A proactive mobile edge cache policy based on the prediction by partial matching

Lincan Li, Chiew Foong Kwong, Qianyu Liu

University of Nottingham
UK | CHINA | MALAYSIA

University of Nottingham Ningbo China, 199 Taikang East Road, Ningbo, 315100, Zhejiang, China.

First published 2020

University of Nottingham
UK | CHINA | MALAYSIA

A proactive mobile edge cache policy based on the prediction by partial matching

Lincan Li[1], Chiew Foong Kwong[1*], Qianyu Liu[2]

[1]Department of Electrical and Electronic Engineering, University of Nottingham Ningbo China, 315100, Ningbo, China

[2]International Doctoral Innovation Centre, University of Nottingham Ningbo China, 315100, Ningbo, China

[*]Corresponding author: Chiew Foong Kwong. Email: chiew-foong.kwong@nottingham.edu.cn

ARTICLE INFO

Keywords:

cache

latency

cache hit rate

content prediction

location prediction

ABSTRACT

The proactive caching has been an emerging approach to cost-effectively boost the network capacity and reduce access latency. While the performance of which extremely relies on the content prediction. Therefore, in this paper, a proactive cache policy is proposed in a distributed manner considering the prediction of the content popularity and user location to minimise the latency and maximise the cache hit rate. Here, a backpropagation neural network is applied to predict the content popularity, and prediction by partial matching is chosen to predict the user location. The simulation results reveal our proposed cache policy is around 27%-60% improved in the cache hit ratio and 14%-60% reduced in the average latency, compared with the two conventional reactive policies, i.e., LFU and LRU policies.

1. Introduction

This article is an extended version of a conference paper presented in 2018 at the Biomedical Engineering, Healthcare, Robotics, and Artificial Intelligence 2018 [1].

With the ubiquitous emergence of the smart mobile devices, e.g., the mobile phones, as well as the trend towards high data rate applications, exponential mobile data have been generated in the wireless network while the current network cannot support the fast growth of the mobile data traffic [2] [3]. As a result, the storage capacity of the wireless network needs to be expanded and the conventional way is the dense-deployment of the base stations (BSs). However, it costs a lot for the mobile operator to upgrade this infrastructure [4]. Therefore, a more cost-effective approach is needed and the caching technique is regarded as an ideal approach [5].

By deploying the cache devices at the edge of the wireless network in proximity to the users, e.g., BSs and user terminals (UTs), and storing the popular contents at the cache devices [6] [7], users can directly retrieve the contents from the edge nodes rather than the remote core network via the backhaul links [8]. Hence, the content access latency can be decreased due to the reduced content transmission distance [9]. In parallel, duplicated requests for the same contents from the core network to the edge of the network can be avoided, which reduces the potential data congestion of the network [10]. Furthermore, due to the increasingly decreased prices of the cache devices, the storage capacity of the wireless network can be more cost-effectively boosted compared to the conventional way [4].

Due to the limited storage capacity of the cache devices, only a part of the contents can be stored in the edge cache devices. Hence, multiple works are focusing on how to design an efficient cache content placement policy. The most common approaches are least frequently used (LFU) and least recently used (LRU), which are referred to as reactive cache policies that determine whether to cache a specific content after it has been requested [11] [12]. In detail, LRU always caches the most recently requested contents while LFU caches the most frequently requested contents [13]. While the reactive cache policy is not efficient during peak hours. Hence, the proactive caching strategy is introduced, by which the content can be cached before the request, and hence the users can access the preferred content immediately when they arrive in new areas.[14] [15].

There are many proactive schemes have been investigated. In [16], a threshold-based proactive cache scheme based on reinforcement is presented, aiming at minimising the average energy cost. In this case, the time variation of the content is considered, which means the content popularity is changed over time rather than static. In practice, only the content whose lifetime is not expired has the potential to be cached. In [17], a caching scheme is presented to improve the cache hit rate and reduce energy consumption by predicting the content popularity distribution. In [18], a proactive cache based on the estimation of the content popularity is presented, targeting increasing the cache hit rate and decreasing the content transmission expenditure. Motivated by deep learning which can improve the accuracy of the content prediction, many works utilise deep learning for proactive caching. In [19], deep learning is utilised to predict the future probability of the content and the predicted content with a high probability will be cached. In [20], a proactive cache policy is

proposed based on a deep recurrent neural network model which can predict the future content requests.

Besides, in [21], a proactive cache policy for the vehicular network is proposed, where the roadside units (RSUs) are equipped with the cache capability under high mobility of the moving vehicles. There, a long-short term memory (LSTM) network is utilised to predict the direction of the moving vehicles. Then the proactive cache problem is modeled as a Markov decision process (MDP) problem and solved by a heuristic ε_n-greedy algorithm. In [22], Gao *et al*. design a proactive cache scheme for the hierarchical network where each small base station (SBS) can perceive the user mobility of its adjacent small base stations (SBSs), aiming at maximising the cache hit rate and minimising the transmission latency. In specific, the users with different moving speeds are clustered into different layers and the cached content deployment problem is solved by a genetic algorithm. In [23], a cooperative cache framework is introduced to increase the cache hit rate and minimise the access latency, in which the prediction by partial matching (PPM) is utilised to predict the vehicles' probability of arriving at the hot areas. The vehicles with long sojourn time in a hot spot are equipped with cache capability and are regarded as cache nodes. A summary of the aforementioned works is shown in Table 1.

Table 1 Summary of existing cache policies.

Policy	Contribution
[11] [12]	LFU and LRU are explained to manage the cache content updating.
[16]	A reinforcement learning-based proactive cache policy is proposed to minimise energy consumption. Here, content popularity is a time-varying variable and only the contents whose lifetime are not expired can be considered to be cached or not.
[17]	An accurate content popularity prediction is adopted to improve the cache hit rate and reduce energy consumption.
[18]	A proactive cache policy is proposed to increase the cache hit rate and decrease the content transmission cost. Here, transfer learning is applied to evaluate the content popularity, and a greedy algorithm is adopted to deal with the cache problem.
[19]	A deep learning algorithm is applied to predict the future probability of the content, and the content with a high predicted probability will be pre-cached.
[20]	A proactive cache policy is proposed to alleviate the data congestion and reduce the average latency, in which a deep recurrent neural network algorithm is adopted to predict future content requests.
[21]	A long-short term memory (LSTM) network is utilised to predict the direction of the moving vehicles, and the proactive cache problem is modeled as MDP and solved by a heuristic ε_n-greedy algorithm.
[22]	A two-layer cache network consisting of several MSBs and SBSs is proposed to improve the cache hit ratio and reduce the average latency. Here, the adjacent SBSs can communicate with each other. Besides, the users with different moving speeds are clustered into different layers, i.e. the MBS or the SBS.
[23]	A cooperative cache framework is proposed to increase the cache hit ratio and reduce access latency. Here, a PPM algorithm is adopted to predict vehicles' probability of arriving in the hot areas.

Different from the aforementioned works singly considering the prediction of the content popularity or the users' location, this extended paper designs a proactive cache policy jointly considering the prediction of the user preference and the user location to minimise the average latency and maximise the cache hit rate, which to the best of our knowledge has not been considered in the prior research works. In detail, a practical scenario is considered, in which the BSs are distributed and the users are mobile. A backpropagation (BP) neural network, one of the deep learning methods, is applied to predict the user preference based on the historical content requests. Furthermore, the user's future location is predicted via PPM which has been introduced in our previous work [1], and the user's preferred content is pre-cached at the location in which the user will highly arrive. The main contributions of this paper are as follows:

- This paper focuses on minimising the average latency and maximising the cache hit rate by jointly considering the content popularity prediction and user location prediction.
- The BP neural network is applied to predict the content popularity, and PPM is chosen to predict the user location.
- The effect of the several parameters on the cache performance is investigated, i.e., the Zipf parameter, the content size, the transmission rate, the distance of the backhaul link, and the distance between the user and the BS.

The remainder of this paper is organised as follows. The system model and the problem formulation are shown in section 2. Section 3 introduces the proactive cache policy. We show the simulation results in section 4 and conclude in section 5.

2. System model and problem formulation

In this section, we describe the system model, state the assumption, and formulate the problem.

2.1. System model

For each time slot *t* whose period is one hour, the proposed proactive cache policy adopts the PPM algorithm to obtain the probability of the user arriving at different locations. The location with the highest value is regarded as the future location. In parallel, the prediction of the user preference is trained via the BP neural network. Once the predicted user preference and the future location are obtained, the popular contents in the user preference are pre-cached at the future location. Consequently, once the user arrives in this location in the next time slot (*t*+1), the user can immediately obtain the requested content. However, if the prediction is not accurate, the BS needs to retrieve the requested content from the core network and then send it to the users, which imposes a more latency consumption issue.

As shown in Figure 1, the distributed cache architecture consists of the following network equipment (NE): a core network, $\mathcal{M}$ cache-enabled BSs, and $\mathcal{N}$ mobile users. The $\mathfrak{m}^{th}$ BS is denoted by BS_m for $1 < \mathfrak{m} < \mathcal{M}$, the n^{th} user is denoted by U_n for $1 < n < \mathcal{N}$, the circular coverage area of BS_m is denoted as A_m, and the set of the users served by BS_m is denoted as U_m. Let $\mathcal{F}_n=\{f_{n1}, f_{n2},\ldots, f_{nk}\}$ denotes the set of k contents requested by U_n. The content popularity distribution of a user, *i.e.* user preference, follows Zipf distribution law [24]. The popularity of the $\mathfrak{R}^{th}$ content requested by U_n is characterized as:

$$\mathfrak{p}_n(\mathfrak{R},\mathfrak{M},\mathfrak{U}) = \frac{\frac{1}{\mathfrak{R}^{\mathfrak{M}}}}{\sum_1^{\mathfrak{U}}\frac{1}{\mathfrak{U}^{\mathfrak{M}}}}, \quad (1)$$

where $\mathfrak{R}$ is the rank of the content in $\mathcal{F}_n$, $\mathfrak{M}$ is the Zipf parameter for $0< \mathfrak{M}<1$, and $\mathfrak{U}$ is the total number of contents in $\mathcal{F}_n$.

Let $\mathcal{F}_{U_m}$ represents the set of the contents requested at BS_m, $\mathfrak{p}_{U_\mathfrak{m}}$ represents the content popularity at BS_m and $\mathcal{C}=\{1,2,\ldots,\mathcal{H}\}$ represents the library of all the contents requested by $\mathcal{N}$ mobile users served by $\mathcal{M}$ BSs. Assume each BS can store $\hbar$ contents at most for $\hbar< \mathcal{H}$, and each content has the same size $\mathcal{B}$. Besides, one user only requests one content at most for each time slot *t*.

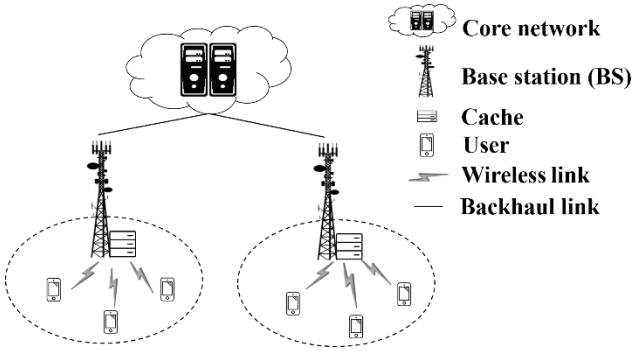

Figure 1: The cache-enabled network.

2.2. Problem formulation

Based on the mention before, our target is to minimise the access latency $\mathfrak{T}$, which is comprised of the transmission latency and propagation latency [25]. The transmission latency is caused by transmitting the content from EN_i to EN_j [26], in which EN_i and EN_j are any two network equipment. According to [27], the transmission rate $\mathbb{R}_{(i,j)}$ is calculated as:

$$\mathbb{R}_{(i,j)} = \mathrm{B}\log_2(1+\frac{\rho\mu}{\sigma^2}), \quad (2)$$

where B (Hz) is the available spectrum bandwidth, ρ is the transmitted power, σ^2 is the noise power and μ is the channel gain between EN_i and EN_j.

Therefore, the transmission latency $\mathfrak{T}_{t_{(i,j)}}^{c}$ based on the size of the requested content c and the transmission rate is derived as:

$$\mathfrak{T}_{t_{(i,j)}}^{c} = \frac{\mathcal{S}}{\mathbb{R}_{(i,j)}}, \quad (3)$$

where $\mathcal{S}$ is the size of requested content c.

The propagation latency $\mathfrak{T}_{p_{(i,j)}}^{c}$ is defined as the time of propagating the requested content c from EN_i to EN_j. Affected by the propagation speed of the electromagnetic wave and the distance between the EN_i and EN_j, the propagation latency $\mathfrak{T}_{p_{(i,j)}}^{c}$ is expressed as:

$$\mathfrak{T}_{p_{(i,j)}}^{c} = \frac{\mathcal{J}_{(i,j)}}{v}, \quad (4)$$

where v is the propagation speed of the electromagnetic wave in the corresponding channel, $\mathcal{J}_{(i,j)}$ is the distance between EN_i and EN_j.

Therefore, the access latency is expressed as:

$$\mathfrak{T} = \mathfrak{T}_{t_{(i,j)}}^{c} + \mathfrak{T}_{p_{(i,j)}}^{c} \quad (5)$$

$$= \frac{\mathcal{S}}{\mathbb{R}_{(i,j)}} + \frac{\mathcal{J}_{(i,j)}}{v}. \quad (6)$$

In detail, the content can be directly retrieved from BS if it is hit at the BS, i.e., the content is cached at the BS. Hence the latency $\mathfrak{T}_{hit}$ of cached content is shown as:

$$\mathfrak{T}_{hit} = \frac{\mathcal{S}}{\mathbb{R}_{(u,b)}} + \frac{\mathcal{J}_{(u,b)}}{v_{hit}}, \quad (7)$$

where $\mathbb{R}_{(u,b)}$ is the transmission rate between a user and a BS, $\mathcal{J}_{(u,b)}$ is the distance between a user and a BS, and v_{hit} is the propagation speed of the electromagnetic wave in the air.

Otherwise, the content needs to be retrieved from the core network via the backhaul links if the content is missed at the BS, i.e., the content is not cached at the BS. According to [28], the transmission rate $\mathbb{R}_{(u,core)}$ from the core network to the BS is shown as

$$\mathbb{R}_{(b,core)} = \mathrm{R}^*, \quad (8)$$

where R^* is the maximal transmission rate of the network.

Therefore, the latency of a missed content $\mathfrak{T}_{missed}$ consisting of the transmission latency $\mathfrak{T}_{t_{(u,core)}}^{c}$ and the propagation latency of a missed content is expressed as:

$$\mathfrak{T}_{t_{(u,core)}}^{c} = \frac{\mathcal{S}}{\mathrm{R}^*} + \frac{\mathcal{S}}{\mathbb{R}_{(u,b)}}, \quad (9)$$

$$\mathfrak{T}_{p_{(u,core)}}^{c} = \frac{\mathcal{J}_{(u,b)}}{v_{hit}} + \frac{\mathcal{J}_{(b,core)}}{v_b}, \quad (10)$$

$$\mathfrak{T}_{missed} = \frac{\mathcal{S}}{\mathbb{R}_{(u,b)}} + \frac{\mathcal{S}}{\mathrm{R}^*} + \frac{\mathcal{J}_{(u,b)}}{v_{hit}} + \frac{\mathcal{J}_{(b,core)}}{v_b}, \quad (11)$$

where $\mathcal{J}_{(u,b)}$ is the distance between the user and the BS, $\mathcal{J}_{(b,core)}$ is the distance between the BS and the core network, and v_b is the propagation speed of the electromagnetic wave in the backhaul link.

Therefore, the average system latency $\overline{\mathfrak{T}}$ is calculated as:

$$\overline{\mathfrak{T}} = \frac{\sum_1^N [\left(\frac{\mathcal{S}}{\mathbb{R}_{(u,b)}} + \frac{\mathcal{J}_{(u,b)}}{v_{hit}}\right) + (1-\mathcal{Z})(\frac{\mathcal{S}}{\mathrm{R}^*} + \frac{\mathcal{J}_{(b,core)}}{v_{bk}})]}{N}, \quad (12)$$

where $\mathcal{Z}$ is the cache hit rate and is calculated as follows:

$$\mathcal{Z} = \frac{\sum_{n=1}^{\mathcal{N}} \sum_1^{\mathcal{G}} F(R_{i_{U_n}})}{\sum_{n=1}^{\mathcal{N}} \mathcal{G}}, \quad (13)$$

where $R_{i_{U_n}}$ is the content requests of U_n, $\mathcal{G}$ is the number of request times of $R_{i_{U_n}}$. The $F(R_{i_{U_n}})$ is calculated as

$$F(R_{i_{U_n}}) = \begin{cases} 1, & R_{i_{U_n}} \textit{ is cached} \\ 0, & R_{i_{U_n}} \textit{ is not ccahed} \end{cases} \quad (14)$$

The problem of minimising the average system latency is modeled as follows

$$\mathbf{P\_1}: \min \overline{\mathfrak{T}} \tag{15}$$

$$\text{s.t. } 0<v_{bk}<v_{hit} \leq 3\times 10^{8} m/s \tag{16}$$

$$0\leq \mathcal{Z} \leq 1 \tag{17}$$

3. The proactive cache based on the content popularity prediction and future location prediction

In this section, a proactive cache policy is proposed to address **P_1**. Firstly, the user preference is predicted according to the backpropagation (BP) neural network. Besides, we introduce the future location prediction based on the prediction by partial matching (PPM) algorithm. The proposed cache policy minimises the average system latency by pre-caching the predicted popular content at the correspondingly predicted location.

3.1. The content popularity prediction based on backpropagation neural network

User preference is the content probability distribution of individual user and content popularity is the content probability distribution of a cluster of users. Due to the characteristic of the user preference that a small number of contents account for most of the data traffic, the cache policy considers caching the popular content to reduce the complexity of the computation. Hence, the set of the popular contents of U_n is denoted as $\mathbb{P}_{U_n} = \{y_{U_n}^1, y_{U_n}^2, \ldots, y_{U_n}^k\}$, which contains k samples by choosing the top k contents with the highest probability from the user preference. Therefore, the set of the popular contents at BS_m is denoted as $\mathbb{P}_{BSm} = \{\mathbb{P}_1, \mathbb{P}_2, \ldots \mathbb{P}_{U_n} \ldots, \mathbb{P}_{U_m}\}$.

After obtaining the popular content database of BS_m, the BP neural network, as shown in Figure 2, is applied to predict the content popularity. The proposed neural network is comprised of three layers, namely the input layer, hidden layer, and output layer. The number of the neuron cells in the input layer and the output layer is equal to the cache storage k. The content requests of U_n are collected each hour and denoted as a training data set. Besides, two continuous training data sets are chosen to optimise the parameter of the neural network. The value y_{in} for the input layer is the request times of the top k popular contents in the former training data set. The value y_{real} is the request times of the top k popular contents in the latter training data set. Furthermore, mean squared error (MSE) is utilised as the loss function in the content prediction. The *MSE* is formulated as

$$\text{MSE} = \frac{\sum_1^k (y_{real} - y_{pre})^2}{k}, \tag{18}$$

where y_{pre} is the value of the output layer.

Besides, the Relu function is chosen as the activation function, which is expressed as

$$Relu(y_{in}) = \begin{cases} 0, & y_{in} < 0 \\ y_{in}, & y_{in} \geq 0 \end{cases}. \tag{19}$$

With the help of stochastic gradient descent (SGD), the proposed neural network can optimally predict the content popularity after enough training.

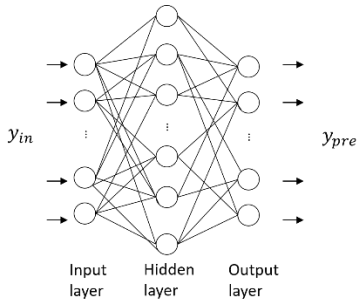

Figure 2: The BP neural network.

3.2. The future location prediction based on a prediction by partial matching

Before the location prediction, the historical location information is collected from a real environment model as shown in Figure 3. The areas labeled by red symbols are regarded as the hot spots with long sojourn time. The historical location information sequence is denoted as $\mathcal{L}$ which is related to the hot spots.

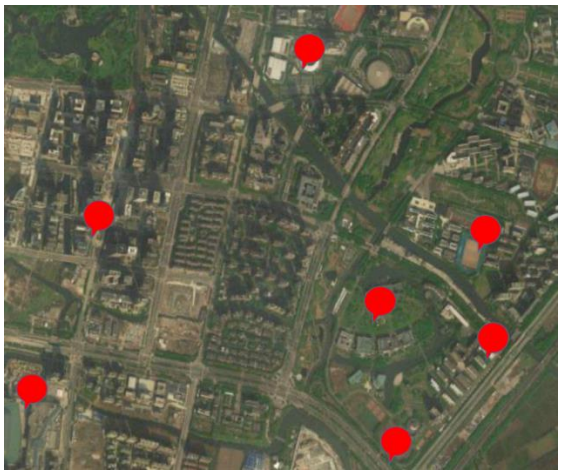

Figure 3: The user movement model.

After obtaining the historical location information $\mathcal{L}$, PPM is applied to predict the user's future location. PPM is a data compression method based on the finite context and it has been proven effective for the location prediction [23]. The probability of the future location *y* appearing after the given context *Con* is model as *P(ycon)*, where *Con* is the sequence of the location and the length of the sequence is called order [29]. Furthermore, PPM proposes an escape mechanism to deal with the zero-frequency problem [30]. When escape occurs, *i.e. y* is missed after *Con*. Then the PPM outputs an escape probability defined as *Pesc(esc|Con).* The computation of PPM is shown in Algorithm 1. Firstly, PPM checks whether *y* appears after *Con*. If *y* appears, PPM records the number of appearing times and outputs the probability $P(y|Con)$, otherwise, PPM outputs the escape probability *Pesc(esc|Con).* Under the escape situation, PPM restarts to check whether *y* appears after the new *Con* (the order of which is the original order minus 1). The process is finished until *y* appears after *Con* or the order is -1. The predictive probability of the future location is the multiple of the sub-probabilities and the calculation is shown as:

$$P = \prod_1^{step} P_i, \quad (20$$

$$P_i = \begin{cases} P(y|Con) = \frac{N_y}{N_{esc}+N_{Con}}, \ y\ appears\ after\ Con \\ Pesc(esc|Con) = \frac{N_{esc}}{N_{esc}+N_{Con}}, \ y\ escapes\ after\ Con \end{cases}, \quad (21)$$

where P_i is the probability of step i, N_y represents the number of the times of y appearing after *Con*, N_{esc} represents the number of the characters appearing after *Con*, and N_{Con} represents the number of the times of all the characters appearing after *Con*.

Once the probabilities of the possible locations are obtained via PPM, these obtained probabilities are ranked in descending order. The location with the highest probability is regarded as the future location.

Algorithm 1 PPM algorithm

Input: historical data $\mathcal{L}$

Output: the probability P of the future location y

```
Initialize P=0
order = h, j =1
if y appears after Con
    P_j=P(y|Con)
    Process finished
else
    P_j=P(esc|Con)
    h - =1, j + =1
    Restart the step 3-7
The process is finished until y appears after Con or h =-1.
Output j, and P=∏_1^j P_j
```

Here is an example to help understand PPM computation by giving a user path $\{L_1, L_3\}$ and the future location L_4 in the historical data sequence $\mathcal{L}$ ={$L_1, L_2, L_3, L_4, L_5, L_1, L_3, L_1, L_4, L_1, L_2, L_4, L_3, L_4, L_1$}. First, since the sequence{ L_1, L_3, L_4 } cannot be found from the historical data sequence, the escape probability P(esc| L_1, L_3) is outputted based on *Pesc(esc|Con)* in Eq. (21), as shown in Eq. (22). Then the new order is 1 and consequently, the new context is { L_3 }. The new sequence { L_3, L_4 } can be found from the historical date sequence, and therefore the probability P(L_4| L_3) is obtained based the $P(y|Con)$ in Eq. (21), as shown in Eq. (23). Finally, the probability P(L_4| L_1, L_3) is obtained based on Eq. (20), as shown in Eq. (24).

$$\mathrm{P(esc|}\ L_1, L_3) = \frac{N_{(\mathrm{esc|}\ L1,L3)}}{N_{(\mathrm{esc|}\ L1,L3)}+N_{(L1|\ L1,L3)}} = \frac{1}{1+1} = \frac{1}{2}, \quad (22)$$

where $N_{(\mathrm{esc|}\ L1,L3)}$ is the number of the characters appearing after $\{L_1, L_3\}$, and $N_{(L1|\ L1,L3)}$ is the number of the times of all the characters appearing after $\{L_1, L_3\}$ since only L_1 appears after $\{L_1, L_3\}$.

$$\mathrm{P}(L_4|\ L_3) = \frac{N_{(L4|\ L3)}}{N_{(\mathrm{esc|}\ L3)}+N_{(L4|\ L3)}+N_{(L1|\ L3)}} = \frac{2}{2+2+1} = \frac{2}{5}, \quad (23)$$

where $N_{(L4|\ L3)}$ is the number of L_4 appearing after L_3, $N_{(\mathrm{esc|}\ L3)}$ is the number of the characters appearing after L_3 and $N_{(L1|\ L3)}$ is the number of L_1 appearing after L_3. The sum of $N_{(L4|\ L3)}$ and $N_{(L1|\ L3)}$ is called the number of the times of all the characters appearing after L_3.

$$P(L4|L1, L3) = P(esc|L1,L3) \times P(L4|L3) = \frac{1}{2} \times \frac{2}{5} = \frac{1}{5}, \quad (24)$$

3.3. The pre-deployment of the popular content at the future location

In each time slot *t*, the users' future locations in which users will highly arrive at the next time slot *t+1* are predicted via PPM. In parallel, the user preference at *t+1* is predicted via BP neural network. The top *w* contents with the highest number of request times are regarded as the popular contents in the future. After that, these popular contents are pre-deployed at the corresponding future location. Hence, in the next time slot *t+1*, if the prediction is correct, users can immediately obtain their preferred contents, which extremely reduces the average system latency.

4. Simulation results and analyzation

In this section, we consider a distributed BS caching network which consists of 10 BS, 30 users, and 6 locations. The number of content requests of each user is 3000. The comprehensive simulation shows the performance of our proposed policy, LFU, and LRU in terms of the average latency and cache hit rate. The specific parameter settings are shown in Table 2. The program is modeled via PyTorch language in Pycharm software.To further show the improvement of our proposed policy in terms of the cache hit rate and the reduction of our proposed policy in terms of the cache hit rate compared with LFU and LRU policies, we propose the growth ratio $\mathcal{P}_G$ and the reduction ratio $\mathcal{P}_R$, which are expressed as:

$$\mathcal{P}_G = \frac{\mathcal{C}_{our} - \mathcal{C}_{existing}}{\mathcal{C}_{existing}}, \quad (25)$$

$$\mathcal{P}_R = \frac{\overline{\mathfrak{T}}_{existing} - \overline{\mathfrak{T}}_{our}}{\overline{\mathfrak{T}}_{existing}}, \quad (26)$$

where $\mathcal{C}_{our}$ and $\mathcal{C}_{existing}$ is the cache hit rate of our proposed policy and any one of the LFU and LRU policies, respectively. $\overline{\mathfrak{T}}_{our}$ and $\overline{\mathfrak{T}}_{existing}$ is the average latency of our proposed policy and any one of the LFU and LRU policies, respectively.

Table 2: The simulation parameter settings

symbol	value
v_{bk}	1×10^7 m/s
v_{hit}	3×10^8 m/s
$\mathbb{R}_{(i,j)}$	10~50 Mbps
σ^2	1W
ρ	3W

$\mathcal{J}_{(u,b)}$	10~50km
$\mathcal{J}_{(bs,core)}$	100km
$\mathcal{B}$	30Kb~450Kb
$\mathfrak{M}$	1.1~1.8
$\mathcal{Q}$	1%~10%
δ	2%~10%

Figure 4 reveals the cache hit rate (represented in percentage) of our proactive policy and the conventional reactive policies, *i.e.*, LFU and LRU. The number of the total content requests is 6000, the Zipf parameter of each user varies between 1.7 and 1.8. Besides, to demonstrate the effect of the cache capacity on the cache performance, we introduce the cache capacity ratio $\delta = \frac{\mathcal{h}}{\mathcal{H}}$. And in this simulation, we assume δ= 2%, 4%, 6%, 8% and 10%. Horizontally, the cache hit rates of LFU, LRU, and our proposed policy increase with the larger cache capacity ratio. The tendency demonstrates that increasing the cache capacity can improve the cache hit rate since more popular contents can be cached. We also notice that our proactive policy has the highest cache hit rate, which is around 10-25% higher than that of LFU and LRU policies, no matter how the Zipf parameter varies. Therefore, our proposed policy outperforms the other two policies.

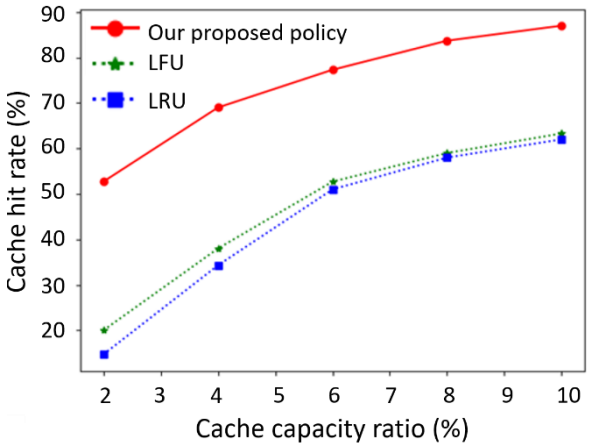

Figure 4: The cache hit rate vs. cache capacity ratio $\delta = \frac{\mathcal{Q}}{\mathcal{H}}$.

Figure 5 investigates the effect of the Zipf parameter $\mathfrak{M}$ on the cache hit rate of our proposed policy with the other two policies as mentioned before. We assume δ is 10%, and the Zipf parameter of each user varies in the range [1.1, 1.2], [1.2, 1.3], [1.3, 1.4], [1.4, 1.5], [1.5, 1.6], [1.6, 1.7] and [1.7, 1.8]. As the Zipf parameter grows, the cache hit rates of all the cache policies increase. The reason is that fewer contents are taking up more content requests as the Zipf parameter grows, and hence the popular content becomes more popular. Considering the fixed number of the total content request, the number of content reduces. With the same capacity, the cache has a higher chance to store more contents and the cached contents are more popular, which contributes to a higher cache hit rate. Furthermore, the slopes of the three curves are gradually reduced. The reason is with the larger Zipf parameter, the newly cached popular contents have fewer content requests compared with the initially cached contents. We also notice that the two reactive policies have a relatively close cache hit rate, and the cache hit rate of our proposed policy is around 24%-38% higher than that of the two reactive policies.

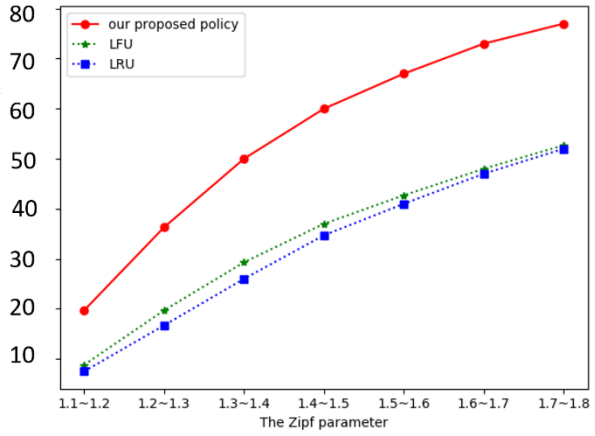

Figure 5:The cache hit rate vs. the Zipf parameter.

The relation between the average latency and the size of the content is displayed in Figure 6. Here, the size of the content is 30Kb, 200Kb, 200Kb, 250Kb, 300Kb, 350Kb, and 450Kb, respectively. Besides, we set the cache capacity ratio δ is 10% and the fluctuation of the Zipf parameter is between 1.7 and 1.8, the distance between the user and the BS is 10km and the distance of the backhaul link is 100km. As the size of the content grows, the average latencies of all the policies increase. The reason is the transmitter consumes more time to send the content into the channel as the size of the content grows. Vertically, the average latency obtained by our proposed policy is around 60% reduced compared with LFU and LRU regardless of the size of the content, which implies our proposed policy outperforms the two reactive policies.

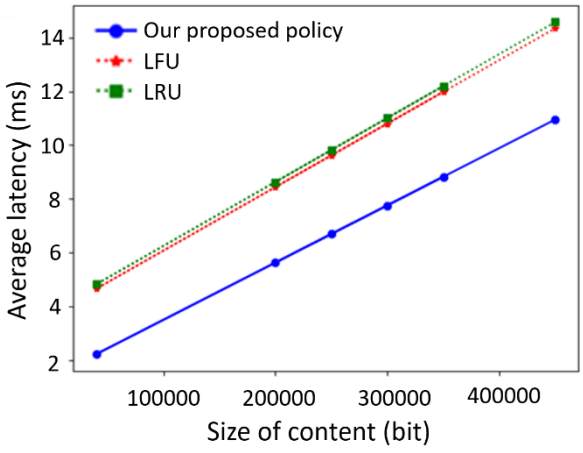

Figure 6: The average latency vs. the size of the content.

Figure 7 shows the relationship between the average latency and the transmission rate between the user and BS. Here, the content size is 400Kb, the storage capacity ratio is 10%, the distance between the user and the BS is 10km and the distance of the backhaul link is 100km. The transmission rate between user and

BS is 10Mbps, 20Mbps, 30Mbps, 40Mbps, and 50Mbps, respectively. As the transmission rate between user and BS grows, the average latencies of all the policies reduce. The reason is that, with the larger transmission rate, the latency between the user and the BS is reduced. Also, the average latency of our proposed policy is 31%-64% reduced compared with the other two policies.

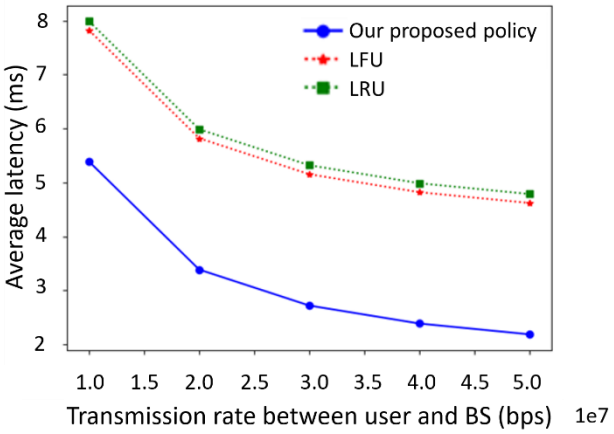

Figure 7:The average latency vs. transmission rate between the user and the BS.

As shown in Figure 8, the average latency is plotted as a function of the Zipf parameter. Here, the Zipf parameter of each user varies in the range [1.1, 1.2], [1.2, 1.3], [1.3, 1.4], [1.4, 1.5], [1.5, 1.6], [1.6, 1.7] and [1.7, 1.8], respectively. Besides, the transmission rate between the user and the BS is 50Mbps, the content size is 400Kb, the storage capacity ratio δ is 10%, the distance between the user and the BS is 10km and the distance of the backhaul link is 100km. As the Zipf parameter increase, the average latencies of three policies are reduced. The reason is that, with the increase of the Zipf parameter, more contents are cached locally, and hence fewer contents need to be retrieved from the remote core network. And the latency from the BS is lower than from the core network. Also, as the Zipf parameter grows, the slopes of the three curves gradually decrease. The tendency is caused since the newly cached contents are less popular than the initially cached contents. Furthermore, our proposed policy is around 14%-53% reduced in terms of the average latency compared with the two reactive policies.

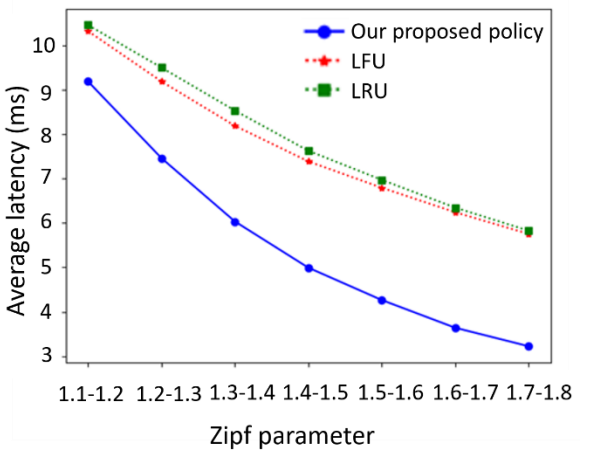

Figure 8: The average latency vs. Zipf parameter.

The effect of the cache capacity ratio δ on the average latency is shown in Figure 9. In this simulation, we assume δ= 2%, 4%, 6%, 8% and 10%. Besides, the content size is 400Kb, the transmission rate is 50Mbps, the distance between the user and the BS is 10km and the distance of the backhaul link is 100km. The cache capacity ratio δ is varied from 2% to 10%. It can be noticed that the average latencies of three policies decrease with the increment of the cache capacity ratio. The fact is that a larger cache capacity means more contents can be cached. As a result, more long-distance propagation time consumption from the core network to the BS can be avoided. Also, the average latency of our proposed policy is around 35%-55% reduced compared with the LFU and LRU.

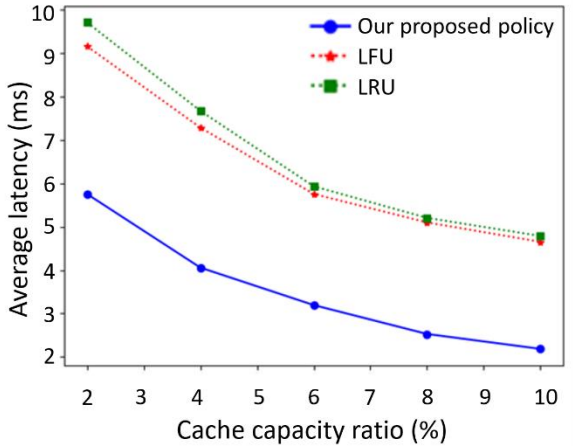

Figure 9: The average latency vs. cache capacity ratio.

5. Conclusion

In this paper, a proactive cache policy is proposed in a distributed manner to minimise the average latency, as well as maximising the cache hit rate. An accurate prediction is achieved to make sure the proactive cache policy can have a high cache performance. In specific, a BP neural network is applied to predict the content popularity, and a PPM algorithm is applied to predict the user location. The simulation results (Fig.4 and Fig.5 simulations) reveal our proposed cache policy is around 10%-38% improved in terms of the cache hit rate no matter how the cache capacity and Zipf parameter vary, compared with LFU and LRU policies. As for the average latency, our proposed policy has at least 14% decrease no matter how parameters change, i.e., the variation of the content size (Fig.6 simulation), the transmission rate between the user and BS (Fig.7 simulation), the Zipf parameter (Fig.8 simulation) and the cache capacity (Fig.9 simulation). Consequently, our proposed policy outperforms LFU and LRU policies.

References

[1] L. Li, C. F. Kwong, F. Chen, Q. Liu, and J. Wang, 'Predicting future location in mobile cache based on variable order of prediction-by-partial-matching algorithm', in *IET Conference Publications*, 2018, pp. 4–4.

[2] E. Bastug *et al.*, 'Big data meets telcos: A proactive caching perspective', *J. Commun. Networks*, vol. 17, no. 6, pp. 549–557, 2015.

[3] Cisco, 'Cisco Annual Internet Report (2018–2023)', *Cisco*, pp. 1–41, 2020.

[4] R. Wang, X. Peng, J. Zhang, and K. B. Letaief, 'Mobility-aware caching for content-centric wireless networks: Modeling and methodology', *IEEE Commun. Mag.*, vol. 54, no. 8, pp. 77–83, 2016.

[5] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, 'A Survey on Mobile Edge Computing: The Communication Perspective', *IEEE Commun. Surv. Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[6] F. M. Modesto and A. Boukerche, 'An analysis of caching in information-centric vehicular networks', *IEEE Int. Conf. Commun.*, 2017.

[7] Y. Li, C. Zhong, M. C. Gursoy, and S. Velipasalar, 'Learning-based delay-aware caching in wireless D2D caching networks', *IEEE Access*, vol. 6, pp. 77250–77264, 2018.

[8] S. Zhang, N. Zhang, P. Yang, and X. Shen, 'Cost-Effective Cache Deployment in Mobile Heterogeneous Networks', *IEEE Trans. Veh. Technol.*, vol. 66, no. 12, pp. 12264–12276, 2017.

[9] Z. Luo, M. LiWang, Z. Lin, L. Huang, X. Du, and M. Guizani, 'Energy-Efficient Caching for Mobile Edge Computing in 5G Networks', *Appl. Sci.*, vol. 7, no. 6, p. 557, 2017.

[10] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, 'A Survey on Mobile Edge Networks: Convergence of Computing, Caching and Communications', *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[11] H. Ahlehagh and S. Dey, 'Video-aware scheduling and caching in the radio access network', *IEEE/ACM Trans. Netw.*, vol. 22, no. 5, pp. 1444–1462, 2014.

[12] S. Shukla, O. Bhardwaj, A. A. Abouzeid, T. Salonidis, and T. He, 'Proactive Retention-Aware Caching with Multi-Path Routing for Wireless Edge Networks', *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1286–1299, 2018.

[13] Y. Jiang, M. Ma, M. Bennis, F. C. Zheng, and X. You, 'User preference learning-based edge caching for fog radio access network', *IEEE Trans. Commun.*, vol. 67, no. 2, pp. 1268–1283, 2019.

[14] N. Golrezaei, A. Molisch, A. G. Dimakis, and G. Caire, 'Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution', *IEEE Commun. Mag.*, vol. 51, no. 4, pp. 142–149, 2013.

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

[16] S. O. Somuyiwa, A. Gyorgy, and D. Gunduz, 'A Reinforcement-Learning Approach to Proactive Caching in Wireless Networks', *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1331–1344, 2018.

[17] M. Yan *et al.*, 'Assessing the energy consumption of 5G wireless edge caching', *2019 IEEE Int. Conf. Commun. Work. ICC Work. 2019 - Proc.*, vol. 7, 2019.

[18] T. Hou, G. Feng, S. Qin, and W. Jiang, 'Proactive Content Caching by Exploiting Transfer Learning for Mobile Edge Computing', *GLOBECOM 2017 - 2017 IEEE Glob. Commun. Conf.*, pp. 1–6, 2017.

[19] K. Thar, N. H. Tran, T. Z. Oo, and C. S. Hong, 'DeepMEC: Mobile edge caching using deep learning', *IEEE Access*, vol. 6, pp. 78260–78275, 2018.

[20] L. Ale, N. Zhang, H. Wu, D. Chen, and T. Han, 'Online proactive caching in mobile edge computing using bidirectional deep recurrent neural network', *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5520–5530, 2019.

[21] L. Hou, L. Lei, K. Zheng, and X. Wang, 'A Q - Learning-Based Proactive Caching Strategy for Non-Safety Related Services in Vehicular Networks', *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4512–4520, 2019.

[22] N. Gao, X. Xu, Y. Hou, and L. Gao, 'A Mobility-aware Proactive Caching Strategy in Heterogeneous Ultra-Dense Networks', *IEEE Int. Symp. Pers. Indoor Mob. Radio Commun. PIMRC*, vol. 2019–Septe, 2019.

[23] L. Yao, A. Chen, J. Deng, J. Wang, and G. Wu, 'A Cooperative Caching Scheme Based on Mobility Prediction in Vehicular Content Centric Networks', *IEEE Trans. Veh. Technol.*, vol. 9545, no. c, pp. 1–10, 2017.

[24] D. Liu and C. Yang, 'Caching at Base Stations with Heterogeneous User Demands and Spatial Locality', *IEEE Trans. Commun.*, vol. 67, no. 2, pp. 1554–1569, 2019.

[25] F. Jiang, Z. Yuan, C. Sun, and J. Wang, 'Deep Q-Learning-Based Content Caching With Update Strategy for Fog Radio Access Networks', *IEEE Access*, vol. 7, pp. 97505–97514, 2019.

[26] Y. Li, M. C. Gursoy, and S. Velipasalar, 'A delay-aware caching algorithm for wireless D2D caching networks', *2017 IEEE Conf. Comput. Commun. Work. INFOCOM WKSHPS 2017*, pp. 456–461, 2017.

[27] H. Wu, J. Zhang, Z. Cai, F. Liu, Y. Li, and A. Liu, 'Towards Energy-Aware Caching for Intelligent Connected Vehicles', *IEEE Internet Things J.*, vol. 4662, no. c, pp. 1–1, 2020.

[28] J. Zhang *et al., Joint resource allocation for latency−sensitive services over mobile edge computing*

networks with caching, vol. 6, no. 3. IEEE, p. 4283−4294.

[29] I. Burbey and T. L. Martin, 'Predicting future locations using prediction-by-partial-match', *Proc. first ACM Int. Work. Mob. entity localization Track. GPS-less Environ. - MELT '08*, p. 1, 2008.

[30] S. K. Pulliyakode and S. Kalyani, 'A Modified PPM Algorithm for Online Sequence Prediction Using Short Data Records', *IEEE Commun. Lett.*, vol. 19, no. 3, pp. 423–426, 2015.