Received 27 March 2021; accepted 14 April 2021. Date of publication 22 April 2021; date of current version 30 April 2021.

Digital Object Identifier 10.1109/OJCOMS.2021.3075071

Trend-Aware Proactive Caching via Tensor Train Decomposition: A Bayesian Viewpoint

SAJAD MEHRIZI, THANG X. VU (Member, IEEE), SYMEON CHATZINOTAS (Senior Member, IEEE), AND BJÖRN OTTERSTEN (Fellow, IEEE)

Interdisciplinary Center for Security, Reliability and Trust, University of Luxembourg, 1855 Luxembourg, Luxembourg

CORRESPONDING AUTHOR: S. MEHRIZI (e-mail: mehrizis@gmail.com)

This work was supported in part by National Research Fund (FNR), Luxembourg, under the Projects "LISTEN" and "PROCAST" and in part by European Research Council (ERC) under the Project "AGNOSTIC." This paper was accepted in part at IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), 2020 [1].

ABSTRACT Content caching at base stations is an effective solution to cope with the unprecedented data traffic growth by prefetching contents near to end-users. To proactively servicing users, it is of high importance to extract predictive information from data requests. In this paper, we propose an accurate content request prediction algorithm for improving the performance of edge caching systems. In particular, we develop a Bayesian dynamical model through which a complex nonlinear latent temporal trend structure in the content requests can be accurately tracked and predicted. The dynamical model also leverages tensor train decomposition to capture content-location interactions to further enhance the accuracy of predictions. To infer the model's parameters, we derive an approximation of the posterior distribution based on variational Bayes (VB) method with an embedded Kalman smoother algorithm. Based on the predictions of the proposed model, we design a cost-efficient proactive cooperative caching policy which adaptively utilizes network resources and optimizes the content delivery. The advantage of the proposed caching scheme is demonstrated via numerical results using two real-world datasets, which show that the developed Bayesian dynamical model substantially outperforms reference methods that ignore the temporal trends and content-location interactions.

INDEX TERMS Proactive caching, content request prediction, tensor train decomposition, Bayesian modeling, temporal trend.

I. INTRODUCTION

MOBILE traffic is explosively increasing, under the constant pressure of data hungry applications such as on-demand video streaming [2]. The proactive caching, as an effective approach for mitigating this issue, aims at exploiting predictable user demand behavior in smoothing out communication network traffic by prefetching the most popular contents close to end-users during off-peak hours [3]. Leveraging these predictive abilities, network resources can be pre-allocated more efficiently by servicing predictable peak-hour requests.

To design an accurate content prediction algorithm, it is crucial to understand the underlying structures in the requests. In particular, content requests may exhibit three important patterns:

1) Non-linear temporal trends: In reality, there exist different kinds of temporal request trends among different types of contents. For example, it has been observed that a video can go through multiple phases of increase and decrease in requests during its life-cycle [4].
2) The interactions among contents: The requests for some contents may exhibit a similar pattern, for instance due to sharing the same category of features.
3) The local content popularities and their interactions across locations (i.e., spatial correlations): It has been observed that the requests for the majority of contents are location-dependent [5]. For example, users in residential, academic, and other type of areas are interested in specific categories of contents. Yet, the requests in various locations are not independent and they may

exhibit similar patterns. For example, content popularity can spread through word-of-mouth across social connections.

Most of prior work on content request prediction focused mainly on static scenario and ignored the time evolution pattern in the requests, e.g., [6]–[8]. Content requests, however, exhibit time-dependency which can have considerable effect on the caching performance. The auto-regressive moving-average (ARMA) in time-series is the most widely-used model for prediction [9]–[11]. The ARMA model specifically is designed for continuous-valued data and it may not be suitable for count-valued content requests. In other words, due to the Gaussian assumption, it is likely to predict negative values for the requests which are not meaningful. In [12], a prediction algorithm was designed where the requests are restricted to be on the probability simplex space. Though the algorithm therein guarantees the positiveness of the requests in prediction, it has an important shortcoming. Particularly, in many caching policies, e.g., as in [13], it is critical to predict the actual request rate (e.g., to measure data traffic) over time and location in a network. Therefore, the normalized requests may not be usable for such polices. Additionally, both the ARMA model and the proposed model in [12] are linear and also ignore the latent structures in the content requests, as mentioned in points 1-3, and hence, they can be quite restrictive.

Deep neural network (DNN) modeling approach has been used in [14]–[16] for the prediction task. DNN models are non-linear and potentially can capture complex structures in the requests. An importing drawback of most DNN models (and also ARMA model) is that they require rich historical data for prediction. However, content providers continuously release new contents for which there is rare or no historical data. Additionally, different from conventional content delivery networks, the number of requests is very small at the network edge [17] which may limit their applications in edge caching systems due to overfitting problem.

The authors of [18]–[22] proposed to apply reinforcement learning (RL) to learn the underlying dynamics of the request for cache optimization. Nevertheless, as it was shown in [23], RL may not be yet competitive in comparison to simple heuristics, e.g., random caching. The reason is that RL typically requires millions of learning samples which leads to slow reaction times in dynamic environments [24]. In caching systems this issue is significant since servers can face quickly changing unexpected data traffic patterns.

Bayesian approaches are widely popular for tackling data scarcity problem and being robust against overfitting [25]. In our previous work [26], we proposed a Bayesian model based on matrix factorization approach to capture the interactions among the contents in a time varying scenario. More recently in [27], we introduced a dynamical model using a CANDECOM/PARAFAC (CP) tensor decomposition to capture content-location interactions.

Parallelly, there has been a great interest in exploring optimal proactive cooperative caching policy and performance analysis for an efficient use of network resources. The vast majority of work, however, focused on static scenarios or designed policies with fixed resource dimensioning [13], [27], [28], and [29]. A more efficient and realistic scheme is to design a policy that adaptively adjusts network resources according to the dynamic of content popularity. This can be achieved thanks to the recent technological developments on cloud service providers where physical resources are mapped to virtual resources [30]. As a result, the network operator can dimension the consumed resources as required.

In this paper, we consider edge caching systems with dynamic resource dimensioning and introduce a flexible dynamical model to capture the nonlinear trend information of the requests. Our contributions are as follows:

- We model the observed requests by a tensor with three modes, i.e., content, location and time. By deploying tensor train factorization approach, we decompose the requests into three latent factors (content, location, time) through which complex interactions can be captured. The train factorization method numerically is more stable and provides a more accurate representation of the data than the classical CP factorization method [31], [32]. Subsequently, by assuming that the time factors follow a structured Gaussian process, complex temporal trend structures in the requests can be modeled.
- Due to insufficient request observations in the network edge [17], we adhere to the Bayesian principle, which is very efficient for dealing with the problem of overfitting, for model learning. Moreover, a simple-to-implement approximation method based on the Kalman smoother and VB is derived to infer the parameters of Bayesian dynamical model. The Bayesian trend-aware train decomposition model developed in this work has a more expressiveness power than the non-trend CP decomposition model in [27]. Therefore, the VB algorithm designed here is more sophisticated than the one in [27].
- To provide an end-to-end caching system design, a joint routing and content placement policy is designed for cooperative caching to minimize network cost by adaptively utilizing the network resources, i.e., cache memory and link bandwidth, in time-varying scenario. The caching policy is coupled over time and is a function of future requests therefore it is infeasible to solve for online setting. To tackle the issue, we predict a sequence of future requests by leveraging the dynamical model to optimize a multistage caching policy defined over a prediction horizon.
- The caching policy is a mixed-integer non-linear programming problem which is NP-hard and difficult to solve. To overcome the challenge of the formulated policy, we develop an approximation algorithm based on difference of convex (DC) programming which can be solved in polynomial time.

- We show via numerical results that the developed Bayesian dynamical model is substantially outperforms reference methods which ignore the temporal trends and content-location interactions using two real-world datasets.

The rest of the paper is organized as follows. Section II describes the Bayesian dynamical model for proactive caching. In Section III, we explain the VB approximation algorithm. In Section IV, we describe an adaptive cooperative caching policy. Section V provides numerical results. Finally, VI concludes the paper.

Notation: Scalars are denoted by lower-case letters, e.g., a, vectors are denoted by bold-face lowercase letters, e.g., $\mathbf{a}$, and matrices are denoted by bold-face uppercase letters, e.g., $\mathbf{A}$. The superscript $(.)^T$ denotes matrix transpose. The expectation is denoted by $E[.]$. $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, and the symbol $\sim$ is used to mean "distributed as". $\mathcal{G}(\alpha, \beta)$ denotes gamma distribution with shape α and rate β, and Pois(r) denotes Poisson distribution with rate r.

II. BAYESIAN TREND MODEL FOR PROACTIVE CACHING

Consider a cellular network composed of N base stations (BSs) which are distributed over a geographical area and cooperatively serve theirs users. Each BS $n \in \mathcal{B} \triangleq \{1, \ldots, N\}$ has a cache with limited storage capacity and can store a subset of contents from library $\mathcal{M} \triangleq \{1, \ldots, M\}$ with M contents. Over time, the users in a cell submit requests from library $\mathcal{M}$ towards the BS's cache. All BSs are connected to the content server through back-haul links via mobile network operator core and we assume all the processing tasks are performed in this core. The mobile network operator aims to provide the users a proactive caching service for the peak hours. To this end, it collects requests from the BSs, makes prediction and subsequently caches the contents based on caching policy. Hence, to obtain an efficient caching policy, an algorithm for tracking and predicting the requests is essential. In this section, we introduce a Bayesian trend model by which tracking and prediction can be performed accurately.

Lets consider a three-mode tensor $\mathbf{D} \in \mathbb{Z}^{M \times N \times T}$ consisting of request observations during T time slots, where element $\mathbf{D}(m, n, t)$ denoting the total number of requests for content m by the users at cell n at time slot t. We assume that the t-th frontal slice of $\mathbf{D}$, denoted more compactly by $\mathbf{D}_t$, can be decomposed under Poisson likelihood as:

$$\begin{aligned} \mathbf{D}_t(m, n) &\sim \text{Pois}(r_{mnt}), \\ r_{mnt} &= \sum_{k_2=1}^{K_2} \sum_{k_1=1}^{K_1} \text{A}(k_1, m)\mathbf{Z}_t(k_1, k_2)\mathbf{B}(k_2, n), \end{aligned} \quad (1)$$

where vectors $\mathbf{A}(:, m) \in \mathbb{R}_+^{K_1 \times 1}$ and $\mathbf{B}(:, n) \in \mathbb{R}_+^{K_2 \times 1}$ contain the latent-factor representations of content m and cell n respectively; and $\mathbf{Z}_t \in \mathbb{R}_+^{K_1 \times K_2}$ models the interactions of the latent factors at time t. Moreover, $k_1 = 1, \ldots, K_1$ is the column index of matrices $\mathbf{A}$ and $\mathbf{Z}_t$; $k_2 = 1, \ldots, K_2$ is the row index of matrices $\mathbf{B}$ and $\mathbf{Z}_t$; and r_{mnt} is defined as the mean of the requests $\mathbf{D}_t(m, n)$, i.e., the mean of the requests for content m at cell n during time slot t. Generally, it is assumed $K_1 < M$ and $K_2 < N$ such that the latent factors offer a more compact description of the requests. The decomposition approach deployed in (1) is called tensor train factorization [31] and [32]. This is a modification of probabilistic CP tensor factorization in [33] but with three differences. First, the Poisson substitutes the Gaussian which makes the model to be suitable for the requests that are non-negative integers. Second, the latent factors are positive which improves the interpretability of the model. Third, the time factor, $\mathbf{Z}_t$, is a full matrix which enhances the expressiveness power of the model. We also note that the CP tensor factorization method proposed in [27] for Poisson data is a special case of the formulation in (1), where the time factor matrix is diagonal. A graphical view of decomposition model (1) is shown in Fig. 1.

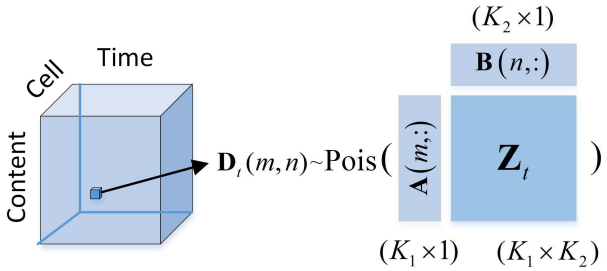

FIGURE 1. The tensor train factorization for content requests.

Using the property of the Poisson model, (1) can be equivalently reformulated as [34]:

$$\hat{\mathbf{D}}_{mnt}(k_1, k_2) \sim \text{Pois}(\mathbf{A}(k_1, m)\mathbf{Z}_t(k_1, k_2)\mathbf{B}(k_2, n)), \quad (2)$$

where $\hat{\mathbf{D}}_{mnt}$ is an auxiliary latent variable such that $\mathbf{D}_t(m, n) = \sum_{k_1=1}^{K_1} \sum_{k_2=1}^{K_2} \hat{\mathbf{D}}_{mnt}(k_1, k_2)$. The reformulation in (2) facilitates the inference, as will be shown in the following.

Regarding the time factor $\mathbf{Z}_t$, which reveals the time evolution in the requests, it is rational to assume that it varies smoothly over time. Specifically, we consider the following hierarchical Gaussian process model:

$$\begin{aligned} \mathbf{x}_{0t} &= \boldsymbol{\Pi}_0 \mathbf{x}_{0t-1} + \mathbf{x}_{1t} + \mathbf{e}_{0t}, \\ \mathbf{x}_{1t} &= \boldsymbol{\Pi}_1 \mathbf{x}_{1t-1} + \mathbf{x}_{2t-1} + \mathbf{e}_{1t}, \\ &\vdots \\ \mathbf{x}_{Lt} &= \boldsymbol{\Pi}_L \mathbf{x}_{Lt-1} + \mathbf{e}_{Lt}, \end{aligned} \quad (3)$$

and the time evolution in $\mathbf{Z}_t$ is modeled as $\mathbf{Z}_t = \exp(\mathbf{X}_{0t})$, where $\mathbf{x}_{\ell,t} = \text{vec}(\mathbf{X}_{\ell,t})$ and vec(.) is the vectorization operator. Moreover, $\mathbf{e}_{\ell,t}$ is a random variable represents unpredictable change in latent variable $\mathbf{x}_{\ell,\mathbf{t}}$ between time t and time $t-1$, and it is assumed that $\mathbf{e}_{\ell,t} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}_\ell^{-1})$. Additionally, matrix $\boldsymbol{\Pi}_\ell \in \mathbb{R}^{K_1 K_2 \times K_1 K_2}$ captures the dependency between $\mathbf{x}_{\ell,t}$ and $\mathbf{x}_{\ell,t-1}$.

The Gaussian model in (3) consists of $L+1$ latent layers and constructs a non-linear trend model [35]. Specifically, the zeroth latent layer, modeled by $\mathbf{x}_{0t}$, captures temporal locality in $\mathbf{Z}_t$ and equivalently in the content requests. The first latent layer, modeled by $\mathbf{x}_{1t}$, captures a linear trend in the evolution of $\mathbf{x}_{0t}$. The second layer, modeled by $\mathbf{x}_{2t}$, captures a linear trend in $\mathbf{x}_{1t}$ which results a quadratic trend in $\mathbf{x}_{0t}$. Similarly, the Lth layer, modeled by $\mathbf{x}_{Lt}$, captures a linear trend in $\mathbf{x}_{L-1,t}$ which results a quadratic trend in $\mathbf{x}_{L-2,t}$, a cubic trend in $\mathbf{x}_{L-3,t}$ and so on. Therefore, by increasing the number of latent layers a non-linear trend model can be constructed. Due to this particular presumed underlying structure on the way latent factors $\mathbf{Z}_t$ are generated, the model can uncover a complex time evolution pattern in the requests.

Now, we explain the form of priors we choose for the parameters of dynamical model. The motivation in the following priors is to satisfy conjugacy property for efficient inference [36]. Similar to [34], we use the following hierarchical gamma priors for the elements of $\mathbf{B}$ as:

$$\mathbf{B}(k_2, n) \sim \mathcal{G}\Big(\beta, \hat{\mathbf{b}}(n)\Big), \tag{4}$$

$$\hat{\mathbf{b}}(n) \sim \mathcal{G}\Big(\hat{\beta}, \hat{\hat{\beta}}\Big). \tag{5}$$

The gamma priors in (4) promote the location factors to have sparse representations. In particular, the gamma distribution puts non-zero mass on zero value and has an exponential shape for $\beta \leq 1$; therefore it is sparsity promoting. The sparsity level increases as β decreases. Additionally, defining hyper-priors on location-specific gamma rate parameters captures the overall activity level of users across locations. This hierarchical structure captures the heterogeneity across locations, some can have a larger number of active users than others. Similarly, we use the following hierarchical priors for the elements of $\mathbf{A}$ as:

$$\mathbf{A}(k_1, m) \sim \mathcal{G}\big(\alpha, \hat{\mathbf{a}}(m)\big), \tag{6}$$

$$\hat{\mathbf{a}}(m) \sim \mathcal{G}\Big(\hat{\alpha}, \hat{\hat{\alpha}}\Big). \tag{7}$$

Likewise, the hierarchical gamma priors in (6) and (7) promote the content latent factors to have sparse representations and also capture the heterogeneity across contents, some can be more popular than others.

The priors for $\mathbf{\Lambda}_\ell$ and $\mathbf{\Pi}_\ell$ are jointly modeled by a Gaussian-gamma density as:

$$p(\boldsymbol{\lambda}_\ell(k), \boldsymbol{\pi}_\ell(k)) = \mathcal{N}\Big(\boldsymbol{\pi}_\ell(k); \eta_\ell, \frac{1}{\kappa_\ell \boldsymbol{\lambda}_\ell(k)}\Big)\mathcal{G}(\boldsymbol{\lambda}_\ell(k)|\tau_\ell, \rho_\ell), \quad \forall k = 1, \ldots, K_1K_2. \tag{8}$$

where, for simplicity, we assume that they are diagonal such that $\boldsymbol{\lambda}_\ell = \text{diag}(\mathbf{\Lambda}_\ell)$ and $\boldsymbol{\pi}_\ell = \text{diag}(\mathbf{\Pi}_\ell)$. We additionally assume that the prior of initial states in (3) to be Gaussian $\mathbf{x}_{\ell 0} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{\ell,init}^{-1})$. A graphical representation of the Bayesian model is shown in Fig. 2.

For ease of notation, we collect all unknowns into a single parameter denoted by $\mathbf{h} = \{\mathbf{A}, \hat{\mathbf{a}}, \mathbf{B}, \hat{\mathbf{b}}, \{\{\{\hat{\mathbf{D}}_{mnt}\}_{m=1}^{M}\}_{n=1}^{N}\}_{t=1}^{T}, \{\{\mathbf{x}_{\ell t}\}_{t=0}^{T}, \mathbf{\Lambda}_\ell, \mathbf{\Pi}_\ell\}_{\ell=0}^{L}\}$. Our goal is to develop a method to compute the full posterior of variables in $\mathbf{h}$ given the observed content requests over T time slots, stated as:

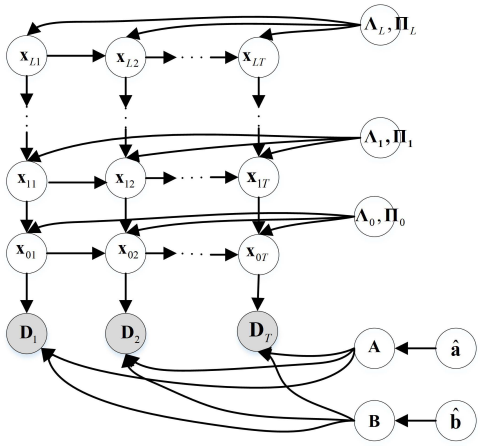

FIGURE 2. A graphical representation of the Bayesian model.

$$p(\mathbf{h}|\mathbf{D}_{1:T}) = p(\mathbf{h}, \mathbf{D}_{1:T}) / \int p(\mathbf{h}, \mathbf{D}_{1:T}) d\mathbf{h}. \tag{9}$$

III. MODEL LEARNING VIA VARIATIONAL BAYES

An exact Bayesian inference in (9) needs to integrate over all the variables encapsulated in $\mathbf{h}$, which is analytically intractable. To find a scalable Bayesian inference, we approximate $p(\mathbf{h}|\mathbf{D}_{1:T})$ with factorized density $q(\mathbf{h}) = \prod_j q(\mathbf{h}_j)$, found by maximizing a variational lower bound to log marginal likelihood, $\log p(\mathbf{D}_{1:T})$, [36]. By using the coordinate decent method, it is shown that the optimized j-th variational factor has the following update [36]:

$$\log q_j(\mathbf{h}_j) \propto E_{\sim q(\mathbf{h}_j)}\big[\log p(\mathbf{h}, \mathbf{D}_{1:T})\big]. \tag{10}$$

The notation $E_{\sim q(\mathbf{h}_j)}[.]$ denotes an expectation with respect to all variables except $\mathbf{h}_j$. Since any term that does not depend on $\mathbf{h}_j$ can be constant under the expectation and can be subsumed into the constant term, we can rewrite (10) as:

$$\log q_j(\mathbf{h}_j) \propto E_{\sim q(\mathbf{h}_j)}\big[\log p(\mathbf{h}_j|)\big], \tag{11}$$

where $p(\mathbf{h}_j|.)$ is the un-normalized (or normalized) conditional posterior of $\mathbf{h}_j$.

We employ the following factorization for $q(\mathbf{h})$:

$$q(\mathbf{h}) = q(\mathbf{A})q(\hat{\mathbf{a}})q(\mathbf{B})q\Big(\hat{\mathbf{b}}\Big)\Big\{\prod_{\ell=1}^{L} q\big(\mathbf{x}_{\ell,0:T}\big)\Big\}\Big\{\prod_{t=0}^{T} q(\mathbf{x}_{0t})\Big\} \Big\{\prod_{t=1}^{T}\prod_{m=1}^{M}\prod_{n=1}^{N} q\Big(\hat{\mathbf{D}}_{mnt}\Big)\Big\}\Big\{\prod_{\ell=0}^{L} q(\mathbf{\Lambda}_\ell, \mathbf{\Pi}_\ell)\Big\}. \tag{12}$$

We now present the forms of the optimized variational factors in the following. We drop subscript $\sim q(\mathbf{h}_j)$ in (10) and (11) for notational brevity, and all the expectations need to be computed with respect to the variables inside the expectation argument under the variational density unless we explicitly mention it.

TABLE 1. A summary of variational updates.

Variational density	Parameters
$q(\mathbf{A}(k_1,m)) = \mathcal{G}(\mathbf{A}(k_1,m);\boldsymbol{\alpha}_{a_m}(k_1),\boldsymbol{\beta}_{a_m}(k_1))$	$\boldsymbol{\alpha}_{a_m}(k_1) = \alpha + \sum_{t,n,k_2} E[\hat{\mathbf{D}}_{mnt}(k_1,k_2)]$ $\boldsymbol{\beta}_{a_m}(k_1) = E[\hat{\mathbf{a}}(m)] + \sum_{t,n,k_2} E[\exp(\mathbf{x}_{0t}(k'_{k_1k_2}))\mathbf{B}(k_2,n)]$
$q(\hat{\mathbf{a}}(m)) = \mathcal{G}(\hat{\mathbf{a}}(m);\alpha_{\hat{a}_m},\beta_{\hat{a}_m})$	$\alpha_{\hat{a}_m} = \alpha + \alpha K_1$ $\beta_{\hat{a}_m} = \alpha + \sum_{k_1} E[\mathbf{A}(k_1,m)]$
$q(\mathbf{B}(k_2,n)) = \mathcal{G}(\mathbf{B}(k_2,n);\boldsymbol{\alpha}_{b_n}(k_2),\boldsymbol{\beta}_{b_n}(k_2))$	$\boldsymbol{\alpha}_{b_n}(k_2) = \beta + \sum_{t,m,k_1} E[\hat{\mathbf{D}}_{mnt}(k_1,k_2)]$ $\boldsymbol{\beta}_{b_n}(k_2) = E[\hat{\mathbf{b}}(m)] + \sum_{t,m,k_1} E[\exp(\mathbf{x}_{0t}(k'_{k_1k_2}))\mathbf{A}(k_1,m)]$
$q(\hat{\mathbf{b}}(n)) = \mathcal{G}(\hat{\mathbf{b}}(n);\alpha_{\hat{b}_n},\beta_{\hat{b}_n})$	$\alpha_{\hat{b}_n} = \beta + \beta K_2$, $\beta_{\hat{b}_n} = \beta + \sum_{k_2} E[\mathbf{B}(k_2,n)]$

1) MULTINOMIAL UPDATE FOR $q(\hat{\mathbf{D}}_{mnt})$

By using the variable augmentation method in (2) and the property of Poisson and multinomial [37], the conditional posterior of $\hat{\mathbf{d}}_{mnt} = \text{vec}(\hat{\mathbf{D}}_{mnt})$ is a multinomial density given by:

$$p\Big(\hat{\mathbf{d}}_{mnt}|\Big) = \text{Multi}\Big(\hat{\mathbf{d}}_{mnt};\mathbf{D}_t(m,n),\mathbf{p}_{mnt}\Big), \tag{13}$$

where

$$\mathbf{p}_{mnt} = \frac{\tilde{\mathbf{p}}_{mnt}}{\sum_k \tilde{\mathbf{p}}_{mnt}(k)},$$
$$\tilde{\mathbf{p}}_{mnt}\big(k'_{k_1k_2}\big) = \mathbf{A}(k_1,m)\exp(\mathbf{X}_{0t}(k_1,k_2))\mathbf{B}(k_2,n),$$

with $k'_{k_1k_2} = (k_2-1)K_1 + k_1$, $\forall k_1 = 1,\ldots,K_1$, $\forall k_2 = 1,\ldots,K_2$. By substituting (13) into (11), we obtain:

$$q\Big(\hat{\mathbf{d}}_{mnt}\Big) \propto \frac{1}{\prod_k \hat{\mathbf{d}}_{mnt}(k)} \prod_k \Big(\breve{\mathbf{p}}_{mnt}(k)\Big)^{\hat{\mathbf{d}}_{mnt}(k)}, \tag{14}$$

where

$$\breve{\mathbf{p}}_{mnt}\big(k'_{k_1k_2}\big) = \exp\Bigg(E\Bigg[\log(\mathbf{A}(k_1,m)\exp(\mathbf{X}_{0t}(k_1,k_2))\mathbf{B}(k_2,n)) - \log\Bigg(\sum_{k_1,k_2}\mathbf{A}(k_1,m)\exp(\mathbf{X}_{0t}(k_1,k_2))\mathbf{B}(k_2,n)\Bigg)\Bigg]\Bigg). \tag{15}$$

We can see that by normalizing $\breve{\mathbf{p}}_{mnt}$, expression (14) is in a form of multinomial density. Therefore, we assume $q(\hat{\mathbf{d}}_{mnt})$ is a multinomial density given by:

$$q\Big(\hat{\mathbf{d}}_{mnt}\Big) = \text{Multi}\Big(\hat{\mathbf{d}}_{mnt};\mathbf{D}_t(m,n),\breve{\mathbf{p}}_{mnt}\Big), \tag{16}$$

with

$$\breve{\mathbf{p}}_{mnt} = \frac{\breve{\tilde{\mathbf{p}}}_{mnt}}{\sum_k \breve{\tilde{\mathbf{p}}}_{mnt}(k)}. \tag{17}$$

Note that the second term inside the expectation in (15), which cannot be computed analytically, is automatically canceled out due to the normalization in (17).

2) GAMMA UPDATES FOR $q(\mathbf{A})$

The gamma and Poisson distributions in (6) and (2) are conjugate. It is straightforward to show that the conditional posterior of $\mathbf{A}(k_1,m)$ is a gamma density given as:

$$p(\mathbf{A}(k_1,m)|) = \mathcal{G}\Bigg(\mathbf{A}(k_1,m);\alpha + \sum_{t,n,k_2}\hat{\mathbf{D}}_{mnt}(k_1,k_2), \hat{\mathbf{a}}(m) + \sum_{t,n,k_2}\exp\big(\mathbf{x}_{0t}\big(k'_{k_1k_2}\big)\big)\mathbf{B}(k_2,n)\Bigg). \tag{18}$$

By substituting (18) into (11), the vartaional density $q(\mathbf{A}(k_1,m))$ is again a gamma density. Table 1 provides the exact expression.

Gamma updates for $q(\hat{\mathbf{a}}), q(\mathbf{B}), q(\hat{\mathbf{b}})$: The updates for these densities can be derived in similar way as for $q(\mathbf{A})$ and are summarized in Table 1.

Gaussian-gamma updates for $q(\boldsymbol{\Lambda}_\ell, \boldsymbol{\Pi}_\ell)$: It can be shown that the optimized variational distribution (41) has the following form (the proof is given in the Appendix):

$$q(\boldsymbol{\pi}_\ell(k)|\boldsymbol{\lambda}_\ell(k)) = \mathcal{N}\Big(\boldsymbol{\pi}_\ell(k);\boldsymbol{\eta}_\ell(k),\frac{1}{\boldsymbol{\lambda}_\ell(k)\boldsymbol{\kappa}_\ell(k)}\Big), \tag{19}$$
$$q(\boldsymbol{\lambda}_\ell(k)) = \mathcal{G}\big(\boldsymbol{\lambda}_\ell(k);\boldsymbol{\tau}_\ell(k),\boldsymbol{\rho}_\ell(k)\big), \tag{20}$$

where:

$$\boldsymbol{\kappa}_\ell(k) = \Big(E\Big[\hat{\hat{\mathbf{x}}}_{k\ell}^T\hat{\hat{\mathbf{x}}}_{k\ell}\Big] + \kappa_\ell\Big),$$
$$\boldsymbol{\eta}_\ell(k) = \frac{\Big(E\Big[\hat{\hat{\mathbf{x}}}_{k\ell}^T\big(\hat{\mathbf{x}}_{k\ell} - \hat{\mathbf{x}}_{k\ell+1}\big)\Big] + \kappa_\ell\eta_\ell\Big)}{\boldsymbol{\kappa}_\ell(k)},$$
$$\boldsymbol{\tau}_\ell(k) = \tau_\ell + T/2,$$
$$\boldsymbol{\rho}_\ell(k) = \rho_\ell + \frac{1}{2}\Bigg[E\Big[\big(\hat{\mathbf{x}}_{k\ell} - \hat{\mathbf{x}}_{k\ell+1}\big)^T\big(\hat{\mathbf{x}}_{k\ell} - \hat{\mathbf{x}}_{k\ell+1}\big)\Big] + \kappa_\ell\eta_\ell^2 - \frac{\Big(E\Big[\hat{\hat{\mathbf{x}}}_{k\ell}^T\big(\hat{\mathbf{x}}_{k\ell} - \hat{\mathbf{x}}_{k\ell+1}\big)\Big] + \kappa_\ell\eta_\ell\Big)^2}{\boldsymbol{\kappa}_\ell(k)}\Bigg],$$
$$\hat{\mathbf{x}}_{k\ell} = [\mathbf{x}_{\ell1}(k),\ldots,\mathbf{x}_{\ell T}(k)]^T,$$
$$\hat{\hat{\mathbf{x}}}_{k\ell} = \big[\mathbf{x}_{\ell0}(k),\ldots,\mathbf{x}_{\ell,T-1}(k)\big]^T.$$

Gaussian Updates for $q(\mathbf{x}_{0(0:T)})$: The functional dependency of (11) on $\mathbf{x}_{0t}$ has the following form[1]:

$$\log q(\mathbf{x}_{0t}) \propto \mathbf{c}_{1t}^T\mathbf{x}_{0t} - \mathbf{c}_{2t}^T \exp(\mathbf{x}_{0t}) - \frac{1}{2}\mathbf{x}_{0t}^T\big(\bar{\mathbf{\Lambda}}_0 + \mathbf{\Upsilon}_0\big)\mathbf{x}_{0t} + \mathbf{x}_{0t}^T E\Big[\bar{\mathbf{\Lambda}}_0\big(\mathbf{x}_{1(t)} + \bar{\mathbf{\Pi}}_0\mathbf{x}_{0(t-1)} + \bar{\mathbf{\Pi}}_0\big(\mathbf{x}_{0t+1} - \mathbf{x}_{1(t+1)}\big)\big)\Big], \tag{21}$$

where $\mathbf{c}_{1t} = \sum_{m,n} E[\hat{\mathbf{d}}_{mnt}]$, $\bar{\mathbf{\Lambda}}_\ell = E[\mathbf{\Lambda}_\ell]$, $\bar{\mathbf{\Pi}}_\ell = E[\mathbf{\Pi}_\ell]$, $\mathbf{c}_{2t}(k'_{k_1k_2}) = \sum_{m,n} E[\mathbf{A}(k_1,m)\mathbf{B}(k_2,n)]$, and $\mathbf{\Upsilon}_\ell = E[\mathbf{\Pi}_\ell\mathbf{\Lambda}_\ell\mathbf{\Pi}_\ell]$. The Gaussian process $\mathbf{x}_{0t}$ is used as a prior for the natural (or equivalently rate) parameter of the Poisson distribution in which they are not conjugate. Because there does not exist close-form expression for (21), similar to [26], we approximate it by a Gaussian density using the Laplace method as:

$$q(\mathbf{x}_{0t}) \approx \mathcal{N}\Big(\mathbf{x}_{0t}; \mathbf{m}_{0t}, \mathbf{Q}_{0t}^{-1}\Big), \tag{22}$$

where $\mathbf{m}_{0t} = \mathbf{x}_{0t}^*$ and $\mathbf{x}_{0t}^*$ is the point of maxima of (21). Moreover, $\mathbf{Q}_{0t} = -[\nabla^2 f(\mathbf{x}_{0t}{}^*)]$ where $\nabla^2 f(\mathbf{x}_{0t}{}^*)$ is the Hessian matrix of (21) at $\mathbf{x}_{0t}^*$.

Gaussian updates for $q(\mathbf{x}_{\ell 0:T})$ for $\ell > 0$: Unlike in (21), there is no source of non-conjugacy for $\mathbf{x}_{\ell 0:T}$ and since they construct a singly connected network with Gaussian density in each node, the update follows a Gaussian density. The required quantities during the inference and the prediction are marginal $q(\mathbf{x}_{\ell t})$ and pairwise marginal $q(\mathbf{x}_{\ell t}, \mathbf{x}_{\ell t+1})$ which are again Gaussian densities. To this end, we deploy a Kalman smoother algorithm. The modification is that, except $\mathbf{x}_{\ell t}$, all variables are replaced with their expectations under the vartaional distributions. Please refer to [38] for more details.

The marginal density $q(\mathbf{x}_{\ell t})$ can be computed in two recursions: forward and backward. By combing the two recursions, we obtain:

$$q(\mathbf{x}_{\ell t}) = \mathcal{N}\Big(\mathbf{x}_{\ell t}; \mathbf{m}_{\ell t}, \mathbf{Q}_{\ell t}^{-1}\Big) \propto \vec{q}(\mathbf{x}_{\ell t})\overleftarrow{q}(\mathbf{x}_{\ell t}), \tag{23}$$

where $\vec{q}(\mathbf{x}_{\ell t})$ and $\overleftarrow{q}(\mathbf{x}_{\ell t})$ are the forward and backward density messages, respectively.

The forward message is computed as:

$$\vec{q}(\mathbf{x}_{\ell t}) \propto \hat{q}(\mathbf{x}_{\ell t}) \exp E\big[\log p\big(\mathbf{x}_{\ell-1,t}|\mathbf{x}_{\ell,t}, \mathbf{x}_{\ell-1,t-1}\big)\big], \tag{24}$$

where

$$\hat{q}(\mathbf{x}_{\ell t}) \propto \int d\mathbf{x}_{\ell,t-1}\vec{q}\big(\mathbf{x}_{\ell,t-1}\big) \exp E\big[\log p\big(\mathbf{x}_{\ell t}|\mathbf{x}_{\ell,t-1}, \mathbf{x}_{\ell+1,t}\big)\big]. \tag{25}$$

Assuming $\vec{q}(\mathbf{x}_{\ell,t-1}) = \mathcal{N}(\vec{\mathbf{m}}_{\ell,t-1}, \vec{\mathbf{Q}}_{\ell,t-1}^{-1})$, the integrand in (25) is a Gaussian quadratic function in $\mathbf{x}_{\ell,t-1}$. Thus, (25) has a closed-form solution which is a Gaussian density (after normalization) given as:

$$\hat{q}(\mathbf{x}_{\ell t}) = \mathcal{N}\Big(\mathbf{x}_{\ell t}; \hat{\mathbf{m}}_{\ell t}, \hat{\mathbf{Q}}_{\ell t}^{-1}\Big), \tag{26}$$

1. Note that we will not pay special attention to the boundaries of $\mathbf{x}_{\ell t}$ at $t = 0$ and $t = T$ and details can be worked out easily.

where

$$\begin{aligned}
\hat{\mathbf{Q}}_{\ell t} &= \bar{\mathbf{\Lambda}}_\ell - \bar{\mathbf{\Pi}}_\ell\bar{\mathbf{\Lambda}}_\ell\hat{\hat{\mathbf{Q}}}_{\ell t}\bar{\mathbf{\Lambda}}_\ell\bar{\mathbf{\Pi}}_\ell,\\
\hat{\hat{\mathbf{Q}}}_{\ell t} &= \Big[\vec{\mathbf{Q}}_{\ell,t-1} + \mathbf{\Upsilon}_{\boldsymbol{\ell}}\Big]^{-1},\\
\hat{\mathbf{m}}_{\ell t} &= \hat{\mathbf{Q}}_{\ell t}^{-1}\Big[\bar{\mathbf{\Pi}}_\ell\bar{\mathbf{\Lambda}}_\ell\hat{\hat{\mathbf{Q}}}_{\ell t}\vec{\mathbf{Q}}_{\ell,t-1}\vec{\mathbf{m}}_{\ell,t-1} + \bar{\mathbf{\Lambda}}_\ell\mathbf{m}_{\ell+1,t} - \bar{\mathbf{\Pi}}_\ell\bar{\mathbf{\Lambda}}_\ell\hat{\hat{\mathbf{Q}}}_{\ell t}\bar{\mathbf{\Lambda}}_\ell\bar{\mathbf{\Pi}}_\ell\mathbf{m}_{\ell+1,t}\Big].
\end{aligned}$$

From (26) and (24), we obtain

$$\vec{q}(\mathbf{x}_{\ell t}) = \mathcal{N}\Big(\mathbf{x}_{\ell t}; \vec{\mathbf{m}}_{\ell t}, \vec{\mathbf{Q}}_{\ell t}^{-1}\Big), \tag{27}$$

where

$$\begin{aligned}
\vec{\mathbf{Q}}_{\ell t} &= \bar{\Lambda}_{\ell-1} + \hat{\mathbf{Q}}_{1t}\\
\vec{\mathbf{m}}_{\ell t} &= \vec{\mathbf{Q}}_{\ell t}^{-1}\Big[\hat{\mathbf{Q}}_{\ell t}\hat{\mathbf{m}}_{\ell t} + \mathbf{N}_{\ell-1}\big(\mathbf{m}_{\ell-1,t} - \bar{\mathbf{\Pi}}_{\ell-1}\mathbf{m}_{\ell-1,t-1}\big)\Big].
\end{aligned}$$

The backward density message is computed by:

$$\overleftarrow{q}(\mathbf{x}_{\ell t}) \propto \int \overleftarrow{q}\big(\mathbf{x}_{\ell,t+1}\big) \exp \times E\big[\log p\big(\mathbf{x}_{\ell t+1}|\mathbf{x}_{\ell t}, \mathbf{x}_{\ell+1,t+1}\big) p\big(\mathbf{x}_{\ell-1,t+1}|\mathbf{x}_{\ell,t+1}, \mathbf{x}_{\ell-1,t}\big)\big]d\mathbf{x}_{\ell,t+1}. \tag{28}$$

Assuming $\overleftarrow{q}(\mathbf{x}_{\ell,t+1}) = \mathcal{N}(\mathbf{x}_{\ell,t+1}; \overleftarrow{\mathbf{m}}_{\ell,t+1}, \overleftarrow{\mathbf{Q}}_{\ell,t+1}^{-1})$, the integrand is a Gaussian quadric function in $\mathbf{x}_{\ell,t+1}$. Thus, expression (28) has a closed-form solution and is given by:

$$\overleftarrow{q}(\mathbf{x}_{ht}) = \mathcal{N}\Big(\mathbf{x}_{\ell t}; \overleftarrow{\mathbf{m}}_{\ell t}, \overleftarrow{\mathbf{Q}}_{\ell t}^{-1}\Big), \tag{29}$$

where

$$\begin{aligned}
\overleftarrow{\mathbf{m}}_{\ell t} &= \overleftarrow{\mathbf{Q}}_{\ell t}^{-1}\Big(- \bar{\mathbf{\Pi}}_\ell\bar{\mathbf{\Lambda}}_\ell\mathbf{m}_{\ell+1,t+1} + \mathbf{Q}_{\ell t}^*\bar{\mathbf{\Pi}}_\ell\bar{\mathbf{\Lambda}}_\ell\mathbf{m}_{\ell+1,t+1} + \mathbf{Q}_{\ell t}^*\Big(\bar{\mathbf{\Lambda}}_{\ell-1}\big(\mathbf{m}_{\ell-1,t+1} - \bar{\mathbf{\Pi}}_{\ell-1}\mathbf{m}_{\ell-1,t}\big) + \mathbf{Q}_{\ell t}^*\overleftarrow{\mathbf{Q}}_{\ell,t+1}\mathbf{m}_{\ell,t+1}\Big)\Big),\\
\mathbf{Q}_{\ell t}^* &= \bar{\mathbf{\Pi}}_\ell\bar{\mathbf{\Lambda}}_\ell\Big(\bar{\mathbf{\Lambda}}_{\ell-1} + \bar{\mathbf{\Lambda}}_\ell + \overleftarrow{\mathbf{Q}}_{\ell,t+1}\Big)^{-1},\\
\overleftarrow{\mathbf{Q}}_{\ell t} &= \mathbf{\Upsilon}_\ell - \mathbf{Q}_{\ell t}^*\bar{\mathbf{\Lambda}}_\ell\bar{\mathbf{\Pi}}_\ell.
\end{aligned}$$

To run the backward recursion, an initial condition is required for $\mathbf{x}_{\ell T}$. This is usually set to be $q(\mathbf{x}_{\ell T}) = 1$ which can be obtained by a zero mean Gaussian density with high variance.

Now, according to (23), by combining the forward and backward density messages, we obtain:

$$q(\mathbf{x}_{\ell t}) = \mathcal{N}\Big(\mathbf{x}_{\ell t}; \mathbf{m}_{\ell t}, \mathbf{Q}_{\ell t}^{-1}\Big), \tag{30}$$

where

$$\begin{aligned}
\mathbf{Q}_{\ell t} &= \vec{\mathbf{Q}}_{\ell t} + \overleftarrow{\mathbf{Q}}_{\ell t},\\
\mathbf{m}_{\ell t} &= \mathbf{Q}_{\ell t}^{-1}\Big(\vec{\mathbf{Q}}_{\ell t}\vec{\mathbf{m}}_{\ell t} + \overleftarrow{\mathbf{Q}}_{\ell t}\overleftarrow{\mathbf{m}}_{\ell t}\Big).
\end{aligned}$$

Similarly, the pairwise marginal can be computed as:

$$q\big(\mathbf{x}_{\ell t}, \mathbf{x}_{\ell,t+1}\big) \propto \vec{q}(\mathbf{x}_{\ell t})\overleftarrow{q}\big(\mathbf{x}_{\ell,t+1}\big) \times \exp E_{\sim q(\mathbf{x}_{\ell t}, \mathbf{x}_{\ell,t+1})}$$

$$\times \Big[\log p\big(\mathbf{x}_{\ell,t+1}|\mathbf{x}_{\ell t}, \mathbf{x}_{\ell+1,t+1}\big) \times p\big(\mathbf{x}_{\ell-1,t+1}|\mathbf{x}_{\ell,t+1}, \mathbf{x}_{\ell-1,t}\big)\Big]. \quad (31)$$

By using the Schur complement, we can show that the covariance between $\mathbf{x}_{\ell,t}$ and $\mathbf{x}_{\ell,t+1}$ is given by:

$$\mathbf{Q}_{\ell(t,t+1)} = \Big(\vec{\mathbf{Q}}_{\ell t} + \boldsymbol{\Upsilon}_{\boldsymbol{\ell}} - \bar{\boldsymbol{\Lambda}}_\ell \bar{\boldsymbol{\Pi}}_\ell \tilde{\mathbf{Q}}_t^{-1} \bar{\boldsymbol{\Pi}}_\ell \bar{\boldsymbol{\Lambda}}_\ell\Big)^{-1} \bar{\boldsymbol{\Pi}}_\ell \bar{\boldsymbol{\Lambda}}_\ell \tilde{\mathbf{Q}}_t^{-1},$$
$$\tilde{\mathbf{Q}}_t = \bar{\boldsymbol{\Lambda}}_\ell + \bar{\boldsymbol{\Lambda}}_{\ell-1} + \bar{\mathbf{Q}}_{\ell,t+1}.$$

We summarise our inference algorithm to learn the model parameters in Algorithm 1. Specifically, we initialize the parameters of the variational distributions and iteratively updating them, according to their derivations as in the above, until convergence.

A. PREDICTION

After computing the posterior distribution, we will predict the request rates at H-step ahead from time slot T, for all $H > 0$, which can be obtained by computing the mean of the posterior predictive distribution, $p(r_{mn,T+H}|\mathbf{D}_{1:T})$, as:

$$\begin{aligned}\bar{r}_{m,n,T+H} &\triangleq E\big[r_{m,n,T+H}|\mathbf{D}_{1:T}\big] \\ &= E_{q(.)}\Big[\mathbf{A}(:,m)^T E_{p\left(\{\mathbf{x}_{\ell,T+H}\}_{h,\ell}\right)}\big[\exp\big(\mathbf{X}_{0,T+H}\big)\big]\mathbf{B}(:,n)\Big] \\ &= \left(\frac{\boldsymbol{\alpha}_{a_m}}{\boldsymbol{\beta}_{a_m}}\right)^T E_{q(.)p\left(\{\mathbf{x}_{\ell,T+h}\}_{h,\ell}\right)}\big[\exp\big(\mathbf{X}_{0,T+H}\big)\big]\left(\frac{\boldsymbol{\alpha}_{b_n}}{\boldsymbol{\beta}_{b_n}}\right),\end{aligned} \quad (32)$$

where

$$p\Big(\big\{\mathbf{x}_{\ell,T+h}\big\}_{h,\ell}\Big) = \prod_{h=1}^{H}\prod_{\ell=0}^{L} p\big(\mathbf{x}_{\ell,T+h}|\mathbf{x}_{\ell,T+h-1}, \mathbf{x}_{\ell+1,T+h}\big).$$

This requires the computation of the expectation of $\exp(\mathbf{X}_{0,T+H})$ under the marginal posterior predictive $q(.)p(\{\mathbf{x}_{\ell,T+h}\}_{h,\ell})$. Because this expectation does not have an analytical expression, we instead use the Jensen's inequality and approximate (32) by:

$$\bar{r}_{m,n,T+H} \approx \left(\frac{\boldsymbol{\alpha}_{a_m}}{\boldsymbol{\beta}_{a_m}}\right)^T \exp\left(E_{q(.)p\left(\{\mathbf{x}_{\ell,T+h}\}_{h,\ell}\right)}\big[\mathbf{X}_{0,T+H}\big]\right) \times \left(\frac{\boldsymbol{\alpha}_{b_n}}{\boldsymbol{\beta}_{b_n}}\right). \quad (33)$$

The operation in (33) requires to compute the expectation of $\mathbf{x}_{0,T+H}$ under $q(.)p(\{\mathbf{x}_{\ell,T+h}\}_{h,\ell})$, which is given by the following recursive formula:

$$E\big[\mathbf{x}_{\ell',T+h}\big] = \sum_{\ell=\ell'}^{L} \bar{\boldsymbol{\Pi}}_\ell E\big[\mathbf{x}_{\ell,T+h-1}\big], \quad \forall \ell' = 0, \ldots, L, h = 2, \ldots, H, \quad (34)$$

with initial conditions:

$$E\big[\mathbf{x}_{\ell',T+1}\big] = \sum_{\ell=\ell'}^{L} \bar{\boldsymbol{\Pi}}_\ell \mathbf{m}_{\ell,T}, \quad \forall \ell' = 0, \ldots, L.$$

We use the H-step ahead predictions in (33) to design a multistage caching policy presented in Section IV.

Algorithm 1: The Coordinate Ascent VB Algorithm

Input: Content requests, $\mathbf{D}$
Output: Estimated posterior density

Random initialisation of parameters $\mathbf{h} = \big\{\mathbf{A}, \hat{\mathbf{a}}, \mathbf{B}, \hat{\mathbf{b}}, \{\{\{\hat{\mathbf{D}}_{mnt}\}_{m=1}^{M}\}_{n=1}^{N}\}_{t=1}^{T}, \{\{\mathbf{x}_{\ell t}\}_{t=0}^{T}, \boldsymbol{\Lambda}_\ell, \boldsymbol{\Pi}_\ell\}_{\ell=0}^{L}\big\}$;
repeat
  Update $\breve{\mathbf{p}}_{mnt}$ using (17), $\forall m = 1, \ldots M, n = 1, \ldots, N, t = 1, \ldots, T$;
  Update $\mathbf{m}_{\ell t}, \mathbf{Q}_{\ell t}$ using (30) and (21), $\forall t = 0, \ldots, T$, $\forall \ell = 0, \ldots, L$;
  Update $\boldsymbol{\eta}_\ell, \boldsymbol{\kappa}_\ell, \boldsymbol{\tau}_\ell, \boldsymbol{\rho}_\ell$ using (19), $\forall \ell = 0, \ldots, L$;
  Update $\boldsymbol{\alpha}_{a_m}, \boldsymbol{\beta}_{a_m}, \boldsymbol{\alpha}_{b_n}, \boldsymbol{\beta}_{b_n}, \alpha_{\hat{a}_n}, \beta_{\hat{a}_m}, \alpha_{\hat{b}_n}, \beta_{\hat{b}_n}$ using Tab. 1 $\forall m = 1, ...M, n = 1, \ldots, N$;
until *Convergence*;

B. COMPUTATIONAL COMPLEXITY

We compute the per-iteration computational complexity of using the developed VB algorithm for approximating the posterior distribution in (9). The individual contributions of updating variational parameters to the overall per-iteration time complexity are as follows. *i*) the time complexity of tpdating the parameters of each $\hat{\mathbf{d}}_{mnt}$, $\forall m = 1, \ldots, M, n = 1, \ldots, N, t = 1, \ldots, T$, is $\mathcal{O}(TK_1K_2T_{nnz})$ where T_{nnz} is the number of non-zero requests during the most densely requested time slot. *ii*) The time complexity of updating the parameters of $\mathbf{A}, \hat{\mathbf{a}}, \mathbf{B}$ and $\hat{\mathbf{b}}$ is $\mathcal{O}(NK_1 + MK_2)$. *iii*) The time complexity of updating the parameters of $\mathbf{x}_{\ell t}$, $\forall \ell = 0, \ldots, L, t = 0, \ldots, T$ is $\mathcal{O}(TK_1K_2(L+1))$. *iv*) The time complexity of updating the parameters of Π_ℓ, Λ_ℓ, $\forall \ell = 0, \ldots, L$, is $\mathcal{O}(K_1K_2(L+1))$. Overall, the time complexity of the VB algorithm is $\mathcal{O}(TK_1K_2T_{nnz} + MK_1 + NK_2 + TK_1K_2(L+1))$. Note that the update of each parameter depends only on the other parameters in its Markov blanket, where the Markov blanket consists of the variable's parents, co-parents, and children in the graphical representation of the Bayesian model, and is independent of other parameters. For instance, the updates of a content factors are independent of all other contents (and similarly for a cell factors). Thus, we can easily parallelize the posterior inference computations to scale up the performance of the VB algorithm.

Remark: Note that, in practice, the number of contents is not fixed and new contents are released over time. In such cases, our proposed dynamical model is still capable of providing accurate popularity prediction with some minor modifications. A simple procedure is to add a dimension in the content space in the tensor when a new content is released. The initial values for the latent factors of this new content are sampled from the prior distribution and will be updated after observing its requests. Nevertheless, an issue with this procedure is that the computational complexity of the learning algorithm increases as more new contents are released over time. To tackle this issue, we can first obtain a coarse popularity prediction by using a much simpler algorithm (e.g., ARMA) to select the most M temporal

popular contents, where M is chosen by the available computation resource at the mobile network operator. Subsequently, these M contents are fed into the dynamical model for higher prediction accuracy. In this way our dynamical model would have a fixed computational complexity in terms of the number of contents. We believe that the time varying nature of the number of contents is crucial to practical applications, which is our next target in our future work for a more detailed algorithm design.

IV. COOPERATIVE CACHING PROBLEM FORMULATION

In this section, we formulate an adaptive cost-aware proactive cooperative caching policy. Following the system model explained in Section II, we assume that the BSs can cooperate via dedicated inter-BS links, e.g., the X2 link. The connectivity between BS n and BS n' is denoted by $\delta_{nn'} \in \{0, 1\}$, where $\delta_{nn'} = 1$ if BS n is connected to Bs n', otherwise $\delta_{nn'} = 0$. The set of BSs connected to BS n is denoted by $\mathcal{B}(n) = \{n' : \delta_{nn'} = 1\}$. When receiving a request for a content, BS n directly serves its users if the content is available in its cache, otherwise it retrieves the content from the neighboring BSs that have the content. If the content is available neither in its cache nor in the other BSs' caches, the BS downloads the content from the content server which holds all the contents via the back-haul link. In this scenario, the content delivery phase is jointly designed with the content placement and routing decisions. In particular, we design a cost efficient caching policy which adaptively utilizes the network resources, i.e., the cache memory and the link bandwidth, over time and is formulated as[2]:

$$\min_{y_{nn'mt}, S_{nt} f_{nn't}} \quad \sum_{t\in\mathcal{T}} C_{1t} + C_{2t} + C_{3t} + C_{4t} \tag{35a}$$

$$\text{s.t.} \quad \sum_{m\in\mathcal{M}} r_{mnt} y_{nn'mt} s_m \le f_{nn't},$$
$$\forall n \in \mathcal{B}(n'), n' \in \mathcal{B}, t \in \mathcal{T} \tag{35b}$$

$$\sum_{m\in\mathcal{M}} s_m y_{nnmt} \le S_{nt}, \forall n \in \mathcal{B}, t \in \mathcal{T} \tag{35c}$$

$$y_{nn'mt} \le y_{n'n'mt}, \forall n \in \mathcal{B}, n' \in \mathcal{B}(n),$$
$$m \in \mathcal{M}, t \in \mathcal{T} \tag{35d}$$

$$\sum_{n'\in\mathcal{B}(n)} y_{nn'mt} + y_{nnmt} \le 1 \quad \forall n \in \mathcal{B},$$
$$m \in \mathcal{M}, t \in \mathcal{T} \tag{35e}$$

$$y_{nn'mt} \in \{0, 1\}, \tag{35f}$$

$$S_{nt} \in [0, S_{max}], \tag{35g}$$

$$f_{nn't} \in \left[0, f_{\max}\right], \tag{35h}$$

where $\mathcal{T} = \{1, \ldots, T\}$,

$$C_{1t} \triangleq \sum_{n\in\mathcal{B}} \sum_{m\in\mathcal{M}} r_{mnt} s_m c_n \left(1 - \sum_{n'\in\mathcal{B}} y_{nn'mt}\right),$$

$$C_{2t} \triangleq \sum_{n\in\mathcal{B}} \sum_{n'\in\mathcal{B}(n)} c_{nn'} f_{nn't}, \quad C_{3t} \triangleq \sum_{n\in\mathcal{B}} c'_n S_{nt},$$
$$C_{4t} \triangleq \sum_{n\in\mathcal{B}} \sum_{m\in\mathcal{M}} c_n s_m y_{nnm(t-1)} \big(1 - y_{nnm(t-1)}\big).$$

In problem (35), the objective function is a sum of four costs: *i*) C_{1t} measures the download cost from the content server; *ii*) C_{2t} is the cost for content routing among the BSs where $f_{nn't}$ is the amount of bandwidth allocated between BS n and BS n' at time t; *iii*) C_{3t} is the caches storage cost where S_{nt} is the amount of memory assigned to the cache at BS n at time time t; and *iv*) C_{4t} is the cache refreshment cost. s_m is the size of content m, $c_{nn'}$ is the cost for transferring data between BS n and BS n', c_n is the cost for transferring data from the content server to BS n and c'_n is the cache storage cost. Moreover, $y_{nn'mt}$ is decision variable defined as:

$$y_{nn'mt} = \begin{cases} 1 & \text{if BS } n' \text{ sends content } m \text{ to BS } n \\ 0 & \text{otherwise.} \end{cases} \tag{36}$$

From the definition of $y_{nn'mt}$, it follows that y_{nnmt} is the cache placement decision variable whose value equals to one if content m is stored at cache n, and zero otherwise.

In problem (35), constraint (35b) ensures the link capacity between BS n' and BS n is less than assigned $f_{nn't}$. Constraint (35c) guarantees the cache memory at BS n is less than assigned S_{nt}. Constraint (35d) ensures that BS n' can send content m to BS n if it is stored in its local BS. Constraint (35e) ensures that BS n can only store content m in its cache or fetch it from only one BS. Finally, constraints (35f), (35g) and (35h) represent the space restriction of the decision variables (i.e., $y_{nn'mt}, f_{nn't}$ and S_{nt}), where $S_{\max}$ and $f_{\max}$ are respectively the maximum cache memory and the maximum link capacity. Note that since, in practice, the caches are large we assume S_{nt} is a continuous real number and can take any real number.

The optimization problem (35) is coupled over time slots thorough C_{4t}. Specifically, a decision at present time depends on the content requests in the future [27]. For instance, it may be optimal to remove a content from a cache for the current time slot if we ignore the future requests. This decision may not be optimal if we knew the content requests during the next time slots. Specifically, the request for this content may increase in the next time slots and therefore keeping this content can save the cost for fetching the content from the server for the next time slot. This means that in order to obtain a global solution, problem (35) needs to be solved jointly over all time slots. This is not feasible in online settings because the future requests are unavailable. A simple and common approach is to ignore the future requests and solve a one shot optimization [27]. Ignoring the future requests, however, degrades the caching performance. In the following section, we propose an approach to achieve improved performance.

A. H-STEP AHEAD PREDICTIVE CACHE OPTIMIZATION

To improve the quality of the solution, we exploit the latent trends in the requests extracted by the Bayesian dynamical

2. We only consider the overall content delivery cost at BS-level and the radio transmission cost does not change our formulation because it can be added up to the BS-level cost.

model and propose a H-step ahead predictive optimization problem. In particular, given the cache state $y_{nnmt'}$ at time slot t', we solve

$$\min_{y_{nn'mt}, S_{nt} f_{nn't}} \sum_{t=t'+1}^{t'+H} C_{1t} + C_{2t} + C_{3t} + C_{4t} \tag{37a}$$

$$\text{s.t. } (35b), (35c), (35d), (35e), (35f), (35g), (35h), \quad \forall t \in \mathcal{T}'. \tag{37b}$$

where $\mathcal{T}' = \{t'+1, \ldots, t'+H\}$. The caching policy requires the true popularity of contents over the next H time slots, $r_{m,n,t'+h}$, $h = 1, \ldots, H$, which are approximated by the predictions computed in (33), i.e., $r_{m,n,t'+h} \approx \bar{r}_{m,n,t'+h}$, $h = 1, \ldots, H$. We note that as a result of multiplication of $y_{nm,t'+h}$ and $y_{nm,t'+h+1}$ in cost function $C_{4,t'+h}$, $h = 1, \ldots, H$, problem (37) is nonlinear mixed-integer programming which is not trivial to solve. Therefore, we equivalently transform the non-linear problem in (37) into a linear problem by linearizing the non-linear terms in C_{4t} using the following Proposition [39]:

Proposition 1: Using the transformation $\chi_{nmh} = y_{nm,t'+h} y_{nm,t'+h+1}$ the following constraints linearize the problem (37).

$$\chi_{nmh} - y_{nm,t'+h} \leq 0, \quad \chi_{nmh} - y_{nm,t'+h+1} \leq 0, \tag{38}$$

where $0 \leq \chi_{nmh} \leq 1$.

Proof: The proof can be derived in a similar way as in [39, Proposition 2]. ∎

Under the above linearization procedure, we can now formulate a linear mixed-integer programming formulation of problem (37) as follows:

$$\min_{y_{nn'mt}, S_{nt} f_{nn't}, \chi_{nmh}} \sum_{t=t'+1}^{t'+H} C_{1t} + C_{2t} + C_{3t} + \tilde{C}_{4t} \tag{39a}$$

$$\text{s.t. } (35b), (35c), (35d), (35e), (35f), (35g), (35h), \ \forall t \in \mathcal{T}' \tag{39b}$$

$$\chi_{nmh} - y_{nm,t'+h} \leq 0, \forall m \in \mathcal{M}, n \in \mathcal{B}, h \in \{1, \ldots, H-1\} \tag{39c}$$

$$\chi_{nmh} - y_{nm,t'+h+1} \leq 0, \forall m \in \mathcal{M}, n \in \mathcal{B}, h \in \{1, \ldots, H-1\} \tag{39d}$$

$$0 \leq \chi_{nmh} \leq 1, \forall m \in \mathcal{M}, n \in \mathcal{B}, h \in \{1, \ldots, H-1\} \tag{39e}$$

where

$$\tilde{C}_{4,t'+h} = \sum_n \sum_m c_n s_m \hat{y}_{nm,t+h},$$

$$\hat{y}_{nm,t+h} = \begin{cases} y_{nnm,t'+1}(1 - y_{nnmt'}) & \text{if } h = 1 \\ y_{nnm,t'+h} - \chi_{nmh} & \text{otherwise} \end{cases}.$$

An optimal solution of problem (39) can be obtained by using global optimization methods, e.g., branch and bound [40].

Since the optimization problem in (39) is mixed-integer programming, there is no guarantee to be solved in polynomial time. Considering the stringent requirement and large-scale of the caching problem, it may not be efficient to implement the exact methods for optimal solution. In the following section we, develop a polynomial time algorithm to approximate the solution of (39).

B. A POLYNOMIAL TIME CACHE OPTIMIZATION

A common approach to approximate a solution of (39) is to relax the problem by replacing binary constraints with continuous interval [0, 1]. This convex relaxation approach often suffers from the fact that the computed solutions can be far from binary and that subsequent heuristic method is required to obtain binary solutions which it may substantially degrade the quality of the solutions. An approach to enhance the solutions is to incorporate a regularization function as a measure of relaxation tightness. In this work, we consider a penalty function $g(x) = x(1-x)$ [40] and relax the optimization problem at time slot t as:

$$\min_{y_{nn'mt}, S_{nt} f_{nn't}, \chi_{nmh}} \sum_{t=t'+1}^{t'+H} C_{1t} + C_{2t} + C_{3t} + \tilde{C}_{4t} + \sum_{m \in \mathcal{M}} \sum_{n \in \mathcal{B}} \sum_{n' \in \mathcal{B}} \zeta_{nn'mt} g(y_{nn'mt}) \tag{40a}$$

$$\text{s.t. } (35b), (35c), (35d), (35e), (35g), (35h), (39c), (39d), (39e), \quad \forall t \in \mathcal{T}' \tag{40b}$$

$$y_{nn'mt} \geq 0, \ \forall t \in \mathcal{T}', \forall m \in \mathcal{M}, n \in \mathcal{B}, n' \in \mathcal{B}, \tag{40c}$$

where $\zeta_{nn'mt}$ is sufficiently large penalty parameter. We can see that for binary solutions the penalty function term vanishes and problem (40) becomes equivalent to problem (39). Note that the minimization problem in (40) is not convex because it is sum of a linear function and a concave function, due to the term $-x^2$ in penalty function $g(x)$, and therefore it can not be solved efficiently. Since the objective function is sum of a linear function and a concave function, we resort to difference of convex (DC) programming approach [41] to solve the problem. The DC procedure is a heuristic algorithm for finding a local optimum by convexifying a convex-concave function which is commonly performed by linearizing the concave term. The DC programming for solving the caching policy in (40) is depicted in Algorithm . Specifically, at iteration ith of the algorithm, we solve problem (40) by linearizing the penalty function term around $y^i_{nn'mt}$. In lines 5-9, we increase the penalty parameters by multiplication with a factor φ only if the improvement of the solutions towards binary values measured by $g(y^{i+1}_{nn'mt})$ is not decreased by a factor δ over the previous iteration. This form of update guarantees the convergence of the algorithm and mitigates numerical issues caused when the penalty parameters grow too large [40]. To check the convergence of the algorithm, a reasonable stopping criterion is when the improvement in the convexified objective function is less than small threshold ϱ.

The DC programming procedure does not guarantee to yield binary solutions and rounding the continuous values

Algorithm 2: DC Programming for Caching Policy

input: Initial values $y^0_{nn'mt}$, $\zeta^0_{nn'm} > 0$, $\varphi > 1$, φ_{max}

Set $i = 0$;

repeat

Set $y^{i+1}_{nn'mt}$ to argument $y_{nn'mt}$ of problem:

$$\min_{y_{nn'mt}, s_{nt} f_{nn't}, \chi_{nmh}} \sum_{t=t'+1}^{t+H} C_{1t} + C_{2t} + C_{3t} + \tilde{C}_{4t} + \sum_{m\in\mathcal{M}} \sum_{n\in\mathcal{B}} \sum_{n'\in\mathcal{B}} \zeta^i_{nn'mt} y_{nn'mt} \nabla g(y^i_{nn'mt})$$

s.t. : (40b), (40c),

Construct set $\mathcal{X} = \{n, n' \in \mathcal{B}, m \in \mathcal{M}, t \in \mathcal{T}' | g(y^{i+1}_{nn'mt}) \geq \delta g(y^i_{nn'mt})\}$;

if $\forall n, n', m, t \in \mathcal{X}$ **then**

$\zeta^{i+1}_{nn'mt} = \min(\varphi \zeta^i_{nn'mt}, \varphi^{\max})$;

else

$\zeta^{i+1}_{nn'mt} = \zeta^i_{nn'mt}$;

$i = i + 1$;

until *Convergence*;

TABLE 2. The values of the dynamical model parameters.

Parameters	Values
L	2
K_2	2
$\hat{\alpha} = \hat{\beta}$	1
η_ℓ	0.5
κ_ℓ	10^{-2}
ρ_ℓ	10^{-4}
$\mathbf{Q}_{0,init}$	$2\mathbf{I}$
$\mathbf{Q}_{1,init}$	$20\mathbf{I}$
$\mathbf{Q}_{2,init}$	$200\mathbf{I}$

may violate some of the constraints. To provide a feasible binary solution, we sort the contributions of the continuous solutions to the objective function in increasing order. Then, in a greedy manner, we successively remove a content from the caches or delete a communication link which has the smallest contribution in the objective function until to satisfy all the violated constraints.

V. NUMERICAL RESULTS

In this section, we numerically evaluate the performance of the proposed dynamical hierarchical Poisson-Gaussian trend (DHPGT) model in Section II and the proposed proactive caching policy in Section IV. To provide exponentially shaped gamma priors for the sparsity of the latent factors, we set $\alpha = \beta = \gamma = 0.5$ unless otherwise stated. The rest of the model hyper-parameters are provided in Table 2. All simulations are run with MATLAB version 2019a on 64bit operating system with processor Intel Core i7-6820HQ CPU @2.70GHz and 8 GB RAM.

A. SYNTHETIC DATA

In this part, we show the performance of DHPGT model using syntactically generated requests. We first evaluate the prediction accuracy of the developed VB approximation. We use the mean absolute error (MAE) as a performance metric:

$$MAE = \frac{1}{MN} \sum_{m,n} |r_{mn,T+1} - \bar{r}_{mn,T+1}|,$$

To generate the synthetic data, we set $M = 25$, $N = 6$, $K_1 = 8$, $\mathbf{\Lambda}_0 = 4\mathbf{\Lambda}_1 = 16\mathbf{\Lambda}_2 = 0.25 \times 10^{-4}\mathbf{I}$, $\mathbf{\Pi}_0 = 0.9\mathbf{I}$, $\mathbf{\Pi}_1 = 0.95\mathbf{I}$ and $\mathbf{\Pi}_3 = 0.98\mathbf{I}$. Additionally, the values of the contents and cells latent factors are randomly drawn from their priors. The results are taken average over 25 different data realizations. In this scenario, we assume that the true latent dimensions, K_1, is unknown and we fit the model with dimensions different from the true one. Fig. 3 shows the MAE versus different values of K_1 for different number time slots T. We can see that, for a fixed T, as K_1 increases the MAE decreases. This is expected since by increasing K_1 the flexibility of the model increases and the data requests be explained more accurately. However, it can be seen that after a specific value of K_1 the MAE increases slightly which is due to over-fitting problem. In addition, we observe that the minimum value of the MAE is attained at higher K_1 values as the number of time slots T increases. In particular, for $T = 75$ and $T = 100$ the minimum value of the MAE occurs at $K_1 = 8$ which is the true latent dimensions. Moreover, we can observe that for small values of K_1 (e.g., at 2 and 4) the MAE increases as T increases. This happens because of under-fitting problem i.e., the model cannot explain the data well with a very small number of parameters.

Fig. 4 illustrates the convergence behavior of variational lower bound to log marginal likelihood for different values of K_1 for $T = 25$. We can see that the variational lower bound increases as the algorithm progresses. Specifically, the developed coordinate ascent VB guarantees to increase at each iteration since each update has a unique solution. It can be seen that as K_1 increases, the VB algorithm takes more time to converge. This is expected since more parameters are required to be estimated as K_1 increases. Moreover, we observe that the variational lower bound increases as K_1 increases from 3 to 5 which indicates that with $K_1 = 3$ the model underfits and more factors are required to describe the data well. On the other hand, we see that the variational lower bound decreases as K_1 increases from 5 to 10. This is because the log marginal likelihood penalizes models with too many parameters, i.e., overfitted models, and assigns them less likelihood [42]. This also confirms our results in Fig. 3.

Furthermore, in Fig. 5, we explore the latent structures inferred by the model. Fig. 5-a shows the trajectories of actual requests, the true request rates and the inferred means of the request rates for a content in a cell over time. We can see that the VB approximation tracks the true request rate accurately. It is also observed that the requests for this content has an increasing trend followed by a decreasing trend over time. Fig. 5-(b) shows the inferred mean of the time latent factor. For the ease of visualization, we only show the latent time factor which has the largest content and

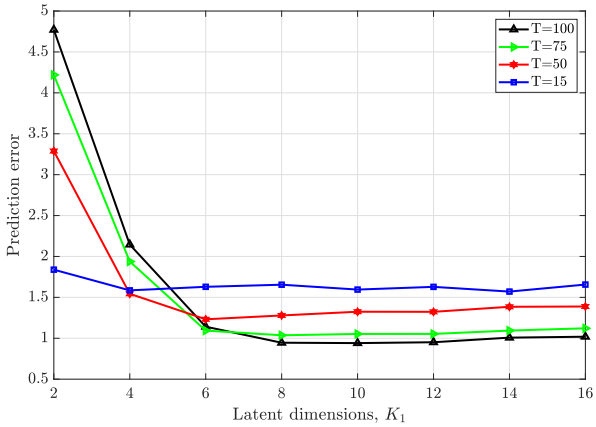

FIGURE 3. The estimation error versus latent dimensions K_1, where the true latent dimensions is 8.

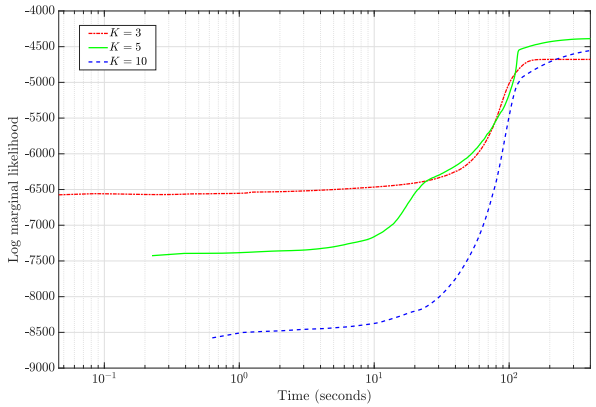

FIGURE 4. The convergence of variational lower bound to log marginal likelihood for different values of K_1.

cell factor weights denoted by k^*. We can observe that this time factor can explain the temporal pattern of the content requests very well. Fig. 5-(c) illustrates the inferred $\mathbf{m}_{0t}(k^*)$ which it can be seen that it has the same temporal pattern as in Fig. 5-(b). Fig. 5-(d) illustrates $\mathbf{m}_{1t}(k^*)$ which we can observe that it captures linear temporal trends in $\mathbf{m}_{0t}(k^*)$. Particularly, when $\mathbf{m}_{0t}(k^*)$ increases (or it has a positive slop), $\mathbf{m}_{1t}(k^*)$ is positive, and when $\mathbf{m}_{0t}(k^*)$ decreases (or it has a negative slop), $\mathbf{m}_{1t}(k^*)$ is negative. Furthermore, Fig. 5-(e) shows the interfered mean of $\mathbf{m}_{2t}(k^*)$ which, with the same argument, captures linear temporal trends in $\mathbf{m}_{1t}(k^*)$.

B. REAL-WORLD DATA

In this section, we show results on prediction accuracy using two real-world MovieLens-20M [43] and Netflix [44] datasets. We choose ratings on a monthly basis where we select the most popular $M = 100$ movies and the most active 1000 users. We only use the ratings during the last 200 and 70 months for the MovieLens and the Netflix datasets respectively because the majority of the ratings are provided during theses time frames. Moreover, since our focus is on count data requests, we discretize non-integer ratings to the their nearest integer values. Additionally, since the datasets

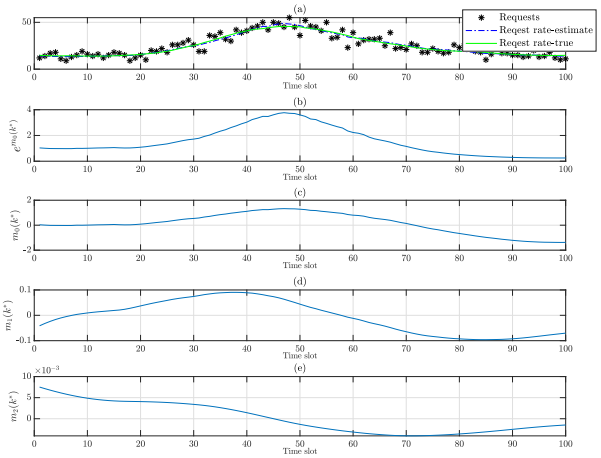

FIGURE 5. The requests trajectory, time latent factor, temporal locality of the time factor, zeroth, first, second latent layers.

do not provide information about the locations of the users, we randomly distribute the users across $N = 10$ cells, each has 100 users. Note that the datasets consist of ratings for each users for movies. Therefore, we have aggregated the users' ratings to obtain BS-level requests in order to adapt the data to our model.

We compare the proposed prediction model scheme against the following two baseline models:

- *Dynamical Poisson Gaussian (DPG) Model [27]:* This model employs a CP tensor decomposition while ignores the temporal trends.
- *Dynamical Independent Hierarchical Poisson-Gaussian (DIHPG) Model:* This model is a simplification of DHPGT model while ignores the temporal trends and the spatial correlations. For this model, we fit DHPGT model to the request observations in cell n independent of the other cells by setting $K_2 = 1$, $\mathbf{B}(n, :) = \mathbf{1}$, $\hat{\mathbf{b}}(n) = 1$ and $L = 0$.
- *ARMA Model:* the requests are modeled as $d_{mnt} = \sum_{i=1}^{I} \varphi_{ni} d_{m,n,t-i} + \sum_{j=1}^{J} \alpha_{nj} n_{m,n,t-j}$ for $t = 1, \ldots, T$ and $m = 1, \ldots, M$, where I and J specify the order of the model, φ_i and α_j are the parameters, and $n_{m,t}$ is white noise error term. We set $I = J = 7$ which is also used in [10]. For its implementation, we used the econometrics MATLAB toolbox. Note that due to the Gaussian noise assumption, ARMA model may predict negative values for the requests. To provide meaningful predictions, we round the negative values to a small positive value e.g., 10^{-5}.

Since the true underlying requests are unknown for the real-world datasets, we define $MAE = \frac{1}{MNT} \sum_{m,n,t} |\mathbf{D}_t(m, n) - \bar{r}_{mnt}|$ which measure the dispensary between the instantaneous requests and the predicted request rates.

Table 3 shows the prediction error of different models on the datasets for various prediction time horizon H. We also give the average burstiness of each dataset which we

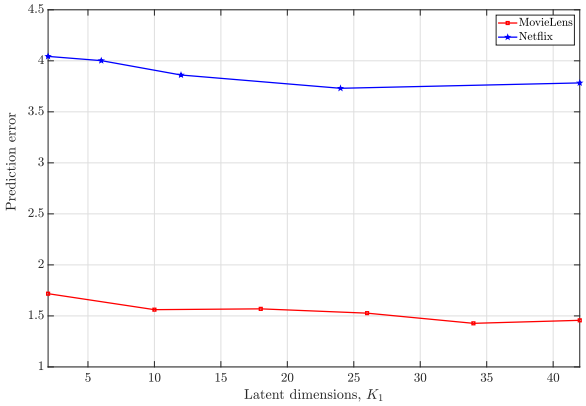

FIGURE 6. Prediction error versus model complexity, K_1, for different datasets.

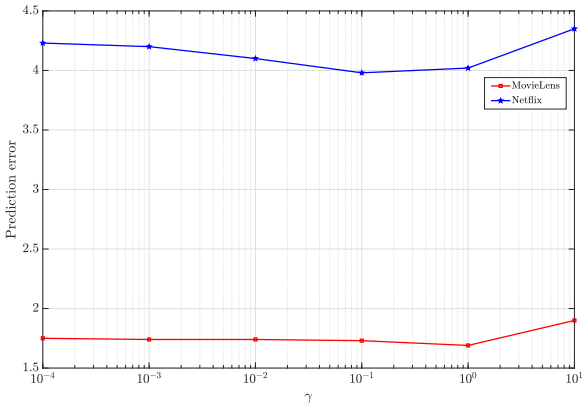

FIGURE 7. Prediction error versus the sparsity level, γ, of the latent factors A and B for different datasets.

define as

$$Bur = \frac{1}{MN(T-1)} \sum_{t} |\mathbf{D}(m,n,t+1) - \mathbf{D}(m,n,t)|,$$

From the table, it can be seen that DHPGT model is more accurate than DIHPG, DPG and ARMA models. Moreover, for DHPGT model, increasing the number latent layers, L, improves the prediction accuracy. This indicates that, by increasing L, DHPGT model can capture a deeper hidden trend structures in the requests and therefore a higher accuracy is attained. Additionally, we observe that DIHPG model has the worst prediction accuracy among DHPGT and DPG models. This is because it ignores important hidden structures in the requests i.e., the temporal trends and the spatial correlations. Furthermore, it can be observed that as the prediction time horizon H increases, MAE increases for all the models which is due to the difficulty to predict a far away future time slot. We can also see that the MoveiLens dataset has a smaller error prediction with respect to the Netflix dataset. This can be explained by examining the burstiness of the datasets. Specially, since the Netflix dataset is more burst in comparison with the MoveiLens dataset, it is also more difficult to predict.

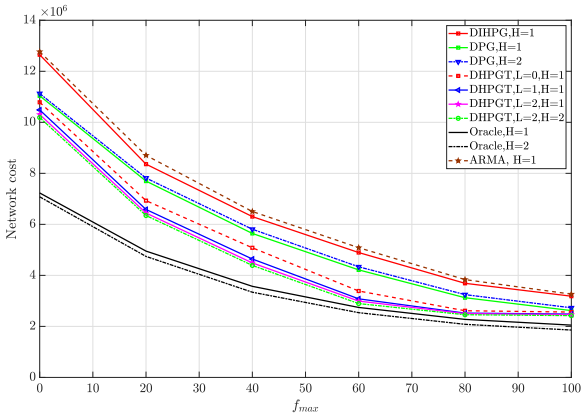

FIGURE 8. The achieved network cost versus link capacity.

We also show the prediction error of DHPGT model in terms of latent dimensions, K_1. Fig. 6 illustrates the prediction error versus K_1 for different datasets. In this scenario, we use $T = 69$ and $T = 199$ time slots for respectively the Netflix and the Movielens datasets for training and the last time slot for prediction. It can be observed that as K_1 increase the error decreases for both datasets. This is expected since by increasing K_1 the flexibility of the model increases and therefore it can explain the data more accurately. Moreover, for each dataset, it can be seen that after a specific value of K_1 ($K_1 = 24$ for the Netflix dataset and $K_1 = 34$ for the MovieLens dataset) the prediction error increases which is due to the overfitting problem. Furthermore, in Fig. 7, we show the prediction error versus the sparsity level of the latent factors $\mathbf{A}$ and $\mathbf{B}$ which is specified by γ for different datasets for $K_1 = 5$. From Fig. 7, we can observe that the model performs fairly robust for wide-range of γ values $(0.01 - 1)$ on both datasets.

C. CACHING PERFORMANCE

In this section, we study the impact of the prediction accuracy on the caching performance by examining on the Netflix dataset. Since the focus is to compare the performance of prediction models, without loss of generality, we assume all the contents have equal sizes of one unit. We set $c_n = 50$, $c_{n,n'} = 5$ and $c'_n = 2$. Furthermore, the results are obtained by computing the global optimum solution of the caching policy optimization in (39). In our implementation, we use the MOSEK package.

Fig. 8 illustrates the aggregate network cost versus f_{max} for $S_{max} = 25$ and for different time horizon prediction H. We also show an oracle approach which knows the requests in advance. As expected, the oracle gives a lower bound to the other prediction approaches. Moreover, it can be seen that DHPGT model outperforms the other models, i.e., DIHPG, DPG and ARMA models, and its performance improves as the number of latent layers L increase. This also confirms the results in Table 3. Additionally, we can see that the network cost decreases as H increases for the oracle approach. This indicates that solving the one shot caching

TABLE 3. Prediction error on different datasets.

Dataset	H	DHPGT, $L=2$	DHPGT, $L=1$	DHPGT, $L=0$	DIHPG	DPG	ARMA
Netflix Bur=4.82	1	3.92	3.95	4.03	4.51	4.16	4.60
	2	4.18	4.20	4.32	4.77	4.48	4.69
	3	4.40	4.40	4.56	4.87	4.75	4.89
MovieLens Bur=2.55	1	2.09	2.10	2.11	2.47	2.38	2.51
	2	2.13	2.13	2.15	2.49	2.40	2.53
	3	2.15	2.15	2.18	2.51	2.43	2.59

TABLE 4. The running time, in seconds, of MOSEK, relaxation and DC programming methods.

Method	$M=50, N=5$	$M=100, N=5$	$M=50, N=10$	$M=100, N=10$
MOSEK	200	638	5.016×10^3	1.7×10^4
DC programming	2	11	60	250
Relaxation	0.1	0.3	0.7	2

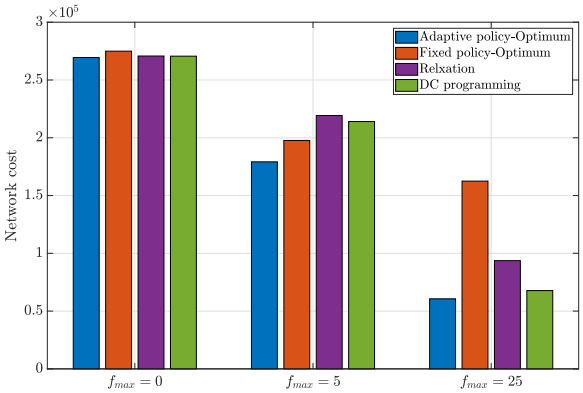

FIGURE 9. The achieved network cost for different approximation method.

policy is sub-optimal even if the true content requests is known in advance. We also observe that solving the caching policy for more than one time slot, $H=2$, can not improve the caching performance for non-trend models, e.g., DPG. On the other hand, our proposed trend model, DHPGT, reduces the network cost for $H=2$. This indicates that our trend model provides more robust and accoutre mean predictions for more than one time slot ahead in comparison to the non-trend-models. Furthermore, Fig. 8 shows that the network cost decreases as f_{max} increases. This is because the BSs can be more cooperative for content sharing which bypasses the need to download the contents via the costly back-haul links.

D. CACHING POLICY APPROXIMATION

Finally, we show numerical results on the performance of different approximation methods for the mixed-integer programming problem in (39). The content sizes are randomly generated from interval $[0,1]$. The parameters of the DC programming problem are set as $\delta=0.25, \varphi=2, \varphi_{max}=10^5, \varrho=10^{-6}$ and $\zeta^0_{nn'm}=2,\ \forall m\in\mathcal{M}, n, n'\in\mathcal{N}$. The results are obtained by choosing 10 randomly selected time slots from the Netflix dataset. The parameters of the caching policy is set as in Fig. 8. Fig. 9 illustrates the performance of different approximation methods for different values of f_{max}. We can see that the DC programming achieves a better performance in comparison to the linear relaxation programming and the performance gap increases as f_{max} increases. We also depict a fixed caching policy which does not optimize the cache sizes and the link capacities. Fig. 9 shows that the adaptive caching policy attains a better performance with respect to the fixed one. This is because the adaptive caching policy can adjust the network resources according the content requests which can save the network cost more efficiently.

To investigate the computational complexity of the optimization problems, we show the running time of different optimization methods in terms of the number of contents and cells in Table 4. The Table shows that the running time for the developed DC programming is much smaller than MSOEK solver and the relaxation method has the smallest running time. Moreover, the running time increases as the number of contents/cells increases for all the methods.

VI. CONCLUSION

In this paper, we introduced a flexible Bayesian dynamical model to capture time-varying content requests in practical edge caching systems. We first developed a probabilistic tensor factorization model which can capture interactions among contents over location and suitable for count-valued requests. Then, a Gaussian process model was used to capture complex non-linear temporal trends. The inference performed based on vartaional and Kalman smoother algorithms, and a simple-to-implement posterior approximation was derived. We subsequently designed a dynamic caching policy which can adapt the network recourses according to dynamics of the content requests. To overcome the non-convexity of the formulated problem, we developed an approximation algorithm based on difference of convex programming which can be solved in polynomial time. In the simulation results, the Bayesian dynamical model accuracy was examined on two real-world deatasets and we showed that it outperforms reference prediction methods.

APPENDIX
PROOF OF (19)

We derive the variational distribution over parameters $\boldsymbol{\lambda}_\ell(k), \boldsymbol{\pi}_\ell(k)$. First we compactly rewrite the dynamic evolution of latent variable $\mathbf{x}_{\ell t}$ as:

$$\hat{\mathbf{x}}_{k\ell} = \boldsymbol{\pi}_\ell(k)\hat{\hat{\mathbf{x}}}_{k\ell} + \hat{\mathbf{x}}_{k,\ell+1} + \hat{\mathbf{e}},$$

where $\hat{\mathbf{x}}_{k\ell} = [\mathbf{x}_{1\ell}(k), \ldots, \mathbf{x}_{T\ell}(k)]^T, \hat{\hat{\mathbf{x}}}_{k\ell} = [\mathbf{x}_{0\ell}(k), \ldots, \mathbf{x}_{\ell,T-1}(k)]^T$ and $\mathbf{e}_{k\ell} = [\mathbf{e}_{\ell 1}(k), \ldots, \mathbf{e}_{\ell T}(k)]^T$. We note that due the factorized form of the prior distribution the posterior distribution automatically has a form of:

$$q(\boldsymbol{\lambda}_\ell(k), \boldsymbol{\pi}_\ell(k)) = q(\boldsymbol{\pi}_\ell(k)|\boldsymbol{\lambda}_\ell(k))q(\boldsymbol{\lambda}_\ell(k)), \tag{41}$$

where we obtain

$$\begin{aligned}
&q(\boldsymbol{\lambda}_\ell(k), \boldsymbol{\pi}_\ell(k)) \\
&=\propto e^{E\left[\log\left(\prod_{t=1}^{T} p(\mathbf{x}_{\ell t}(k)|\mathbf{x}_{\ell,t-1}(k), \mathbf{x}_{\ell+1,t}(k))\right)\right]} \\
&\quad \times p(\boldsymbol{\lambda}_\ell(k)|\boldsymbol{\pi}_\ell(k))p(\boldsymbol{\pi}_\ell(k)) \\
&\propto \boldsymbol{\lambda}_\ell^{T/2}(k) e^{\frac{-\boldsymbol{\lambda}_\ell}{2}\sum_t (\mathbf{x}_{\ell t}(k) - \boldsymbol{\pi}_\ell(k)\mathbf{x}_{\ell,t-1}(k) - \mathbf{x}_{\ell+1,t}(k))^2} \\
&\quad \times p(\boldsymbol{\lambda}_\ell(k)|\boldsymbol{\pi}_\ell(k))p(\boldsymbol{\pi}_\ell(k)) \\
&\propto \mathcal{N}\left(\hat{\mathbf{x}}_{k\ell}; \boldsymbol{\pi}_\ell(k)\hat{\hat{\mathbf{x}}}_{k\ell} + \hat{\mathbf{x}}_{k\ell}, \boldsymbol{\lambda}_\ell(k)\mathbf{I}_{T/2}\right) \\
&\quad p(\boldsymbol{\lambda}_\ell(k)|\boldsymbol{\pi}_\ell(k))p(\boldsymbol{\pi}_\ell(k)).
\end{aligned} \tag{42}$$

The variational distribution of $\boldsymbol{\pi}_\ell(k)$ conditioned on $\boldsymbol{\lambda}_\ell(k)$ can be obtained as:

$$\begin{aligned}
q(\boldsymbol{\pi}_\ell(k)|\boldsymbol{\lambda}_\ell(k)) \propto \mathcal{N}\left(\hat{\mathbf{x}}_{k\ell}; \pi_\ell(k)\hat{\hat{\mathbf{x}}}_{k\ell} + \hat{\mathbf{x}}_{k\ell}, \boldsymbol{\lambda}_\ell(k)\mathbf{I}_{T/2}\right) \\
p(\boldsymbol{\lambda}_\ell(k)|\boldsymbol{\pi}_\ell(k)).
\end{aligned}$$

Since it is a product of two Gaussians, we obtain:

$$\begin{aligned}
&\log q(\boldsymbol{\pi}_\ell(k)|\boldsymbol{\lambda}_\ell(k)) \propto \log \mathcal{N}\left(\hat{\mathbf{x}}_{k\ell}; \pi_\ell(k)\hat{\hat{\mathbf{x}}}_{k\ell} + \hat{\mathbf{x}}_{k\ell}, \boldsymbol{\lambda}_\ell(k)\mathbf{I}_{T/2}\right) \\
&\quad \times \mathcal{N}\left(\pi_\ell(k); \eta_\ell, \frac{1}{\kappa_\ell \boldsymbol{\lambda}_\ell(k)}\right) \\
&\propto \frac{-\boldsymbol{\lambda}_\ell(k)}{2}\left(\hat{\mathbf{x}}_{k\ell} - \boldsymbol{\pi}_\ell(k)\hat{\hat{\mathbf{x}}}_{k\ell} - \hat{\mathbf{x}}_{k\ell+1}\right)^T \\
&\quad \times \left(\hat{\mathbf{x}}_{k\ell} - \boldsymbol{\pi}_\ell(k)\hat{\hat{\mathbf{x}}}_{k\ell} - \hat{\mathbf{x}}_{k\ell+1}\right) \\
&\quad - \frac{\kappa_\ell \boldsymbol{\lambda}_\ell(k)}{2}(\boldsymbol{\pi}_\ell(k) - \eta_\ell)^2 \\
&\propto \frac{-\lambda_\ell(k)}{2}\left(\pi_\ell^2(k)\hat{\hat{\mathbf{x}}}_{k\ell}^T\hat{\hat{\mathbf{x}}}_{k\ell} - 2\pi_\ell(k)\hat{\hat{\mathbf{x}}}_{k\ell}^T(\hat{\mathbf{x}}_{k\ell} - \hat{\mathbf{x}}_{k\ell+1})\right) \\
&\quad - \frac{\kappa_\ell \boldsymbol{\lambda}_\ell(k)}{2}(\boldsymbol{\pi}_\ell(k) - \eta_\ell)^2 \\
&\propto \frac{-\boldsymbol{\lambda}_\ell(k)}{2} \\
&\boldsymbol{\pi}_\ell^2(k)\left(\hat{\hat{\mathbf{x}}}_{k\ell}^T\hat{\hat{\mathbf{x}}}_{k\ell} + \kappa_\ell\right) - 2\boldsymbol{\pi}_\ell(k)\left(\hat{\hat{\mathbf{x}}}_{k\ell}^T(\hat{\mathbf{x}}_{k\ell} - \hat{\mathbf{x}}_{k\ell+1}) + \kappa_\ell\eta_\ell\right).
\end{aligned} \tag{43}$$

The above expression is a quadratic function of $\boldsymbol{\pi}_\ell(k)$ therefore it can be reformulated as a Gaussian density as:

$$\begin{aligned}
&q(\boldsymbol{\pi}_\ell(k)|\boldsymbol{\lambda}_\ell(k)) \\
&= \mathcal{N}\left(\boldsymbol{\pi}_\ell(k); \frac{\left(\hat{\hat{\mathbf{x}}}_{k\ell}^T(\hat{\mathbf{x}}_{k\ell} - \hat{\mathbf{x}}_{k\ell+1}) + \kappa_\ell\eta_\ell\right)}{\left(\hat{\hat{\mathbf{x}}}_{k\ell}^T\hat{\hat{\mathbf{x}}}_{k\ell} + \kappa_\ell\right)}, \right. \\
&\qquad \left. \frac{1}{\boldsymbol{\lambda}_\ell(k)\left(\hat{\hat{\mathbf{x}}}_{k\ell}^T\hat{\hat{\mathbf{x}}}_{k\ell} + \kappa_\ell\right)}\right).
\end{aligned}$$

In order to obtain $q(\boldsymbol{\lambda}_\ell(k))$, we first need compute the following marginal density:

$$\begin{aligned}
&\tilde{p}(\boldsymbol{\lambda}_\ell(k)) \propto \int e^{\left(E\left[\log p\left(\hat{\mathbf{x}}_{k\ell}|\hat{\hat{\mathbf{x}}}_{k\ell}, \hat{\mathbf{x}}_{k\ell+1}, \boldsymbol{\lambda}_\ell(k), \boldsymbol{\pi}_\ell(k)\right)\right]\right)} \\
&\quad \times p(\boldsymbol{\pi}_\ell(k)|\boldsymbol{\lambda}_\ell(k))d\boldsymbol{\pi}_\ell(k) \\
&\propto \int \boldsymbol{\lambda}_\ell^{\frac{T+1}{2}}(k) e^{\left(-\frac{\kappa_\ell \boldsymbol{\lambda}_\ell(k)}{2}(\boldsymbol{\pi}_\ell(k) - \eta_\ell)^2\right)} \\
&\quad \times e^{\left(\frac{-\boldsymbol{\lambda}_\ell(k)}{2}\left(\hat{\mathbf{x}}_{k\ell} - \boldsymbol{\pi}_\ell(k)\hat{\hat{\mathbf{x}}}_{k\ell} - \hat{\mathbf{x}}_{k\ell+1}\right)^T\left(\hat{\mathbf{x}}_{k\ell} - \boldsymbol{\pi}_\ell(k)\hat{\hat{\mathbf{x}}}_{k\ell} - \hat{\mathbf{x}}_{k\ell+1}\right)\right)} d\boldsymbol{\pi}_\ell(k) \\
&= \boldsymbol{\lambda}_\ell^{\frac{T+1}{2}}(k) e^{\left(\frac{-\boldsymbol{\lambda}_\ell(k)}{2}(\hat{\mathbf{x}}_{k\ell} - \hat{\mathbf{x}}_{k\ell+1})^T(\hat{\mathbf{x}}_{k\ell} - \hat{\mathbf{x}}_{k\ell+1}) + \kappa_\ell\eta_\ell^2\right)} \\
&\int e^{\left(\frac{-\boldsymbol{\lambda}_\ell(k)}{2}\left(\boldsymbol{\pi}_\ell^2(k)\left(\hat{\hat{\mathbf{x}}}_{k\ell}^T\hat{\hat{\mathbf{x}}}_{k\ell} + \kappa_\ell\right) - 2\boldsymbol{\pi}_\ell(k)\left(\hat{\hat{\mathbf{x}}}_{k\ell}^T(\hat{\mathbf{x}}_{k\ell} - \hat{\mathbf{x}}_{k\ell+1}) + \kappa_\ell\eta_\ell\right)\right)\right)} \\
&= \boldsymbol{\lambda}_\ell^{T/2}(k) e^{\frac{-\boldsymbol{\lambda}_\ell(k)}{2}\rho_{k\ell}}
\end{aligned} \tag{44}$$

where

$$\begin{aligned}
\rho_{k\ell} = &\left(\hat{\mathbf{x}}_{k\ell} - \hat{\mathbf{x}}_{k\ell+1}\right)^T\left(\hat{\mathbf{x}}_{k\ell} - \hat{\mathbf{x}}_{k\ell+1}\right) + \kappa_\ell\eta_\ell^2 \\
&- \frac{\left(\hat{\hat{\mathbf{x}}}_{k\ell}^T(\hat{\mathbf{x}}_{k\ell} - \hat{\mathbf{x}}_{k\ell+1}) + \kappa_\ell\eta_\ell\right)^2}{\left(\hat{\hat{\mathbf{x}}}_{k\ell}^T\hat{\hat{\mathbf{x}}}_{k\ell} + \kappa_\ell\right)}
\end{aligned}$$

Using the above, the variational density of is computed as:

$$q(\boldsymbol{\lambda}_\ell(k)) \propto \tilde{p}(\boldsymbol{\lambda}_\ell(k))p(\boldsymbol{\lambda}_\ell(k)),$$

which is straightforward to show that it can be formulated by a gamma density as:

$$q(\boldsymbol{\lambda}_\ell(k)) = \mathcal{G}(\boldsymbol{\lambda}_\ell(k); \tau_{k\ell}, \rho_{k\ell}),$$

where

$$\tau_{k\ell} = \tau_\ell + T/2.$$

REFERENCES

[1] S. Mehrizi, S. Chatzinotas, and B. Ottersten, "Content request prediction with temporal trend for proactive caching," in *Proc. IEEE 31th Annu. Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, 2020, pp. 1–7.

[2] *Global Mobile Data Traffic Forecast Update, 2016–2021*, Cisco, San Jose, CA, USA, 2017.

[3] E. Baştuğ *et al.*, "Big data meets telcos: A proactive caching perspective," *J. Commun. Netw.*, vol. 17, no. 6, pp. 549–557, 2015.

[4] H. Yu, L. Xie, and S. Sanner, "The lifecyle of a youtube video: Phases, content and popularity," in *Proc. 9th Int. AAAI Conf. Web Soc. Media*, 2015, pp. 533–542.

[5] A. Brodersen, S. Scellato, and M. Wattenhofer, "Youtube around the world: Geographic popularity of videos," in *Proc. ACM 21st Int. Conf. World Wide Web*, 2012, pp. 241–250.

[6] Y. Jiang, M. Ma, M. Bennis, F. Zheng, and X. You, "User preference learning-based edge caching for fog radio access network," *IEEE Trans. Commun.*, vol. 67, no. 2, pp. 1268–1283, Feb. 2019.

[7] S. Mehrizi, A. Tsakmalis, S. Chatzinotas, and B. Ottersten, "A Bayesian Poisson–Gaussian process model for popularity learning in edge-caching networks," *IEEE Access*, vol. 7, pp. 92341–92354, 2019.

[8] S. Bommaraveni, T. X. Vu, S. Chatzinotas, and B. Ottersten, "Active content popularity learning and caching optimization with hit ratio guarantees," *IEEE Access*, vol. 8, pp. 151350–151359, 2020.

[9] G. Garsun, M. Crovella, and I. Matta, "Describing and forecasting video access patterns," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 16–20.

[10] N. B. Hassine, D. Marinca, P. Minet, and D. Barth, "Popularity prediction in content delivery networks," in *Proc. IEEE 26th Annu. Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*, 2015, pp. 2083–2088.

[11] N. B. Hassine, D. Marinca, P. Minet, and D. Barth, "Caching strategies based on popularity prediction in content delivery networks," in *Proc. IEEE 12th Int. Conf. Wireless Mobile Comput. Netw. Commun. (WiMob)*, Oct. 2016, pp. 1–8.

[12] N. Garg, M. Sellathurai, V. Bhatia, B. Bharath, and T. Ratnarajah, "Online content popularity prediction and learning in wireless edge caching," *IEEE Tran. Commun.*, vol. 68, no. 2, pp. 1087–1100, Feb. 2020.

[13] W. Jiang, G. Feng, and S. Qin, "Optimal cooperative content caching and delivery policy for heterogeneous cellular networks," *IEEE Trans. Mobile Comput.*, vol. 16, no. 5, pp. 1382–1393, May 2017.

[14] V. Fedchenko, G. Neglia, and B. Ribeiro, "Feedforward neural networks for caching: N enough or too much?" *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 46, no. 3, pp. 139–142, 2019.

[15] H. Pang, J. Liu, X. Fan, and L. Sun, "Toward smart and cooperative edge caching for 5G networks: A deep learning based approach," in *Proc. IEEE/ACM 26th Int. Symp. Qual. Service (IWQoS)*, 2018, pp. 1–6.

[16] K. Thar, N. H. Tran, T. Z. Oo, and C. S. Hong, "DeepMec: Mobile edge caching using deep learning," *IEEE Access*, vol. 6, pp. 78260–78275, 2018.

[17] G. Paschos, E. Bastug, I. Land, G. Caire, and M. Debbah, "Wireless caching: Technical misconceptions and business barriers," *IEEE Commun. Mag.*, vol. 54, no. 8, pp. 16–22, Aug. 2016.

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

[19] C. Zhong, M. C. Gursoy, and S. Velipasalar, "A deep reinforcement learning-based framework for content caching," in *Proc. IEEE 52nd Annu. Conf. Inf. Sci. Syst. (CISS)*, 2018, pp. 1–6.

[20] Y. He, F. R. Yu, N. Zhao, V. C. M. Leung, and H. Yin, "Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach," *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 31–37, Dec. 2017.

[21] T. Zhang, Z. Wang, Y. Liu, W. Xu, and A. Nallanathan, "Caching placement and resource allocation for cache-enabling UAV NOMA networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 12897–12911, Nov. 2020.

[22] X. Xu, C. Feng, S. Shan, T. Zhang, and J. Loo, "Proactive edge caching in content-centric networks with massive dynamic content requests," *IEEE Access*, vol. 8, pp. 59906–59921, 2020.

[23] D. S. Berger, "Towards lightweight and robust machine learning for CDN caching," in *Proc. 17th ACM Workshop Hot Topics Netw.*, 2018, pp. 134–140.

[24] M. Hessel *et al.*, "Rainbow: Combining improvements in deep reinforcement learning," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 3215–3222.

[25] Z. Ghahramani, "Probabilistic machine learning and artificial intelligence," *Nature*, vol. 521, no. 7553, pp. 452–459, 2015.

[26] S. Mehrizi, A. Tsakmalis, S. ShahbazPanahi, S. Chatzinotas, and B. Ottersten, "Popularity tracking for proactive content caching with dynamic factor analysis," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC)*, Aug. 2019, pp. 875–880.

[27] S. Mehrizi, S. Chatterjee, S. Chatzinotas, and B. Ottersten, "Online spatiotemporal popularity learning via variational bayes for cooperative caching," *IEEE Trans. Commun.*, vol. 68, no. 11, pp. 7068–7082, Nov. 2020.

[28] X. Li, X. Wang, K. Li, Z. Han, and V. C. Leung, "Collaborative multi-tier caching in heterogeneous networks: Modeling, analysis, and design," *IEEE Trans. Wireless Commun*, vol. 16, no. 10, pp. 6926–6939, Oct. 2017.

[29] S. Ioannidis and E. Yeh, "Adaptive caching networks with optimality guarantees," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 44, no. 1, pp. 113–124, 2016.

[30] C. Liang, F. R. Yu, and X. Zhang, "Information-centric network function virtualization over 5G mobile wireless networks," *IEEE Netw.*, vol. 29, no. 3, pp. 68–74, May/Jun. 2015.

[31] I. V. Oseledets, "Tensor-train decomposition," *SIAM J. Sci. Comput.*, vol. 33, no. 5, pp. 2295–2317, 2011.

[32] A. Cichocki, N. Lee, I. Oseledets, A. Phan, Q. Zhao, and D. P. Mandic, *Tensor Networks for Dimensionality Reduction and Large-scale Optimization: Part 2 Applications and Future Perspectives*. London, U.K.: Now, 2017. [Online]. Available: https://ieeexplore.ieee.org/document/8187112

[33] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. G. Carbonell, "Temporal collaborative filtering with Bayesian probabilistic tensor factorization," in *Proc. SIAM Data Min.*, 2010, pp. 211–222.

[34] P. Gopalan, J. M. Hofman, and D. M. Blei, "Scalable recommendation with hierarchical Poisson factorization," in *Proc. 31st Conf. Uncertainty Artif. Intell. (UAI)*, 2015, pp. 326–335.

[35] R. Prado and M. West, *Time Series: Modeling, Computation, and Inference*. London, U.K.: Chapman and Hall, 2010.

[36] J. Winn and C. M. Bishop, "Variational message passing," *J. Mach. Learn. Res.*, vol. 6, pp. 661–694, Apr. 2005.

[37] J. F. C. Kingman, "Poisson processes," *Encyclopedia of Biostatistics*, vol. 6. Hoboken, NJ, USA: Wiley, 2005.

[38] J. Dauwels, "On variational message passing on factor graphs," in *Proc. IEEE Int. Symp. Inf. Theory*, Nice, France, 2007, pp. 2546–2550.

[39] T. Bektaş, J.-F. Cordeau, E. Erkut, and G. Laporte, "Exact algorithms for the joint object placement and request routing problem in content distribution networks," *Comput. Oper. Res.*, vol. 35, no. 12, pp. 3860–3884, 2008.

[40] D. P. Bertsekas, "Nonlinear programming," *J. Oper. Res. Soc.*, vol. 48, no. 3, pp. 334–334, 1997.

[41] T. Lipp and S. Boyd, "Variations and extension of the convex–concave procedure," *Optim. Eng.*, vol. 17, no. 2, pp. 263–287, 2016.

[42] D. J. MacKay, "Bayesian interpolation," *Neural Comput.*, vol. 4, no. 3, pp. 415–447, 1992.

[43] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, p. 19, 2016.

[44] *Kaggle*. Accessed: Oct. 2020. [Online]. Available: https://www.kaggle.com/netflix-inc/netflix-prize-data