Online Spatiotemporal Popularity Learning via Variational Bayes for Cooperative Caching

Sajad Mehrizi, Saikat Chatterjee, Symeon Chatzinotas and Björn Ottersten

***Abstract*—Herein, we focus on an end-to-end design of a proactive cooperative caching strategy for a multi-cell network. The design is challenging as it involves two interrelated problems: the ability to predict future content popularity and to meet network operation characteristics. To this end, we first formulate a cooperative content caching in order to optimize the aggregated network cost for delivering contents to users. An efficient proactive caching policy requires an accurate prediction of time-varying content popularity. Content popularity has temporal and spatial dependencies and therefore, we develop a probabilistic dynamical model for content popularity prediction by exploiting its spatiotemporal correlations. To achieve an accurate tracking and prediction of content popularity evolution, the proposed dynamical model is non-linear and incorporates non-Gaussian distributions. We use Variational Bayes (VB) approach for estimating the model parameters. The VB provides mathematical tractability. We then develop an online VB method that works with streaming data where content request arrives sequentially. Using extensive simulations study on a real-world dataset, we show that our online VB based dynamical model provides improved performance compared to conventional content caching policies.**

***Index Terms*—Content caching, Multi-cell network, Popularity prediction, Routing, Cache placement, Online variational Bayes.**

I. Introduction

Due to the emergence of communities with a massive number of users, the demands for content (e.g., video) are explosively growing [1]. This growth is challenging the capability of current network architectures to satisfy users' demands with an acceptable quality of experience. A promising approach to mitigate this challenge is to offload network traffic by caching *popular contents* at the network edge [2]. The motivation behind caching is that typically the majority of data traffic is caused by only a small number of popular contents, as indicated by the Zipf-law behavior [3]. Caching these highly popular contents at the edge which can be done during off peak hours bypasses the need for fetching these contents from the content provider through the backhaul links for every request. Therefore, this can significantly reduce the network congestion and improve the user quality of experience.

A central problem in caching systems is content placement i.e., selecting which and where to cache the contents. Several *reactive* placement algorithms have been proposed in [4]. For example, the least recently used (LRU) policy keeps a record of the recent requests for each content, and when there is not enough space it replaces contents with the longest idle time with newly requested ones. Likewise, the least frequently used (LFU) approach ejects the least frequently used contents. These traditional reactive policies and their variants are widely used in practice due to their simplicity. Nevertheless, they are not very efficient in capturing patterns in content requests and therefore may not perform satisfactorily. On the other hand, *proactive* caching empowered with predictive capabilities is particularly efficient in extracting and exploiting the hidden patterns in the requests and therefore can provide significant improvement on back-haul savings and user satisfaction [5]. The limitations of the network resources highlight the need for efficient proactive caching policies with the ability to predict the popularity of contents accurately.

In parallel, cooperative caching can provide great potentials to make efficient use of network resources for multicell-coordinated networks [6]. In this approach, geographically separated base stations (BSs) are allowed to share their caches with their neighbouring BSs in the network. As a consequence of this cooperation, the traffic load on the back-haul links for retrieving contents from the content server can be reduced dramatically.

Our work proposes an end-to-end design of a proactive cooperative caching scheme for a multi-cell network by investigating popularity prediction, joint request routing and content placement algorithms. Our contributions in this paper include:

- We develop a Bayesian dynamical model for content requests that can exploit spatiotemporal correlations to predict the popularity accurately. Bayesian methods have the capability to mitigate the overfitting problem efficiently when the data is scarce which is the case for content requests in edge-caching networks [7].
- We develop a fast and scalable VB technique to learn the Bayesian model parameters. The VB can perform both filtering and smoothing procedures for the dynamical model. We then suggest a window-based algorithm for online setting in which the request observations arrive sequentially over time. Depending on the desired accuracy and computational complexity, the length of the window can be tuned by the network operator which provides more flexibility.
- Moreover, a cooperative caching policy is formulated to minimize the network cost. The network cost includes the cost for transferring data among the BSs in the network, between the content server and the BSs, and updating the caches with new contents. The caching policy takes the predicted popularities as input and provides a cost-efficient delivery scheme for serving users.

S. Mehrizi, S. Chatzinotas and Björn Ottersten are with the Interdisciplinary Centre for Security, Reliability and Trust (SnT) University of Luxembourg, L-1855 Luxembourg (email: sajad.mehrizi, symeon.chatzinotas, bjorn.ottersten@uni.lu).

S. Chatterjee is with the Communication Theory Lab, KTH, Royal Institute of Technology, Stockholm, 10044, Sweden (e-mail: sach@kth.se).

This work was funded by the National Research Fund (FNR), Luxembourg under the projects "LISTEN" and "PROCAST" and also supported by the European Research Council (ERC) under the project "AGNOSTIC".

- Finally, in the simulation results, we evaluate the entire caching system (the popularity prediction and the caching policy design) on a real-world dataset and show the performance improvement of our scheme over some traditional benchmarks as well as the performance gap to the ideal case where the popluartieis can be perfectly predicted.

In the following, we first review the relevant works in Section II. Then, we define the system model in Section III-A. We describe the joint cache placement and request routing policy in Section III-B. In Section IV, we explain the probabilistic dynamical model and the VB approximation algorithm. In Section VI, we describe the window-based approach for online learning. In Section VIII, we present the simulation results and conclude the paper in Section IX.

II. Related works

During recent years, there has been great interests in proactive cooperative content caching [8]–[14]. In [8]–[10], the authors designed polices to minimize the aggregate network cost due to content transfers across the network. In [11], a caching policy was defined to minimize the content downloading delay by taking into account cache deployment costs in the network. The aforementioned works, however, ignored the capacity of communication links among the nodes. In wireless networks, the link capacity is constrained and it needs to considered in the design problem. In [12], a policy was designed to minimize the sum of downloading latency of all users subject to bandwidth capacity of BSs constraint. In this paper, we adapt the policy presented in [12] and formulate a content delivery scheme to minimize the aggregate network cost for delivering the contents to the users while guaranteeing the communication overhand among the BSs. We also incorporate the cache replacement cost into the optimization problem enabling dynamic migration of contents from the content server to the caches for our time-varying scenario.

While a number of results have been published on the caching policy design, there exist only a few methods for popularity prediction. In [15], assuming that the requests follow a Poisson distribution, the required training time was derived to obtain a desirable prediction accuracy. Subsequently, a transfer learning algorithm was proposed to improve the accuracy of prediction. In [16], popularity prediction was modeled as a matrix completion problem where the missing entries are estimated by a matrix factorization technique. A binary logistic classification method was introduced in [17] to classify user interests based on content features. In [18], content features were used to improve the prediction accuracy within the Bayesian framework. The authors of [19] introduced a Bayesian factor analysis model to model the correlations among contents. A main assumption of the aforementioned studies is that the popularity does not change over time or it changes very slowly. Nevertheless, in practice, content popularity is highly dynamic and might change even within a few hours e.g. for viral contents [20].

To tackle the time-varying nature of content popularity, several time-series analysis methods have been used in the literature to model the view count dynamics of videos. As one of the most popular models for time series, the auto-regressive moving-average (ARMA) model is utilized in [21]–[23] for video popularity prediction. However, ARMA modeling suffers from three important weaknesses. Firstly, this model is designed for continuous-valued data, which is not the case for content requests which are count-valued. Secondly, the assumed model is linear which is quite restrictive. Also, it ignores the correlation among contents. In reality, requests for some contents may be highly correlated and exhibit similar patterns, for example, some contents may have the same features or be interesting to the same user community. By appropriately modeling the count-valued nature of the requests and their correlation, a more prediction accuracy can be obtained and as a result a better caching performance. In our recent work in [24], we extended the factor analysis model in [19], which is designed for count-valued requests and also captures their correlation, to track the content popularity in a time-varying scenario. We showed that the model can perform better than the ARMA model by examining a real-world dataset. Moreover, in order to use spatiotemporal information, the authors in [6] suggested a probabilistic approach based on Markov chains. The presented model requires about $N \times M \times M$ parameters for M contents and N cells, which may not be practical to use when the number of contents or cells is large.

In this paper, we present a probabilistic dynamical model which can exploit the spatiotemporal correlation among the count-valued requests. The model is an extension of the factor analysis model in [24] by using a tensor factorization approach [25] to reduce the number of parameters for model parsimony while to capture rich spatiotemporal correlation structures to improve the accuracy of popularity prediction. We should mention that our Bayesian framework is different from neural network modeling approach, which has been attracted attention during recent years, e.g. [26] and [27], and has important advantages. Unlike most neural networks, Bayesian modeling can readily treat uncertainty in a systematic way for robust prediction. Moreover, neural networks usually require a lot of data to function. Applying them naively to edge-caching systems, which suffer from lack of sufficient data, may not provide a satisfactory performance.

Notation: In this paper, 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 trace operation is represented by $tr(.)$. 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.

III. Problem formulation

A. System model

We consider a cache-enabled multi-cell network as illustrated in Fig. 1. A summary of the symbols used herein is given in Table I. The network consists of N BSs and each serves a group of users. Moreover, BS $n \in \mathcal{B} \triangleq \{1, ..., N\}$ is

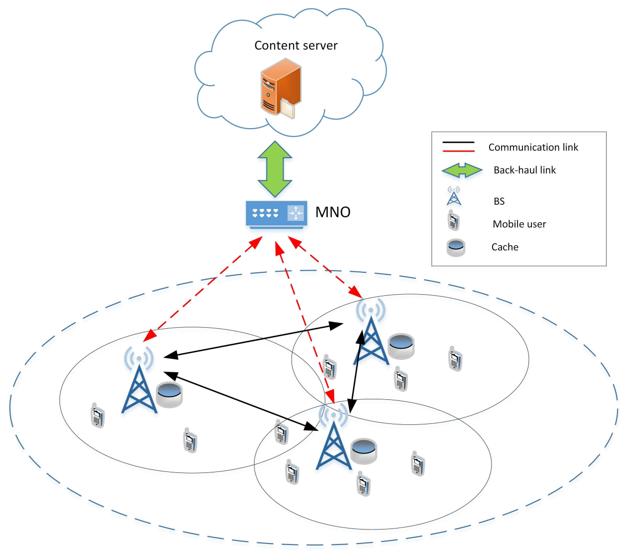


Fig. 1: The network model considered in this paper

Parameters	Description
M	Number of contents
N	Number of BSs
$c_{nn'}$	The cost of transferring data from BS n to BS n'
c_n	The cost of transferring data from content server to BS n
s_m	The size of content m
$\bar{r}_{mnt}$	Popularity of content m at cell n
$y_{nn'mt}$	The decision variable

TABLE I: Summary of Notations

equipped with a limited cache capacity of S_n units. All BSs are connected to the content server through back-haul links via mobile network operator (MNO) core and we assume all the processing tasks are performed in this core. Therefore, in our framework, no computational burden is caused on the end-user devices. Time is divided into different slots $t = 1, 2, ...,$ where each can be a day or a week, and we assume it is fixed and given. The users make random requests at each time slot from a library of M contents denoted by $\mathcal{C} \triangleq \{1, ..., M\}$, where content m has size s_m in data unit. If the requested content by a user is available in the local cache of its associated BS, it will be directly sent to the user. Otherwise, this BS first communicates with the neighboring BSs to check if they have the content. If the content is found in the local cache of a neighboring BS, it will be sent to the BS. If the content is not stored in any of the BSs's caches, it needs to be retrieved from the content server. However, fetching contents from the content server is expensive, e.g. due the incurred unit bandwidth cost for transporting data. Therefore, in order to have a cost-efficient delivery network, the requests need to be served by the BSs inside the network. This requires to properly distribute the contents over all the available BS's caches.

The problem which we consider herein is to minimize the total network cost incurred by delivering the contents to the users. The total network cost is affected by two problems: the content placement (i.e. which content and where to cache) and request routing (i.e. which BS should communicate for transferring the contents) policies. However, these two problems are interrelated and therefore they should be designed jointly. For example, the farther the content is cached from a user, the higher the cost needed for transferring the contents to the user is. This means that each content should be cached as close as possible to the requesting users so as to reduce the cost.

Here, we are concerned with a static scenario in which the request routing policy is designed based on the content popularity, the request rate, but not the instantaneous requests. Therefore, having a perfect popularity estimation is crucial side information for our algorithm in this section. This assumption is made only to reduce the computational complexity of the caching design though it may not be optimal. In other words, designing a routing for each instantaneous request causes a huge computational complexity overhead especially for a large network. This approach is also widely used in the literature, [28].

B. Content caching policy

The network cost consists of two costs. First, the routing cost, C_{Rt}, which can be expressed as

$$C_{Rt} = \sum_{n=1}^{N}\sum_{m=1}^{M}\Big(\sum_{n'\neq n}^{N} \bar{r}_{mnt} s_m y_{nn'mt} c_{nn'}\Big) + \bar{r}_{mnt} s_m \Big(1 - \sum_{n'=1}^{N} y_{nn'mt}\Big) c_n, \tag{1}$$

where $\bar{r}_{mnt}$ is the predicted popularity (request rate)[1] of content m at cell n during time slot t, $c_{nn'}$ is the cost for transferring data between BS n and BS n', and c_n is the cost for transferring data between BS n and and the content server. Moreover, $y_{nn'mt}$ is binary decision variable defined as

$$y_{nn'mt} = \begin{cases} 1 & \text{if BS } n' \text{ sends content } m \text{ to BS } n \\ 0 & \text{otherwise} \end{cases}. \tag{2}$$

Note that from the definition of $y_{nn'mt}$, it follows that y_{nnmt} is the cache placement decision variable and is one if content m is cached at BS n, otherwise is zero. The first term in (1) is the cost for serving the users by the in-network BSs. The second term is the cost for bringing contents from the content server through the back-haul link. Furthermore, we assume $c_{nn'} \ll c_n$, i.e. the cost for retrieving contents from the content server is much higher than from the BSs.

The second cost is associated with refreshing the caches with new contents during the cache placement which is given by

$$C_{Ut} = \sum_{n=1}^{N}\sum_{m=1}^{M} c_n s_m y_{nnmt}\left(1 - y_{nnm(t-1)}\right). \tag{3}$$

We assume that the new contents are downloaded only from the content server, and not from the BSs in the network. This assumption is for simplicity of the network operation. Otherwise, a complex synchronization algorithm is required for transferring the contents among the BSs. Now combining the two costs (1) and (3), the joint content placement and request

[1]Throughout the paper, we use the terms *content popularity* and *content request rate* interchangeably.

routing problem aiming to minimize the overall network cost over T time slots can be formulated as:

$$\min_{y_{nn'mt}} \quad \sum_{t=1}^{T} C_{Rt} + C_{Ut} \tag{4a}$$

$$s.t: \sum_{m=1}^{M} y_{nnmt} s_m \leq S_n \quad \forall n \in \mathcal{B} \tag{4b}$$

$$\sum_{n \neq n'}^{N} \sum_{m=1}^{M} \bar{r}_{mnt} y_{nn'mt} s_m \leq L_{n'} \quad \forall n' \in \mathcal{B} \tag{4c}$$

$$y_{nn'mt} \leq y_{n'n'mt} \quad \forall n, n' \in \mathcal{B}, m \in \mathcal{C} \tag{4d}$$

$$\sum_{n'=1}^{N} y_{nn'mt} \leq 1 \quad \forall n \in \mathcal{B}, m \in \mathcal{C} \tag{4e}$$

$$y_{nn'mt} \in \{0,1\} \quad \forall n, n' \in \mathcal{B}, m \in \mathcal{C}. \tag{4f}$$

In problem (4), constraint (4b) represents the cache capacity limitation in content size unit. Constraint (4c) ensures that the traffic load on BS n' due to serving other BSs is bounded by $L_{n'}$. In other words, this constraint guarantees that the communication overhead among the BSs does not exceed above $L_{n'}$. Constraint (4d) ensures that BS n' can send content m to BS n if it is stored in its local cache. Constraint (4e) ensures that BS n can only store content m in its cache or fetch it from only one BS. Constraint (4f) denotes the space restriction for $y_{nn'mt}$.

There are several issues which make the optimization problem in (4) non-trivial to solve even in offline scenario. Firstly, it is a binary integer nonlinear programming which is NP-hard to solve. Specially, because of the dimensions of problem which is specified by the number of contents and the number cells, it is computationally demanding. Secondly, The objective function is coupled over time horizon thorough the cache update cost in (3). This means that in order to obtain the optimal policy solution, problem (4) needs to be solved over all time slots jointly. In other words, a decision at present is influenced by the future content requests. For instance, by ignoring the future requests, it might be optimal to remove a content from a cache for the current time slot. However, 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. Solving problem (4), however, is not feasible for online settings since the content popularity is required over all time slots which is unknown in practice.

Our approach is to solve the one-shot optimization for each time slot. More specifically, given the decision variable at time slot $t = \tau$, $y_{mnn'(\tau)}$, we solve (4) with respect to $y_{mnn'(\tau+1)}$ for the next $(\tau+1)$th time slot. This gives a one-shot binary integer linear programming (BILP) problem. To solve this problem, we use the branch and bound which is a non-heuristic algorithm and can terminate with a certificate proving that the suboptimal point found is ϵ-suboptimal [29]. In the next section, we focus on designing an algorithm for content popularity prediction.

IV. Probabilistic demand model

In this section, we introduce a dynamical model for content requests in order to accurately predict the local popularities (content popularities in different cells). Our task is to predict future requests given all the requests up to the present time which are denoted as $\mathcal{D}_\tau = (\mathbf{D}_1, ..., \mathbf{D}_\tau)$ where $\mathbf{D}_t \in \mathbb{Z}^{M \times N}$. The element d_{mnt} is the number of requests for content m at cell n at time slot t which is obtained from request observations collected at the MNO core. For the prediction, we would like to compute a probability distribution over the next time slot i.e. $p(\mathbf{D}_{\tau+1}|\mathcal{D}_\tau)$. Once we compute this predictive distribution, we can derive our best guess for the future requests which can be used to optimize the caching policy.

Before diving into the details of the dynamical model, we remark some underlying causes that can impact the way users request the contents. This knowledge can help to construct an accurate model. In practice, there may exist two types of correlations among the contents: inter-cell and intra-cell. Users in the same cell may belong to the same social geography group (e.g university) and they can have similar interest towards contents. This indicates the demands for some contents exhibit the same patterns, i.e. the intra-cell correlation. Moreover, the requests for a content in different cells can be correlated. For example, users in different cells may belong to the same social media groups (e.g Instagram and Facebook) and they may affect their preferences. This request pattern across cells is referred as inter-cell correlation. This underlying prior information about the content requests motivates us to develop a probabilistic dynamical model to capture the two types of correlation to improve the prediction accuracy.

In order to model the correlations mentioned above, we design a probabilistic tensor factorization model. Tensor factorization is a fundamental dimensionality reduction technique in modern machine learning in order to capture higher order structural properties in a tensor data by modeling multilinear interactions among a group of low-dimensional latent factors [30]. Herein, we notice that the request observations can construct a three mode tensor with dimensionality of *content* × *cell* × *time*. Accordingly, similar to [25], we develop a probabilistic model based on CANDECOMP/PARAFAC(CP) factorization approach. In [25], a Gaussian model is used by the assumption that data is continuous-valued. However, the issue with the using of Gaussian model is that it may predict negative value for the request rate which is not meaningful in caching policy (4). Therefore, due to the count-valued nature of content requests, we cannot directly use the model in [25]. In a similar way as in [24], we adapt the CP factorization approach to Poisson distribution, which is common for modeling count-valued data, as

$$d_{mnt} \sim \text{Pois}(\sum_{k=1}^{K} a_{km} b_{kn} z_{kt}), \tag{5}$$

where $\mathbf{a}_m$, $\mathbf{b}_n$ and $\mathbf{z}_t$ are positive latent factor vectors with dimensions K and respectively correspond to content m, cell n and time t. The model in (5) assumes that the request for content m in cell n at time slot t, d_{mnt}, is generated by a

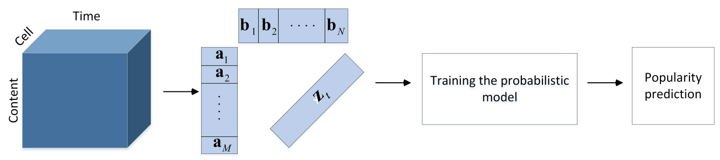


Fig. 2: A view of the tensor factorization algorithm for popularity prediction

Poisson distribution whose rate is $r_{mnt} = \sum_{k=1}^{K} a_{km} b_{kn} z_{kt}$. Note that (5) can also be rewritten as

$$d_{mnt} \sim \text{Pois}\left(\mathbf{a}_m^T \text{Diag}\left(\mathbf{z}_t\right) \mathbf{b}_n\right), \tag{6}$$

where $\text{Diag}\left(\mathbf{z}_t\right)$ denotes a $K \times K$ diagonal matrix with diagonal elements given by $z_{1t}, ... z_{Kt}$. We can interpret $\mathbf{z}_t$ as a basis random vector representing the requests in a hidden low-dimensional space and $\mathbf{a}_m, \mathbf{b}_n$ are weight parameters that project $\mathbf{z}_t$ to the observed space generating the request rates. Fig. 2 depicts a graphical view of the CP tensor factorization which models the correlations by introducing three factors; $\mathbf{a}_m$ factors of contents, $\mathbf{b}_n$ factors of cells, and $\mathbf{z}_t$ factors of time. We also note that in model (6), the correlation among the request rates for content m in cell n at time t and content m' at cell n' at time t' is given by

$$\text{Corr}\left(r_{mn,t}, r_{m'n't'}\right) = \left(\mathbf{a}_m \odot \mathbf{b}_n\right)^T E\left(\mathbf{z}_t \mathbf{z}_{t'}^T\right)\left(\mathbf{a}_{m'} \odot \mathbf{b}_{n'}\right), \tag{7}$$

where $\odot$ is the Hadamard product. Regarding the time factors $\mathbf{z}_t$, since they capture the time dependency of the requests, we assume that they follow a first order Markov process as

$$\mathbf{z}_t = e^{\mathbf{x}_t}, \quad \mathbf{x}_t = \mathbf{x}_{t-1} + \mathbf{e}_t, \tag{8}$$

starting from initial state $\mathbf{x}_0$. $\mathbf{e}_t$ is purely random and represents unpredictable changes in the time dimension between time t and $t-1$ and it is assumed that $\mathbf{e}_t \sim \mathcal{N}\left(\mathbf{0}, \mathbf{Q}\right)$. In (8), the time evolution in $\mathbf{z}_t$ is modeled by $\mathbf{x}_t$ which is a continuous random walk process (or more specifically a Gaussian process) and the exponential transformation ensures that $\mathbf{z}_t$ is positive.

Covariance matrix $\mathbf{Q}$ reveals important information about the time evolution of latent factor $\mathbf{x_t}$. When it is large, $\mathbf{x_t}$ can move longer distances which makes it more forgetful of past information. On the other hand, when it is small, it indicates that $\mathbf{x_t}$ changes very smoothly and does not forget past information. This means that $\mathbf{Q}$ plays an important role on the flexibility of the model. Therefore, we assume that the covariance matrix is unknown and changes over time. Allowing the covariance matrix to be dynamic can help to have a better prediction for unexpected patterns in the requests. For example, due to some social events users may be unexpectedly interested in particular types of contents for a period of time and therefore these contents abruptly become highly popular. Treating the covariance matrix as a constant may not let the model to be flexible enough to track these sudden changes. Also according to (7), the time-varying covariance matrix allows the correlation among the contents to change over time which is a realistic assumption in practice. We model the time varying covariance matrix $\mathbf{Q}_t$ by the generalized Wishart process [31] as

$$\mathbf{Q}_t^{-1} \sim \mathcal{GWP}\left(\nu, k(t, t'; \boldsymbol{\theta})\right), \tag{9}$$

where $\nu > 0$ is the number of degrees of freedom and $k(t, t'; \boldsymbol{\theta})$ is a Gaussian process kernel function with parameters $\boldsymbol{\theta}$. More formally, the marginal distribution of the generalized Wishart process at time t has a Wishart distribution

$$\mathbf{Q}_t^{-1} = \mathbf{U}_t \mathbf{U}_t^T \sim \mathcal{W}\left(k(t, t; \boldsymbol{\theta}), \nu\right), \tag{10}$$

where $\mathbf{U}_t = [\mathbf{u}_{1t}, ..., \mathbf{u}_{vt}]$ and $\mathbf{u}_{vt} \in \mathbb{R}^{K \times 1}$. Matrix $\mathbf{U}_t$ is a sample from a Gaussian process with zero mean and kernel function $k(t, t'; \boldsymbol{\theta})$ as:

$$\mathbf{u}_{it} \sim \mathcal{GP}\left(\mathbf{0}, k(t, t'; \boldsymbol{\theta})\right), \forall i = 1, ..., \nu. \tag{11}$$

For simplicity, we consider the kernel function $k(t, t'; \boldsymbol{\theta}) = \min\left(t, t'\right) \mathbf{Q}$. With this kernel, the Gaussian process in (11) simply becomes the first order Markov process as

$$\mathbf{u}_{it} = \mathbf{u}_{i(t-1)} + \boldsymbol{\zeta}_{it}, \forall i = 1, ..., \nu, \tag{12}$$

starting from initial state $\mathbf{U}_0$ and where $\boldsymbol{\zeta}_{it} \sim \mathcal{N}(\mathbf{0}, \hat{\mathbf{Q}})$. Here, we complete the basic structure of the dynamical model.

V. Bayesian Learning

Now, given the data requests, $\mathcal{D}_\tau$, the goal is to learn the parameters of the model. We invoke the Bayesian method. To this end, we need to specify the form of prior distributions for the parameters of the model. We assume the following priors:

$$\begin{aligned} a_{km} &\sim \mathcal{G}\left(\alpha, \beta\right), \forall k = 1, ..., K, m = 1, ..., M, \\ b_{kn} &\sim \mathcal{G}\left(\alpha', \beta'\right), \forall k = 1, ..., K, n = 1, ..., N, \\ \mathbf{x}_0 &\sim \mathcal{N}(\hat{\mu}_0, \hat{\boldsymbol{\Sigma}}_0), \quad \mathbf{U}_0 \sim \mathcal{MN}(\hat{\boldsymbol{\Upsilon}}_0, \hat{\boldsymbol{\Theta}}_{10}, \hat{\boldsymbol{\Theta}}_{20}), \end{aligned} \tag{13}$$

where $\mathcal{MN}\left(\hat{\boldsymbol{\Upsilon}}_0, \hat{\boldsymbol{\Theta}}_{10}, \hat{\boldsymbol{\Theta}}_{20}\right)$ denotes matrix-variate Gaussian distribution with mean matrix $\hat{\boldsymbol{\Upsilon}}_0$ and covariance matrices $\hat{\boldsymbol{\Theta}}_{10}$ (determining covariance among rows) and $\hat{\boldsymbol{\Theta}}_{20}$ a(determining covariance among columns). Using the Bayes rule, model learning is to estimate the posterior density which is proportional to:

$$p\left(\mathbf{h}|\mathcal{D}_\tau\right) \propto p\left(\mathbf{x}_0\right) p\left(\mathbf{U}_0\right) \prod_{n=1}^{N} \prod_{m=1}^{M} \prod_{k=1}^{K} p\left(a_{km}\right) p\left(b_{kn}\right)$$
$$\prod_{t=1}^{\tau} p\left(\mathbf{U}_t|\mathbf{U}_{t-1}\right) p\left(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{U}_t\right) \prod_{m=1}^{M} \prod_{n=1}^{M} p\left(d_{mnt}|\mathbf{a}_m, \mathbf{b}_n, \mathbf{x}_t\right), \tag{14}$$

where $\mathbf{h} = \left\{\left\{\mathbf{a}_m\right\}_{m=1}^M, \left\{\mathbf{b}_n\right\}_{n=1}^N, \left\{\mathbf{x}_t, \mathbf{U}_t\right\}_{t=0}^\tau\right\}$ is the set of all unknown quantities. Note that we assume that quantities $\left\{K, \hat{\mathbf{Q}}, v, \alpha.\beta, \alpha', \beta'\right\}$ and also the parameters of initial states densities are known hyper-parameters which they can be found by cross validation. The graphical representation of the Bayesian model is shown in Fig. 3. The unshaded and shaded circle nodes respectively indicate unknown variables and the

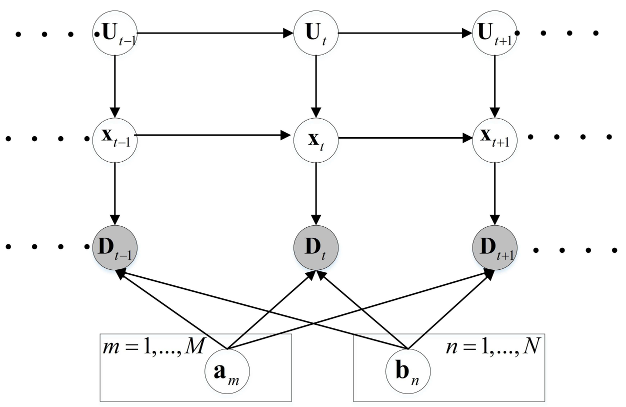
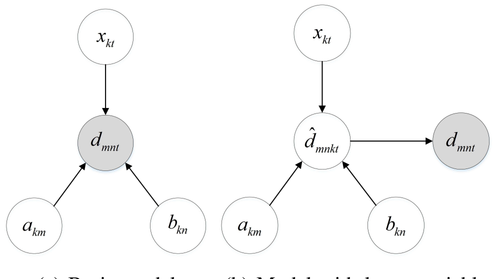


Fig. 3: The Bayesian graphical model

observation requests. The arrows illustrate the causal relations among the variables. The posterior density in (14) is not tractable to compute due the normalization constant which is a high-dimensional integral. One important class of Bayesian models satisfies the conditional conjugacy property meaning that the conditional posterior of a variable and the prior should belong to the same distribution family [32]. A big advantage of such models is that the posterior can be approximated with closed-form expressions. We apply the variable augmentation technique, as used in [33], to introduce a partial conjugacy. The approach is to reformulate (5) by introducing additional latent variables $\hat{d}_{mnkt}$ as:

$$\hat{d}_{mnkt} \sim \text{Pois}\left(a_{km} b_{kn} e^{x_{kt}}\right). \tag{15}$$

From the property of Poisson [34], we see that $d_{mnt} = \sum_{k=1}^{K} \hat{d}_{mnkt}$. With this reformulation, the Bayesian model fulfills the conditional conjugacy for variables $\mathbf{a}_m$, $\mathbf{b}_n$, and $\hat{d}_{mnkt}$ which we can leverage to design a low complexity inference approximation. A part of the Bayesian model with and without the augmented latent variables $\hat{d}_{mnkt}$ is shown in Fig.4.



(a) Basic model (b) Model with latent variable
Fig. 4: Part of the Bayesian model

A. Variational Bayes

In order to approximate the posterior, we employ the VB framework which is briefly explained in this subsection. Variational inference performs the inference task as the problem of minimizing the Kullback-Leibler (KL) divergence between an approximate distribution and the true posterior [32].

More formally, true posterior $p(\mathbf{h}|\mathcal{D}_\tau)$ is approximated by a family of distributions $q(\mathbf{h};\boldsymbol{\lambda})$ with parameters $\boldsymbol{\lambda}$ that minimizes the KL divergence as

$$\min_{q(\mathbf{h};\boldsymbol{\lambda})} \quad \text{KL}\left(q\left(\mathbf{h};\boldsymbol{\lambda}\right) || \, p\left(\mathbf{h}|\mathcal{D}_\tau\right)\right), \quad s.t: \int q\left(\mathbf{h};\boldsymbol{\lambda}\right) d\mathbf{h} = 1, \tag{16}$$

where $\text{KL}\left(q\left(\mathbf{h};\boldsymbol{\lambda}\right) || \, p\left(\mathbf{h}|\mathcal{D}_\tau\right)\right) \triangleq E_{q(\mathbf{h};\boldsymbol{\lambda})}\left\{\log\frac{q(\mathbf{h};\boldsymbol{\lambda})}{p(\mathbf{h}|\mathcal{D}_\tau)}\right\}$. It can be seen that minimizing the KL divergence between $q(\mathbf{h};\boldsymbol{\lambda})$ and $p\left(\mathbf{h}|\mathcal{D}_\tau\right)$ is equivalent to maximizing a lower bound for the marginal log-likelihood of the observations which is also called the evidence lower bound (ELBO). The ELBO can be expressed as:

$$ELBO\left(\boldsymbol{\lambda}\right) = E_{q(\mathbf{h};\boldsymbol{\lambda})}\left(\log\left(\tilde{p}\left(\mathbf{h}|\mathcal{D}_\tau\right)\right)\right) + H\left(q\left(\mathbf{h};\boldsymbol{\lambda}\right)\right), \tag{17}$$

where $H\left(q\left(\mathbf{h};\boldsymbol{\lambda}\right)\right)$ is the entropy of $q(\mathbf{h};\boldsymbol{\lambda})$ and $\tilde{p}\left(\mathbf{h}|\mathcal{D}_\tau\right)$ is the un-normalized version of the true posterior density.

One of the most popular forms of VB is the mean field approximation. In this scheme, the assumption is that the variational distribution is factorized as:

$$q\left(\mathbf{h};\boldsymbol{\lambda}\right) = \prod_i q\left(\mathbf{h}_i;\boldsymbol{\lambda}_i\right), \tag{18}$$

where $\mathbf{h}_i$ is part of $\mathbf{h}$ with $\bigcup_i \mathbf{h}_i = \mathbf{h}$ and $\bigcap_i \mathbf{h}_i = \emptyset$. By applying the mean-field approximation, the objective function in (17) can be optimized via a coordinate ascent like method in which the optimization sub-problem with respect to the $\boldsymbol{\lambda}_i$ variational parameter is given by

$$ELBO\left(\boldsymbol{\lambda}_i\right) = E_{q(\mathbf{h})}\left(\log\tilde{p}\left(\mathbf{h}_i|\mathbf{h}_{\sim i},\mathcal{D}_\tau\right)\right) + H\left(q\left(\mathbf{h}_i;\boldsymbol{\lambda}_i\right)\right), \tag{19}$$

where $\tilde{p}\left(\mathbf{h}_i|\mathbf{h}_{\sim i},\mathcal{D}_\tau\right)$ is the un-normalized conditional posterior of $\mathbf{h_i}$. It turns out that when some variables $\mathbf{h}_i$ satisfy the conjugacy property, choosing $q(\mathbf{h_i})$ in the form of prior leads the maximization problem in (19) to have a closed-form solution. Otherwise, numerical methods may be needed to find its optimal value.

B. Posterior approximation

To approximate (14), we consider the following factorized form for the variational density:

$$\begin{aligned} q\left(.\right) =& q\left(\mathbf{x}_0\right) q\left(\mathbf{U}_0\right) \{\prod_{t=1}^{\tau} q\left(\mathbf{x}_t\right) q\left(\mathbf{U}_t\right) \prod_{m=1}^{M}\prod_{n=1}^{N} q(\hat{\mathbf{d}}_{mnt})\} \\ & \{\prod_{k=1}^{K}\prod_{m=1}^{M} q\left(a_{km}\right)\}\{\prod_{k=1}^{K}\prod_{n=1}^{N} q\left(b_{kn}\right)\}, \end{aligned} \tag{20}$$

where

$$\begin{aligned} & q\left(\mathbf{x}_t\right) = \mathcal{N}\left(\mathbf{x}_t;\boldsymbol{\mu}_t,\boldsymbol{\Sigma}_t\right), \; q(\hat{\mathbf{d}}_{mnt}) = \text{Multi}(\hat{\mathbf{d}}_{mnt}; d_{mnt}, \mathbf{p}_{mnt}), \\ & q\left(a_{km}\right) = \mathcal{G}\left(a_{km};\alpha_{km},\beta_{km}\right), \; q\left(b_{kn}\right) = \mathcal{G}\left(b_{kn};\alpha'_{kn},\beta'_{kn}\right), \\ & q\left(\mathbf{U}_t\right) = \mathcal{MN}\left(\mathbf{U}_t;\Upsilon_t,\boldsymbol{\Theta}_{1t},\boldsymbol{\Theta}_{2t}\right), \end{aligned}$$

and for simplicity, we assume $\boldsymbol{\Theta}_{1t}$, $\boldsymbol{\Theta}_{2t}$ and $\boldsymbol{\Sigma}_t$ are diagonal matrices. These forms of dentists are for simplicity and efficient inference. Specifically, as we will see in the following, the gamma and multinomial variational densities lead to derive closed-form expressions during the coordinate decent updates. Consequently, the ELBO for the probabilistic dynamical model can be written as:

$$ELBO = \sum_{n=1}^{N}\sum_{m=1}^{M}\sum_{t=1}^{\tau}\sum_{k=1}^{K} E_q[\log Pois(\hat{d}_{mnkt}|a_{km}, b_{kn}, x_{kt})]+ \sum_{m=1}^{M}\sum_{k=1}^{K} E_q\left[\log \mathcal{G}\left(a_{km}|\alpha,\beta\right)\right] + \sum_{n=1}^{N}\sum_{k=1}^{K} E_q\left[\log \mathcal{G}\left(b_{kn}|\alpha',\beta'\right)\right] \sum_{t=1}^{\tau} E_q\left[\log \mathcal{N}\left(\mathbf{x}_t|\mathbf{x}_{t-1},\mathbf{U}_t\right)\right] + \sum_{t=1}^{\tau} E_q\left[\log \mathcal{N}\left(\mathbf{U}_t|\mathbf{U}_{t-1}\right)\right]+ H_q\left(q(.)\right), \quad (21)$$

where $d_{mnt} = \sum_{k=1}^{K} \hat{d}_{mnkt}$. Using the coordinate ascend like method, the updates for each dimension of the variational parameters are given as follows:

- Update w.r.t. p_{mnkt}: It can be derived in closed-form as

$$p_{mnkt} = \frac{c_{mnkt}}{\sum_{k=1}^{K} c_{mnkt}}, \quad (22)$$

where $c_{mnkt} = e^{\psi(\alpha_{km})-\ln(\beta_{km})+\psi\left(\alpha'_{kn}\right)-\ln\left(\beta'_{kn}\right)+\mu_{kt}}$.

Proof. See Appendix A. □

- Update w.r.t. β_{km} and α_{km}: They can be derived in closed-forms as

$$\alpha_{km} = \alpha + \sum_{n=1}^{N}\sum_{t=1}^{\tau} d_{mnt}p_{mnkt}, \quad (23)$$

$$\beta_{km} = \beta + \sum_{n=1}^{N}\sum_{t=1}^{\tau} \frac{\alpha'_{kn}}{\beta'_{kn}} e^{\mu_{kt}+\frac{1}{2}[\mathbf{\Sigma}_t]_{kk}}. \quad (24)$$

Proof. See Appendix B. □

- Update w.r.t. β'_{kn} and α'_{kn}: They can be derived in closed-forms as

$$\alpha'_{kn} = \alpha' + \sum_{m=1}^{M}\sum_{t=1}^{\tau} d_{mnt}p_{mnkt}, \quad (25)$$

$$\beta'_{kn} = \beta' + \sum_{m=1}^{M}\sum_{t=1}^{\tau} \frac{\alpha_{km}}{\beta_{km}} e^{\mu_{kt}+\frac{1}{2}[\mathbf{\Sigma}_t]_{kk}}. \quad (26)$$

Proof. The proof is similar to (23) and (24). □

- Update w.r.t. $\boldsymbol{\mu}_t$ and $\mathbf{\Sigma}_t$: The ELBO function has the following form:

$$ELBO\left(\boldsymbol{\mu}_t, \mathbf{\Sigma}_t\right) = \sum_{n=1}^{N}\sum_{m=1}^{M}\sum_{k=1}^{K} \{d_{mnt}p_{mnkt}\mu_{kt} - \frac{\alpha_{km}\beta'_{kn}}{\beta_{km}\beta'_{kn}} e^{\mu_{kt}+\frac{1}{2}[\mathbf{\Sigma}_t]_{kk}}\} - \frac{1}{2} tr\left(\left(\boldsymbol{\mu}_t\boldsymbol{\mu}_t^T + \mathbf{\Sigma}_t\right)\left(\mathbf{\Xi}_t + \mathbf{\Xi}_{t+1}\right)\right) + (\boldsymbol{\mu}_{t-1}^T\mathbf{\Xi}_t + \boldsymbol{\mu}_{t+1}^T\mathbf{\Xi}_{t+1})\boldsymbol{\mu}_t + \log\left|\mathbf{\Sigma}_t\right|, \quad (27)$$

where $\mathbf{\Xi}_t = E\left\{\mathbf{U}_t\mathbf{U}_t^T\right\} = \mathbf{\Theta}_{1t} tr\left(\mathbf{\Theta}_{2t}\right) + \Upsilon_t\Upsilon_t^T$. The optimization for $\boldsymbol{\mu}_t$ and $\mathbf{\Sigma}_t$ doesn't have a closed-form solution. However, since the objective term in (27) is concave, we can use projected Newton-type methods to find its maximum value with fast convergence rate.

- Optimize w.r.t $\mathbf{\Upsilon}_t$ and $\mathbf{\Theta}_t = (\mathbf{\Theta}_{1t}, \mathbf{\Theta}_{2t})$: Similar to (27), there is no closed-form update for these parameters. The functional dependency of the ELBO function on $\mathbf{\Upsilon}_t$ and $\mathbf{\Theta}_t$ has the following form:

$$ELBO\left(\mathbf{\Upsilon}_t, \mathbf{\Theta}_t\right) = \frac{1}{2} E_{q(\mathbf{U}_t)} \log\left|\mathbf{U}_t\mathbf{U}_t^T\right| - h\left(\mathbf{\Upsilon}_t, \mathbf{\Theta}_t\right), \quad (28)$$

where

$$h\left(\mathbf{\Upsilon}_t, \mathbf{\Theta}_t\right) = \frac{1}{2} tr\left(\mathbf{\Upsilon}_t^T\left(\mathbf{C}_t + 2\hat{\mathbf{Q}}^{-1}\right)\mathbf{\Upsilon}_t\right) + tr(\mathbf{\Theta}_{2t} tr(\mathbf{\Theta}_{1t}(\mathbf{C}_t + 2\hat{\mathbf{Q}}^{-1})^T) - tr\left(\mathbf{\Upsilon}_t^T\hat{\mathbf{Q}}^{-1}\left(\mathbf{\Upsilon}_{t-1}+\mathbf{\Upsilon}_{t+1}\right)\right) - \frac{v}{2}\log\left|\mathbf{\Theta}_{1t}\right| - \frac{K}{2}\log\left|\mathbf{\Theta}_{2t}\right|,$$

and $\mathbf{C}_t = E\left\{\left(\mathbf{x}_t - \mathbf{x}_{t-1}\right)\left(\mathbf{x}_t - \mathbf{x}_{t-1}\right)^T\right\}$. Optimizing the objective term in (28) is not trivial. More specifically, function (28) is not convex and also computing the expectation in the first term leads to a complicated expression which is difficult to optimize. To optimize the objective function term, we follow a stochastic approximation approach. However, there is an issue with the expectation term. To be specific, random variable $\mathbf{U}_t$ is a function of the optimization parameters which is not in a standard form for using stochastic approximation techniques. To address this, we note that random variable $\mathbf{U}_t \sim \mathcal{MN}\left(\mathbf{\Upsilon}_t, \mathbf{\Theta}_{1t}, \mathbf{\Theta}_{2t}\right)$ can be equivalently re-parameterized as

$$\mathbf{U}_t = \mathbf{\Theta}_{1t}^{1/2}\mathbf{\Psi}\mathbf{\Theta}_{2t}^{1/2} + \mathbf{\Upsilon}_t, \quad (29)$$

where the elements of $\mathbf{\Psi}$ are normal random variables with zero means and unit variances. By this transformation, we can rewrite (28) as

$$ELBO\left(\mathbf{\Upsilon}_t, \mathbf{\Theta}_t\right) = \frac{1}{2} E_{p(\mathbf{\Psi})} \log|\mathbf{\Theta}_{1t}^{1/2}\mathbf{\Psi}\mathbf{\Theta}_{2t}\mathbf{\Psi}^T\mathbf{\Theta}_{1t}^{1/2} + \mathbf{\Theta}_{1t}^{1/2}\mathbf{\Psi}\mathbf{\Theta}_{2t}^{1/2}\mathbf{\Upsilon}_t^T + \mathbf{\Upsilon}_t\mathbf{\Theta}_{2t}^{1/2}\mathbf{\Psi}^T\mathbf{\Theta}_{1t}^{1/2} + \mathbf{\Upsilon}_t\mathbf{\Upsilon}_t^T| - h\left(\mathbf{\Upsilon}_t, \mathbf{\Theta}_t\right), \quad (30)$$

which is in standard form for using stochastic approximation. To obtain the maximum value of (30), we use a stochastic successive convex approximation method [35]. Please see Appendix C for the implementation details.

The overall description of the VB is illustrated in Alg. 1. Starting by initializing the variational parameters, we iteratively update them until the ELBO converges. To check the convergence of the ELBO, we can examine the normalized incremental change of the variables in each iteration and once it is less than a threshold e.g. 10^{-3} the algorithm stops.

Remark: The coordinate ascent VB algorithm guarantees to monotonically increase the ELBO when each sub-problem is convex and attains a unique maximum [36]. The condition is satisfied for all sub-problems except for (30). In other words, due to the non-convexity of sub-problem (30) and using stochastic approximation for computing its gradient, the ELBO is not guaranteed to be increased at each update. Nevertheless, the convergence of the developed coordinate ascent VB can

Algorithm 1: The coordinate ascent VB algorithm

```
1 Initialize $\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t, \boldsymbol{\Upsilon}_t, \boldsymbol{\Theta}_t, \alpha_{km}, \beta_{km}, {\alpha'}_{kn}, {\beta'}_{kn}$,
    $\forall m = 1, ...M, n = 1, ..., N, k = 1, ..., K, t = 0, ..., \tau$;
2 repeat
3     Update $\alpha_{km}, \beta_{km}, {\alpha'}_{kn}, {\beta'}_{kn}$ using (25)-(28)
        $\forall m = 1, ...M, n = 1, ..., N, k = 1, ..., K$;
4     Update $\mathbf{p}_{mnt}$ using (22),
        $\forall m = 1, ...M, n = 1, ..., N, t = 1, ..., \tau$;
5     Update $\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t$ using (27), $\forall t = 0, ..., \tau$;
6     Update $\boldsymbol{\Upsilon}_t, \boldsymbol{\Theta}_t, \forall t = 0, ...T$ using stochastic
        successive approximation method;
7 until ELBO convergence;
```

be established based on the expectation violation of a first-order optimality condition. Specifically, because of the convex approximation procedure (see Appendix C), the update of $(\boldsymbol{\Upsilon}_t, \boldsymbol{\Theta}_t)$ in sub-problem (30) is unique. Moreover, by choosing proper values for stepsizes $\gamma^{(i)}$ and $\varepsilon^{(i)}$, in (47) and (48), such that to diminish (with $\gamma^{(i)}$ decreasing faster than $\varepsilon^{(i)}$) to zero while not too fast, the convergence analyses in [35] and [37] can be used to show that the gradient of the ELBO converges to zero on average.

C. Prediction

Once the posterior is fit, we use the model to predict the content request rate over the next time slots. We use the mean of the predictive distribution as our best guess for the request rate which can be computed by:

$$\bar{r}_{mn(\tau+1)} = \sum_{k=1}^{K} \frac{\alpha_{km}{\alpha'}_{kn}}{\beta_{km}{\beta'}_{kn}} E_{q(\mathbf{x}_\tau)p(\mathbf{x}_{\tau+1}|\mathbf{x}_\tau)}\left[e^{x_{k(\tau+1)}}\right]. \quad (31)$$

The expectation doesn't have a closed-form expression. We use the Jensen's inequality and approximate it as:

$$E_{q(x_{k\tau})p(x_{k(\tau+1)}|x_{k\tau})}\left[e^{x_{k(\tau+1)}}\right] \approx E_{q(x_{k\tau})}\left[e^{x_{k\tau}}\right] = e^{\mu_{k\tau} + \frac{1}{2}[\boldsymbol{\Sigma}_\tau]_{kk}}.$$

The predicted request rate in (31) can be used to solve the cache optimization problem in (4).

VI. Online Learning

For an online caching update, we need an online prediction algorithm. This cannot be achieved by the VB method in the previous section, which uses the whole request observation sequence and hence is computationally intensive.

To tackle this issue, we suggest a window-based VB algorithm which also offers a trade-off between computational complexity and prediction accuracy. More specifically, instead of processing all observations we use a sliding window with length W. For an online setting, we take the posterior over the variables to be the prior for subsequent inferences. Here, we should remark that there are two types of variables in the model. The global variables that are shared across all the observations i.e. $\mathbf{a}_m, \mathbf{b}_n$ and the local variables that are specific only for each sample observation i.e. $\mathbf{x}_t, \mathbf{U}_t, \hat{\mathbf{d}}_{mnt}$. Lets denote the window observation indexed by τ, each of size W, which are given to the model sequentially. From (25)-(28), it can be seen that we can easily write the updates role for the global parameters recursively. For example for $\mathbf{a}_m$, we have:

$$\alpha_{km} = \alpha_{km}^{(\tau-1)} + \sum_{n=1}^{N}\sum_{w=1}^{W} d_{mn((\tau-1)+w)} p_{mnk((\tau-1)+w)}, \quad (32)$$

$$\beta_{km} = \beta_{km}^{(\tau-1)} + \sum_{n=1}^{N}\sum_{w=1}^{W} \frac{{\alpha'}_{kn}}{{\beta'}_{kn}} e^{\boldsymbol{\mu}_{k((\tau-1)+w)} + \frac{1}{2}\boldsymbol{\Sigma}_{k((\tau-1)+w)}}, \quad (33)$$

where $\alpha_{km}^{(\tau-1)}$ and $\beta_{km}^{(\tau-1)}$ are the inferred variational parameters in the previous window indexed by $\tau - 1$. This recursion provides an incremental update of the parameters. The local parameters are computed as before. Only for $\mathbf{x}_t, \mathbf{U}_t$ their initial states are the inferred last states in the previous window. The VB algorithm is then run until convergence. Once new observations arrive, we slide the window by a time slot and the same procedure is repeated.

There is one important point we should note. When $W > 1$, the observations in subsequent windows overlap and using the recursive updates in (32) causes to over-count the local variables during the global variables update. In order to mitigate this, we can simply subtract overlapped local variables from the update rules in the global variables.

Note that the highest prediction accuracy is attained when the window's length is set as the total number of historical observations. This is, however, may not be possible due the limitations on computational resources at the MNO unit. Therefore, in practical scenarios, the correct window's length will be set based on the available computational resources.

VII. Computational complexity

We compute the per-iteration computational complexity of approximating the posterior distribution in (14) using the developed online VB algorithm. The individual contributions of updating variational parameters to the overall per-iteration time complexity during the τth observation window are as follows. $i)$ Updating the parameters of each $\hat{\mathbf{d}}_{mn(\tau-1+w)}$, $\forall m = 1, .., M, n = 1..., N, w = 1, ..., W$, takes $\mathcal{O}(K)$ time if $d_{mnt} \neq 0$, otherwise 0 time. This can seen from (15) that when $d_{mnt} = 0$, $\hat{d}_{mnkt}$, $\forall k = 1, .., K$, becomes a deterministic variable with zero value. Therefore, only non-zero requests need to be considered. The overall time-complexity is $\mathcal{O}(WKT_{nnz})$ where T_{nnz} is the number of non-zero requests during the most densely requested time slot. $ii)$ The overall time complexity of updating the parameters of a_{km} and b_{kn}, $\forall m = 1, .., M, n = 1..., N, k = 1, ..., K$, is $\mathcal{O}(NK + MK)$. $iii)$ The time complexity of updating each $\mathbf{x}_{(\tau-1+w)}$ and $\mathbf{U}_{(\tau-1+w)}$, $\forall w = 1, ..., W$ are $\mathcal{O}(K^3)$ and $\mathcal{O}((\nu K)^3)$, respectively, which are largely due to matrix inversion operations. Overall, the computational complexity of the online VB is $\mathcal{O}(WKT_{nnz} + NK + MK + WK^3 + W(\nu K)^3)$.

VIII. Simulation Results

In this section, we numerically illustrate the performance of the proposed dynamical model for popularity prediction and cooperative caching policy. For ease of presentation, the proposed dynamical model is abbreviated by DPG which stands for dynamical Poisson-Gaussian. Throughout all the simulations, we set $M = 100$, and $N = 10$. The values of hyperparameters are set as below, which we found the VB works well in our scenarios.

$$\alpha = \beta = \alpha' = \beta' = 0.01, \hat{\mathbf{Q}} = 0.25\mathbf{I}_K, \nu = K+1, \hat{\mathbf{\Sigma}}_0 = \mathbf{I}_K,$$
$$\hat{\boldsymbol{\mu}}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_K), \hat{\Upsilon}_0 \sim \mathcal{MN}(\mathbf{0}, 5\mathbf{I}_K, 5\mathbf{I}_v), \hat{\mathbf{\Theta}}_{10} = \mathbf{I}_K, \hat{\mathbf{\Theta}}_{20} = \mathbf{I}_v$$

Moreover, the platform for the simulations is equipped with Intel(R) Core(TM) i7-6820HQ CPU running at a speed of 2.70GHZ.

W	1	2	4	8	16
Prediction error	1.88	1.42	1.36	1.18	1.10

TABLE II: Prediction error for different values of W

A. Synthetic data

We first evaluate the performance of the online VB learning method on synthetic data. For this scenario, we generate synthetic data based on the proposed probabilistic model. We predict the content popularity $\bar{r}_{mnt}$ for $T = 100$ time slots which in total contains $M \times N \times T = 10^5$ predictions. In this scenario, we set $K = 5$. We use the following metric to measure the prediction error:

$$\text{Err} = \frac{1}{MNT} \sum_{m,n,t} |r_{mnt} - \bar{r}_{mnt}| \tag{34}$$

Tab. II shows the error of the predictions for different values of window length W. We can see that as W increases the prediction accuracy improves. Specifically, increasing W from 2 to 16 reduces the error from 1.88 to 1.10. The reason is that with larger values of W more observations are used to estimate the model's parameters and as a result a more accurate prediction is obtained. In Fig. 5, we show an example of the generated requests, popularities and predicted popularities of a content in four different cells (namely a,b,c and d) for $W = 4$. It can be seen that the predicted popularties are very close to the true values. Additionally, we can observe that the probabilistic model is flexible enough to capture different types of popularity trends across different cells. The top subfigures in Fig. 5 show the content in cells a and b with the same popularity trend. On the other hand, the bottom subfigures show a different trend for the content in cells b and c.

Now, we empirically show the convergence of the Online VB approximation. Fig. 6 shows the normalized incremental change of the variational density parameters versus the computational time. It can be seen that as W increases the amount of time for the convergence of the VB increases. This is expected since by increasing W, the number of latent variables, i.e. the parameters of $\hat{d}_{mnt}, \mathbf{x}_t$ and $\mathbf{U}_t$, during the inference increases which increases the computational complexity of the algorithm. Specifically, we observe that the VB converges fairly fast. For example, for $W = 16$ and at convergence threshold 10^{-3}, it converges around 250 seconds.

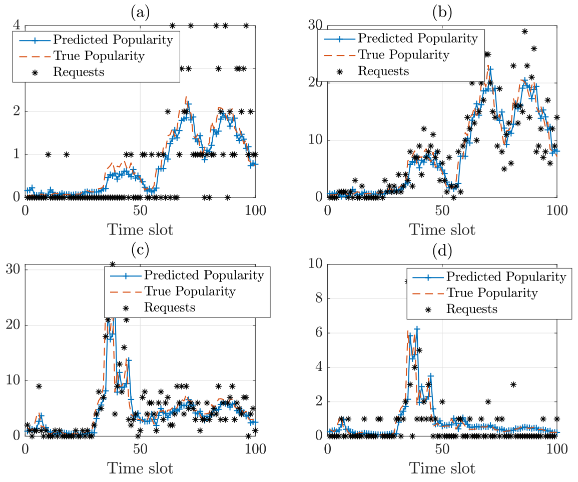


Fig. 5: Request rate trajectories of a content in different cells

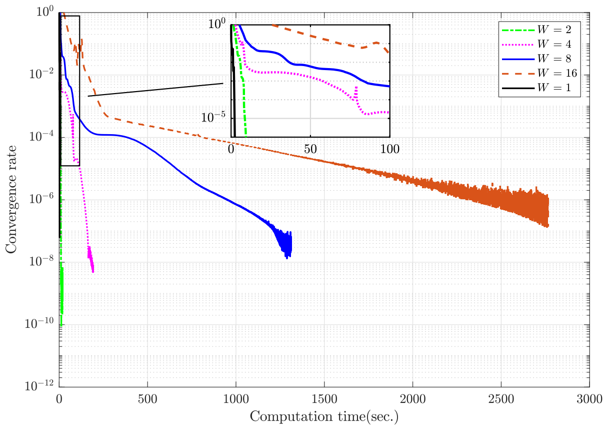


Fig. 6: The convergence behavior of the VB

B. Real-world data

Next, we use a real-world dataset in our numerical experiments. More specifically, we use the MovieLens 20M example [38]. The dataset contains 20 million ratings applied to 27,000 movies by 138,000 users. We choose ratings on a bimonthly basis where we select the most popular $M = 100$ movies and the most active 1000 users. The users are randomly distributed across the cells, each has 100 users. We remove the time slots with no ratings which in total $T = 106$ time slots remains. Similar to [19] and [39], a movie's rate is considered as one request for this movie. Fig. 7 shows the total number of content requests versus time slots. We can see that at the beginning, the number of requests is small. Then, it increases with an abrupt change. For some time slots, it fluctuates with a large variance at almost a fixed level and then it decreases. In our simulations, we keep the first 16 time slots for using as initial training set for the models and the predictions are performed on the remaining 90 time slots.

We next examine the prediction accuracy of the model on this dataset. As benchmarks, we compare with:

- Independent dynamical Poisson Gaussian (IDPG) model: this is a special case of the probabilistic model in (5) which only captures the intra-cell correlations. We fit (5) to the request observations in cell n independent of the other cells by setting its cell factor weight to $\mathbf{b}_n = \mathbf{1}$.

- 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 $t = 1, ..., T$ and $m = 1, ..., M$, 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 [21]. For its implementation, we used the econometrics MATLAB toolbox. Note that due to the Gaussian noise assumption, ARMA model may predict negatives values for the requests. To provide meaningful predictions, we round the negative values to a small positive value e.g. 10^{-5}.
- The most recently observed requests (MROR) approach: the requests for next time slot are assumed to be the same as in the recent time slot.
- The most frequently observed requests (MFOR) approach: the requests for the next time slot are the average of the whole historical observations.

Since the true underlying content popularity is unknown for this real-world dataset, we are unable to use (34) as prediction performance metric. Instead, we use

$$\text{Err} = \frac{1}{MNT} \sum_{m,n,t} |d_{mnt} - \bar{r}_{mnt}| \tag{35}$$

which measures the discrepancy between the predicted popularities and the instantaneous requests.

Tab. III shows the prediction error in (35) for different methods. We also show the prediction error over time slots in Fig. 8 In this scenario, we set $K = 5$ and $W = 4$. It can be seen that the proposed dynamical models, DPG and IDPG, provide a better accuracy with respect to the other approaches. Moreover, DPG model has a better performance in comparison to DPG model. This indicates that modeling the spatial correlations can enhance the prediction accuracy. We can also see that ARMA model has the worst performance in comparison to MFOR and MROR. Moreover, the performance of MFOR approach gets worse as time passes. This is because MFOR can not capture the temporal locality of popularity and as time passes its estimation becomes outdated. On the other hand, MROR approach can capture a short term temporal locality. However, it is not efficient since it only uses the requests within the recent time slot and completely ignores the historical requests.

Prediction approach	DPG	IDPG	MROR	MFOR	ARMA
Prediction error	0.84	0.92	0.97	0.98	1.06

TABLE III: Prediction error of different models

Now, we show the accrued network cost using different prediction approaches. The parameters of the caching policy are set $c_{ns} = 50$, $c_{nn'} = 5$, $L_n = L$, and $S_n = S$. Moreover, since the dataset does not have the size of movies, for simplicity, we assume all the movies have equal sizes of one unit. The results for the network cost are obtained by taking average over time, the number of contents and the number of cells. Moreover, to solve the combinatorial problem

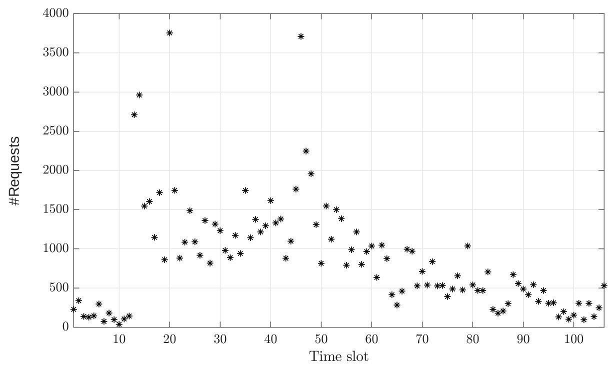


Fig. 7: The total number of content requests in used data set versus time slots

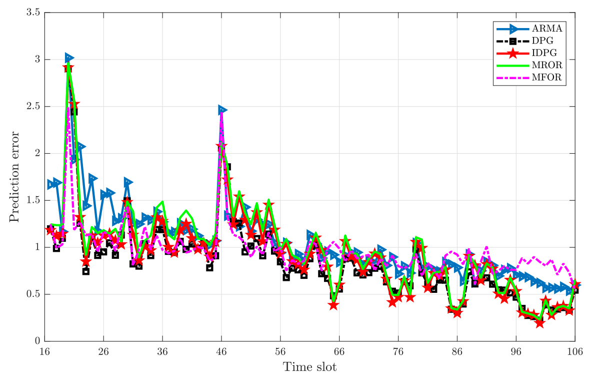


Fig. 8: Prediction error of different models over time

in (4), we use the IBM ILOG CPLEX optimization studio version 12.9.0.0 with its default setup [40].

Fig. 9 shows the average network cost versus K. For this scenario, we set $L = 25$ and $S = 25$. In the figure, we also show an oracle approach which knows in advance the content requests. It can be seen that the proposed dynamical models, DPG and IDPG, outperform the other approaches and are close to the oracle. This confirms the results on prediction accuracy in Tab. III and Fig. 8. Additionally, we can see that DPG model achieves a better caching performance in comparison to IDPG model which indicates that modeling the spatial correlations is beneficial for improving the caching performance. Furthermore, it can be observed that as K increases, the network cost decreases for both DPG and IDPG models. The reason is that as K increases the expressiveness power of the models increases and they can explain the observations more accurately. Furthermore, among MROR, MFOR and ARMA, the former performs the best while the latter performs the worst. We also show the performance of the proposed model, DPG, for different values of W for $K = 5$ in Tab. IV. It can be seen that as W increases the network cost decreases which is due to a better prediction for larger values of W.

W	1	2	4	8	16
Average network cost	21.73	21.45	20.90	20.81	20.79

TABLE IV: Average network cost for different values of W

Finally, we investigate the performance of the proactive cooperative caching policy. To compare with, we consider three traditional reactive policies: LRU, LFU [4] and LFRU [41].

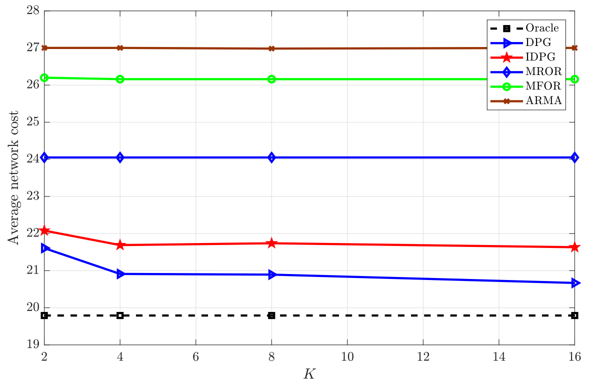


Fig. 9: Average network cost versus latent factors dimensions, K, for different models

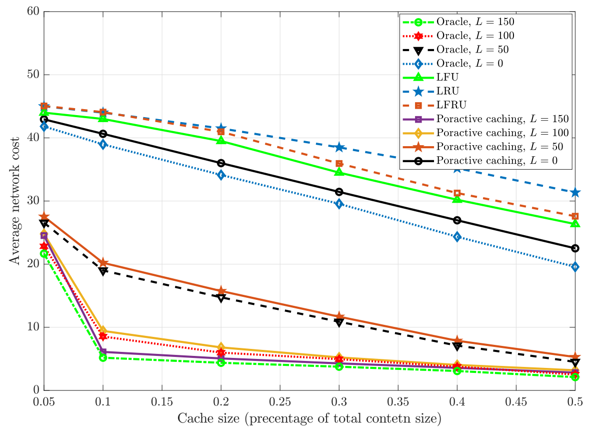


Fig. 10: Average network cost versus cache size for different caching policies

Fig. 10 shows the network cost versus the cache capacity. From the figure, it can be seen that the performance of caching policies improve as the cache size increases. This is excepted since more contents can be stored in the caches avoiding the need of fetching them from the content served or the neighboring BSs. Moreover, we observe that for the cooperative policy the network cost decreases as L increases. This is expected because as L increases, the BSs are allowed to support more data traffic for the routing and therefore to cooperate more for sending contents to the their neighbors. Consequently, most of the requests will be served inside the network bypassing the need of fetching contents from the far way sever which is too costly. Furthermore, it can be seen that our proactive cooperative caching policy performs very close to the coracle and substantially outperforms reactive LRU, LFU and LFRU polices. Specifically, the performance gap between reactive polices (LRU, LFU and LFRU) and proactive cooperative policy increases significantly as L increases.

IX. Conclusions

In this work, we introduced a cooperative caching policy for cache-enabled multi-cell networks. We designed a joint routing and cache placement strategy to minimize the network cost to fulfill users requests. Then, we proposed a probabilistic dynamical model for the content requests which can track complex real-world datasets. The model is powerful enough to capture the correlations among the requests of contents both within a cell and across cells. The model learning is performed in the Bayesian way which is very appreciated to tackle overfitting. Due to intractability of the posterior, we developed an approximation algorithm based on the Variational Bayes which can be solved efficiently. Finally, a window-based online learning was developed which can even reduce the computational complexity of the Variational Bayes inference but with less accuracy. The online learning is also very important when requests arrive sequentially and online cache replacement is required. Experimental results on a real-world dataset confirmed that our proposed popularity prediction method performs well in comparison to benchmark algorithms.

Appendix

A. Proof of (22)

First, we note that the conditional posterior of latent variable $\hat{d}_{mnkt}$ has a multinomial distribution as [34]:

$$\hat{d}_{mnkt} \sim \text{Multi}(d_{mnt}, (a_{km}b_{kn}e^{x_{kt}})/\sum_{k=1}^{K} a_{km}b_{kn}e^{x_{kt}}). \tag{36}$$

We replace conditional distribution (36) into (19) and optimize the ELBO with respect to $\mathbf{p}_{mnt}$. The functional dependency of the ELBO on $\mathbf{p}_{mnt}$ as

$$ELBO\,(\mathbf{p}_{mnt}) = E_q\left[\log \text{Multi}(\hat{\mathbf{d}}_{mnt}|d_{mnt}, a_{km}, b_{kn}, x_{kt})\right] + H(q(\hat{\mathbf{d}}_{mnt})) \tag{37}$$

The objective function in (37) can be simplified as

$$ELBO\,(\mathbf{p}_{mnt}) = E_{q(\hat{\mathbf{d}}_{mnt})}\{\log d_{mnt}! - \sum_{k=1}^{K} \log \hat{d}_{mnkt}! + \sum_{k=1}^{K} \hat{d}_{mnkt} \log \hat{c}_{mnkt}\} + H(q(\hat{\mathbf{d}}_{mnt}))$$

where $\hat{c}_{mnkt} = e^{E_{\sim q(\hat{\mathbf{a}}_{mnt})}[\log(a_{km}b_{kn}e^{x_{kt}}) - \log(\sum_{k=1}^{K} a_{km}b_{kn}e^{x_{kt}})]}$. After some mathematical operations and ignoring constant terms, the optimization sub-problem can be written as

$$\min_{p_{mnkt}} d_{mnt} \sum_{k=1}^{K} p_{mnkt} \log\left(\frac{p_{mnkt}}{\hat{c}_{mnkt}}\right), \quad \text{s.t.} \quad \sum_{k=1}^{K} p_{mnkt} = 1$$

where the constraint ensures that parameters p_{mnkt} are normalized. Using the method of Lagrangian multipliers, we aim to solve

$$\frac{\partial}{\partial p_{mnkt}}[d_{mnt} \sum_{k=1}^{K} p_{mnkt} \log(\frac{p_{mnkt}}{\hat{c}_{mnkt}}) + \lambda(\sum_{k=1}^{K} p_{mnkt} - 1)] = 0 \tag{38}$$

where λ is the Lagrange multiplier for the normalization constraint. Setting

$$p_{mnkt} \propto_k \hat{c}_{mnkt} \tag{39}$$

achieves the desired outcome. Here, $\propto_k$ indicates the proportionality across k. We can compute a closed-form expression for p_{mnkt}. To this end, we compute the expectation terms in the exponent of $\hat{c}_{mnkt}$. We first note that there is no need to compute the second term which is difficult to obtain. Since due to the normalization, it is eliminated, automatically. The expectation in the first term is computed as

$$E\left\{\log\left(a_{km}b_{kn}e^{x_{kt}}\right)\right\} = \psi\left(\alpha_{km}\right) - \ln\left(\beta_{km}\right) + \psi\left({\alpha'}_{kn}\right) - \ln\left({\beta'}_{kn}\right) + \mu_{kt} \tag{40}$$

where $\psi\left(.\right)$ is the diagmma function. By replacing (40) in (39), we obtain (22).

B. *Proof of* (23) *and* (24)

The functional dependency of the ELBO in α_{km} and β_{km} is given by:

$$ELBO\left(\alpha_{km}, \beta_{km}\right) = \sum_{n=1}^{N}\sum_{m=1}^{M}\sum_{t=1}^{\tau}\sum_{k=1}^{K} E_q\left[\log Pois\left(\hat{d}_{mnkt}|a_{km}, b_{kn}, x_{kt}\right)\right] + \sum_{m=1}^{M}\sum_{k=1}^{K} E_q\left[\log \mathcal{G}\left(a_{km}|\alpha, \beta\right)\right] + H\left(q\left(a_{km}\right)\right) \tag{41}$$

The ELBO can be simplified as

$$ELBO\left(\alpha_{km}, \beta_{km}\right) = \left(\sum_{t=1}^{\tau}\sum_{n=1}^{N} E[\hat{d}_{mnkt}] + \alpha - 1\right)\left(\psi\left(\alpha_{km}\right) - \log\beta_{km}\right) - \frac{\alpha_{km}}{\beta_{km}}\left(\sum_{t=1}^{\tau}\sum_{n=1}^{N} E[b_{kn}e^{x_{kt}}] + \beta\right) + \alpha_{km} - \log\beta_{km} + \log\Gamma\left(\alpha_{km}\right) + \left(1-\alpha_{km}\right)\psi\left(\alpha_{km}\right) \tag{42}$$

By equating the gradient of ELBO with respect to β_{km} to zero, we obtain

$$\frac{\partial ELBO(\alpha_{km}, \beta_{km})}{\partial \beta_{km}} = 0 \rightarrow \beta_{km} = \frac{\alpha_{km}\left(\sum_{t=1}^{\tau}\sum_{n=1}^{N} E[b_{kn}e^{x_{kt}}] + \beta\right)}{\sum_{t=1}^{\tau}\sum_{n=1}^{N} E[\hat{d}_{mnkt}] + \alpha} \tag{43}$$

Similarly, we compute the gradient of the ELBO with respect to α_{km} as

$$\frac{\partial ELBO\left(\alpha_{km}, \beta_{km}\right)}{\partial \alpha_{km}} = \left(\sum_{t=1}^{\tau}\sum_{n=1}^{N} E[\hat{d}_{mnkt}] + \alpha\right)\psi'\left(\alpha_{km}\right) - \frac{1}{\beta_{km}}\left(\sum_{t=1}^{\tau}\sum_{n=1}^{N} E[b_{kn}e^{x_{kt}}] + \beta\right) + 1 - \alpha_{km}\psi'\left(\alpha_{km}\right) \tag{44}$$

By replacing (43) into (44) and setting the gradient equal to zero we obtain:

$$\alpha_{km} = \sum_{n=1}^{N}\sum_{t=1}^{\tau} E[\hat{d}_{mnkt}] + \alpha \tag{45}$$

By replacing (45) into (43), we get

$$\beta_{km} = \sum_{n=1}^{N}\sum_{t=1}^{\tau} E[b_{kn}e^{x_{kt}}] + \beta \tag{46}$$

C. *Stochastic successive convex approximation method*

To solve the optimization problem in (30), we use the stochastic successive convex approximation proposed in [35]. The algorithm provides a framework for optimizing the excepted value of a (possibly non-convex) cost function parametrized by a random variable when the expectation cannot be computed exactly. More specifically, the original problem is decomposed into different subproblems, using a stochastic convex approximation, which can be solved in parallel. For the convergence of the algorithm, each subproblem should have a unique and well-defined solution.

To find the maxmimum value of (30), we equivalently find the minimum value of $f\left(\Upsilon_t, \Theta_t\right) = -ELBO\left(\Upsilon_t, \Theta_t\right)$. We decompose $f\left(\Upsilon_t, \Theta_t\right)$ into two stochastic convex approximation subproblems, one for $\boldsymbol{\Upsilon}_t$ and one for $\boldsymbol{\Theta}_t$. Hereafter, we drop time index t for notational simplicity.

First, we derive an approximation for $\boldsymbol{\Upsilon}$. At iteration $i+1$, for subproblem $\boldsymbol{\Upsilon}$, we solve the following convex approximation function of $f\left(\Upsilon, \Theta\right)$.

$$\hat{f}(\boldsymbol{\Upsilon}; \boldsymbol{\Theta}^{(i)}, \boldsymbol{\Upsilon}^{(i)}, \boldsymbol{\Psi}^{(i)}) = \varepsilon^{(i)}(g(\boldsymbol{\Upsilon}; \boldsymbol{\Theta}^{(i)}, \boldsymbol{\Upsilon}^{(i)}, \boldsymbol{\Psi}^{(i)}) + h(\boldsymbol{\Upsilon}; \boldsymbol{\Theta}^{(i)})) + (1-\varepsilon^{(i)})tr((\boldsymbol{\Upsilon} - \boldsymbol{\Upsilon}^{(i)})^T \mathbf{G}^{(i)}) \tag{47}$$

where $g\left(\boldsymbol{\Upsilon}; \boldsymbol{\Theta}^{(i)}, \boldsymbol{\Upsilon}^{(i)}\right)$ is a linear approximation of the non-convex part of the objective function with respect to $\boldsymbol{\Upsilon}$ at $(\boldsymbol{\Theta}^{(i)}, \boldsymbol{\Upsilon}^{(i)}, \boldsymbol{\Psi}^{(i)})$ and is given by

$$g(\boldsymbol{\Upsilon}; \boldsymbol{\Theta}^{(i)}, \boldsymbol{\Upsilon}^{(i)}, \boldsymbol{\Psi}^{(i)}) = -tr\left(\boldsymbol{\Upsilon}^T \tilde{\mathbf{F}}^{(i)}\right)$$

where

$$\tilde{\mathbf{F}}^{(i)} = {\mathbf{F}^{(i)}}^{-1}({\boldsymbol{\Theta}_1^{(i)}}^{1/2} \boldsymbol{\Psi}^{(i)} {\boldsymbol{\Theta}_2^{(i)}}^{1/2} + \boldsymbol{\Upsilon}^{(i)})$$

and

$$\mathbf{F}^{(i)} = {\boldsymbol{\Theta}_1^{(i)}}^{1/2} \boldsymbol{\Psi}^{(i)} \boldsymbol{\Theta}_2^{(i)} {\boldsymbol{\Psi}^{(i)}}^T {\boldsymbol{\Theta}_1^{(i)}}^{1/2} + {\boldsymbol{\Theta}_1^{(i)}}^{1/2} \boldsymbol{\Psi}^{(i)} {\boldsymbol{\Theta}_2^{(i)}}^{1/2} {\boldsymbol{\Upsilon}^{(i)}}^T + \boldsymbol{\Upsilon}^{(i)} {\boldsymbol{\Theta}_1^{(i)}}^{1/2} {\boldsymbol{\Psi}^{(i)}}^T {\boldsymbol{\Theta}_2^{(i)}}^{1/2} + \boldsymbol{\Upsilon}^{(i)} {\boldsymbol{\Upsilon}^{(i)}}^T$$

Function $h\left(\Upsilon; \boldsymbol{\Theta}^{(i)}\right)$ preserves the convex structure of the objective function. Moreover, matrix $\mathbf{G}^{(i)}$ is an incremental estimate of the gradient of objective function and has the form of

$$\mathbf{G}^{(i)} = (1-\varepsilon^{(i)})\mathbf{G}^{(i-1)} + \varepsilon^{(i)}(-\tilde{\mathbf{F}}^{(i)} + \nabla_{\Upsilon} h(\Upsilon^{(i)}, \boldsymbol{\Theta}^{(i)}))$$

where parameter $\varepsilon^{(i)} \in (0, 1]$ is a sequence of properly chosen step-sizes. The update at iteration $i+1$ takes the form of

$$\boldsymbol{\Upsilon}^{(i+1)} = \boldsymbol{\Upsilon}^{(i)} + \gamma^{(i+1)}\left(\bar{\boldsymbol{\Upsilon}} - \boldsymbol{\Upsilon}^{(i)}\right) \tag{48}$$

where $\gamma^{(i)} \in (0, 1]$ and $\bar{\boldsymbol{\Upsilon}}$ is the optimal solution of sub-problem

$$\bar{\boldsymbol{\Upsilon}} = \arg\min_{\Upsilon} \hat{f}\left(\Upsilon; \boldsymbol{\Theta}^{(i)}, \Upsilon^{(i)}, \boldsymbol{\Psi}^{(i)}\right) \tag{49}$$

Problem (49) is an unconstrained quadratic problem whose solution can be obtained by equating its gradient equal to zero and it is given by

$$\overline{\boldsymbol{\Upsilon}}^T = (\varepsilon^{(i)} {\tilde{\mathbf{F}}^{(i)}}^T + \varepsilon^{(i)}(\Upsilon_{t-1}^T + \Upsilon_{t+1}^T)\hat{\mathbf{Q}}^{-1} - (1-\varepsilon^{(i)}){\mathbf{G}^{(i)}}^T)(\varepsilon^{(i)}(\mathbf{C} + 2\hat{\mathbf{Q}}^{-1}))^{-1} \tag{50}$$

With a similar approach, we obtain the solutions of the optimization sub-problems for $\boldsymbol{\Theta}_1$ and $\boldsymbol{\Theta}_2$

$$\boldsymbol{\Theta}_j^{(i+1)} = \boldsymbol{\Theta}_j^{(i)} + \gamma^{(i+1)} \left([\bar{\boldsymbol{\Theta}}_j]^+ - \boldsymbol{\Theta}_j^{(i)} \right), \quad j = 1, 2 \tag{51}$$

where

$$\begin{aligned}\bar{\boldsymbol{\Theta}}_1 =& \frac{\nu}{2} [\frac{(1-\varepsilon^{(i)})}{\varepsilon^{(i)}} \mathbf{G}_{\Theta_1}^{(i)} - \tilde{\mathbf{F}}_{\Theta_1}^{(i)} + \\ & tr\left(\boldsymbol{\Theta}_2^{(i)}\right)\left(\mathbf{C} + 2\mathbf{Q}^{-1}\right) \odot \mathbf{I}]^{-1}\end{aligned}$$

$$\begin{aligned}\bar{\boldsymbol{\Theta}}_2 =& \frac{K}{2} [\frac{(1-\varepsilon^{(i)})}{\varepsilon^{(i)}} \mathbf{G}_{\Theta_2}^{(i)} - \tilde{\mathbf{F}}_{\Theta_2}^{(i)} + \\ & tr\left(\boldsymbol{\Theta}_1^{(i)}\left(\mathbf{C} + 2\mathbf{Q}^{-1}\right)\right) \odot \mathbf{I}]^{-1}\end{aligned}$$

$$\begin{aligned}\tilde{\mathbf{F}}_{\Theta_1}^{(i)} =& \frac{1}{2} (\boldsymbol{\Psi}^{(i)} \boldsymbol{\Theta}_2^{(i)} \boldsymbol{\Psi}^{(i)^T} \boldsymbol{\Theta}_1^{1/2^{(i)}} \mathbf{F}^{(i)^{-1}} \boldsymbol{\Theta}_1^{-1/2^{(i)}} + \\ & \mathbf{F}^{(i)^{-1}} \Upsilon \boldsymbol{\Theta}_2^{1/2^{(i)}} \boldsymbol{\Psi}^{(i)^T} \boldsymbol{\Theta}_1^{-1/2^{(i)}}) \odot \mathbf{I}\end{aligned}$$

$$\begin{aligned}\tilde{\mathbf{F}}_{\Theta_2}^{(i)} =& \frac{1}{2} (\boldsymbol{\Psi}^{(i)^T} \boldsymbol{\Theta}_1^{1/2^{(i)}} \mathbf{F}^{(i)^{-1}} \boldsymbol{\Theta}_1^{1/2^{(i)}} \boldsymbol{\Psi}^{(i)} + \\ & \Upsilon^{T^{(i)}} \mathbf{F}^{(i)^{-1}} \boldsymbol{\Theta}_1^{1/2^{(i)}} \boldsymbol{\Psi}^{(i)} \boldsymbol{\Theta}_2^{-1/2^{(i)}}) \odot \mathbf{I}\end{aligned}$$

$$\begin{aligned}\mathbf{G}_{\Theta_j}^{(i)} =& \left(1 - \varepsilon^{(i)}\right) \mathbf{G}_{\Theta_j}^{(i-1)} + \\ & \varepsilon^{(i)} (-\tilde{\mathbf{F}}_{\Theta_j}^{(i)} + \nabla h_{\boldsymbol{\Theta}_j}(\Upsilon^{(i)}, \boldsymbol{\Theta}^{(i)})), \forall j = 1, .., 2\end{aligned}$$

Analogous with stochastic gradient methods, proper diminishing step-sizes for $\varepsilon^{(i)}$ and $\gamma^{(i)}$ are required to ensure convergence. An example of such step-sizes are:

$$\gamma^{(i)} = \frac{1}{i^{0.65}}, \quad \varepsilon^{(i)} = \frac{1}{i^{0.55}} \tag{52}$$

which we found are appropriate.

References

[1] Cisco, "Global mobile data traffic forecast update, 2016–2021," *White Paper*, 2017.

[2] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8402–8413, Dec. 2013.

[3] L. Breslau, Pei Cao, Li Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: evidence and implications," in *Proc. IEEE INFOCOM*, March 1999, pp. 126–134.

[4] S. Podlipnig and L. Böszörmenyi, "A survey of web cache replacement strategies," *ACM Comput. Surv.*, vol. 35, no. 4, pp. 374–398, 2003.

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

[6] L. Yang, J. Cao, G. Liang, and X. Han, "Cost aware service placement and load dispatching in mobile cloud systems," *IEEE Trans. Comput*, vol. 65, no. 5, pp. 1440–1452, May 2016.

[7] 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, 2016.

[8] J. Li, T. K. Phan, W. K. Chai, D. Tuncer, G. Pavlou, D. Griffin, and M. Rio, "Dr-cache: Distributed resilient caching with latency guarantees," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 441–449.

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

[10] ——, "Jointly optimal routing and caching for arbitrary network topologies," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1258–1275, 2018.

[11] S. M. S. Tanzil, W. Hoiles, and V. Krishnamurthy, "Adaptive scheme for caching youtube content in a cellular network: Machine learning approach," *IEEE Access*, vol. 5, pp. 5870–5881, May 2017.

[12] 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, 2016.

[13] L. Pu, L. Jiao, X. Chen, L. Wang, Q. Xie, and J. Xu, "Online resource allocation, content placement and request routing for cost-efficient edge caching in cloud radio access networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 8, pp. 1751–1767, Aug 2018.

[14] A. Gharaibeh, A. Khreishah, B. Ji, and M. Ayyash, "A provably efficient online collaborative caching algorithm for multicell-coordinated systems," *IEEE Trans. Mobile Comput.*, vol. 15, no. 8, pp. 1863–1876, Aug 2016.

[15] B. Bharath, K. Nagananda, and H. V. Poor, "A learning-based approach to caching in heterogenous small cell networks," *IEEE Trans. Commun.*, vol. 64, no. 4, pp. 1674–1686, 2016.

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

[17] 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.

[18] 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. 92 341–92 354, 2019.

[19] ——, "Content popularity estimation in edge-caching networks from Bayesian inference perspective," in *Poc. IEEE Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2019, pp. 1–6.

[20] G. S. Paschos, G. Iosifidis, M. Tao, D. Towsley, and G. Caire, "The role of caching in future communication systems and networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1111–1125, June 2018.

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

[22] N. B. Hassine, R. Milocco, and P. Minet, "ARMA based popularity prediction for caching in content delivery networks," in *Proc. IEEE Wireless Days*, March 2017, pp. 113–120.

[23] K. N. Doan, T. Van Nguyen, T. Q. S. Quek, and H. Shin, "Content-aware proactive caching for backhaul offloading in cellular network," *IEEE Trans. Wireless Commun.*, vol. 17, no. 5, pp. 3128–3140, May 2018.

[24] S. Mehrizi, A. Tsakmalis, S. ShahbazPanahi, S. Chatzinotas, and B. Ottersten, "Popularity tracking for proactive content caching with dynamic factor analysis," in *2019 IEEE/CIC International Conference on Communications in China (ICCC)*, Aug 2019, pp. 875–880.

[25] 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.

[26] V. Fedchenko, G. Neglia, and B. Ribeiro, "Feedforward neural networks for caching: Enough or too much?" *ACM SIGMETRICS Performance Evaluation Review*, vol. 46, no. 3, pp. 139–142, 2019.

[27] H. Pang, J. Liu, X. Fan, and L. Sun, "Toward smart and cooperative edge caching for 5g networks: A deep learning based approach," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*. IEEE, 2018, pp. 1–6.

[28] 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.

[29] D. P. Bertsekas, *Nonlinear programming*. Athena scientific Belmont, 1999.

[30] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[31] A. G. Wilson and Z. Ghahramani, "Generalised Wishart processes," in *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, ser. UAI'11. Arlington, Virginia, United States: AUAI Press, 2011, pp. 736–744.

[32] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[33] P. Gopalan, J. M. Hofman, and D. M. Blei, "Scalable recommendation with hierarchical poisson factorization," in *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, ser. UAI'15. Arlington, Virginia, United States: AUAI Press, 2015, pp. 326–335.

[34] J. F. C. Kingman, “Poisson processes,” *Encyclopedia of biostatistics*, vol. 6, 2005.

[35] Y. Yang, G. Scutari, D. P. Palomar, and M. Pesavento, “A parallel decomposition method for nonconvex stochastic multi-agent optimization problems,” *IEEE Trans. Signal Process.*, vol. 64, no. 11, pp. 2949–2964, 2016.

[36] M. Opper and D. Saad, *Advanced mean field methods: Theory and practice*. MIT press, 2001.

[37] Y. Xu and W. Yin, “Block stochastic gradient iteration for convex and nonconvex optimization,” *SIAM Journal on Optimization*, vol. 25, no. 3, pp. 1686–1716, 2015.

[38] F. M. Harper and J. A. Konstan, “The movielens datasets: History and context,” *ACM Transactions on Interactive Intelligent Systems*, vol. 5, no. 4, p. 19, 2016.

[39] S. Müller, O. Atan, M. van der Schaar, and A. Klein, “Context-aware proactive content caching with service differentiation in wireless networks,” *IEEE Trans. Wireless Commun.*, vol. 16, no. 2, pp. 1024–1036, 2017.

[40] www.cplex.com.

[41] M. Bilal and S.-G. Kang, “A cache management scheme for efficient content eviction and replication in cache networks,” *IEEE Access*, vol. 5, pp. 1692–1701, 2017.