IJCNN 2019. International Joint Conference on Neural Networks. Budapest, Hungary. 14-19 July 2019

Adaptive Edge Caching based on Popularity and Prediction for Mobile Networks

Li Li
School of Computing and Information Systems
The University of Melbourne
Melbourne, Australia
lli10@student.unimelb.edu.au

Sarah Erfani
School of Computing and Information Systems
The University of Melbourne
Melbourne, Australia
sarah.erfani@unimelb.edu.au

Chien Aun Chan
Department of Electrical and Electronic Engineering
The University of Melbourne
Melbourne, Australia
chienac@unimelb.edu.au

Christopher Leckie
School of Computing and Information Systems
The University of Melbourne
Melbourne, Australia
caleckie@unimelb.edu.au

***Abstract*—Edge caching in mobile networks can improve users' experience, reduce latency and balance the network traffic load. However, edge caching requires suitable strategies for determining what files to pre-fetch at which cell and at what time. Due to the heterogeneity of users' content preferences and mobility, caching based only on popularity has limitations. Considering that cells located in different places have different predictability, in this paper, we propose an adaptive edge caching algorithm based on content popularity as well as the individual's prediction results to provide an optimal caching strategy, aiming to maximize the cache hit rate with acceptable file replacement cost. A heuristic optimization strategy based on genetic algorithms is presented, along with a prediction model based on an improved Markov model for each user according to the historical data. In the model, similar users are clustered based on their behavior patterns. We evaluate our algorithm on a simulation dataset as well as a 3-week real-life dataset from China Mobile. The results show that our optimal caching strategy can improve the cache hit rate compared with other methods, especially when the storage capacity is small and the similarity in content requests of users is low.**

***Index Terms*—mobile edge caching, cache deployment optimization**

I. Introduction

Content caching at the edge of the mobile network has attracted increasing research attention recently. Caching at the wireless network edge (close to users) can help reduce latency, balance network traffic load, and improve users' experience [1]. However, since the storage capacity for each cell is limited, it is important to design appropriate caching strategies to balance the trade-off between the quality of users' experience, and the limited storage capacity. In this paper, we focus on the problem of finding a strategy for caching at the edge of the network.

To design an appropriate caching strategy, we need to answer the question: what contents should be pre-fetched, at which cells and when? This is a challenging task, due to the following factors: (i) *Limited storage size*: Since the storage space in the edge node is limited, it is not possible to cache all the contents. Therefore, only those content items that can make the most profit (maximum hit rate/minimum cost) should be selected to be cached. (ii) *Users' mobility*: Users in the network may frequently be handed off from one cell to another. (iii) *Users' heterogeneous preferences*: The probability that each content item is requested by a specific user during a certain period can differ among individuals.

A simple caching strategy is to choose the most popular contents at the network edge. The limitation of this method is that the real popularity of the contents in the network is unknown and users' behavior (movement and content requests) in the network can be dramatically different, in which case popularity-aware caching may not be effective. A recent study [2] shows that the behavior of users in the network, including content, location, and mobility, affects the performance of edge content caching. In [3], the authors also noted that statistical patterns of content requests both in aggregated form and on a per-user basis should be considered in the content deployment problem.

Although there has already been some research on mobile users' behavior prediction [4], [5], [6], the prediction of users' behavior may not be easy. For example, in a city center, people move in various directions in a nondeterministic manner. Their behavior may be not predictable, leading to a low prediction accuracy. This will reduce the performance gain of edge caching and introduce extra costs, such as the energy consumed in the backhaul and edge cells due to ineffective content placement. An important property of cells in this content is their predictability in terms of users' movements. The predictability of a cell is the observed likelihood of being correct when we predict that a user will move into a given cell next. For example, if the prediction model predicts that 100 users will next move into a given cell, but only 60 actually do so, then we consider the predictability of that cell

978-1-7281-2009-6/$31.00 paper N-19458.pdf

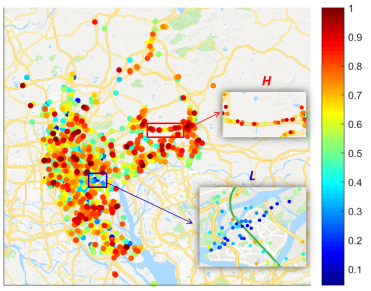

Fig. 1. Averaged prediction accuracy of the top 2,000 most visited cells within one city in southern China. Area H denotes an area with high prediction accuracy, and area L indicates an area with low prediction accuracy.

is 0.6. In Fig. 1, using the prediction model we propose in this paper, the average prediction accuracy of the top 2,000 most visited cells within a city in southern China is shown. We can see that different cells may have different levels of predictability. For those cells in area L, the prediction accuracy for users' behavior is low. Whereas for cells located along a main road (area H), the prediction accuracy is relatively high. This indicates that we should not totally rely on the prediction results when developing the caching strategy.

Therefore, we propose to consider not only the prediction results from the prediction model, but also the predictability of the cells. The basic idea is that, for cells with low predictability, we would prefer to cache the most popular files; whereas in other cells with high prediction accuracy, we would prefer to cache files according to the results from our prediction model. According to this, we propose to construct a proactive caching strategy considering two aspects: the popularity and the prediction results of users' location and content requests. In particular, we propose a formal model to optimize these two aspects of the proactive model.

To summarize, our main contributions are:

- We formalize the edge caching strategy problem in cellular networks, to maximize the hit rate over the whole network under a replacement constraint. An adaptive caching strategy based on general popularity and personalized predictions is also proposed, and a heuristic solution based on genetic algorithms is provided for solving the optimization problem.
- We propose a Markov based model for the prediction of users' behavior in terms of movement and requests, which is inspired by [4]. To overcome the "cold-start" limitation of Markov models when a new cell is visited, a clustering method is also proposed to find users who share similar behavior patterns.
- Simulations are provided to evaluate the performance of our proposed algorithm. We also test our algorithm on a real-life dataset from China Mobile. The results show that our algorithm outperforms all the existing schemes, especially when the cache capacity is low and the distribution of content popularity is skewed.

The remainder of the paper is organized as follows. Section II provides a review of the related work. Section III introduces the model of the system and our adaptive edge caching strategy. In Section IV, a heuristic method based on genetic algorithms is proposed to solve our proposed optimization model. The prediction model is presented in Section V. The details of our experiments on simulation data as well as real life data and a discussion of the results of our methods are shown in Sections VI and VII. Section VIII concludes our work and proposes some future directions.

II. Related Work

The edge caching problem has attracted extensive attention recently due to its advantages for reducing latency, as well as relieving the heavy overhead burden on the network backhaul [7]. The current research mainly focuses on issues like the caching architecture design, content deployment and delivery [8]. This paper falls within the problem of content deployment and delivery.

Two common caching schemes are Least Frequent Used (LFU) and Least Recently Used (LRU). They are simple but not robust methods, since their performance can be reduced by the heterogeneity of users in the network. To fully exploit the edge resources, as mentioned in several works, e.g., [9], [10], [11], popularity-based cache placement schemes have been widely deployed to maximize the hit rate. For example, in [9], the authors found that content popularity varies at different locations. A linear prediction model is built to estimate the future hit rate of contents at different locations, and according to the results, contents that can have the highest hit rate will be cached. The authors in [11] proposed a model called PopCaching, which can predict the popularity and make the content caching decision on-line to maximize the hit rate. Naifu et al. [12] proposed a linear prediction model to estimate the future content requests based on historical data, and then an on-line cache replacement optimization model was built based on the future popularity prediction. Although these methods considered the change of popularity, none of them considered the characteristics of the mobile users' behaviors in the network.

Recently, some works [2], [3], [13] have started to consider the movement and request characteristics of the mobile users in the wireless network. The basic idea is that if the users' trajectories and requests can be predicted based on historical data, then the cell can proactively cache appropriate files and the user can download the pre-fetched files along their trajectory. To achieve this, as mentioned in [3], we should not only consider the content popularity, but also the user's preference, which can be predicted and play a key role in the design of caching. In [13], the authors also pointed out that taking user mobility into consideration is critical for caching design in content centric wireless networks. Ge et al. [2] provided a thorough study on the behavior of mobile video users. A geo-collaborative caching strategy was also provided. They divided the cache storage into two parts according to the fractions of two types of users: single-location users and multi-location users. Dong et al. [14] argued that the common

assumption that the preferences are identical among all users is not true in practice, and showed that optimizing caching policy with individual users' preferences is beneficial. However, in [14], the user preferences were assumed known a priori, which actually cannot be known perfectly in advance in real-life applications.

In this paper, we not only propose to integrate a personalized prediction model into the edge caching problem, but also provide an optimization model that can adaptively decide the preferences for popularity and prediction results. To the best of our knowledge, this is the first algorithm that considers the predictability of different cells when developing the cache deployment strategy.

III. System Model and Problem Formulation

In this section, we provide the description of the system model, and formulate the optimization problem.

A. Assumptions

Cells in the network: There are K cells in the network which can deploy caches, denoted as $C = \{c_1, c_2, ..., c_K\}$. However, the storage of a cell for caching is limited.

Proactive caching: We assume that the network is refreshed at fixed time slots. At the beginning of each time slot the selected content files are pre-fetched at each cell so that user's content request can be processed with reduced latency. A time period with L time slots is represented as $T = \{t_1, t_2, ..., t_L\}$.

Files in the network: Suppose that there are N content files that may be requested in the network, denoted as $F = \{f_1, f_2, ..., f_N\}$. These files have been sorted by popularity, which is $p_F = \{p_1, p_2, ..., p_N\}$, where $p_i \geq p_j \geq 0$, $\forall i \leq j$, and $\sum_{f=1}^{N} p_f = 1$. We also assume that all files have the same unit size and the content popularity distribution does not change during the time period we considered.

Users in the network: We assume that the maximum number of mobile users in the network is M, and the mobile users in the network we are considering can leave or enter the network at any time. The user set is represented as $U = \{u_1, u_2, ..., u_M\}$. We also assume that at each time slot a user is served by its closest cell.

B. Caching Strategy

To improve the user experience at an acceptable cost, we plan to deploy different caching strategies for cells with different predictability levels. For the cells with a higher prediction accuracy, we prefer to pre-cache documents according to the users' prediction results, whereas in those cells where mobile users' behavior are unpredictable, we consider pre-caching files according to their popularity, as shown in Fig. 2. Based on historical data of the mobile users, a prediction model is built (refer to Section V), and then an optimization model is proposed to obtain the preferences for popularity and personalized prediction results.

Therefore, we can represent the caching strategy as $\Phi = [\phi_1, ..., \phi_c, ..., \phi_K]$, where $c = 1, ..., K$, is the percentage of space that is used for popularity caching for cell c. For each cell, if we represent the files selected by popularity as F^1, and the files selected by prediction results as F^2, they should satisfy:

$$F^1, F^2 \subset F \ and \ F^1 \cap F^2 = \emptyset. \tag{1}$$

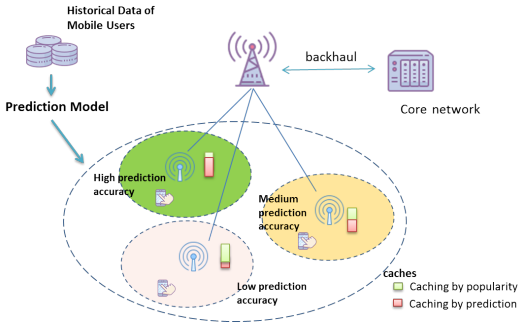

Fig. 2. An example of the adaptive caching framework.

Suppose the storage of each cell is limited and fixed, denoted as s. Then the space limits for popularity caching and personalized caching of cell c would be $s \cdot \phi_c$ and $s \cdot (1 - \phi_c)$, respectively. This means F^1 and F^2 should satisfy:

$$|F^1| = s \cdot \phi_c, \quad |F^2| = s \cdot (1 - \phi_c), \tag{2}$$

where $|\cdot|$ denotes the number of files. If $\phi_c = 1$, it means that the files in cell c are cached solely according to the general popularity. If $\phi_c = 0$, it means that the files in cell c are cached by the personalized prediction. (2) indicates that there are $s + 1$ discrete values for selection in ϕ_c, i.e., $\phi_c \in \{0, 1/s, 2/s, \ldots, 1\}$.

The caching strategy for each cell can be described as the following two steps:

S1: First, the file f_i is cached according to the top popularity,

$$p_{f_i} \geq p_{f_j}, \quad \forall f_i \in F^1, f_j \in F - F^1. \tag{3}$$

The cached file set F^1 contains the files of the highest popularity.

S2: Then we select files from the rest of the file set $F - F^1$ according to our prediction results. Files with the highest prediction confidences are selected. If we use $\hat{a}^t_{ufc} \in \{0, 1\}$ to represent the prediction result of whether or not user u will request file f in cell c at time t, then the predicted request frequency for each cell can be calculated as:

$$\hat{q}^t_{fc} = \sum_{u=1}^{M} \hat{a}^t_{ufc}. \tag{4}$$

We select the file f_i based on the top prediction frequency:

$$\hat{q}^t_{f_i c} \geq \hat{q}^t_{f_j c}, \quad \forall f_i \in F^2, f_j \in F - F^1 - F^2. \tag{5}$$

The cached files in F^2 have the highest prediction confidence. All the selected files in $F^1 \cup F^2$ will be cached in cell c at time t.

Based on these caching strategies, we can obtain the caching deployment of a file in a cell at a given time. We use a binary

variable b_{fc}^t to represent whether or not the file f is cached in cell c at time t,

$$b_{fc}^t = \begin{cases} 1 & file\ is\ cached \\ 0 & file\ is\ not\ cached \end{cases}. \quad (6)$$

The value of $\hat{a}_{ufc}^t$ is known a priori based on the prediction model, and the value of $\hat{q}_{fc}^t$ can be obtained accordingly. Given a value of ϕ_c, then the caching strategy b_{fc}^t of cell c can be achieved based on the popularity and the prediction.

C. Problem Formulation

Caching replacement: The replacement cost for refreshing caching files at time t is defined as the number of files that are cached at time slot t but not at the previous time slot $t-1$,

$$R_c^t = \begin{cases} \sum_{f=1}^{N} b_{fc}^t \cdot (1 - b_{fc}^{t-1}), & t \geq 2 \\ 0, & t = 1 \end{cases}. \quad (7)$$

We suppose that the replacement cost for all the cells at time t should be less than an upper threshold $Cost_r$,

$$\sum_{c=1}^{K} R_c^t \leq Cost_r. \quad (8)$$

Hit rate: A cache hit occurs when the requested data can be found in its cache. We use $a_{ufc}^t \in \{0, 1\}$ to represent whether the user u requests file f in cell c at time t or not. Then the average hit rate of cached data in cell c is:

$$H_c^t = \frac{\sum_{f=1}^{N} \sum_{u=1}^{M} a_{ufc}^t \cdot b_{fc}^t}{\sum_{f=1}^{N} \sum_{u=1}^{M} a_{ufc}^t}, \quad c \in C, t \in T. \quad (9)$$

The overall average hit rate of the network during the entire time period is:

$$\mathbf{H} = \frac{\sum_{t=1}^{L} \sum_{c=1}^{K} \sum_{f=1}^{N} \sum_{u=1}^{M} a_{ufc}^t \cdot b_{fc}^t}{\sum_{t=1}^{L} \sum_{c=1}^{K} \sum_{f=1}^{N} \sum_{u=1}^{M} a_{ufc}^t}. \quad (10)$$

Optimization model: In order to provide consistent Quality of Experience (QoE) to users, in our model, the objective is to maximize the hit rate of the whole network during a certain time period. Therefore, the optimization problem is:

$$\begin{aligned} max: \quad & \mathbf{H}(\Phi) \\ s.t. \quad & (2), (8). \end{aligned}$$

The value of the objective function (10) can be calculated given a specific caching strategy, i.e., the value of Φ. In Algorithm 1, we provide the pseudo-code for calculating the objective function $\mathbf{H}(\Phi)$ in (10) given $\Phi = [\phi_1, ..., \phi_c, ..., \phi_K]$. Apart from Φ, the other three inputs are the popularities of files which are assumed to be known a priori, users' real requests and user's predicted requests obtained by our prediction model, which will be discussed in detail in Section V.

The optimal values are sought under two constraints, which are the division of the storage capacity constraint in (2) and the total replacement cost constraint in (8). The storage capacity constraint, which determines the proportion of cell storage used for content selected by popularity versus content selected by prediction, is directly related to the value of ϕ_c. As for the total replacement cost constraint, the optimal state of each cell, which is calculated independently of other cells, may not satisfy the total replacement cost constraint in (8). Therefore, in calculating the optimal caching strategy, the states of all cells are essentially interconnected and need to be considered simultaneously.

For each cell c, there are $s+1$ possible discrete states for ϕ_c, i.e., $\phi_c \in \{0, 1/s, 2/s, \ldots, 1\}$. The non-linearity of the total replacement cost constraint in (8) makes a fast algorithm with reduced complexity infeasible. The complexity of the problem, which includes K cells of $s+1$ discrete states, is $(s+1)^K$. The total replacement cost constraint is non-linear due to the non-linearity in (7), and thus the optimization problem is non-linear and cannot be solved in polynomial time. This combinatorial problem makes it intractable to find the global optimal solution because of its exponentially large search space and the inclusion of a highly non-linear constraint. To solve the non-linear optimization problem, we propose to use stochastic optimization, such as Genetic Algorithms (GA), to find a probably local optimum. GA are considered as an appropriate choice for solving the current problem since the optimization model is highly non-linear and discontinuous [15]. High-quality solutions for this optimization can be obtained based on bio-inspired operators including mutation, crossover and selection. The details are shown in the next section.

Algorithm 1 Calculate the value of the objective function

Input: caching strategy $\Phi = [\phi_1, ..., \phi_c, ..., \phi_K]$, popularity of all the files, users' real request a_{ufc}^t and predicted request $\hat{a}_{ufc}^t$

Output: objective function $\mathbf{H}(\Phi)$

```
for each time t → 1, L do
  for each cell c → 1, K do
    F^1 = ∅, F^2 = ∅;
    // cache by popularity
    extract top s · φ_c files to F^1 from F based on the popularity;
    // cache by prediction
    calculate the predicted request frequency q̂^t_fc using (4) based on the predicted user request â^t_ufc;
    extract top s·(1−φ_c) files to F^2 from F − F^1 based on the predicted request frequency;
    for each file c → 1, N do
      b^t_fc ← 0
      if c ∈ F^1 ∪ F^2 then
        b^t_fc ← 1
      end if
    end for
  end for
end for
calculate H using (10)
return hit rate H
```

IV. Heuristic solution

Genetic Algorithms (GA), which were first proposed in [16], are a heuristic search and optimization technique inspired by natural selection, the process that drives biological evolution. In this paper, a genetic algorithm is applied as the optimization search method, which is described in Algorithm 2. Basically, there are four steps.

S1: Generate a set of initial solutions (represented by chromosomes) as the first population. Each chromosome Φ, is a possible solution to the optimization problem. It stores an array of values, and each value (also called gene) represents the $\phi \in \{0, 1/s, 2/s, \ldots, 1\}$ value of each mobile cell.

S2: Calculate the fitness score $f(\Phi)$ of each chromosome Φ. The fitness function f determines how fit a chromosome is, which considers the objective function $\mathbf{H}(\Phi)$ in (10) and the penalty of the constraint in (8).

$$f = \mathbf{H}(\Phi) + g\left(max\left\{\sum_{c=1}^{K} R_c^t - Cost_r, 0\right\}\right), \quad (11)$$

where $g(\cdot)$ is the penalty function as proposed in [17].

S3: Generate n_P chromosomes as the successor population to replace the source population using the following steps.

a) *Selection* In the selection phase, we select two chromosomes Φ_1 and Φ_2 from the source population. The idea of the selection phase is to select the fittest individuals and let them pass their genes to the next generation. Individuals with high fitness have more chance to be selected for reproduction. In this paper, Roulette Wheel Selection [18] is adopted.

b) *Crossover* We apply a crossover operator to Φ_1 and Φ_2 to generate a child chromosome Φ_{child} by combining the genetic information of two parent chromosomes. We adopt a k-point crossover operator [19], which incorporates k randomly picked crossover points from the parent chromosomes. The crossover rate r_c refers to the fraction of the next generation that are produced by crossover.

c) *Mutation* We apply an extended Power mutation [20], which is suitable for solving integer optimization problems, to produce Φ'_{child} by changing r_m of the gene values in Φ_{child}. Mutation helps introduce diversity within the population and prevent premature convergence.

d) Add the new child chromosome Φ'_{child} to the successor population.

S4: Evaluate the termination criterion. Calculate the fitness score $f(\Phi)$ in (11) for n_P chromosomes. The algorithm terminates (converges) if it cannot produce new offspring that are significantly improved from those of the previous generation, and the chromosome with the highest fitness score is returned as the solution. If the criterion is not met, return to *S3*.

Algorithm 2 Genetic Algorithm

Input: population size n_P, crossover points k, crossover rate r_c, mutation rate r_m
Output: solution Φ

```
randomly generate n_P chromosomes as the first population;
evaluate the fitness score f(Φ) of each chromosome Φ in the first population;
while the termination condition is not satisfied do
    for i → 1, n_P do
        selection select two chromosomes Φ_1 and Φ_2 in the parent population according to the fitness scores;
        crossover apply k-point crossover to Φ_1 and Φ_2 to obtain the child chromosome Φ_child with a crossover rate r_c;
        mutation mutate a portion of r_m genes in Φ_child to generate Φ'_child;
        append Φ'_child to the successor population;
    end for
end while
return Φ with the highest fitness score
```

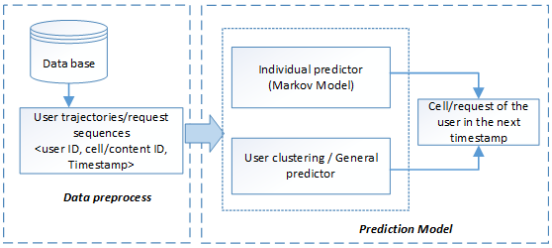

Fig. 3. The framework of our Markov-based prediction model.

V. Markov-based Prediction Model with User Clustering

A user's movements can be characterized by a Markov stochastic process, which assumes that the next state is only influenced by the current state and a fixed number of previous states. In this section, we develop our Markov-based prediction model for user's mobility and preferences. The framework is shown in Fig. 3. Basically, there are two steps: (1) In the first step, users' movements and content request behavior (sequences) are extracted from the database; (2) In the second step, to address the cold start problem a prediction model with an improvement by user clustering is proposed for more robust prediction. Here we discuss the details about the second step, which includes a second-order Markov model improved by user clustering. The direct benefit of the user clustering algorithm is an improved prediction accuracy of the user's next position/file request.

A. Markov Model

Here a second-order Markov model is adopted. In the training process, the transition probability of one user traveling

from cells (c_{i_1}, c_{i_2}) to cell c_j is denoted as $p_{(c_{i_1},c_{i_2}),c_j}$, which can be obtained by:

$$p_{(c_{i_1},c_{i_2}),c_j} = p(c_j|(c_{i_1}, c_{i_2})) = N_{ij}/N_i, \tag{12}$$

where N_{ij} is the occurrence of the user moving from cells (c_{i_1}, c_{i_2}) to cell c_j, and N_i is the frequency of cells (c_{i_1}, c_{i_2}) being visited by the user, (c_{i_1}, c_{i_2}) are the previous two cells visited by the user consecutively. The values of N_{ij} and N_i are calculated by the statistics of users in the data preprocessing.

Then in the prediction part, at time slot $t-1$, given the user's current cell c_{t-1} and previous cell c_{t-2}, we can predict the cell that the user will visit in the next time slot t to be:

$$\hat{c}_t = \arg\max_c \ p_{(c_{t-2},c_{t-1}),c}. \tag{13}$$

Similarly, we can make the prediction for users' file request behavior in the next time slot $\hat{f}_t$. According to the predicted location $\hat{c}_t$ and file request $\hat{f}_t$, we can obtain whether or not the user will request file f in cell c at time t, i.e., the value of $\hat{a}^t_{ufc}$ as used in (4).

B. User Clustering

To overcome the problem that the Markov prediction may not work if the current cell has never occurred in the user's historical mobility behavior, we propose to utilize the historical mobility transition matrix from other similar users [4]. The basic idea is that, when the prediction cannot be made by using the user's own historical mobility pattern, we use the mobility pattern of users who have similar patterns to predict the next cell of this user.

Here we choose to use iVAT [21], [22] for user clustering. iVAT is a useful tool for visual assessment of clustering tendency [23], which displays a reordered dissimilarity matrix as a gray-scale image with a modified version of Prim's minimal spanning tree algorithm. There are three steps included as follows:

S1: *Feature selection.*

a) Each mobile user can be represented as a vector, $B_1 = [t_1, ..., t_j, ..., t_K]$, where K is the number of cells, and t_j is the number of time stamps the user has spent in cell c_j.

b) The visiting sequence is treated as a string, and the bigrams can be extracted for each user. Then a vector of $K(K-1)$ bigrams can be obtained, which is denoted as B_2.

c) After normalizing the two vectors B_1 and B_2 separately, we concatenate these two vectors together as the feature vector for each user.

S2: *Dissimilarity measurement*. Cosine distance is selected here as the dissimilarity measurement between two users.

S3: The dissimilarity matrix obtained from *S2* is fed to iVAT for clustering.

Fig. 4 shows an example of the iVAT clustering result for 114 users in a small district. We can clearly see that seven clusters are detected for these users. Here notice that not all the users can be clustered together, since some users may behave quite differently from others. For those distinct users, the proposed improvement of the user clustering does not work.

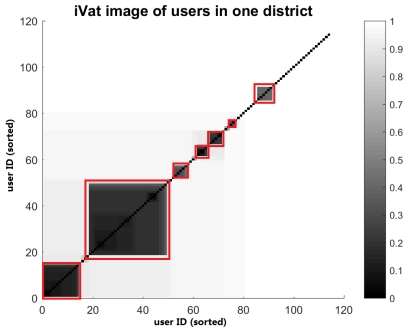

Fig. 4. An example of user clusters of one district based on iVAT clustering.

The procedure of the prediction model is summarized as follows: we first extract the previous two cells of the user; if the user never visited these two cells consecutively before, the prediction model obtained by similar users will be applied, otherwise the individual prediction model will be applied for prediction.

VI. Simulation

In this section, we describe our simulator to evaluate the effectiveness of our algorithm in terms of the cache hit rate compared with three benchmarks.

A. Simulation Setup

We simulate the mobility patterns of 2,500 users on a simulation area with 64 (8×8) cells using the smoothly truncated Levy walk algorithm [24], which simulates users' mobility pattern using the preset probability for travel distance, pause length, and change in direction. Users' content requests are modeled based on Poisson arrivals. The number of all the content files in the network is assumed to be 500,000. The popularity distribution of the files follows a Zipf distribution, which has been used in many existing works [10].

$$p_F \sim Zipf(\alpha, N),$$

where α is the distribution skewness parameter, and N is the total number of files in the network. The larger α is, the higher the skewness of the distribution. The total runtime of the simulation is around 1.5 hours.

B. Benchmarks

We compare the performance of our method with the following benchmarks:

- **Least Frequently Used (LFU)**: When the cache is full, discard the least recently used items first.
- **Least Recently Used (LRU)**: Purge the item with the lowest reference frequency when the cache is full.
- **Popularity Caching (PC)**: The most popular contents over the whole network will always be cached at each

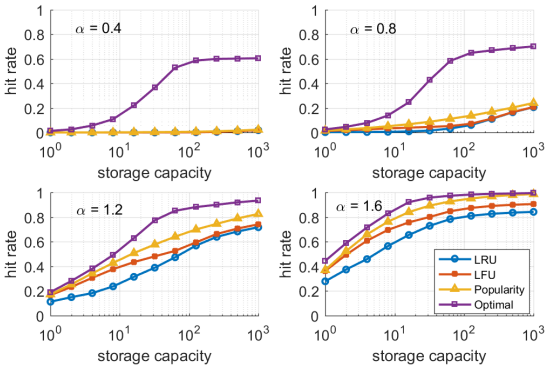

Fig. 5. Averaged cache hit rate under different cache storage capacity ($s = 10^0, ..., 10^3$) and four different α values ($\alpha = 0.4, 0.8, 1.2, 1.6$).

cell. Here we suppose the popularity of contents is known and not changing during the time period we are considering.

C. Performance Comparison

Here, we divide the cells randomly into 4 groups, and set the content request prediction accuracy as 30%, 50%, 70% and 90% respectively. We investigate the performance of our proposed method with varying cache capacity ($s = 10^0, ..., 10^3$) and with various skewness parameters ($\alpha = 0.4, 0.8, 1.2, 1.6$) in terms of the averaged cache hit rate. Fig. 5 shows the results during a high network load of 90%, which means approximately 2,250 users out of 2,500 are active in the simulation area. We can extract the following insights: (1) As the storage capacity increases, the hit rates of all the algorithms grow, but our optimal method grows the fastest. This indicates that our method is better suited for resource limited networks. (2) The performance of all the methods increases when α turns larger, and LRU, LFU and Popularity Caching are very sensitive to the skewness of the content popularity distribution. (3) In the experiments, our optimal solution always outperforms the other benchmark methods, and the performance of the popularity based method is better than LFU and LRU. Especially, when α is low, the hit rate of our method is much higher than the other three methods. For example, when $\alpha = 0.4$, the hit rate of our method can achieve 0.6, while the hit rates of the other three methods are quite low, less than 0.05. This means that our method has better performance when the mobile users have a lower similarity in content requests. (4) Our proposed method can effectively reduce the cache storage requirement. For example, if the target cache hit rate is 0.5, when $\alpha = 0.4$ and $\alpha = 0.8$, our optimal caching needs a cache capacity of around 100 files, while other benchmark methods require more than 1,000 files.

D. Parameter Selection and Performance of GA

There are four parameters in GA, i.e., population size n_P, number of crossover points k, crossover rate r_c and mutation rate r_m. Here, we mainly analyze the effect of n_P and r_c. For the mutation rate r_m, normally a small value of $0.5\% - 1\%$ is suggested, and here we select $r_m = 1\%$. The number of crossover points is selected as $k = 5$. The replacement cost constraint $Cost_r$ is selected to be $0.3sK$, where s is the cache storage of a cell and K is the total number of cells. Given different values of n_P and r_c with $\alpha = 0.8$ and $s = 100$, Table I compares three metrics including the number of generations needed to converge to the final state, the averaged hit rate and the maximum hit rate of the final generation. To ensure consistent results with less randomness, given a set of n_P and r_c values, the averaged results of the GA running 20 times rather than a single run are listed in Table I. A population size of 10 cannot preserve the population diversity, and the converged value is much lower than that of $n_P = 20$. As the population size n_P further increases from 20 to 30, the averaged hit rate and the maximum hit rate remain essentially unchanged. A variation in the crossover rate r_c between 0.6 and 0.8 has basically no effect on the results, and $r_c = 0.7$ leads to the fastest convergence. Thus, we select $r_c = 0.7$ in GA for the case studies presented in this paper. A sample of the GA convergence curve is shown in Fig. 7.

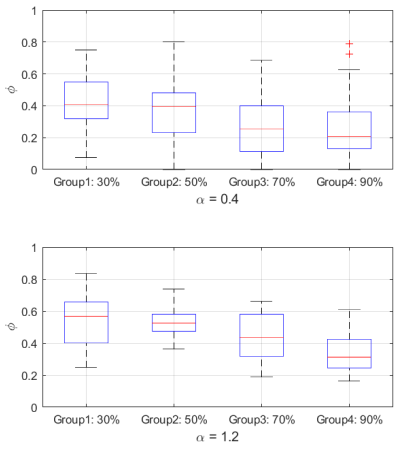

Fig. 6. The ϕ values of cell groups with different values of prediction accuracy. For example, in Group1 the prediction accuracy is set as 30%.

TABLE I
PARAMETER ANALYSIS OF GA

n_P	r_c	Generations	Average hit rate	Maximum hit rate
10	0.7	21	0.6144	0.6288
20	0.7	45	0.6506	**0.6555**
30	0.7	62	0.6516	0.6557
20	0.6	77	0.6496	0.6505
20	0.8	65	0.6515	0.6547

E. Analysis of Optimal ϕ

The value of ϕ indicates the proportion we should cache according to popularity. Fig. 6 shows the ϕ values of cell groups with different levels of accuracy when $\alpha = 0.4$ and $\alpha = 1.2$. Two key findings are listed as follows: (1) The ϕ value decreases as the prediction accuracy increases, which means that for cells with high predictability, we should have greater trust in the prediction result, and vice versa; (2) For the same group of cells, higher α results in larger ϕ. This

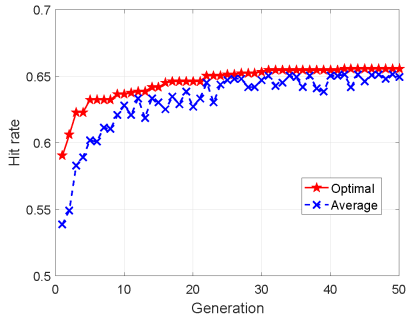

Fig. 7. Convergence curve of GA at $n_P = 20$, $r_c = 0.7$.

indicates that when mobile users have higher similarity, the performance of caching by popularity becomes better.

VII. Experiments and Evaluation

In this section, we evaluate our proposed model on a real-life dataset to further confirm its effectiveness. Case studies on two cities in the southern part of China are provided. We also analyze the influence of cache storage capacity and the similarity degree of mobile users on the hit rate for different methods.

A. Dataset Description

The real-life dataset, which was originally collected by China Mobile™, is preprocessed to extract certain fields and protect users' privacy. In the dataset, we have 5,000 mobile users from one province in the southern part of China (nearly 80,000 cells). The total size of the data is 11.1 GBytes. Each user's behavior is recorded every 5 minutes during the period from 23:55 14/11/2015 to 23:50 05/12/2015, which means that there are 288 time slots per day and three weeks (6,048 time slots) in total. For each user, we have the cell location, service type ID (e.g., QQ, WeChat), and downlink/uplink bytes at different timestamps. Considering the limitation of the dataset, i.e., we only have the service type information in the dataset but not the content request information, in our experiments we assume that the content popularity follows the Zipf distribution. All the users' request information is obtained by simulation based on the service type, and noise is added to the prediction results.

B. Performance of Prediction Model

There have been many work in the area of prediction, whereas in this paper, our main contribution is how to design the caching strategy based on popularity and prediction. However, we can still obtain an improved prediction accuracy by applying our proposed model. Here we evaluate the effectiveness of our proposed prediction model. Using datasets from three cities, the prediction accuracy of four models, i.e., the first-order, second-order, third-order Markov Model and our proposed model, are compared. Here, the prediction accuracy is defined as the ratio between the number of correct predictions and the total number of predictions for all the users.

TABLE II
Comparison of Prediction Accuracy

Model	City *A*	City *B*	City *C*
1^{st}-order Markov	0.56 +/- 0.20	0.53 +/- 0.17	0.59 +/- 0.22
2^{nd}-order Markov	**0.60 +/- 0.21**	0.61 +/- 0.16	0.66 +/- 0.19
3^{rd}-order Markov	0.51 +/- 0.22	0.57 +/- 0.18	0.62 +/- 0.21
Our proposed	**0.60 +/- 0.21**	**0.65 +/- 0.16**	**0.69 +/- 0.18**

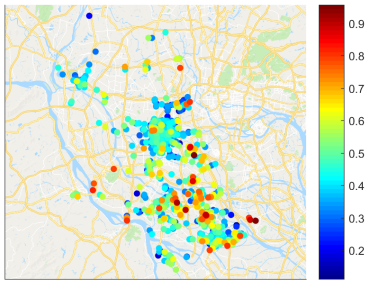

Fig. 8. Averaged prediction accuracy of mobile users in the top 2,000 most visited cells within city A.

As shown in Table II, the second-order Markov Model achieves the highest prediction accuracy compared with other orders. Thus, the second-order Markov Model is selected as the basic prediction model before applying the proposed improvement based on user clustering. The selection of the second-order Markov Model is consistent with the conclusion from other researches (e.g., [25]). The results also show that our proposed method outperforms the other three methods in terms of accuracy.

C. Case Study - City A

We use the data of the first two weeks as the training set, and the third week as the test set. We build a Markov model for users' movements by the data in the training set, and test the model on the test set to get our prediction result. The data in the training set is adopted to build the second-order Markov Model with user clustering improvement, and the data in the test set is applied to test our proposed model. Our first case study is on the data from city A. In city A, there are 7,796 cells in our data. Here, 1,000 cells with high visit frequency and 462 users are taken into consideration, as shown in Fig. 8.

We set $\alpha = 1.6$, $N = 100,000$ and $s = 10$. Based on the characteristics of the data set, we assume that 20% of the files can be refreshed at the beginning of each time slot and the refresh rate is 5 minutes. Fig. 9 shows the averaged cache hit rates of different methods. The selected 1,000 cells are sorted by the averaged prediction accuracy. Here we only consider the uncertainty of users' mobility, which implies that the request of each user at each time is supposed to be known.

The results in Fig. 9 show that when the files are cached by popularity, LFU or LRU, the hit rates of different cells are similar, which are all around 0.82. The reason of the relatively low hit rates is that these methods cannot differentiate the predictability of different cells. By caching files using the optimal strategy in our proposed method, the hit rates improve

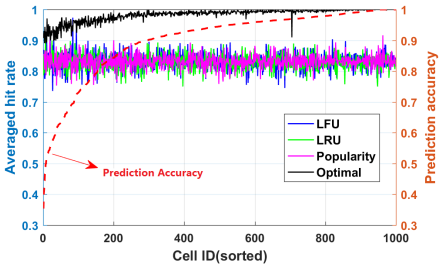

Fig. 9. Averaged cache hit rate comparison among our method and three benchmarks for city A when $\alpha = 1.6$ and $s = 10$.

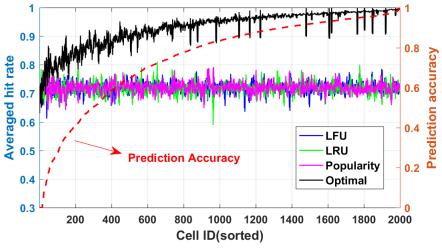

Fig. 10. Averaged cache hit rate comparison among our method and three benchmarks for city B when $\alpha = 1.4$ and $s = 10$.

substantially, especially for those cells with high prediction accuracies. The (average) hit rate of the optimal caching strategy found by our proposed method is 0.91.

D. Case Study - City B

In city B, there are 12,027 cells available in the data set. Here, the top 2,000 most visited cells and 1,015 users are taken into consideration. Fig. 1 shows the prediction accuracy of each cell in the city. We use the same setting of parameters as in Section VII-C, except that $\alpha = 1.4$ for city B. The average cache hit rates of our proposed method and the other three benchmark methods are obtained and compared in Fig. 10.

The results of city B in Fig. 10 are similar to the results of city A in Fig. 9. In city B, our proposed method still outperforms the three benchmark methods. Compared with city A, there exists a decrease of the average cache hit rates of the three benchmark methods from 0.82 to 0.7 in city B, which can be primarily attributed to the lower α value. The accuracy of the optimal caching strategy by our proposed prediction model in city B is also lower than city A.

E. Effect of Varying Content Popularity Distribution and Varying Storage Capacity

We compare our proposed method with the three benchmark methods with varying storage sizes under four different α values ($\alpha = 0.2, 0.6, 1.0, 1.4$) for city A, as shown in Fig. 11. We find that the hit rate turns higher as the storage size s increases, and the hit rate also increases with a larger value of α. The averaged hit rate of the optimal caching strategy obtained by our proposed method is higher than that of the other methods in all cases.

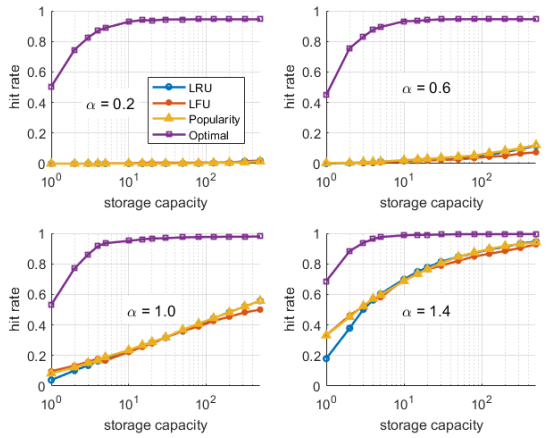

Fig. 11. Averaged cache hit rates with varying storage limit and varying α value with perfect prediction on content requests.

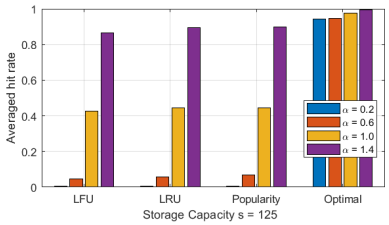

Fig. 12. Averaged cache hit rates comparison with varying α value when $s = 125$.

Fig. 12 shows the average hit rates of our optimal method and the three other benchmark methods when the cache storage capacity is $s = 125$. We can clearly see that our method is always the best among all these four methods; especially when α is small, the hit rate of our method is much higher than those of the other methods. This demonstrates again that our method can better deal with the heterogeneity of users.

If we assume that different cells have different prediction accuracies on the content prediction, then the hit rate of our algorithm will decrease slightly, since our algorithm can adaptively balance between popularity and prediction, as shown in Fig. 13. In this experiment, we divide the cells randomly into 4 groups, and set the content request prediction accuracy as 0.3, 0.5, 0.7 and 0.9 respectively. Since the performance of LFU, LRU and PC is not affected by the prediction accuracy, in Fig. 13, only the results of our optimal method are shown. By assuming a noise in the prediction accuracy, the hit rate of our proposed method reduces slightly due to the decreased predictability of users.

VIII. Conclusions

In this paper, we proposed an adaptive edge caching algorithm for mobile networks to improve users' experience and reduce cost. In our algorithm, mobile users' historical behavior in terms of mobility and content requests is applied to make predictions, and an optimization model is built based on it. In the optimization model, the optimal caching strategy

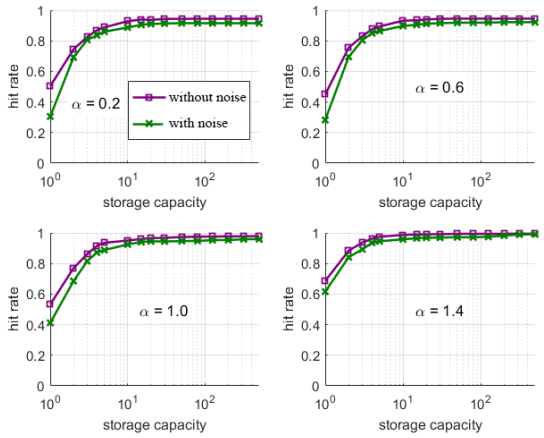

Fig. 13. Averaged cache hit rates of our proposed method with varying storage limit and varying α value with/without perfect prediction on content requests.

is obtained by adaptively dividing the cache storage into two parts, one based on popularity and the other based on prediction. What's more, a robust prediction model based on a second-order Markov model with improvement by iVAT clustering is proposed for mobile users' behavior prediction. Experiments on both simulation data and real-life data show that our algorithm outperforms three benchmarks, LFU, LRU and Popularity Caching in terms of the averaged hit rate. The advantage of our model is more prominent when the storage capacity is limited and the similarities of mobile users are relatively low. The results demonstrate that our method is more suitable for resource limited networks and more robust for the heterogeneity of mobile users' behavior.

In the future, some potential research directions are: (1) Improve the user behavior prediction model by considering the temporal characteristics, e.g., the effects of different time periods of a day or different days of a week. (2) Propose a joint prediction model for users' movement and content request behavior. In this paper, we predict users' movements and content requests separately, but there may exist a correlation between these two behaviors. (3) Further increase the hit rate by incorporating the temporal characteristics into the caching model. In this paper, the users' movement and content request behavior in all the time periods are taken into consideration as a whole. In the future, we may further improve the caching model by considering different time periods of a day or different days of a week separately.

Acknowledgments

This work is partially supported by China Scholarship Council.

References

[1] K. Poularakis and L. Tassiulas, "Cooperation and information replication in wireless networks," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 374, 2016.

[2] G. Ma, Z. Wang, M. Zhang, J. Ye, M. Chen, and W. Zhu, "Understanding performance of edge content caching for mobile video streaming," *IEEE Journal on Selected Areas in Communications*, vol. 35, pp. 1076–1089, 2017.

[3] D. Liu, B. Chen, C. Yang, and A. F. Molisch, "Caching at the wireless edge: design aspects, challenges, and future directions," *IEEE Communications Magazine*, vol. 54, pp. 22–28, 2016.

[4] Y. Qiao, Z. Si, Y. Zhang, F. B. Abdesslem, X. Zhang, and J. Yang, "A hybrid markov-based model for human mobility prediction," *Neurocomputing*, vol. 278, pp. 99–109, 2018.

[5] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," *Science*, vol. 327, pp. 1018–1021, 2010.

[6] S. Zhang, N. Zhang, P. Yang, and X. Shen, "Cost-effective cache deployment in mobile heterogeneous networks," *IEEE Transactions on Vehicular Technology*, vol. 66, pp. 11 264–11 276, 2017.

[7] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "Mobile edge computing: Survey and research outlook," *CoRR*, pp. 1–30, 2017.

[8] S. Zhang, P. He, K. Suto, P. Yang, L. Zhao, and X. Shen, "Cooperative edge caching in user-centric clustered mobile networks," *IEEE Transactions on Mobile Computing*, vol. 17, pp. 1791–1805, 2018.

[9] P. Yang, N. Zhang, S. Zhang, L. Yu, J. Zhang, and X. Shen, "Dynamic mobile edge caching with location differentiation," in *Proc. IEEE Global Communications Conference (GLOBECOM'17)*, Singapore, 2017, pp. 1–6.

[10] L. Qiu and G. Cao, "Popularity-aware caching increases the capacity of wireless networks," in *Proc. IEEE Conference on Computer Communications (INFOCOM'17)*, Atlanta, USA, 2017, pp. 1–9.

[11] S. Li, J. Xu, M. van der Schaar, and W. Li, "Popularity-driven content caching," in *Proc. IEEE International Conference on Computer Communications (INFOCOM'16)*, San Francisco, USA, 2016, pp. 1–9.

[12] N. Zhang, K. Zheng, and M. Tao, "Using grouped linear prediction and accelerated reinforcement learning for online content caching," in *Proc. IEEE International Conference on Communications Workshops (ICC Workshops'18)*, Kansas City, USA, 2018, pp. 1–6.

[13] R. Wang, X. Peng, J. Zhang, and K. B. Letaief, "Mobility-aware caching for content-centric wireless networks: modeling and methodology," *IEEE Communications Magazine*, vol. 54, pp. 77–83, 2016.

[14] D. Liu, C. Yang, and V. C. M. Leung, "When exploiting individual user preference is beneficial for caching at base stations," in *Proc. IEEE International Conference on Communications Workshops (ICC Workshops'18)*, Kansas City, USA, 2018, pp. 1–6.

[15] S. Rashidi and S. Sharifian, "A hybrid heuristic queue based algorithm for task assignment in mobile cloud," *Future Generation Computer Systems*, vol. 68, pp. 331–345, 2017.

[16] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[17] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, pp. 311–338, 2000.

[18] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. New York, USA: Oxford University Press, Inc., 1996.

[19] T. D. Gwiazda, *Crossover for single-objective numerical optimization problems*. Tomasz Gwiazda, 2006, vol. 1.

[20] K. Deep, K. P. Singh, M. L. Kansal, and C. Mohan, "A real coded genetic algorithm for solving integer and mixed integer optimization problems," *Applied Mathematics and Computation*, vol. 212, pp. 505–518, 2009.

[21] J. C. Bezdek and R. J. Hathaway, "Vat: a tool for visual assessment of (cluster) tendency," in *Proc. International Joint Conference on Neural Networks (IJCNN'02)*, Honolulu, USA, 2002, pp. 2225–2230.

[22] L. Wang, U. T. V. Nguyen, J. C. Bezdek, C. A. Leckie, and R. Kotagiri, "ivat and avat: Enhanced visual analysis for cluster tendency assessment," in *Proc. Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'10)*, Hyderabad, India, 2010, pp. 16–27.

[23] L. Li and C. Leckie, "Trajectory pattern identification and anomaly detection of pedestrian flows based on visual clustering," in *Proc. Intelligent Information Processing VIII (IIP'16)*, Melbourne, Australia, 2016, pp. 121–131.

[24] L. Cao and M. Grabchak, "Smoothly truncated levy walks: Toward a realistic mobility model," in *Proc. IEEE International Performance Computing and Communications Conference (IPCCC'14)*, Austin, USA, 2014, pp. 1–8.

[25] S. Uppoor, "Understanding and exploiting mobility in wireless networks," Ph.D. dissertation, Lyon, INSA, 2013.