A Learning-Based Approach for Proactive Caching in Wireless Communication Networks

Yuyang Wang, Yun Chen, Haibo Dai, Yongming Huang, and Luxi Yang
School of Information Science and Engineering, Southeast University, Nanjing, China
Email: {wangyuyang, yunchen, hbdai, huangym, lxyang}@seu.edu.cn

Abstract—**Proactive caching is a promising technology in 5G wireless networks. Small-cell base stations (SBS) can cache popular contents to assist the macro base station, and proactive caching are considered to cope with the weak backhaul links of SBSs. However, obtaining popular contents and making the optimal caching strategy may be challenging. In this paper, a novel learning-based approach is proposed, in which regularized singular value decomposition (RSVD)-based collaborative filtering (CF) is used to estimate the content popularity and transfer learning (TL) is adopted to improve the estimation accuracy. Then considering the interaction between users and SBSs, a distributed iterative algorithm is designed to make a caching strategy with the goal to maximize the number of users who can be served by neighboring SBSs. Experiments have been conducted to evaluate the performance of the proposed algorithms and simulation results demonstrate the effectiveness of our learning-based approach for proactive caching.**

I. Introduction

With the development of mobile communications, mobile smartphones and social networks, a wide variety of online services are provided through mobile terminals, which leads to explosive and rapid growth of mobile data [1]. As a result, the excessive demand for data is draining the limited spectrum resources of wireless transmissions, especially the wireless links between base stations and users, and the wireless backhual links between base stations and the core network. To cope with this problem, a promising solution is to cache popular contents at the edge of mobile networks.

Considering the large amount of mobile data, more users and more diverse user requirements in wireless networks, proactive caching is proposed in [2], which tracks users' requesting frequency and analyzes historical data to predict the popularity of contents. In real scenario of proactive caching, the popularity of content files can not be known perfectly, so various approaches are employed to estimate the popularity profile. In [3]–[5], several learning-based approaches to estimate the popularity profile for devising caching mechanisms have been investigated. Besides, caching without prior knowledge of the popularity distribution is considered for femtocell networks in [6], where it is shown that distributed caching is an NP-hard problem and approximation algorithms are proposed for video content delivery. In [7], the theoretical analysis of the implications of learning the popularity profile on the training time is studied to achieve an offloading loss which is close to the optimal policy.

In proactive caching, the aforementioned references have some problems. Firstly, estimating the content popularity matrix which is assumed to be largely unknown is very challenging due to the data sparseness and cold-start problems [8]. Considering the inefficient performance of supervised machine learning which is used to estimate the popularity matrix, a transfer learning (TL)-based approach are used to improve the estimation accuracy by transferring knowledge from other domains [4] [5]. However, the TL caching approaches mentioned above are based on the classical collaborative filtering (CF) learning techniques, which do not have superior performance and can be further improved by regularized singular value decomposition (RSVD). Secondly, in the existing approaches, the proactive caching decision is made by storing the most popular content greedily until no storage space remains after obtaining the estimated content popularity matrix. However, the popular caching strategy cause several neighbouring small-cell base stations (SBS) to cache the same popular contents, which clearly results in waste of cache resources. Considering such a condition, the caching strategy can be optimized if the interactions between the edge of networks are taken into account [9].

In this paper, a learning-based approach deriving from TL and RSVD-based CF techniques for proactive caching is proposed, and our contributions are summarized as follows. Firstly, in order to estimate the content popularity matrix, traditional CF technique is ameliorated by RSVD. Secondly, considering the data sparseness and cold-start problems, transfer learning is used to improve the estimation accuracy, which is done by transferring and learning hidden features of popularity matrix from the source domain, such as a social network or a device-to-device (D2D) network [10], to the target domain. Thirdly, after obtaining the estimated content popularity matrix, a distributed iterative algorithm is proposed to make the optimal proactive caching strategy. Finally, compared with other traditional methods, experiments are presented to show the effectiveness of the proposed approach. Generally, with our proposed caching scheme, not only can the heavy traffic load be relieved, as well the request latency can be decreased, which will definitely result in better user experience.

The rest of this paper is organized as follows. Section II describes the system model and problem statement under consideration. In Section III, a caching approach using TL and RSVD-based CF techniques is presented in detail to estimate the content popularity matrix, then a distributed iterative algorithm is proposed to develop a caching strategy.

The performance of the proposed approach is evaluated by experiments in Section IV and finally concluding remarks are provided in Section V.

II. System Model and Problem Statement

In this section, we present the system model followed by the main problem addressed in the paper.

A. System Model

A heterogeneous cellular network is considered, including N users, one macro base station (MBS) and M SBSs. The set of users is denoted by $\mathcal{N} = \{1, 2, \ldots, N\}$, and the set of SBSs is denoted by $\mathcal{M} = \{1, 2, \ldots, M\}$. The set of users and SBSs are randomly distributed in the MBS's service scope. Each user independently requests a file from the content server in service provider. The set of files is denoted by $\mathcal{F} = \{1, 2, \ldots, F\}$, where the size of each content file f is B bits. The popularity of content files is specified by the distribution $\mathcal{P} = \{P_1, P_2, \ldots, P_F\}$, which is well modeled by the Zipf-like distribution [11]:

$$P_f = \frac{1/f^{\alpha}}{\sum_{i=1}^{F} 1/i^{\alpha}}, \tag{1}$$

where α is the exponent factor which characterizes the distribution and reflects different content popularities. The distribution mentioned above describes a content popularity in the ordered case. For simplicity, the distribution $\mathcal{P}$ is assumed to be stationary across time. Therefore, the content popularity matrix is given by $\mathbf{P} \in \mathbb{R}^{N \times F}$, where each entry P_{nf} represents the probability that user n requests content f. In order to simulate the delivery when the user requests the content, we assume that each SBS has a finite cache size C and caches contents from the set $\mathcal{F}$. Cache size C means the SBS can cache up to C files, of which each length is B bits.

In the modeled cellular network, if a user requests a content cached by a nearby SBS, then the SBS serves the request, otherwise the request is served by the MBS. The relationships among the users, SBSs and contents are modeled as follows. We define a cache matrix $\mathbf{X} \in \mathbb{R}^{F \times M}$, where entry x_{fm} returns 1 if SBS m caches the content f and 0 otherwise, and define a serve matrix $\mathbf{Y} \in \mathbb{R}^{M \times N}$, where entry y_{mn} returns 1 if SBS m can serve user n and 0 otherwise. If a request that user n needs content f can be served by a nearby SBS, then $\max_{m \in \mathcal{M}}\{x_{fm} y_{mn}\} = 1$, otherwise $\max_{m \in \mathcal{M}}\{x_{fm} y_{mn}\} = 0$.

Now, we present a simple communications protocol to determine which SBSs a specific user can be served by, and these SBSs are defined as neighboring SBSs. It is supposed that a SBS is located at a, a user is located at b. If $||a-b|| < R$, where R is the radius of SBS, the connection between the user and the SBS is established. Hence, the serve matrix $\mathbf{Y}$ can be known.

B. Problem Statement

In a typical wireless communication network, the MBS deliver a content file to a user by the backhaul link. This results in backhaul congestion during peak traffic hours. To alleviate this problem, caching the most popular files at the SBSs proactively is proposed. Then, the request from a user can be served directly by one of the neighboring SBSs. Therefore, the caching strategy should be made optimally. Estimating the content popularity matrix $\mathbf{P}$ is of high importance in proactive caching. Since $\mathbf{P}$ is assumed to be largely unknown and the number of users and files is extremely large, traditional CF or TL techniques lead to poor performance. Thus, we come up with a better learning-based approach in the estimate.

Then, based on the complete estimated matrix $\mathbf{P}$, an optimal caching decision (cache matrix $\mathbf{X}$) should be made. Instead of using greedy algorithm to store the most popular files, we establish an optimization problem as follows:

$$\begin{aligned} &\max_{\mathbf{X}} \sum_{n=1}^{N} \sum_{f=1}^{F} P_{nf} \max_{m \in \mathcal{M}} \{x_{fm} y_{mn}\} \\ &\text{s.t.} \quad \sum_{f=1}^{F} x_{fm} \leq C, \forall m \in \mathcal{M}, \end{aligned} \tag{2}$$

where the objective function means the expected number of users who can be served by neighboring SBSs. Therefore, an algorithm is designed to solve the optimization problem.

III. Estimating the Popularity Matrix and Making Caching Strategy

In this section, we propose a caching approach using TL and RSVD-based CF techniques to estimate the content popularity matrix, then a distributed iterative algorithm is designed to make an optimal caching strategy.

A. Estimating the Popularity Matrix

To estimate the popularity matrix, which is assumed to be largely unknown, RSVD-based CF is a classical approach to learn from a sparse matrix and make a estimate of the complete matrix. In this section, we improve the objective function of RSVD-based CF, and then use TL to increase the estimation accuracy.

1) RSVD-based CF to Estimate: The proposed RSVD-based CF caching procedure is composed of training and prediction parts. In the training part, the goal is to constantly optimize the estimate of popularity matrix $\mathbf{P}$, where every SBS builds a model based on the available information regarding users' preferences/ratings for files. The entry in the sparse popularity matrix $\mathbf{P}$ is defined as P_{ij}, and we transform $\mathbf{P}$ into the following equivalent form: $\mathbf{R} = \{(i, j, r) : r = P_{ij}, P_{ij} \neq 0\}$. The three elements of each row represent the user identity i, the file identity j, and the user's ratings r for the file, which record all known information of popularity matrix. To estimate the unknown entries of $\mathbf{P}$, we use RSVD to decompose the matrix and construct an estimate of the popularity matrix with a rank of k: $\mathbf{P} \approx \mathbf{N}^T\mathbf{F}$, where the two factor matrices are $\mathbf{N} \in \mathbb{R}^{k \times N}$ and $\mathbf{F} \in \mathbb{R}^{k \times F}$. The physical significance of these two factor matrices can be explained as that each user and file have k features, each row of the matrices is the eigenvector of corresponding user or file. Then the objective function of the optimization problem can be established by the sum of squares of errors between the predicted values and the true values, and the optimization problem is given by:

$$\min_{i,j\in\mathbf{P}} \sum_{(i,j)\in\mathbf{P}} (P_{ij} - n_i^T f_j)^2, \tag{3}$$

where the sum is over the (i,j) user/file pairs in the training set, and n_i and f_j represent the i-th and j-th columns of $\mathbf{N}$ and $\mathbf{F}$ respectively. However, if data in the training set is trained according to the above formula, the results of estimation will be over-fitting, which will lead to poor performance of the training model. In order to avoid over-fitting, we add a regularization parameter λ to the objective function, which is used to balance the regularization and fitting in data training process. Since all the values in n_i and f_j are variable and we do not know which variables will result in over-fitting problems, all of them are balanced, which refers to RSVD.

The initial popularity matrix of our algorithm (ie, the input training matrix) is $\mathbf{R}$, where the entry r_{ij} corresponding to P_{ij} is the known rating data, and the estimated value is defined as $\hat{r}_{ij} = n_i^T f_j$. For convenience, we denote the error between the true value and the estimated value as $e_{ij} = r_{ij} - \hat{r}_{ij}$. Because the user's rating on the file depends not only on the relationship between the user and the file, but also on the user's and the file's own features, a baseline predictor is added to the estimated value, which is denoted as $\hat{r}_{ij} = \mu + b_i + b_j + n_i^T f_j$, where μ is the average of all the ratings, b_i stands for the quality of each user i relative to μ, and b_j stands for the quality of each file j relative to μ, both b_i and b_j are also balanced with a regularization parameter λ. Accordingly, the optimization problem is improved and can be defined as follows:

$$\min_{i,j\in\mathbf{R}} \sum_{(i,j)\in\mathbf{R}} (r_{ij} - \mu - b_i - b_j - n_i^T f_j)^2 + \lambda(b_i^2 + b_j^2 + |n_i|^2 + |f_j|^2). \tag{4}$$

Then we can use a stochastic gradient descent algorithm to optimize (4) and the estimate of the popularity matrix is made. The optimized RSVD-based CF is proposed above. However, to cope with the sparsity problem, the proactive caching approach can be improved by employing a TL-based approach.

2) TL to Improve the Estimate: The proposed TL-based approach is to utilize the knowledge obtained from users' interactions with a social network (termed the source domain), and transfer the knowledge from source domain to help learn the users request pattern (target domain) by cleverly combining samples from the source domain and the target domain. The prior information can be obtained from social networks with D2D interactions, which is considered to follow a D2D content distribution. We suppose that there are N^s users and F^s files in the source domain, and the popularity matrix of source domain is $\mathbf{P}^s$ (or $\mathbf{R}^s$). Therefore, by judiciously combining the samples from source domain and target domain, we define the TL domain, where there are N^{tl} users and F^{tl} files, and the popularity matrix is $\mathbf{P}^{tl}$ (or $\mathbf{R}^{tl}$). When using TL, we also establish the two factor matrices $\mathbf{N} \in \mathbb{R}^{k\times N^{tl}}$ and $\mathbf{F} \in \mathbb{R}^{k\times F^{tl}}$. Then, the optimization problem is given by

$$\begin{aligned}\min_{i,j\in\mathbf{R}^{tl}} &\sum_{(i,j)\notin\mathbf{R}^s} (r_{ij} - \mu - b_i - b_j - n_i^T f_j)^2 \\ &+ \lambda(b_i^2 + b_j^2 + |n_i|^2 + |f_j|^2) \\ &+ \delta \sum_{(i,j)\in\mathbf{R}^s} (r_{ij} - \mu - b_i - b_j - n_i^T f_j)^2 \\ &+ \lambda(b_i^2 + b_j^2 + |n_i|^2 + |f_j|^2),\end{aligned} \tag{5}$$

where δ is used to balance the information from target domain and source domain. Having the above objective function, we need to optimize it by training, and a stochastic gradient descent algorithm is used. Specifically, we take the objective function's partial respect to b_i, b_j, n_i and f_j, and these four variables is changed with the negative gradient. Therefore, the four variables are updated as follows:

$$b_i = \begin{cases} b_i + \gamma(e_{ij} - \lambda b_i), & (i,j) \notin \mathbf{R}^s \\ b_i + \delta\gamma(e_{ij} - \lambda b_i), & (i,j) \in \mathbf{R}^s, \end{cases} \tag{6}$$

$$b_j = \begin{cases} b_j + \gamma(e_{ij} - \lambda b_j), & (i,j) \notin \mathbf{R}^s \\ b_j + \delta\gamma(e_{ij} - \lambda b_j), & (i,j) \in \mathbf{R}^s, \end{cases} \tag{7}$$

$$n_i = \begin{cases} n_i + \gamma(e_{ij} f_j - \lambda n_i), & (i,j) \notin \mathbf{R}^s \\ n_i + \delta\gamma(e_{ij} f_j - \lambda n_i), & (i,j) \in \mathbf{R}^s, \end{cases} \tag{8}$$

$$f_j = \begin{cases} f_j + \gamma(e_{ij} n_i - \lambda f_j), & (i,j) \notin \mathbf{R}^s \\ f_j + \delta\gamma(e_{ij} n_i - \lambda f_j), & (i,j) \in \mathbf{R}^s. \end{cases} \tag{9}$$

In the training described by the algorithm above, we need to obtain the deviation between the current estimate and the true value. For the measure of the error, we use the root mean square error (RMSE) [12]. Similar to $\mathbf{R}$, we suppose that the test matrix is $\mathbf{R}^{test}$ and the number of test values T_E is equal to the number of rows in $\mathbf{R}^{test}$. Therefore, the evaluation metrics is given by

$$RMSE = \sqrt{\sum_{(i,j,r_{ij})\in R} (r_{ij} - \hat{r}_{ij})^2 / T_E}. \tag{10}$$

The related pseudocode is presented in Algorithm 1, which uses RSVD-based CF and TL techniques to estimate the popularity matrix and the evaluation metrics RMSE is also given.

B. Making Caching Strategy

After obtaining the estimate of the popularity matrix, the optimal caching strategy should be made based on the estimate, by solving the optimization problem (2). Firstly, the caching strategy $\mathbf{X}_m$ of every SBS m is the m-th columns of $\mathbf{X}$, and the strategy set is denoted by $\mathcal{X}_m = \{\mathbf{X}_m\}$, where $\mathbf{X}_m = (x_{1m}, x_{2m}, \ldots, x_{Fm})^T, x_{fm} \in \{0,1\}$. Then, we define the utility function of every SBS m as $U_m(\mathbf{X}) = \sum_{n\in\mathcal{L}_m} \sum_{f=1}^{F} P_{nf} \max_{m\in\mathcal{M}} \{x_{fm} y_{mn}\}$, where $\mathcal{L}_m$ represent the set of users that SBS m can serve. In order to describe the optimal caching strategy, we define that best strategy

Algorithm 1 Popularity Estimate Algorithm

```
Require: $R$, $R^s$, $R^{test}$
Ensure: $P$, $rmse$
1: function LOADFILEANDINITIAL($R, R^s, R^{test}$)
2:     Load $R, R^s, R^{test}$
3:     Create arrays $b_i, b_j, n_i, f_j; RateMatrix, \mu \leftarrow R, R^s$
4:     for every element in $b_i, b_j, n_i, f_j$ do
5:         $b_i, b_j \leftarrow 0; n_i, f_j \leftarrow rand()/10$
6:     end for
7: end function
8: function TRAIN($\gamma, \lambda, nIter$)
9:     $Rmse, RNum, r_{ij} \leftarrow 0; LRmse \leftarrow 1000$
10:    for $n = 1 \rightarrow nIter$ do
11:        $Rmse \leftarrow 0, RNum \leftarrow 0$
12:        for $i = 1 \rightarrow UserNum$ do
13:            for $j = 1 \rightarrow IterNum$ do
14:                $r_{ij} \leftarrow \mu + b_i[i] + b_j[j] + n_i[i] \cdot f_j[j]$
15:                $e \leftarrow RateMatrix[i][j] - r_{ij}$
16:                Update $b_i[i], b_j[j], n_i[i], f_j[j]$in Eq.(6-9)
17:                $Rmse \leftarrow Rmse + e^2, RNum++$
18:            end for
19:        end for
20:        $Rmse \leftarrow \sqrt{Rmse/RNum}$
21:        if $Rmse > LRmse$ then $break$
22:        end if
23:        $LRmse \leftarrow Rmse, \gamma \leftarrow 0.9\gamma$
24:    end for
25:    $P \leftarrow \mu + b_i + b_j + n_i \cdot f_j$
26:    $rmse \leftarrow \sqrt{\sum (R^{test} - P)^2 / T_E}$
27:    return $P$,$rmse$
28: end function
```

set (BSS) is the strategy set that maximize the utility of SBS m given the strategies of the rest of SBSs, $\mathbf{BSS}_m = \arg\max_{\mathbf{X}_m \in \mathcal{X}_m} U_m(\mathbf{X}_m, \mathbf{X}_{-m})$. Furthermore, to cache files as many as possible, we define the strategy set that the cache of SBS m is full as $\mathbf{Fu}_m$.

In order to obtain the optimal solution of caching strategy, we suppose that every SBS exchanges information with neighbors to improve its strategy, so a distributed iterative algorithm is designed as shown in Algorithm 2. In the initialization, every SBS caches C content files. In each iteration, a set of non-interacting SBSs is selected, calculating the current utility functions and finding their best strategy sets independently, and then, the new strategy of the SBSs can be selected accordingly. Finally, the new utility functions and caching strategy are updated.

IV. Experiments and Results Analysis

In this section, simulation experiments are performed, and we provide results analysis to demonstrate the effectiveness of the proposed algorithms.

We consider a (1000m $\times$ 1000m) area with the SBSs and users randomly and independently distributed in the area. The parameter values used in our calculations are as follows: $M = 10$, $N = 100$, $F = 100$, $C = 10$, $\alpha = 1$, $\gamma = 0.05$, $\lambda = 0.15$, $k = 16$, $R = 300$m, $B = 1$Mb. We randomly generate the entire popularity matrix in target domain according to Zipf distribution, from where the training set $\mathbf{R}$ is randomly selected with 5% of ratings. Similarly, matrix in source domain is randomly generated according to D2D content distribution, from where $\mathbf{R}^s$ is randomly selected with 10% of ratings. An interaction graph between SBSs and users is shown in Fig. 1.

In the stage of estimating the popularity matrix, we compare the proposed Algorithm 1 with the approach without TL. The comparison chart between the two approaches regarding to evaluation metrics RMSE is shown in Fig. 2, where RMSE reflects the estimation accuracy of popularity matrix. Fig. 2 shows that with the increase of user-file scale, both RMSEs are reduced, which indicates that the estimated matrix is closer to the true value. More importantly, the performance of the proposed algorithm with TL is clearly better than the approach without TL. Therefore, the effectiveness of Algorithm 1 in estimating the popularity matrix is demonstrated.

To verify the overall performance of the proposed caching

Algorithm 2 Decentralized Cache Algorithm

```
1: Initialize $i \leftarrow 0$; $\forall m \in \mathcal{M}, \mathbf{X}_m(i) = (1, \ldots, 1, 0, \ldots, 0)$
2: for $i = 1 \rightarrow MaxIterNum$ do
3:     Randomly select a set of SBSs $\mathcal{H}(i)$
4:     for every SBS $m \in \mathcal{H}(i)$ do
5:         Calculate $U_m(\mathbf{X}_m(i), \mathbf{X}_{-m}(i))$
6:     end for
7:     for every SBS $m \in \mathcal{H}(i)$ do
8:         Calculate $\mathbf{BSS}_m$
9:         if $\mathbf{X}_m(i) \notin \mathbf{BSS}_m$ then
10:            $\hat{\mathbf{X}}_m(i) \leftarrow \mathbf{BSS}_m \cap \mathbf{Fu}_m$
11:        end if
12:        if $\mathbf{X}_m(i) \in \mathbf{BSS}_m$ then
13:            $\hat{\mathbf{X}}_m(i) \leftarrow \mathbf{BSS}_m \setminus \mathbf{X}_m(i)$
14:        end if
15:        Calculate $U_m(\hat{\mathbf{X}}_m(i), \mathbf{X}_{-m}(i))$, update $\mathbf{X}_m(i)$
16:    end for
17: end for
```

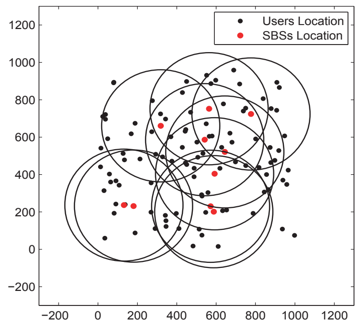

Fig. 1. The distribution of SBSs and users

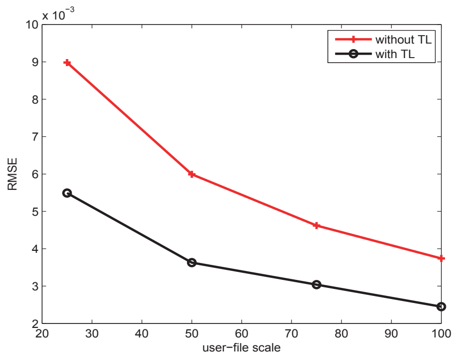

Fig. 2. RMSE versus the user-file scale with scale 100 representing that the number of users and files are both 100

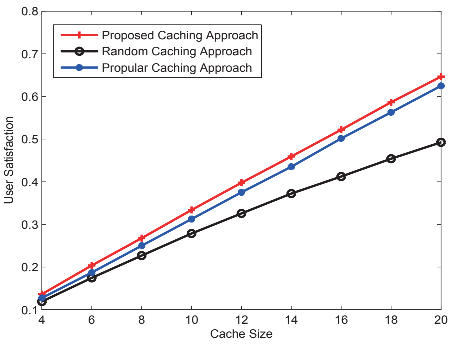

Fig. 3. Users' satisfaction versus the cache size of SBSs

approach, we compare the approach with two other approaches: a) the random caching approach, in which contents are cached uniformly at random; b) the popular caching approach, in which the content popularity matrix is estimated via Algorithm 1, and the most popular contents are stored accordingly. In the simulation model, users' satisfaction represented by the ratio of users served by neighboring SBSs is the main comparison metric.

Fig. 3 shows the impact of the cache size of SBSs. The users' satisfaction increases with the cache size of SBSs for all approaches. When the cache size of SBSs is larger, users can request more files from SBSs so the probability of being served by neighbouring SBSs also becomes larger, and we expect that all the users can be served by SBSs when the cache size is large enough. Our proposed approach performs better than the popular caching approach and the random caching approach, and the superiority of our proposed approach is demonstrated.

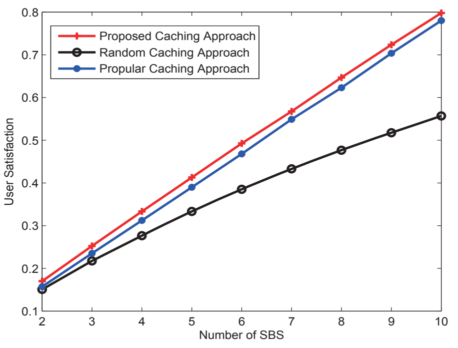

Fig. 4. Users' satisfaction versus the number of SBSs

Fig. 4 shows the impact of the number of SBSs. The users' satisfaction increases with the number of SBSs for all approach. The users can be served by more SBSs when the density of SBSs becomes larger, so users will have higher probability to acquire content files from neighbouring SBSs, and it is expected that all the users can be served by SBSs when the SBSs is dense enough. The proposed approach performs better than the popular caching approach and the random caching approach, which verifies the effectiveness of the proposed algorithms.

V. Conclusion

In this paper, we propose a novel learning-based approach for proactive caching. In the estimating stage, the proposed approach optimize the traditional CF by RSVD and use TL to improve the estimation accuracy by ingeniously transferring the prior information from social networks. Then, in the caching decision-making stage, we consider the interaction between users and SBSs by establishing the utility function of every SBSs, and a distributed iterative algorithm is designed to make a caching strategy based on the purpose to maximize the expected number of users who can be served by neighboring SBSs. Finally, experiments are conducted to evaluate the performance of the algorithms, and simulation results demonstrate the effectiveness of our learning-based approach for proactive caching.

References

[1] Cisco, "Global mobile data traffic forecast update, 2013–2018," *white paper*, 2014.

[2] 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, pp. 82–89, Aug 2014.

[3] A. Sengupta, S. Amuru, R. Tandon, R. M. Buehrer, and T. C. Clancy, "Learning distributed caching strategies in small cell networks," in *Proc. IEEE Int. Symp. Wireless Commun. Syst.*, pp. 917–921, Aug 2014.

[4] E. Bastug, M. Bennis, and M. Debbah, "Anticipatory caching in small cell networks: A transfer learning approach," in *1st KuVS Workshop on Anticipatory Networks*, 2014.

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

[6] 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, pp. 8402–8413, Dec 2013.

[7] B. N. Bharath, K. G. Nagananda, and H. V. Poor, "A learning-based approach to caching in heterogenous small cell networks," *IEEE Trans. Commun.*, vol. 64, pp. 1674–1686, April 2016.

[8] J. Lee, M. Sun, and G. Lebanon, "A comparative study of collaborative filtering algorithms," *arXiv preprint arXiv:1205.3193*, 2012.

[9] H. Dai, Y. Huang, and L. Yang, "Game theoretic max-logit learning approaches for joint base station selection and resource allocation in heterogeneous networks," *IEEE J. Sel. Areas Commun.*, vol. 33, pp. 1068–1081, June 2015.

[10] H. Dai, Y. Huang, R. Zhao, J. Wang, and L. Yang, “Resource optimization for device-to-device and small cell uplink communications underlaying cellular networks,” *IEEE Trans. Veh. Technol.*, vol. PP, no. 99, pp. 1–1, 2017.

[11] J. Llorca, A. M. Tulino, K. Guan, and D. C. Kilper, “Network-coded caching-aided multicast for efficient content delivery,” in *Proc. IEEE Int. Conf. Commun.*, pp. 3557–3562, June 2013.

[12] W. Pan, E. W. Xiang, and Q. Yang, “Transfer learning in collaborative filtering with uncertain ratings.,” in *AAAI*, vol. 12, pp. 662–668, 2012.