Online Tensor Completion Based Prediction For Wireless Edge Caching
Navneet Garg, Member, IEEE, and Tharmalingam Ratnarajah, Senior Member, IEEE.

Abstract—Wireless edge caching is a popular strategy to avoid backhaul congestion in the next generation networks, where the content is cached in advance at base stations to serve redundant requests during peak congestion periods. In the edge caching data, the missing observations are inevitable due to dynamic selective popularity. Among the completion methods, the tensor-based models have been shown to be the most advantageous for missing data imputation. Also, since the observations are correlated across time, files, and base stations, in this paper, we formulate the cooperative caching with recommendations as a fourth-order tensor completion and prediction problem. Since the content library can be large leading to a large dimension tensor, we modify the latent norm-based Frank-Wolfe (FW) algorithm with towards a much lower time complexity using multi-rank updates, rather than rank-1 updates in literature. This significantly lower time computational overhead leads in developing an online caching algorithm. With MovieLens dataset, simulations show lower reconstruction errors for the proposed algorithm as compared to that of the recent FW algorithm, albeit with lower computation overhead. It is also demonstrated that the completed tensor improves normalized cache hit rates for linear prediction schemes.

I. INTRODUCTION

With the continuous development of various intelligent devices and various sized innovative application services such as high quality video feeds, software updates, news updates, etc., wireless mobile communications has been experiencing an unprecedented traffic surge with a lot of redundant and repeated information, which limits the capacity of the fronthaul and backhaul links [1]. To lower the redundant traffic, caching has emerged as an effective solution for reducing the peak data rates by pre-fetching the most popular contents in the local cache storage of the base stations (BS). In the recent years, caching at the BS is actively feasible due to the reduced cost and size of the memory [2]. In [2] where cache-enabled networks are classified into macro-cell, heterogeneous and D2D networks, given a set of a content library and the respective content popularity profile, content placement and delivery have been investigated in order to optimize the backhaul latency delay in [3], server load in [4], cache miss rate in [5]–[7], etc. With the known popularity profile, reinforcement learning approaches [8], [9] are studied for learning the content placement. However, in practice, this profile is time-varying and not known in advance, therefore, it needs to be estimated from the past observations of the content requests. To estimate future popularities, deep learning based prediction is employed with huge training data in [10], [11]. In [12], auto regressive (AR) prediction cache is used to predict the number of requests in the time series. Linear prediction approach is investigated for video segments in [13]. To learn popularities independently across contents, online policies are presented for cache-awareness in [14], low complexity video caching in [1], [15], user preference learning in [16], etc. These works on prediction focus independently on the BSs to estimate the future content popularities. However, the demands across files are correlated [17], since other similar contents can be served from the cache with the same features as the requested content in order to maximize the cache hit, which is also known as soft cache hit [18]. Regarding that in literature [17], [19]–[24], recommendation based caching has been carried out based on low-rank decomposition, deep reinforcement learning, etc. Moreover, the content is also correlated across base stations, as studied in the recent works on in-network caching solutions [25], [26], where base stations are jointly allocated cache contents. Furthermore, regarding the correlation of popularities across time slots, in edge caching literature, to avoid prediction, cache placement is performed to maximize different objectives such as cache hit rate [6], average success probability [27], [28], etc., given the past information up to the present time slot. The maximization problems of these objectives can be simplified to the prediction problem. Therefore, in this work, we focus on prediction of such correlated demands to improve the caching performance using a tensor approach. Due to these correlations and missing data, it is difficult to store and predict the content popularities for large content library. Thus, after modeling the tensor completion problem, we modify the Frank-Wolfe approach for the solution, followed by linear prediction methods. A brief review of tensor completion approaches is given as follows.

A. Tensor completion methods

In the past decades, tensor completion is intensively researched due to its wide applications in a variety of fields, such as computer vision [29]–[31], multi-relational link prediction [32]–[34], and recommendation system [35], [36]. The goal of tensor completion is to recover an incomplete tensor from partially observed entries. To the best of our knowledge, tensor completion methods can be categorized into decomposition based and rank-minimization based methods.

Decomposition based methods aim to factorize the incomplete tensor into a sequence of low-rank factors and then predict the missing entries via the latent factors. In recent years, CANDECOMP/PARAFAC (CP) decomposition [37]...
and Tucker decomposition [38] are the two most studied and popular models applied in tensor completion in [39], [40], [41]. Although CP and Tucker methods obtain good performance for low order tensors, yet their performance rapidly degrades for higher-order tensors. Moreover, the computing of CP-rank is NP hard and the number of parameters of Tucker decomposition is exponential with the order of given tensors. Recently, a tensor decomposition model, called Tensor-Ring (TR) decomposition [42]–[45], is proposed to process high-order tensors. TR decomposition can express a higher-order tensor by a multi-linear product over a sequence of lower-order latent cores. A notable advantage of TR decomposition is that its total number of parameters increases linearly with the order of the given tensor, reducing the curse of dimensionality as compared to Tucker decomposition. Attracted by these features, TR decomposition has drawn lots of attention such as TR based weighted optimization [46], alternating least squares [47], low-rank factors.

Rank-minimization based methods exploits the low-rank structure to complete the tensor. Since the rank minimization is a non-convex and NP-hard problem, overlapped nuclear norm [48]–[50] and latent nuclear norm [51], [52] have been defined as the convex surrogates of tensor rank. The former norm assumes low-rank across all modes and thus perform poorly, when the target is low-rank in certain modes [48]. The latter norm assumes few modes in low-rank and often performs better than the former [51]. However, these two norms are based on the unbalanced mode-\(k\) unfolding, and thus, causing lack of capturing global information for higher-order tensors.

B. Contributions

In this paper, inspired by the TR decomposition, we employ convex TR-based latent nuclear norm [53] based on cyclic unfolding, i.e. overcoming the drawback of unbalanced unfolding. The tensor completion task can be cast as a convex optimization problem to minimize the latent norm, which is solved via modified Frank-Wolfe (FW) algorithm [53], [54] with cyclic unfolding to provide an efficient solution towards lower time complexity. In this work, FW method is modified to obtain a low time-complexity solution via gradient descent updates. Furthermore, simulations on MovieLens dataset are carried out to verify the algorithm and completion results for cache hit rate in caching. The contributions of this paper can be summarized as follows.

- First, the missing entry problem in edge caching is cast as a tensor completion problem, where the entries are correlated across base stations, files and time. To solve the tensor completion problem, we modify the FW algorithm from [53] towards a lower time complexity solution. The modified approach is significantly faster than that in [53]. Using the proposed tensor completion, an online content caching algorithm is presented to improve the cache hit rates.
- Simulations are performed for MovieLens dataset towards the convergence and observing the effect of latent factors. The cache hit rate performance of caching is plotted for both mean based and linear prediction methods, which also shows normalized cache hit rate improvements compared to conventional CP decomposition (CPD) [55] and the FW method in [53]. For higher decomposition rank, the proposed method provides better hit rates than that with [53].

Related work on TR based completion

The presented random-mode FW algorithm is related to the previously studied latent-norm based methods [51]–[53] and Tensor-Ring based methods [47], [50], [51] employed the latent nuclear norm, while [52] defined a new latent nuclear norm via Tensor Train. However, they are based on unbalanced mode-\(k\) unfolding. [47] applied TR decomposition with alternating least squares for completion, while [56] proposed TR low-rank factors. To reduce the computational complexity per iteration, [50] utilize an overlapped TR nuclear norm. Further, to reduce the number of parameters for selection, [53] proposed a new latent TR-nuclear norm.

Organization

The rest of this paper is organized as follows. The prediction in the edge caching framework is presented in Section II. In Section III tensor completion algorithm is provided. Section IV investigates simulation results for the real-world dataset. Finally, the paper is concluded in section V.

Notations

Scalars, vectors, and matrices are respectively denoted by lowercase, boldface lowercase, and bold capital letters. A tensor of order \(N > 3\) is denoted by calligraphic letter \(X\). The notation \(X(i_1, i_2, \ldots, i_N)\) represents an element in \(X\), while \(X(:, i_2, \ldots, i_N)\) and \(X(:, i_3, \ldots, i_N)\) denotes a fiber along mode 1 and a slice along mode 1 and mode 2 respectively. Inner product of two tensors \(X\) and \(Y\) of the same size is given as \(\langle X, Y \rangle = \sum_{i_1, i_2, \ldots, i_N} X(i_1, i_2, \ldots, i_N) Y(i_1, i_2, \ldots, i_N)\) and the Frobenius norm can be obtained as \(\|X\|_F^2 = \langle X, X \rangle\). Notations \(tr(A)\) and \(\|A\|_s\) defines the trace and the nuclear norm of a matrix \(A\).

II. System model

We consider a multi-cell network with \(N_{BS}\) base stations, where each base station serves multiple users. Each user requests contents from a fixed library. Let the content library be indexed by the set \(F = \{1, \ldots, F\}\), where each content is assumed to be of equal size with 1 unit. In the time slot \(t \in \{1, \ldots, T\}\), the \(bth\) base station has a cache of size \(L_{BS}\), and the \(F \times 1\) vector \(c_{0}\) describes the status of cache, that is, if the \(fth\) content is cached, \(c_{0}(f) = 1\) (else 0 for not cached) with the cache size constraint \(c_{0}1_F = L_{BS}\). For fractional caching, where a portion of content is cached rather than the whole file, we have \(c_{0}(f) \in [0, 1]\) denoting the fraction of the \(fth\) content being cached, for each \(f \in F\). Before delivering the requested contents from users, a subset of popular contents are cached in the cache of the \(bth\) base station. Based on the requested contents, we define a direct hit to be the cache hit when the requested content is present in the
cache, while indirect hits are the hits in cache for which users choose as an alternative to the requested content based on the recommendations, when the requested content is unavailable in the cache. Each base station collects the data about the direct and indirect hits for each time slot.

Let $D_t = \{ D_{fib,t}, \forall f, i \in F, b = 1, \ldots, N_{BS}, j = t - \tau + 1, \ldots, t \}$ be a fourth-order tensor representing the aggregated data of number of direct and indirect content requests across all base stations and for previous $\tau$ time slots ($t - \tau + 1, \ldots, t$). The four dimensions of the tensor $D_t$ respectively represent the requested content’s index, indices of requested contents based on recommendations, base station index, and time slot. In other words, in the $t^{th}$ time slot at $b^{th}$ BS, the number of recommended requests for the $i^{th}$ file, when the $f^{th}$ content is primarily requested, is denoted as $D_{fib,t}$. Thus, the size of tensor is $I_1 \times I_2 \times I_3 \times I_4$, where $I_1 = I_2 = F$, $I_3 = N_{BS}$ and $I_4 = \tau$. Note that only few entries of the tensor $D_t$ can be observed, since the library is large and a few files are popular in a given time slot. Therefore, before processing this sparse tensor to derive the cache placement scheme, a tensor completion approach is essential. The following subsection describes the dynamics of the caching system.

A. Time slot

The structure of the time slot is shown in Figure 1. The first phase is the content placement (CPL) phase, where contents are placed in each BS’s cache based on the present information at base stations. The subsequent phase is the content delivery phase, where the content is delivered from the cache as per the requests from users. The next phase is dedicated for information exchange, where multiple base stations exchange the information about the number of hits and requests for better cache placements in the next time slot. For the CPL phase, the content placement strategy is chosen as a combination of two methods, that is, linear prediction of contents’ demands followed by most-popular caching (MPC) scheme. Since accurate prediction requires the data available for different contents, tensor completion problem is considered prior to performing prediction.

Moreover, in practice, the demands (number of requests for contents) are correlated across different contents and base stations, reducing the rank of the tensor $D$. Thus, an independent prediction for individual base station and for each file can cause performance degradation. To deal with these correlation issues, authors in literature considers in-network caching[57], joint reinforcement learning[8], etc. However, in this work, we focus on improving the caching performance using tensor completion methods.

B. Normalized cache hit rate and content placement

In literature [16], [58], to measure caching performance, several measures like cache hit (miss) rate, average success probability (ASP), etc have been considered. Improvements in these measures also relies on the prediction of demands. Cache hit rate at the $b^{th}$ base station in the $t^{th}$ time slot can be written as

$$H_{bt} = \frac{\sum_{i \in F} D_{fib,t} \cdot c_{bt}(f)}{\sum_{i \in F} D_{fib,t}},$$

which evaluates the cache placement policy.

Note that the number of requests at a BS is random with unknown distribution, and not available in advance. That is, the decision for the $f^{th}$ file is set based on the previous number of requests. The objective of caching is to find the best content placement strategy to maximize the hit rate above. If in $(t + 1)^{th}$ time slot the number of requests $D_{fib,t+1}$ is known in advance, we can choose $L_{BS}$ files with largest number of requests. On the other hand, when $D_{fib,t+1}$ is not known, it is natural to maximize the expected value given the previous information as

$$\arg \max_{c_{b,t+1}, \forall b} \mathbb{E}_D \left[ \sum_b H_{b,t+1} | D_t \right]$$

and

$$\arg \max_{c_{b,t+1}, \forall b} \mathbb{E}_D \left[ \sum_{f \in F} c_{bt}(f) \mathbb{E}_D \left[ \frac{\sum_{i \in F} D_{fib,t+1}}{\sum_{i \in F} D_{fib,t+1}} | D_t \right] \right],$$

in which $D_t$ denotes the conditional mean estimate of the normalized number of requests. This
estimate is the prediction of demands at \( t + 1 \), given the past \( \tau \) observations of demands until time slot \( t \). Note that given the prediction estimates, the solution for caching strategy includes the \( L_{BS} \) contents which have largest values in the estimate. To compute this estimate, the probability distribution of normalized requests must be known. However, the number of users’ requests are random and non-stationary with unknown distribution. Therefore, in the following, we obtain linear prediction estimation using least squares.

C. Linear prediction

Let \( \hat{D}_{fb,t} = \sum_{f \in F} D_{fb,m} \) denote the normalized number of requests, such that \( \sum_{f \in F} \hat{D}_{fb,t} = 1 \). To obtain the linear prediction, the normalized demands are assumed to evolve as a linear combination of demands in the temporal dimension as

\[
\hat{D}_{fb,t+1} \approx \sum_{m=1}^{M} c_{bm} \hat{D}_{fb,m}, \forall f \in F, \tag{3}
\]

where \( M \) is the order of prediction and \( c_{bm} \) are the prediction coefficients. These coefficients are obtained by solving least squares fit problem, given the previous \( \tau \) observations \( \hat{D}_{fb,j}, j = t - \tau + 1, \ldots, t \) for each \( b \) as

\[
\min_{c_{0:b} \forall I} \sum_{f \in F} \sum_{j=0}^{\tau - M - 1} \left( \hat{D}_{fb,t-j} - \sum_{m=1}^{M} \hat{D}_{fb,t-j-m} c_{bm} \right)^2 \tag{4a}
\]

subject to

\[
\sum_{m=1}^{M} c_{bm} \hat{D}_{fb,m} \geq 0, \forall f \in F \tag{4b}
\]

\[
\sum_{f \in F} \sum_{m=1}^{M} c_{bm} \hat{D}_{fb,m} = 1, \tag{4c}
\]

where the constraints provide the non-negative values. Let the prediction estimate be denoted as \( \hat{D}_{fb,t+1} = \sum_{m=1}^{M} c_{bm} \hat{D}_{fb,m} \). Mean based approach can be considered as a special case of linear prediction approach, i.e. \( c_{bm} = 1/d, \forall i \), \( i = 1, \ldots, d \). Since the data is sparse and correlated across files and base stations, it is difficult to directly obtain accurate predictions via these methods. Therefore, after filling the entries via tensor completion method, the above linear prediction can be used to find the future popularity estimates. For simplicity, in later sections, we recall this linear prediction method using the notation \( \{ \hat{D}_{fb,t+1}, \forall f, b \} \leftarrow LP \{ D_t \} \).

D. Tensor preliminaries: Circular unfolding and folding

To efficiently represent the information in higher order tensors, authors in [50, 59] defined a balance unfolding scheme based on circular unfolding. For an \( N \)-order tensor \( X \), the tensor circular unfolding matrix, denoted by \( \mathcal{X}_{(k,d)} \) of size \( I_k \times J_k = I_a I_{a+1} \ldots I_k \times I_{k+1} \ldots I_{a-1} \), can be written as

\[
\mathcal{X}_{(k,d)}(i_1,i_{a+1} \ldots i_k \times i_{k+1} \ldots i_{a-1}) = X(i_1,i_2,\ldots,i_N), \tag{5}
\]

where \( d \) is a positive integer and

\[
a = \begin{cases} k - d + 1, & d \leq k; \\ k - d + 1 + N, & d > k. \end{cases} \tag{6}
\]

The above unfolding with the given mode \( k \) and shift \( d \) provides the balanced unfolding. Note that the balance of the above unfolding depends on the shift \( d \). Similarly, the folding of a matrix \( X \) of size \( I_a I_{a+1} \ldots I_k \times I_{k+1} \ldots I_{a-1} \) along the mode \( k \) and shift \( d \) provides the tensor \( [X]_{(k,d)} \) of size \( I_1 \times \ldots \times I_N \).

E. Latent tensor norm with cyclic unfolding

The cyclic unfolding based latent tensor norm is defined for an \( N \)-order tensor \( X \) as follows [54, 55]

\[
\|X\|_{TR} = \min_{X_{1}+\ldots+X_{N}} \sum_{k=1}^{N} \|X_{k}(k,d)\|_{*}, \tag{7}
\]

where the minimum is over \( N \) tensors \( \{X_{k}\}_{k=1}^{N} \), and \( X_{k}(k,d) \) denotes the low-rank unfolding of \( X_{k} \) along the mode \( k \) with a given value of \( d \).

F. Tensor completion problem

Let \( T \in \mathbb{R}^{I_1 \times \ldots \times I_N} \) be an \( N \)-order sparse tensor with the observed entries. The location of entries in \( T \) is denoted by another indicator tensor \( I \), where \( I(i_1,i_2,\ldots,i_N) = 1 \) when \( T(i_1,i_2,\ldots,i_N) \neq 0 \), and 0, otherwise. The notation \( |I| \) and \( T(I) \) define the number of non-zeros in \( I \) and the entries of \( T \) for the corresponding non-zeros indices in \( I \), respectively.

The optimization problem for low-rank tensor completion using TR-latent nuclear norm is cast as

\[
\min_{X_{k} \forall k} \|X\|_{TR} \tag{8a}
\]

subject to

\[
X = \sum_{k=1}^{N} X_{k}, X(I) = T(I), \tag{8b}
\]

where \( X_{k} \) are component tensors and unfolded cyclically. To solve the above optimization, recently in [34, 53], Frank Wolfe algorithm is used to obtain parameters independent iterative procedure for tensor completion. However, the computation complexity is large due to optimum mode selection \( k \), which requires SVD of \( X_{k} \) along each of \( N \) modes (since \( k = 1, \ldots, N \)); and the iterative compression of basis matrices, where SVD, QR and a quadratic optimization is performed. In this work, we propose a procedure with much lower computational overhead and along with significantly better reconstruction error. The details are described in the following.

III. TENSOR COMPLETION ALGORITHM

A. Rank efficient modified FW algorithm

Under the Frank-Wolfe framework, the optimization problem in (8a) can be rewritten as

\[
\min_{X} F(X) \tag{9a}
\]

subject to \( \|X\|_{TR} \leq \beta, \tag{9b} \)
where $F(\mathcal{X}) = \frac{1}{2} \| \mathcal{X}(\mathcal{I}) - \mathcal{T}(\mathcal{I}) \|_F^2$, and $\beta > 0$. The constraint $\| \mathcal{X} \|_{TR} \leq \beta$ is also related to the rank constraint, that is, if the rank constraint is not specified, the rank of the solution can be high. However, for the proposed procedure, we will show that with the rank constraint specified, the above constraint can be relaxed; in other words, low rank solutions can be obtained with the specified $\beta$ limit.

The above optimization is solved via the gradient descent steps in the FW algorithm. It can be observed that the constraint $\mathcal{X} = \sum_{k=1}^N \mathcal{X}_k$ is also included in the TR-norm constraint. For the cyclic unfolding, we can write

$$\mathcal{X} = \sum_{k=1}^N \left[ \mathcal{X}_{k,(k,d)} \right]_{(k,d)}, \tag{10}$$

where the matrices $\mathcal{X}_{k,(k,d)}$ are of low-rank, say $R_k$. Given the rank constraint of overall TC problem (say $R$), all $\mathcal{X}_k$ must satisfy $\sum_{k=1}^N R_k \leq R$. Note that for notational simplicity, we omit the iteration index. The update of the gradient descent can be given as

$$\mathcal{X} \leftarrow \mathcal{X} - \gamma \mathcal{S}, \tag{11}$$

where $\gamma > 0$ and the tensor $\mathcal{S}$ represent the gradient of $F(\mathcal{X})$, $\nabla F = \mathcal{X}(\mathcal{I}) - \mathcal{T}(\mathcal{I})$. The efficient representation of $F(\mathcal{X})$ with a decomposition and the TR-norm constraint can be obtained as follows.

1) **Constraint linear optimization:** The problem of finding the tensor gradient of $F(\mathcal{X})$ to satisfy the constraint (11), can be expressed as

$$\mathcal{S} = \arg \max_{\| \mathcal{S} \|_{TR} \leq \beta} \langle \mathcal{S}, \nabla F \rangle,$$

where the objective is maximize the correlation between the above two tensors. Note that the objective function is linear. One possible solution is to choose $\mathcal{S} = \beta \frac{\nabla F}{\| \nabla F \|_F}$. However, we also need low-rank decomposition of $\mathcal{S}$ for the tensor completion problem. Therefore, we first find the optimum mode for cyclic unfolding, and then, leverage SVD for decomposition components. To find the optimum mode, we compare the first dominant singular value of the unfolded tensor $\nabla F$ along different modes, that is,

$$k^* = \arg \max_k \sigma_{max} \left[ (\nabla F)_{(k,d)} \right],$$

where the notation $\sigma_{max}(\mathbf{A})$ denotes the maximum eigenvalue of the matrix $\mathbf{A}$. Let the folded gradient have SVD as $(\nabla F)_{(k,d)} = \mathbf{U}_k \Sigma_k \mathbf{V}_k^T$, where the singular values are assumed in a descending order. For a low-rank tensor completion, rank is a constraint, say $r_k$. We will present how to obtain $r_k$ later. Therefore, we have the gradient solution as

$$\mathcal{S} = \beta \left[ \mathbf{U}_{k^*} (1 : r_{k^*}) \Sigma_k (1 : r_k) \mathbf{V}_{k^*}^T (1 : r_{k^*}) \right]_{(k^*,d)}, \tag{12}$$

which for a matrix $\mathbf{A}(1 : m, 1 : n)$ denotes the submatrix with entries belonging to the first $m$ rows and first $n$ columns of $\mathbf{A}$; and $\mathbf{A}(1 : n)$ is the submatrix with first $n$ columns of $\mathbf{A}$; $\mathbf{1}_n$ denotes $n \times 1$ vectors of ones.

**Remark (rank-1 updates):** For gradient representation $\mathcal{S}$, instead of choosing $r_k$ rank, rank-1 updates can be considered (for a suboptimal solution with larger overhead) as

$$\mathcal{S} = \beta \left[ \mathbf{U}_{k^*} (1) \mathbf{V}_{k^*}^T (1) \right]_{(k^*,d)}, \tag{13}$$

which is inefficient than as compared to (12) due to the fact that the structure of singular values $\Sigma_k$ is absent. Algorithm in [34], [53] gathers many such singular value (and values), and find the structure of singular values via the compression step.

**Remark (max-mode selection):** In the above step, the mode $k^*$ is selected based on maximum singular value. For an $I_1 \times \cdots \times I_N$ tensor, the singular value of an unfolding is maximum when its dimensions are minimum. If the $k^{th}$ unfolding has dimension $I_k \times J_k$, then the value $k^*$ can be selected as

$$k^* = \arg \min_k \min \left\{ I_k, J_k \right\},$$

which can significantly reduce the computational complexity of mode-selection, that is, the computations for $N$ singular values.

2) **Line search:** The line search problem is to find the step size, which can be expressed as

$$\gamma = \arg \min_{\gamma \geq 0} \| \{ \mathcal{X}(\mathcal{I}) - \gamma \mathcal{S} \} - \mathcal{T}(\mathcal{I}) \|_F^2 \tag{15a}$$

$$= \arg \min_{\gamma \geq 0} \bar{a} \gamma^2 - 2 \bar{b} \gamma \tag{15b}$$

$$= \max \left\{ \frac{\bar{b}}{\bar{a}}, 0 \right\}, \tag{15c}$$

where $\bar{a} = \| \mathcal{S}(\mathcal{I}) \|_F^2$, $\bar{b} = \langle \mathcal{X}(\mathcal{I}) - \mathcal{T}(\mathcal{I}), \mathcal{S}(\mathcal{I}) \rangle$, and the solution is obtained by differentiation. The max-operator arises due to $\gamma \geq 0$ constraint. With $\gamma$ and $\mathcal{S}$ obtained, the tensor $\mathcal{X}$ can be updated using the equation (11).

3) **Decomposition update:** Thanks to the decomposition, one does not need to store the whole tensor $\mathcal{S}$ or $\mathcal{X}$. We can store SVD components and step size as a representation of the update of the unfolded-component tensors. In other words, for the selected optimum mode-$k^*$, singular vectors along each mode are updated from, the update equation (11) can be simplified as

$$\mathcal{X} \leftarrow \mathcal{X} - \gamma \mathcal{S} \tag{16a}$$

$$= \sum_{k=1}^N \left[ \mathcal{X}_{k,(k,d)} \right]_{(k,d)} - \gamma \mathcal{S} \tag{16b}$$

$$= \sum_{k \neq k^*} \left[ \mathcal{X}_{k,(k,d)} \right]_{(k,d)} + \left[ \mathcal{X}_{k^*,(k^*,d)} \right]_{(k^*,d)} - \bar{\gamma} \mathbf{1}_1 \mathbf{1}_n^T \tag{16c}$$

where for a matrix $\mathbf{A}(1 : m, 1 : n)$ denotes the submatrix with entries belonging to the first $m$ rows and first $n$ columns of $\mathbf{A}$; and $\mathbf{A}(1 : n)$ is the submatrix with first $n$ columns of $\mathbf{A}$; $\mathbf{1}_n$ denotes $n \times 1$ vectors of ones.
\( k^* \)-th mode as

\[
\begin{align*}
U_{k^*} &\leftarrow \left[ U_{k^*}, -\tilde{U}_{k^*} (1 : r_{k^*}) \right], \\
V_{k^*} &\leftarrow \left[ V_{k^*}, \tilde{V}_{k^*} (1 : r_{k^*}) \right], \\
\Sigma_{k^*} &\leftarrow \begin{bmatrix}
\Sigma_{k^*} & 0 \\
0 & \frac{\gamma_2 \Sigma_{k^*} (1 : r_{k^*}, 1 : r_{k^*}) J_{k^*}}{\tilde{I}_{k^*} \Sigma_{k^*} (1 : r_{k^*}, 1 : r_{k^*}) J_{k^*}}
\end{bmatrix},
\end{align*}
\]

(17a)

(17b)

(17c)

(17d)

where the components \( U_k, \Sigma_k, V_k, R_k, k \neq k^* \) remains unchanged during this iteration. Regarding the update of \( r_k \), there are two constraints, rank of the unfolded matrix \( r_k \leq \min \{ I_k, J_k \} - R_k \), and the overall rank constraint

\[
\sum_{k=1}^N \bar{R}_k = R - \sum_{k=1}^N R_k.
\]

Thus, for the next iteration, the update for \( r_k \) can be written as

\[
r_k \leftarrow \min \left\{ I_k - R_k, \bar{J}_k - R_k, R - \sum_{i=1}^N R_i \right\},
\]

(18)

which also defines the stopping criteria, that is, the iterative procedure stops if \( r_k \) reaches 0.

Remark (Compression step): It can be seen that due to rank constraint \( r_k \), the above representation does not explode into many basis matrices. Thus, no-compression is required here, which significantly reduces the computational overhead, as compared to [34], [53]. Further, if the compression step is leveraged into the proposed algorithm, the reconstruction errors can be further reduced. However, for an online algorithm, this step is omitted.

4) Algorithm: The Algorithm 1 presents the proposed TC procedure, which combines the steps obtained in the above subsections. In addition to the input observed tensor \( T \), the method requires to specify the required rank \( R \) and the low-rank reconstruction error limit \( \beta \). After the initialization of variables \( X = 0, R_k = r_k = 0, U_k = \Sigma_k = V_k = 0, \forall k \), first the optimum mode \( k^* \) is selected and the corresponding unfolded mode SVD of \( \nabla F \) is computed. Based on the value of \( r_k \), the gradient representation \( S \) and the step size \( \gamma \) are calculated via rank-\( r_k \) truncated SVD. Subsequently, the rest of variables are updated including \( \bar{X}, U_k, \Sigma_k, V_k, \) and \( R_{k^*} \). For usage, the algorithmic procedure is denoted as \( \bar{X} \leftarrow TCA(T, R) \).

The value of \( r_k \) denotes the number of available dimensions in the \( k^{th} \) mode. The value \( r_k = 0 \) means either, the overall rank constraint \( R = \sum_k R_k \) is satisfied, or, mode-\( k \) rank is reached, i.e., \( R_k = \min \{ I_k, J_k \} \). The former constraint leads to the stopping criteria, while the latter adds the pruning step for the search space of optimum mode selection, that is, \( N \leftarrow N \setminus \{ k^* \} \).

Proposition 1. Given the rank constraint \( R \), the Algorithm 1 is independent of \( \beta \).

Proof: In the algorithm, the update equation (11) depends on the product \( \gamma \beta \). By the definition of \( \gamma \), we write

\[
\gamma = \frac{|X|}{\|S(X)\|_F \|S(I)\|_F}.
\]

Substituting the value \( S \) from (12) provides \( \beta \gamma = \text{constant} \), where the constant is specified via the singular values in the unfolding in \( k^* \)-mode of \( \nabla F \). In other words, \( \gamma \) adjusts itself according to \( \beta \) in each iteration, leading to \( \beta \)-independent algorithm. This is also verified via simulations. 

Algorithm 1: Rank efficient modified FW algorithm, TCA(T, R).

Input: \( T, \beta, R, d \).

Output: \( U_k, \Sigma_k, V_k, \forall k; \sum_{k=1}^N \| U_k \Sigma_k \Sigma_k^T (k, d) \|_F \leq \beta \).

1: Initialize \( X = 0, R_k = r_k = 0, U_k = \Sigma_k = V_k = 0, \forall k \).

2: Initialize \( N = \{1, \ldots, N\} \).

3: for \( n = 1, 2, \ldots, n_{\max} \) do

4: Set the tensor \( \nabla F = X = T - \nabla I \).

5: Obtain \( k^* = \arg \max \{ \Sigma_{k^*} \} \).

6: Get the SVD \( (\nabla F)_{k^*, d} = \tilde{U}_{k^*} \Sigma_{k^*} \tilde{V}_{k^*} \in \mathbb{R}^{I_k \times J_k} \).

7: Update \( r_k \leftarrow \min \{ I_k - R_k, J_k - R_k, R - \sum_{k=1}^N R_k \} \).

8: if \( r_k \leq 0 \) then

9: Break the loop.

end if

10: Compute \( S \) from (12).

11: Get step size \( \gamma \) from (15c).

12: Update \( X \leftarrow X - \gamma S \).

13: Update \( U_k, \Sigma_k, V_k \).

14: Update \( R_k = r_k + r_k \).

15: if \( r_k = 0 \) then

16: \( N \leftarrow N \setminus \{ k^* \} \).

17: end if

18: end if

19: end for

20: Return \( U_k, \Sigma_k, V_k, \forall k \).

5) Time and space complexity: Let \( X \) be an \( N \)-order tensor of dimension \( I \times \ldots \times I \). In the algorithm 1 rank-\( R \) SVD is computed for the compactly unfolded matrix of size \( I^d \times I^{N-d} \), which incurs the complexity \( O(I^{2d}I^{N-d}) = O(I^{N+d}) \).

Regarding space complexity for rank-\( R \) decomposition, \( I^d R + I^{N-d} R + R \) valued space is required for \( U_k, V_k, \Sigma_k, \forall k \), and \( |I| \) space for the observable tensor.

B. Online prediction and caching algorithm

Since the above tensor completion is fast, it can be used in an online manner for edge caching, as shown in the Algorithm 2. In this algorithm, the tensor completion and linear prediction methods are employed in the information exchange phase, whereas the cache placement phase is dedicated to placing the content according to the predicted number of normalized requests.

IV. Simulation Results

Simulations are performed on the MovieLens dataset [60], where fourth order tensors (\( N = 4 \)) are constructed from the movie ratings with dimensions \( F \times F \times N_{BS} \times \tau \), where \( F = 128, N_{BS} = 3 \), and \( \tau = 10 \). Each time slot entry is set by aggregating the rating for 30 days based on the timestamps given. \( M = 6 \)-th order prediction is performed. For tensor completion, values \( d = 1, \beta = 10^5, R = nN, n = 2, 4, 6, 8, 10, 12 \) are chosen. For edge caching, \( L_{BS} = 32 \)
Algorithm 2 Online prediction and caching algorithm.

Input: $\mathcal{D}_t, \forall t, R$.

Output: $c_{bt}, \forall b, t$

1: for $t = r, r + 1, \ldots$ do
2: \hspace{0.5em} IE phase: Observe the tensor $\mathcal{D}_t$.
3: \hspace{0.5em} Apply $\hat{X} \leftarrow TCA(\mathcal{D}_t, R)$.
4: \hspace{0.5em} Employ $\{\hat{D}_{fb,t+1}, \forall f, b\} \leftarrow LP(\hat{X})$.
5: \hspace{0.5em} CPL phase: Based on $\hat{D}_{fb,t+1}, \forall f$, obtain MPC placement $c_{bt}$, for each $b$.
6: \hspace{0.5em} CD phase: deliver contents from as per users’ requests.
7: end for

Figure 2. RSE versus the execution time (in MATLAB) for the proposed algorithm and the algorithm in [53], for $\beta = 10^5$ and $R = nN$, $\forall n = 2, 4, 6, 8, 10, 12$.

is chosen. The algorithm is compared for normalized reconstruction errors, defined as

$$RSE = \frac{||X(I) - T(I)||_F}{||T(I)||_F}, \tag{19}$$

which equals to 1 at the start of first iteration, since $X = 0$. Two prediction methods are chosen, that is, linear predictions with optimum coefficients (LP) and with equal coefficients (MP). The performance of edge caching is measured in terms of cache hit rate. Algorithms are run on a windows PC with Intel Xeon CPU E3-1230 v5 (3.40GHz) and 32GB RAM.

A. Convergence

Figure 2 plots the convergence in terms of RSE for the proposed algorithm, and the FW algorithm in [53] for $\beta = 10^5$ and $R = nN$, $\forall n = 2, 4, 6, 8, 10, 12$. It can be observed that the proposed algorithm outperforms significantly as compared to the one in [53]. It achieves less RSE in less iterations (e.g. at $R = 2N = 32$, $RSE \approx 10^{-3}$ for Yu et al, and $RSE < 10^{-5}$ for the proposed one), and each iteration executes in less time as well (total time is 0.2 seconds versus 200 seconds approximately).

B. Effect of latent factors $R$ and $\beta$

To observe the effect of the factor $\beta$, Figure 3 plots the RSE with respect to $\beta$ at $R = 6N$ for all the three methods, while Figure 4 shows with respect to the rank $R$ with the fixed $\beta = 10^5$. It can be seen that for all values of $\beta$, the proposed algorithm provides much less RSE than for CPD and the method in [53]. Moreover, the proposed algorithm is invariant to changes in $\beta$, as shown in Proposition 1. Regarding the plots versus rank- $R$, the proposed method performs significantly better at lower ranks. As the rank is increased, the performance difference between the proposed and the FW algorithm decreases. For sufficiently high rank $R > 45$, FW provides minutely better performance, where the difference in RSE is the order of $10^{-6}$.
observed tensor. It will significantly reduce the time and space of large tensor is both time and space intensive. Therefore, in the order of 10^10 as compared to the CP decomposition. The algorithm provides significantly better cache hit rate (75%) in hierarchical caching networks,” IEEE Transactions on Communications, vol. 64, no. 5, pp. 2092–2103, 2016.

5. B. Blaszczyzyn and A. Giovanidis, “Optimal geographic caching in cellular networks,” in IEEE International Conference on Communications (ICC), 2015, pp. 3358–3363.

6. B. Serbetci and J. Gsoleing, “Optimal geographical caching in heterogeneous cellular networks with nonhomogeneous helpers,” arXiv preprint arXiv:1710.09626, 2017.

7. A. Papazafeiropoulos and T. Ratnarajah, “Modeling and performance of uplink cache-enabled massive mimo heterogeneous networks,” IEEE Transactions on Wireless Communications, vol. 17, no. 12, pp. 8136–8149, 2018.

8. A. Sadeghi, F. Sheikhholeslami, and G. B. Giannakis, “Optimal and scalable caching for 5G using reinforcement learning of space-time popularities,” IEEE Journal of Selected Topics in Signal Processing, vol. 12, no. 1, pp. 180–190, 2018.

9. N. Garg, M. Sellathurai, V. Bhatia, and T. Ratnarajah, “Function approximation based reinforcement learning for edge caching in massive mimo networks,” IEEE Transactions on Communications, 2020.

10. J. Yin, L. Li, H. Zhang, X. Li, A. Gao, and Z. Han, “A prediction-based coordination caching scheme for content centric networking,” in WOCC, 2018, pp. 1–5.

11. W.-X. Liu, J. Zhang, Z.-W. Liang, L.-X. Peng, and J. Cai, “Content popularity prediction and caching for ICN: A deep learning approach with SDN,” IEEE Access, vol. 6, pp. 5075–5089, 2018.

12. H. Nakayama, S. Ata, and I. Oka, “Caching algorithm for content-oriented networks using prediction of popularity of contents,” in IFIP/IEEE International Symposium on Integrated Network Management (IM), 2015, pp. 1171–1176.

13. Y. Zhang, X. Tan, and W. Li, “Ppc: Popularity prediction caching in icn,” IEEE Communications Letters, vol. 22, no. 1, pp. 5–8, Jan 2018.

14. Y. Hang, S. M. A. Kazimi, K. Thar, M. G. R. Alam, and C. S. Hong, “Cache aware user association for wireless heterogeneous networks,” IEEE Access, vol. 7, pp. 3472–3485, 2019.

15. J. Wu, Y. Zhou, D. M. Chiu, and Z. Zhu, “Modeling dynamics of online video popularity,” IEEE Transactions on Multimedia, vol. 18, no. 9, pp. 1882–1895, Sep. 2016.

16. Y. Jiang, M. Ma, M. Bennis, F. Zheng, and X. You, “User preference learning-based edge caching for fog radio access network,” IEEE Transactions on Communications, vol. 67, no. 2, pp. 1268–1283, Feb 2019.

17. Y. Wang, M. Ding, Z. Chen, and L. Luo, “Caching placement with recommendation systems for cache-enabled mobile social networks,” IEEE Communications Letters, vol. 21, no. 10, pp. 2266–2269, 2017.

18. P. Sermpezis, T. Giannakas, T. Spyropoulos, and L. Vigneri, “Soft cache hits: Improving performance through recommendation and delivery of related content,” IEEE Journal on Selected Areas in Communications, vol. 36, no. 6, pp. 1300–1313, 2018.

19. L. E. Chatzieleftheriou, M. Karaliopoulos, and I. Koutsopoulos, “Jointly optimizing content caching and recommendations in small cell networks,” IEEE Transactions on Mobile Computing, vol. 18, no. 1, pp. 125–138, 2019.

20. P. Cheng, C. Ma, M. Ding, Y. Hu, Z. Lin, Y. Li, and B. Vucetic, “Localized small cell caching: A machine learning approach based on rating data,” IEEE Transactions on Communications, vol. 67, no. 2, pp. 1663–1676, 2019.

21. K. Guo and C. Yang, “Temporal-spatial recommendation for caching at base stations via deep reinforcement learning,” IEEE Access, vol. 7, pp. 58 519–58 532, 2019.

22. W. He, Y. Su, X. Xu, Z. Luo, L. Huang, and X. Du, “Cooperative content caching for mobile edge computing with network coding,” IEEE Access, vol. 7, pp. 67 695–67 707, 2019.

C. Cache hit rate

To evaluate the performance of the proposed method for edge caching scenario, Figure 5 shows the normalized cache hit rates averaged over time steps for three different methods with the cache size of 32. The results are averaged over 220 time slots, wherein each time slot a tensor completion problem is solved and future popularity is predicted using linear prediction (LP) and mean-prediction (MP) methods. It can be observed that compared to CPD, the CHR improvement is around 75.8%, and compared to FW algorithm, the improvement is present at higher ranks e.g., R = 8N. Mean prediction and linear prediction perform approximately similar for CHR. As the rank of completion is increased, minor increments can be observed in the CHR performance.

V. Conclusion

In this paper, we have proposed an improved time complexity based modified FW algorithm based on gradient descent, which has been shown to significantly outperform in term of computational complexity and performance. This algorithm needs only a few iterations, less than the specified rank Rank. Further, for the wireless edge caching, we have formulated content recommendation and prediction problem into a tensor completion. Using the completed tensor, we have employed linear prediction methods to obtain future popularities. For this application, it is shown via simulations that after completion, the algorithm provides significantly better cache hit rate (75%) as compared to the CP decomposition.

For the edge caching, the library size is typically large of the order of 10^4. Performing tensor completion on such a large tensor is both time and space intensive. Therefore, in future, we shall explore to find the independent blocks in the observed tensor. It will significantly reduce the time and space requirements and better tensor decomposition can be found for the remaining block tensors as compared to the full tensor.

![Figure 5. Average cache hit rate for two linear prediction methods with three different completion methods for MovieLens dataset with R = 2N, 4N, 6N ranks and β = 10^6.](image-url)
