# A Learning Approach to Edge Caching with Dynamic Content Library in Cellular Networks

1st Xinruo Zhang
*Wolfson School*
*Loughborough University*
Loughborough, UK
X.Zhang@lboro.ac.uk

2nd Gan Zheng
*Wolfson School*
*Loughborough University*
Loughborough, UK
g.zheng@lboro.ac.uk

3rd Sangarapillai Lambotharan
*Wolfson School*
*Loughborough University*
Loughborough, UK
s.lambotharan@lboro.ac.uk

4th Mohammad Reza Nakhai
*Department of Informatics*
*King's College London*
London, UK
reza.nakhai@kcl.ac.uk

5th Kai-Kit Wong
*University College London*
London, UK
kai-kit.wong@ucl.ac.uk

***Abstract*—This paper focuses on joint edge cache placement and content delivery problem at a base station (BS) in the presence of spatio-temporal unknown content dynamics, where the BS can satisfy its users' content demand either directly from its local cache or by fetching from the content server. Unlike previous works that assume a static content library, we consider a more realistic non-stationary content library, where new contents may emerge over time at different locations, and we propose that the new contents cached at local users can be utilized by the BS to timely update its flexible cache memory in addition to its routine off-peak main cache update from the content server. To account for the traffic demand variations and the limited caching space at the BS, a user-aided non-stationary bandit-inspired caching algorithm is developed to gradually optimize the caching policy with the target of maximizing the weighted network utility in the long run. Simulation results validate that the proposed strategy outperforms various benchmark designs.**

***Index Terms*—non-stationary bandit, edge caching, dynamic content library**

## I. INTRODUCTION

Global mobile data traffic is experiencing an explosive growth and is predicted to reach 48.3 Exabytes per month by 2021, of which, 82 percent will be video traffic [1]. The backhaul data rate demand between the base stations (BSs) and the core network incurred by such rapid traffic growth, nevertheless, has become the major revenue and technical bottlenecks for the network operators, especially during peak traffic periods [2]. Since a large portion of backhaul traffic is contributed by duplicate data transmission [2], caching popular contents such as video and social media that are repeatedly requested by users near the wireless network edge, e.g., in the BSs' local memories, has recently become one of the research focuses. Most approaches in the literature assume limited cache storage and perfect knowledge of time-invariant content popularity distribution at BSs, and then design content delivery [2], content placement [3] as well as joint content placement and delivery [4] strategies in various network s-cenarios. Their assumption of content popularity distribution known in advance at the BSs, nevertheless, is not realistic in practical scenarios. In recent years, employing machine learning techniques to estimate the unknown content popularity and proactively cache the popular contents at the BSs prior to users' content requests, has attracted the attention of researchers. The authors in [5], [6] propose multi-armed bandit (MAB)-based caching strategies to estimate the unknown but time-invariant content popularity distribution. The authors in [7] introduce a regret learning based per-BS caching strategy to learn spatio-temporal traffic demands and capture local and global content popularity trade-off. However, the aforementioned works simply ignore the fact that the contents can be dynamic over time: new contents are constantly introduced to the system and their popularity distributions may change over time. The authors in [8] and [9] focus on the cache placement design for dynamic content library. However, the authors in [8] propose an ON-OFF traffic model under the assumption that the arriving content request processes at different caches are temporal correlated. [9] adopts both global and local caches and proposes an age based threshold policy taking into account both the frequency of requests and the content age.

In contrast to the existing caching designs that assume stationary content library and/or time-invariant content popularity, this paper considers a non-stationary content library, where the content popularity may change over time and locations. We propose a three-phase procedure at different time scales for joint content placement and delivery at a BS. Phase I is the BS's main cache update phase at off-peak times, where the cache unit is updated from the content server. Phase II is the BS's flexible cache update phase during peak traffic hours, where the flexible cache can be more frequently updated with users' cached new contents. Phase III is the content delivery phase at individual time instances, where the content demand is satisfied from either the BS's local cache or the content server with different serving rewards. To adaptively track the spatio-temporal variations of users' content demands, a user-aided caching algorithm based on non-stationary bandit principles is developed to sequentially optimize the caching policy at the BS, with the objective of maximizing the average weighted network utility.

## II. SYSTEM MODEL AND PROBLEM FORMULATION

### A. *System Scenario and Assumptions*

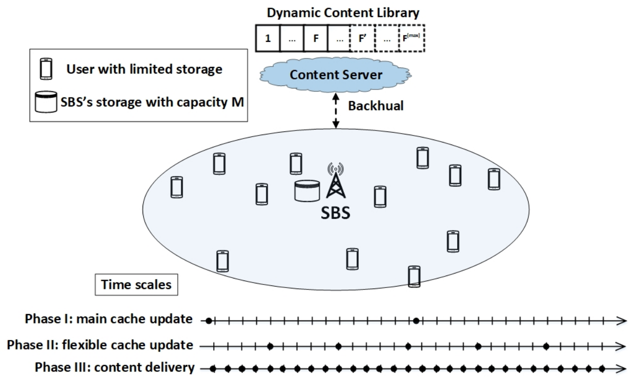
Dynamic Content Library
1
...
F
...
F'
...
F^[max]
User with limited storage
SBS's storage with capacity M
Content Server
Backhual
SBS
Time scales
Phase I: main cache update
Phase II: flexible cache update
Phase III: content delivery

Fig. 1. Illustration of the system scenario and three-phase procedure at different time scales for joint cache placement and content delivery.

As illustrated in Fig. 1, we consider a cache-enabled BS with storage capacity of $M$ that serves its $N_u$ users, indexed by $\mathcal{L}_u = \{1, \cdots, N_u\}$, based on perfect knowledge of channel state information. The BS is connected to the content server via capacity-limited backhaul link and attempts to offload backhaul traffic as much as possible via prefetching popular data from the core network during off-peak traffic hours and delivering to its users at peak times. Let the time horizon be divided into discrete time slots and indexed as $\mathcal{T} = \{1, \cdots, T\}$.

*1) Non-stationary Content Library:* We consider a non-stationary finite content library at the content server, where new contents are constantly introduced into the system. Let us denote by $F$ and $F^{[\text{max}]}$, respectively, the initial and the maximum numbers of contents in the content library, and denote by $\mathcal{F}^t = \{1, \cdots, F'\}$ the content library with individual content sizes of $\{m_f\}_{f \in \mathcal{F}^t}$ at the $t$-th time slot, $t \in \mathcal{T}$, where $F'$ indicates the current number of contents in the library. It is evident that $F \leq F' \leq F^{[\text{max}]}$. In case the content library has $F' = F^{[\text{max}]}$ available contents at time slot $t$, whilst the new contents are continuously emerging with time, the least frequently used contents at the content server will be evicted and replaced by the newly emerged contents. Note that our considered dynamic content library naturally leads to the time-varying content popularity, which is unknown to the BS.

*2) New Contents at Users and Cache Unit at the BS:* Each user has a local cache memory and can only cache one content at each time. At the end of each time slot, each user randomly updates its cache memory with one of its requested contents. In addition, newly emerged contents may be cached at some random users either via being generated by the local users themselves, or by being brought in through other networks such as wireless local area network. The users are motivated to upload these potentially popular new contents to the BS for the incentive payment. From the BS's perspective, let us denote by $\kappa^{[\text{user}]} \in [0, 1]$ the per-unit per-time-slot equivalent cost of the users' uploading incentives, which covers various costs such as the cost for allocating extra bandwidth for users' contents uploading. To fully exploit users' cached contents, a portion with capacity of $\xi M$, $\xi < 1$ of the BS's cache unit is allocated as the flexible cache memory that can be timely updated from users' caches, whilst the remainder of the cache unit will be updated at a more infrequent pace, for instance, from the content server during off-peak traffic hours.

*3) Users' Content Demands:* At each time slot $t$, $t \in \mathcal{T}$, user $u$, $u \in \mathcal{L}_u$, may request up to $N_f^u$ number of different contents that are not cached by itself from the BS according to some content popularity distribution $\{\theta_{u,f}^t\}_{f \in \mathcal{F}^t}$ that is unknown to the BS. In addition to the temporal variability of the content popularity, we further consider the spatial diversity of the content popularity distribution among users, i.e., different content preferences at users, and denote by $\Delta_u$ the circular shift of the content popularity distribution at user $u$. Let the content demands of user $u$ at time $t$ be denoted by $\mathbf{d}_u^t = \{d_{u1}^t, \cdots, d_{uf}^t, \cdots, d_{uF'}^t\}$, where the binary scalar $d_{uf}^t \in \{0, 1\}$ indicates whether or not the content $f$ is requested by user $u$ at time $t$, and $\|\mathbf{d}_u^t\|_0 \leq N_f^u$.

*4) Three-Phases Procedure:* We consider a three-phase procedure at different time scales of $\tau^{[\text{main}]}$, $\tau^{[\text{flex}]}$ and $t \in \mathcal{T}$, for joint cache placement and content delivery in this paper.

- In Phase I, i.e., for every $\tau^{[\text{main}]}$ time slots, the cache placement policy will be designed at the BS and the contents will be dispatched from the content server via backhaul links to update BS's cache unit.
- In Phase II, i.e., for every $\tau^{[\text{flex}]}$ time slots, $N_f^{[\text{new}]}$ number of new contents are added to the content server and might be cached by some random users. Each user broadcasts its cached content directory to the BS, and the BS will then make a decision on whether or not to update its flexible cache with the users' cached contents.
- In Phase III, i.e., at each time slot $t$, $t \in \mathcal{T}$, the BS satisfies the instantaneous content requests of its users, i.e., $\{\mathbf{d}_u^t\}_{u \in \mathcal{L}_u}$, either by directly from its cache unit or by retrieving from the content server with different per-unit gross serving gains of $\pi^{[\text{cache}]}$ and $\pi^{[\text{server}]}$, respectively. To take into account the backhaul traffic offloading as well as to fully utilize the BS's cache unit, let us assume $\pi^{[\text{cache}]}$, $\pi^{[\text{server}]}$ $\in [0, 1]$ and $\pi^{[\text{server}]} < \pi^{[\text{cache}]}$.

*5) Content Caching and Content Delivery:* Let us define the binary vector $\mathbf{c}_t^{[\text{p}]} = \{c_f^t \in \{0, 1\}, \ \forall f \in \mathcal{F}^t\}$ as the content placement policy at time $t$, $t \in \mathcal{T}$, where $c_f^t = 1$ and $c_f^t = 0$, respectively, indicate that the content $f$ is cached and is not cached at the BS. This caching policy will be designed in Phase I for BS's main cache update, and might be updated in Phase II for the portion of flexible cache memory. Let us denote the content uploading policy for user $u$ at time $t, t \in \mathcal{T}$ as $\mathbf{c}_t^{[\text{u}]} = \{c_{u,f}^t \in \{0, 1\}, \sum_{u \in \mathcal{L}_u} c_{u,f}^t \leq 1, \ \forall u \in \mathcal{L}_u, f \in \mathcal{F}^t\}$, where $c_{u,f}^t \in \{0, 1\}$ represents whether or not the content $f$ is uploaded from user $u$ to the BS. It is obvious that $c_f^t \geq \sum_{u \in \mathcal{L}_u} c_{u,f}^t$. Then, the net serving gain of the BS caching content $f$ and serving user $u$ at time slot $t, t \in \mathcal{T}$, can be defined as

$$G^t_{f,u} = m_f d^t_{uf}(\pi^{[\text{cache}]} c^t_f - \sum_{u' \neq u, u' \in \mathcal{L}_u} \kappa^{[\text{user}]} c^t_{u',f}), \quad \forall t \in \mathcal{T}, u \in \mathcal{L}_u, f \in \mathcal{F}^t, \tag{1}$$

where the terms at the right hand side of (1) denote, respectively, the gross serving gain for the BS serving user $u$ with content $f$ directly from its local cache, and the per-time-slot equivalent cost if the content $f$ cached at the BS is uploaded from the other local users. Let us denote by $\mathbf{c}^{[\text{s}]}_t = \{s^t_f \in \{0,1\},\ \forall f \in \mathcal{F}^t\}$ the content retrieving policy at time $t$, where $s^t_f \in \{0,1\}$ denotes whether or not the content $f$ is retrieved from the content server by the BS at time $t$. The user demand of content $f,\ f \in \mathcal{F}^t$ at each time slot $t,\ t \in \mathcal{T}$ will be satisfied by serving either from the content server or from the BS's cache, as

$$s^t_f + c^t_f = 1,\ \forall f \in \mathcal{F}^t, \sum_{u \in \mathcal{L}_u} d^t_{uf} \neq 0. \tag{2}$$

The net serving gain of the BS serving user $u$ by retrieving content $f$ from the content server at time slot $t$, is given by

$$G^{[\text{s}],t}_{f,u} = m_f d^t_{uf} \pi^{[\text{server}]} s^t_f, \forall t \in \mathcal{T}, u \in \mathcal{L}_u, f \in \mathcal{F}^t. \tag{3}$$

Let us denote by $\Psi^t_u$ the channel gain between the BS and user $u$ at time slot $t, t \in \mathcal{T}$ and denote by $P^{[\text{Tx}]}_u$ the transmit power from the BS to user $u$. With normalized bandwidth, the instantaneous downlink data rate for user $u$ at time $t$, can be expressed as

$$R^t_u = \log_2(1 + \frac{P^{[\text{Tx}]}_u \Psi^t_u}{\sum\limits_{u' \in \mathcal{L}_u, u' \neq u} P^{[\text{Tx}]}_{u'} \Psi^t_u + \sigma^2_u}), \forall u \in \mathcal{L}_u, \tag{4}$$

where $\sigma^2_u$ is the variance of the additive white Gaussian noise at user $u$. At each time slot $t,\ t \in \mathcal{T}$, the content demand $f$ of user $u$ is satisfied either from the BS's cache unit with the net serving gain of $G^t_{f,u}$, or from the content server with the net serving gain of $G^{[\text{s}],t}_{f,u}$. The instantaneous serving reward, i.e., the weighted data rate, for serving user $u$ with content $f$ at time slot $t,\ t \in \mathcal{T}$, can be defined as

$$\mathcal{R}(d^t_{uf}) = (G^t_{f,u} + G^{[\text{s}],t}_{f,u}) R^t_u,\ \forall u \in \mathcal{L}_u, f \in \mathcal{F}^t, t \in \mathcal{T}. \tag{5}$$

Note that (5) jointly considers the backhaul traffic offloading as well as the content delivery by assigning different weighting factors, i.e., $G^t_{f,u}$ or $G^{[\text{s}],t}_{f,u}$, to the transmission data rate. This serving reward will be useful in designing cache placement policy as well as content delivery policy in the subsequent sections.

### *B. Problem Formulation*

The objective of the BS is to design a joint caching policy $\{\mathbf{c}^{[\text{p}]}_t, \mathbf{c}^{[\text{u}]}_t, \mathbf{c}^{[\text{s}]}_t\}$ to offload as much traffic as possible from the content server, whilst ensuring the quality of service (QoS) of the users. Hence, the problem of interest can be formulated as the maximization of the long-term average reward of the network, i.e., the average weighted network utility, as

$$\max_{\{\mathbf{c}^{[\text{p}]}_t, \mathbf{c}^{[\text{u}]}_t, \mathbf{c}^{[\text{s}]}_t\}} \left\{ \lim_{T \to \infty} \frac{1}{T} \sum_{t \in \mathcal{T}} \sum_{f \in \mathcal{F}^t} \sum_{u \in \mathcal{L}_u} \mathcal{R}(d^t_{uf}) \right\} \tag{6}$$

$$\begin{aligned}
\text{s.t.} \quad & \text{C1}: \sum_{f \in \mathcal{F}^t} m_f c^t_f \leq M,\ \forall t \in \mathcal{T}, \\
& \text{C2}: s^t_f + c^t_f = 1,\ \forall f \in \mathcal{F}^t, t \in \mathcal{T}, \sum_{u \in \mathcal{L}_u} d^t_{uf} \neq 0, \\
& \text{C3}: \sum_{u \in \mathcal{L}_u} c^t_{u,f} \leq 1,\ \forall f \in \mathcal{F}^t, t \in \mathcal{T}, \\
& \text{C4}: c^t_f \geq \sum_{u \in \mathcal{L}_u} c^t_{u,f},\ \forall f \in \mathcal{F}^t, t \in \mathcal{T}, \\
& \text{C5}: c^t_f \in \{0,1\},\ \forall f \in \mathcal{F}^t, t \in \mathcal{T}, \\
& \text{C6}: c^t_{u,f} \in \{0,1\},\ \forall f \in \mathcal{F}^t, t \in \mathcal{T}, \\
& \text{C7}: s^t_f \in \{0,1\},\ \forall f \in \mathcal{F}^t, t \in \mathcal{T},
\end{aligned}$$

where the constraint C1 represents that the total size of the cached contents cannot exceed the capacity of the BS's cache unit. C2 guarantees that the users' content demands at each time slot will be satisfied either from the BS's cache unit or from the content server. C3 denotes that the content $f$ can be uploaded from at most one local user to the BS. C4 indicates that the content uploaded from the users will be stored at the BS's cache unit. C5 - C7 specify that the joint caching policy $\{c^t_f\}_{f \in \mathcal{F}^t, t \in \mathcal{T}}, \{c^t_{u,f}\}_{f \in \mathcal{F}^t, t \in \mathcal{T}}, \{s^t_f\}_{f \in \mathcal{F}^t, t \in \mathcal{T}}$ are binary variables, respectively.

## III. USER-AIDED LEARNING-BASED CONTENT CACHING STRATEGY

In this section, the problem in (6) will be gradually solved through a user-aided reinforcement learning-based three-phase procedure at different time scales. The main objective of the proposed caching algorithm is to progressively optimize the joint caching policy $\{\mathbf{c}^{[\text{p}]}_t, \mathbf{c}^{[\text{u}]}_t, \mathbf{c}^{[\text{s}]}_t\}$ time slot by time slot, such that the long-term average reward, i.e., the average weighted network utility, can be maximized. More specifically, the learning processes in Phase I and Phase II of the proposed algorithm, respectively, aim at tracking the variations in user demands in order to design $\{\mathbf{c}^{[\text{p}]}_t\}$ in constraints C1, C4 and C5 of problem (6) at every $\tau^{[\text{main}]}$ time slots, and design $\{\mathbf{c}^{[\text{u}]}_t\}$ in constraints C3 and C6 at every $\tau^{[\text{flex}]}$ time slots. The content retrieving and delivery policy, i.e., $\{\mathbf{c}^{[\text{s}]}_t\}$ in constraints C2 and C7, will be satisfied at each individual time slot $t,\ t \in \mathcal{T}$, in Phase III of our proposed algorithm.

In the sequel, the proposed user-aided learning-based caching algorithm is introduced to offload the backhaul traffic as much as possible while ensuring the QoS of the users. Let us consider a non-stationary generalization of the MAB problem that models a system of $F'$ actions whose expected rewards are independent and identically distributed over time with unknown means and may vary across time. The objective of the agent in the MAB problem is to maximize the accumulated reward over time via exploring the environment to find profitable actions, while exploiting current knowledge to make the empirically best decisions among a set of actions [10]. The caching problem investigated in this paper can be modelled as a non-stationary bandit problem, where the BS can be regarded as the agent and $F'$ actions correspond to the current library of $F'$ contents at the content server, whose content popularity evolves over time. As per (5), it is obvious that in order to maximize the long-term average reward of the network, the user will be served from BS's local cache with the top priority, and by retrieving from the content server with the least priority. Hence, the associated reward of action $f, f \in \mathcal{F}^t$ can be defined as the aggregated data rate for satisfying users' content demand of $f$, i.e., $\sum_{u \in \mathcal{L}_u} R_u^t d_{uf}^t m_f$.

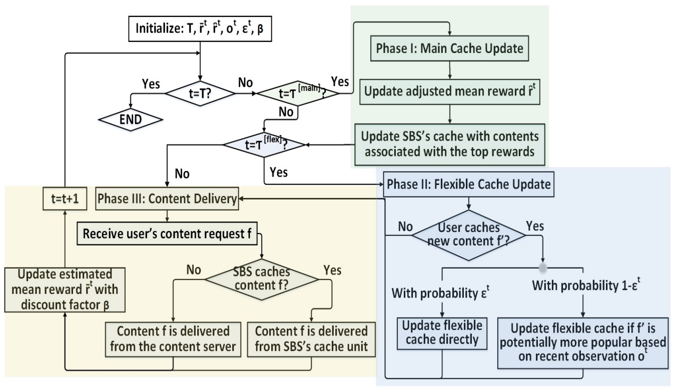
Initialize: T, r̄^t, r̂^t, o^t, ε^t, β
t=T?
Yes
No
END
t=T^[main]?
Yes
No
Phase I: Main Cache Update
Update adjusted mean reward r̂^t
Update SBS's cache with contents associated with the top rewards
t=T^[flex]?
No
Yes
t=t+1
Phase III: Content Delivery
Receive user's content request f
Update estimated mean reward r̄^t with discount factor β
SBS caches content f?
No
Yes
Content f is delivered from the content server
Content f is delivered from SBS's cache unit
Phase II: Flexible Cache Update
User caches new content f'?
No
Yes
With probability ε^t
With probability 1-ε^t
Update flexible cache directly
Update flexible cache if f' is potentially more popular based on recent observation o^t

Fig. 2. Flowchart of the proposed caching algorithm.

**Algorithm 1** *User-aided learning-based caching algorithm*

```
1:  Initialize: time horizon $T$, temporary reward matrix $\mathbf{r} = \{r_f^t, \forall f \in \mathcal{F}^t, t \in \mathcal{T}\}$, estimated mean reward vector $\bar{\mathbf{r}}^t = \{\bar{r}_f^t, \forall f \in \mathcal{F}^t\}$, adjusted mean reward vector $\hat{\mathbf{r}}^t = \{\hat{r}_f^t, \forall f \in \mathcal{F}^t\}$, recent content demand observation vector $\mathbf{o}^t = \{o_f^t, \forall f \in \mathcal{F}^t\}$, exploration/exploiation trade-off $\epsilon^t$, discount factor $\beta$, $\rho^t$, time slot count $t = 1$.
2:  REPEAT
3:  If $t = \tau^{[\text{main}]}$, Phase I. Main Cache Placement
4:    Update $\hat{\mathbf{r}}^t$, as $\hat{r}_f^t = \bar{r}_f^t + \rho^t \sqrt{\frac{2\log\ n_t}{T_f}}$, where $\rho^t \propto \sum_{t'=1}^{t} \sum_{u \in \mathcal{L}_u} \beta^{(t-t')} m_f d_{uf}^{t'}$, $T_f = \sum_{t'=1}^{t} \beta^{(t-t')} I_{\{c_f^{t'}=1\}}$ is the discounted number of times content $f$ has been cached by the BS and $n_t = \sum_{f \in \mathcal{F}^t} T_f$.
5:    Update BS's cache unit with contents associated with the top reward values based on $\hat{\mathbf{r}}^t$ in a sorted order, such that $\sum_{f \in \mathcal{F}^t} m_f c_f^t \leq M$.
6:  End If
7:  If $t = \tau^{[\text{flex}]}$, Phase II. Flexible Cache Update
8:    Observe its local users' cached content directory.
9:      If A new content $f'$ is cached by its local user $u$
10:     -with probability $\epsilon^t$, update BS's flexible cache directly;
11:     -with probability $1 - \epsilon^t$, update recent content request vector $\mathbf{o}^t$, and update the flexible cache only if $o_{f'}^t$ is larger than that of the current content in flexible cache.
12:     End if
13: End If
14: Phase III. Content Delivery at Each Time Slot $t$
15:   Receive user demands $\{\mathbf{d}_u^t\}_{u \in \mathcal{L}_u}$.
16:     If the content $f \in \mathcal{F}^t$ requested by user $u, \forall u \in \mathcal{L}_u$ is cached at the BS, i.e., $d_{uf}^t c_f^t = 1$
          -Serve user $u$ directly from its local cache.
        Else
          -Serve user $u$ by fetching from the content server.
        End if
17: Update $\mathbf{r}$ as $r_f^t = \sum_{u \in \mathcal{L}_u} R_u^t d_{uf}^t m_f,\ \forall f \in \mathcal{F}^t$.
18: Update $\bar{\mathbf{r}}^t$ as $\bar{r}_f^t = \frac{\sum_{t'=1}^{t} r_f^{t'} \beta^{(t-t')}}{\sum_{t'=1}^{t} \beta^{(t-t')}}, \forall f \in \mathcal{F}^t$.
19: Update $t = t + 1$.
20: UNTIL $t = T$
```

The details of the proposed learning-based caching strategy are described in Algorithm 1 and Fig. 2, where a three-phase procedure at different time scales for joint content caching and delivery is proposed and explained below:

*1) Phase I:* For every $\tau^{[\text{main}]}$ time slots, a perturbation procedure is applied to the estimated mean reward $\bar{\mathbf{r}}^t$ according to step 4 in Algorithm 1. Such adjustment implements a trade-off between exploring the contents that are not frequently cached and may yield a better accumulated reward in the future by artificially increasing their estimated mean reward, and exploiting the contents associated with the top mean rewards so far based on the past observations. Due to the fact that the content library is massive and evolutive, the standard upper confidence bound (UCB) algorithm [10] may not be able to catch up with such rapid variations. Hence, we modified the UCB algorithm by adding a discount factor $\beta$ as well as introducing a weighting factor $\rho^t$ that is proportional to the long-term discounted content demands. Such modification will encourage the BS to cache those contents that are frequently requested in recent times but are not cached that often.

*2) Phase II:* For every $\tau^{[\text{flex}]}$ time slots, the contents associated with the lowest reward in BS's flexible cache will be replaced by the new contents cached at the users accordingly. To be specific, with the probability of $\epsilon^t$, we explore new contents cached by local users via updating BS's flexible cache directly. With the probability of $1 - \epsilon^t$, the flexible cache will only be updated with the more potentially popular new contents, based on the recent content demand observation $\mathbf{o}^t = \{o_f^t, \forall f \in \mathcal{F}^t\}$, where $o_f^t$ is the recent observation of the request number of content $f$.

*3) Phase III:* At each time slot $t$, $t \in \mathcal{T}$, users' instantaneous content requests are received and satisfied according to the delivery policy in step 16 of Algorithm 1. Then, the rewards of the individual requested contents are updated at the BS.

## IV. Simulation Results

Consider a downlink network where a BS serves $N_u = 8$ randomly deployed users. The non-stationary content library at the server has an the initial library of $F = 100$ contents and has a finite capacity of $F^{[\max]} = 150$. Each content size varies as $m_f \in \{1, 2\}$ and the capacity of BS's cache unit is $M = 15$ with $\xi = 0.2$. The Phase I of BS's main cache update phase occurs at every $\tau^{[\text{main}]} = 9$ time slots and the Phase II of flexible cache update phase is executed every $\tau^{[\text{flex}]} = 3$ time slots, where $N_f^{[\text{new}]} = 3$ new contents will be added to the content library and might be cached by maximally 3 random users. The per-unit gross gains for the BS to serve its users directly from local cache unit, and by retrieving contents from the content server are, respectively, $\pi^{[\text{local}]} = 1$ and $\pi^{[\text{server}]} = 0.3$, whilst the per-unit per-time-slot equivalent cost of users' uploading incentives is $\kappa^{[\text{user}]} = 0.2$. We adopt the commonly used Zipf distribution with exponent of $\gamma^t = 2$ [5], given by

$$\theta^t_{u,f\to\Delta_u} = \frac{f^{-\gamma^t}}{\sum_{f=1}^{F'} f^{-\gamma^t}}, \; f \in \mathcal{F}^t, \tag{7}$$

to model the actual content preferences at the individual users that is unknown to the BS, where the circular shifts are $\{\Delta_u\} = \{0, 2, \cdots, 14\}$. Each user has up to $N_f^u = 3$ number of different content requests at each time slot $t$ according to the actual content popularity distribution $\{\theta^t_{u,f}\}_{f\in\mathcal{F}^t}$. The channel gain is modelled as $\Psi^t_u = \mathbf{h}^{t^2}_u G_a L_u \sigma^2_F e^{-0.5\frac{(\sigma_s \ln 10)^2}{100}}$ [11], where $\mathbf{h}^t_u \sim \mathbb{CN}(0, 1)$, $G_a = 10$ dBi is the antenna gain, $L_u(\text{dB}) = 128.1+37.6\log_{10}(\ell)$ [12] denotes the path loss over a distance of $\ell$ km and $\sigma_s = 8$ dB is log-normal shadowing standard deviation. The other simulation parameters are described, unless otherwise stated, as follows: identical transmit power $P_u^{[\text{Tx}]} = 23$ dBm for all users, exploration/exploitation trade-off $\epsilon^t = 0.8$ and discount factor $\beta = 0.9$. Three designs that consider no user cache, i.e., no Phase II evolved, are chosen as the benchmark designs, namely, the algorithm in [5], the algorithm in [6] and the random caching design. We further employ a user-aided popularity-known caching design to show the performance upper bound. For fair comparison, identical constraints have been applied to all strategies. These benchmark schemes are introduced as follows.

*1) Benchmark design in [5]:* The benchmark design in [5] estimates the content popularity distribution at the BS via the standard UCB algorithm. The estimated mean reward is given by $\bar{r}^t_f = \frac{\hat{o}^t_f}{\hat{n}^t_f}$, where $\hat{o}^t_f$ and $\hat{n}^t_f$ denote, respectively, the long-term observation of the total request number of content $f$ up to time $t$, and the total number of times the requests of content $f$ are satisfied directly from BS's cache.

*2) Benchmark design in [6]:* The benchmark design in [6] learns the content popularity distribution at the BS via the combinatorial UCB algorithm [10] based on the past observation of user content demands. The instantaneous reward associated with file $f$ is defined as $r^t_f = \sum_{u\in\mathcal{L}_u} m_f d^t_{uf}$, and the estimated mean reward is given by $\bar{r}^t_f = \frac{\sum_{t'=1}^{t} r^{t'}_f}{T_f}$, where $T_f$ is the number of times content $f$ has been cached by the BS. For fair comparison, we have embedded the benchmark designs in [5] with a sliding-window of 100 time slots [13], which relies on a local empirical average of the observed rewards and emphasizes more on the recent observations, so as to better adapt to our considered scenario.

*3) Random caching design:* A random caching design that randomly caches contents at the BS in Phase I regardless of user content demand, is employed to indicate the performance lower bound.

*4) Popularity-known caching design:* The user-aided popularity-known caching design has perfect knowledge of actual content popularity distribution $\{\theta^t_{u,f}\}$ in advance at the BS. It allows the BS to update its cache unit in Phase I from the content server and to update its flexible cache in Phase II with local users' cached contents based on prior knowledge of $\{\theta^t_{u,f}\}$ at the individual users.

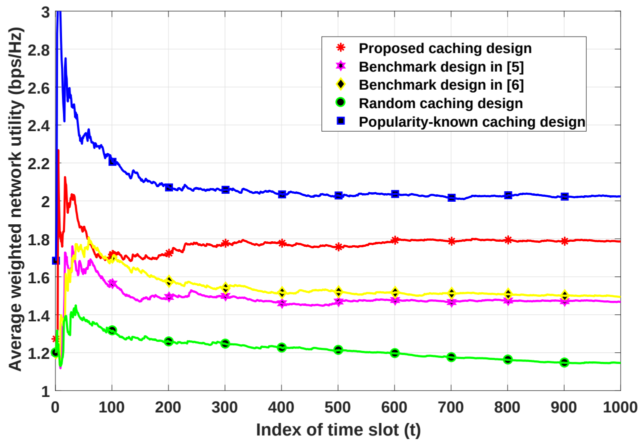

Fig. 3. Comparison of average reward for various strategies.

Fig. 3 presents the comparison of the average reward, i.e., the weighted network utility, of the proposed caching design against the benchmark designs in [5], [6], the random caching design and the popularity-known caching design at the individual time slots. As can be observed from Fig. 3, the proposed design outperforms all benchmark designs and the performance gap between the proposed design and the popularity-known design narrows with the increasing number of time slots. This is due to the fact that the proposed strategy takes into account the spatio-temporal variations in users' content demands, and maximally benefits from users' caches through timely updating the BS's flexible cache with the potentially more popular new contents cached by users in addition to the routine off-peak main cache update from the content server, thus provides a better adaptation to the user content demand uncertainties. In contrast, the performance of all the benchmark designs degrades with increasing time slots, since neither the evolution of the content library nor the potential of user cache has been taken into consideration in the nature of their designs. Furthermore, it is obvious that the random caching design has the worst performance among all of the strategies, as it simply caches contents randomly without involving any learning process to estimate the unknown variations in user demands.

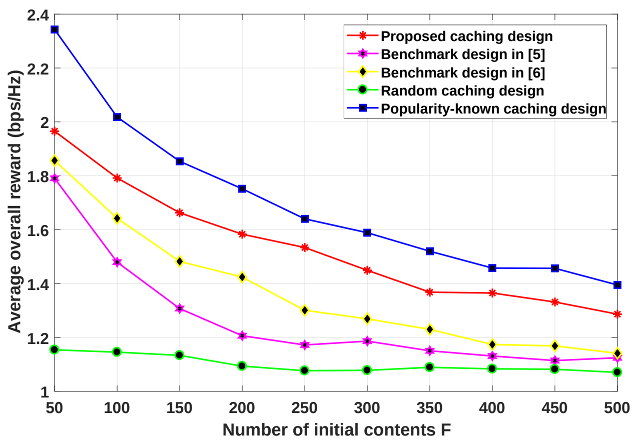

Fig. 4. Comparison of average overall rewards of various strategies for different number of initial contents $F$.

Fig. 4 illustrates the average overall reward of the proposed caching design against the benchmark designs in [5], [6], the random caching design and the popularity-known caching design for different number of initial contents $F$ at the content server. The average overall reward is obtained by averaging over $T = 1000$ time slots for each value of $F$, and the finite capacity of content library is set as $F^{[\max]} = F + 50$. As can be observed from the figure, the average performance of all strategies degrades with increasing value of $F$, since a larger content library naturally results in more users' content requests being satisfied by the content server. It is worth noticing that the gap between the proposed design and the benchmark designs in [5], [6] increases with increasing value of $F$, whilst, the increment of $F$ has less impacts on the random caching design. This is due to the fact that the standard UCB algorithm adopted in [5], [6] fails to provide quick adaptation to the variations when the content library, i.e., the number of actions, is massive.

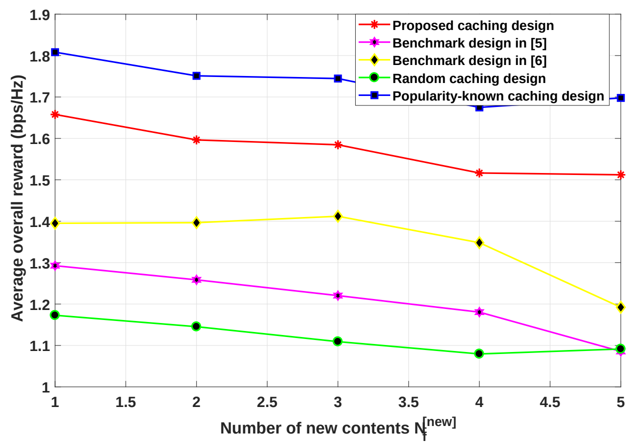

Fig. 5. Comparison of average overall rewards of various strategies for different number of emerged new content $N_f^{[\mathrm{new}]}$.

Fig. 5 compares the average overall rewards of all strategies for various number of new content $N_f^{[\mathrm{new}]}$ when the number of initial library of contents is $F = 200$. One may conclude from the figure that though performing better than the random caching design, the performance of benchmark designs in [5] and [6] decreases with increasing value of $N_f^{[\mathrm{new}]}$, and drops significantly when $N_f^{[\mathrm{new}]} \geq 4$. The reason is that neither the time-varying content popularity nor the non-stationary content library has been taken into account in [5] and [6], and, for a larger value of $N_f^{[\mathrm{new}]}$, it is more challenging for them to adapt to the rapid variations in users' content demands.

## V. Conclusion

The joint caching placement and content delivery problem at a BS is studied in this paper, where the time-varying content popularity is unknown *a priori* and the content library evolves over time. To keep track of the dynamic content library, a portion of cache unit at the BS is assigned as the flexible cache that can be timely updated with the contents cached by users in addition to its routine off-peak main cache update from the content server. Considering three phases of different time scales for main cache update, flexible cache update and content delivery, a user-aided learning-based caching algorithm is proposed to maximize the long-term average weighted utility of the network. Simulation results confirm the superiority of the proposed foresighted strategy in achieving a significant performance improvement over various benchmark designs.

## References

[1] Cisco, "Cisco Visual Networking Index: Forecast and Methodology, 2016-2021", White Paper, Sep. 2017.

[2] L. Li, G. Zhao and R. S. Blum, "A Survey of Caching Techniques in Cellular Networks Research Issues and Challenges in Content Placement and Delivery Strategies," *IEEE Communications Surveys & Tutorials*, vol.20, no.3, pp.1710-1732, Mar. 2018.

[3] E. Bastug, M. Bennis and M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Communications Magazine*, vol.52, no.8, pp.82-89, Aug. 2014.

[4] Z. Zhao *et al.*, "Cluster Content Caching: An Energy-Efficient Approach to Improve Quality of Service in Cloud Radio Access Networks," *IEEE JSAC*, vol.34, no.5, pp.1207-1221, May 2016.

[5] J. Song, M. Sheng, T. Q. S. Quek, C. Xu and X. Wang, "Learning-Based Content Caching and Sharing for Wireless Networks," *IEEE Trans. on Communications*, vol.65, no.10, pp.4309-4324, Oct. 2017.

[6] P. Blasco and D. Gunduz, "Learning-based optimization of cache content in a small cell base station," *IEEE ICC*, pp.1897-1903, Jun. 2014.

[7] S. Tamoor-ul-Hassan, S. Samarakoon, M. Bennis, M. Latva-aho and C. S. Hong, "Learning-Based Caching in Cloud-Aided Wireless Networks," *IEEE Communications Letters*, vol.22, no.1, pp.137-140, Jan. 2018.

[8] M. Garetto, E. Leonardi, and S. Traverso, "Efficient Analysis of Caching Strategies under Dynamic Content Popularity," *IEEE INFOCOM*, Apr. 2015, pp. 2263-2271.

[9] M. Leconte *et al.*, "Placing dynamic content in caches with small population," *IEEE INFOCOM*, Apr. 2016, pp. 1-9.

[10] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction," Cambridge MA, US: The MIT press, 2017.

[11] W. N. S. F. Wan Ariffin *et al.*, "Sparse Beamforming for Real-time Resource Management and Energy Trading in Green C-RAN", *IEEE Transactions on Smart Grid*, vol.8, no.4, pp.2022-2031, Jul. 2017.

[12] 3GPP, "TR 36.814 V9.2.0: Further Advancements for E-UTRA Physical Layer Spects (Release 9)," *Available: http://www.3gpp.org*, Mar. 2017.

[13] A. Garivier, E. Moulines, "On Upper-Confidence Bound Policies for Non-Stationary Bandit Problems," arXiv:0805.3415, May 2008.