A Joint Scheduling and Content Caching Scheme for Energy Harvesting Access Points with Multicast

Linhao Dong*, Dusit Niyato†, Dong In Kim* and Dinh Thai Hoang†
*College of Information and Communication Engineering, Sungkyunkwan University (SKKU), Suwon, Korea 440-746
Email: linhaodong@skku.edu; dikim@skku.ac.kr
†School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798
Emails: {dniyato,thdinh}@ntu.edu.sg

***Abstract*—In this work, we investigate a system where users are served by an access point that is equipped with energy harvesting and caching mechanism. Focusing on the design of an efficient content delivery scheduling, we propose a joint scheduling and caching scheme. The scheduling problem is formulated as a Markov decision process and solved by an on-line learning algorithm. To deal with large state space, we apply the linear approximation method to the state-action value functions, which significantly reduces the memory space for storing the function values. In addition, the preference learning is incorporated to speed up the convergence when dealing with the requests from users that have obvious content preferences. Simulation results confirm that the proposed scheme outperforms the baseline scheme in terms of convergence and system throughput, especially when the personal preference is concentrated to one or two contents.**

***Index Terms*—Scheduling, energy harvesting, caching mechanism, multicast, Markov decision process, reinforcement learning.**

I. Introduction

In recent years, multimedia services over the Internet evince a trend of dominating mobile data traffic [1]. However, available spectrum resources lag behind the demanding bandwidth for mobile users, which most likely degrades the quality-of-service (QoS) in the network. To improve the system throughput, the base station (BS) or the access point (AP) with cache capability is deployed at several hot places in the wireless network [2], which is designed to pre-download the most requested contents from the data servers for a local group. Due to the short communication distances and simple data access protocol between the caching AP and the end users, higher transmission rate is achieved with reduced latency.

Many issues have been investigated when implementing the caching AP in wireless networks, such as cache replacement policy [3], [4], joint optimization with caching [5], [6], and request admission control [7]. In [3], two cache access mechanisms based on information update were proposed for a multi-cell system. In [4], a distributed cache replacement strategy according to both the popularity and the transmission cost for the BSs was proposed. For system optimization, authors in [5] proposed joint transmission and caching strategies that combine content replacement and delivery phases. To improve the spectrum efficiency, a resource allocation was proposed in [6] with consideration of the social behavior. In [7], a cache admission control and replacement policy was developed to reduce the cache miss ratio.

Recently, energy harvesting (EH) technology has demonstrated its effectiveness in green communications that helps the system acquire more energy from natural or ambient RF sources [8]. For scenarios of multimedia communication and Internet-of-Things (IoT), equipping nodes with EH and cache technologies is expected to leverage the energy efficiency. In [9], a popularity based proactive content push mechanism was proposed for EH powered small-cell BSs in heterogeneous networks, where optimal push policy is obtained by solving a Markov decision process (MDP). However, the impact of the cache size on the push policy was not considered. In [10], an online power control scheme is designed for EH powered BSs equipped with cache. By solving an infinite horizon dynamic problem (DP), the optimal power policy can be found. The authors in [11] proposed a joint admission control and content caching scheme for competitive EH powered caching APs. This scheme makes the decision not only on accepting incoming requests, but also on caching such requested contents. In [12], a content caching and distribution framework was proposed for OFDM networks, where popular contents are distributively cached at EH powered APs which are geometrically closer to requested users. However, the aforementioned works model users as a local group, while ignoring the individual preference to the contents. Besides, the algorithms in these works are only validated in well-explored Markov models, which is impractical for dynamic systems.

In this work, we investigate a system which contains an EH powered AP equipped with a cache, a content server linked to the AP by a backhaul, and mobile users. The AP schedules the serving order of users and chooses an optimal action. The harvested energy is fully used by the AP for content delivery. The contributions of this paper are summarized as follows:

- We propose a joint scheduling and caching scheme for a multi-user scenario with multicast transmission. To achieve a quick convergence in performance, users' content preferences are learned and incorporated with rewards, while the content popularity is unknown in

The work was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIP) under Grant 2017R1A2B2003953.

978-1-5090-5019-2/17/$31.00 ©2017 IEEE

advance.

- We apply a model-free reinforcement learning algorithm to search the optimal policy, where the transition probability between any two states is not necessarily known by the AP. By learning and updating the state-action value function (Q-function), the AP recursively obtains the optimal action.
- To deal with the large state space MDP (i.e., the numbers of users and contents, and the size of cache are large), the Q-function is linearly approximated, which largely reduces the storage space. In each iteration, the AP tunes the parametric weights by only comparing the current value with the previous one, which further reduces the memory for storing weights.

Simulation results show that with the preference learning, the proposed joint scheduling and caching scheme is superior to the scheme without preference learning in terms of convergence and system throughput, especially when dealing with requests from the users that have narrow content preference.

II. System Model

We consider a multi-user system in which one AP serves users in a TDMA manner. As shown in Fig. 1, the AP harvests energy from the environment and stores it in a rechargeable battery with finite capacity B units. We assume that sending a request spends a much smaller duration compared to sending a content packet. Hence, it can be further assumed that the AP receives requests from users simultaneously at the beginning of each cycle. After receiving a content request d, the AP checks whether this content is in the cache. If the content has already been cached, the AP can reply the request by delivering the content; otherwise, such a content can be downloaded from the server that is linked to the AP by high speed backhaul connection and forwarded to the requesting users. Note that one content can be requested from more than one user. Before executing any action, the AP checks the battery level to ensure its survival, i.e., at least $B_{\min}$ $(0 < B_{\min} < B)$ units are required. If the battery level $b < B_{\min}$, the AP switches into sleep mode and harvests energy only. We assume that the energy accumulation takes more cycles than that consumed by transmission in this system. Once the request is replied by the AP, a reward is gained.

To consider users' personalized preferences to the contents, the request frequency is incorporated with the reward function in this work. However, such information should be learned during the service since it is unknown in advance. Hence, the reward value should be calculated with the preference and updated according to the incoming requests. The AP schedules the packets with a TDMA mode and serves users in both unicast and multicast transmission.

III. Problem Formulation

In this section, we formulate the scheduling scheme as a reinforcement learning problem. To concrete the system setting, the following sets are defined:

- $\mathcal{U}$ is defined as the set of users;

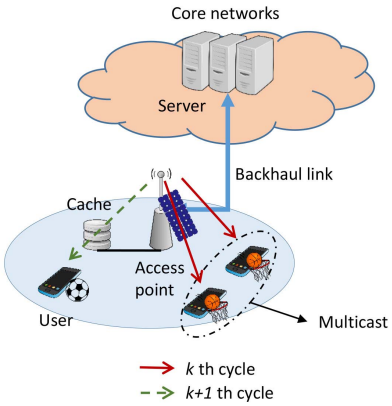


Fig. 1: The TDMA based system model in downlink, where the AP is equipped with an energy harvester and a cache; the basketball and football icons indicate different requests.

- $\mathcal{T}$ is defined as the whole content set at the server;
- $\mathcal{C}$ is defined as the set of cached contents, and it must hold that $\mathcal{C} \subset \mathcal{T}$;

Once received the request d from users, the AP can select one of 4 actions at each cycle: 1) a_0, the AP rejects the incoming request, which drains no battery energy but harvests energy only; 2) a_1, the AP accepts the request and sends the content if it is cached, consuming 2 units of energy; 3) a_2, the AP accepts the request and downloads the corresponding content from the server if it is not in the cache, but does not cache it, consuming 3 units of energy; 4) a_3, the AP executes content downloading and caches the content if it has not been cached, which consumes 4 units totally[1]. We define $\boldsymbol{A} = \{a_0, a_1, a_2, a_3\}$ as the action space and let $a \in \boldsymbol{A}$ denote a certain action. We also assume that the AP harvests 1 unit of energy per cycle with a certain success rate. Note that only action a_3 changes the cache state.

A. State Space and Partial Observable Transition Probabilities

The transition among all states is treated as a discrete Markov chain. We define $\mathscr{B}$, $\mathscr{C}$ and $\mathscr{D}$ as the energy level, the cache content and the incoming request state spaces, respectively. For the AP, the state space can be written as

$$\mathscr{S} = \big\{(\mathscr{B}, \mathscr{C}, \mathscr{D}); b \in \{0, \ldots, B_{\min}, \ldots, B\}, c \in \mathscr{C}, d \in \mathscr{D}\big\}. \quad (1)$$

The state is then defined as a variable $s = (b, c, d)$, where b and c denote the battery state and cache state at the AP for serving users, respectively, and d represents the request state.

We now give the transition probabilities from state b to the next one b' for all actions. During each cycle, we assume that the AP harvests one battery unit with success rate λ that follows the Bernoulli distribution [8] in the simulator[2].

[1]Note that the data caching consumes energy at the microarchitecture level [13].

[2]This setting simplifies the real harvesting scenario; the energy arrival can also be assumed to be a Poisson process [10].

Besides, when the battery level is below $B_{\min}$, one of the 4 actions is allowed to be executed only once regardless of the remaining level, and then the AP switches to the sleep mode for charging.

If $a = a_0$ is selected by the AP, no energy will be consumed, and the battery will be charged by one unit with the rate λ. When the battery level reaches the limit B, i.e., $b = B$, no extra battery unit can be harvested, so such state is the absorbing state with transition probability of 1. The transition probability for user i thus can be written in two cases, as

$$B_{b,b'}(a_0)|_{b<B} = \begin{cases} 1-\lambda, & b' = b \\ \lambda, & b' = b+1; \end{cases} \tag{2}$$

and

$$B_{b,b'}(a_0)|_{b=B} = \begin{cases} 1, & b' = b \\ 0, & b' = b+1. \end{cases} \tag{3}$$

When the battery level is greater than the survival level, i.e., $b \geq B_{\min}$, the AP can select one action among a_1, a_2 and a_3 if $B_{\min} = 4$, and consumes 2, 3, and 4 battery units, respectively. After taking the selected action, the AP harvests 1 battery unit with rate λ. Hence, the transition probability $B_{b,b'}(a_{1,2,3})$ describes the transition from the initial battery state before taking action to the final state after harvesting one battery unit as following cases:

$$B_{b,b'}(a_{1,2,3})|_{B_{\min} \leq b \leq B} = \begin{cases} \lambda|_{a=a_1}, & b' = b-1 \\ 1-\lambda|_{a=a_1} \text{ or } \lambda|_{a=a_2}, & b' = b-2 \\ 1-\lambda|_{a=a_2} \text{ or } \lambda|_{a=a_3}, & b' = b-3 \\ 1-\lambda|_{a=a_3}, & b' = b-4 \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

If certain energy arrival patterns are introduced, this process can be re-modeled as a continuous state Markov chain [10].

Now we derive the transition probability for the cache state. According to the action protocol, the cache updates its contents only when $a = a_3$. Hence, the transition probability for other actions can be written as

$$C_{c,c'}(a_{0,1,2}) = \begin{cases} 1, & c' = c \\ 0, & c' \neq c. \end{cases} \tag{5}$$

The number of states is determined on the size of cache and the number of contents in the server. Let $n = |\mathcal{T}|$ and $h = |\mathcal{C}|$, where $|\cdot|$ denotes the cardinality. If we write the transition probability of all cache states as a single matrix form $\boldsymbol{C}_{c,c'}(a_{0,1,2})$, the size of such matrix is $\binom{n}{h} \times \binom{n}{h}$, where $\binom{\cdot}{\cdot}$ is the Bernoulli operator. Before action a_3 is selected, the AP has to ensure that no requested content is cached. Otherwise, a_1 is likely to be selected since such an action saves energy. To this end, the transition probability under a_3 can be written as

$$C_{c,c'}(a_3) = \begin{cases} \frac{1}{h}, & c' = c \cup \{d_m\}, \forall d_m \in \mathcal{C} \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Since the transition probability $\boldsymbol{D}_{d,d'}$ from one request state d to the next d' is hardly to be estimated in our system, the total space transition probability can be viewed partially observable.

B. System Throughput

In this work, the data throughput is defined as the total number of successfully delivered content packets to the requesting users within one cycle. Assuming that each delivery is perfect, for a particular user i, the immediate throughput can be written as

$$u_s^{(i)}(a) = \begin{cases} 0, & a = a_0 \\ 1, & a = a_{1,2,3}. \end{cases} \tag{7}$$

Note that when action a_0 is selected, no contents will be delivered.

For state s, we define $\pi(s) = a$ as the policy by which the action a is selected at state s, and a fixed policy $\boldsymbol{\pi} = \pi(s)$ can be determined by the AP with executing a series of actions. If the AP harvests energy successfully in every cycle before transmission, the total throughput during a time window T for user i is defined as

$$U_{\boldsymbol{\pi}}^{(i)} = \sum_{k=1}^{T} u_{s_k}^{(i)}(\pi(s_k)), \tag{8}$$

where k is the cycle index. The total throughput of the user group can be written as

$$U_{\boldsymbol{\pi}} = \sum_{i \in \mathcal{U}} \sum_{k=1}^{T} u_{s_k}^{(i)}(\pi(s_k)). \tag{9}$$

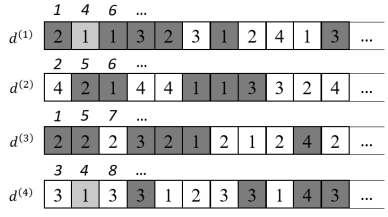
The policy can be randomly designed, being independent from individual demands. However, if the users are treated as a whole group, the content preference shall be easily erased, which leads to a sub-optimal policy. To address such problem, we propose a scheduling scheme that maximizes the system throughput with multicast transmission mode.

IV. Joint Scheduling and Caching Scheme

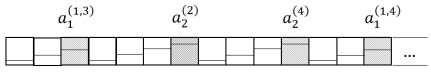
The AP schedules the transmission for different users with a serving order in a TDMA mode. To achieve the least energy consumption as possible, two principles are required in the scheduling:

- If the requested contents are cached, the AP serves these requests first by forwarding the cached contents;
- If a content is commonly requested by more than one user, the AP replies the requests in multicast mode for energy saving.

As shown in Fig. 2, the AP schedules the packet delivery for users with unique request with a lower priority. We now detail the multicast mode. In each cycle, the AP delivers only one requested content from either the cache (a_1) or the server ($a_{2,3}$). We assume that the AP receives all requests simultaneously. If a certain content is requested by more than one user, the AP will switch to the multicast mode to respond these users with shared request. On the other hand, if the content is only requested by one user, the AP will



(a)



(b)

Fig. 2: The scheduling protocol for users, where harvesting 1 battery unit is always successful in this figure. Once the AP received a shared request (illustrated by shaded numbers), a multicast transmission is in operation. The italic numbers show the serving order, and contents 1 and 2 are assumed to be cached at the very beginning.

switch to the unicast mode. We set the transition based on the incoming requests. For each request, there is one possible user combination out of $\sum_{l=1}^{|\mathcal{U}|}\binom{|\mathcal{U}|}{l}$. Hence, the number of request states is $n\sum_{l=1}^{|\mathcal{U}|}\binom{|\mathcal{U}|}{l}$, and the number of global states is $B\times\binom{n}{h}\times n\sum_{l=1}^{|\mathcal{U}|}\binom{|\mathcal{U}|}{l}$.

A. Defining State-Action Value Function

When one packet is delivered to the user, a reward is collected by the AP. To incorporate the reward with individual content preference, we define $\boldsymbol{f}^{(i)} = \left[f^{(i)}(d_0),\ldots,f^{(i)}(d_m),\ldots,f^{(i)}(d_n)\right]^T$, $d_m\in\mathcal{T}$ as the request frequency vector, and each element represents how frequent a content is requested by user i, as

$$f^{(i)}(d_m)=\frac{\text{\# of requesting } d_m}{\text{\# of requesting all contents}},\forall i\in\mathcal{U},d_m\in\mathcal{D} \tag{10}$$

and $\sum_{d_m\in\mathcal{D}}f^{(i)}(d_m)=1$. The frequency vector $\boldsymbol{f}^{(i)}$ is updated whenever a content request is replied. We assume that the higher value the content frequency is, the more preferable this content is. We also define the cost vector $\kappa=[\kappa(a_0),\kappa(a_1),\kappa(a_2),\kappa(a_3)]^T$, where $\kappa(a_0)<\kappa(a_1)<\kappa(a_2)<\kappa(a_3)$ due to the fact that a_3 consumes the most battery units while a_0 consumes the least. Hence, the reward for serving user i is given as

$$\begin{aligned}&R^{(i)}_{s_k,s_{k+1}}(\pi(s_k))\\&=u^{(i)}_{s_k}(\pi(s_k))f^{(i)}(d_m)-\beta\kappa(\pi(s_k)),d_m\in\mathcal{D}\end{aligned} \tag{11}$$

where $\beta>0$ is a scalar that transforms the action cost to part of the value. One can see that (11) is dependent on the content preference of user i, i.e., $R^{(i)}_{s_k,s_{k+1}}(\pi(s_k))\propto u^{(i)}_{s_k}(\pi(s_k))f^{(i)}(d_m)$, which encourages the AP to cache popular contents for high rewards. Note that when $a=a_0$, $R^{(i)}_{s_k,s_{k+1}}(\pi(s_k))=-\beta\kappa(\pi(s_k))\le 0$; in other words, if the AP rejects the incoming request, it will receive a penalty. If the preference learning is not considered, $f^{(i)}(d_m)$ should be replaced by $1/n$ for fairness. We hereby define $\xi^{(i)}_k$ as the binary number that indicates whether user i is served in cycle k. If user i requests d_m, $\xi^{(i)}_k(d_m)=1$, otherwise 0. Hence, the total reward for serving users during cycle k can be equivalent written as

$$R_{s_k,s_{k+1}}(\pi(s_k))=\log\left(1+\sum_{i\in\mathcal{U}}\xi^{(i)}_k(d_m)R^{(i)}_{s_k,s_{k+1}}(\pi(s_k))\right). \tag{12}$$

We use logarithm function on the right hand side to avoid the possible prompt jump between two consecutive rewards. For instance, when the content at kth cycle is requested by 100 users and the content at $(k+1)$th cycle is only requested by one user, the difference between the total rewards is around $\log(1+100)-\log(1+1)\approx 3.9$. However, if the total reward is defined as the sum, i.e., $\sum_{i\in\mathcal{U}}\xi^{(i)}_k(d_m)R^{(i)}_{s_k,s_{k+1}}(\pi(s_k))$, the difference will be $101-2=99$, which would introduce a slow convergence during the function update.

The Q-function is defined as $Q_{\boldsymbol{\pi}}(s,a)$, which takes the expected sum of the discounted rewards starting at state s, taking action a, and then following policy $\boldsymbol{\pi}$

$$\begin{aligned}&Q_{\boldsymbol{\pi}}(s,a)\\&=E_{\boldsymbol{\pi}}\left[\sum_{k=0}^{T}\gamma^k R_{s_k,s_{k+1}}(\pi(s_k))\mid s_0=s,a_0=a\right],\end{aligned} \tag{13}$$

where $0<\gamma\le 1$ is the discount factor. Under a fixed policy, the Q-function satisfies Bellman equation [14] that is derived recursively as

$$\begin{aligned}Q_{\boldsymbol{\pi}}(s,\pi(s))&=E_{\boldsymbol{\pi}}\left[\sum_{k=0}^{T}\gamma^k R_{s_k,s_{k+1}}(\pi(s_k))\mid s_0=s\right]\\&=\sum_{s'\in\mathcal{S}}P_{s,s'}(\pi(s))\left[R_{s,s'}(\pi(s))+\gamma Q_{\boldsymbol{\pi}}(s',\pi(s'))\right].\end{aligned} \tag{14}$$

Since the throughput (8) is contained in the reward (11), maximizing (9) is equivalent to the following problem

$$\boldsymbol{\pi}^*=\arg\max_{\boldsymbol{\pi}}Q_{\boldsymbol{\pi}}(s,\pi(s)). \tag{15}$$

In fact, the transition probability $P_{s,s'}(\pi(s))$ is not fully known initially. Thus, this problem is addressed by the reinforcement learning (RL) through interactions with the environment (simulator) [15]. In this paper, we apply the online learning method, SARSA [16], to seek an optimal policy. By collecting the one-step-ahead sample, Q-function is updated as

$$\begin{aligned}&Q(s_{k+1},\pi^\epsilon(s_k))\\&=R_{s_k,s_{k+1}}(\pi^\epsilon(s_{k+1}))+\gamma Q(s_k,\pi^\epsilon(s_k)),\end{aligned} \tag{16}$$

where $\pi^\epsilon(s)$ represents the ϵ-greedy action selection method which can be written as

$$\pi^\epsilon(s)=\begin{cases}\arg\max_a Q(s,a_{0,2,3}), & \text{with probability } 1-\epsilon\\ \text{randi}(\boldsymbol{A}\setminus\{a_1\}), & \text{with probability } \epsilon,\end{cases} \tag{17}$$

where randi randomly selects an action from $\boldsymbol{A}$ except for a_1. ϵ-greedy method can well balance the exploration and the exploitation.

B. Linear Approximation of Q-Function

Apparently, if the total number of states is large, storing the Q-function value for each state requires a considerable memory space. For instance, when $B = 10$, $n = 20$, $h = 10$ and $|\mathcal{U}| = 40$, the number of values is up to $4.06 \times 10^{19} \times 4$, which is impossible for the AP to store such a gigantic set. Therefore, a practical method is to project the Q-function upon a subspace that approximates the exact function values. We hereby apply the weighted sum of features to represent the original function [14]. Hence, (13) can be approximated as

$$Q_{\boldsymbol{\pi}}(s,a) \approx \hat{Q}_{\boldsymbol{\pi}}(s,a,\boldsymbol{w}) = \boldsymbol{\phi}^T(s,a)\boldsymbol{w}, \tag{18}$$

where $\boldsymbol{\phi}(s,a)$ is the feature vector that represents the characteristics of each state and $\boldsymbol{w}$ is the weight vector.

To ensure that all characteristics are properly covered, we set the feature vector in Boolean domain as the follows:

$$\begin{aligned}\boldsymbol{\phi}^T(s,a) = \Big[&1, \phi_{\boldsymbol{b},1}, \ldots, \phi_{\boldsymbol{b},B}, \\ &\phi_{\boldsymbol{d},1}, \ldots, \phi_{\boldsymbol{d},n}, \phi_{\boldsymbol{c},1}, \ldots, \phi_{\boldsymbol{c},n}, \phi_{\boldsymbol{i},1}, \ldots, \phi_{\boldsymbol{i},|\mathcal{U}|}\Big]\end{aligned} \tag{19}$$

which can be divided into 4 groups corresponding to the battery level, the incoming content request, the cached contents and the requesting users. To illustrate how these features work, the battery and user features are detailed as

$$\phi_{\boldsymbol{b},j} = \begin{cases} 1, & \text{current level } j \\ 0, & \text{otherwise} \end{cases} \tag{20}$$

and

$$\phi_{\boldsymbol{i},d_m} = \begin{cases} 1, & \text{users who request content } d_m \\ 0, & \text{otherwise,} \end{cases} \tag{21}$$

respectively. Groups of battery and request have only one binary can be set as "1" but others (cache, user) are allowed to have more. In addition, features of different states are distinguishable, which eliminates the ambiguousness and guarantees a good coverage for value update.

By substituting (18) into (16), the difference of Q-function values between the current and the next states is written as

$$\begin{aligned}\delta = R_{s_k,s_{k+1}}(\pi^\epsilon(s_k)) + \gamma\phi^T(s_{k+1},a_{k+1})\boldsymbol{w} \\ - \phi^T(s_k,a_k)\boldsymbol{w}.\end{aligned} \tag{22}$$

Accordingly, the weight vector $\boldsymbol{w}$ is updated as [17]:

$$\boldsymbol{w} := \boldsymbol{w} + \alpha\delta\phi(s_k,a_k) \tag{23}$$

where the real number $\alpha > 0$ is the learning rate. Note that the feature vector can be learned by the AP, so only low-dimensional vector $\boldsymbol{w}$ needs to be stored in the memory. For the previous "$4.06 \times 10^{19} \times 4$" example, there need to be only 90×4 entries to represent all states, which largely reduces the storage cost. To this end, the scheduling scheme is summarized in **Algorithm 1**.

Algorithm 1 Joint scheduling and caching mechanism

procedure SCHEDULING
Input: B,$B_{\min}$,λ,α,n,h and γ
Output: π
  $\boldsymbol{w} \leftarrow$ arbitrary initial value
  $\langle s,a\rangle \leftarrow \langle s_0, \pi^\epsilon(s_0)\rangle$
  while *requests left* **do**
    descendingly sort requests with popularity
    for each sorted request d **do**
      if $b < B_{\min}$ **then**
        the AP harvests energy until $b \geq B_{\min}$
      else
        if d is cached **then** $a' \leftarrow a_1$
        else
          $a' \leftarrow \pi^\epsilon(s)$
          if $a' == a_3$ **then**
            cache the demanding content d
          end if
        end if
        $\hat{Q}^+(s,a,\boldsymbol{w}) \leftarrow R_{s,s'}(a') + \gamma\phi^T(s',a')\boldsymbol{w}$
        $\delta \leftarrow \hat{Q}^+(s,a,\boldsymbol{w}) - \hat{Q}(s,a,\boldsymbol{w})$
        $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha\delta\phi(s,a)$
        $\langle s,a\rangle \leftarrow \langle s',a'\rangle$
      end if
    end for
  end while
  return $\boldsymbol{\pi}$
end procedure

V. Performance Evaluation

In this section, we demonstrate the performance of our proposed joint scheduling and caching scheme. Based on the user behavior of requesting contents, the users are roughly divided into two types, namely:

- *Type I User*: each user has a narrow range of interests (up to 2);
- *Type II User*: each user has a wide range of interests (more than 2).

In our simulations, we showcase the impact of user preference on the system throughput with different numbers of users.

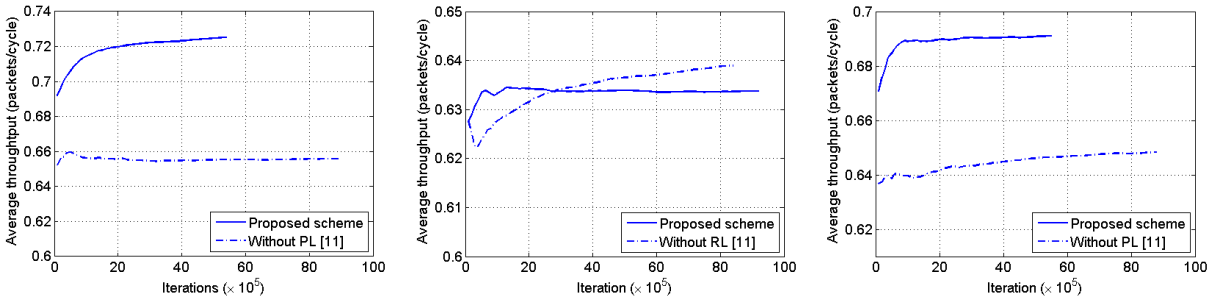
A. Simulation Setup

For simplicity, the delay and failure in data transmission of the backhaul link are ignored, and all issues of QoS are not considered. The energy harvesting rate is set to $\lambda = 0.5$. We set the battery capacity to $B = 10$ and its survival level as $B_{\min} = 5$. The value transfer rate is set to $\beta = 0.1$. For the RL process, the learning rate, discount factor and the ϵ-greedy factor are set to $\alpha = 0.2$, $\gamma = 0.3$ and $\epsilon = 0.5$, respectively. All elements of $\boldsymbol{w}$ are initialized to "1".

B. Numerical Results

We demonstrate the impact of preference learning on the system throughput in Figs. 3(a) and (b), where $n = 20$, $h = 10$ and $|\mathcal{U}| = 40$. Each user sends 10^6 requests to the AP. Note that the request sequence has a limited length, which means the update iterations of Q-function must be within a certain limit as well.

Considering the fairness of comparison, we re-shape the scheme in [11] as the RL and let it incorporate with multicast as well. The only difference between the proposed scheme and the one in [11] (named as "Without PL [11]" in figures) is that preference learning (PL) is not applied in [11]. For type



(a) (b) (c)

Fig. 3: The comparison of the average throughput over time horizon between the two schemes: (a) type I users; (b) type II users; and (c) mixed type I and II users.

I users, it can be observed that the proposed scheme improves the system throughput via learning procedure. In addition, the throughput of the proposed scheme is greater than that of "Without PL [11]" up to 10%, and the proposed scheme requires less iterations to update the Q-function. For type II users (shown in Fig. 3(b)), when the number of iterations is greater than 3×10^6, the throughput of "Without PL [11]" is superior to that of the proposed scheme because each user requests all contents with nearly equal chance, which suggests that the preference learning based on historical records updates each entries of $\boldsymbol{f}$ close to $1/n$. However, this performance gap is marginal.

Likewise, we demonstrate the performance of the two schemes for a mixed group of type I and II users in Fig. 3(c), which represents a general case in the network. It can be seen that the throughput of both schemes can be improved during the learning procedure. The proposed scheme outperforms the scheme in [11] in terms of throughput and Q-function update iterations. From both Figs. 3(a) and (b), it is obvious that the proposed scheme converges more quickly. With the help of preference learning, when a popular content is requested, the AP tends to cache such content instead of randomly rejecting the request, resulting in a less number of iterations.

VI. Conclusion and Future Work

In this work, we have proposed a joint scheduling and caching scheme for a system where users are served by the energy harvesting powered access point that is equipped with caching mechanism. The scheduling problem has been formulated as a Markov decision process (MDP) and solved by the reinforcement learning method, SARSA. To deal with a large state space, we have applied the linear approximation to the Q-function, which significantly reduces the memory space for storing the function values. Simulation results show that the proposed scheme outperforms the other baseline scheme in terms of convergence and system throughput for type I users and the mixed type user groups. For type II users, the proposed scheme outperforms the other baseline scheme at the early delivery stage.

To extend this work, a cross-layer design problem that makes a trade-off between the energy efficiency and the data throughput could be addressed as a mixed discrete and continuous MDP in the future work.

References

[1] X. Tao *et al.*, "Real-time personalized content catering via viewer sentiment feedback: a QoE perspective," *IEEE Netw.*, vol. 29, no. 6, pp. 14-19, Nov.-Dec. 2015.

[2] U. Niesen, D. Shah and G. W. Wornell, "Caching in Wireless Networks," *IEEE Trans. Inf. Theory*, vol. 58, no. 10, pp. 6524-6540, Oct. 2012.

[3] H. Chen, Y. Xiao and X. Shen, "Update-Based Cache Access and Replacement in Wireless Data Access," *IEEE Trans. Mobile Comput.*, vol. 5, no. 12, pp. 1734-1748, Dec. 2006.

[4] J. Gu *et al.*, "Distributed cache replacement for caching-enable base stations in cellular networks," in *Proc. IEEE ICC'2014*, Sydney, NSW, 2014, pp. 2648-2653.

[5] M. Gregori *et al.*, "Wireless Content Caching for Small Cell and D2D Networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1222-1234, May 2016.

[6] H. Wu *et al.*, "Resource allocation for wireless caching in socially-enabled D2D communications," *Proc. IEEE ICC'2016*, Kuala Lumpur, 2016, pp. 1-6.

[7] S. Lim *et al.*, "A novel caching scheme for Internet based mobile ad hoc networks," in *Proc. IEEE ICCCN'2003*, 2003, pp. 38-43.

[8] S. Zhou *et al.*, "GreenDelivery: proactive content caching and push with energy-harvesting-based small cells," *IEEE Commun. Mag.*, vol. 53, no. 4, pp. 142-149, April 2015.

[9] J. Gong *et al.*, "Proactive push with energy harvesting based small cells in heterogeneous networks," in *Proc. IEEE ICC'2015*, London, 2015, pp. 25-30.

[10] A. Kumar and W. Saad, "On the tradeoff between energy harvesting and caching in wireless networks," *Proc. IEEE ICC Workshop'2015*, London, 2015, pp. 1976-1981.

[11] D. Niyato *et al.*, "Joint admission control and content caching policy for energy harvesting access points," in *Proc. IEEE ICC'2016*, Kuala Lumpur, 2016, pp. 1-6.

[12] X. Huang and N. Ansari, "Content Caching and Distribution in Smart Grid Enabled Wireless Networks," *IEEE Internet Things J.*, vol. 4, no. 2, pp. 513-520, April 2017.

[13] D. M. Brooks *et al.*, "Power-aware microarchitecture: design and modeling challenges for next-generation microprocessors," *IEEE Micro*, vol. 20, no. 6, pp. 26-44, Nov/Dec 2000.

[14] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *J. Mach. Learn. Res.*, vol. 4, pp. 1107-1149, Dec. 2003.

[15] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1998.

[16] G. A. Rummery and M. Niranjan, *Online Q-learning using connectionist systems (tech. rep. no. cued/f-infeng/tr 166)*, Cambridge University Engineering Department, 1994.

[17] F. S. Melo, S. P. Meyn and M. I. Ribeiro, "An analysis of reinforcement learning with function approximation," in *ICML'2008*, Helsinki, 2008, pp. 664-71.