t, C_t^l, C_t^g, and action a_t, the cache hit probability in decision epoch $t+1$ is a deterministic value. That is,

$$\begin{aligned}\mathbb{P}(C_{t+1}^l \mid \mathbf{s}_t, a_t) = {}& \mathbf{I}(a_t \neq 1)\mathbf{I}(G_t = 0)\mathbf{I}(C_{t+1}^l = C_t^l)\\ &+ \mathbf{I}(a_t \neq 1)\mathbf{I}(G_t > 0)\mathbf{I}\left(C_{t+1}^l = C_t^l(1-G_t)\right)\\ &+ \mathbf{I}(a_t = 1)\mathbf{I}(G_t = 0)\mathbf{I}\left(C_{t+1}^l = C_t^l + C_t^g\right)\\ &+ \mathbf{I}(a_t = 1)\mathbf{I}(G_t > 0)\mathbf{I}\left(C_{t+1}^l = (C_t^l + C_t^g)(1-G_t)\right).\end{aligned}$$

The first term considers the case when CS content update was not performed in decision epoch t and no new content is generated in decision epoch $t+1$. Hence, the cache hit probability remains the same in decision epoch t. The second term considers the case when CS content update was not performed in decision epoch t and a new file with popularity G_t is generated, which corresponds to equation (2). The third term considers the case when CS content update was performed in decision epoch t and no new file is generated,

which corresponds to equation (3). The last term considers the case when CS content update was performed in decision epoch t and a new file is generated in decision epoch $t+1$.

5) Queue Dynamics: In decision epoch $t+1$, the newly arrived user request $Y_{n,t}^{\text{local}}$ is first added to the request queue at the n-th CS. If the action $a_t = 2$, then one user request is served and the queue length is reduced by one. We have

$$\begin{aligned}\mathbb{P}(Q_{n,t+1}^{\text{local}} \mid \mathbf{s}_t, a_t) = \mathbf{I}\big(Q_{n,t+1}^{\text{local}} = \min\big(Q_{n,t}^{\text{local}} \\ + Y_{n,t}^{\text{local}}, Q_{\max}\big) - \mathbf{I}(a_t = 2)\big).\end{aligned} \tag{10}$$

Similarly, for the user request queue corresponding to the MBS, we have

$$\begin{aligned}\mathbb{P}(Q_{t+1}^{\text{MBS}} \mid \mathbf{s}_t, a_t) = \mathbf{I}\big(Q_{t+1}^{\text{MBS}} = \min\big(Q_{t}^{\text{MBS}} \\ + Y_{t}^{\text{MBS}},\ Q_{\max}\big) - \mathbf{I}(a_t = 3)\big).\end{aligned} \tag{11}$$

6) Overall State Transition Probability: Given the current state vector $\mathbf{s}_t$ and action a_t, the state transition probability to the next state $\mathbf{s}_{t+1}$ can be written as

$$\begin{aligned}\mathbb{P}(\mathbf{s}_{t+1} \mid \mathbf{s}_t, a_t) &= \mathbb{P}(C_{t+1}^{l} \mid \mathbf{s}_t, a_t)\mathbb{P}(C_{t+1}^{g} \mid \mathbf{s}_t, a_t) \\ &\times \prod_{n=1}^{N} \mathbb{P}(Q_{n,t+1}^{\text{local}} \mid \mathbf{s}_t, a_t)\mathbb{P}(Q_{t+1}^{\text{MBS}} \mid \mathbf{s}_t, a_t) \\ &\times \mathbb{P}(G_{t+1}) \prod_{n=1}^{N} \mathbb{P}(Y_{n,t+1}^{\text{local}} \mid C_t^{l})\mathbb{P}(Y_{t+1}^{\text{MBS}} \mid C_t^{l}).\end{aligned} \tag{12}$$

C. Rewards and Optimal Policy

We consider the problem of maximizing the total throughput during peak hours, where the throughput in each decision epoch is used as the reward. This can be expressed as

$$r(\mathbf{s}_t, a_t) = \begin{cases} 0, & \text{if } a_t = 0 \text{ or } 1 \\ \sum_{n=1}^{N} \mathbf{I}(Q_{n,t}^{\text{local}} > 0) & \text{if } a_t = 2 \\ 1, & \text{if } a_t = 3. \end{cases} \tag{13}$$

In decision epoch t, the throughput of the network is zero when the MBS stays idle or pushes a new file to the CSs. When action $a_t = 2$, the CSs are scheduled to simultaneously serve user requests, therefore the throughput corresponds to the total number of CSs which have non-empty user request queues. When action $a_t = 3$, the MBS serves one user request, therefore the throughput is equal to one file per time slot.

Let $\pi = \{a_t(\mathbf{s}_t), \forall\ \mathbf{s}_t \in \mathcal{S}, t \in \mathcal{T}\}$ denote a scheduling policy for the formulated MDP problem, where $a_t(\mathbf{s}_t)$ is the scheduling decision given the current state $\mathbf{s}_t$. Let Π denote the set of feasible policies, where $a_t(\mathbf{s}_t) \in \mathcal{A}_t(\mathbf{s}_t)$, for all $t \in \mathcal{T}$. Since G_t, $Y_{1,t}^{\text{local}}$, …, $Y_{N,t}^{\text{local}}$, and Y_t^{MBS} are random variables, we aim to determine the optimal scheduling policy that maximizes the expected total throughput of the HetNet over a finite horizon from $t = 0$ to $T-1$. The optimal policy π can be determined by solving the following optimization problem

$$\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}\left\{ \sum_{t=0}^{T-1} r(\mathbf{s}_t, a_t(\mathbf{s}_t)) \mid \pi \right\}. \tag{14}$$

IV. Finite Horizon Dynamic Programming

In this section, we present an algorithm which solves the throughput maximization problem (14) by using backward induction and dynamic programming [10]. Let $J_t(\mathbf{s}), \forall\ t \in \mathcal{T}$, denote the functions representing the maximum expected total throughput between decision epochs t and $T-1$, when the current system state is $\mathbf{s}$, where $\mathbf{s} \in \mathcal{S}$. $J_t(\mathbf{s})$ can be determined by backward induction starting from $t = T-1$ and back to $t = 0$. At decision epoch $T-1$, we have

$$J_{T-1}(\mathbf{s}) = \max_{a_{T-1} \in \mathcal{A}(\mathbf{s})} r(\mathbf{s}, a_{T-1}). \tag{15}$$

For decision epochs $t = T-2, \ldots, 0$, we have

$$\begin{aligned}J_t(\mathbf{s}) = \max_{a_t \in \mathcal{A}(\mathbf{s})} \Big[r(\mathbf{s}, a_t) \\ + \sum_{\mathbf{s}_{t+1} \in \mathcal{S}} \mathbb{P}(\mathbf{s}_{t+1} \mid \mathbf{s}, a_t) J_{t+1}(\mathbf{s}_{t+1}) \Big],\end{aligned} \tag{16}$$

where the optimal policy of the MDP is $\pi^* = \{a_t^*(\mathbf{s}),\ \forall\ \mathbf{s} \in \mathcal{S}, t \in \mathcal{T}\}$, where

$$\begin{aligned}a_t^*(\mathbf{s}) = \arg\max_{a_t \in \mathcal{A}(\mathbf{s})} \Big[r(\mathbf{s}, a_t) \\ + \sum_{\mathbf{s}_{t+1} \in \mathcal{S}} \mathbb{P}(\mathbf{s}_{t+1} \mid \mathbf{s}, a_t) J_{t+1}(\mathbf{s}_{t+1}) \Big].\end{aligned} \tag{17}$$

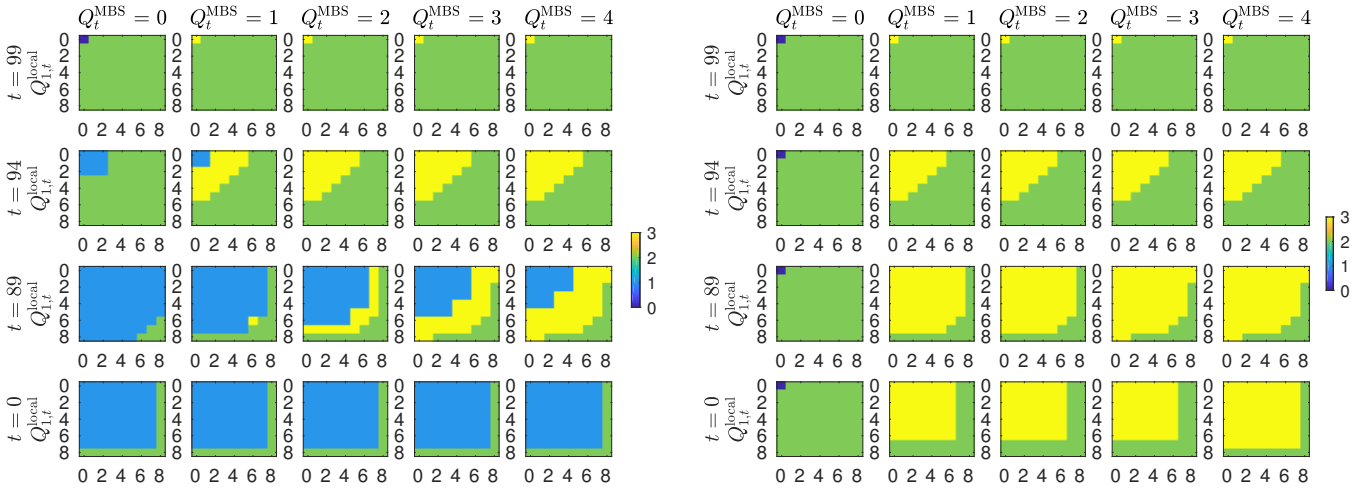
Using these optimality conditions, we have the following monotonicity result.

Theorem 1: The expected throughput between decision epochs t and $T-1$, $J_t(\mathbf{s}_t)$, is monotonically non-decreasing with respect to the buffer occupancy, Q_t^{MBS}, $Q_{1,t}^{\text{local}}$, …, $Q_{N,t}^{\text{local}}$.

Proof: (By induction) Consider state variable vectors $\mathbf{s}$ and $\mathbf{s}^+$, where $Q_n^{\text{local}} \le Q_n^{\text{local+}}$, for all $n \in \{1, \ldots, N\}$[3], $Q^{\text{MBS}} \le Q^{\text{MBS+}}$, and the other state variables being equal. We define this relation as $\mathbf{s}^+ \succcurlyeq \mathbf{s}$. From equation (13), we have $r(\mathbf{s}^+, a_t) \ge r(\mathbf{s}, a_t),\ \forall\ a_t,\ \mathbf{s}^+ \succcurlyeq \mathbf{s}$. Therefore, from equation (15), we have $J_{T-1}(\mathbf{s}^+) \ge J_{T-1}(\mathbf{s}), \forall\ \mathbf{s}^+ \succcurlyeq \mathbf{s}$. Now suppose $J_{t+1}(\mathbf{s}^+) \ge J_{t+1}(\mathbf{s}),\ \forall\ \mathbf{s}^+ \succcurlyeq \mathbf{s}$, we would like to show $J_t(\mathbf{s}^+) \ge J_t(\mathbf{s}), \forall\ \mathbf{s}^+ \succcurlyeq \mathbf{s}$. Suppose the optimal policy in decision epoch t is $a_t^*(\mathbf{s}), \forall\ \mathbf{s} \in \mathcal{S}$. We have

$$\begin{aligned}&J_t(\mathbf{s}) \\ &= r(\mathbf{s}, a_t^*(\mathbf{s})) + \sum_{\mathbf{s}_{t+1} \in \mathcal{S}} \mathbb{P}(\mathbf{s}_{t+1} \mid \mathbf{s}, a_t^*(\mathbf{s})) J_{t+1}(\mathbf{s}_{t+1}) \\ &\overset{(a)}{\le} r(\mathbf{s}^+, a_t^*(\mathbf{s})) \\ &\quad + \sum_{\mathbf{s}_{t+1} \in \mathcal{S}} \mathbb{P}(\mathbf{s}_{t+1} \mid \mathbf{s}, a_t^*(\mathbf{s})) J_{t+1}(\mathcal{P}_s(\mathbf{s}_{t+1},\ \mathbf{s}^+ - \mathbf{s})) \\ &= r(\mathbf{s}^+, a_t^*(\mathbf{s})) + \sum_{\mathbf{s}_{t+1}^+ \in \mathcal{S}} \mathbb{P}(\mathbf{s}_{t+1}^+ \mid \mathbf{s}^+, a_t^*(\mathbf{s})) J_{t+1}(\mathbf{s}_{t+1}^+) \\ &\overset{(b)}{\le} r(\mathbf{s}^+, a_t^*(\mathbf{s}^+)) + \sum_{\mathbf{s}_{t+1}^+ \in \mathcal{S}} \mathbb{P}(\mathbf{s}_{t+1}^+ \mid \mathbf{s}^+, a_t^*(\mathbf{s}^+)) J_{t+1}(\mathbf{s}_{t+1}^+) \\ &= J_t(\mathbf{s}^+).\end{aligned}$$

[3]For notation simplicity, we use Q_n^{localx} and Q^{MBSx} to represent the corresponding queue lengths of the state variable vector $\mathbf{s}^{\text{x}}$.



(a) $C_t^g = 1$ (b) $C_t^g = 0$

Fig. 2: Optimal actions in different decision epochs and combinations of user request queue lengths.

where we define $\mathcal{P}_s(\cdot) : \mathcal{S} \times \mathcal{S} \mapsto \mathcal{S}$, $\mathcal{P}_s(\mathbf{s}^1, \mathbf{s}^2) = \mathbf{s}^3$, with $Q_n^{\text{local3}} = \min(Q_{\max}, \ Q_n^{\text{local1}} + Q_n^{\text{local2}})$, $\forall \ n$, $Q^{\text{MBS3}} = \min(Q_{\max}, \ Q^{\text{MBS1}} + Q^{\text{MBS2}})$, and the other state variables being the same as $\mathbf{s}^1$. Since $\mathbf{s}^+ \succcurlyeq \mathbf{s}$, we have $\mathcal{P}_s(\mathbf{s}_{t+1}, \ \mathbf{s}^+ - \mathbf{s})) \succcurlyeq \mathbf{s}_{t+1}$. Equation (13) and the induction assumption give inequality (a). Inequality (b) is due to that $a_t^*(\mathbf{s}^+)$ is the optimal action. ■

Our proposed optimal cache content update scheduling algorithm is shown in Algorithm 1. In the planning phase, the MBS computes the optimal scheduling policies in different decision epochs and stores them in a look-up table. In the deployment phase, the MBS first uses the tracked and observed information to determine its current state and then finds the optimal scheduling policy from the look-up table. Based on its observations in decision epoch t, the MBS updates the new file generation and user request arrivals for decision epoch $t+1$. The algorithmic complexity is $O(T|\mathcal{S}||\mathcal{A}|^2)$ [11], where $|\cdot|$ represents the cardinality of a set.

Algorithm 1 Optimal Peak Hour Cache Content Update Scheduling Algorithm

1: Planning Phase:
2: Set $J_{T-1}(\mathbf{s}_{T-1})$, according to (15).
3: Set $t := T-2$.
4: **while** $t \geq 0$ **do**
5: Calculate $J_t(\mathbf{s}_t)$, $\forall \ \mathbf{s}_t \in \mathcal{S}$ according to (16).
6: Find the optimal action $a_t^*(\mathbf{s}_t)$, according to (17).
7: Set $t := t-1$.
8: **end while**
9: Deployment Phase:
10: Set $t := 0$
11: **while** $t \leq T-1$ **do**
12: Track the MBS queue length Q_t^{MBS}.
13: Track the new arrivals at the MBS Y_t^{MBS}.
14: **for** $n = 1$ to N **do**
15: Track $Q_{n,t}^{\text{local}}$ and $Y_{n,t}^{\text{local}}$.
16: **end for**
17: Track C_t^l, C_t^g, and G_t.
18: Set $\mathbf{s}_t := (C_t^l, G_t, C_t^g, Q_{1,t}^{\text{local}}, \ldots, Q_{N,t}^{\text{local}}, Q_t^{\text{MBS}}, Y_{1,t}^{\text{local}}, \ldots, Y_{N,t}^{\text{local}}, Y_t^{\text{MBS}})$.
19: Obtain a_t from the stored $a_t^*(\mathbf{s}_t)$ values.
20: Update the queue lengths based on the new arrivals and action a_t, according to (10) and (11).
21: Update $Y_{1,t}^{\text{local}}, \ldots, Y_{N,t}^{\text{local}}, Y_t^{\text{MBS}}$, and G_t based on observations in the current decision epoch.
22: Set $t := t+1$.
23: **end while**

V. Performance Evaluation

In this section, we evaluate the proposed optimal CS content update scheduling algorithm. Unless specified otherwise, we consider a HetNet with one MBS and three CSs. We set $\lambda_u = 0.4$ [requests/second] and $\lambda_f = 0.1$ [files/second]. The cache hit probability is discretized into $N_c = 20$ levels. There are two possible popularity levels of a newly generated file, i.e., $\mathcal{L} = \{0, 0.1, 0.3\}$. The total number of decision epochs $T = 100$. The length of a time slot is one second. The file size is 100 MB. The maximum queue length is eight for all queues. We compare the performance of our proposed algorithm with two heuristics, which use different scheduling policies instead of line 19 in Algorithm 1. For the periodic update heuristic algorithm with period $1/\lambda_f$, CS content update is performed once every $1/\lambda_f$ decision epochs. In other decision epochs, CS content update action is not allowed, and the action set is $\hat{\mathcal{A}}(\mathbf{s}_t) = \mathcal{A}(\mathbf{s}_t) \backslash \{1\}$. For the greedy algorithm, the scheduling policy aims to maximize the throughput only in the current decision epoch. In the case when there is only one CS that has user requests to serve and the MBS also has user requests to serve, the greedy algorithm randomly decides between letting the MBS or CS serve a user request.

In Fig. 2, the optimal policies for $C_t^l = 0.5$, $Q_{3,t}^{\text{local}} = 0$ and different combinations of Q_t^{MBS}, $Q_{1,t}^{\text{local}}$, and $Q_{2,t}^{\text{local}}$ are shown for different decision epochs. From Fig. 2(a), we observe that given the same combination of user request queue lengths, the optimal policy varies in different decision epochs. Consider the case when $Q_t^{\text{MBS}} = 1$ and $Q_{1,t}^{\text{local}} = Q_{2,t}^{\text{local}} = 3$. In decision epoch $t = 0$, the optimal decision policy is to update the

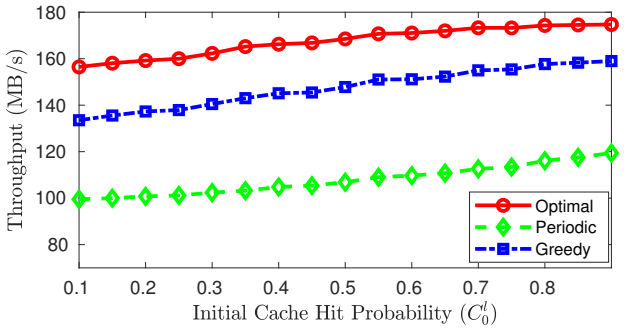


Fig. 3: Comparison of the total throughput of the proposed optimal algorithm vs. the heuristics under different initial cache hit probability.

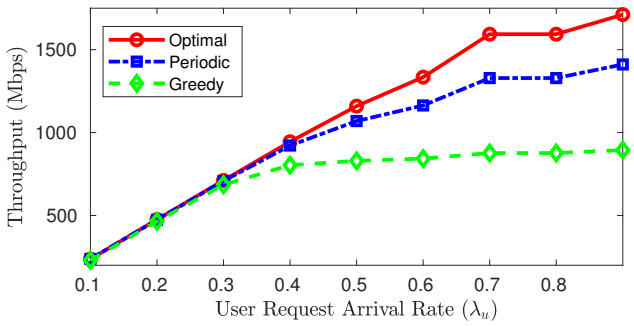


Fig. 4: Comparison of the total throughput of the proposed optimal algorithm vs. the heuristics under different user request arrival rate.

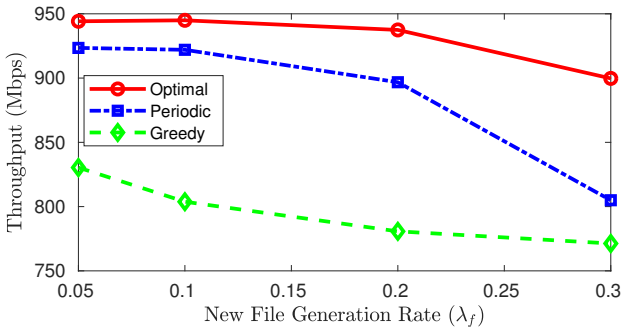


Fig. 5: Comparison of the total throughput of the proposed optimal algorithm vs. the heuristics under different of new file generation rate.

CS content, while in decision epoch $t = 99$, the optimal action is to serve a user request. This demonstrates the trade-off between throughput in the current decision epoch and the expected throughput in future decision epochs. Towards the end of the peak hour, the optimal policy maximizes the immediate reward, while at the beginning of the peak hour, the optimal policy prioritizes future expected throughput.

Fig. 3 shows the average total throughput of all 100 decision epochs against the initial cache hit probability C_0^l. The proposed optimal CS cache content update scheduling algorithm outperforms the two heuristics in all settings. The throughputs of all three algorithms become higher when C_0^l increases. When the value of C_0^l is small, the throughput of the HetNet at the beginning of the peak hour is lower, since more user requests need to be served by the MBS directly.

Fig. 4 shows the average total throughput of the three algorithms against the user request arrival rate λ_u. The proposed algorithm outperforms the two heuristics in all settings. The advantage is more apparent when the user request arrival rate is higher. This justifies our initial assumption that the higher λ_u is, the more critical the efficient use of spectral resources becomes. On the other hand, when λ_u is low and the MBS can handle all user requests on its own without the CSs, the throughputs of all three algorithms become the same.

Fig. 5 shows the average total throughput of the algorithms against the new file generation rate λ_f. We observe that the throughputs of all three algorithms become lower as the new file generation rate increases. This is because as more frequent cache updates are being scheduled, the total number of user requests being served is reduced.

VI. Conclusion

In this paper, we proposed an optimal cache content update scheduling algorithm for HetNets. We formulated the CS content update problem as an MDP problem and designed an algorithm that maximizes the total throughput under the uncertainty in the arrival of new user requests and file generations. The optimal scheduling policy was obtained using dynamic programming. Based on the structure of the optimal policy obtained in this paper, in our future work, we will apply threshold-based policy to reduce the state-space, hence reduce the amount of memory resource required to solve the MDP problem. We also plan to apply model-free techniques, such as deep reinforcement learning, to accommodate unknown system parameters, such as the user request arrival rates and future file popularities.

References

[1] V. W. S. Wong, R. Schober, D. W. K. Ng, and L. Wang, *Key Technologies for 5G Wireless Systems*. Cambridge University Press, 2017.

[2] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8402–8413, Dec. 2013.

[3] P. Blasco and D. Gunduz, "Learning-based optimization of cache content in a small cell base station," in *Proc. of IEEE Int'l Conf. on Commun. (ICC)*, Sydney, Australia, Jun. 2014.

[4] T. Hou, G. Feng, S. Qin, and W. Jiang, "Proactive content caching by exploiting transfer learning for mobile edge computing," in *Proc. of IEEE Global Commun. Conf. (GLOBECOM)*, Singapore, Dec. 2017.

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

[6] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[7] B. Zhou, Y. Cui, and M. Tao, "Stochastic content-centric multicast scheduling for cache-enabled heterogeneous cellular networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 9, pp. 6284–6297, Sep. 2016.

[8] Z. Ding, P. Fan, G. K. Karagiannidis, R. Schober, and H. V. Poor, "NOMA assisted wireless caching: Strategies and performance analysis," *IEEE Trans. Commun.*, vol. 66, no. 10, pp. 4854–4876, Oct. 2018.

[9] X. Wang, Y. Bao, X. Liu, and Z. Niu, "On the design of relay caching in cellular networks for energy efficiency," in *Proc. of IEEE INFOCOM Workshop*, Shanghai, China, Apr. 2011.

[10] D. P. Bertsekas, *Dynamic Programming and Optimal Control, vol. 1, 4th ed.* Belmont, MA: Athena Scientific, 2017.

[11] M. L. Littman, T. L. Dean, and L. P. Kaelbling, "On the complexity of solving Markov decision problems," in *Proc. of Conf. on Uncertainty in Artificial Intelligence*, San Francisco, CA, Aug. 1995.