A Predictive and Trajectory-Aware Edge Service Allocation Approach in a Mobile Computing Environment

Ling Huang, Chongqing Institute of Engineering, China*
Bin Shuai, Chongqing University, China

ABSTRACT

The mobile edge computing (MEC) model is featured by the ability to provision elastic computing resources close to user requests at the edge of the internet. This paradigm moves traditional digital infrastructure close to mobile networks and extensively reduces application latency for mobile computing tasks like online gaming and video streaming. Nevertheless, it remains a difficulty to provide effective and performance-guaranteed edge service offloading and migration in the MEC environment. Most existing contributions in this area consider task offloading as an offline decision-making process by exploiting transient positions of mobile requesters as model inputs. In this work, instead, the authors develop a predictive-trajectory-aware and online MEC task offloading strategy. Simulations based on real-world MEC deployment datasets and a campus mobile trajectory datasets clearly illustrate that this approach outperforms state-of-the-art ones in terms of effective service rate and migration overhead.

KEYWORDS

Edge Computing, Mobility, Service Allocation

1. INTRODUCTION

With the booming of Internet of Things (IoT) and mobile communication technologies, mobile edge (MEC) terminals and nodes are becoming increasingly popular in provisioning computational resources and accessibility to application requesters. MEC supports varying types of computation-intensive applications, such as graphical guided system based on Artificial Intelligence (AI), video gaming, augmented vehicular reality system (Lai, 2018). It (Beck, 2014) helps to transfers the computing task load from the centralized clouds to the edge of the network close to the requesters (Abbas, 2018). In the MEC paradigm, base stations are equipped with a certain amount of computing infrastructures. Thus, they are responsible for both wireless communication and task execution. Mobile requesters are allowed to offloading computational-intensive tasks to the MEC nodes, in this way, requesters can be access these service directly without performing in thire own device or resorting to remote clouds. And that are featured by much less communication overhead and energy consumption than traditional clouds (Xu, 2020 & Chen, 2019).

However, various difficulties, especially the real-time resource allocation, are still to be properly addressed. In a typical edge computing paradigm, MEC servers are usually enhanced with computing components and storage (Wu, 2019). Usually, mobile requesters can move away
from of the communication coverage of a certain server which they previously contact and lose the communication connection. In case of a connection loss, a requester has to re-contact another MEC node and probably experiences service interruptions which potentially affect to user-perceived quality-of-service (QoS) (Peng, 2019). Therefore, QoS-guaranteed allocation of requesters to suitable MEC nodes with optimized requester coverage rate and minimized reallocation overhead becomes a critical issue.

Most of traditional methods (Lai, 2018 & Peng, 2019 & Yang, 2017 etc.) in this direction are still limited due to the fact they consider offline offloading decision making by employing transient requester positions as model inputs. However, such methods can be ineffective due to the fact that real-world edge requesters are often with high mobility and the offloading actions are thus supposed to be decided in a dynamic way. Instead of considering instantaneous and transient positions of mobile requesters, in this paper, we consider continuing requester trajectories as model inputs and propose a predictive-trajectory-aware online service allocation method. We conduct extensive simulations as well and illustrate that our proposed method outperforms traditional ones in terms of effective service rate and migration overhead.

The remaining part of this paper is organized as follows. Section 2 describes existing methods of task offloading and service allocation in the MEC environment. Section 3 describes and models the service allocation problem. Section 4 presents our method and the simulative results are illustrated in section 5. Finally, in the last section we present some conclusive remarks.

2. RELATED WORK

Task offloading and allocation problems are receiving increasing research attention in the area of edge computing. For example, Chen et al. (Chen, 2016) formulated the distributed offloading-decision-making problem into a multi-user offloading game and designed a efficient resource offloading model based on game theory for achieving Nash equilibrium between energy efficiency and offloading performance. Chen et al. (Chen, 2019) considered task offloading and migration problem of service function chain under multiple constraints and solved the problem through two phases, namely, using a dynamic programming algorithm for yielding the initial packing solutions and using a heuristic method for yielding the final offloading decisions based on the initial packing plans. Yang et al. (Yang, 2020) considered edge task offloading with varying priorities and developed a multi-task-learning-based feed forward neural-network model for joint optimization of task offloading solutions. Zhang et al. (Zhang, 2020) aimed at optimizing network energy efficiency and developed a stochastic mixed-integer nonlinear programming problem and Lyapunov-theory-based strategy for deciding allocation of both computational and radio resources in a dense MEC environment. Xue et al. (Xue, 2021) proposed a scheduling strategy for joint task offloading and resource allocation in multi-requester, multi-task and multi-node MEC environment. They aimed at optimizing system processing capability and developed a problem-decomposition-based approach that is capable of solving subproblems through the Lagrange multiplier method.

Hu et al. (Hu, 2019) aimed at minimizing communication energy consumption and combined the optimal stop theory with the migration decision for yielding dynamic offloading decisions. Zhang et al. (Zhang, 2020) formulate the allocation problem into a Modulo Satisfiability problem and propose a deep Q-learning algorithm for yielding adjustable allocation paths. Wu et al. (Wu, 2021) considered a dynamic requesters arrival process and developed a distributed reactive approach by employing a fuzzy control mechanism for yielding the real-time allocation decisions. Huang et al. (Huang, 2020) used a binary offloading policy for generating high-quality offloading and wireless resource allocation decisions for wireless channels with time-varying performance. They considered a reinforcement-learning-oriented online offloading framework and illustrated that their approach is highly effective in terms of time complexity.
A careful investigation into the aforementioned studies illustrates that current works are still limited in many aspects: 1) most existing methods consider MEC task offloading as an offline decision making process and leverage transient positions of requesters as model inputs. In this way, they ignore the intrinsic dynamicness of the MEC environment, especially the mobility of requesters; 2) they usually assumed a simultaneous batch-arrival pattern of incoming edge requesters, formulated the EUA problem to an optimization problem with constraints, and thus solved it in a static optimization way. However, the real-world pattern could be arbitrary and general in terms of the inter-arrival time and geographical location, the traditional batch-processing-based allocation process might result in additional waiting time.

3. PROBLEM FORMULATION AND SYSTEM MODEL

3.1 Mobile Edge Environment

A typical MEC environment comprises base stations enhanced with computing facilities. Mobile application providers are allowed to offload their computational tasks on these MEC nodes. Mobile requesters are often with mobility and thus their applications should be migratable to guarantee that the resource provisioning is not interrupted.

Figure 1. The scenario of requester moving

As illustrated in Fig.1, requester $u_1$ moves from the communication range of server $s_2$ to $s_1$ when he moves from point $u_1^k$ to point $u_1^n$. $u_1$ is unreachable from $s_2$ when at point $u_1^n$. Thus, the application on $u_1$ is supposed to be moved to $s_1$ before leaving the communication coverage of server $s_2$. $u_1$ can choose to migrate the application to server $s_3$ when $u_1$ is between $u_1^l$ and $u_1^k$, and migrate to server $s_1$ when $u_1$ is between $u_1^k$ and $u_1^m$; or to migrate to server $s_1$ when requester $u_1$ is between $u_1^k$ and $u_1^l$. However, different migration activities may illustrate varying performance. Thus, a smart algorithm for choosing an appropriate migration plan in terms of low migration overhead is in high need.
Table 1. Notation and description

| Notation | Description |
|----------|-------------|
| $U$      | The set of mobile requesters |
| $n$      | The number of requesters |
| $u_i$    | The $i_{th}$ requester in $U$ |
| $u_i^t$  | The location of $u_i$ at time $t$ |
| $\text{start}_i$ | The first time when requester $u_i$ request service |
| $S$      | The set of edge node stations, i.e., MEC server set |
| $m$      | The number of servers |
| $s_j$    | The $j_{th}$ edge server in $S$ |
| $R_j$    | The communication range of edge server $s_j$ |
| $C_j$    | The capacity of edge server $s_j$ |
| $X$      | The history trajectory of requester |
| $X_i^t$  | The historical moving trajectory of $u_i$ at time $t$ |
| $Y_i^t$  | The real future trajectory of $u_i$ at time $t$ |
| $\hat{Y}_i^t$ | The predicted trajectory of at time $t$ |
| $T$      | The working period |
| $T_k$    | The $k_{th}$ time slice |
| $N_{T_k}$ | The set of online requesters in time slice $T_k$ |
| $[u_{i,T_k}]$ | The online time of requester $u_i$ in time slice $T_k$ |
| $\lambda_i^t$ | A boolean indicator of whether $u_i$ is inline at time $t$ |
| $\beta_{i,j}^t$ | A boolean indicator of whether $u_i$ is severed by $s_j$ at time $t$ |
| $\gamma_i^t$ | A boolean indicator of whether $u_i$ is moved at time $t$ |

Table 1 continued on next page
3.2 The Model of Trajectory Prediction

For a mobile requester \( u \) at time \( t \), \( X^t = \left( p^s_{start}, \ldots, p^{t-1}, p^t \right) \) indicates its historical moving trajectory before time \( t \), where \( p^k = (x^k, y^k) \), and \( Y^t = \left( p^{t+1}, p^{t+2}, \ldots, p^{t+pred} \right) \) denotes his actual moving trajectory after time \( t \). We need to obtain a corresponding predicted trajectory that is as close to \( Y^t \) as possible. Assume that \( \hat{Y}^t = \left( \hat{p}^{t+1}, \hat{p}^{t+2}, \ldots, \hat{p}^{t+pred} \right) \) denotes the predicted trajectory, where \( \hat{p}^k = (\hat{x}^k, \hat{y}^k) \), the Euclidean distance between \( Y^t \) and \( \hat{Y}^t \) should be minimized:

\[
\text{Min } \sum_{t=1}^{pred} \text{dis}\left(p^t, \hat{p}^t\right) \text{ the } \text{dis}\left(p^t, \hat{p}^t\right) \text{ is subject to:}
\]

\[
\text{dis}\left(p^t, \hat{p}^t\right) = \sqrt{(x^t - \hat{x}^t)^2 + (y^t - \hat{y}^t)^2}
\]

3.3 The Model of Allocation and Migration

The problem of predictive edge service allocation can be described as the following formulation: there are \( m \) base stations \( S = s_1, s_2, \ldots, s_m \) in the edge computing environment, each of which is equipped with an amount of computing and storage facilities; \( R_j \) is the communication range of station \( s_j \), \( C_j \) the maximum capacity of requesters that can be served for MEC node \( s_j \); these are
mobile \( n \) requesters \( U = u_1, u_2, \ldots, u_n \) and the \( i_{\text{th}} \) requester initiate a service request at time \( \text{strat}_i \) take a service request. At any time slice \( T_k' \), the set of requesters requesting service are \( u_{r_i} \), for each \( u_i \in U_{T_k} \) it follows that \( i \in N_{T_k} \). Total time \( T \) is divided into a series of time slice \( r = r_1, r_2, \ldots, r_k \); \( |u_{r_i}| \) means the number of track points of requester \( u_i \) in time slices \( T_k' \).

To guarantee that mobile requesters are served with continuing edge resource provisioning, we aim to optimize the service time for requesters and reduce the migration counts. In each time slice \( T_k' \):

\[
\begin{align*}
\text{Max} : & \sum_{t \in T_k} \sum_{j=0}^{m} \sum_{i \in N_{T_k}} \left[ u \right]_{i_t, j_t} \\
\text{Min} : & \sum_{t \in T_k} \sum_{j=0}^{m} \sum_{i \in N_{T_k}} \left[ u \right]_{i_t, j_t}
\end{align*}
\]

s.t:
\[
\begin{align*}
\sum_{i \in N_{T_k}} 2_{i, j} & \leq C_j \\
d_{i, j} \leq R_j & \text{ if } 2_{i, j} = 1
\end{align*}
\]

where \( \Rightarrow \) is a boolean indicator of whether \( u_i \) is inline at time \( t \), \( \Rightarrow \) a boolean identifier of whether \( u_i \) is served by node \( s_j \) at time \( t \), \( \Rightarrow \) a boolean indicator of whether \( u_i \) is moved at time \( t \), \( d_{i, j} \) the distance of \( u_i \) and \( s_j \) at time \( t \). As illustrated in (3) and (4), the optimization problem aims at both optimizing the average requester service rate and minimizing the counts of migration. (5) is the capacity constraint, and (6) is the geographical location constraint.

4. PREDICTIVE-TRAJECTORY- AWARE ALLOCATION

Instead of considering static allocation, we develop a dynamic and predictive-trajectory-aware allocation framework, which comprises a multi-sliding-window-polynomial-trajectory-prediction method, named \textit{MSWPP}, and a dynamic service scheduling and migration strategy, named \textit{PreMig}.

For the trajectory prediction purpose, we leverage a polynomial function for fitting historical moving traces and multi sliding windows for prediction of future trajectories. \textit{MSWPP} is based on the least square method for fitting the requester history trajectory into a polynomial function. There are \( k \) different points \( (x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k) \), we need to identify a polynomial \( Px \) of degree \( mm < k - 1 \), such that \( P(x_i) \) as close to \( y_i, i = 1, 2 \ldots, k \). Assuming the polynomial is:
4.1 TRAJECTORY PREDICTION

\[ P(x) = a_0 + a_1 x + \ldots + a_k x^k = \sum_{j=0}^{k} a_j x^j \quad (m < k - 1) \]

Take \( k \) points into polynomials:

\[
\begin{align*}
 a_0 + a_1 x_1 + a_2 x_1^2 + \ldots + a_k x_1^m - y_1 &= R_1 \\
 a_0 + a_1 x_2 + a_2 x_2^2 + \ldots + a_k x_2^m - y_2 &= R_2 \\
 \vdots && \vdots \\
 a_0 + a_1 x_k + a_2 x_k^2 + \ldots + a_k x_k^m - y_k &= R_k \\
\sum_{j=0}^{m} a_j x_i^j - y_i &= R_i \quad (i = 1, 2, \ldots, k)
\end{align*}
\]

The resulting formulation is:

\[
\sum_{j=0}^{m} a_j x_i^j - y_i = R_i \quad (i = 1, 2, \ldots, k)
\]

Note that \( R_i \quad (i = 1, 2, \ldots, k) \) can be greater than zero, thus the least square method can be used to determine appropriate \( a_j \quad (j = 0, 1, \ldots, k) \) for minimizing \( A \):

\[
\sigma = \sum_{i=1}^{k} R_i^2 = \sum_{i=0}^{k} \left( \sum_{j=0}^{m} a_j x_i^j - y_i \right)^2
\]

The solution of above formula can be obtained by solving matrix \( MA = N \),

\[
M = \begin{bmatrix}
1 & \cdots & x_1 & x_1^m \\
1 & \cdots & x_2 & x_2^m \\
\vdots & \ddots & \vdots & \vdots \\
1 & \cdots & x_k & x_k^m
\end{bmatrix}, A = \begin{bmatrix}
a_0 \\
a_1 \\
\vdots \\
a_k
\end{bmatrix}, N = \begin{bmatrix}
y_0 \\
y_1 \\
\vdots \\
y_k
\end{bmatrix}
\]

where \( M^T \) is the transpose matrix of \( M \), \( M^T MA = M^T N \) and \( M^T M = W \), when \( W \neq 0 \), we have that:

\[
W^{-1} WA = W^{-1} M^T N, \quad A = W^{-1} M^T N.
\]

In this way, the coefficient matrix \( A \), e.t., the coefficient \( a_j \quad (j = 1, 2, \ldots, k) \) of polynomial \( P(x) \) can be obtained.

As can be seen in Algorithm 1, MSWPP, it takes latitude and longitude coordinate of requester’s history trajectory as inputs, and leverages polynomial fitting functions \( P_{LAT}(x) \) and \( P_{LON}(x) \) obtained through (7–11) for predicting the further coordinates of \( P_{LAT}(k + i), P_{LON}(k + i) \).

The detailed steps of the MSWPP algorithm is given below. As can be seen, the step size of sliding windows is variable, where short ones are for capturing recent-most motions and longer ones are for predicting long-term motions.

Algorithm 1: MSWPP \( \mu \)
Input: history trajectory \( X \); predict length \( L \); distance control threshold \( \eta \); reduction factor \( \cdot \); windows numbers \( W \); max windows length \( l_{\text{max}} \).

Output: the predict trajectory \( \hat{Y} \)

1. \( \text{Step} = \{1:1,2:2,3:5\} \) set the window-step mapping dictionary;

2. \( L = \min\left(\text{len}(X)/5,l_{\text{max}}\right) \) get the length of sliding window;

3. \( D_{\text{avg}} \) calculate the average distance of the requester’s most recent movement by sampling the nearest \( l \) points in \( X \);

4. \( Y_{\text{dict}} = \{\} \);

5. FOR \( w=W \) TO 1 DO

6. \( s = \text{step}[w] \);

7. \( X' \) sampling \( X \) in reverse order with \( s \) as the step size;

8. \( \text{LAT}, \text{LON} \) get latitude and longitude data according to \( X' \);

9. \( P_{\text{LAT}}(x), P_{\text{LON}}(x) \) obtain the fitting function calculated by Eqs.(7–11);

10. \( \text{Last } \_ \text{point} \) the last track point of \( X \);

11. \( x = l \) set a polynomial variable used to calculate the predicted value;

12. FOR \( i=1 \) TO \( L \) DO

13. \( t=1 \) set time step size of prediction;

14. \( \text{Pred } \_ \text{point} = (P_{\text{LAT}}(x+t), P_{\text{LON}}(x+t)) \);

15. WHILE \( \text{dis} (\text{last } \_ \text{point, pred } \_ \text{point}) > \eta D_{\text{avg}} \) DO

16. \( t = \mu t \);

17. \( \text{pred } \_ \text{point} = (P_{\text{LAT}}(x+t), P_{\text{LON}}(x+t)) \);

18. \( Y_{\text{dict}}[s*t] = \text{pred } \_ \text{point} \);

19. \( \text{Last } \_ \text{point} = \text{pred } \_ \text{point} \);

Algorithm 1 continued on next page
Pseudo codes in 15-19 are intended for appropriately handling intervals of predicted trace points, through introducing a threshold $\eta$ and a reduction factor $\mu$. For doing so, if distance between two contiguous predict trace point $\text{dis}(X_i, X_j)$ calculated by Eq.(2) is greater than $\eta D_{avg}$, the corresponding step size is shrunk according to the factor of $\mu$ and thus the distance interval of between predicted trace points falls below $\eta D_{avg}$.

4.2 Allocation and Migration Strategy

As illustrated in Algorithm 2, PreMig has two parts: 1) identifying the overloaded node in the current environment every $T_{\text{flash}}$ time, and reallocate requesters within each of these nodes. It avoids over-
frequent updates of edge node status and checks node status with intervals of $T_{\text{flash}}$. 2) allocating new requesters and unallocated ones. It reallocate requesters that are within the coverage range of an edger server, by moving out requesters with low stay time and moving in unallocated requesters with long stay time, when nodes are overloaded.

Algorithm 3: Reallocation

Input: $s \rightarrow$ a overloaded node; $U \rightarrow$ the requesters set;

1. $X \rightarrow$ get the requesters accessible to node $s$;
2. $X' \rightarrow$ get requesters allocated in $s$ and haven’t other available node to migrate from $X'$;
3. IF $|X'| = s.capacity$ THEN

4. $Temp = []$;

5. FOREACH $u \in (X - X')$ DO

6. $\hat{Y} \rightarrow$ get predict trajectory of $u$ by MSWPP;
7. $w \rightarrow$ calculate the stay time of $\hat{Y}$ in $s$;
8. $temp.append((u, w))$
9. sort temp according to $w$ value in a descending order;
10. $L = s.capacity - |X'|$

11. FOR $i = temp.length$ TO 1 DO

12. $u' = temp[i]$;
13. IF $i > L$ THEN

14. IF $u'$ is allocated in other node $s$ THEN
15. move out the service of $u'$ from node $s$

16. IF $i \leq L$ THEN

17. IF $u'$ is unallocated THEN
18. allocated request $u'$ to node $s$
19. IF $u'$ is allocated in other node THEN

20. migrate the service of $u'$ form other node to $s$
The reallocation procedure is given in Algorithm 3: 1) finding all requesters within the communication range of current overloaded node and those that are allocated but unable to be moved out (pseudo codes in 1-2); 2) for each requester being reallocated, predicting its future trajectory by using MSWPP, and calculate the stay time $w$ (pseudo codes in 5-9); 3) sorting these requesters in a descending order, and then choosing the first $L$ requesters for reallocation (pseudo codes in 10-16); 4) for requesters allocated in the current overloaded node and ranked under $L$, migrating them out and invoking AllocateSingleUser to further handling (pseudo codes in 17-19).

Algorithm 4: AllocateSingleUser

| Input: $u$ unallocated requester; $Su$ MEC server set; |
|---------------------------------------------------------|
| 1. $S' \sim$ get the accessible MEC servers for requester $u$ at current time $t$; |
| 2. $S' \sim$ remove the overload server in $S'$; |
| 3. if is not empty then $S'$ |
| 4. $\hat{Y} \sim$ get predict trajectory of $u$; |
| 5. $temp = [ ]$ |
| 6. FOREACH $s \in S'$ DO |
| 7. $w \sim$ get the stay time of $\hat{Y}$ in $s$; |
| 8. $temp.append((s,w))$ |
| 9. sort temp according $w$ value in a descending order; |
| 10. $s' = temp[0][0]$ |
| 11. allocate requester $u$ to node $s'$ |

Figure 2. Mobile trajectories and edge server deployment
5. EXPERIMENT AND ANALYSIS

In this section, we leverage a mobile trajectory dataset obtained in a real campus environment of 1km² and the EUA dataset (Lai, 2018 & Peng 2019 & Ma, 2020 & Liu 2019). The trajectory dataset includes mobile traces of 500 students within 30 minutes. Fig.2 illustrates a part of such trajectories and the positions of 59 edge servers.

The simulation incorporates 3 different scenarios:

- Low-density: The total number of requesters is 500.
- Medium-density: The total number of requesters is 800.
- High-density: The total number of requesters is 1200.

The occurrence times of service requests are assumed to be uniformly distributed. In contrast, the service times of requesters are stochastic. The detailed requester information of three scenarios are exhibited in Figs.3,5,7. The occurrence times distribution of service request is illustrated in its subgraph (a), the service times distribution of requesters illustrate in its subgraph (b), the number distribution of requesters at the same time illustrate in its subgraph (c), the speed distribution of requesters illustrate in its subgraph (d).

Figure 3. The requester detail information in low-density Occurrence times distribution of service request

![Figure 3. The requester detail information in low-density Occurrence times distribution of service request](image-url)
Figure 4. The average service rate and migrate count in low-density: The average service rate

(a) The average service rate

(b) Migrate count

Figure 5. The requester detail information in medium density: Occurrence times distribution of service request

(a) Occurrence times distribution of service request

(b) Number distribution of requesters at the same time

(c) Service times distribution of requesters

(d) Speed distribution of requesters
Figure 6. The average service rate and migrate count in medium-density: The average service rate

(a) The average service rate

(b) Migrate count

Figure 7. The requester detail information in high-density: Occurrence times distribution of service request

(a) Occurrence times distribution of service request

(b) Number distribution of requesters at the same time

(c) Service times distribution of requesters

(d) Speed distribution of requesters
Figure 8. The average service rate and migrant count in high-density: The average service rate

(a) The average service rate

(b) Migrate count

Table 2. Experimental data comparison of different service allocation strategies in edge environment with different requester densities

| Requester density | Algorithm | Total service time | Average service rate | Migration counts |
|-------------------|-----------|--------------------|----------------------|------------------|
| Low               | PreMig    | 133898             | 99.16                | 1458             |
|                   | Greedy    | 133707             | 99.06                | 2112             |
|                   | MobMig    | 133783             | 99.10                | 1570             |
|                   | BP        | 133741             | 99.09                | 3012             |
|                   |           |                    |                      |                  |
| Medium            | PreMig    | 197057             | 91.48                | 3468             |
|                   | Greedy    | 196988             | 91.47                | 4129             |
|                   | MobMig    | 196815             | 91.27                | 3595             |
|                   | BP        | 196373             | 91.36                | 5581             |
|                   |           |                    |                      |                  |
| High              | PreMig    | 227594             | 71.48                | 5133             |
|                   | Greedy    | 227629             | 71.53                | 5573             |
|                   | MobMig    | 226340             | 70.99                | 5216             |
|                   | BP        | 225732             | 70.88                | 8130             |
We consider BP, MobMig and Greedy as baseline algorithms for comparison:

- **BP** (Lai, 2018): it regards the allocation problem as a static global optimization problem and utilizes the Lexicographic Goal Programming technique to yield bin packing optimal solutions.
- **MobMig** (Peng, 2019): It predicts the requester’s movement according to the speed and direction of the requester’s current position. It utilizes the residence time of the requester in the server communication range as the fitness value and leverages a genetic strategy for identifying the near-optimal allocation plan.
- **Greedy**: it allocates unserved requesters to the nearest available MEC servers.

Figures 4, 6, 8 illustrate the comparison curve of average service rate in their subgraph (a) and migration times in their subgraph (b) of different methods in 3 scenarios with different requesters density. As can be seen from these figures, the migrations times curve is the lowest in all three scenarios and average service time curve is almost the highest. Then, the detailed experimental data are given in Table I, clearly, our proposed method outperforms its peers in all 3 scenarios in terms of average user service rate, migration counts.

**CONCLUSION**

This paper target at the edge requester allocation problem in an MEC computing environment. Instead of considering static allocation strategies, we consider dynamic service allocation with mobility and propose predictive-trajectory-aware and migration-enabled approach for online allocation. Experimental results illustrate that our framework beats its peers in terms of average requester service rate and migration counts.
REFERENCES

Abbas, N., Zhang, Y., Taherkordi, A., & Skeie, T. (2018). Mobile edge computing: A survey. *IEEE Internet of Things Journal*, 5(1), 450–465. doi:10.1109/JIOT.2017.2750180

Beck, M., Werner, M., Feld, S., & Schimper, T. (2014). Mobile edge computing: A taxonomy. *Proc. of the Sixth International Conference on Advances in Internet, 48–55.*

Chen, X., Jiao, L., Li, W., & Fu, X. (2016). Efficient Multi-User Computation Offloading for Mobile-Edge Computing. *IEEE/ACM Transactions on Networking*, 24(5), 2795–2808. doi:10.1109/TNET.2015.2487344

Chen, Y., Zhang, N., Zhang, Y., Chen, X., Wu, W., & Shen, X. S. (2019). Energy efficient dynamic offloading in mobile edge computing for internet of things. *IEEE Transactions on Cloud Computing.*

Chen, Y., Zhang, N., Zhang, Y., Chen, X., Wu, W., & Shen, X. S. (2019). Toffee: Task offloading and frequency scaling for energy efficiency of mobile devices in mobile edge computing. *IEEE Transactions on Cloud Computing.*

Chen, Y. T., & Liao, W. (2019). Mobility-Aware Service Function Chaining in 5G Wireless Networks with Mobile Edge Computing. *IEEE International Conference on Communications.* doi:10.1109/ICC.2019.8761306

Hu, J., Wang, G., Xu, X., & Lu, Y. (2019). *Study on Dynamic Service Migration Strategy with Energy Optimization in Mobile Edge Computing.* Mobile Information Systems. doi:10.1155/2019/5794870

Huang, L., Bi, S., & Zhang, Y. J. A. (2020). Deep Reinforcement Learning for Online Computation Offloading in Wireless Powered Mobile-Edge Computing Networks. *IEEE Transactions on Mobile Computing*, 19(11), 2581–2593. doi:10.1109/TMC.2019.2928811

Lai, P., He, Q., Abdelrazek, M., Chen, F., Hosking, J., Grundy, J., & Yang, Y. (2018). Optimal edge user allocation in edge computing with variable sized vector bin packing. *International Conference on Service-Oriented Computing*, 230–245. doi:10.1007/978-3-030-03596-9_15

Liu, F. R. Yu., Teng, Y., Leung, V. C. M., & Song, M. (2019). Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing. *IEEE Transactions on Wireless Communications*, 18(1), 695–708. doi:10.1109/TWC.2018.2885266

Liu, Y., He, Q., Zheng, D., Zhang, M., Chen, F., & Zhang, B. (2019). Data caching optimization in the edge computing environment. *2019 IEEE International Conference on Web Services (ICWS)*, 99–106. doi:10.1109/ICWS.2019.00027

Ma, Y., Zhang, J., Wang, S., Xia, Y., Chen, P., Wu, L., & Zheng, W. (2020). A novel approach to cost-efficient scheduling of multi-workflows in the edge computing environment with the proximity constraint Algorithms and Architectures for Parallel Processing, 655–668.

Nath, S., & Wu, J. (2020). Dynamic computation offloading and resource allocation for multi-user mobile edge computing. *GLOBECOM 2020-2020 IEEE Global Communications Conference*, 1–6.

Peng, Q., Xia, Y., Feng, Z., Lee, J., Wu, C., Luo, X., Zheng, W., Liu, H., Qin, Y., & Chen, P. (2019). Mobility-Aware and Migration-Enabled Online Edge User Allocation in Mobile Edge Computing. *2019 IEEE International Conference on Web Services (ICWS)*, 91–98. doi:10.1109/ICWS.2019.00026

Peng, Q., Xia, Y., Wang, Y., Wu, C., Zheng, W., Luo, X., Panz, S., Ma, Y., & Jiang, C. (2020). A decentralized collaborative approach to online edge user allocation in edge computing environments. In *2020 IEEE International Conference on Web Services (ICWS).* IEEE. doi:10.1109/ICWS49710.2020.00045

Wu, Peng, Xia, Ma, Zheng, Xie, Pang, Li, Fu, Li, & Liu. (2021). Online user allocation in mobile edge computing environments: A decentralized reactive approach. *Journal of Systems Architecture, 113.*

Wu, H., Deng, S., Li, W., Yin, J., Li, X., Feng, Z., & Zomaya, A. Y. (2019). Mobility-aware service selection in mobile edge computing systems. *2019 IEEE International Conference on Web Services (ICWS)*, 201–208. doi:10.1109/ICWS.2019.00042

Xh, A., Wz, A., Jy, A., Ly, B., & Chai, K. (2020). Market-based dynamic resource allocation in mobile edge computing systems with multi server and multi-user - sciencedirect. *Computer Communications.*
Xu, X., Liu, X., Xu, Z., Dai, F., Zhang, X., & Qi, L. (2020). Trust-oriented iot service placement for smart cities in edge computing. *IEEE Internet of Things Journal, 7*(5), 4084–4091. doi:10.1109/JIOT.2019.2959124

Xue & An. (2021). Joint Task Offloading and Resource Allocation for Multi-Task Multi-Server NOMA-MEC Networks. *IEEE Access, 9*(16), 152–163.

Yang, B., Cao, X., Bassey, J., Li, X., & Qian, L. (2020). Computation offloading in multi-access edge computing: A multi-task learning approach. *IEEE Transactions on Mobile Computing.*

Yang, L., Liu, B., Cao, J., Sahni, Y., & Wang, Z. (2017). Joint computation partitioning and resource allocation for latency sensitive applications in mobile edge clouds. *2017 IEEE 10th International Conference on Cloud Computing (CLOUD),* 246–253. doi:10.1109/CLOUD.2017.39

Zhang, M., Huang, H., Rui, L., Hui, G., Wang, Y., & Qiu, X. (2020). A service migration method based on dynamic awareness in mobile edge computing. doi:10.1109/NOMS47738.2020.9110389

Zhang, Q., Gui, L., Hou, F., Chen, J., Zhu, S., & Tian, F. (2020). Dynamic Task Offloading and Resource Allocation for Mobile-Edge Computing in Dense Cloud RAN. *IEEE Internet of Things Journal, 7*(4), 3282–3299. doi:10.1109/JIOT.2020.2967502