An approximate dynamic programming approach for collaborative caching

Xinan Yang[a] and Nikolaos Thomos[b]

[a]xyangk@essex.ac.uk, Department of Mathematical Sciences, University of Essex, UK;
[b]nthomos@essex.ac.uk, School of Computer Science and Electronic Engineering, University of Essex, UK.

ARTICLE HISTORY
Compiled April 27, 2020

ABSTRACT
In this paper, online collaborative content caching in wireless networks is studied from network economics point of view. The cache optimization problem is first modelled as a finite horizon Markov Decision Process that incorporates an autoregressive model to forecast the evolution of the content demands. The complexity of the problem grows exponentially with the system parameters, and even though a good approximation to the cost-to-go can be found, the single-stage decision problem is still NP-hard. To deal with cache optimization in industrial-size networks, a novel methodology called *rolling horizon* is proposed that solves the dimensionality of the problem by freezing the cache decisions for a short number of periods to construct a value function approximation. Then, to address the NP-hardness of the single-stage decision problem, two simplifications/reformulations are examined: (a) to limit the number of content replicas in the network and (b) to limit the allowed content replacements. The results show that the proposed approach can reduce the communication cost by over 84% compared to that of running Least Recently Used (LRU) updates on offline schemes in collaborative caching. The results also shed light on the trade-off between the efficiency of the caching policy and the time needed to run the online cache optimization algorithm.

KEYWORDS
Collaborative caching, online/offline caching, popularity dynamics, finite horizon MDP, approximate dynamic programming

1. Introduction

During the last few years, there was an explosion of data traffic in cellular networks. This increase puts pressure on network operators' infrastructure and renders inefficient the current model according to which the base stations (BS) receive the requested content through the core network using expensive bandwidth-limited backhaul links. The backhaul links may become congested because of the increased data traffic, and this may result in users experiencing excess delays and low Quality of Experience (QoE). High QoE can be preserved by densifying the network of BSs, *i.e.*, installing a bigger number of BS; however, this solution does not scale well with the number of wireless devices. A more efficient solution would be to exploit the spatial diversity of users' requests. This can be done by caching popular content at BSs' caches so that it is closer to the end-users. This helps to decrease the load of the backhaul links and combat download delays. However, such caching solutions do not take full advantage of the correlation of content requests in different BSs. Enabling BSs collaboration reduces the number of content replicas cached at the BSs. Nevertheless, deciding on the optimal cache placement/eviction policies for a collaborative network is a complex problem. This calls for new collaborative caching algorithms that consider cache update schedules and solve large instances of the cache optimization problem.

In the wireless caching systems, the cache optimization problem is hard to solve due to the plethora of the requested contents, the dynamic nature of the requests and base stations have limited cache space. This problem becomes even more challenging as the vast majority of the generated data are video files, which have considerable size and strict delivery deadlines. Thus, even when base stations determine the optimal caching policy independently, the cache optimization problem is still NP-hard as it can be mapped to a knapsack problem (Martello and Toth 1990b). In practice, network operators can adopt simple cache update policies such as the Least Recently Used (LRU), Least Frequently Used (LFU), or other more advanced methods (Yang et al. 2019; Abad et al. 2019; Bharath et al. 2018; Muller et al. 2017), that are applied separately to each BS. These methods are intuitive and show good performance in case of independent BS cache optimization, but do not work well in collaborative caching because the optimal replacement decisions also depend on the network topology.

Collaborative wireless caching was first presented by Golrezaei et al. (2013) as an efficient way to exploit the spatial correlation of the requests when users can cache part of the content. Then, it was generalized in BSs networks by Poularakis et al. (2014, 2016); Khreishah et al. (2016); Maniotis et al. (2020). These offline caching schemes assume that the content popularity profile is known. They make content replacement decisions assuming that future content requests will follow the considered model. Obviously, the efficiency of these caching schemes depends heavily on the sufficiency of content request statistics that are used for parameter fitting and are sensitive to sudden changes in popularity. Online caching algorithms were proposed as a remedy to non-stationary demands (Li et al. 2016; Zhang et al. 2018; Yang et al. 2019; Abad et al. 2019; Chattopadhyay et al. 2018b; Neglia et al. 2018; Bharath et al. 2018; Chattopadhyay et al. 2018a; Gharaibeh et al. 2016; Muller et al. 2017; Saltarin et al. 2018). These schemes learn the optimal caching policies using recent content requests. Online caching schemes are classified in non-collaborative caching schemes (Zhang et al. 2018; Yang et al. 2019; Abad et al. 2019; Chattopadhyay et al. 2018b; Neglia et al. 2018; Bharath et al. 2018; Muller et al. 2017; Li et al. 2016), where caches decide independently content placement/eviction policies, and collaborative online caching schemes (Chattopadhyay et al. 2018a; Gharaibeh et al. 2016; Muller et al. 2017; Saltarin et al. 2018), where these policies are jointly decided. Besides the efficiency of online caching schemes, their main drawbacks are that: (a) they do not consider how mobile network operators (MNOs) update the cached content, i.e., in off-peak hours cache updates all the content can be potentially updated, and (b) they have high complexity, and hence can only solve small instances of the cache optimization problem.

Edge caching in wireless networks is revisited in this paper to address the above drawbacks. The considered network consists of small-scale base stations (SCBSs) with limited storage space that collaborate to accommodate users' requests for content. SCBSs are connected with neighbouring SCBSs through the core network via high-speed links. When a user issues a request for a content, if it is not cached in the SCBSs where the user lies, the request is forwarded to the neighbouring SCBS that belongs to the shortest path connecting the former SCBS with the SCBS which has the content. This path can be built using, for example, content advertisement messages (Marandi et al. 2017) that inform SCBSs about which contents are available to other SCBSs incurring low overhead. In case none of the SCBS has the requested content or the cost to acquire it from the macro-cell base station (MCBS) is lower, the content is retrieved from the MCBS. In order to decide from where to retrieve the content, the cost of updating the content (this includes the communication cost) is considered. Once the content is located in an SCBS, the content is sent back to the user following the reverse path of the path followed when content was requested. The above content request model resembles the way information is requested in information-centric networks by Saltarin et al. (2018), however, the proposed solution is generic and applicable to any network following such a content request model. More details regarding the communication model are provided in Section 3.

The considered scenario has similarities with the one examined by Khreishah and Chakareski (2015); Gharaibeh et al. (2016), where offline caching solutions are presented. Different from these works, the focus here is online caching solutions. Cache content is placed free-of-charge at the SCBSs in off-peak hours and updated during the day according to the encountered content popularity dynamics considering the cost of update. The caching optimization problem is first modelled as a finite-horizon Markov Decision Process (MDP) that incorporates an auto-regressive model to forecast the evolution of the content demands. The problem grows exponentially with the system parameters, and even though a good approximation to the cost-to-go can be found, the problem is NP-hard. To allow the deployment of the proposed solution to industrial-size networks, a methodology that simplifies the solution process from two perspectives is presented. The dimensionality problem is addressed by the proposed *rolling horizon* method, which freezes the cache decisions for a short number of periods in order to construct a value function approximation. The performance of the proposed scheme is compared against several schemes with and without prediction. The results show that *rolling horizon* approach outperforms all other schemes. As the computational complexity of the problem can still be high, two simplifications of the single-stage decision-making problem are studied: (a) to limit the number of content replicas in the network and (b) to limit the content updates. The results show that besides these simplifications, large gains over the comparison schemes are noticed, and the performance is still close to a theoretical lower bound. The contributions of this work are summarized as follows:

- a dynamic programming structure with finite horizon is proposed to solve the online cache optimization problem;
- the dimensionality of the problem is addressed with a rolling horizon approach. Further, two heuristics are proposed to resolve the NP-hardness of the cache updating problem that appears in every single stage;
- the proposed online caching algorithm is extensively evaluated and compared against state-of-the-art schemes.

The rest of the paper is organized as follows. In Section 2, related literature is briefly reviewed and in Section 3, the considered scenario is discussed. Then, the online cache optimization model is introduced in Section 4. In Section 5, the proposed solution based on rolling-horizon and simplification to cope with the dimensionality and NP-hardness in decision making are presented. The performance of proposed schemes is compared against several schemes in Section 6. Finally, conclusions are drawn in Section 7.

2. Related work

Collaborative offline wireless caching (Golrezaei et al. 2013) is an efficient mechanism that exploits the spatial correlation of the requests. BSs collaborate with wireless devices that offer their caching space to store popular content. Coded caching can further advance the performance of these systems, as content reuse and cache hit ratio increase, while content delivery delay decreases (Maddah-Ali and Niesen 2014). Poularakis et al. (2014, 2016); Khreishah et al. (2016) propose to use a large number of SCBSs to lower the data retrieval delays in mobile networks. The cached content in the SCBSs is decided centrally by the MNO. The SCBSs work in concert with an MCBS to deliver content that is not available to the SCBSs, which is requested from MCBS. Poularakis et al. (2014) firstly cast the problem as an unsplittable hard capacitated metric facility location problem and solve it by an approximation algorithm. Cache assisted delivery of scalable video data (Thomos et al. 2015) is studied by Poularakis et al. (2016) where multiple MNOs collaborate by offering part of their SCBSs cache space to other MNOs. It is shown that the cache optimization is an instance of the multiple-choice knapsack problem and admits a pseudo-polynomial time-optimal solution. The relation between partial caching and users' retention rate for video is investigated by Maggi et al. (2018).

To deal with non-stationary content requests, online caching was proposed. The optimal online cache policy can be determined using reinforcement learning algorithms (Zhang et al. 2018; Abad et al. 2019; Chattopadhyay et al. 2018b; Muller et al. 2017). When the freshness of the data is an important decision factor, caching algorithms should consider the age of information (Zhang et al. 2018; Abad et al. 2019). More efficient caching policies can be found by taking into account both the global and local content popularity (Chattopadhyay et al. 2018b). Chattopadhyay et al. (2018b) uses linear function approximation to lower the computational cost, which is inspired by the additive form of the overall cost. This approximation allows to solve larger instances of the problem, e.g., more files. Context-aware caching is proposed by Muller et al. (2017) for proactively deciding the cache allocation strategy. The content popularity is learned using a contextual multi-armed bandit algorithm that has guaranteed convergence. However, this algorithm still requires a significant number of iterations to learn data popularity. A linear regression model is used to estimate the content popularities by Yang et al. (2019). The problem is first formulated as a time-averaged hit rate maximization problem and then reformulated as a time-averaged regret minimization problem. Two algorithms based on simulated annealing are introduced by Neglia et al. (2018) to deal with non-stationary content requests. According to the work of Bharath et al. (2018), caches are updated when the offloading cost, i.e., the cost introduced when files are delivered through the backhaul, exceeds a threshold. An online popularity-aware caching scheme that refreshes the cached content according to the revealed content requests is presented by Li et al. (2016). This scheme considers a single cache network, and thus the algorithm is not appropriate for collaborative caches case. Exploiting BSs collaboration opportunities can reduce offloading cost (Chattopadhyay et al. 2018a; Saltarin et al. 2018) and/or reduce the network use (Gharaibeh et al. 2016). Collaboration opportunities may emerge due to the fact the coverage areas of BSs overlap (Chattopadhyay et al. 2018a), where a Gibbsian based sampling method is used to determine the optimal caching strategy. This enables sequential cache updates and policies are updated only when contents are delivered to users through the backhaul link. Collaboration between caches can also be achieved in content-centric networks (CCN) (Gharaibeh et al. 2016) by giving Incentives to Internet Service Providers (ISPs) to cache data for other ISPs. This solution, although it is of low complexity, cannot be trivially used in other wireless networks due to CCN's content demand model. Although the above mentioned online caching methods are efficient, they disregard that the cached contents in MNO handled networks are updated free-of-charge in the off-peak hours, and the majority of them have limitations in the size of the networks and the number of content they can handle.

3. System model

The network setting depicted in Fig. 1 is considered in this paper. This represents an MNO handled network comprising an MCBS that communicates through the backhaul link with a set $\mathcal{M} = \{1, 2, \dots, M\}$ of SCBSs. Let $\mathcal{M}^{'} = \{\mathcal{M} \cup 0\}$ be the augment set that includes the MCBS and the SCBSs, where the index 0 stands for the MCBS. Each SCBS is connected with its neighbouring SCBSs through the core network via high-speed links[1]. In the network, there is a set $\mathcal{U} = \{1, 2, \dots, U\}$ of users who request to receive files contained in the content catalogue $\mathcal{N} = \{1, 2, \dots, N\}$. Each file $n \in \mathcal{N}$ has size v_n bytes and is associated with a parameter λ_n^t that represents the number of requests for the nth file in time period t. The files are unsplittable, and hence they should be fully retrieved only from one base station (either MCBS or SCBS), however a file $n \in \mathcal{N}$ can be stored in multiple SCBSs. The SCBSs have limited storage capacity, and thus they can cache only a part of the content catalogue, while the MCBS can store the entire content catalogue. The storage capacity of the mth SCBS is denoted as b_m. The proposed online cache optimization problem described in the next sections can be solved centrally at the MCBS. This is a natural choice as this base station is aware of the network topology and is connected with all the SCBSs.

When a user $u \in \mathcal{U}$ issues a request for a content, e.g., $n \in \mathcal{N}$, it first directs this request to the closest SCBS, e.g., mth SCBS. If the content is cached in this SCBS, the delivery of the content incurs no extra cost. However, in case the content is not found at the mth SCBS, the content request is forwarded to neighboring SCBS specified by its forwarding table (step 1 in Fig. 1). If these do not possess the requested file also, they forward the content request to one of the neighboring SCBSs indicated by MCBS (step 2 in Fig. 1). This process is repeated until an SCBS is found that has the requested content. If no SCBS possesses the requested file, the content is retrieved from the MCBS (steps 1' and 2' in Fig. 1). When the requested content is found, the content is delivered back to the user who requested the content following the backward path of the one followed to reach the content (steps 3 and 4, or 3' and 4' in Fig. 1). This protocol can be realized using the content advertisement messages (Marandi et al. 2017), where SCBSs forwarding tables are updated periodically after receiving such messages. These messages are compressed by means of Bloom filters to limit the communication overhead. Pull-based content advertisement mechanisms (Marandi et al. 2019) can further reduce the overhead. Specifying the exact communication protocol is beyond the scope of this paper.

When the content requested by user u, who is in SCBS m's communication range, is retrieved from the cache of the SCBS (or MCBS) $m^{'} \in \mathcal{M}^{'}$, the delivery cost is equal to $c^u_{m,m^{'}}$; this accounts for the communication cost which includes the cost of using the network infrastructure. Without loss of generality, it is assumed that the cost depends on the shortest distance between the m and $m^{'} \in \mathcal{M}^{'}$, with $c^u_{m,m} = 0$. When

[1]The high-speed links can be backhaul links connecting the edge nodes of the core network, i.e., the SCBSs.

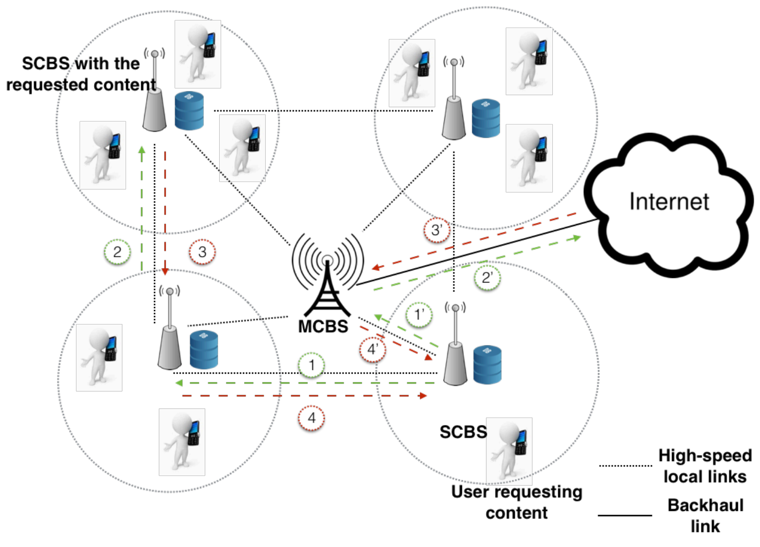

Figure 1. Considered network setting.

none of the SCBSs has the demanded content, it is retrieved from the MCBS through the backhaul link with a cost $c^u_{m,0}$, where $c^u_{m,0} > c^u_{m,m'}$. Although the content request model resembles the way information is requested in information-centric networks, the proposed caching algorithm is generic and is not tied to any specific communication protocol.

3.1. *Cache update mechanism*

The cache update flowchart is depicted in Fig. 2. When a content is requested in an SCBS, the following steps take place. The request is first processed by the *Content Request Processor*, which is responsible for forwarding the request to the *Cache Management* module and also for accumulating statistics regarding the requested content that are periodically send to the MCBS. The latter module checks whether the demanded content is stored in the SCBS's cache. If this is the case, the content is retrieved from SCBS's *Local Cache*. Differently, depending on SCBSs forwarding tables, the request is forwarded to a neighboring SCBSs or the MCBS where it is processed by their *Content Request Processor*. Upon retrieval of the content from other SCBSs or the MCBS, this is returned to the user. In our system, decisions regarding updating content to an SCBS cache are made in present decision time slots. In these slots, the statistics regarding encountered content requests at all SCBSs are forwarded to MCBSs *Content Popularity Forecasting* module. The overall statistics are then processed by the *Cache Decision* module, which runs the proposed *rolling horizon* algorithm to decide the cache update. These decisions are sent to all SCBSs' nodes *Cache Update* module that is responsible for removing content from SCBSs caches and adding new content to them. It also populates the forwarding tables of the SCBS.

4. Online Dynamic Programming model

In this section, the studied online collaborative cache optimization problem is formulated as a finite-horizon MDP to reflect the dynamics of the system. Cache decisions are made online and every T decision stages the cached content can be updated free-

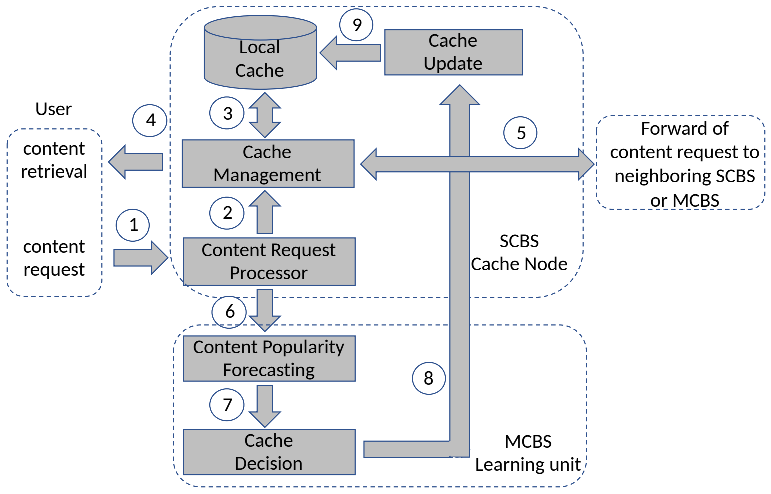

Figure 2. Modules of the considered cache update system.

of-charge, i.e., the cost for updating the cached content at the SCBSs is zero.

The states of the finite-horizon MDP are defined as $S_t = (\vec{x}_1^t, \vec{x}_2^t, ..., \vec{x}_M^t)$, where $\vec{x}_m^t = (x_{1m}^t, x_{2m}^t, ..., x_{Nm}^t)^T$ is a N dimensional $0-1$ vector that indicates whether a content is cached in an SCBS. x_{nm}^t is equal to 1 if content n is cached in SCBS m in stage t, otherwise is 0. Recall that M and N represent the number of SCBSs and the number of contents, respectively.

The actions of the finite-horizon MDP are composed of two sub-actions corresponding to the addition and/or the eviction of a content to/from the cache. The sub-action of adding a content to the cache is defined as $\vec{a}_t = (\vec{a}_1^t, \vec{a}_2^t, ..., \vec{a}_M^t)$, where $\vec{a}_m^t = (a_{1m}^t, a_{2m}^t, ..., a_{Nm}^t)^T$ is a N dimensional vector with a_{nm}^t be equal to 1 when content n is added in SCBS m in stage t, otherwise is 0. Similarly the sub-action of evicting a content from the cache is defined as $\vec{d}_t = (\vec{d}_1^t, \vec{d}_2^t, ..., \vec{d}_M^t)$, with $\vec{d}_j^t = (d_{1m}^t, d_{2m}^t, ..., d_{Nm}^t)^T$ being a N dimensional vector and d_{nm}^t is 1 if content n is evicted from SCBS m in stage t, differently is 0. Therefore, the state S_{t+1} and the x_{nm}^{t+1} evolve with the time as follows

$$S_t = S_{t-1} + \vec{a}_t - \vec{d}_t, \text{ and } x_{nm}^t = x_{nm}^{t-1} + a_{nm}^t - d_{nm}^t, \ n \in \mathcal{N}, m \in \mathcal{M}.$$

For each state S_t, the feasible action set (eligible actions) for stage t is given by

$$\chi_t = \{\vec{a}_t, \vec{d}_t | \quad \begin{array}{l} a_{nm}^t \leq 1 - x_{nm}^{t-1}, d_{nm}^t \leq x_{nm}^{t-1}, n \in \mathcal{N}, m \in \mathcal{M} \\ \sum_{n=1}^{N} v_n (x_{nm}^{t-1} + a_{nm}^t - d_{nm}^t) \leq b_m, m \in \mathcal{M}\} \end{array},$$

where the first inequality does not allow to add a content to an SCBS cache if it is already cached in it, while the second means that a content cannot be evicted from a cache if it is not there. The third inequality does not permit the total size of the cached content in an SCBS to exceed the cache capacity.

In each stage $t = 1, ..., T-1$, the decisions regarding which contents to add and/or evict to/from the cache of the mth SCBS are made by solving the Bellman's equation

$$V_{t-1}(S_{t-1}) = \min_{\vec{a}_t, \vec{d}_t \in \chi_t} \Big[I(\vec{a}_t, \vec{d}_t) + \min_{y^u_{n(m,m')}} \Big\{ \sum_n \sum_{u \in \mathcal{U}} \sum_{m' \in \mathcal{M}'} \lambda^t_n c^u_{m,m'} y^u_{n(m,m')} | \sum_{m'=0}^{M} y^u_{n(m,m')} = 1, y^u_{n(m,m')} \leq x^t_{nm'} \Big\} + V_t(S_t) \Big] \tag{1}$$

where $V_t(S_t)$ represents the future cost of the MDP model when the system is in state S_t, and λ^t_n represents the number of requests in time period t. The variable $y^u_{n(m,m')}$ is 1 if a request for content n issued by a client u at SCBS m is satisfied with the content copy at m', and $c^u_{m,m'}$ is the associated content delivery cost. The parameter $I(\vec{a}_t, \vec{d}_t)$ corresponds to the penalty coming as a result of actions $\vec{a}_t$ and $\vec{d}_t$:

$$I(\vec{a}_t, \vec{d}_t) = \gamma \sum_{n=1}^{N} \sum_{m=1}^{M} (a^t_{nm} + d^t_{nm}), \tag{2}$$

where γ stands for the cost (per content) incurred when adding/removing a content to/from a cache. In this paper, the penalty is linear to the number of contents to add/evict, but other models can be used. Since at the end of the horizon (e.g., at midnight) the contents update does not incur any penalty, two adjacent decision periods (e.g., days) can be considered independently by setting boundary condition $V_T(S_T) = 0$. Thus, during the first stage all the cached contents are updated free-of-charge, and the penalty term $I(\vec{a}_t, \vec{d}_t)$ in (1) is 0. Therefore, the decision problem becomes

$$V_0(S_0) = \min_{\vec{x}^1} \Big[\min_{y^u_{n(m,m')}} \Big\{ \sum_n \sum_{u \in \mathcal{U}} \sum_{m' \in \mathcal{M}'} \lambda^1_n c^u_{m,m'} y^u_{n(m,m')} | \sum_{m'=0}^{M} y^u_{n(m,m')} = 1, y^u_{n(m,m')} \leq x^1_{nm'} \Big\} + V_1(\vec{x}^1) \Big], \tag{3}$$

In the proposed scheme, differently from the standard offline approaches where it is assumed that content requests follow a long term distribution, e.g., Zipf, an auto-regressive model is employed to estimate the expected number of content requests $\lambda^t_n, n \in \mathcal{N}$. If ϵ is a noise parameter that captures the randomness in the evolution of λ^t_n, the number of content requests is estimated as

$$\tilde{\lambda}^t_n = \mu_t \cdot (\sum_{\tau=1}^{H} \beta_\tau \lambda^{t-\tau}_n + \epsilon), \tag{4}$$

where H is the number of previous stages considered during the prediction of content requests. The coefficients $\beta_\tau, \tau = 1, ..., H-1$ correspond to the weights of each of the H previous stages to the prediction process. Apparently, it is $\sum_{\tau=1}^{H} \beta_\tau = 1$. The parameter μ_t stands for the average number of per-content requests during stage t. This parameter varies with the stage index to reflect the time-varying demand pattern over the day. It is worth noting that in this paper, it is assumed that the auto-regressive model parameters, μ_t, β, and the distribution for ϵ are known. The determination of the optimal values of these parameters is an interesting problem, but it is out of the scope of this paper. Finally, it should be noted that in the employed auto-regressive model, the number of requests is not affected by the age of information, i.e., how long the content has been available to the content catalogue. Finally, it is assumed that content will not disappear before stage T. These are reasonable assumptions for a horizon of $T = 24$ hours.

5. Rolling horizon cache optimization algorithm

The online dynamic programming cache optimization model presented in the previous section suffers from the curse of dimensionality. Specifically, at every stage t an integer problem with $(M^2 + 2M)N$ variables should be solved. Given that the content cache status is a binary variable, the discrete state space has $\left(\sum_{l=0}^{b} C_N^l\right)^M$ components, where b is the cache capacity of every SCBS (assume they all have the same size) and C_N^l represents the number of combinations of l elements from N objects. Further, the decision problem solved in every stage is NP-hard as can be mapped to a multi-dimensional Knapsack problem (Martello and Toth 1990a). To overcome these difficulties, approximate solutions are proposed in this paper.

Backward induction algorithm needs the value function (i.e., cost) for stage t, i.e., $V_t(S_t)$, to optimize the cache decisions in stage $t-1$. If this cost is calculated for all possible states S_t, the decision problem in (1) will only contain information for the current stage $t-1$ and, thus, can be solved independently in each stage. Recall that $V_t(S_t)$ represents the future cost of following the optimal decisions starting from state S_t, which captures in its value both the information evolvement and the optimal decision policy. Considering how content popularity evolves, it is unlikely that contents are added to the cache according to the current optimal decision, and then be removed right in the next stage. This means that once a decision is made based on the current status, this decision is not expected to change for a short period of time until the system dynamics dominate the significance of the current solution. Therefore, a rolling horizon approximation over a short horizon Γ is constructed, where the optimal policy obtained at stage t is used for Γ subsequent periods. By doing this, the influence of the here-and-now decision to the far future is negligible, but the evolution of the content popularity in Γ stages is explicitly considered in the decision problem.

If the *rolling horizon* algorithm is implemented for Γ periods into the future, the decision problem at stage t becomes:

$$\begin{aligned}\tilde{V}_{t-1}(S_{t-1}) \approx \min_{\vec{a}_t,\vec{d}_t \in \mathcal{X}_t} \Big[I(\vec{a}_t,\vec{d}_t) + \min_{y^u_{n(m,m')}} \Big\{ \sum_n \sum_{u\in\mathcal{U}} \\ \sum_{m'\in\mathcal{M}'} (\lambda_n^t + \sum_{\tau=1}^{\Gamma} \tilde{\lambda}_n^{t+\tau}) c^u_{m,m'} y^u_{n(m,m')} | \sum_{m'=0}^{M} y^u_{n(m,m')} = 1, y^u_{n(m,m')} \le x^t_{nm'} \Big\} \Big]\end{aligned} \tag{5}$$

where $\tilde{\lambda}_n^{t+\tau}$ is the forecasted number of requests in stage $t+\tau$ and is computed by (4).

The above approximation removes the computational burden of estimating $V_t(S_t)$, but the difficulty of finding the optimal decisions $\vec{a}_t, \vec{d}_t$ in (5) remains unsolved. The two simplifications of the online cache optimization problem presented in the next subsection enable finding well-performing solutions with reduced complexity. Let P represents the inner decision problem of (5), i.e., finding the minimum communication cost (cost of satisfying all content demands during a period) under the current cache status. Without loss of generality, $(\lambda_n^t + \sum_{\tau=1}^{\Gamma} \tilde{\lambda}_n^{t+\tau})$ is replaced in the rolling horizon update formula (5) by a variable $\bar{\lambda}_n^t$ because this simplification can be used with other forecasting mechanisms.

$$P(\vec{x}^t) := \min \sum_n \sum_{u\in\mathcal{U}} \sum_{m'\in\mathcal{M}'} \bar{\lambda}_n^t c_{m,m'}^u y_{n(m,m')}^u \tag{6a}$$

$$s.t. \sum_{m'\in\mathcal{M}'} y_{n(m,m')}^u = 1, n\in\mathcal{N}, u\in\mathcal{U} \tag{6b}$$

$$y_{n(m,m')}^u \le x_{nm'}^t, n\in\mathcal{N}, u\in\mathcal{U}, m'\in\mathcal{M}^{'} \tag{6c}$$

$$y_{n(m,m')}^u \in \{0,1\}, n\in\mathcal{N}, u\in\mathcal{U}, m'\in\mathcal{M}^{'} \tag{6d}$$

5.1. *Allow a single copy of a content to be cached in SCBSs network*

Based on the observation that there are far more contents than the total capacity of all the SCBSs, which in practice holds, the decision problem is simplified further by restricting contents to be cached at one SCBS in the network. This is an efficient strategy when content popularity is given by a smooth distribution, i.e., the popularity of contents does not vary significantly. Under this simplification, when content $n\in\mathcal{N}$ is cached in SCBS $m\in\mathcal{M}$, all requests for it are served by the SCBS, which cached the content if it is less costly than receiving it from the backhaul. Therefore, the problem $P(\vec{x}^t)$ can be decomposed into sub-problems solved by contents. For content n, it is:

$$P_n(\vec{x}^t) := \min \sum_{u\in\mathcal{U}} \sum_{m'\in\mathcal{M}'} \bar{\lambda}_n^t c_{m,m'}^u y_{n(m,m')}^u \tag{7a}$$

$$s.t. \sum_{m'\in\mathcal{M}'} y_{n(m,m')}^u = 1, u\in\mathcal{U} \tag{7b}$$

$$y_{n(m,m')}^u \le x_{nm'}^t, u\in\mathcal{U}, m'\in\mathcal{M}^{'} \tag{7c}$$

$$y_{n(m,m')}^u \in \{0,1\}, u\in\mathcal{U}, m'\in\mathcal{M}^{'} \tag{7d}$$

If the content n is cached in SCBS $\hat{m}$, i.e., $x_{n\hat{m}}^t = 1$ and each content is cached in only one SCBS, then $x_{nm'}^t = 0, \forall m' \neq \hat{m}$. This means that $y_{n(m,m')}^u = 0, \forall m' \neq \hat{m}$. Hence, it should be $y_{n(m,\hat{m})}^u = 1 - \sum_{m'\neq\hat{m}} y_{n(m,m')}^u = 1$ in order to to satisfy (7b). In this case, the minimum usage cost is $\sum_{u\in\mathcal{U}} \lambda_n^t c_{m,\hat{m}}^u$ for each content n. However, if the content n is not cached at any SCBSs, the minimum usage cost is trivially equal to $\sum_{u\in\mathcal{U}} \lambda_n^t c_{m,0}^u$. Hence, under a cache placement $x_{nm'}^t$, the total communication cost is computed as:

$$\sum_n \lambda_n^t \sum_{u\in\mathcal{U}} \left[c_{m,0}^u (1 - \sum_{m'=1}^{M} x_{nm'}^t) + \sum_{m'=1}^{M} c_{m,m'}^u x_{nm'}^t \right] \tag{8}$$

This removes the need to solve the problem P, and (1) becomes

$$V_{t-1}(S_{t-1}) = \min_{\vec{a}_t,\vec{d}_t \in \chi_t} \Big\{ I(\vec{a}_t,\vec{d}_t) + \sum_n \lambda_n^t \sum_{u\in\mathcal{U}} \Big[c_{m,0}^u + \sum_{m'=1}^{M} (c_{m,m'}^u - c_{m,0}^u) x_{nm'}^t \Big] + V_t(S_t) \Big\}. \tag{9}$$

Note that the simplified problem in (9) does not contain content delivery decision variables y, and hence the size of the action space is greatly reduced from $(M^2+2M)N$ to $2MN$. Considering the simplification incurred by the rolling-horizon approximation described by (5), the cache and delivery optimization problem can be written as

$$\min I(\vec{a}_t,\vec{d}_t) + \sum_n (\lambda_n^t + \sum_{\tau=1}^{\Gamma} \tilde{\lambda}_n^{t+\tau}) \sum_{u\in\mathcal{U}} \left[c_{m,0}^u + \sum_{m'=1}^{M} (c_{m,m'}^u - c_{m,0}^u)(x_{nm'}^{t-1} + a_{nm'}^t - d_{nm'}^t) \right] \tag{10a}$$

$$s.t.\ a_{nm'}^t \le 1 - x_{nm'}^{t-1}, \forall n \in \mathcal{N}, \forall m' \in \mathcal{M} \tag{10b}$$

$$d_{nm'}^t \le x_{nm'}^{t-1}, \forall n \in \mathcal{N}, \forall m' \in \mathcal{M} \tag{10c}$$

$$\sum_{n=1}^{N} v_n (x_{nm'}^{t-1} + a_{nm'}^t - d_{nm'}^t) \le b_{m'}, \forall m' \in \mathcal{M} \tag{10d}$$

$$\sum_{m'\in\mathcal{M}} (x_{nm'}^{t-1} + a_{nm'}^t - d_{nm'}^t) \le 1, \forall n \in \mathcal{N} \tag{10e}$$

$$a_{nm'}^t, d_{nm'}^t \in \{0,1\}, \forall n \in \mathcal{N}, \forall m' \in \mathcal{M} \tag{10f}$$

It is worth to note that the simplified problem in (10a) has all the constraints of the original online minimization problem (1) and in addition has (10e) which prohibits the caching of a content in more than one SCBSs. This constraint enables the use of the closed-form formula (8) for computing the optimal usage cost, and thus decreases the cost of determining the optimal solution. As it will become clear in the evaluation section in the majority of the cases, the proposed simplification results in the same cache allocation with that of the original problem (1) unless the total cache capacity at SCBSs is higher than what is needed to cache all contents.

5.2. *Limit the number of allowed content replacements*

Despite the simplification considered in the previous subsection, the action space could still be large. Therefore, an alternative greedy heuristic (Algorithm 1) is proposed to solve (5), which examines contents with decreasing order of popularity. As shown in Algorithm 1, the greedy heuristic turns the optimization problem into a sequential decision-making problem by content. This is possible because demands for contents are independent of each other. P_n represents the cost of satisfying customer demands given the existing caching locations. $\Delta(x_{n'\leftarrow n,m}^t)$ corresponds to the difference in the cost when content n' is replaced by content n at the cache of the mth SCBS. Therefore, the overall optimization problem boils down to a content-by-content replacement problem where popular contents replace less popular ones in each SCBS. Please note that when P_n values are calculated (lines 5 and 7 in Algorithm 1), the popularity is forecasted as $\bar{\lambda}_n^t = (\lambda_n^t + \sum_{\tau=1}^{\Gamma} \tilde{\lambda}_n^{t+\tau})$.

Algorithm 1 Greedy heuristic for content replacement

```
1:  Input: maximum number of content replacements per cache, $r$
2:  Sort contents in an decreasing order of the forecasted number of downloads
3:  for $n^{'} = 1, \dots, r$ do
4:      for $m = 1, \dots, M$ do
5:          $\Delta(a^t_{n'm}) = \gamma + [P_{n'}(x^t_{n'm} = 1) - P_{n'}(x^t_{n'm} = 0)]$
6:          for each content $n$ that is cached on SCBS $m$ $(x^{t-1}_{nm} = 1)$ do
7:              $\Delta(d^t_{nm}) = \gamma + [P_n(x^t_{nm} = 0) - P_n(x^t_{nm} = 1)]$
8:              $\Delta(x^t_{n' \leftarrow n,m}) = \Delta(a^t_{n'm}) + \Delta(d^t_{nm})$
9:          end for
10:     end for
11:     if $(\min \Delta(x^t_{n' \leftarrow n,m}) < 0)$ then
12:         Carry out the corresponding replacement.
13:     end if
14: end for
```

Note that in Algorithm 1, a content replacement happens when the replacement of a content leads to a reduction of the total usage cost, regardless of whether contents involved in this replacement are cached anywhere else in the network. This means that multiple SCBSs may have a copy of the same content, which is different from the assumption used in Section 5.1. The cost reduction is calculated by considering a fixed-length horizon into the future, as what what have been used in (5). The proposed greedy heuristic is of low computational complexity, as it only requires to compare the parts of the value function which are relevant to the added and evicted contents in an SCBS. Therefore, the computational complexity of the algorithm is linear with respect to the number of allowed replacements r per decision stage. This type of heuristic is derived naturally by examining the structure of the optimization problem in (5) and has great potential of providing very good solutions given a large enough number of cache updates is allowed.

6. Experimental results

In this section, the proposed *rolling horizon* algorithm is evaluated for various settings. Before presenting the experimental results, several comparison schemes are introduced to benckmark the proposed policy.

6.1. *Comparison schemes*

6.1.1. Offline cache optimization with online updates based on LRU

The first comparison scheme decides which contents to cache at the SCBSs in two phases: offline and online. For both phases, the content updates at the SCBSs' caches are decided centrally considering the SCBSs' communication model described in Section 3. During the offline phase, cache update decisions are made assuming that content popularity follows a Zipf distribution with a known skewness parameter. Let p_n represent the popularity of content n, the optimization problem is formally expressed as

$$\min \sum_{n\in\mathcal{N}} p_n \Big(\sum_{u\in\mathcal{U}} \sum_{m'\in\mathcal{M}'} c^u_{m,m'} y^u_{n(m,m')} \Big) \tag{11a}$$

$$s.t. \sum_{n\in\mathcal{N}} v_n x_{nm'} \leq b_{m'}, m' \in \mathcal{M} \tag{11b}$$

$$\sum_{m'\in\mathcal{M}'} y^u_{n(m,m')} = 1, u \in \mathcal{U}, n \in \mathcal{N} \tag{11c}$$

$$y^u_{n(m,m')} \leq x_{nm'}, u \in \mathcal{U}, m' \in \mathcal{M}^{'}, n \in \mathcal{N} \tag{11d}$$

where the constraint (11b) corresponds to the capacity constraint, which prevents the total volume of contents cached in a SCBS from exceeding its capacity. The constraint (11c) ensures that all user requests are satisfied. Finally, the constraint (11d) imposes that a content can be obtained from an SCBS only if it is cached in it.

In the online phase, the cached content at the SCBSs is updated using the LRU policy. The online phase captures sudden content popularity changes and demands for contents that were not previously available in the content catalogue. In each decision stage, the least popular cached contents are replaced by the most popular ones, as observed in this stage. Multiple content replacements are allowed, provided that the subject to a cache decision content is more popular than the one to be replaced. Depending on whether caching multiple copies of the same content is allowed within the network, two variants of the LRU policy are considered: $LRU(S)_r$ and $LRU(M)_r$, where r stands for the maximum number of allowed replacements, while S and M indicate whether a single copy or multiple copies per content are allowed, respectively.

6.1.2. *Myopic policy*

In addition to the offline-online scheme, the proposed policy is also compared with alternative approximate solutions in the dynamic settings. The *Myopic* policy is a short-sighted approach which only considers the immediate cost when solving the Bellman's equation (1). This means that the future expected cost, $V_t(S_t)$, by carrying out the optimal policy in all future stages, is set to zero. Therefore, the cache updates are decided by solving:

$$\hat{V}_{t-1}(S_{t-1}) \approx \min_{\vec{a}_t, \vec{d}_t \in \chi_t} \Big[I(\vec{a}_t, \vec{d}_t) + \min_{y^u_{n(m,m')}} \Big\{ \sum_n \sum_{u\in\mathcal{U}} \tag{12a}$$

$$\sum_{m'\in\mathcal{M}'} \lambda^t_n c^u_{m,m'} y^u_{n(m,m')} | \sum_{m'=0}^{M} y^u_{n(m,m')} = 1, y^u_{n(m,m')} \leq x^t_{nm'} \Big\} \Big]. \tag{12b}$$

6.1.3. *One-step improvement policy*

Another policy used for evaluation purpose is the *One-step* improvement policy. This policy considers the long-run usage cost in addition to the immediate cost as an approximation to $V_t(S_t)$. For the computation of the long-run usage cost, it is assumed that both the content popularity and the caching plan will not be updated in the future. Under this assumption, the long-run usage cost is captured by a Zipfian distribution, which estimates the average popularity of every content over the entire decision horizon based on the encountered content requests. To capture the evolution of the number of requests per stage, the Zipfian distribution is rescaled. Hence, if the total number of content requests is constant over the future periods and a percentage p_n of them is for content n, the expected number of requests for content n in the future periods evolves as follows:

$$\tilde{\lambda}_n^{t+\tau} = p_n(\sum_{\hat{n}=1}^{N} \lambda_{\hat{n}}^t), \tau = 1, ..., T-t,$$

Then, the cache updates are decided by solving

$$\begin{aligned} \bar{V}_{t-1}(S_{t-1}) \approx \min_{\vec{a}_t, \vec{d}_t \in \chi_t} \Big[I(\vec{a}_t, \vec{d}_t) + \min_{y^u_{n(m,m')}} \Big\{ \sum_n \sum_{u \in \mathcal{U}} \\ \sum_{m' \in \mathcal{M}'} (\lambda_n^t + \sum_{\tau=1}^{T-t} \tilde{\lambda}_n^{t+\tau}) c^u_{m,m'} y^u_{n(m,m')} | \sum_{m'=0}^{M} y^u_{n(m,m')} = 1, y^u_{n(m,m')} \leq x^t_{nm'} \Big\} \Big] \end{aligned} \tag{13}$$

Note that for both *Myopic* and *One-step improvement* policies, the cache update decisions are made by solving (1) with an approximated $V_t(S_t)$, therefore, the simplifications discussed in Sections 5.1 and 5.2 are also applicable to them.

6.2. *Experimental results settings*

The performance of all schemes under comparison is evaluated for various settings summarized in Table 1. "Ratio" column corresponds to the ratio between the cumulative size of the contents in the content catalogue and the total cache capacity of the SCBSs. The third column in Table 1 depicts the SCBS network topology. For all cases, SCBSs form a grid network where a crossover point indicates the existence of an SCBS. In addition, all SCBSs are connected with a direct link to the MCBS. The communication cost to deliver a demanded content to a client is linearly dependent to the length of the shortest path between the request node and the supply node, while the downloading cost of a content from the MCBS $c^u_{m,0}$ is set to 20.

The parameter γ in Table 1 indicates the penalty cost of each content update as used in equation 2. This parameter is associated with the potential delay of meeting users' requests for adding and evicting contents. This cost should be larger than the cost of downloading a content directly from the MCBS, which is the minimum delay (cost) one has to wait for updating a content in an SCBS. Also, this cost should be smaller than the cost incurred when all content requests are served by the MCBS in one stage, as otherwise there would not be any reason to alter the current cache allocation. Thus, for all network settings and content update ratios, the parameter γ are set to 100, which is five times the cost of requesting a content from the MCBS. For all experiments, the β_τ parameters used in our forecasting model in (4) are $(0.6, 0.3, 0.1)$. Finally, hourly cache updates and $T = 24$ are considered.

6.3. *Numerical Results*

In this section, the performance of the proposed online cache optimization schemes (*rolling horizon* policies RH1-RH3, where the number corresponds to the size of the

Table 1. Evaluation scenarios settings

Index	#SCBS	Network topology	#Contents	Penalty(γ)	Capacity	Ratio	#New contents per hour
Ins 1.1					1	30%	
Ins 1.2	3	$\begin{bmatrix}\bullet & \bullet & \bullet\end{bmatrix}$	10	100	2	60%	1
Ins 1.3					3	90%	
Ins 1.4					4	120%	
Ins 2.1					10	30%	
Ins 2.2	3	$\begin{bmatrix}\bullet & \bullet & \bullet\end{bmatrix}$	100	100	20	60%	2
Ins 2.3					30	90%	
Ins 2.4					40	120%	
Ins 3.1					4	24%	
Ins 3.2					8	48%	
Ins 3.3	6	$\begin{bmatrix}\bullet & \bullet & \bullet\\ \bullet & \bullet & \bullet\end{bmatrix}$	100	100	12	72%	2
Ins 3.4					17	102%	
Ins 4.1					20	24%	
Ins 4.2					40	48%	
Ins 4.3	6	$\begin{bmatrix}\bullet & \bullet & \bullet\\ \bullet & \bullet & \bullet\end{bmatrix}$	500	100	60	72%	5
Ins 4.4					80	96%	
Ins 5.1					10	24%	
Ins 5.2					20	48%	
Ins 5.3	12	$\begin{bmatrix}\bullet & \bullet & \bullet & \bullet\\ \bullet & \bullet & \bullet & \bullet\\ \bullet & \bullet & \bullet & \bullet\end{bmatrix}$	500	100	30	72%	5
Ins 5.4					40	96%	
Ins 6.1					20	24%	
Ins 6.2					40	48%	
Ins 6.3	12	$\begin{bmatrix}\bullet & \bullet & \bullet & \bullet\\ \bullet & \bullet & \bullet & \bullet\\ \bullet & \bullet & \bullet & \bullet\end{bmatrix}$	1000	100	60	72%	10
Ins 6.4					80	96%	
Ins 7.1					17	25.5%	
Ins 7.2					33	49.5%	
Ins 7.3	15	$\begin{bmatrix}\bullet & \bullet & \bullet & \bullet & \bullet\\ \bullet & \bullet & \bullet & \bullet & \bullet\\ \bullet & \bullet & \bullet & \bullet & \bullet\end{bmatrix}$	1000	100	50	75%	10
Ins 7.4					66	99%	

horizon) are examined and compared with those presented in Section 6.1. All the reported results are averages of 100 simulations, i.e., realizations of different content requests.

6.3.1. Performance evaluation for exact solution of Bellman equation

To compare the approximation schemes without affected by the solution approach, the best action of (1) with an underlying policy is found by solving the optimization problem exactly without any simplification as proposed in Section 5.1 and 5.2. As optimal actions can be computed only for very small size problems due to the computational complexity, results are only shown for settings *Ins 1.k*, *Ins 2.k*, and *Ins 3.k*, $k \in \{1, 2, 3, 4\}$(see Table 1).

Table 2 shows the proportional performance (usage cost) of an evaluated policy compared to the lower bound LB and the offline policy, as described in Section 6.1.1. The proportional performance is calculated as $(* - LB)/(x_0 - LB)$, where x_0 indicates the cost of using the optimal offline cache decisions for the entire horizon without allowing any online updates. LB is equivalent to the cost of a scheme where accurate demands are known before updating online the cache and cache updates are done in all the stages before the end of the horizon T without any additional cost, i.e., the penalty cost γ in (2) is set to 0. Hence, LB can be seen as an unreachable lower bound (set to 0 by default) of any practical policy. For the $LRU(S)_r$ and $LRU(M)_r$ policies r is equal to the size of the content catalogue which means all contents can be potentially updated, while replacements only take place when there is an uncached content with higher popularity than the cached ones.

From the usage cost results presented in Table 2, it is obvious that apart from $LRU(M)_r$, all other online schemes achieve a usage cost lower than x_0 for all testing scenarios. This justifies the benefits of online updating. In most testing scenarios, the proposed *rolling horizon* method (RH1, RH2, RH3) outperforms other policies. The

Table 2. Usage cost for the online models assuming exact solution of (1)

.

		Offline(Zipf) + LRU		**Dynamic programming**					**Offline**
Index	LB	$LRU(S)_r$	$LRU(M)_r$	Myopic	One-step	RH1	RH2	RH3	x_0
Ins 1.1	0.00	0.6983	2.2488	0.2824	0.6817	**0.2559**	0.2727	0.3013	1.00
Ins 1.2	0.00	0.4989	2.2919	**0.1787**	0.4132	0.1812	0.2093	0.2224	1.00
Ins 1.3	0.00	0.2346	1.9823	0.1241	0.1745	**0.1084**	0.1179	0.1285	1.00
Ins 1.4	0.00	**0.1027**	2.4033	0.1338	0.1446	0.1052	0.1055	0.1067	1.00
Ins 2.1	0.00	0.9045	3.5173	0.4392	0.7614	**0.3450**	0.3667	0.4054	1.00
Ins 2.2	0.00	0.8026	4.0877	0.3454	0.4677	**0.2804**	0.2977	0.3292	1.00
Ins 2.3	0.00	0.4957	4.4549	0.1594	0.2945	**0.1564**	0.1774	0.1995	1.00
Ins 2.4	0.00	0.1041	5.3883	0.1350	0.2170	0.0997	0.0982	**0.0976**	1.00
Ins 3.1	0.00	0.9017	3.6483	0.4489	0.8109	**0.3403**	0.3590	0.3954	1.00
Ins 3.2	0.00	0.9117	4.4121	0.3848	0.6879	**0.2942**	0.3090	0.3437	1.00
Ins 3.3	0.00	0.7353	4.6935	0.2733	0.4700	**0.2326**	0.2496	0.2786	1.00
Ins 3.4	0.00	0.1547	5.2946	0.1367	0.2141	0.0974	0.0912	**0.0904**	1.00

Table 3. Cache Hit Ratio for online models assuming exact solution of (1)

		Offline(Zipf) + LRU		**Dynamic programming**					**Offline**
Index	LB	$LRU(S)_r$	$LRU(M)_r$	Myopic	One-step	RH1	RH2	RH3	x_0
Ins 1.1	0.0853	0.0883	**0.0970**	0.0794	0.0850	0.0829	0.0819	0.0861	0.0652
Ins 1.2	0.1506	0.1546	**0.1769**	0.1431	0.1460	0.1462	0.1497	0.1500	0.0944
Ins 1.3	0.1748	0.2112	**0.2540**	0.1872	0.1925	0.1959	0.1989	0.1970	0.0775
Ins 1.4	0.2792	0.2451	0.2828	0.2303	**0.2831**	0.2498	0.2626	0.2696	0.1346
Ins 2.1	0.0895	0.0980	**0.1147**	0.0870	0.0926	0.0922	0.0937	0.0947	0.0703
Ins 2.2	0.1519	0.1667	**0.2078**	0.1466	0.1533	0.1536	0.1567	0.1568	0.1284
Ins 2.3	0.1877	0.2039	**0.2893**	0.1938	0.1928	0.1971	0.1971	0.1991	0.1481
Ins 2.4	0.2819	0.2639	**0.3596**	0.2692	0.2963	0.2725	0.2791	0.2782	0.2230
Ins 3.1	0.0617	0.0617	**0.0750**	0.0561	0.0617	0.0595	0.0593	0.0610	0.0430
Ins 3.2	0.1071	0.1073	**0.1391**	0.0981	0.1074	0.1022	0.1027	0.1045	0.0837
Ins 3.3	0.1491	0.1438	**0.1980**	0.1306	0.1431	0.1315	0.1370	0.1374	0.1155
Ins 3.4	0.1679	0.1652	**0.2616**	0.1710	0.1758	0.1322	0.1737	0.1384	0.1275

best performance is achieved by RH1, which assumes a horizon of one. This means forecasting into the future does help in reducing costs (compared to the performance of *Myopic* policy, which only considers the immediate usage cost), while freezing the system status for too many stages will lose its significance in capturing system dynamics.

Further from Table 2, it can be observed that the performance gap between RH policies and *Myopic* and $LRU(S)_r$ policy becomes smaller as SCBSs' cache capacity increases. Indeed, when SBCSs' capacity is large, content popularity evolution becomes less important as there is sufficient space to cache most of or even all the contents. In such a case, close to optimal content update decisions can be made using only the immediate information. Differently, when capacity is very limited, the consideration of looking at several stages ahead leads to large usage cost reduction. Besides, in most cases the $LRU(S)_r$ policy without considering the explicit optimization model (1) performs worse than the *Myopic* policy, which solves the optimization model ignoring the future usage cost. This shows the importance of incorporating proper optimization in collaborative caching. It is also worth to note that although one would expect *One-step* improvement policy to perform better than the *Myopic* as it considers the immediate usage cost together with an estimation of the future cost, this does not happen because *One-step* improvement policy assumes a Zipfian model to capture the future usage cost and no future updates of the cached contents, which leads to an inaccurate approximation of the future cost $V_t(S_t)$.

Next, the achieved Cache Hit Ratio (CHR) by all the schemes under comparison

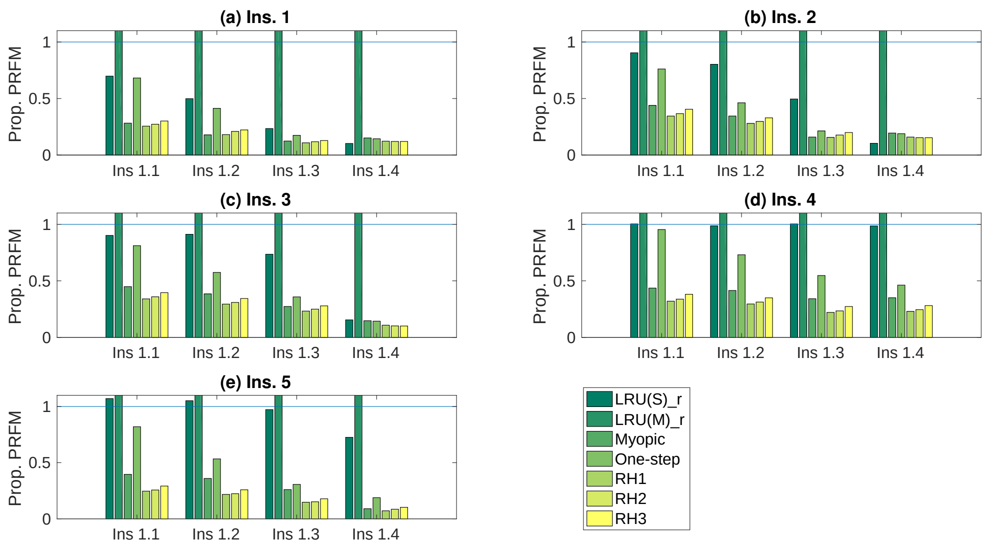

Figure 3. Numerical results for online model when approximate solution is obtained by solving (9).

is shown in Table 3. The results make clear that $LRU(M)_r$ achieves the highest or very close to the highest CHR although it performs worse than the other schemes in terms of cost. This is because, in a collaborative network serving more content requests from SCBSs close to the users requires multiple copies, which contradicts the aim of collaborative caching to maximize the "re-use" of the cached content (i.e., serve users with content cached in other SCBSs).

6.3.2. Performance evaluation of allowing a single copy of every content

In this subsection, the impact on the performance of the simplification in Section 5.1, allowing a single content copy in the SCBSs network, is investigated. Considering this simplification, the usage cost is minimized by solving (9) instead of (1). This approximation model removes cache delivery y variables from the decision problem, and, thus, largely reduces the problem size and the solution time.

The evaluation results are summarized in Fig. 3. Y-axis shows the proportional performance of the corresponding methodology, calculated by $(* - LB)/(x_0 - LB)$. By comparing these results with the ones in Table 2, it is observed that the results are nearly the same with that of solving the decision problem optimally given that the benchmark policies, say x_0 and LB, do not change with the reformulation. However, the solution time is largely reduced from 5 hours to 7 minutes, for performing 100 simulations of *Ins 3* settings. This method needs in average 0.18 seconds to find the optimal cache update policy for *Ins 3* in every stage.

Interestingly, the performance of the *One-step* policy improves when contents can be cached in only one SCBS in the network. This is explained by the fact that the longer the prediction period it considers, the larger are the differences in the total demands of contents, which results in the popularity of some contents to be overestimated. Differences in the usage cost reduction can also be noted when cache capacity is large. This happens as in this case optimally the content should have been cached in multiple SCBSs, and because of the restriction of one-copy per content restriction, an error is introduced in the reformulation. To overcome this problem, the proposed heuristic replacement approach is examined in the next section, which facilitates a fast solution

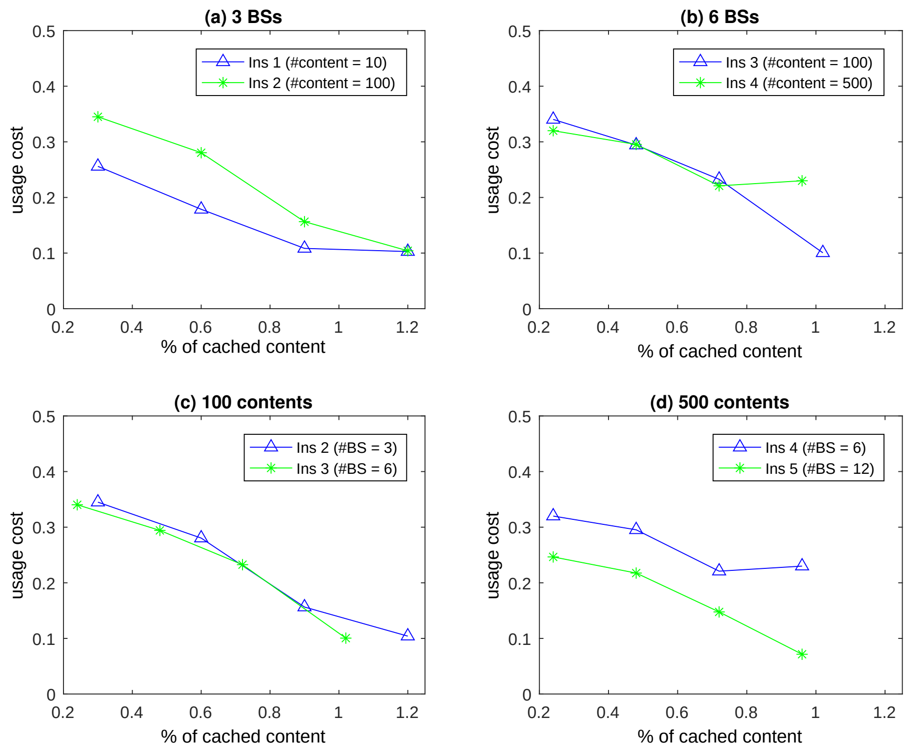

Figure 4. Comparison of usage cost over different parameter settings for online models, when approximate solution is obtained by solving (9).

for the online decision problem without imposing the one-copy per-content condition.

Two network topologies with three and six SCBSs are considered for investigating the impact of the content catalogue size on the usage cost. The results are shown in Figs. 4(a) and (b), where the proportional usage cost is that of the best performing *rolling horizon* policy. From this evaluation, it can be concluded that given the same cached ratio, in most cases smaller catalogue size leads to better usage costs. Different content catalogue sizes are considered in Figs. 4(c) and (d) for the same network topologies. The results make clear that the higher the number of SCBSs is, the better the performance of the RH policy is. This can be explained as follows: the larger the network topology is, the higher delivery cost it incurs, and the higher penalty one receives by sub-optimal content cache placements and updates. Therefore, larger profits are expected by updating the cached content in the SCBSs by considering both the immediate cost and a future prediction of it.

6.3.3. Performance evaluation of limiting the number of content replacements

In this section, the performance of the proposed heuristic solution in Section 5.2 is investigated. This scheme solves the cache optimization problem by setting the maximum number of content replacements equal to the total cache capacity of the network. Therefore, potentially all the contents that are currently cached in any SCBS (including copies in different SCBSs) can be updated at every stage. This updating scheme is used indifferently on all dynamic programming policies, however the forecasting strategy employed by each policy determines whether a replacement leads to a lower usage cost. Note that this approach allows multiple copies of contents at different SCBSs.

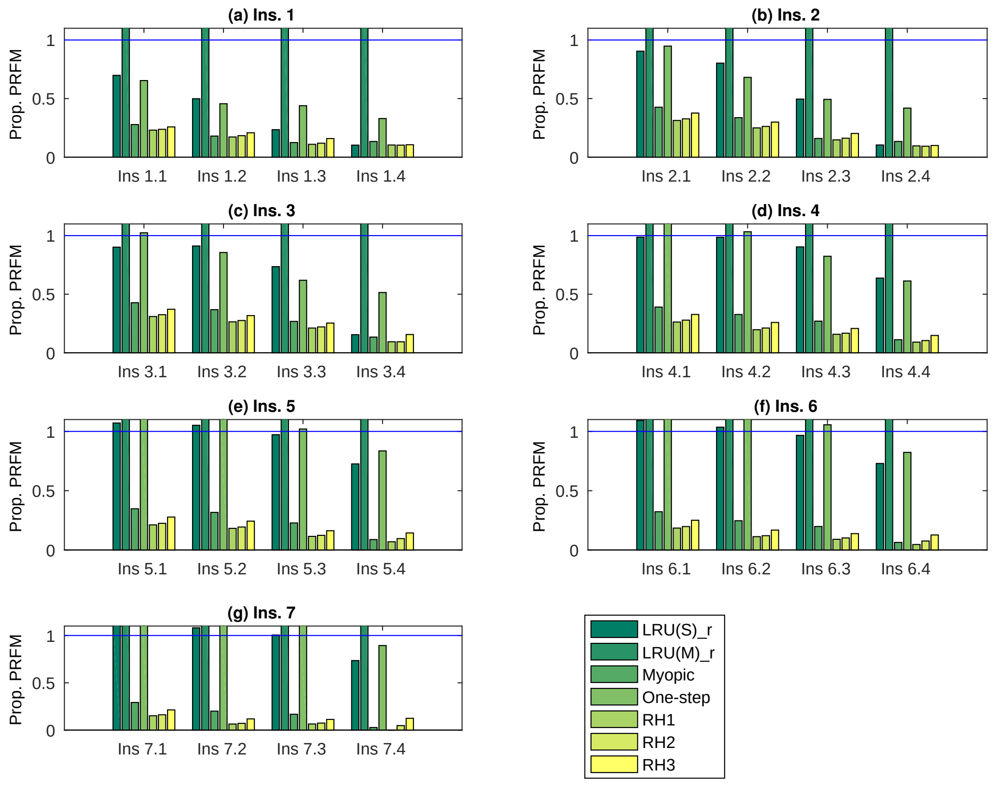

Figure 5. Numerical results for online model when replacement heuristic is used on all DP policies.

The results of this evaluation are summarized in Fig. 5. By comparing these results with those in Table 2, where Bellman's equation is solved exactly, it can be seen that the performance of the dynamic programming policies changes slightly. The usage cost for $LRU(M)_r$ and $LRU(S)_r$ policies remain the same in both comparisons, as the replacement heuristic is only applied to dynamic programming policies. It is observed that in some cases this heuristic gives even slightly smaller overall cost.

Further, by comparing the results of the dynamic programming policies with that shown in Fig. 3 it is noted that the replacement heuristic improves the performance significantly for all $*.4$ scenarios, which is attributed to the relaxation of the one-copy per-content condition. Also, from Fig. 5 is observed that the proposed schemes RH1-RH2 achieve the smallest usage cost. More importantly, the performance of the RH policies improves with the size of the studied problem (bigger network and number of contents). For example, for *Ins.7*, which corresponds to a topology consisting of 15 SCBSs and 1000 contents, the RH1 cost is only 0.04% higher than the theoretical lower bound, which cannot be achieved in practice as the replacement penalty is zero (assume no repay when replacing a cached content by another). This improved performance on larger networks justifies the reliability of RH policies in practice. Also, the exact number of cache updates (replacements) one should allow does not have to be optimized. The number of maximum replacements always can be set to a large number, as in practice, a replacement decision is only made when it reduces the immediate cost together with a short prediction in the future that is captured in (5). Nevertheless, the computation time is largely reduced with the heuristic. For larger instances, e.g., *Ins.7*, for every stage, the cache update decision can be made in approximately 1 sec.

7. Discussion and conclusions

In this paper, a *rolling horizon* collaborative cache optimization scheme is presented for wireless networks. The gains of the proposed scheme exceed 69-99% the performance of the offline caching schemes. In order to reduce the complexity of the *rolling horizon* scheme, future content updates/replacements costs are approximated by considering only a limited history horizon. To further reduce the time needed to solve the online cache optimization problem, two simplifications of the problem are proposed: (a) to limit the number of content replicas in the network and (b) to limit the allowed content replacements. The experimental results show that policies achieving the higher cache hit ratio do not necessarily coincide with the ones achieving the minimal usage cost. Also, the results make clear the value of considering future information when deciding the cache update policy. When cache capacity is limited, restricting the number of content replicas is shown to be very efficient. Limiting the allowed content updates, in general, leads to easy-to-deploy schemes, especially when the *rolling horizon* approximation is used. Nevertheless, the size of the horizon depends on the computational capacity and the affordable inaccuracy of forecasting.

References

Abad, M. S. H., E. Ozfatura, O. Ercetin, and D. Gunduz (2019, Jan.). Dynamic Content Updates in Heterogeneous Wireless Networks. In *Annual Conf. on Wireless On-demand Network Systems and Services, WONS'19*, Wengen, Switzerland.

Bharath, B. N., K. G. Nagananda, D. Gunduz, and H. V. Poor (2018, Sep.). Caching With Time-Varying Popularity Profiles: A Learning-Theoretic Perspective. *IEEE Trans. on Communications 66*(8), 3837–3847.

Chattopadhyay, A., B. Blaszczyszyn, and H. P. Keeler (2018a, Feb.). Gibbsian On-Line Distributed Content Caching Strategy for Cellular Networks. *IEEE Trans. on Wireless Communications 17*(2), 969–981.

Chattopadhyay, A., B. Blaszczyszyn, and H. P. Keeler (2018b, Feb.). Optimal and Scalable Caching for 5G Using Reinforcement Learning of Space-Time Popularities. *IEEE Journal on Selected Topics on Signal Processing 12*(1), 180–190.

Gharaibeh, A., A. Khreishah, B. Ji, and M. Ayyash (2016, Aug.). A Provably Efficient Online Collaborative Caching Algorithm for Multicell-Coordinated Systems. *IEEE Trans. on Mobile Computing 15*(8), 1863–1876.

Gharaibeh, A., A. Khreishah, and I. Khalil (2016, Apr.). An O(1)-Competitive Online Caching Algorithm for Content Centric Networking. In *Proc. IEEE INFOCOM*, San Francisco, CA, USA.

Golrezaei, N., A. F. Molisch, A. G. Dimakis, and G. Caire (2013, Apr.). Femtocaching and Device-to-Device Collaboration: A New Architecture for Wireless Video Distribution. *IEEE Communications Magazine 51*(4), 142–149.

Khreishah, A. and J. Chakareski (2015, Apr.). Collaborative Caching for Multicell-Coordinated Systems. In *Proc. IEEE INFOCOM Workshop Communication and Networking Techniques for Contemporary Video*, Hong Kong.

Khreishah, A., J. Chakareski, and A. Gharabeih (2016, Aug.). Joint Caching, Routing, and Channel Assignment for Collaborative Small-Cell Cellular Networks. *IEEE Journal on Selected Areas in Communications 34*(8), 2275–2284.

Li, S., J. Xu, M. van der Schaar, and W. Li (2016, Dec.). Trend-Aware Video Caching Through Online Learning. *IEEE Trans. Multimedia 18*(12), 2503–2516.

Maddah-Ali, M. A. and U. Niesen (2014, May). Fundamental Limits of Caching. *IEEE Trans. Information Theory 60*(5), 2856–2867.

Maggi, L., L. Gkatzikis, G. Paschos, and J. Leguay (2018, Jan.). Adapting Caching to Audience Retention Rate. *Computer Communications 116*, 159–171.

Maniotis, P., E. Bourtsoulatze, and N. Thomos (forthcoming 2020). Tile-based joint caching and delivery of 360^o videos in heterogeneous networks. *IEEE Trans. on Multimedia*.

Marandi, A., T. Braun, K. Salamatian, and N. Thomos (2017, Jun.). BFR: a Bloom Filter-based Routing Approach for Information-Centric Networks. In *IFIP Networking Conference, Networking'17*, Stockholm, Sweden.

Marandi, A., T. Braun, K. Salamatian, and N. Thomos (2019, Jan.). Pull-based Bloom Filter-based Routing for Information-Centric Networking. In *IEEE Consumer Communications & Networking Conference, CCNC'19*, Las Vegas, NV, USA.

Martello, S. and P. Toth (1990a). *Knapsack Problems.* New York: Wiley.

Martello, S. and P. Toth (1990b). *Knapsack Problems: Algorithms and Computer Implementations.* New York: Wiley.

Muller, S., O. Atan, M. van der Schaar, and A. Klein (2017, Feb.). Context-Aware Proactive Content Caching With Service Differentiation in Wireless Networks. *Trans. on Wireless Communications 16*(2), 1024–1036.

Neglia, G., D. Carra, and P. Michiardi (2018, Feb.). Cache Policies for Linear Utility Maximization. *IEEE/ACM Trans. on Networking 26*(1), 302–313.

Poularakis, K., G. Iosifidis, A. Argyriou, I. Koutsopoulos, and L. Tassiulas (2016, Apr.). Caching and Operator Cooperation Policies for Layered Video Content Delivery. In *Proc. IEEE INFOCOM 2016*, San Francisco, CA, USA.

Poularakis, K., G. Iosifidis, and L. Tassiulas (2014, Oct.). Approximation Algorithms for Mobile Data Caching in Small Cell Networks. *IEEE Trans. Communications 62*(10), 3665–3677.

Saltarin, J., T. Braun, E. Bourtsoulatze, and N. Thomos (2018, May). PopNetCod: A Popularity-based Caching Policy for Network Coding enabled Named Data Networking. In *Proc. of IFIP Networking Conference, Networking18*, Zurich, Switzerland.

Thomos, N., E. Kurdoglu, P. Frossard, and M. V. der Schaar (2015, Jun.). Adaptive Prioritized Random Linear Coding and Scheduling for Layered Data Delivery from Multiple Servers. *IEEE Trans. on Multimedia 17*(6), 893–906.

Yang, P., N. Zhang, S. Zhang, L. Yu, J. Zhang, and X. Shen (2019, Apr.). Content Popularity Prediction Towards Location-Aware Mobile Edge Caching. *IEEE Trans. on Multimedia 21*(4), 915–929.

Zhang, N., K. Zheng, and M. Tao (2018, May). Using Grouped Linear Prediction and Accelerated Reinforcement Learning for Online Content Caching. In *Proc. of IEEE Int. Conf. on Communications Workshops, ICCW18*, Kansas City, MO, USA.