EURASIP Journal on Advances in Signal Processing

RESEARCH Open Access

Content popularity prediction for cache-enabled wireless B5G networks

Shiwei Lai[1,2], Rui Zhao[2], Yulin Wang[3*], Fusheng Zhu[1*] and Junjuan Xia[2]

*Correspondence:
wyl@gzhu.edu.cn;
zhufusheng@gdcni.cn
[1]Guangdong New Generation Communication and Network Innovative Institute (GDCNi), Guangzhou, China
[3]School of Life Science, Guangzhou University, Guangzhou, China
Full list of author information is available at the end of the article

Abstract
Inthis paper, we study the cache prediction problem for mobile edge networks where there exist one base station (BS) and multiple relays. For the proposed mobile edge computing (MEC) network, we propose a cache prediction framework to solve the problem of contents prediction and caching based on neural networks and relay selection, by exploiting users' history request data and channels between the relays and users. The proposed framework is then trained to learn users' preferences by using the users' history requested data, and several caching policies are proposed based on the channel conditions. The cache hit rate and latency are used to measure the performance of the proposed framework. Simulation results demonstrate the effectiveness of the proposed framework, which can maximize the cache hit rate and meanwhile minimize the latency for the considered MEC networks.

Keywords: Mobile edge computing, Popularity prediction

1 Introduction

Recently, the increase in the number of mobile devices (MDs) and Internet data has presented huge impacts and challenges on mobile communications [1–3]. The proposed mobile edge network effectively solves the problem of high load and low latency in the process of user-Internet interaction [4, 5] and becomes a powerful tool for improving communication quality and user experience [6–8]. In the mobile edge network scenarios, as the network edge nodes are close to the users, the system can capture and analyze the effective information from a large amount of data without uploading it to the cloud for processing. The mobile edge network can reduce network congestion and latency to ensure data timeliness and user experience. Therefore, mobile edge computing (MEC) and mobile edge caching are two important approaches in mobile edge networks.

With more and more mobile users and smart devices accessing the Internet, computation offloading is an important research area in the MEC network. Users transfer computational tasks to edge servers to reduce latency, user computational energy consumption, and system transmission costs. In [9, 10], the authors have offloaded computational tasks to edge nodes and proposed some intelligent learning based schemes to reduce system latency and energy consumption. In [11–13], the authors have considered a dynamic offloading strategy in the system of multiple users and multiple computational access nodes.

In the mobile edge network, when users send contents requests to the remote server through the central base station, a data transfer is required by the users from the base station (BS) to the remote contents-providing server. When a large number of users send requests in a short period of time, this process can cause tremendous pressure on the network and degrade user experience. By combining caching with mobile edge networks, we can reduce transmission latency and energy consumption, improve user experience, reduce repeated transmission of the same contents, and improve transmission efficiency. In [14], the authors have combined caching with mobile edge networks to investigate a cache-assisted MEC network. And the results have shown that wireless caching networks can effectively reduce transmission time, and caching can significantly mitigate the impact of increasing computational task size. In [15–17], the authors have considered the communication, caching, and computation problems in multi-user cache-assisted MEC systems and proposed a joint caching and offloading scheme.

To deal with the explosive growth of data traffic and user demand, caching becomes a promising technology in recent years and receives a lot of attention in reducing network traffic and alleviating backhaul load. Huge cache space at the BSs and relays can increase vendors costs in practice, we generally consider limited cache space at the MEC scenario. In [18], the authors have considered caching problem in MEC scenarios with limited storage. Using edge caching technology, we can shift a large amount of data traffic to the edge of the network, such as access points, small cell base stations, and mobile users. We can use various wireless access technologies to improve network spectrum efficiency, network coverage and network capacity in order to reduce communication costs. In [19], the authors have investigated the problem of contents popularity prediction in fog radio access networks and used deep learning to predict contents popularity. It has been shown that reinforcement learning can be used to find the optimal cache replacement policy with the aim of maximizing the cache hit ratio.

In this paper, we propose a cache prediction framework in the MEC network to maximize the cache hit rate and reduce latency. First, we propose a wireless MEC network with a BS and several relays to solve the problem of contents prediction and caching policy. By analyzing the communication cost, we formulate the problem as two subproblems: maximize the cache hit rate and minimize the request latency. Next, we perform contents prediction by training deep neural networks to learn the preferences of users around the BS. And then we analyze the relay-to-user channel to determine the location of requested contents and the caching policy. Finally, the simulation results demonstrate that the proposed approach can improve the cache hit rate and reduce system latency.

The outline of this article is organized as follows. We investigate the system model and formulate the problem for the considered system in Section 2. The proposed problem solution framework is described in Section 3. Section 4 presents the simulation results and conclusions are shown in Section 5.

2 System model and problem formulation

As shown in Fig. 1, we consider a wireless edge cache-enabled network, in which BS is equipped with a storage and connected to M relays $\{R_m | m = 1, 2, \ldots, M\}$ with cache space through backhaul link. Among them, the relay node covers N MDs $\{MD_n | n =$

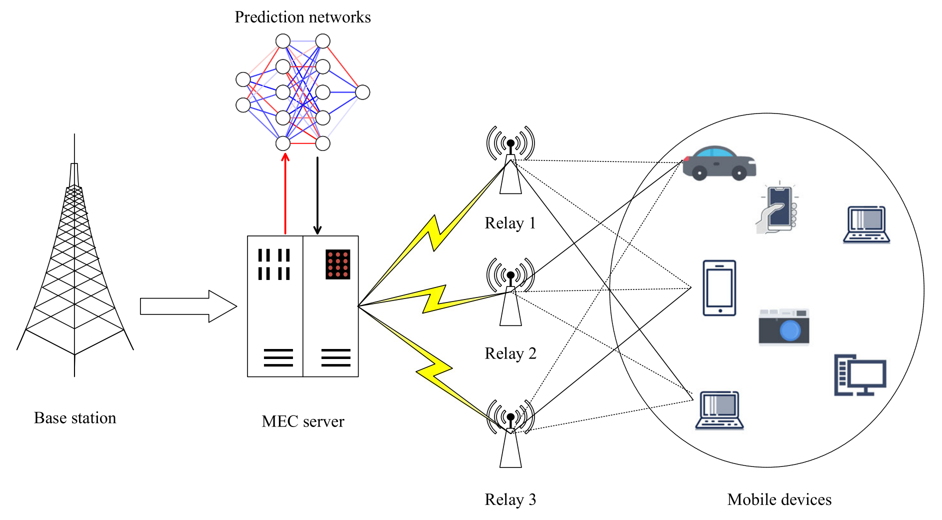

Fig. 1 System model

$1, 2, \ldots, N\}$, and the buffer space of each relay node is C. The MEC server is located in BS, which regularly predicts the popularity of files through the collected historical data and updates the caching strategies of the relay nodes.

2.1 System model

Let $\{T_i(\alpha_i, \beta_i, \gamma_i)|i = 1, 2, \ldots, I\}$ denote the information of computational task store at the BS, where α_i is the size of input computational task, β_i is the number of CPU cycles required to accomplish the task, and γ_i denotes the size of computation result of the task. In order to maximize the use of the BS and relays with limited space, and to satisfy the needs of most MDs, we need to accurately predict the request contents of MDs around the BS, and then to compute the contents in advance at the BS. Therefore, we use cache hit rate to measure the performance of prediction. In particular, the cache hit rate P_{hit} is defined as

$$P_{hit} = \frac{\sum_{n=1}^{N} \sum_{i=1}^{I} x_{n,i}}{U}, \tag{1}$$

where U is the total number of requests sent by users around the BS, and $x_{n,i}$ is the caching strategy defined as,

$$x_{n,i} = \begin{cases} 1 \text{ if the file } i \text{ result requested by the user } n \text{ is cached at the BS,} \\ 0 \text{ otherwise.} \end{cases} \tag{2}$$

The BS gets the files that users may request by predicting the file popularity, calculates these files in the edge server, and then sends the results to the corresponding relay node for caching. We assume that the data rate of the wireless link between the BS and relay R_m based on the Shannon theory is given by,

$$C_{B,m} = W_{B,m} \log_2 \left(1 + \frac{P_{B,m}|h_{B,m}|^2}{\sigma_{B,m}^2} \right), \tag{3}$$

where $W_{B,m}$ denotes the wireless bandwidth and $h_{B,m} \sim \mathcal{CN}(0, \epsilon_{B,m})$ denotes the channel gain between the BS and relay R_m [20–22]. $P_{B,m}$ is the transmit power at the BS and $\sigma_{B,m}^2$

is the variance of the additive white Gaussian noise at the BS [23–25]. Let f_B denote the computational capability of the BS, and express the computational latency as

$$L^i_{compute} = \frac{\beta_i}{f_B}. \tag{4}$$

The transmission latency, caused by the BS sending the task result T_i to the relay R_m, can be calculated as,

$$L^i_{B,m} = \frac{\gamma_i}{C_{B,m}}. \tag{5}$$

Similarly, the data rate of the wireless link between the relay R_m and the user MD_n based on the Shannon theory is given by,

$$C_{m,n} = W_{m,n} \log_2 \left(1 + \frac{P_{m,n}|h_{m,n}|^2}{\sigma^2_{m,n}}\right), \tag{6}$$

where $W_{m,n}$ denotes the wireless bandwidth and $h_{m,n} \sim \mathcal{CN}(0, \epsilon_{m,n})$ denotes the channel gain between the relay R_m and the user MD_n. $P_{m,n}$ is the transmit power at the relay R_m and $\sigma^2_{m,n}$ is the variance of the additive white Gaussian noise at the relay R_m. The transmission latency, caused by the relay R_m sending the task result T_i to the user MD_n, can be calculated as,

$$L^i_{m,n} = \frac{\gamma_i}{C_{m,n}}. \tag{7}$$

In this paper, the BS uses idle time to compute the task and transmit the result of the task to the nearby relay in advance according to the prediction in the proposed scenario. When the user request the task results, the user can get the correspond results from the nearby relay without waiting. The waiting latency is mainly caused by computation latency at the BS and transmission latency from the BS to the relay. We reduced waiting latency of the user by increasing the predictive cache hit rate. So, the latency reduction can be expressed as

$$L^i_{re} = x_{n,i}\left(L^i_{compute} + L^i_{B,m}\right). \tag{8}$$

The higher L^i_{re} indicates higher cache hit rate.

2.2 Problem formulation

The problem in this study consists of two subproblems: maximizing cache hit ratio and minimizing request latency.

2.2.1 Maximizing cache hit ratio

The problem of maximizing the cache space of the BS can be translated into the problem of maximizing cache hit rate, which can be written as

$$\max \quad P_{hit} \tag{9a}$$

$$\text{s.t.} \quad \text{C\_1}: \sum_{i=1}^{I} T_i \leq L, \tag{9b}$$

where L is the maximum cache space of BS. C_1 indicates that the number of files cached at the BS which cannot exceed the cache space limit of the BS. Similarly, P_{hit} gets higher while $L_{re} = \sum L^i_{re}$ gets higher.

2.2.2 Minimizing request latency

When there are multiple relays around the user, the BS needs to consider which relay should send the results after predictions, so that the user would take less time to get the file results. From (6), we can see that by considering the transmission channel condition and bandwidth between the relays and the user, the user MD_n would take less time to get the results of task T_i. For the considered system, the goal of latency in the process of contents placement can be expressed as

$$\min \quad L_{pl} = \sum_{m=1}^{M} \sum_{n=1}^{N} \sum_{i=1}^{I} L_{m,n}^{i}. \tag{10}$$

The conventional approach for predictive modeling involves complex feature engineering and deep analysis of the data by hand. In this paper, we aim to improve hit rate by learning user preferences through large historical data. A deep neural network (DNN) [26–28] based predictive framework needs to be developed to learn user preferences. We can feed the data directly into the networks without manual processing and can mine more information from the data. So we use DNN to solve the problem of content popularity prediction, which is given as follows.

3 The proposed problem solution method

In this paper, we propose a framework to solve the problem of the contents prediction and contents caching in MEC system. Specifically, BS adopts DNN to make cache prediction after collecting the users' information and cache the corresponding files according to the location of the users and the BS. The proposed framework is described in Algorithm 1.

3.1 Neural network for request prediction

Firstly, users need to send requests to the nearby BSs when they perform online activities, such as online browsing and shopping. The BS can observe and record historical information about the users' behaviors of sending requests. In our proposed system model, we use the BS to collect user information, which includes data of user attributes and request contents. Through the collection of these information by the BS, we can mine the users request behavior.

Secondly, we need to pre-process the user information after collecting. When there are categorical features in the users' and items' information dataset, the values of categorical features are generally discrete rather than continuous. We then need to digitize the categorical features. Generally, the categorical features are converted into one-hot encoding when dealing with such categorical features. However, because of the long categorical features, this approach has some problems, such as sparse encoding and huge input dimensions. Therefore, in this paper, when preprocessing the users and items information dataset, we convert these categorical features into numbers, which are used as the index of the embedding matrix of size (K, L), where K is the number of categorical features and L is the size of the input layer. The embedding matrix is used before the training phase to transform the input of a positive integer into a fixed size vector.

We then generate the training and testing samples based on the users and items data after pre-processing. Figure 2 shows the proposed contents prediction framework. We use two independent neural networks to extract user features $\{u_1, u_2, \ldots, u_N\}$ and file features $\{r_1, r_2, ..., r_I\}$, and then input the proposed user and file features into the next network.

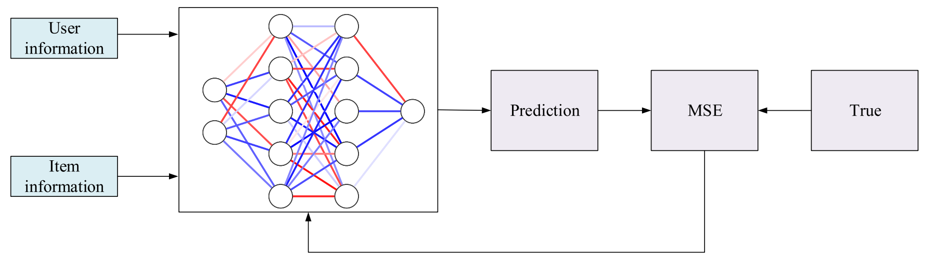

Fig. 2 The proposed contents prediction framework

Next, we obtain the feature matrices of MDs and contents of size $(1, L)$. The activation function of the output layer is ReLU which from [29] is generally defined as

$$f(x) = \begin{cases} 0, & \text{if} \quad x < 0, \\ x, & \text{otherwise.} \end{cases} \tag{11}$$

The preference values of the MDs is generated by the inner product of feature matrices. The problem in this work is that we need to predict not a pre-defined category, but an arbitrary real number. The neural networks solving regression problems generally have only one output node, and the output value of this node is the predicted value. The loss function can well reflect the gap between the trained network model and the actual data. We gradually adjust the trained network by calculating the loss function to make the loss smaller and make the prediction model more accurate. For regression problems, we use mean squared error (MSE) in this paper which is the expected value of the square of difference between the estimated value and the true value. The loss function of MSE can be obtained from [30] as

$$MSE = (y - f(x))^2, \tag{12}$$

where y is the true value and $f(x)$ predicted value of the model.

Finally, we can use the trained network to evaluate the new computational tasks. If the new tasks have high popularity, it indicates that the tasks match the preference of most users around the BS and there are high probabilities that the users will request the tasks in the future, which can be computed and cached in advance by the base station with MEC server.

In this paper, the proposed prediction framework uses the full-connection networks of DNN to learn the users preferences. The computational complexity of the proposed prediction framework is from the matrix operation of the full-connection networks. Therefore, the total computational complexity of the proposed framework is $O(L \sum_{j=1}^{J} K_{j-1}K_j)$, where L is input dimension, J is the number of network layer and K_j represents the neural size at the jth layer $(1 \leq j \leq J)$.

3.2 Caching strategy

After predicting the contents and computing the results at the BS for the considered system, we perform a relay selection to the results which come from the BS. We propose a cache policy to store the results, and then the users are able to get the requested task result in a shorter amount of time. For the task result T_i, the BS selects relay according to the channel condition between the relay and the user who requests the task T_i. When there are M relays around the user MD_n who request the task T_i, the transmission date rate can be obtained from (6) as

$$\theta_m = \max_{m\in[1,M]} \{C_{1,n}, C_{2,n}, \ldots, C_{M,n}\}. \tag{13}$$

From the set $\Theta = \{\theta_m | m = 1, 2, \ldots, M\}$, we can select a relay which has the largest θ_m among M of the $MD_n - R_m$ links according to the exhaustive search approach. We have the minimized latency for the considered system from (10) in the contents placement.

Algorithm 1 Prediction algorithm of popularity based on Neural Network

```
0: Input: user features $u_n$, file features $r_i$ and historical file popularity of corresponding
   users
0: Output: file cache policy $\Theta$
0: Initialize of neural network weight $\omega$
0: Initialize cache space $x$
0: Training process
0:   for t = 1:T do
0:     Extracting user features $u_n$ and file features $r_i$
0:     The predicted file popularity is obtained by neural network
0:     Update network parameters through MSE
0:     $MSE = (y - f(x))^2$
0:     Performing gradient descent
0:   end for
0: Save model
0: Prediction process
0: Loading model
0:   for $t_1$ = 1:N do
0:     for $t_2$ = 1:I do
0:     Forecast user $u_n$ preference for all files
0:     end for
0:   end for
0: Exhaustive search
0:   Obtain cache policy $\Theta$ according to (13)
```

4 Results and discussion

In this section, we present the performance of the proposed framework. We used the MovieLens dataset containing over 100,000 ratings from 6000 users on almost 4000 movies. Considering that the users' ratings reflect the users preference level for the movie, we use the number of high rating as the number of users who requests for the movie. A movie with higher ratings means that more users prefer this movie. High rating movies are more likely to requested in the future those of low ratings. The framework of training phase of the MovieLens dataset can be described in Fig. 3. In addition, we consider the channels following the Rayleigh flat fading in the considered system [31, 32]. We assume that all the computational tasks have the same size. The transmit power at the BS and relays are set to 10 W and 5 W, respectively. The bandwidth between the BS and relays

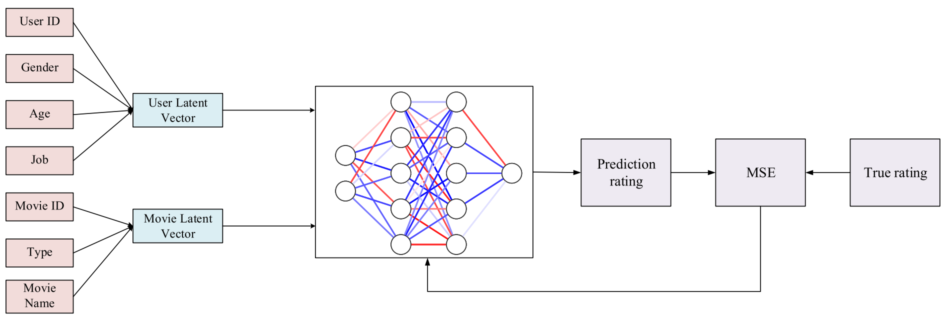

Fig. 3 The simulation contents prediction framework

is set to 40 MHz. The bandwidth between the relays and MDs ranges from 1 MHz to 10 MHz. The computation capacity of the BS f_B is set to 10×10^9 cyc/s. The parameters setting in simulation are summarized in Table 1. The training process of the proposed prediction framework is shown in Fig. 4, which can indicate that the loss decreased with the number of training steps increasing and the loss finally reaches convergence.

Figure 5 shows the performance comparison of the cache hit rate, where the cache size of the BS ranges from 50 to 500. We can observe from Fig. 5 that the cache hit rate increases as the cache size of the BS increases. The reason is that the BS can store more computation tasks and improve the probability of hitting the requests. In addition, for comparison, we plot the strategy of "Random," which indicates that the BS stores tasks randomly without considering users' preferences. The cache hit rate of the proposed framework is higher than "Random" when the number of users is 3400. For example, when the cache size is 150, the cache hit rate of the proposed framework is about twice as high as random caching. This is because that the proposed framework can learn users' preferences after training and select tasks that the users around the BS are most likely to request. When the number of users becomes smaller, more users' requests can be satisfied when there is a certain amount of cache space. So, for the proposed framework, the cache hit rate of 1700 users is higher than the users number of 4100 and 3400. On the contrary, only more cache space is available to meet more user requests when the number of users is large. These results verify that the proposed framework can predict the users' preferences accurately when cache size varies from 50 to 500.

Figure 6 shows the performance comparison of the cache hit rate, where the number of users is ranges from 500 to 3000. We can observe from Fig. 6 that the cache hit rate

Table 1 The parameters of simulation

Parameter	**Value**
Number of relays	3
Size of task α_i (MB)	30
Required CPU cycles of task β_i (10^9 cycles)	2
Result of task γ_i (MB)	3
Transmit power $P_{B,m}$(W)	10
Transmit power $P_{m,n}$(W)	5
Bandwidth $B_{B,m}$ (MHz)	40
Bandwidth $B_{m,n}$ (MHz)	[1, 10]
Computational capability f_B (10^9 cycles/s)	10

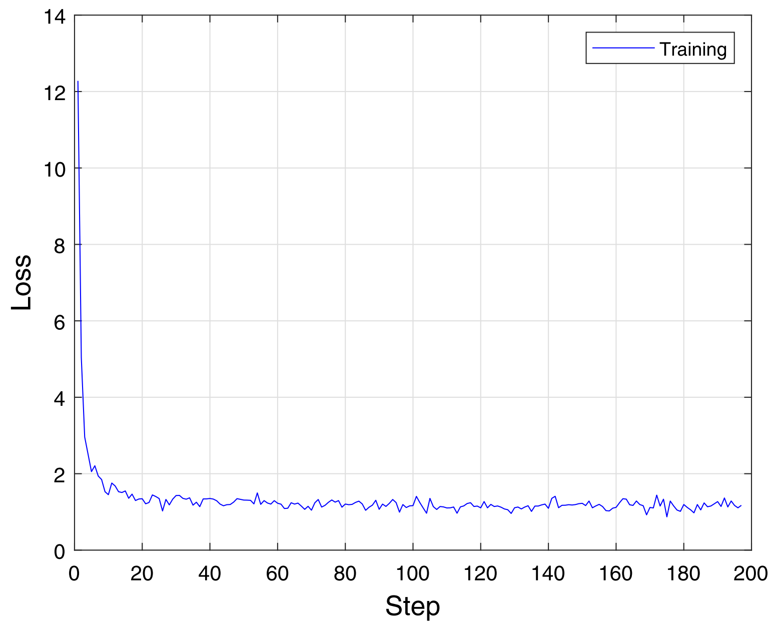

Fig. 4 The loss of the training process

increases as the number of users increases. The reason why the cache hit increases with number of users is that we cache the tasks with higher predicted popularity when the storage space is certain. The tasks with higher popularity are wanted by more users. So, we have a higher probability of hitting users' requests when the number of users is large. In addition, the hit rate of the proposed prediction strategy is still higher than that of random caching when the cache size of the BS is the same. For example, when the number of the users is 2500, the cache hit rate of the proposed prediction strategy is about 25% higher

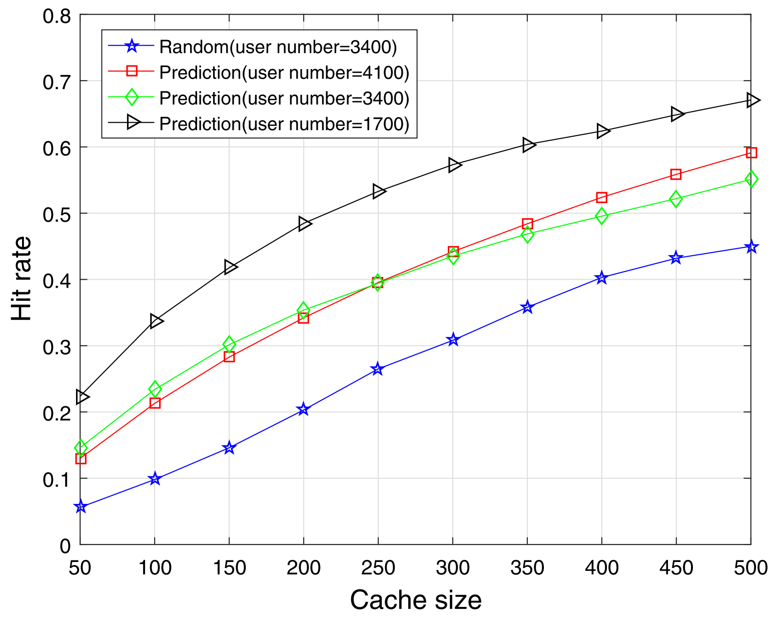

Fig. 5 Cache hit rate versus cache size of the BS

than the random strategy. This is because the BS can store the tasks with higher predicted popularity after learning. With the increase in storage space, the BS is able to store more tasks for a certain number of requests from users, and this situation increases the probability of satisfying user requests. So, when the cache sizes are 450 and 500, the cache hit rates of both schemes are higher than the scheme where the cache size is 400. This further demonstrates that the proposed strategy is effective in the process of prediction caching.

Figure 7 shows the performance comparison of the cache hit rate, where the number of tasks ranges from 500 to 1500. From Fig. 7, we observe that the cache hit rate decreases as the number of tasks increases. The reason behind this is that the number of tasks the user requests may exceed the BS's limited cache space. Moreover, when they have the same cache size of the BS, the hit rate of the proposed prediction caching is better than that of random caching. For example, when the number of tasks is 700, the cache hit rate of the proposed prediction caching is about twice as high as the random strategy. In addition, we find that the size of cache space of the BS is also related to the cache hit rate. When the cache sizes are 450 and 500, the cache hit rates of both schemes are higher than that of the scheme with cache size 400. This is because that limited cache space can only store a fixed number of tasks and satisfy a limited amount of user requests. The results prove that the cache hit rate is affected by the number of tasks and cache space.

Figure 8 demonstrates the performance of the latency reduction, where the cache size of the BS ranges from 50 to 500. For comparison, we plot the presentation of "Random" and "Without prediction," where "Random" indicates that the BS selects tasks randomly, while "Without prediction" indicates that the BS can not store tasks. We observe from Fig. 8 that the latency reduction increases as the cache size of the BS increases. This is because the BS can store more computation tasks and compute them during free time. The more tasks being stored at the BS, the more time the considered system can save. Particularly, the latency reduction of the proposed predictive caching is higher than those of both the "Random" and "Without prediction" strategies when the number of users is 3400. From Fig. 7, we can see that when the number of users is 3400 and the cache size is 200, prediction caching's L_re is about 70% higher than that of 'Random.' This is because the BS stores and computes the tasks in advance, which can reduce latency of computation and transmission. The proposed strategy can select tasks according to user preferences while "Random" can not. So, when the number of users becomes smaller, the latency reduction for the whole system is lower than that of the other prediction scheme with high users number. On the contrary, the more cache space the BS has, the more latency is reduced. These results verify that the proposed strategy can reduce the system latency effectively when the BS has different sizes of cache space.

Figure 9 demonstrates the performance of the latency reduction, where the number of users ranges from 500 to 3000. We observe from Fig. 9 that the latency reduction increases as the number of users increases. The reason is that more users means more requests sent by the users, so when the BS can store computation tasks and compute them during free time, the latency reduction of the considered system increases. Particularly, latency reduction of the proposed predictive caching is the highest when comparing with "Random" and "Without prediction" strategies, where the cache size of the BS is 400. For example, the prediction caching saves the most time when the number of users is 2500, which is about 20% higher than "Random." This is because the considered system predicts and stores the contents most likely to be requested by users, effectively reducing latency.

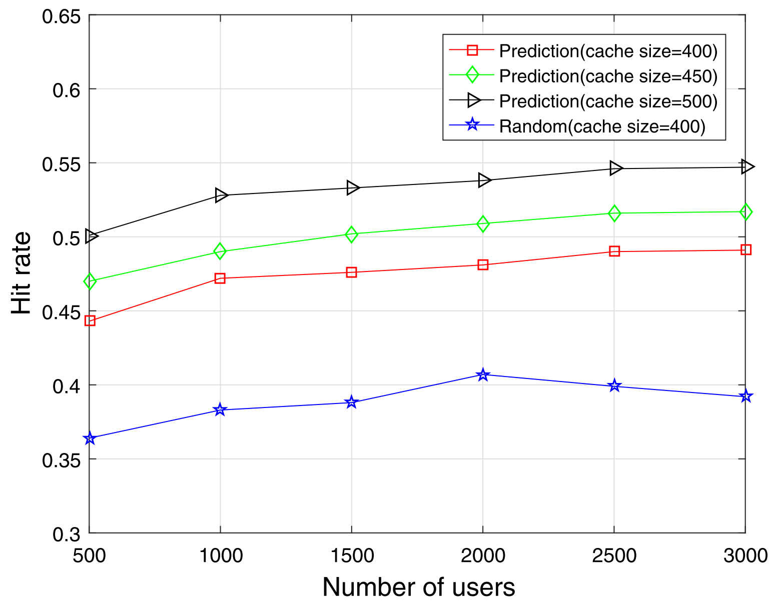

Fig. 6 Cache hit rate versus the number of MDs

In addition, when the cache size of the BS are 450 and 500, the latency reductions of both schemes are higher than that of the scheme with cache size 400. So, when the cache space of the BS becomes bigger, the latency reduction for the whole system is higher than those of the other prediction schemes with less cache space. Therefore, the more cache space the BS has, the more latency is reduced. The results prove that latency reduction is affected by the number of users and cache size.

Figure 10 demonstrates the performance of the latency reduction, where the number of tasks ranges from 500 to 1500.From Fig. 10, we observe that the latency reduction

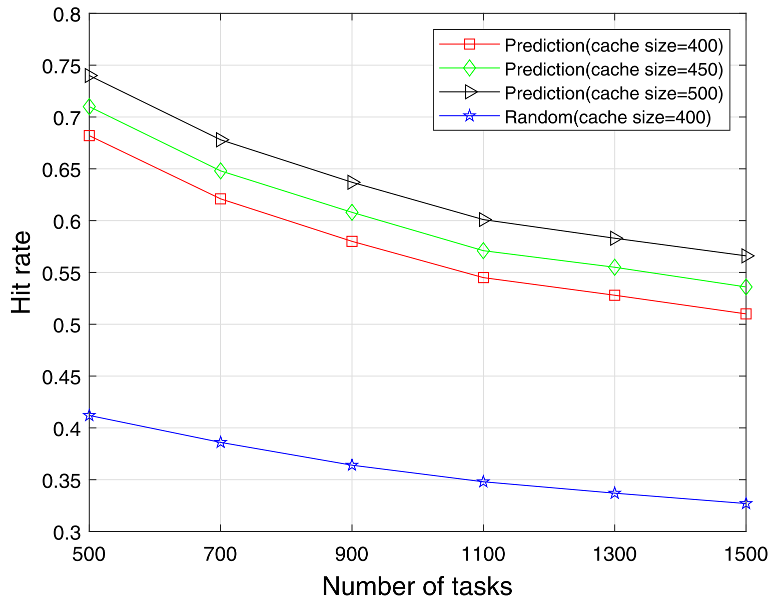

Fig. 7 Cache hit rate versus the number of tasks

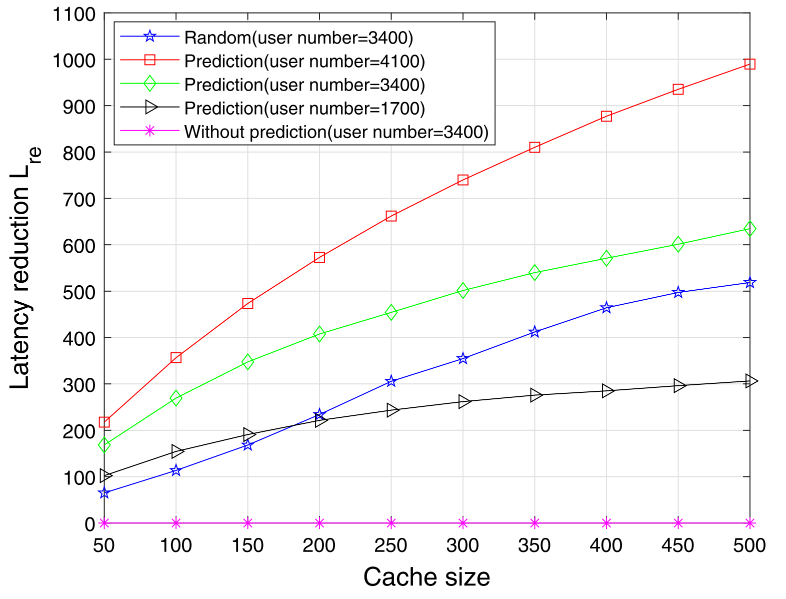

Fig. 8 Latency reduction versus cache size of the BS

increases as the number of tasks increases. The reason is that when the BS stores computation tasks and computes them in advance, the users of the considered system experience less latency to obtain the results they want. Particularly, latency reductions of "Random" and "Without prediction" strategies are lower than that of the proposed predictive caching when the cache size of the BS is 400. For example, the prediction caching saves the most time when the number of users is 1100, which is about 55% higher than "Random." This is because by predicting, storing, and computing the contents most likely to be requested by the users, the proposed system takes less amount of time. In addition, the cache size

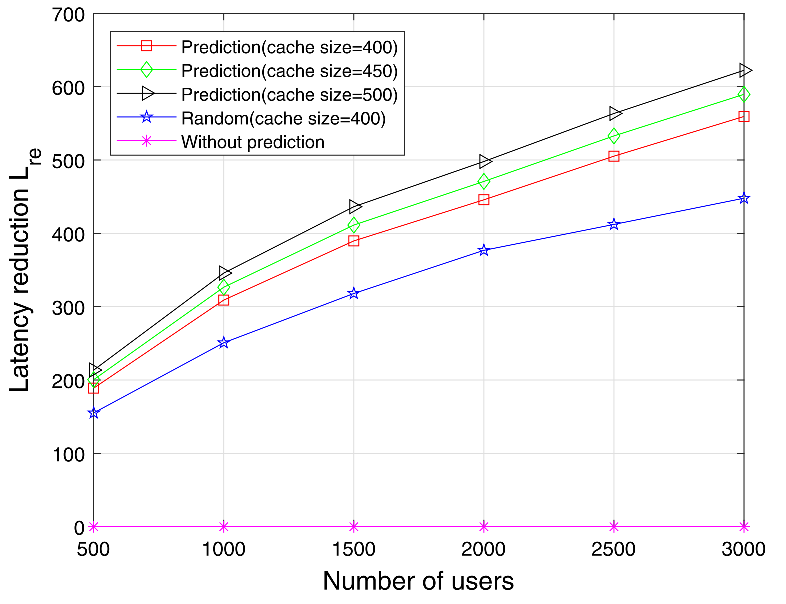

Fig. 9 Latency reduction versus the number of MDs

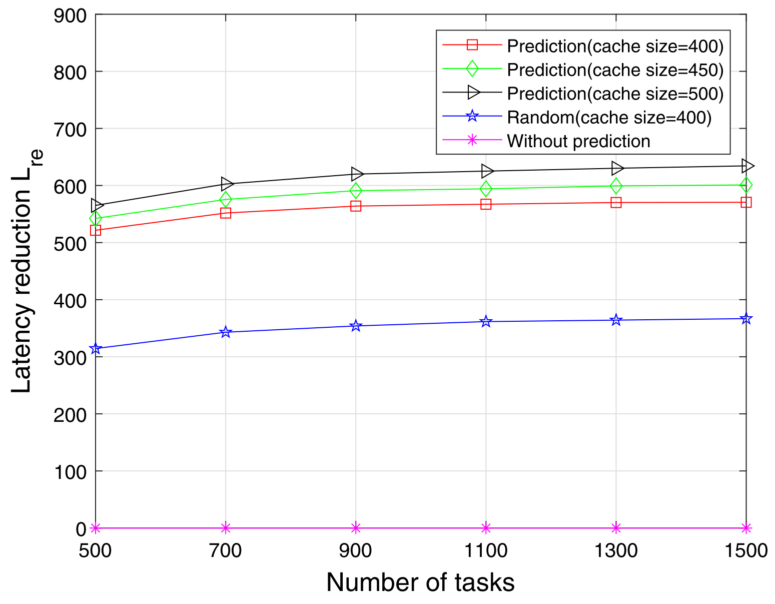

Fig. 10 Latency reduction versus the number of tasks

of the BS is related to the latency reduction. when the cache size of the BS is 400, latency reduction is lower than those of schemes with cache sizes 450 and 500, respectively. This is because bigger cache size means that more tasks are being computed at the BS after prediction, and the system can satisfy more requests sent by the users. So, the latency reduction becomes higher when the cache size becomes larger for the whole system. This further confirms that the proposed predictive caching is effective.

Figure 11 shows the performance comparison of the latency L_{pl}, where the cache size of each relay ranges from 50 to 150. For comparison, we plot the presentation of "LCD" and "Worst," where "LCD" indicates the placement strategy of largest contents diversity, while "Worst" indicates the tasks results placed on the relay with the worst channel. We can observe from Fig. 11 that the latency L_{pl} increases as the cache size of each relay increases. The reason is that the relay can store more task results and the users can get more tasks results from relay. So the considered system spends more time. In addition, the latency L_{pl} of the proposed placement strategy is lower than those of "LCD" and "Worst". Comparing these three methods, we can see that when the cache size of each relay is 110, the proposed placement strategy has the lowest cost, which is about 50% lower than "LCD" and about 70% lower than "Worst." This is because that we put the task results in the relay with the channel best connected to the intended user. These results confirm that the proposed placement strategy can reduce system latency effectively.

5 Conclusions

In this paper, we considered the problem of predictive caching in the proposed MEC network. The problem of the contents prediction and contents caching are solved by the proposed framework in the considered MEC system. We used the cache hit rate and latency to measure the performance of prediction. Specially, we used neural network for request prediction which was trained to learn users' preferences. The cache policies were obtained by the channel conditions between the relays and users. Simulation results

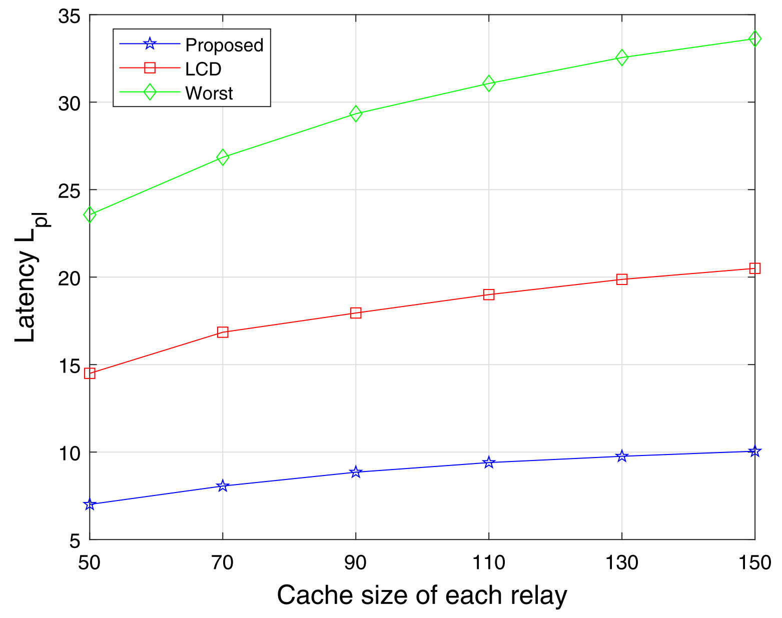

Fig. 11 Latency versus the cache size of each relay

were shown to prove that the proposed framework could improve the cache hit rate and reduce system latency. In the future, we will continue to focus on the cache prediction and explore other wireless technologies to extend our MEC model. Moreover, we will incorporate some other intelligent algorithms such as the deep learning based algorithms [33–35], or the federated learning based algorithms [36–38], into the considered systems, in order to further enhance the MEC system performance.

Abbreviations
BS: Base station; MEC: Mobile edge computing; MD: Mobile device; DNN: Deep neural network; MSE: Mean squared error; LCD: Largest contents diversity

Acknowledgements
Not applicable

Authors' contributions
S. Lai designed the proposed framework and performed the simulations, R. Zhao helped improve the caching strategy, Y. Wang helped revise the manuscript in both the structure and grammar check, F. Zhu helped enhance the design of the deep neural networks, and J. Xia was responsible for the check of the full paper and simulations. Y. Wang and F. Zhu are the corresponding authors of this paper. All authors read and approved the final manuscript.

Funding
This work is supported in part by the NSFC under grant 61871139, in part by the Innovation Team Project of Guangdong Province University under grant 2016KCXTD017, in part by the Science and Technology Program of Guangzhou under grant 201807010103, in part by the Graduate Innovative Research Grant Program of Guangzhou University under Grant 2020GDJC-M24, in part by the International Science and Technology Cooperation Projects of Guangdong Province (No. 2020A0505100060), in part by the Natural Science Foundation of Guangdong Province (No. 2021A1515011392), and in part by the research program of Guangzhou University (No. YK2020008/YJ2021003).

Availability of data and materials
The authors state the data availability in this manuscript.

Declarations

Competing interests
The authors declare that there is no conflict of interest regarding the publication of this paper.

Author details
[1]Guangdong New Generation Communication and Network Innovative Institute (GDCNi), Guangzhou, China. [2]School of Computer Science, Guangzhou University, Guangzhou, China. [3]School of Life Science, Guangzhou University, Guangzhou, China.

Received: 28 April 2021 Accepted: 23 July 2021
Published online: 14 August 2021

References
1. C. Li, J. Wang, F. Zheng, J. M. Cioffi, L. Yang, Overhearing-based co-operation for two-cell network with asymmetric uplink-downlink traffics. IEEE Trans. Signal Inf. Process. Netw. **2**(3), 350–361 (2016)
2. B. Wang, F. Gao, S. Jin, H. Lin, G. Y. Li, Spatial- and frequency-wideband effects in millimeter-wave massive MIMO systems. IEEE Trans. Signal Process. **66**(13), 3393–3406 (2018)
3. C. Li, P. Liu, C. Zou, F. Sun, J. M. Cioffi, L. Yang, Spectral-efficient cellular communications with coexistent one- and two-hop transmissions. IEEE Trans. Veh. Technol. **65**(8), 6765–6772 (2016)
4. X. Li, M. Zhao, Y. Liu, L. Li, Z. Ding, A. Nallanathan, Secrecy analysis of ambient backscatter noma systems under I/Q imbalance. IEEE Trans. Veh. Technol. **69**(10), 12286–12290 (2020). https://doi.org/10.1109/TVT.2020.3006478
5. X. Li, H. Mengyan, Y. Liu, V. G. Menon, A. Paul, Z. Ding, I/Q imbalance aware nonlinear wireless-powered relaying of B5G networks: security and reliability analysis. IEEE Trans. Netw. Sci. Eng. **1**, 1–1 (2020). https://doi.org/10.1109/TNSE.2020.3020950
6. J. Zhao, S. Ni, Multiband cooperation for 5g hetnets: a promising network paradigm. IEEE Veh. Technol. Mag. **14**(4), 85–93 (2019)
7. S. Lai, Intelligent secure mobile edge computing for beyond 5G wireless networks. Phys. Commun. **45**(101283), 1–8 (2021)
8. J. Zhao, X. Sun, Q. Li, X. Ma, Edge caching and computation management for real-time internet of vehicles: an online and distributed approach. IEEE Trans. Intell. Transp. Syst. **22**(4), 2183–2197 (2021)
9. R. Zhao, Deep reinforcement learning based mobile edge computing for intelligent internet of things. Phys. Commun. **43**, 1–7 (2020)
10. Y. Guo, Efficient and flexible management for industrial internet of things: a federated learning approach. Comput. Netw. **192**, 1–9 (2021)
11. L. Chen, Intelligent ubiquitous computing for future UAV-enabled MEC network systems. Clust. Comput. **99**, 1–8 (2021)
12. C. Li, J. Xia, F. Liu, D. Li, L. Fan, G. K. Karagiannidis, A. Nallanathan, Dynamic offloading for multiuser muti-cap MEC networks: a deep reinforcement learning approach. IEEE Trans. Veh. Technol. **70**(3), 2922–2927 (2021)
13. Y. Guo, S. Lai, Distributed machine learning for multiuser mobile edge computing systems. IEEE J. Sel. Top. Signal Process. **99**, 1–12 (2021)
14. J. Xia, Secure cache-aided multi-relay networks in the presence of multiple eavesdroppers. IEEE Trans. Commun. **67**(11), 7672–7685 (2019)
15. X. Lai, Secure mobile edge computing networks in the presence of multiple eavesdroppers. IEEE Trans. Commun. **PP**, 1–12 (2021)
16. W. Zhou, PSO based offloading strategy for cache-enabled mobile edge computing UAV networks. Clust. Comput. **99**, 1–8 (2021)
17. X. Lai, Cybertwin-driven mobile edge computing for internet of everything with cochannel interference. IEEE Trans. Ind. Inf. **99**, 1–12 (2021)
18. M. K. Somesula, R. R. Rout, D. V. L. N. Somayajulu, Contact duration-aware cooperative cache placement using genetic algorithm for mobile edge networks. Comput. Netw. **193**, 108062 (2021). https://doi.org/10.1016/j.comnet.2021.108062
19. H. Feng, Y. Jiang, D. Niyato, F. Zheng, X. You, in *2019 IEEE Global Communications Conference, GLOBECOM 2019, Waikoloa, HI, USA, December 9-13, 2019*, Content popularity prediction via deep learning in cache-enabled fog radio access networks (IEEE, New York, 2019), pp. 1–6
20. X. Hu, C. Zhong, Y. Zhu, X. Chen, Z. Zhang, Programmable metasurface-based multicast systems: design and analysis. IEEE J. Sel. Areas Commun. **38**(8), 1763–1776 (2020)
21. X. Hu, C. Zhong, Y. Zhang, X. Chen, Z. Zhang, Location information aided multiple intelligent reflecting surface systems. IEEE Trans. Commun. **68**(12), 7948–7962 (2020)
22. X. Hu, J. Wang, C. Zhong, Statistical CSI based design for intelligent reflecting surface assisted MISO systems. Sci China: Inf. Sci. **63**(12), 222303 (2020)
23. K. He, Learning based signal detection for MIMO systems with unknown noise statistics. IEEE Trans. Commun. **69**, 3025–3038 (2021)
24. G. Gui, F. Liu, J. Sun, J. Yang, Z. Zhou, D. Zhao, Flight delay prediction based on aviation big data and machine learning. IEEE Trans. Veh. Technol. **69**(1), 140–150 (2020)
25. Y. Zhang, J. Wang, CV-3DCNN: complex-valued deep learning for CSI prediction in FDD massive MIMO systems. IEEE Wirel. Commun. Lett. **10**(2), 266–270 (2021)
26. K. He, Ultra-reliable MU-MIMO detector based on deep learning for 5G/B5G-enabled IoT. Phys. Commun. **43**, 1–7 (2020)
27. Z. Wang, An adaptive deep learning-based UAV receiver design for coded MIMO with correlated noise. Phys. Commun. **45**(101295), 1–8 (2021)
28. J. Xia, D. Deng, D. Fan, A note on implementation methodologies of deep learning-based signal detection for conventional MIMO transmitters. IEEE Trans. Broadcast. **66**(3), 744–745 (2020)
29. X. Glorot, A. Bordes, Y. Bengio, in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*. ed. by G. J. Gordon, D. B. Dunson, and M. Dudík, Deep sparse rectifier neural networks, vol. 15 (JMLR.org, London, 2011), pp. 315–323

30. J. Heaton, Ian Goodfellow, Yoshua Bengio, and Aaron Courville: deep learning. Genet. Program Evolvable Mach. **19**(1-2), 305–307 (2018)
31. J. Zhang, Y. Zhang, C. Zhong, Z. Zhang, Robust design for intelligent reflecting surfaces assisted MISO systems. IEEE Commun. Lett. **24**(10), 2353–2357 (2020)
32. Q. Tao, J. Wang, C. Zhong, Performance analysis of intelligent reflecting surface aided communication systems. IEEE Commun. Lett. **24**(11), 2464–2468 (2020)
33. J. Xia, L. Fan, Computational intelligence and deep reinforcement learning for next-generation industrial IoT. IEEE Trans. Netw. Sci. Eng. **99**, 1–12 (2021)
34. J. Yang, D. Ruan, J. Huang, X. Kang, Y. Shi, An embedding cost learning framework using GAN. IEEE Trans. Inf. Forensics Secur. **15**, 839–851 (2020)
35. M. Liu, F. Tang, 6G: opening new horizons for integration of comfort, security, and intelligence. IEEE Wirel. Commun. **27**(5), 126–132 (2020)
36. S. Tang, Dilated convolution based CSI feedback compression for massive MIMO systems. IEEE Trans. Veh. Technol. **99**, 1–5 (2021)
37. Z. Zhao, System optimization of federated learning networks with a constrained latency. IEEE Trans. Veh. Technol. **99**, 1–5 (2021)
38. S. Tang, Battery-constrained federated edge learning in UAV-enabled IoT for B5G/6G networks. Phys. Commun. **47**(101381), 1–9 (2021)

Publisher's Note