CONTENT PLACEMENT LEARNING FOR SUCCESS PROBABILITY MAXIMIZATION IN WIRELESS EDGE CACHING NETWORKS

Navneet Garg, Mathini Sellathurai†, Tharmalingam Ratnarajah

School of Engineering, The University of Edinburgh, Edinburgh, UK
†Heriot-Watt University, Edinburgh, UK

ABSTRACT

To meet increasing demands of wireless multimedia communications, caching of important contents in advance is one of the key solutions. Optimal caching depends on content popularity in future which is unknown in advance. In this paper, modeling content popularity as a finite state Markov chain, reinforcement Q-learning is employed to learn optimal content placement strategy in homogeneous Poisson point process (PPP) distributed caching network. Given a set of available placement strategies, simulations show that the presented framework successfully learns and provides the best content placement to maximize the average success probability.

1. INTRODUCTION

To handle the future demands of wireless mobile communications, caching has been used to effectively reduce peak data rates by pre-storing the most popular contents in advance [1]. In caching, traffic load during peak periods shifts to off-peak periods, by fetching the "anticipated" popular contents; e.g., reusable video streams are stored in base station cache and are reused during off-peak hours [2]. In [3, 4], optimal content placement in cellular networks is provided to maximize cache hit rate, while [5, 6] present caching policy to maximize the success probability and area spectral efficiency. [7, 8] derive the local caching policy by minimizing cache miss probability. In [9], lower bounds for local and global caching are derived. Therefore, in a given network, caching policy can be obtained when popularity profile is known. In order to improve the quality of experience of the users, caching relies on the available observations to predict what to cache. Several works employ different models for predicting content popularity. [10, 11] employ neural networks and deep learning based approaches for prediction. [12] models popularity using auto-regressive (AR) model to predict the time series. [13] predicts content requests for video segments using a linear model. In [14], Q-learning is used to obtain the caching decisions to minimize cache refreshing cost when the popularity is modeled as Markov chain, considering a local section of the heterogeneous network.

In this paper, the global popularity profile of a PPP network is modeled as a finite state Markov chain. This paper considers the problem of finding the optimal caching probabilities from a given set of caching vectors in order to maximize the average success probability of the network. For a large set, searching for caching vectors for each state of Markov chain is computationally intensive. Therefore, Q-learning is employed. Simulations show that in few iterations, Q-learning learns the optimal caching policy.

2. SYSTEM MODEL

Consider a cellular network where the positions of base stations (BSs) are spatially distributed according to a two-dimensional (2D) homogeneous Poisson point process (PPP) Φ_{bs} with density $\lambda > 0$. The performance of the network is evaluated for a typical user o positioned at the Cartesian origin. Due to Slivnyak-Mecke theorem, stationary and isotropy of PPP, the results of a typical user apply to any user randomly located on 2D plane.

It is assumed that each user requests a specific content among the content set $\mathcal{C} \coloneqq \{c_1, c_2, \ldots, c_N\}$ of N files. Each content is of same size and normalized to 1. Moreover, in a network, each content has its popularity, which is assumed to be known. Let $\mathbf{f} = [f_1, \ldots, f_N]^T$ be the popularity distribution, i.e., $\mathbf{f}^T\mathbf{1} = 1$. This distribution follows a Zipf probability mass function, i.e., the probability that a user requests content c_j is given as $f_j = \frac{j^{-\gamma}}{\sum_{i=1}^{N} i^{-\gamma}}, \quad 1 \leq j \leq N$, where $\gamma < 1$ is Zipf exponent. It is also assumed that a cache memory of size L is available on each BS. The memory inventory is denoted by $\mathcal{L}_i$, which is a subset of $\mathcal{C}$, such that $|\mathcal{L}_i| \leq L$. Considering a probabilistic method, where content is independently placed in caches, the probability that content c_j is stored in a given BS is expressed as $a_j = \Pr[c_j \in \mathcal{L}_i], \quad 1 \leq j \leq N$. The probability that a typical user finds the desired content in a cache depends on the distribution of the random set $\mathcal{L}_i$ only through the one-set coverage probabilities a_j. These probabilities do not define the distribution on the random set $\mathcal{L}_i$. It defines the content placement policy for the network. Hence, it should satisfy the constraint $\sum_{j=1}^{N} a_j \leq L$. [3] states that the above condition is a necessary and sufficient condition for the existence of a distribution on the random set $\mathcal{L}_i$ satisfying $|\mathcal{L}_i| \leq L$ almost surely.

Average Success Probability (ASP): We consider user association based on both CSI and cached files in each helper. Specifically, when a user requests l^{th} file, it associates with the BS in the set $\Phi_{bs}(l)$ that has the strongest received power. Assuming equal power allocation (P), the received power at the user is $|h_i|^2 P r_i^{-\alpha}$, where $|h_i|^2 \sim \exp(1)$ and r_i are the channel power and the distance between i^{th} BS and user, and α is the path loss exponent. The downlink SINR at the typical user that request l^{th} file from i^{th} BS is given as

$$\Gamma_{il} = \frac{|h_i|^2 P r_i^{-\alpha}}{\underbrace{\sum_{j \in \Phi_{bs} \setminus \{i\}} |h_j|^2 P r_j^{-\alpha} + \sigma^2}_{I_i = I_{il} + I_{il}^c}}, \tag{1}$$

where $I_{il} = \sum_{j \in \Phi_{bs}(l) \setminus \{i\}} \frac{|h_j|^2 P}{r_j^{\alpha}}$, $I_{il}^c = I_i - I_{il}$, and σ^2 is the noise power. The term I_{il} represents the interference from the BSs that cache l^{th} file, while I_{il}^c is the interference from those BSs who do not cache l^{th} file. From the user's perspective, we use success probability measure to reflect quality of service, which is defined as the probability that the achievable rate of a typical user exceeds the rate requirements R_0. The average success probability can be written as

$$P_a(\mathbf{f}, \mathbf{a}) \coloneqq \sum_l f_l \Pr\left\{W \log_2\left(1 + \Gamma_{il}\right) \geq R_0\right\},$$

where W is the transmission bandwidth.

Theorem 1. *Average success probability of a typical user requesting l^{th} file, which has popularity f_l and caching probability a_l, is given as*

$$P_a(\mathbf{f}, \mathbf{a}) = \sum_l f_l g(a_l),$$

where

$$g(a_l) = a_l C \int_0^{\infty} \exp\left[-s_0 r_i^{\alpha}\left(\frac{\sigma^2}{P}\right)\right] \times \exp\left[-\left(a_l A + (1 - a_l) B + a_l C\right) r_i^2\right] (dr_i^2) \tag{2}$$

and $s_0 = 2^{\frac{R_0}{W}} - 1$, $A = 2\pi\lambda_{bs} s_0^{\frac{2}{\alpha}} \frac{1}{\alpha} \int_{\frac{1}{s_0}}^{\infty} \left(\frac{u^{\frac{2}{\alpha}-1}}{1+u}\right) du$,

$B = 2\pi\lambda_{bs} s_0^{\frac{2}{\alpha}} \frac{1}{\alpha} \int_0^{\infty} \left(\frac{u^{\frac{2}{\alpha}-1}}{1+u}\right) du$, $C = \pi\lambda_{bs}$.

Proof. Proof is given in [15]. □

Since heterogeneous networks are usually interference limited, it is reasonable to neglect the noise i.e., $\sigma^2 = 0$. In this case, the following corollary simplifies ASP.

Corollary 2. *For interference limited case, i.e., $\sigma^2 = 0$ or at high SNR, average success probability is given as*

$$P_a(\mathbf{f}, \mathbf{a}) = \sum_l f_l g_0(a_l),$$

where $g_0(a_l) = \frac{a_l C}{a_l A + (1 - a_l) B + a_l C}$.

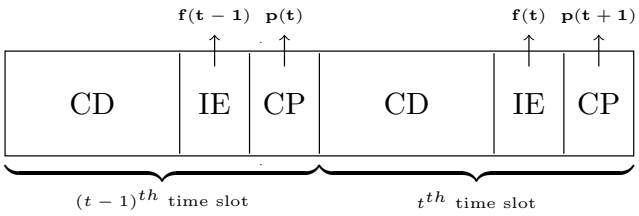

Fig. 1. The time slot structure and the evolution of key quantities. The slots can be of unequal length. CD: Content Delivery, CP: Content placement, IE: Information Exchange.

2.1. Timing Model

It is assumed that caching is carried out in a slotted fashion over slots $t = 1, 2, \ldots$ as shown in Figure 1. This figure shows the structure of each time slot.

At the beginning of a time slot, content delivery phase takes place as the user request arrives. If a requested file is stored in cache, it will be served, thus, incurring almost zero cost. On the other hand, if the requested file is not available in the cache, SB must fetch it using the backhaul link, incurring a considerable cost in terms of price, processing and delay.

The second phase pertains to information exchange, where the BSs exchange their observed popularity profiles to the network operator, and in return receive the estimated local popularity profile. Since in a PPP network, the perspective of the system as a whole is important, only the global popularity profile is considered for the present learning model.

Finally, in the last phase of a time slot, the content placement is carried out, i.e., the optimal selection of files are stored for the next time slot. For the PPP system as a whole, caching probabilities vector ($\mathbf{a}$) denotes the action which decides the caching at each BS in the PPP system. Let $\mathcal{A} \coloneqq \left\{\mathbf{a} | \mathbf{a} \in [0,1]^N, \mathbf{a}^T \mathbf{1} = L\right\}$ denote the set of all feasible actions. At the end of time slot t, the files are selectively cached for the content delivery phase of time slot $t+1$. It means that the caching action at time t, $\mathbf{a}(t)$ depends on the popularity distribution of previous time slot, $\mathbf{f}(t-1)$. Therefore, after observing the popularity in time slot t, the overall system state can be given as

$$\mathbf{s}(t) \coloneqq \begin{bmatrix} \mathbf{f}(t) \\ \mathbf{a}(t) \end{bmatrix}. \tag{3}$$

2.2. Popularity Profile Dynamics

As depicted in Figure 2, popularity profiles are modeled using Markov chains. Specifically, popularity profiles will be assumed generated by an underlying Markov process with $|\mathcal{F}|$ states collected in the set $\mathcal{F} \coloneqq \left\{\mathbf{f}_1, \ldots, \mathbf{f}_{|\mathcal{F}|}\right\}$. Although $\mathcal{F}$ is known, the underlying transition probabilities are unknown, which is a practical assumption. Since $\mathbf{a}(t) \in \mathcal{A}$ and

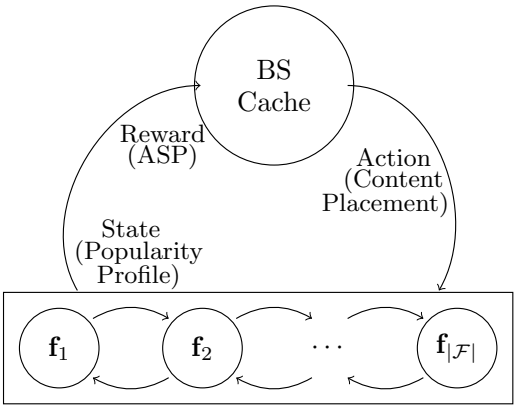

Fig. 2. Learning model, where in the environment, popularity varies according to Markov chain.

$\mathbf{f}(t) \in \mathcal{F}$, the set of states can be given as

$$\mathcal{S} = \mathcal{F} \times \mathcal{A} = \left\{ \mathbf{s} = \begin{bmatrix} \mathbf{f} \\ \mathbf{a} \end{bmatrix} \middle| \mathbf{f} \in \mathcal{F}, \mathbf{a} \in \mathcal{P} \right\}.$$

3. REINFORCEMENT LEARNING (RL)

The performance of a caching strategy can be measured by how well a network of BSs successfully communicates the the cached files to the users. We define the overall cost of a PPP network by average success probability (ASP), i.e.,

$$c(\mathbf{s}(t)) \coloneqq 1 - P_a(\mathbf{f}(t), \mathbf{a}(t)) \tag{4}$$

where $\mathbf{a}(t)$ is the caching vector selected at the end of $(t-1)^{th}$ time slot. At the end of each time slot, this cost is computed. Since $\mathbf{f}(t)$ and $\mathbf{a}(t)$ denote the global popularity distribution and average availability of cached files in the network respectively, the ASP is the reasonable choice to select as a performance measure of a network. The action $\mathbf{a}(t)$ controls the caching of the whole PPP network. Thus, it is selected to maximize the success probability. In this approach, the learner seeks the optimal policy by interactively making sequential decisions, and observing the corresponding costs.

Let us define the policy function $\pi : \mathcal{S} \to \mathcal{A}$, which maps any state to the action set. Under policy $\pi(\cdot)$, the caching is carried out via action $\mathbf{a}(t+1) = \pi(\mathbf{s}(t))$, dictating the placement policy for the network at time $t+1$. Caching performance is measured through the so-termed state value function

$$V_\pi(\mathbf{s}(t)) = \lim_{T \to \infty} \mathbb{E}\left[\sum_{\tau=t}^{T} \gamma^{\tau - T} c(\mathbf{s}(t)) \right], \tag{5}$$

which is the total average cost incurred over an infinite time horizon, with future terms discounted by factor $\gamma \in [0, 1)$. Since taking action influences the future states, future costs are always affected by past and present actions. Discount factor captures this effect, whose tuning trades off current versus future costs. The objective of this paper is to find the optimal policy π^* such that the average cost of any state is minimized $\pi^* = \arg\min_\pi V_\pi(\mathbf{s})$.

Optimality Conditions: Let $\Pr(\mathbf{s}, \mathbf{s}'|\mathbf{a})$ be the transition probability of going from the current state $\mathbf{s}$ to the next state $\mathbf{s}'$ under action $\mathbf{a}$. Bellman equations express the state value function in a recursive fashion as

$$V_\pi(\mathbf{s}) = c(\mathbf{s}) + \gamma \sum_{\mathbf{s}' \in \mathcal{S}} \Pr(\mathbf{s}, \mathbf{s}'|\pi(\mathbf{s})) V_\pi(\mathbf{s}'), \forall \mathbf{s}$$

For Q-learning, define the state-action value function as

$$Q_\pi(\mathbf{s}, \mathbf{a}) = c(\mathbf{s}) + \gamma \sum_{\mathbf{s}' \in \mathcal{S}} \Pr(\mathbf{s}, \mathbf{s}'|\mathbf{a}) V_\pi(\mathbf{s}'), \forall \mathbf{s}, \mathbf{a}. \tag{6}$$

Policy Iteration algorithm: When $\Pr(\mathbf{s}, \mathbf{s}'|\mathbf{a})$ is known, one can readily obtain $V_\pi(\mathbf{s})$, and policy iteration can be used to obtain optimal policy π. The policy iteration algorithm initialized with π_0 proceeds with the following updates at i^{th} iteration.

1. *Policy Evaluation:* For a fixed policy π_i, obtain $V_{\pi_i}(\mathbf{s})$, for all $\mathbf{s} \in \mathcal{S}$.
2. *Policy Iteration:* Update policy using $\pi_{i+1}(\mathbf{s}) = \arg\min_{\mathbf{a}} Q_{\pi_i}(\mathbf{s}, \mathbf{a}), \forall \mathbf{s}$.

Optimal Caching: Q-learning is an online RL method to jointly infer the optimal policy and estimate the optimal state-value function. The optimal policy can be obtained as

$$\pi^* = \arg\min_{\mathbf{a}} Q_\pi(\mathbf{s}, \mathbf{a}), \forall \mathbf{s}$$

which provides the optimal state-action values $Q^*(\mathbf{s}, \mathbf{a}) = Q_{\pi^*}(\mathbf{s}, \mathbf{a})$. The Q-function and value function are related by

$$V^*(\mathbf{s}) \coloneqq V_{\pi^*}(\mathbf{s}) = \min_{\mathbf{a}} Q^*(\mathbf{s}, \mathbf{a})$$

which gives from (6)

$$Q^*(\mathbf{s}, \mathbf{a}) = c(\mathbf{s}) + \gamma \sum_{\mathbf{s}' \in \mathcal{S}} \Pr(\mathbf{s}, \mathbf{s}'|\mathbf{a}) \min_{\mathbf{a}} Q^*(\mathbf{s}, \mathbf{a}) \tag{7}$$

Q-learning: The instantaneous error can be written as

$$\varepsilon(\mathbf{s}(t), \mathbf{a}(t)) \coloneqq \frac{1}{2} \left[\underbrace{c(\mathbf{s}) + \gamma \min_{\mathbf{a}'} Q(\mathbf{s}(t), \mathbf{a}')}_{T_Q(t)} - Q(\mathbf{s}(t), \mathbf{a}(t)) \right]^2$$

The Q-function is updated as using stochastic gradient as

$$Q_t(\mathbf{s}(t), \mathbf{a}(t)) = (1 - \beta_t) Q_t(\mathbf{s}(t), \mathbf{a}(t)) + \beta_t T_Q(t) \tag{8}$$

Algorithm 1 ASP maximization via Q-learning.

1: Initialize state $\mathbf{s}(0)$ randomly and $Q_0(\mathbf{s}, \mathbf{a}) = 0 \forall \mathbf{s}, \mathbf{a}$
2: **for** $t = 1, 2, \ldots$ **do**
3: After content delivery at $t-1$ time slot, do the following

4: *Content Placement: take action* $\mathbf{a}(t-1)$ *for time* t *chosen probabilistically*

$$\mathbf{a}(t-1) = \begin{cases} \arg\min_{\mathbf{a}} Q_{t-1}(\mathbf{s}, \mathbf{a}) & w.p.\ 1-\epsilon \\ random\ \mathbf{a} \in \mathcal{A} & w.p.\ \epsilon \end{cases}$$

5: *Information Exchange:* popularity profile $\mathbf{a}(t)$ is revealed based on user requests
6: set $\mathbf{s}(t) = \mathbf{a}(t)$ and compute $c(\mathbf{s}(t))$
7: Update

$$Q_t(\mathbf{s}(t), \mathbf{a}(t)) = (1-\beta_t) Q_{t-1}(\mathbf{s}(t), \mathbf{a}(t)) + \beta_t \left[c(\mathbf{s}(t)) + \gamma \min_{\mathbf{a}'} Q_{t-1}(\mathbf{s}(t), \mathbf{a}') \right] \quad (9)$$

8: **end for**

The procedure has been summarized in Algorithm 1. In this algorithm, after initialization of Q-matrix, the action (caching vector) is randomly selected with probability ϵ and greedy action is selected with probability $1-\epsilon$. After content placement phase, next state is obtained, which is then used to update the Q-values. This procedure is repeated until convergence.

4. SIMULATION RESULTS

For simulations, popularity profile set contains Zipf distributed profiles with different parameters $s = 0.7, 1$ and $L = 2$. Action set is selected randomly with $|\mathcal{A}| = 8$. The transition probabilities are set as $P = \begin{bmatrix} 0.6 & 0.4 \\ 0.2 & 0.8 \end{bmatrix}$. PPP parameters for computing ASP are as follows: noise power $\sigma^2 = 0$, BS density $\lambda_{bs} = 200$, bandwidth $W = 24$kHz, path loss exponent $\alpha = 3.5$, and rate threshold $R_0 = 1$.

Figure 3 illustrates the ASP values for different iterations with small state-action space $|\mathcal{F}| = 2$, $N = 10$, $|\mathcal{A}| = 8$ in (a) and large state-action space $|\mathcal{F}| = 16$, $N = 20$, $|\mathcal{A}| = 32$ in (b). From Figure 3 (a), it can be seen that in few iterations, Q-learning learns the appropriate action for each of the two states. After that, optimal action (content placement) is selected for each state. It can be observed from Figure 3 (b) that for increased number of contents, number of iterations for convergence increases. In Q-learning, exploration and exploitation controls the convergence. Higher value of ϵ in Algorithm 1 supports greater possibility of exploration. The greedy selection, i.e., exploitation occurs with probability $1-\epsilon$. In 3 (b), the convergence is attained near 2×10^3 iterations. A band after convergence represents the different costs (ASP) after choosing action for each different state. However, some other ripples are also present after convergence, indicating the exploration, which can be controlled by ϵ.

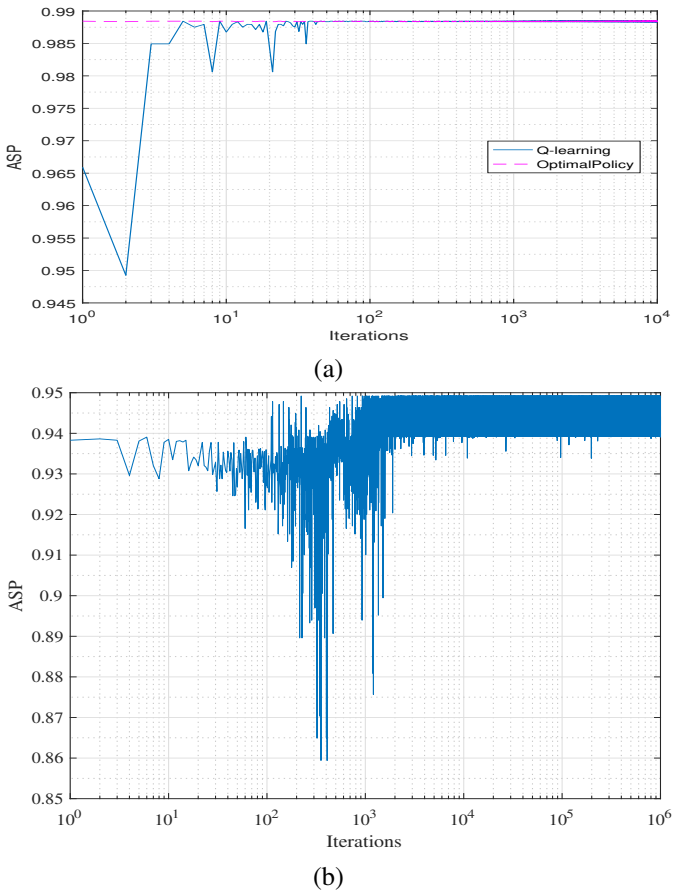

Fig. 3. ASP maximization with iterations with (a) small state-action space $|\mathcal{F}| = 2$, $N = 10$, $|\mathcal{A}| = 8$, (b) large state-action space $|\mathcal{F}| = 16$, $N = 20$, $|\mathcal{A}| = 32$.

5. CONCLUSION

In this paper, for a PPP network, global content popularities have been modeled using a finite state Markov chain. Given a set of caching probability vectors, a Q-learning method has been presented to find the optimal content placement. Simulations show that the Q-learning learns the best placement among the set and for large state-action space, it takes more iterations to reach convergence.

Acknowledgment

This work was supported in part by the U.K. Engineering and Physical Sciences Research Council under Grant EP/P009549/1 and EP/P009670/1 and in part by the U.K.-India Education and Research Initiative Thematic Partnerships under Grant DSTUKIERI-2016-17-0060 and UGCUKIERI 2016-17-058.

6. REFERENCES

[1] Karthikeyan Shanmugam, Negin Golrezaei, Alexandros G Dimakis, Andreas F Molisch, and Giuseppe Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 8402–8413, 2013.

[2] Konstantinos Poularakis and Leandros Tassiulas, "On the complexity of optimal content placement in hierarchical caching networks," *IEEE Transactions on Communications*, vol. 64, no. 5, pp. 2092–2103, 2016.

[3] Bartlomiej Blaszczyszyn and Anastasios Giovanidis, "Optimal geographic caching in cellular networks," in *IEEE International Conference on Communications (ICC)*, 2015, pp. 3358–3363.

[4] Berksan Serbetci and Jasper Goseling, "Optimal geographical caching in heterogeneous cellular networks with nonhomogeneous helpers," *arXiv preprint arXiv:1710.09626*, 2017.

[5] Dong Liu and Chenyang Yang, "Caching policy toward maximal success probability and area spectral efficiency of cache-enabled hetnets," *IEEE Transactions on Communications*, vol. 65, no. 6, pp. 2699–2714, 2017.

[6] Dong Liu, Binqiang Chen, Chenyang Yang, and Andreas F Molisch, "Caching at the wireless edge: design aspects, challenges, and future directions," *IEEE Communications Magazine*, vol. 54, no. 9, pp. 22–28, 2016.

[7] Konstantin Avrachenkov, Jasper Goseling, and Berksan Serbetci, "A low-complexity approach to distributed cooperative caching with geographic constraints," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 1, pp. 27, 2017.

[8] Konstantin Avrachenkov, Xinwei Bai, and Jasper Goseling, "Optimization of caching devices with geometric constraints," *Performance Evaluation*, vol. 113, pp. 68–82, 2017.

[9] Mohammad Ali Maddah-Ali and Urs Niesen, "Fundamental limits of caching," *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2856–2867, 2014.

[10] Jiaying Yin, Lixin Li, Huisheng Zhang, Xu Li, Ang Gao, and Zhu Han, "A prediction-based coordination caching scheme for content centric networking," in *WOCC*, 2018, pp. 1–5.

[11] Wai-Xi Liu, Jie Zhang, Zhong-Wei Liang, Ling-Xi Peng, and Jun Cai, "Content popularity prediction and caching for ICN: A deep learning approach with SDN," *IEEE access*, vol. 6, pp. 5075–5089, 2018.

[12] Hiroki Nakayama, Shingo Ata, and Ikuo Oka, "Caching algorithm for content-oriented networks using prediction of popularity of contents," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2015, pp. 1171–1176.

[13] Yuanzun Zhang, Xiaobin Tan, and Weiping Li, "PPC: Popularity prediction caching in ICN," *IEEE Communications Letters*, vol. 22, no. 1, pp. 5–8, 2018.

[14] Alireza Sadeghi, Fatemeh Sheikholeslami, and Georgios B Giannakis, "Optimal and scalable caching for 5g using reinforcement learning of space-time popularities," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 180–190, 2018.

[15] Navneet Garg, Vimal Bhatia, Bharath Bettagere, Mathini Sellathurai, and Tharmalingam Ratnarajah, "Online learning models for content popularity prediction in wireless edge caching," *arXiv preprints*, 2019.