IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 2, 2020, accepted July 20, 2020, date of publication August 5, 2020, date of current version August 27, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3014379

Active Content Popularity Learning and Caching Optimization With Hit Ratio Guarantees

SRIKANTH BOMMARAVENI, (Student Member, IEEE), THANG X. VU, (Member, IEEE), SYMEON CHATZINOTAS, (Senior Member, IEEE), AND BJÖRN OTTERSTEN, (Fellow, IEEE)
Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, 1511 Luxembourg City, Luxembourg

Corresponding author: Srikanth Bommaraveni (srikanth.bommaraveni@uni.lu)

This work was supported in part by the National Research Fund, Luxembourg, through the Project AGNOSTIC under Grant 742648, in part by the FNR Bilateral Project LARGOS under Grant 12173206, and in part by the FNR CORE ProCAST under Grant C17/IS/11691338.

ABSTRACT Edge caching is an effective solution to reduce delivery latency and network congestion by bringing contents close to end-users. A deep understanding of content popularity and the principles underlying the content request sequence are required to effectively utilize the cache. Most existing works design caching policies based on global content requests with very limited consideration of individual content requests which reflect personal preferences. To enable the optimal caching strategy, in this article, we propose an Active learning (AL) approach to learn the content popularities and design an accurate content request prediction model. We model the content requests from user terminals as a demand matrix and then employ AL-based query-by-committee (QBC) matrix completion to predict future missing requests. The main principle of QBC is to query the most informative missing entries of the demand matrix. Based on the prediction provided by the QBC, we propose an adaptive optimization caching framework to learn popularities as fast as possible while guaranteeing an operational cache hit ratio requirement. The proposed framework is model-free, thus does not require any statistical knowledge about the underlying traffic demands. We consider both the fixed and time-varying nature of content popularities. The effectiveness of the proposed learning caching policies over the existing methods is demonstrated in terms of root mean square error, cache hit ratio, and cache size on a simulated dataset.

INDEX TERMS Edge caching, active learning, matrix completion, content popularity.

I. INTRODUCTION

The proliferation of numerous cellular devices and their demand for data-hungry services result in exponential growth in mobile data traffic. According to Cisco annual report [2], the data traffic is predicted to increase by sevenfold between 2017 and 2022. This explosive growth in data traffic challenges the capabilities of current network architectures. As the data traffic is primarily dominated by repeated requests for a few popular contents [3], a promising solution to overcome this challenge is to offload network traffic and store repeatedly requested contents at the network edge [4]. Moreover, appropriate caching at the edge results in overall reduced the latency and downloading time. This motivates the need for developing new algorithms to effectively cache the contents at the edge to avoid the duplicate transmissions.

Typically content caching is divided into two phases: a content placement and a content delivery. The content placement phase depends on the historic requests of the users and it is limited by the cache storage size. On the other hand, the content delivery depends on serving the request of a user upon the arrival of the request at the edge. Some of the works that are based on joint design of content placement and content delivery are [4]–[10]. All these works focus on improving the utilization of the cache assuming the users' requests to the contents are known in advance. Therefore, the performance of these methods depends on the accuracy of users' content requests (UCR). Although, in practice, the UCR is unknown and needs to be estimated [11]. Several works focused on prediction of UCR based

The associate editor coordinating the review of this manuscript and approving it for publication was Ghufran Ahmed.

on machine learning methods [11]–[16]. However, in a real-world application, the popularity of the contents is non-stationary, and predicting the UCR is therefore a challenging task. In the edge network, only a small set of requests are observed in a given window of time. However, to improve the long term average performance of cache, the number of requests to the not-yet requested contents (missing requests) needs to be known. As mentioned earlier, it is difficult to collect the UCR, hence, machine learning methods have been adopted to accurately model and estimate UCR probability for missing requests in recommendation systems [11], [17], [18]. For non-stationary UCR, it is important to design an iterative recommendation model to predict future content popularity. In this aspect, active learning (AL) is an indispensable tool to collect the data and steers the learning process towards achieving the accuracy goal by actively selecting the most useful data points to query. It is also referred to as query learning or optimal experimental design [19]. The main idea behind AL in edge caching is that the mobile edge computing (MEC) server interacts with the end users by posing queries or recommendations. This way the server obtains knowledge about the end users' preferences from time to time and predicts their future content requests. From the observed and estimated content requests the popularity of the contents is determined, which then is used as an effective measure for making caching decisions. We see the problem of estimating the content requests as an active learning matrix completion problem where the entries of the matrix are UCR.

A. LITERATURE REVIEW

During recent years, several papers investigated caching at edge network under various objectives such as minimizing latency, network congestions, maximizing user's quality of experience (QoE), or energy efficiency. The following works assume that content popularity is known as a priory. In [5], the authors optimize the cache placement phase to minimize the average latency and total throughput during the delivery phase. Caching the most popular files in distributed small cell access points to minimize the total average delay of all the users is proposed in [4]. In [6], the authors aim to maximize the delivery phase's rate by considering both placement and delivery phases jointly. A distributed caching policy is proposed in [7] for device-to-device systems based on social awareness and the matching theory. The authors of [17], proposed a cache placement algorithm to reduce network congestion in the back-haul link and maximize the QoE. To minimize the download latency over multiple distributed caches, the authors in [20] proposed an optimization problem as a maximization of a submodular function subject to matroid constraints for content placement. The performance of caching under uncoded and coded strategies is analyzed in [21].

Content popularity prediction-based edge-caching has been considered in [11]–[16], [22], [23]. The authors in [11] propose a proactive caching algorithm by leveraging social networks and D2D communications, by assuming the content requests follow Zipf distribution. To learn the local and global space-time popularities of contents a reinforcement learning framework is proposed in [13]. They also assume that the popularities of contents follow Zipf distribution. A reinforcement learning framework for the dynamic content update using the Markov decision process is proposed [23] with cache and user requests as state space and content eviction and content retaining as action space. The online content popularity is learned based on the context information of connected users by modeling a caching policy as the multi-armed-bandit problem [14]. From the perspective of regional users, an online content popularity prediction algorithm is proposed based on content features to predict content popularity in [12] using logistic regression. The authors in [15], used content features to improve the prediction accuracy in the Bayesian framework. They assume that the demands for the content follow the Poisson distribution and thus incorporate bias in the process. A location-aware content prediction model using a linear model, ridge regression is proposed in [16] based on location and content features. A transfer learning-based caching mechanism is modeled based on learning and transferring the hidden latent features which are extracted from device to device interactions to maximize the back-haul offloading gains in [22]. The aforementioned works assume that content popularity does not change over time or it changes very slowly. However, in practice, the content popularity is non-stationary [24] e.g, viral videos.

B. CONTRIBUTIONS

In this article, we consider an active learning framework to predict the UCR in edge caching systems. A novel adaptive caching framework is proposed to learn the popularities as fast as possible while guaranteeing an operational cache hit ratio. Our contributions are as follows:

- Firstly, we propose an Active learning method to predict the UCR in edge caching networks. We formulate the content caching as the matrix completion problem of demands observed at the small base stations (SBS).
- We then estimate the popularity of the contents using active learning based query-by-committee matrix completion algorithm. Three passive matrix completion algorithms, i.e., singular value thresholding, unconstrained nuclear norm minimization, and matrix factorization, are served as the committee members. Compared with collaborative filtering [25], which has only one matrix completion algorithm compared to QBC.
- Based on the predicted content popularity, we propose two caching policies which maximize the query and guarantee the minimum CHR requirements. The proposed caching algorithms are model-free and applicable to both static and time-varying content popularity models.
- The performance analysis of both caching polices over existing method is established through Monte Carlo simulations. It is shown that the proposed caching schemes

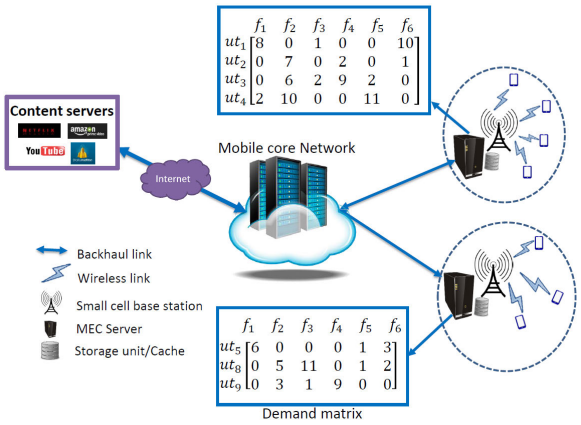


FIGURE 1. The network model.

perform better than random and most popular caching schemes.

The rest of the paper is organized as follows. Section II describes the system model. In section III, we propose an active learning matrix completion algorithm. Section IV presents the caching and query strategies. Section V presents the simulation results and finally, conclusions are drawn in Section VI.

Notation: Lower or upper case letters represent scalars, boldface upper case for matrices, boldface lower case for vectors, $[.]_{a,b}$ represents the element in row a and column b of a matrix, $\|.\|_*$ represents nuclear norm, $\|.\|_F$ represents the frobenius norm, $\odot$ is element-wise product or also called as Hadamard product, $\otimes$ represents Kronecker product, $(.)^T$ denotes the transpose operator, $|.|$ represents the cardinality of set and $\mathbf{1}$ represents a vector of all ones.

II. SYSTEM MODEL

A. SYSTEM MODEL

We consider a heterogeneous cellular network scenario with small base stations connected to the mobile core network over reliable back-haul links, as shown in Fig. 1. Each SBS is equipped with a mobile edge computing server to process the content requests and finite storage memory to cache the contents. Denote $\mathcal{U} = \{t_1, \ldots, t_K\}$ as the set of the user terminals (UTs) connected to the serving SBS, which has access to a library $\mathcal{F} = \{f_1, \ldots, f_F\}$ of F contents at the content server. Let l_i be the size of file f_i and denote $\mathbf{l} = [l_1, \ldots, l_F]$. Further, the coverage areas of the SBS are assumed to be disjoint, thus a UT can only be connected to the closest SBS at a time. Each SBS cache can store up to D Gigabits (Gbits). Upon receiving users' requests, the SBS first checks it's local storage. If the requested content is available in the SBS's cache, it can be served immediately. Otherwise, the requested content is fetched from the server before being sent to the UT.

B. DEMAND MATRIX

The SBS keeps track of content demands from its users via a demand matrix $\mathbf{L} \in \mathbb{Z}^{+K\times F}$, whose rows and columns of $\mathbf{L}$ represent anonymous UT profiles and the contents, respectively. An element $[L]_{k,f} \in \mathbb{Z}^+$ represents the number of requests for content f from UT k. In reality, each UT usually requests only a small number of contents, therefore the demand matrix is sparse and largely rectangular. This is because the number of contents at the content server is much large compared to the UTs connected to SBS. Due to the mobility of UTs the size of demand matrix changes over time, but the preferences of UT remain the same since each UT has its profile history. To maximize the total requests served by the SBS's cache, the demands of content are estimated by predicting the missing entries of the demand matrix. In the next section, we propose an active learning-based matrix completion for the estimation of missing entries of the demand matrix.

TABLE 1. Notations and definitions.

Notation	Definition
$\mathcal{U}$	Set of user terminals
$\mathcal{F}$	Set of content files
$\mathbf{l}$	length of files vector
D	Size of cache at each SBS
$\mathbf{L}$	Demand matrix at SBS
$[L]_{k,f} \in \mathbb{Z}^+$	Number of requests for content f from UT k
N	Number of committee members
$\mathbf{M}_n$	Predicted matrix by the n-th committee model
Q	Query budget
$\mathbf{p}$	Popularity vector
$\mathbf{u}$	Uncertainty vector

III. ACTIVE LEARNING BASED DEMAND MATRIX COMPLETION

Active learning(AL) is a sub-field of machine learning and artificial intelligence: which is the study of computer systems that improve through experience and training. In short, it is a special case of semi-supervised learning in which a learning algorithm can interactively query the oracle/human annotator to acquire high-quality data for training [26]. In the context of matrix completion, AL aims to find the most informative missing entries of the matrix. In the considered caching problem, each SBS has a demand matrix which represents the number of requests for contents from the UTs'. Since each UT requests only a small subset of contents of the total content library, which results in the demand matrix to be sparse with a lot of missing entries. To estimate the missing entries of the demand matrix we use AL-based matrix completion which is discussed in the III-B. The main advantage in AL is that it has the freedom to choose the data it wants to learn from through queries in achieving good performance results. This freedom of choice of data to train helps the learning method fast and helps to estimate the real-time requests. The queries/recommendations are generated based on informativeness, which means a very informative missing entry is that the matrix completion method has a hard time determining its actual true value. This is because if we ask the UT which then responds with its correct true value, knowing the correct true value of the missing entry would improve our matrix completion predictions for similar difficult missing entries. On the other hand, if the matrix completion method is very confident when predicting a missing entry, then knowing it's actual true value is not very helpful since the predicted value will more than likely be correct.

A. QUERYING

A query is defined as the response received from the user terminal to the system. Queries are selected in serial means one at a time or either in batches means several to be labeled at once. In the serial setting, the model/learner selects a single most informative data point to label from a pool of unlabelled data points. It then adds the data point to the training set and retrains the model/learner. Due to this, it is not suitable for many applications, since the process of inducing a model from training data may be slow and expensive [27]. On the other hand in the batch setting, it is more natural to acquire labels for many different data points at once. It is important to notice that in the batch querying, the data points should be diverse (to avoid redundancy) as well as be informative to the model/learner.

B. DEMANDS ESTIMATION BASED ON QBC MATRIX COMPLETION

To estimate the demand matrix missing entries we employ AL-based Query-by-committee(QBC) [28] matrix completion algorithm. The intuition of the QBC approach is that it maintains a group of models that are all trained on the same training data. Each model in the group then predicts on how to label potential input points. The informative missing entries are selected for querying from the input points for which they disagree the most which result in high variance. The fundamental idea of QBC is that the committee of models aim to minimize the version space (set of models consistent with the training set) [29], which is the subset of all hypothesis space[1] that are consistent with the known entries. To constrain the size of version space as small as possible, QBC uses the uncertainty of the predictions for each missing entry. The choice of models in the committee can be selected in many ways [26], e.g., using simple sampling [28].

A fundamental challenge in the estimation of the demand matrix is that only a small subset of requests are observed. This result in the data sparsity and cold start problems [1]. AL-based QBC tackles the data sparsity and cold start problems at the root, by identifying the most informative and useful data that better represents the UT preferences through queries [30]. This can be done in various forms, through serial setting by requesting the UT to assess content-by-content or through batch setting by requesting the UT to assess several contents at a time. Let the total number of passive matrix completion models in the committee be N. The predicted matrix by the n-th committee model is represented by $\mathbf{M}_n$ for $n \in [1, \ldots, N]$. In the following, we have used the three low-rank approximation variants of passive matrix completion algorithms as the members of the committee.

1) SINGULAR VALUE THRESHOLDING

Singular value thresholding (SVT) [31] is a standard low-rank approximation matrix completion algorithm. In this method, the algorithm is iterative and produces a sequence of matrices at each step, and performs soft-thresholding on the singular values obtained at each step. It is inferred as the convex relaxation of a rank minimization problem of a matrix. The minimization problem is defined as,

$$\begin{aligned} &\underset{\mathbf{M}_n}{\text{minimize}} \ \|\mathbf{M}_n\|_* \\ &\text{subject to } [M_n]_{k,f} = [L]_{k,f}, \quad \forall (k,f) : [\Omega]_{k,f} = 1. \end{aligned}$$

To compute the singular values and singular vectors efficiently, we use PROPACK [32]. Unlike Matlab built-in function for 'svds', PROPACK uses the iterative Lanczos algorithm to compute singular values and singular vectors which is efficient. It is about ten times faster than Matlab built-in function for 'svds'.

2) UNCONSTRAINED NUCLEAR NORM MINIMIZATION

In this method, fixed point and Bregman iterative algorithms are used to solve the nuclear norm minimization of a matrix [33]. The homotopy approach is used together with an approximate singular value decomposition (SVD) procedure, which results in a very fast, robust, and powerful algorithm called fixed-point continuation with approximate SVD (FPCA). The unconstrained nuclear norm minimization of a matrix is given as,

$$\underset{\mathbf{M}_n}{\text{minimize}} \ \mu \|\mathbf{M}_n\|_* + \|\Omega \odot (\mathbf{L} - \mathbf{M}_n)\|_F^2,$$

where μ is the regularization constant to avoid overfitting and is determined by cross-validation. Both the nuclear norm and the Frobenius norm are convex functions which results in the whole equation to a convex function. This is solved directly using CVX solver [34] in Matlab.

3) MATRIX FACTORIZATION

Matrix factorization [35] is a way of reducing a matrix into its latent factor space. The minimization of a regularized squared error on the set of observed entries is given as,

$$\underset{\mathbf{X},\mathbf{Y}}{\text{minimize}} \ \|\Omega \odot (\mathbf{L} - \mathbf{X}\mathbf{Y}^T)\|_F^2 + \lambda(\|\mathbf{X}\|_F^2 + \|\mathbf{Y}\|_F^2),$$

where $\mathbf{X} \in \mathbb{R}^{K \times r}$ and $\mathbf{Y} \in \mathbb{R}^{F \times r}$ are two latent factor matrices such that $\mathbf{M}_n = \mathbf{X}\mathbf{Y}^T$, r is the rank, and λ is the regularization constant that is determined by cross-validation. We used an alternating least squares method to solve the minimization problem. Since both the latent factors are unknowns the minimization problem is not convex. But, the minimization problem becomes quadratic by fixing one of the unknown and can be solved optimally. When one of the unknown is fixed the equation is solved as a least-squares problem with other unknown and vice versa, is continued until convergence.

C. POPULARITY AND UNCERTAINTY ESTIMATION

The final estimated demand matrix, denoted by $\mathbf{E}$, is calculated as the average of the predicted matrices by the committee members as,

$$\mathbf{E} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{M}_n. \tag{1}$$

[1] set of possible approximations of true function that the algorithm can create.

The caching at the SBS is done based on two factors: popularity and uncertainty. The popularity indicates the frequency in the demands of the contents across all UTs. The popularity of the content f by the committee member n, say p_f^n, is defined as,

$$p_f^n = \sum_{k=1}^{K} [\mathbf{M}_n]_{k,f}, \; n = 1, \ldots, N, \; f = 1, \ldots, F. \quad (2)$$

However, each committee member predicts the matrices (i.e. $\{\mathbf{M}_n\}_{n=1}^N$) differently, which results in different popularity values (i.e., $\{p_f^n\}_{n=1}^N$) associated with file f. The uncertainty of file f, denoted by u_f, is defined as the range of the predicted popularity values for a file across all committee member estimates associated with file f and is mathematically defined as,

$$u_f = \max_{n=1,\ldots,N} p_f^n - \min_{n=1,\ldots,N} p_f^n. \quad (3)$$

With the defined notations, the popularity of the file f, say p_f, is defined as the mean of the popularities of estimates by all the committee i.e.,

$$p_f = \frac{1}{N} \sum_{n=1}^{N} p_f^n, f = 1, \ldots, F. \quad (4)$$

Further, the popularity and uncertainty vectors are defined as $\mathbf{p} = [p_1, \ldots, p_F]$ and $\mathbf{u} = [u_1, \ldots, u_F]$ respectively. In the sequel, we jointly optimize the caching and query processes.

IV. CACHE AND QUERY STRATEGY

In this section, we propose two caching policies to maximize the caching and query performance based on the predicted popularities.

The storage at the SBS is used to store both the popular and uncertain (most informative contents). This allows the system to leverage the trade-off between exploration and exploitation. Here, the exploration is associated with the uncertainty of the contents, and exploitation is associated with the popularity of the contents. Therefore, the system finds more information about UT's preferences by exploring the uncertain contents and then exploits the popular content to maximize the system performance, e.g., cache hit ratio (CHR). In the following, we propose two caching policies for storing the contents at the cache.

A. FIXED-MEMORY CACHING POLICY

In this caching policy, the storage at the SBS is divided into two parts, one part stores the most popular contents which is referred to as exploitation of contents, and the second part stores the uncertain contents which is referred to as exploration of contents. The number of uncertain contents to be cached alongside with the popular contents in the cache is given by query budget Q ($< D$) Gbits. The selection set and cache placement vector is defined by $\mathcal{C}$ and binary vector $\mathbf{x} \in \{0, 1\}^{F \times 1}$. As described in the Algorithm 1, the top $D-Q$ contents of $\mathbf{p}$ and the top Q contents of $\mathbf{u}$ are selected to cache placement $\mathbf{x}$. The stopping criteria of the algorithm is defined

Algorithm 1 Fixed-Memory Caching With Active Matrix Completion

1: Initialize: $\mathbf{L}, \Omega, N, D, Q$
2: repeat
3: $[\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_N] = \text{QBC}(\mathbf{L}, \Omega, N)$,
4: calculate $\mathbf{p}$ and $\mathbf{u}$ as given in III respectively
5: Sorting and Indexing: $[\mathbf{p}_{value}, \mathbf{p}_{index}] = \text{sort}(\mathbf{p}, \text{descend})$ $[\mathbf{u}_{value}, \mathbf{u}_{index}] = \text{sort}(\mathbf{u}, \text{descend})$
6: Selection set: $\mathcal{C}\,\mathcal{C} = \{\mathcal{F}_i \cup \mathcal{F}_{i^\dagger} \mid i = \mathbf{p}_{index}[1 : D - Q], \quad i^\dagger = \mathbf{u}_{index}[1 : Q]\}$
7: Placement vector: $\mathbf{x} \in \{0, 1\}^{F \times 1}$

$$[x]_f = \begin{cases} 1, & f \in \mathcal{C} \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

8: Query generation: Get the uncertain entries of the Q placed contents to query.
9: until: stopping criteria

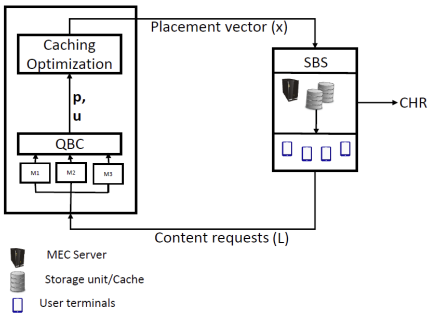


FIGURE 2. System model which iterates between placing files in the cache and receiving feedback of the number of requests.

when the uncertainty vector becomes zeros which implies that all the entries of demand matrix are perfectly known as a result the placement vector does not change and will have a maximum cache hit ratio.

B. ADAPTIVE CACHING POLICY

In this subsection, we define an adaptive caching policy based on optimization formulation for cache placement, shown in Fig. 2. Since we want to store both popular and uncertain contents in the cache we define two binary vectors which indicate 1 if the content is stored in cache and 0 if it is not stored. Let $\mathbf{x}_1 \in \{0, 1\}^{F \times 1}$ and $\mathbf{x}_2 \in \{0, 1\}^{F \times 1}$ be the binary vectors which define the cache placement of the contents. $\mathbf{x}_1$ includes files selected which are popular to achieve exploration while $\mathbf{x}_2$ are uncertain contents to exploitation. We formulate an optimization problem that maximizes the exploration under guaranteed cache hit ratio as,

$$\begin{aligned} \mathcal{P}_1 : \max_{\mathbf{x}_1, \mathbf{x}_2} \; & (\mathbf{u} \odot \mathbf{l}^T)\mathbf{x}_2 \\ \text{s.t. } C_1 : \; & \mathbf{x}_1, \mathbf{x}_2 \in \{0, 1\}^{F \times 1}, \\ C_2 : \; & \mathbf{x}_1 + \mathbf{x}_2 \leq \mathbf{1}, \\ C_3 : \; & \mathbf{x} = \mathbf{x}_1 + \mathbf{x}_2, \\ C_4 : \; & \mathbf{x}^T \mathbf{l} \leq D, \\ C_5 : \; & \frac{((\mathbf{p} - \mathbf{u}) \odot \mathbf{l}^T)\mathbf{x}_1}{(\mathbf{p} \odot \mathbf{l}^T)} \geq \theta. \end{aligned} \quad (6)$$

Remarks:

- Formally, $\mathbf{x}_1$ and $\mathbf{x}_2$ are two binary decision variables for caching the popular contents and to querying the uncertain contents respectively.
- The objective of the function $\mathcal{P}_1$ is responsible for maximizing the exploration of uncertain contents.
- The constraints C_2 and C_3 describes the files to cache in the storage.
- C_4 is the maximum storage constraint.
- And the last constraint C_5 denote the guaranteed cache hit ratio greater than or equal to a given CHR target $\theta \in [0, 1]$. Note that for known files, the corresponding uncertainty in $\mathbf{u}$ is zero.

The optimization problem $\mathcal{P}_1$ regulates the trade-off between exploration and exploitation by ensuring the guaranteed cache hit ratio (GCHR) greater than or equal to θ. The queries are generated based on the binary decision variable $\mathbf{x}_2$. We used Mosek solver of CVX [34] to solve the optimization problem $\mathcal{P}_1$ in Matlab.

Due to the nature of optimization problem $\mathcal{P}_1$, whenever the variance of uncertain contents goes to zero implies when there is no exploration the storage will not be utilized completely. The size of remaining storage is given as,

$$D_{new} = D - (\mathbf{x}^T \mathbf{l}). \tag{7}$$

Based on the remaining storage, we formulate an optimization problem that maximizes the CHR by storing the most popular contents in the remaining storage and is referred to as enhanced GCHR. So, when the uncertain vector is zero we focus on the exploitation of the learned phase by utilizing the full cache storage. This is mathematically formulated as,

$$\begin{aligned} \mathcal{P}_2 : \max_{\mathbf{y}} \ & \frac{(\mathbf{p} \odot \mathbf{l}^T)\mathbf{y}}{(\mathbf{p} \odot \mathbf{l}^T)} \\ \text{s.t. } & C_1 : \mathbf{y} \in \{0, 1\}^{F \times 1}, \\ & C_2 : \mathbf{x} + \mathbf{y} \leq \mathbf{1}, \\ & C_3 : \mathbf{y}^T \mathbf{l} \leq D_{new}. \end{aligned} \tag{8}$$

Remarks:

- Where $\mathbf{y}$ is the binary decision variable for caching the popular contents and $\mathbf{x}$ is the placement vector of the optimization problem $\mathcal{P}_1$.
- C_2 describes the files to cache.
- C_3 is the storage constraint.

The optimization problem $\mathcal{P}_2$ aims at maximizing the cache hit ratio by storing the most popular contents. The additional contents to be stored after the optimization problem $\mathcal{P}_1$ to fill the cache completely is given by $\mathbf{y}$. The optimization problem $\mathcal{P}_2$ is solved using the Mosek solver of CVX [34]. Note that, in both the caching policies the MEC server check and skip the contents, if the contents are already in the cache. The popularity and uncertainty of the contents change over time, so before making content eviction and replacement the popularity and uncertainty of contents are calculated.

V. NUMERICAL RESULTS

We evaluate the performance of the proposed caching policies through numerical results. First, we consider the stationary demand model and evaluate the performance of the fixed-memory caching policy. Then, we consider the non-stationary demand model and evaluate the performance of the adaptive caching policy. For both the scenarios, we consider the total number of contents (F) at the server is 100 and the number of user terminals (K) connected to SBS at a given time is 30.[2] The UTs connected to SBS changes from time due to mobility and as a result, the demand matrix also changes over time, for simulations we consider an average of 30 UTs connected to SBS at any given time. The content requests from the UTs for the contents can follow any distribution since our proposed caching polices are model free. However, we consider that the requests follow a Zipf-like distribution denoted by,

$$P_k(f) = \omega / f^{\alpha}, \tag{9}$$

where $\omega = \left(\sum_{f=1}^{F} 1/f^{\alpha} \right)^{-1}$ and α is the Zipf skewness factor.

A. STATIONARY DEMAND MODEL

We analyze the performance of fixed-memory caching policy as metrics of root mean square (RMSE), cache hit ratio (CHR), and back-haul load. For this, we generate a demand matrix $\mathbf{L}_{true}$ with all entries and delete 98% of entries to evaluate the performance of the active learning query approach in finding the missing entries. with the help of notations defined, the metrics are defined as

$$\text{RMSE} = \frac{1}{\|\mathbf{L}_{true}\|_F} \|\mathbf{L}_{true} - \mathbf{L}\|_F, \tag{10}$$

$$\text{CHR} = \frac{\mathbf{p}_{true}\mathbf{x}^T}{\mathbf{p}_{true}\mathbf{1}^T}. \tag{11}$$

The back-haul load is defined as the number of new contents fetched from the content server. Note that, before storing the contents in the cache, the MEC server will check and skip the contents if it is already stored. The performance obtained by random query strategy and collaborative filtering [25] is used as the benchmark for comparison of the results and matrix factorization is used as the baseline without query strategy. In figure 3, we evaluate the performance of the fixed-memory caching policy via the RMSE metric. The aim is to study the rate of reduction of RMSE When more and more contents are explored. Initially, the active learning approach performs poorly since 98% of the entries are missing. As a result, the random caching and collaborative filtering scheme achieve smaller RMSE until iteration 25 for $Q = 2$. This effect is explained by the property called incoherence property [36]. It is seen that after 25 iterations the rate of decrease in RMSE is faster with active learning compared to random caching and collaborative filtering. This is due to the fact that using active learning the contents to a query

[2]We limit the total UTs to n_{max}

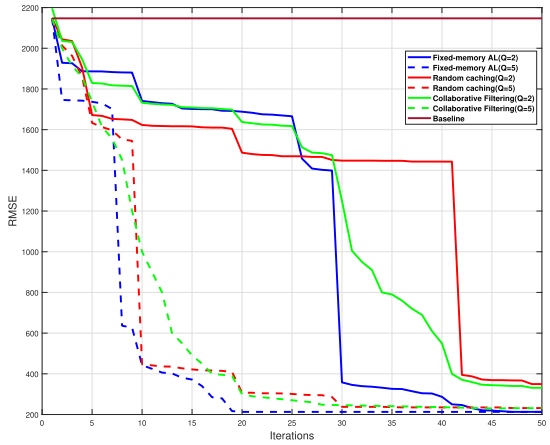


FIGURE 3. RMSE performance comparison, storage size = 30, and skewness factor $\alpha = 0.8$.

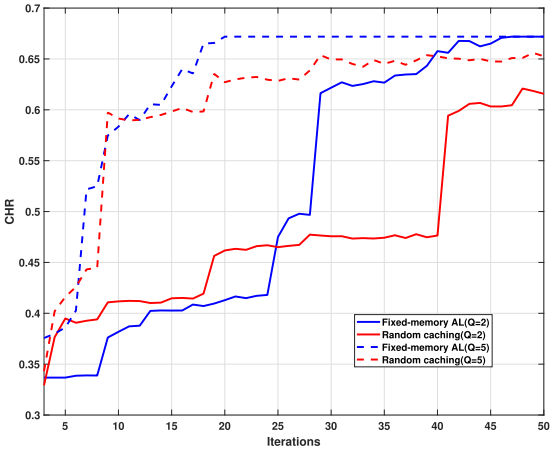


FIGURE 4. Impact of CHR, storage size = 30, and skewness factor $\alpha = 0.8$.

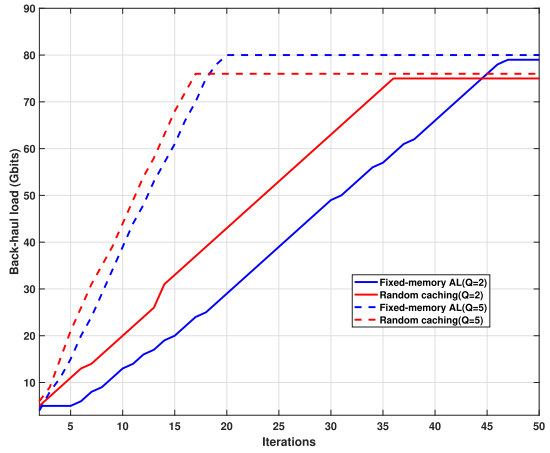


FIGURE 5. Impact of back-haul load, storage size = 30, and skewness factor $\alpha = 0.8$.

are selected in a way that yields a finite information gain. By increasing the query budget to $Q = 5$, AL-based caching outperforms the random caching policy after iteration 8.

Figure 4 presents the CHR as a function of the query iterations. Similar to figure 3, the active learning with query budget is 2 approach has poor performance until iteration

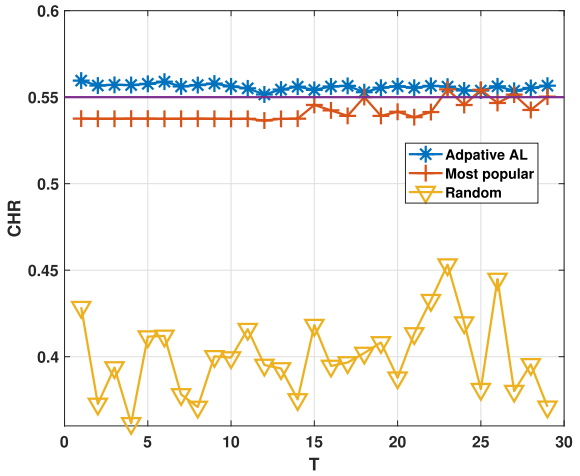


FIGURE 6. Impact of CHR, storage size = 20, and skewness factor $\alpha = 0.7$.

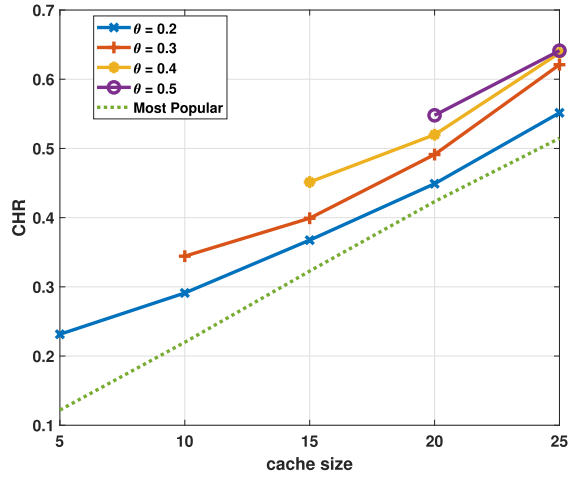


FIGURE 7. The average over time CHR vs cache capacity.

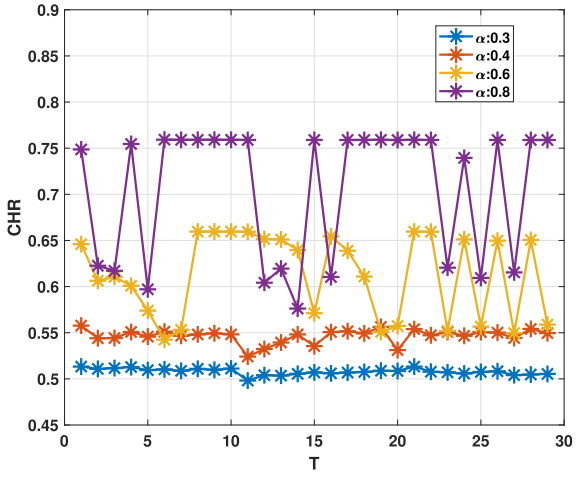
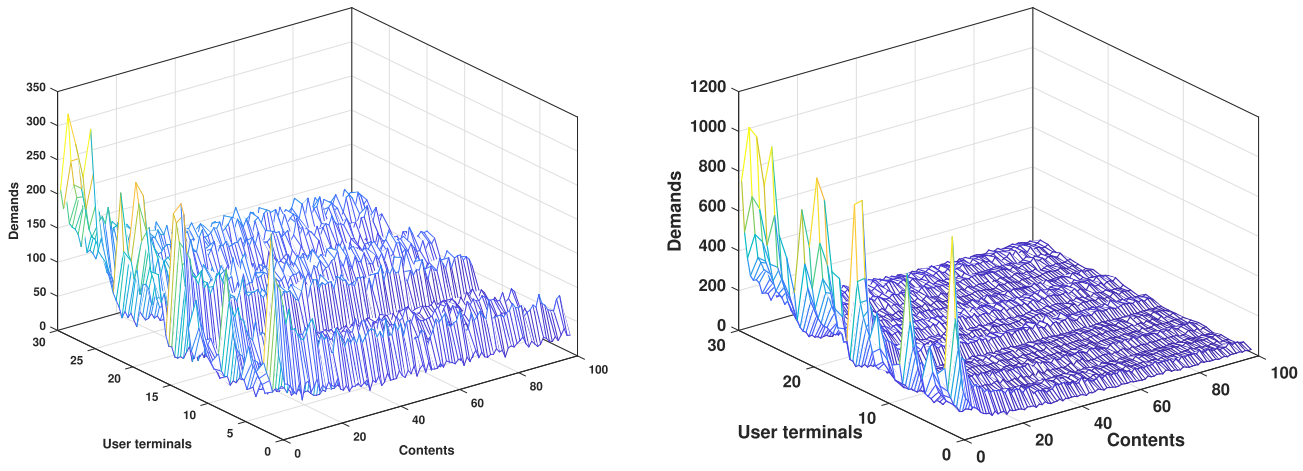


FIGURE 8. Impact of CHR vs Skewness factor.

25 due to incoherence property. After 25 iterations CHR improves drastically compared to the random caching policy for the same reason mentioned in figure 3. However, the performance of active learning can be improved by increasing



(a) Skewness factor = 0.3

(b) Skewness factor = 0.7

FIGURE 9. Data distribution at time slot T = 1.

the query budget and is illustrated with the query budget to 5 in figure 4. By increasing the query budget induces higher back-haul load as shown in figure 5. The performance of active learning with query budget 5 can be achieved without increasing the back-haul load with query budget 2 with the number of iterations. This impact can be seen around the iteration 45-50 in figure 4. The performance of the AL-based caching scheme in terms of the back-haul load is shown in figure 5. Active learning with query budget 2 requires lesser back-haul load compared to a random querying strategy. However, the active learning method with query budget 2 imposes a slightly higher back-haul load after 45 iterations and converges. This slight increase in back-haul load results in higher CHR as explained in figure 4. Moreover, increasing the query budget results in higher back-haul load, this is observed in figure 5.

B. NON-STATIONARY DEMAND MODEL

In this sub-section, we consider the non-stationary online demands for the contents, also we analyze the performance of the adaptive caching policy. The relation between user terminals and contents at t^{th} time slot are modelled by demand matrix represented as $\mathbf{L}_t \in \mathbb{Z}^{+K\times F}$. The entries of the demand matrix represent the number of times the contents have been requested by the user terminals. To capture the evolution of content requests over time, we assume that $\mathbf{L}_t$ follows a first-order Markov process:

$$\begin{aligned} \mathbf{L}_t &= \mathbf{L}_{t-1} + \mathbf{E}_t, \\ \mathbf{L}_t &= \max(\mathbf{L}_t, 0) \quad t = 1 \ldots T. \end{aligned} \tag{12}$$

where $\mathbf{E}_t \sim \mathcal{N}(0, \mathbf{Q})$. $\mathbf{Q}$ is modeled as a correlation between user terminals and contents given by $\mathbf{Q} = \mathbf{Q}_F \otimes \mathbf{Q}_U$. Thus, the dependencies are modeled as the Kronecker product of covariance matrices $\mathbf{Q}_F$ of contents and $\mathbf{Q}_U$ of user terminals. The covariance matrices generated are symmetric and positive semi-definite, with unit diagonal and other elements in the closed interval $[-1, 1]$ [37].

We generate the entire demand matrix $\mathbf{L}_{true,1}$ at the initial time slot T = 1, we then use first-order Markov process to generate a sequence of demand matrices over the time from multivariate normal distribution as described above. We model the observed $\mathbf{L}_1$ at T = 1, by deleting the 50% of entries randomly. This implies only 50% of entries are observed at an initial time slot. From the T = 2, the missing entry percentage depends on the contents placed in the cache and forgetting factor, which is defined as, more importance is given to recent observations and less importance to earlier data [38]. In our simulations, we use fixed forgetting factor based on the bootstrap method.

The effect of the skewness factor can be seen in figure 9. As seen in the figure the contents are uniformly distributed when the skewness factor is equal to 0.3 and are more skewed when it is equal to 0.7. In figure 6, the performance of the proposed active learning caching is compared with the most popular caching and random caching schemes. The cache hit ratio is shown as the function of time. The goal of this study shows that the proposed caching scheme always maintains the cache hit ratio above 0.55 which is a guaranteed cache hit ratio. While the most popular and random caching schemes perform poorly than the proposed scheme.

In figure 7, we show the average cache hit ratio as a function of varying cache size for different guarantee cache hit ratio (θ). As seen in the figure, as θ increases the average cache hit ratio also increases. For greater θ values need higher cache size, for example when $\theta = 0.3$ the valid cache size is 10 Mbits below this the proposed optimization problem is infeasible. Higher θ value needs a higher cache size. The impact of the cache hit ratio as a function of the skewness factor is shown in figure 8. The cache hit ratio increase with the skewness factor. This is because for higher skewness factor there are only a few most popular contents and our proposed learning model effectively finds those contents and cache them for exploitation, as a result, higher skewness factor results in a higher cache hit ratio.

VI. CONCLUSION

In this article, we studied a novel active learning-based caching framework at the edge node. By formulating the content caching problem as a matrix completion problem, an active learning query by committee approach was used to predict the missing entries. The most informative missing entries are selected based on uncertainty measure to query. The interactive learning between the system and user terminals helps the user terminals become more self-aware of their own likes/dislikes while at the same time providing new information to the system which helps in better estimation of the popularity of contents. Our proposed caching framework is model-free, it can be used either for fixed or time-varying popularity learning situations. The superiority in the performance of both the caching policies over the state-of-the-art method is established through simulations.

ACKNOWLEDGMENT

This article was presented in part at the IEEE Asilomar Conference on Signals, Systems, and Computers, 3-6 November 2019, Pacific Grove, CA, USA.

REFERENCES

[1] S. Bommaraveni, T. X. Vu, S. Vuppala, S. Chatzinotas, and B. Ottersten, "Active content popularity learning via query-by-committee for edge caching," in *Proc. 53rd Asilomar Conf. Signals, Syst., Comput.*, Nov. 2019, pp. 301–305.

[2] Cisco, "Cisco visual networking index: Forecast and trends, 2017–2022," White Paper, Feb. 2019.

[3] B. Perabathini, E. Bastug, M. Kountouris, M. Debbah, and A. Conte, "Caching at the edge: A green perspective for 5G networks," in *Proc. IEEE Int. Conf. Commun. Workshop (ICCW)*, Jun. 2015, pp. 2830–2835.

[4] 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.

[5] T. X. Vu, L. Lei, S. Vuppala, A. Kalantari, S. Chatzinotas, and B. Ottersten, "Latency minimization for content delivery networks with wireless edge caching," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.

[6] M. A. Maddah-Ali and U. Niesen, "Cache-aided interference channels," *IEEE Trans. Inf. Theory*, vol. 65, no. 3, pp. 1714–1724, Mar. 2019.

[7] J. Li, M. Liu, J. Lu, F. Shu, Y. Zhang, S. Bayat, and D. N. K. Jayakody, "On social-aware content caching for D2D-enabled cellular networks with matching theory," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 297–310, Feb. 2019.

[8] E. Bastug, M. Bennis, and M. Debbah, "Cache-enabled small cell networks: Modeling and tradeoffs," in *Proc. 11th Int. Symp. Wireless Commun. Syst. (ISWCS)*, Aug. 2014, pp. 649–653.

[9] K. Poularakis, G. Iosifidis, and L. Tassiulas, "Approximation algorithms for mobile data caching in small cell networks," *IEEE Trans. Commun.*, vol. 62, no. 10, pp. 3665–3677, Oct. 2014.

[10] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[11] 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.

[12] Y. Jiang, M. Ma, M. Bennis, F.-C. 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.

[13] 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.

[14] 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, Feb. 2017.

[15] S. Mehrizi, A. Tsakmalis, S. Chatzinotas, and B. Ottersten, "A feature-based Bayesian method for content popularity prediction in edge-caching networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, May 2019, pp. 1–6.

[16] P. Yang, N. Zhang, S. Zhang, L. Yu, J. Zhang, and X. Shen, "Content popularity prediction towards location-aware mobile edge caching," *IEEE Trans. Multimedia*, vol. 21, no. 4, pp. 915–929, Apr. 2019.

[17] E. Bastug, J.-L. Guenego, and M. Debbah, "Proactive small cell networks," in *Proc. ICT*, May 2013, pp. 1–5.

[18] L. E. Chatzieleftheriou, M. Karaliopoulos, and I. Koutsopoulos, "Jointly optimizing content caching and recommendations in small cell networks," *IEEE Trans. Mobile Comput.*, vol. 18, no. 1, pp. 125–138, Jan. 2019.

[19] B. Settles, "Active learning literature survey," Univ. Wisconsin–Madison, Madison, WI, USA, Comput. Sci. Tech. Rep. 1648, 2009.

[20] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: Wireless video content delivery through distributed caching helpers," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 1107–1115.

[21] T. X. Vu, S. Chatzinotas, and B. Ottersten, "Edge-caching wireless networks: Performance analysis and optimization," *IEEE Trans. Wireless Commun.*, vol. 17, no. 4, pp. 2827–2839, Apr. 2018.

[22] E. Bastug, M. Bennis, and M. Debbah, "A transfer learning approach for cache-enabled wireless networks," in *Proc. 13th Int. Symp. Modeling Optim. Mobile, Ad Hoc, Wireless Netw. (WiOpt)*, May 2015, pp. 161–166.

[23] P. Wu, J. Li, L. Shi, M. Ding, K. Cai, and F. Yang, "Dynamic content update for wireless edge caching via deep reinforcement learning," *IEEE Commun. Lett.*, vol. 23, no. 10, pp. 1773–1777, Oct. 2019.

[24] A. Tatar, M. D. de Amorim, S. Fdida, and P. Antoniadis, "A survey on predicting the popularity of Web content," *J. Internet Services Appl.*, vol. 5, no. 1, p. 8, Dec. 2014.

[25] E. Bastug, M. Bennis, E. Zeydan, M. A. Kader, I. A. Karatepe, A. S. Er, and M. Debbah, "Big data meets telcos: A proactive caching perspective," *J. Commun. Netw.*, vol. 17, no. 6, pp. 549–557, Dec. 2015.

[26] B. Settles, *Active Learning*. San Rafael, CA, USA: Morgan & Claypool, 2012.

[27] B. Settles, "From theories to queries: Active learning in practice," in *Proc. Active Learn. Exp. Design Workshop Conjunct. (AISTATS)*, vol. 16, May 2011, pp. 1–18.

[28] H. S. Seung, M. Opper, and H. Sompolinsky, "Query by committee," in *Proc. 5th Annu. workshop Comput. Learn. Theory (COLT)*, 1992, pp. 287–294.

[29] T. M. Mitchell, "Generalization as search," *Artif. Intell.*, vol. 18, no. 2, pp. 203–226, Mar. 1982.

[30] M. Elahi, F. Ricci, and N. Rubens, "Active learning in collaborative filtering recommender systems," in *E-Commerce and Web Technologies*, M. Hepp and Y. Hoffner, Eds. Cham, Switzerland: Springer, 2014, pp. 113–124.

[31] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM J. Optim.*, vol. 20, no. 4, pp. 1956–1982, Jan. 2010.

[32] R. M. Larsen. *PROPACK–Software for Large and Sparse SVD Calculations*. [Online]. Available: http://sun.stanford.edu/rmunk/PROPACK/

[33] S. Ma, D. Goldfarb, and L. Chen, "Fixed point and Bregman iterative methods for matrix rank minimization," *Math. Program.*, vol. 128, nos. 1–2, pp. 321–353, Jun. 2011.

[34] I. CVX Research. (Aug. 2012). *CVX: MATLAB Software for Disciplined Convex Programming, Version 2.0*. [Online]. Available: http://cvxr.com/cvx

[35] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.

[36] E. J. Candes and B. Recht, "Exact low-rank matrix completion via convex optimization," in *Proc. 46th Annu. Allerton Conf. Commun., Control, Comput.*, Sep. 2008, pp. 806–812.

[37] K. Numpacharoen and A. Atsawarungruangkit, "Generating correlation matrices based on the boundaries of their coefficients," *PLoS ONE*, vol. 7, no. 11, pp. 1–7, Nov. 2012, doi: 10.1371/journal.pone.0048902.

[38] T. J. Brailsford, J. H. W. Penm, and R. D. Terrell, "Selecting the forgetting factor in subset autoregressive modelling," *J. Time Ser. Anal.*, vol. 23, no. 6, pp. 629–649, Nov. 2002, doi: 10.1111/1467-9892.00283.

SRIKANTH BOMMARAVENI (Student Member, IEEE) received the bachelor's degree in electronics and communication engineering from Jawaharlal Nehru Technological University Hyderabad, Hyderabad, India, in 2013, and the M.Sc. degree in computer and communications technology from Saarland University, Saarbrücken, Germany, in 2017. He is currently pursuing the Ph.D. degree with the Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, Luxembourg. His research interests include data analysis, machine learning, and its applications in the field of telecommunications.

THANG X. VU (Member, IEEE) was born in Hai Duong, Vietnam. He received the B.S. and M.Sc. degrees in electronics and telecommunications engineering from the VNU University of Engineering and Technology, Vietnam, in 2007 and 2009, respectively, and the Ph.D. degree in electrical engineering from the University of Paris-Sud, France, in 2014. From 2007 to 2009, he was a Research Assistant with the Department of Electronics and Telecommunications, VNU University of Engineering and Technology. From September 2010 to May 2014, he was with the Laboratory of Signals and Systems, a joint Laboratory of CNRS, CentraleSupélec, and the University of Paris-Sud XI. From July 2014 to January 2016, he was a Postdoctoral Researcher with the Information Systems Technology and Design Pillar, Singapore University of Technology and Design, Singapore. He is currently a Research Associate with the Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg. His research interests include wireless communications, with particular interests in cache-assisted 5G, cloud radio access networks, resources allocation and optimization, cooperative diversity, channel and network decoding, and iterative decoding. In 2010, he received the Allocation de Recherche Fellowship to study the Ph.D. degree in France.

SYMEON CHATZINOTAS (Senior Member, IEEE) received the M.Eng. degree in telecommunications from the Aristotle University of Thessaloniki, Thessaloniki, Greece, in 2003, and the M.Sc. and Ph.D. degrees in electronic engineering from the University of Surrey, Surrey, U.K., in 2006 and 2009, respectively. He was involved in numerous research and development projects with the National Center for Scientific Research Demokritos, Institute of Informatics Telecommunications, the Center of Research and Technology Hellas, Institute of Telematics and Informatics, and the Center of Communication Systems Research, Mobile Communications Research Group, University of Surrey. He is currently the Deputy Head of the Interdisciplinary Centre for Security, Reliability, and Trust, SIGCOM Research Group, University of Luxembourg, Luxembourg, and a Visiting Professor with the University of Parma, Italy. He has over 250 publications, 2000 citations, and H-index of 25 according to Google Scholar. His research interests include multiuser information theory, cooperative/cognitive communications, and wireless networks optimization. He was a co-recipient of the 2014 Distinguished Contributions to Satellite Communications Award and the CROWNCOM 2015 Best Paper Award. He serves on the Satellite and Space Communications Technical Committee and the IEEE Communications Society.

BJÖRN OTTERSTEN (Fellow, IEEE) was born in Stockholm, Sweden, in 1961. He received the M.S. degree in electrical engineering and applied physics from Linköping University, Linköping, Sweden, in 1986, and the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, USA, in 1990. He has held research positions at the Department of Electrical Engineering, Linköping University, the Information Systems Laboratory, Stanford University, the Katholieke Universiteit Leuven, Leuven, Belgium, and the University of Luxembourg, Luxembourg. From 1996 to 1997, he was the Director of research with ArrayComm, Inc., a start-up in San Jose, CA, USA, based on his patented technology. In 1991, he was appointed as a Professor of signal processing with the Royal Institute of Technology (KTH), Stockholm. From 1992 to 2004, he was the Head of the Department for Signals, Sensors, and Systems, KTH. From 2004 to 2008, he was the Dean of the School of Electrical Engineering, KTH. He is currently the Director of the Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg. He is a Fellow of EURASIP. He is a member of the Editorial Board of *Signal Processing* (EURASIP), the *EURASIP Journal on Advances in Signal Processing*, and *Foundations and Trends in Signal Processing*. He was a recipient of the IEEE Signal Processing Society Technical Achievement Award, in 2011, the European Research Council Advanced Research Grant, from 2009 to 2013, and the European Research Council Advanced Research Grant, since 2017. He has coauthored journal articles that received the IEEE Signal Processing Society Best Paper Award, in 1993, 2001, 2006, and 2013, and seven IEEE conference Best Paper Awards. He has served on the Editorial Board of *IEEE Signal Processing Magazine*. He has served as an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING.