IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 10, 2019, accepted July 27, 2019, date of current version December 3, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2937337

Mobile Edge Computing-Enhanced Proximity Detection in Time-Aware Road Networks

YAQIONG LIU[1], (Member, IEEE), MUGEN PENG[2], (Senior Member, IEEE), AND GUOCHU SHOU[1]

[1]School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China
[2]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China

Corresponding author: Yaqiong Liu (liuyaqiong@bupt.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant No. 61901052, Fundamental Research Funds for the Central Universities under Grant No. 2018RC03, and 111 Project of China under Grant No. B17007. A part of this paper was presented in PIMRC 2019.

ABSTRACT Given a set of moving objects as well as their friend relationships, a time-aware road network, and a time threshold per friend pair, the proximity detection problem in time-aware road networks is to find each pair of moving objects such that the time distance (defined as the shortest time needed for two moving objects to meet each other) between them is within the given threshold. The problem of proximity detection is often encountered in autonomous driving and traffic safety related applications, which require low-latency, real time proximity detection with relatively low communication cost. However, (i) most existing proximity detection solutions focus on the Euclidean space which cannot be used in road network space, (ii) the solutions for road networks focus on static road networks and do not consider time distance and thus cannot be applied in time-aware road networks, and (iii) there are no works aiming to simultaneously reduce the communication cost, the communication latency, and computational cost. Motivated by these, we first design a low-latency proximity detection architecture based on Mobile Edge Computing (MEC) with the purpose of achieving low communication latency, then propose a proximity detection method including a client-side algorithm and a server-side algorithm, aiming at reducing the communication cost, and subsequently propose server-side computational cost optimization techniques to reduce the computational cost. Experimental results show that our MEC enhanced proximity detection architecture, our proximity detection method, and the server-side computational cost optimization techniques can reduce the communication latency, the communication cost, and the computational cost effectively.

INDEX TERMS Cost optimization, low latency, mobile edge computing, proximity detection, time-aware road networks, time distance.

I. INTRODUCTION

In a road network, how to effectively detect whether the moving users are within proximity or not, is referred to as the problem of proximity detection in road networks. In a dynamically changing road network, proximity detection among a large number of moving users plays an important role in ensuring traffic safety, guaranteeing assisted driving, and realizing the future large-scale autonomous driving.

With the spreading of modern mobile devices like smart phones, PDAs or car navigation systems and the development of positioning technologies such as GPS, WiFi, cellular base station positioning or A-GPS, users can conveniently obtain their positions and send their location information to control center servers or other users' mobile devices. Mobile users communicate with servers or other users frequently, leading to a large number of communication messages referred to as communication cost, consuming a lot of network bandwidth.

Given a large number of mobile users and time distance T_ϵ, time-aware proximity detection problem in road networks is to find a solution which not only can continuously detect which pairs of users among all users are within proximity based on **time distance**, but also can achieve the objectives of reducing communication cost, communication latency, and computational cost, so as to save the network bandwidth and improve the reliability and efficiency of proximity detection. Here, time distance refers to the shortest time needed for two users to meet each other.

However, most existing proximity detection solutions adopt traditional client-server (C/S) architecture or

The associate editor coordinating the review of this manuscript and approving it for publication was Debashis De.

absolutely distributed peer-to-peer (P2P) architecture, based on geographical distance (Euclidean distance or road network distance), facing the bottlenecks of long communication latency, long computational time, and large communication cost. Moreover, most existing solutions neglect a fact that in most cases time distance is more important and meaningful than geographical distance. For example, an autonomous driving vehicle must avoid those oncoming vehicles which may collide with it within a short time rather than those vehicles running in parallel with it which have very close geographical distance to it. In addition, adopting absolutely distributed P2P architecture may incur too many communication messages since every two users communicate with each other in P2P architecture, and moreover, P2P architecture cannot provide a global view on the proximity states of all moving users in the network. Therefore, P2P architecture cannot be applicable to this problem. In contrast, traditional C/S architecture can provide a global view on the proximity states of all moving users, but it faces the problems of long communication latency and large computational time.

In order to reduce the communication latency, Mobile Edge Computing (MEC) [1], [2] was firstly proposed by ETSI (European Telecommunications Standards Institute) in 2014. MEC provides an IT service environment and cloud-computing capabilities at the edge of the mobile network, within the Radio Access Network (RAN) and in close proximity to mobile subscribers. In the MEC architecture, servers are also deployed at the edge. MEC platforms can reduce network latency by enhancing the computation and storage capabilities of the edge network. As a key technology towards 5G, MEC creates an effective way to realize 5G [3], [4].

Regarding the abovementioned background, this paper conducts researches on proximity detection problem in road networks based on time distance. The contributions of this paper are summarized as follows.

- We define a proximity detection problem in time-aware road networks and use the metric of time distance to judge whether two users are within proximity or not.
- We adopt the Mobile Edge Computing (MEC) paradigm to design the proximity detection architecture, which can reduce the communication latency between the clients and the server greatly.
- We propose a time-aware mobile region based proximity detection method, namely, TMRBD, including a client-side algorithm and a server-side algorithm, to solve the time-aware proximity detection problem, which can reduce the communication cost effectively.
- We propose server-side computational cost optimization techniques, which can reduce the computational time at the server side to a large extent.

The remainder of this paper is organized as follows. Section II presents related work on spatial query processing and proximity detection in Euclidean space or road network space. Section III proposes the problem setting and useful definitions to model the time-aware road network. Section IV presents the MEC enhanced architecture for proximity detection. Section V proposes a mobile region based proximity detection method to solve proximity detection problem for mobile users. Section VI introduces the server-side computational cost optimization techniques. Section VII studies the effectiveness and efficiency of our proposed solutions. Finally, Section VIII concludes this paper.

II. RELATED WORK

In this section, we first present related work on spatial query processing as proximity detection is a kind of spatial queries, subsequently present existing works on proximity detection in Euclidean space, and finally review related work on proximity detection in road networks.

A. SPATIAL QUERY PROCESSING

As an important spatial query, proximity detection plays a key role in the traffic safety related applications. Recently, researchers have proposed various approaches for spatial query processing on moving objects. Generally, the approaches for spatial query processing on moving objects can be classified into two classes, snapshot-based methods and continuous query monitoring based methods.

Snapshot based methods such as [5]–[13] aim at efficiently processing queries that are issued at certain points of time. Some of these methods [5], [10], [11], [14] exploit index structures that can cope with moving objects. Reference [6] studies multiple k nearest neighbor queries. Reference [7] processes nearest neighbor queries in road networks. Reference [8] studies voronoi based k nearest neighbor queries. Reference [9] proposes an architecture integrating network and Euclidean information, which can be applied to the most popular spatial queries, namely nearest neighbors, range search, closest pairs and e-distance joins. Reference [12] proposes an index structure called the TPR*-tree for predictive queries. Reference [13] focuses on two common spatial queries, namely, nearest neighbor queries and window queries.

Some of continuous query monitoring based methods such as [15]–[26] focus on Euclidean space. Among these works, [15] adopts the distributed P2P architecture and proposes a ''strips'' algorithm; [16] presents several location updating methods and compares their efficiency; [19] proposes a generic framework for monitoring continuous spatial queries; [17], [18], [20], [21], [23], [25], [26] resolve range-monitoring queries, continuously moving queries, k nearest neighbor queries, and nearest neighbor queries, respectively; another two works [22], [24] also deal with spatial queries in Euclidean space.

Many existing works for spatial query processing do not address our requirements for proximity detection. For instance, [27], [28] focus on the k nearest neighbor (kNN) query. Reference [29] focuses on reverse nearest neighbor (RNN) queries. References [30], [31] deal with continuous nearest neighbor (CNN) queries. Reference [32] deals with e-Distance joins query. References [33]–[35] resolve

spatial keyword queries. Reference [36] handles optimal location queries. Reference [37] studies the top k most influential facilities over a set of uncertain objects. All these above-mentioned works do not provide algorithms to detect and to monitor proximity relations.

B. PROXIMITY DETECTION IN EUCLIDEAN SPACE

The proximity detection problem has been studied in [15], [38]–[44]. Most existing proximity detection solutions focus on Euclidean space where the distance between two objects is determined solely by their relative positions in Euclidean space. References [38], [39] develop client-server solutions that focus on reducing the communication cost of proximity detection. The dynamic centered circle method proposed by [38] assigns each user a circle such that the minimum distance between any two circles is above the distance threshold. However, the circle is static, causing that the user moves outside it soon, triggering a location update to the server. Reference [39] employs moving sector regions for tracking users' locations and detecting the proximity among them at the server. The moving sector region of a user is described by three parameters with predefined values: an angular threshold R, the minimum speed V_{min} and the maximum speed V_{max}. Other solutions include absolutely distributed P2P solutions like the strips algorithm [15] and centralized solutions like [41]. [15] requires each user to maintain a strip for each of his friends, and thus its performance does not scale well to a large number of friends. Reference [41] applies self-tuning policies in Euclidean space and the server detects whether the Euclidean distance of each friend pair is within the proximity threshold. Another work [40] generalizes the proximity detection problem to the constrained detection problem. A constraint is satisfied when a specified set of k objects can be enclosed by a circle with a diameter of at most ϵ. They propose a centralized solution, which tracks objects in a space-partitioning grid. An object (i.e., client) does not issue any location updates to the server until it enters another cell of the grid. Based on the locations of the objects in the grid, their solution identifies the objects that definitely satisfy the constraints and the objects that definitely dissatisfy the constraints. Reference [42] processes proximity queries as a batch and focus on probing rather than location update to enable communication-efficient proximity detection. Reference [43] studies the proximity detection problem with a safe region that is constructed by trajectory prediction techniques. Reference [44] proposes a method named ''TROY'', which considers the influence of the distance between two users on the energy cost of the mobile terminals, aiming to reduce the energy cost of the mobile terminals. However, ''TROY'' does not take the number of communication messages into consideration and in some cases it even incurs more communication messages than other methods.

C. PROXIMITY DETECTION IN ROAD NETWORKS

There are some works focusing on road network space. A network graph embedding technique to speed up distance-range and kNN queries is proposed in [45]. However, the key issue in this paper is distance-range and kNN queries instead of the proximity detection problem. It has not given any solutions for proximity detection. A more related work (CPMRN [46]) proposes a region-based (zone-based) update strategy for continuous proximity monitoring in road networks. The key idea is to store both a proximity region and a separation region for each client and as long as the client does not reach the boundary of one of its regions, it does not need to update the proximity/separation join results. Another work [47] defines three types of proximity relations that induce location constraints to model continuous spatial-temporal queries among sets of moving objects in road networks, but they do not give a solution to the proximity detection problem in road networks. Another representative work to tackle proximity detection in road networks is [48], which not only proposes a fixed-radius mobile detection method, but also adopts a self-tuning policy to automatically adjust the radius of the mobile region.

The works above focus on static road networks. However, in most cases, the road networks are time-aware rather than static. Few works resolve proximity detection problem in time-aware road networks. The work [49] studies proximity queries in time-dependent road networks using graph embedding technique. However, the purposes of this work do not include reducing the communication cost or the communication latency. Therefore, there is still a critical demand of developing efficient proximity detection solutions in time-aware road networks to minimize the communication cost as well as the communication latency and computational cost.

In this paper, we extend our previous work [48] and address the proximity detection problem in time-aware road networks. In time-aware road networks, the speeds of each object is dependent upon time, and we adopt the metric of time distance to measure the proximity of each pair of friends. If the time distance between a pair of friends is smaller than or equal to the proximity threshold (i.e., time threshold), we say that this friend pair is within proximity. Aiming at reducing the communication latency, we propose a proximity detection architecture enhanced by MEC. Aiming to reduce the communication cost, we propose four pruning lemmas, and propose client-side and server-side algorithms. We further propose computational cost optimization techniques for the purpose of minimizing the computational cost.

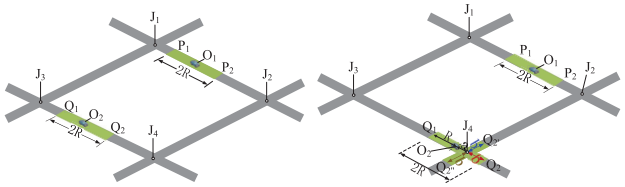
III. PROBLEM STATEMENT

In this section, we define time-aware road network, and afterwards give the definition of the proximity detection problem in time-aware road networks.

A. DEFINITIONS AND NOTATIONS

We extend the definitions of [48], and present the definitions as well as some notations to model the time-aware road network.

Definition 1 Node: A node is a junction of road segments, or say, an intersection of two or more different road segments. ♡



(a) mobile regions at time T_1 (b) mobile regions at time $T_2 = T_1 + \Delta T$

FIGURE 1. Fixed-radius mobile regions in a road network.

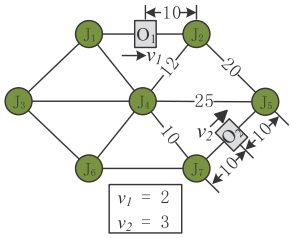


FIGURE 2. An example of a time-aware road network.

As shown in Fig. 1(a), J_1, J_2, J_3 and J_4 are typical junctions.

Definition 2 Edge: An edge is a line segment connecting two adjacent nodes. An edge e is represented as a quadruple (eid, nid_{from}, nid_{to}, len), where, eid is the identifier of e, nid_{from} and nid_{to} refer to the identifiers of the starting and ending nodes of e, and len is the length of e. ♡

As shown in Fig. 1(a), (J_1, J_2), (J_1, J_3), (J_3, J_4), and (J_2, J_4) are all edges.

Definition 3 Network Point: A network point is a two-dimensional point located on the edges of a road network.♡

For example, P_1, Q_1, P_2, Q_2, J_1, J_2, J_3 and J_4 in Fig. 1(a) are all network points.

Definition 4 Network Distance: Given two network points P and P' in the road network, their network distance is given by

$$D(P, P') = \min_{\substack{i\in\{s,t\},\\ i'\in\{s',t'\}}} (D(P, J_i) + D(J_i, J_{i'}) + D(J_{i'}, P')) \quad (1)$$

♡

where J_s, J_t, and $J_{s'}$, $J_{t'}$ are the two end nodes of the edges on which P and P' lie, respectively. The network distance between two network points in a road network is the length of the shortest path between the two points. For example, in Fig. 1(a), the Euclidean distance between P_1 and Q_1 is denoted as $|P_1Q_1|$, while the network distance between them is:

$$D(P_1, Q_1) = \min_{\substack{i\in\{1,2\},\\ i'\in\{3,4\}}} (D(P_1, J_i) + D(J_i, J_{i'}) + D(J_{i'}, Q_1)).$$

Definition 5 Line Segment: A line segment is a segment between two network points on the same edge. The first network point is the starting point and the second network point is the ending point. ♡

As shown in Fig. 1(a), $\overline{P_1P_2}$ and $\overline{Q_1Q_2}$ are line segments.

Definition 6 Time-Aware Road Network: A time-aware road network $G_T = (N, E)$ is a directed graph, which is comprised of two finite sets N and E, representing the set of junctions (nodes) and set of edges, respectively. Every edge (J_i, J_j) has two functions associated with it: $v_{max}((J_i, J_j), t, O)$, and $len(J_i, J_j)$, which denote the maximum speed allowed on edge (J_i, J_j) for object O when departing J_i at time t, and the length of edge (J_i, J_j), respectively. Note that the average speed to travel along each segment for each kind of object is time dependent while the length of each edge is independent of time. ♡

For example, Fig. 2 depicts a simple road network.

Definition 7 Time Distance: The time distance between two moving objects at time t is defined as the least travel time needed for the two objects to meet each other if departing from their current positions at time t. ♡

Taking Fig. 2 as an example, at a certain time stamp, objects O_1 and O_2 are moving towards J_2 and J_5, respectively. The network distance between O_1 and O_2 is computed as follows: $D(O_1, O_2) = 10 + 20 + 10 = 40$. In time-aware road networks, the maximum speeds of the moving objects are dependent upon time. Suppose O_1 and O_2 are moving with a constant velocity of v_1 and v_2, respectively, then the time distance $T_{avg}(O_1, O_2) = \frac{D(O_1,O_2)}{v_1+v_2} = \frac{40}{2+3} = 8$.

Definition 8 Mobile Region: A mobile region of an object is one line segment on one edge, or several line segments on several edges. More formally, a mobile region can be represented by a tree composed of a sequence of line segments. If the radius of a mobile region is R, the network distance from the root of this tree to each of the leaves is equal to $2\times R$. We denote a mobile region of an object O_m at time t by $R_m(t)$. ♡

Suppose $(A_1, A_2, \ldots, A_p)$ are vertices of $R_m(t)$, and $(B_1, B_2, \ldots, B_q)$ are vertices of $R_n(t)$, then $R_m(t)$ and $R_n(t)$ have $p-1$ edges and $q-1$ edges, respectively.

$$R_m(t) = \bigcup_{1\le i\le p-1} e_i^{(m)}, \quad R_n(t) = \bigcup_{1\le j\le q-1} e_j^{(n)}. \quad (2)$$

where, $e_i^{(m)}$ and $e_j^{(n)}$ represent an arbitrary edge of $R_m(t)$ and $R_n(t)$, respectively. $e_i^{(m)} = \overline{e_i^{(m)}.from, e_i^{(m)}.to}$, $e_j^{(n)} = \overline{e_j^{(n)}.from, e_j^{(n)}.to}$, where $e_i^{(m)}.from$ and $e_i^{(m)}.to$ represent the end points of edge $e_i^{(m)}$; $e_j^{(n)}.from$ and $e_j^{(n)}.to$ represent the end points of edge $e_j^{(n)}$.

As shown in Fig. 1(b), the mobile region $R_1(t)$ of O_1 at time t equals $\{\overline{P_1P_2}\}$ and the mobile region $R_2(t)$ of O_2 at time t equals $\{\overline{Q_1J_4}, \overline{J_4Q_2}, \overline{J_4Q_{2'}}, \overline{J_4Q_{2''}}\}$.

Notations: We give the notations used in this paper in Table 1.

B. PROBLEM SETTING

Definition 9 Proximity Detection in Time-Aware Road Networks: Given a time-aware road network G_T, a set of moving objects as well as the friendship between them, the average speed functions for each edge for each category of objects

TABLE 1. Notations.

Notation	Meaning
N	the total number of users in a road network
T_ϵ	time proximity threshold
m	the average number of friends per user
R	the radius of the mobile region
(J_i, J_j)	the edge connecting junctions n_s and n_t
$\overline{PQ}$	line segment $\overline{PQ}$
$\|PQ\|$	Euclidean distance between P and Q
$D(P,Q)$	network distance between P and Q
$R_m(t)$	mobile region of Object O_m at time t
$v_{max}(O)$	the maximum speed of object O
$v_{max}((J_i, J_j), t, O)$	the maximum speed allowed on edge (J_i, J_j) for object O if leaving J_i at time t
$d_{min}(R_m(t), R_n(t))$	the lower bound of the Euclidean distance between $R_m(t)$ and $R_n(t)$
$D_{min}(R_m(t), R_n(t))$	the lower bound of the network distance $D(R_m(t), R_n(t))$ at time t
$D_{max}(R_m(t), R_n(t))$	the upper bound of the network distance $D(R_m(t), R_n(t))$ at time t
$t_{min}(R_m(t), R_n(t))$	lower bound of the time distance between $R_m(t)$ and $R_n(t)$ based on $d_{min}(R_m(t), R_n(t))$ at time t
$T_{min}(R_m(t), R_n(t))$	lower bound of the time distance between $R_m(t)$ and $R_n(t)$ based on $D_{min}(R_m(t), R_n(t))$ at time t
$T_{max}(R_m(t), R_n(t))$	upper bound of the time distance between $R_m(t)$ and $R_n(t)$ at time t
$e_i^{(m)}$	the i-th edge of object O_m's mobile region $R_m(t)$
$e_j^{(n)}$	the j-th edge of object O_n's mobile region $R_n(t)$

with respect to time, and a time proximity threshold T_ϵ, the proposed proximity detection problem in time-aware road networks G_T is to design efficient solutions which incur low communication cost, low communication latency, and low computational cost, to find whether the time distance between each pair of friends is no more than T_ϵ. ♡

Note that the metric used to measure proximity is the time distance between two moving objects instead of the network distance between them.

We assume that all moving objects are equipped with positioning devices. All servers are equipped with a map of the road network and are aware of the length of each edge and the coordinates of each junction of the road network. The server aims to check the proximity relationship of each friend pair every epoch (e.g., every second).

IV. MEC ENHANCED PROXIMITY DETECTION ARCHITECTURE

A. ARCHITECTURE

As mentioned in Section I, one objective is to reduce the communication latency between the clients and the server as much as possible since time delay should be as short as possible for the reliability and efficiency of proximity detection in road networks. To this end, we design the proximity detection architecture based on MEC, as shown in Fig. 3, since MEC can offer a service environment with low latency, high-bandwidth, and direct access to real-time network information.

In our MEC enhanced proximity detection architecture, the core network is located in the center of communication networks, and multiple MEC servers are deployed at multiple edge clouds. Each moving client communicates with the nearest MEC server, instead of communicating with the central server via the core network. The communication mechanism is as follows: the client can send update messages to the server

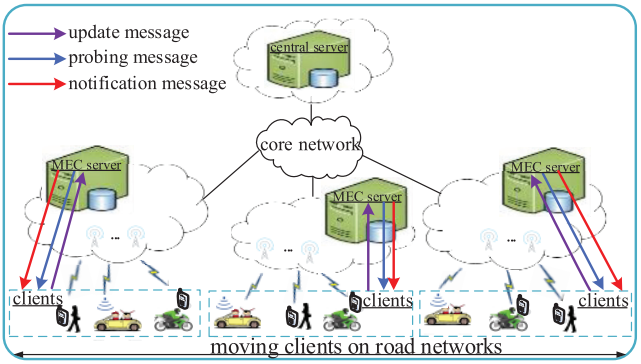


FIGURE 3. Proximity detection architecture based on MEC.

to report its location and other motion parameters such as its speed; the server can probe the client regarding its motion state, and the server can also send notification messages to a client to tell it with whom it is within proximity. Therefore, the communication cost includes the number of update messages, the number of probing messages and the number of notification messages. Thus, the normal communication mechanism between the client and the server is preserved and the advantages of MEC can be adopted. Meanwhile, in the traditional client-server architecture, the central server is in charge of all the computation of the proximity relationships among all mobile clients, which causes high computational burden and high computational complexity for the central server. In contrast, in our new proximity detection architecture, each MEC server only needs to take charge of the proximity detection for the clients which communicate with it so that it has low computational burden.

It is noteworthy that, most clients can find their friends which are in proximity with them inside the coverage of one MEC server. However, for some clients which are located at the border of the coverage of one MEC server, their friends which are in proximity with them may not be inside the

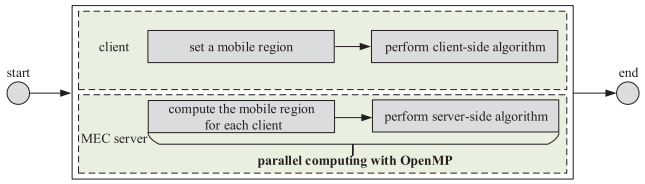


FIGURE 4. Flow diagram.

coverage of the same MEC server, but may be inside the coverage of another MEC server and thereby communicate with another MEC server. Thus, to ensure the accuracy of proximity detection, each MEC server reports those clients which are located at the border of its coverage to the central server, and let the central server take charge of the proximity detection of these clients which are at the border of the coverage of each MEC server.

In other words, proximity detection for most clients are carried out at MEC servers, and only those clients which are located at the border of the coverage of each MEC server require the central server to check their proximity. The detection results computed by the central server will be updated to the corresponding MEC server. Note that in our proposed MEC based proximity detection architecture, both the majority of clients whose proximity relationship with other clients are calculated by the MEC server and the minority of clients whose proximity relationship with other clients are calculated by the central server communicate with their nearest MEC server and do not communicate with the central server, which guarantees the low-latency characteristics of our proposed architecture.

B. FLOW DIAGRAM

The flow diagram of our proximity detection system is shown in Fig. 4. For the purpose of realizing proximity detection with low communication cost and low computational cost, the client and the servers need to do their respective work continuously.

At the client side, each client has a mobile region, inside which as long as the client is, it has no need to send update messages to the server; The client performs the client-side algorithm which will be detailed in Algorithm 3 in Section V-C.

At the MEC server side, the work of the server includes computing the mobile region dynamically for each moving client and performing the server-side algorithm which will be detailed in Algorithm 4 in Section V-C. The abovementioned tasks are done by adopting multi-threading parallel computing with OpenMP,[1] the details of which will be given in Section VI-B.

V. ALGORITHMS: TIME-AWARE MOBILE REGION BASED DETECTION METHOD

Based on the proposed architecture, we present a proximity detection method, namely, Time-aware Mobile Region Based Detection (TMRBD), with a mobile region for each client. We first present the mobile regions used in our method, and then propose four pruning lemmas which make use of the lower bound and upper bound of the time distance between mobile regions of two clients. We finally give the algorithms at the client side and server side, respectively.

A. MOBILE REGION IN TIME-AWARE ROAD NETWORKS

We define a mobile region for each client in such a way that unless a client moves outside its mobile region, it does not need to send an update message to the server initiatively. Let R denote the fixed radius of the mobile region of each client, T_{last} denote the time stamp of last update, T_{cur} denote the current time stamp where $T_{cur} = T_{last} + \Delta T$, $P_{T_{last}}$ denote the position of the client at time T_{last}, $P_{T_{cur}}$ denote the current position of the client at current time T_{cur}, and let v_{avg} denote the average velocity during time period (T_{last}, T_{cur}). Then the exact network distance between $P_{(T_{last}+\Delta T)}$ and $P_{T_{last}}$ is calculated as follows:

$$D(P_{(T_{last}+\Delta T)}, P_{T_{last}}) = v_{avg} \cdot \Delta T \tag{3}$$

1) ALGORITHMS FOR COMPUTING MOBILE REGIONS

The algorithms for computing the mobile regions are given in Algorithm 1 and Algorithm 2. The key idea is starting from the current position *pos* which is the center of the final mobile region *MR*, recursively calculating the forward and backward line segments one by one, until the total length *len* from the center *pos* to the forward ending point and the backward ending point is equal to R.

Algorithm 1 The Computation Algorithm of Mobile Regions: **ComputeMobileRegion**

Input: the radius R of mobile regions, the current position *pos* of a moving object, the edge e on which the moving object lies
Output: the mobile region *MR*

1 $len = 0$;
2 line segment ls_{prev}, ls_{next};
3 $ls_{prev}.p_1 = pos$;
4 $ls_{prev}.p_2 = e.p_1$;
5 **ComputeNewSegment**(len, ls_{prev}, $ls_{prev}.p_1$);
6 $ls_{next}.p_1 = pos$;
7 $ls_{next}.p_2 = e.p_2$;
8 **ComputeNewSegment**(len, ls_{next}, $ls_{next}.p_1$);

Algorithm 1 describes the key function, i.e., **ComputeMobileRegion** (R, *pos*, e), for computing the mobile region *MR* for an object. In Algorithm 1, the variable *len* represents the current length from the center *pos* to the two ending points of the current mobile region. At the beginning, *len* is set to be 0 (Line 1), indicating that the current mobile region is still empty and there are no segments added to the mobile region; Then, two line segments ls_{prev}, ls_{next} are created (Line 2), representing the backward and forward

[1] https://www.openmp.org/

Algorithm 2 Void ComputeNewSegment

Input: len, ls, p
Output: MR

```
1  if len + ls.length ≥ R then
2      nextlen = R − len ;
3      newP.x = ls.p1.x + (nextlen / ls.length) ∗ (ls.p2.x − ls.p1.x);
4      newP.y = ls.p1.y + (nextlen / ls.length) ∗ (ls.p2.y − ls.p1.y);
5      len = R ;
6      line segment ls0;
7      ls0.p1 = ls.p1;
8      ls0.p2 = newP;
9      MR.push_back(ls0);
10 end
11 else
12     MR.push_back(ls);
13     newP = ls.p2;
14     len+ = ls.length;
15     for each outgoing edge ei from ls.p2 && ei! = ls do
16         ComputeNewSegment(len, ei, newP);
17     end
18 end
```

line segments, with pos being its starting point $ls_{prev}.p_1$ or $ls_{next}.p_1$ (Line 3, Line 6) and $e.p_1$ or $e.p_2$ being its ending point (Line 4 or Line 7), where e represents the edge which this object lies on and $e.p_1$ or $e.p_2$ represents the two ending points of edge e; Subsequently, this function calls another function **ComputeNewSegment**(len, ls_{prev}, $ls_{prev}.p_1$) or **ComputeNewSegment**(len, ls_{next}, $ls_{next}.p_1$) (Line 5 or Line 8) which is detailed in Algorithm 2.

Algorithm 2 describes the function **ComputeNewSegment** (len, ls, p) which recursively computes the new line segment that should be added into the mobile region MR one by one, taking the current length len, the candidate line segment ls, and the starting point p of ls as input. If $len+ls.length \geq R$ (Line 1), which demonstrates that the line segment ls should not be fully added into the final mobile region MR, then we compute the new line segment ls_0 which is a part of ls (Lines 2-8), and add it into the final mobile region MR (Line 9); Otherwise (Line 11), the entire line segment ls should be fully added into the mobile region MR (Line 12), meanwhile we update the value of len which is the current total length from the center pos to the ending point $newP$ (i.e., $ls.p_2$) (Lines 13-14), and then, for each outgoing edge e_i from $ls.p_2$ except for ls itself (Line 15), we recursively call the function **ComputeNewSegment**(len, e_i, $newP$) (Line 16) to compute the next line segment that should be added into the final mobile region MR.

2) A RUNNING EXAMPLE OF MOBILE REGIONS

We give an example of mobile regions in a time-aware road network. As shown in Fig. 1, O_1 and O_2 are two clients moving along edges (J_1, J_2) and (J_3, J_4). In Fig. 1(a), suppose two clients O_1 and O_2 have just reported their positions to the server at time T_1, and the line segments $\overline{P_1P_2}$ and $\overline{Q_1Q_2}$ with length $2R$ highlighted in green are the mobile regions of O_1 and O_2, with O_1 and O_2 being the midpoints of $\overline{P_1P_2}$ and $\overline{Q_1Q_2}$, respectively. In Fig. 1(b), at time T_2, O_1 and O_2 send position update messages to the server again, then line segment $\overline{P_1P_2}$ with length $2R$ is the new mobile region of O_1 with the midpoint O_1. As O_2 approaches J_4, its mobile region is no longer completely within the edge (J_3, J_4). Suppose O_2 is a-unit distance away from J_4, where $a < R$, then the leftmost point Q_1 of its mobile region is $(R + a)$ units away from J_4. Here, O_2 has three different directions to move along when it arrives at junction J_4, and therefore edges $\overline{J_4Q_2}$, $\overline{J_4Q_{2'}}$, $\overline{J_4Q_{2''}}$ are included into its mobile region, where $|\overline{J_4Q_2}| = |\overline{J_4Q_{2'}}| = |\overline{J_4Q_{2''}}| = b = R - a$.

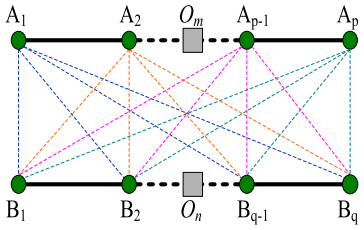


FIGURE 5. Two moving objects and their mobile regions.

B. PRUNING LEMMAS

In time-aware road networks, the server checks whether the time distance between each pair of friends which it takes charge of is within time proximity threshold or not. In fact, the server does not need to probe every client regarding his location and speed continuously. Sometimes it is unnecessary to probe some friend pairs because we can safely prune such friend pairs. In order to facilitate the server to avoid checking some friend pairs unnecessarily, we propose four pruning lemmas as follows.

1) UNQUALIFIED FRIEND PAIRS PRUNING BASED ON LOWER BOUND OF TIME DISTANCE

Theorem 1: Given two mobile regions $R_m(t)$ and $R_n(t)$, as depicted in Fig. 5, then the following inequality holds.

$$d_{min}(R_m(t), R_n(t)) \leq D(R_m(t), R_n(t)) \tag{4}$$

where, $d_{min}(R_m(t), R_n(t))$ gives a lower bound of $D(R_m(t), R_n(t))$ from the point of view of *Euclidean* distance, and can be computed by the following Equation:

$$d_{min}(R_m(t), R_n(t)) = \min_{1\leq i\leq p, 1\leq j\leq q} |A_i, B_j| \tag{5}$$

where, $(A_1, A_2, \ldots, A_p)$ are vertices of $R_m(t)$, and $(B_1, B_2, \ldots, B_q)$ are vertices of $R_n(t)$.

Theorem 2: Given two mobile regions $R_m(t)$ and $R_n(t)$, as depicted in Fig. 5, then the following inequality holds.

$$D_{min}(R_m(t), R_n(t)) \leq D(R_m(t), R_n(t)) \tag{6}$$

where, $D_{min}(R_m(t), R_n(t))$ gives a lower bound of $D(R_m(t), R_n(t))$ from the point of view of *network* distance,

and can be computed by the following Equation:

$$D_{min}(R_m(t), R_n(t)) = \min_{1\le i\le p, 1\le j\le q} D(A_i, B_j) \quad (7)$$

where, $(A_1, A_2, \ldots, A_p)$ are vertices of $R_m(t)$, and $(B_1, B_2, \ldots, B_q)$ are vertices of $R_n(t)$. ♠

Following our previous work on proximity detection in static road networks [48], Theorem 1 and 2 hold obviously (the proof can be found in [48]). Theorem 1 and 2 give two lower bounds of the distance between two mobile regions.

However, note that in this paper, the metric to measure proximity is the time distance. Hence we must compute the lower bounds of the time distance between two mobile regions.

Theorem 3: Given two moving objects O_m and O_n, and their mobile regions $R_m(t)$ and $R_n(t)$, as depicted in Fig. 5, the lower bound of the time distance between them based on $d_{min}(R_m(t), R_n(t))$ can be given as the following formula.

$$t_{min}(R_m(t), R_n(t)) = \frac{d_{min}(R_m(t), R_n(t))}{v_{max}(O_m) + v_{max}(O_n)} \quad (8)$$

where, $d_{min}(R_m(t), R_n(t))$ can be obtained by Eq. 5; $v_{max}(O_m)$ and $v_{max}(O_n)$ represent the maximum speed of O_m and O_n, respectively. ♠

Theorem 4: Given two moving objects O_m and O_n, and their mobile regions $R_m(t)$ and $R_n(t)$, as depicted in Fig. 5, the lower bound of the time distance between them based on $D_{min}(R_m(t), R_n(t))$ can be given as the following formula.

$$T_{min}(R_m(t), R_n(t)) = \frac{D_{min}(R_m(t), R_n(t))}{v_{max}(O_m) + v_{max}(O_n)} \quad (9)$$

where, $D_{min}(R_m(t), R_n(t))$ can be obtained by Eq. 7; $v_{max}(O_m)$ and $v_{max}(O_n)$ represent the maximum speed of O_m and O_n, respectively. ♠

Lemma 1 Unqualified Pair Pruning I: For objects O_m and O_n, if $t_{min}(R_m(t), R_n(t))$ is larger than the time threshold T_ϵ, then the time distance between this friend pair must be larger than T_ϵ, thus this friend pair should be pruned. ♣

Proof: The proof of Lemma V-B.1 is straightforward.

Lemma 2 Unqualified Pair Pruning II: For objects O_m and O_n, if $T_{min}(R_m(t), R_n(t))$ is larger than the time threshold T_ϵ, then the time distance between this friend pair must be larger than T_ϵ, thus this friend pair should be pruned. ♣

Proof: The proof of Lemma V-B.1 is straightforward.

According to Lemma V-B.1 and V-B.1, the server can prune those friend pairs whose time distance is surely larger than the proximity threshold. Hence the server does not need to probe these friend pairs and thus lots of probing messages can be saved.

2) QUALIFIED FRIEND PAIRS PRUNING BASED ON UPPER BOUND OF TIME DISTANCE

Theorem 5: Given two moving objects O_m and O_n, as well as their mobile regions $R_m(t)$ and $R_n(t)$, as depicted in Fig. 5, then the following inequality holds.

$$D(R_m(t), R_n(t)) \le D_{max}(R_m(t), R_n(t)) \quad (10)$$

where

$$\begin{aligned} D_{max}(R_m(t), R_n(t)) &= \max_{\substack{1\le i\le p-1\\ 1\le j\le q-1}} \{\min\{D(e_i^{(m)}.from, e_j^{(n)}.from), \\ &\quad D(e_i^{(m)}.from, e_j^{(n)}.to), \\ &\quad D(e_i^{(m)}.to, e_j^{(n)}.from), \\ &\quad D(e_i^{(m)}.to, e_j^{(n)}.to)\} + |e_i^{(m)}| + |e_j^{(n)}|\}. \end{aligned} \quad (11)$$

where, $(A_1, A_2, \ldots, A_p)$ are vertices of $R_m(t)$, and $(B_1, B_2, \ldots, B_q)$ are vertices of $R_n(t)$; $|e_i^{(m)}|$ and $|e_j^{(n)}|$ represent the length of edge $e_i^{(m)}$ and $e_j^{(n)}$, respectively; $e_i^{(m)} = \overline{e_i^{(m)}.from, e_i^{(m)}.to}$, and $e_j^{(n)} = \overline{e_j^{(n)}.from, e_j^{(n)}.to}$. ♠

Following our previous work on proximity detection in static road networks [48], Theorem 5 holds obviously (its proof can be found in [48]). Theorem 5 gives the upper bound of the network distance between two mobile regions. Next we compute the upper bound of the time distance between two mobile regions.

Theorem 6: Given two moving objects O_m and O_n, and their mobile regions $R_m(t)$ and $R_n(t)$, as depicted in Fig. 5, the upper bound of the time distance between them can be given as the following formula.

$$T_{max}(R_m(t), R_n(t)) = \frac{D_{max}(R_m(t), R_n(t))}{v_{min}(O_m) + v_{min}(O_n)} \quad (12)$$

where, $v_{min}(O_m)$ and $v_{min}(O_n)$ represent the minimum speed of O_m and O_n, respectively. ♠

Lemma 3 Qualified Pair Pruning: Given two moving objects O_m and O_n, as well as their mobile regions $R_m(t)$ and $R_n(t)$, if $T_{max}(MR_m(t), MR_n(t))$ is no larger than T_ϵ, then this friend pair must be within proximity and should be selected into the proximity result set. ♣

According to Lemma V-B.2, the server can avoid probing those friend pairs whose time distance is surely no larger than the proximity threshold and therefore many probing messages can be saved.

3) TIME STAMPS PRUNING BASED ON LOWER BOUND OF TIME DISTANCE

Lemma 4 Unqualified Time Stamps Pruning: Given two moving objects O_m and O_n, as well as their mobile regions $R_m(t)$ and $R_n(t)$, if the lower bound of the time distance between their mobile regions is still larger than $T_\epsilon + \Delta T * a$, then from the current time epoch until the next a time epoch, the two moving clients are surely not within proximity, and therefore these time stamps should be pruned. ♣

According to Lemma V-B.3, the server can avoid computing the lower bound or upper bound of time distance of each friend pair whose time distance is surely larger than the proximity threshold within time period $[t_{cur}, t_{cur} + \Delta T * a]$, and thus avoid probing those friend pairs. Therefore, many probing messages can be saved.

C. CLIENT-SIDE AND SERVER-SIDE ALGORITHMS

Based on the definition of mobile regions and the four pruning lemmas proposed above, we present our client-side and server-side algorithms of our TMRBD method in this subsection.

The client-side algorithm is described in Algorithm 3, which mainly aims to reduce the number of update messages sent by the client. When the client moves outside its mobile region (Line 1), the client sends an update message containing its speed *speed* and location *location* to the server (Line 2); When the client receives a probing message from the server (Line 4), it will also send an update message to the server (Line 5). Otherwise, the client will not send update messages.

Algorithm 3 Client-Side Algorithm of TMRBD

```
1 if Client O moves beyond its mobile region then
2 |  O.UpdateToServer(speed, location);
3 end
4 if Client O receives a probing message from the server
  then
5 |  O.UpdateToServer(speed, location);
6 end
```

The server-side algorithm is described in Algorithm 4, which mainly aims to reduce the probing messages sent by the server. The time epoch starts from the initial time stamp *initTS* and the server checks the proximity every ΔT time units (Line 1). At each time stamp t, the server first receives the update messages sent by the clients (Line 2); Then, the server checks whether each pair of moving friends is within proximity or not. We use the variable $nextTS[i]$ to denote the 'next' time stamp at which the server needs to check for the next time whether the i-th friend pair is within proximity. Note that the 'next' time stamp $nextTS[i]$, does not mean $t+\Delta T$, as the server may not need to check proximity of this friend pair for several continuous time epochs. Initially, set all $nextTS[i]$ to be the current time stamp t (Lines 3-5). For each pair of friends, (i) if the current time stamp is less than $nextTS[i]$ (Line 8), then at the current time stamp the server does not need to check the i-th friend pair (Line 9); (ii) if Lemma V-B.1 is satisfied (Line 11), then Lemma V-B.3 is also satisfied, so the server calculates the value of variable a in Lemma V-B.3, and in the next a time stamps the server does not need to check their proximity, so we use a to update $nextTS[i]$, and this pair of friends is surely not in proximity so they do not need to be checked (Lines 12-14); (iii) if Lemma V-B.1 is satisfied (Line 16), which means Lemma V-B.3 can be also satisfied, so the server calculates the value of variable a in Lemma V-B.3, and in the next a time stamps the server does not need to check their proximity, so we use the value of a to update $nextTS[i]$, and this pair of friends is surely not in proximity so they do not need to be checked (Lines 17-19); (iv) if Lemma V-B.2 is satisfied (Line 21), which means this pair of friends is surely within proximity, so the server needs to notify the two friends about their proximity (Line 22) and we update $nextTS[i]$ to be the next time stamp (Line 23); (v) otherwise (Line 25), the server probes the client that has not updated to the server at the current epoch (Lines 26-28, 29-31), then computes the time distance between them (line 32), notifies them about their proximity if the time distance is no larger than T_ϵ (Lines 33), and finally updates $nextTS[i]$ (Line 35).

Algorithm 4 Server-Side Algorithms of TMRBD

```
1  for (t = initTS; t ≤ MaxTS; t+ = ΔT) do
2  |  server.receiveUpdateFromClients(speed, location);
3  |  for (i = 0; i ≤ FriendPairs.size(); i + +) do
4  |  |  nextTS[i] = t;
5  |  end
6  |  for (i = 0; i ≤ FriendPairs.size(); i + +) do
7  |  |  ⟨O_m, O_n⟩ is the i-th friend pair;
8  |  |  if t < nextTS[i] then
9  |  |  |  continue;
10 |  |  end
11 |  |  if t_min(R_m(t), R_n(t)) > T_ε_m,n then
12 |  |  |  a = (int) (T_min(R_m(t),R_n(t))−T_ε_m,n) / ΔT;
13 |  |  |  nextTS[i] = t + (a + 1) * ΔT + 1;
14 |  |  |  continue;
15 |  |  end
16 |  |  if T_min(R_m(t), R_n(t)) > T_ε_m,n then
17 |  |  |  a = (int) (T_min(R_m(t),R_n(t))−T_ε_m,n) / ΔT;
18 |  |  |  nextTS[i] = t + (a + 1) * ΔT;
19 |  |  |  continue;
20 |  |  end
21 |  |  if T_max(R_m(t), R_n(t)) ≤ T_ε_m,n then
22 |  |  |  server.notify(O_m, O_n);
23 |  |  |  nextTS[i] = t + ΔT;
24 |  |  end
25 |  |  else
26 |  |  |  if ! client.update(O_m) then
27 |  |  |  |  server.probe(O_m);
28 |  |  |  end
29 |  |  |  if ! client.update(O_n) then
30 |  |  |  |  server.probe(O_n);
31 |  |  |  end
32 |  |  |  if T(O_m, O_n) ≤ T_ε_m,n then
33 |  |  |  |  server.notify(O_m, O_n);
34 |  |  |  end
35 |  |  |  nextTS[i] = t + ΔT;
36 |  |  end
37 |  end
38 end
```

VI. SERVER-SIDE COMPUTATIONAL COST OPTIMIZATION

Another objective is to reduce the server-side computational cost. To achieve this objective, we utilize two methods, i.e., offline junction-to-junction network distance precomputation, and parallel computing with OpenMP.

A. OFFLINE JUNCTION-TO-JUNCTION NETWORK DISTANCE PRECOMPUTATION

The calculation of time distance requires the lower bound and upper bound of the network distance between the two mobile regions, which should be calculated using the junction-to-junction network distance, i.e., the shortest network distance between two junctions. It is too time consuming if we compute the junction-to-junction network distance online every time. A promising way is to precompute each pair of junction-to-junction network distance offline. Therefore, we use the APSP (all-pair shortest path) algorithm, namely, Floyd algorithm, to precompute each pair of junction-to-junction network distance in the road network.

B. PARALLEL COMPUTING WITH OpenMP

OpenMP[2] is an industry-standard, platform-independent parallel programming library built into all modern C and C++ compilers. Unlike complex parallel platforms, OpenMP is designed to make it relatively easy to add parallelism to existing sequential programs.

In our proximity detection problem, there are so many moving clients in the network and each moving client has some certain number of friends. Therefore, there can be millions of friend pairs which need to be checked to determine whether they are within proximity or not. In this case, there are millions of friend pairs in the loop. To meet the requirement of proximity detection for millions of friend pairs at each server in each epoch, parallel computing with OpenMP is adopted, which can run the loop in parallel using multiple parallel threads according to the number of cores of the multi-core CPU. Thus each parallel thread only takes charge of a portion of the friend pairs so that the total computational time can be reduced to a great extent.

VII. EXPERIMENTAL STUDY

We conduct experiments to evaluate the performance of the proposed algorithms and techniques, including the communication cost of the proposed TMRBD method, the communication latency reduction by utilizing MEC, the communication cost influenced by utilizing MEC, and the computational cost of server-side computational cost optimization techniques.

A. EXPERIMENTAL SETUP

1) EXPERIMENTAL PREPARATION

Table 2 summarizes the default values and ranges of the parameters used in our experiments. We use the framework of network-based moving objects [50] to generate moving objects on three different road networks (Oldenburg road network, a part of New York city (NY) road network, called *p*NY for short, and San Joaquin road network). The Oldenburg road network[‡] contains 6105 nodes and 7035 edges, the *p*NY road network[§] contains 500 nodes and 1155 edges, and the San Joaquin road network[¶] contains 18263 nodes and 23874 edges. In total, we generate $N = 100,200$ moving objects during 100 time stamps. We normalize the spatial domain size of the road networks to $[0, 1000]^2$. After normalization, the average length of the edges of the three road networks become 7.395, 35.873, and 3.54, respectively.

[2]https://www.openmp.org/

TABLE 2. Default values of parameters.

Parameter	Default value	Range
N	100200	500∼100200
T_ϵ	3	3∼20
m	30	5∼40
R	7.395 (Oldenburg) or 35.873 (pNY) or 3.54 (San Joaquin)	0.01∼200

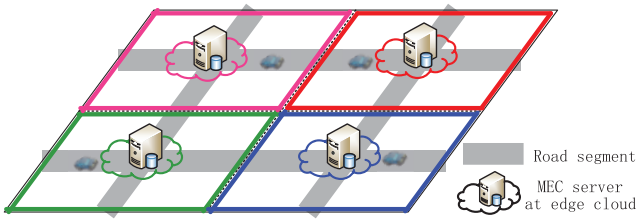


FIGURE 6. MEC server deployment scheme on the road network.

All experiments are implemented in Microsoft Visual Studio 2017 using C/C++ on a desktop with Inter(R) Core(TM) i7-7820 CPU @2.90 GHz processor and 32.0 GB RAM, running 64-bit Windows 10 operating system.

2) MEC SERVER DEPLOYMENT FOR EXPERIMENTS

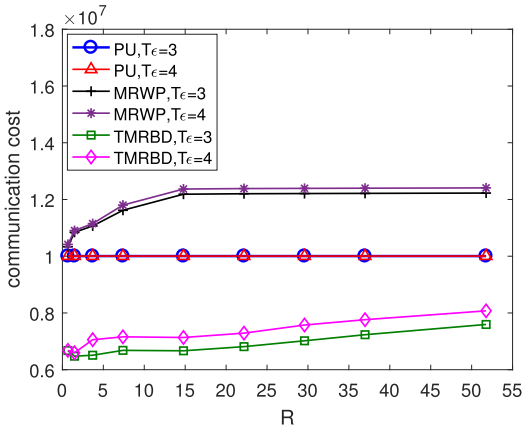
In our experiments, we deploy MEC servers on the road networks according to the following strategy. As shown in Fig. 6, we deploy four MEC servers **uniformly** on the road network. Since the whole road network has already been normalized to a square, it can be divided into four subsquares with equal size, and the four MEC servers are located at the center of each subsquare. Thus, the serving area of each MEC server is a circle with the MEC server as its center and $\frac{\sqrt{2}L}{4}$ as the radius, where L is the side length of the road network square.

B. EXPERIMENTS ON TMRBD

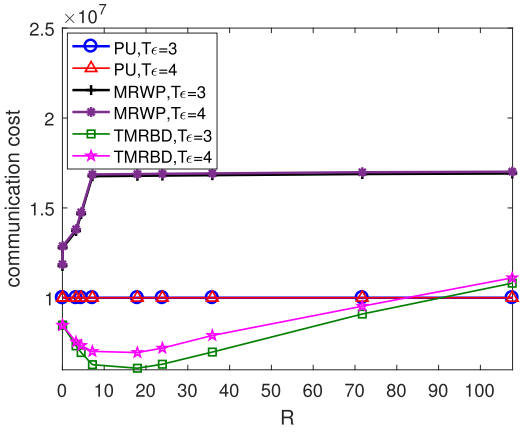
This subsection reports the results of performance evaluation of our TMRBD method. Unless pointed out specifically, the values of parameters are set as their default values given in Table 2.

In addition to the proposed TMRBD algorithms, we also simulate two baseline methods, i.e., PU (Periodic Update) method, and MRWP (Mobile Region Without Pruning) method. PU method does not involve a mobile region but allows a client to periodically (e.g., every time epoch) send update messages to the server. MRWP method sets mobile regions for moving users, but it does not adopt pruning strategies at the server side.

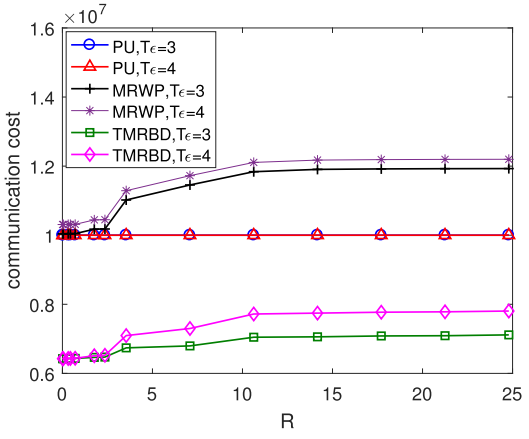
In order to compare the performance of the three approaches on communication cost, we plot the communication cost incurred by the three approaches as a function of the radius R of the mobile regions with respect to different values of T_ϵ ($T_\epsilon = 3, 4$) on the three road networks, as illustrated in Fig. 7. Observe that on all the three road networks,



(a) Oldenburg road network



(b) pNY road network



(c) San Joaquin road network

FIGURE 7. **Communication cost comparison w.r.t. the mobile region radius R.**

no matter how R changes and no matter for $T_\epsilon = 3$ or $T_\epsilon = 4$, (i) the PU method and the MRWP method always induce large communication cost; and (ii) our proposed TMRBD method incurs the least communication cost.

The reasons are as follows. (i) The PU method has no mobile regions and requires every client to send update messages in every epoch, so that it induces large updating cost and it shows straight lines in the figure as it has nothing to do with the radius R. (ii) Though the MRWP maintains a mobile region for each client, it does not adopt pruning strategies so that the server has to send probing messages to the client if the client has not sent update messages to the server at each epoch, which leads to large probing cost in addition to the updating cost. (iii) Our proposed TMRBD method adopts both the client-side updating strategy and the server-side pruning strategies so that it reduces both the client-side updating cost and the server-side probing cost. These results demonstrate that our proposed TMRBD method can reduce the communication cost effectively.

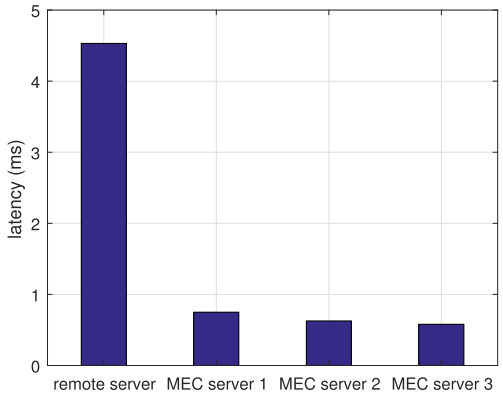


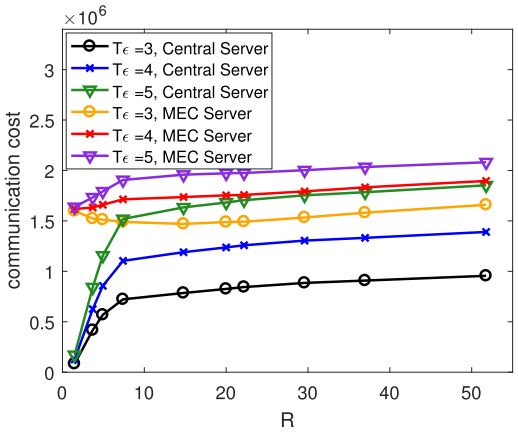
FIGURE 8. **Communication latency of traditional client-server architecture vs. latency of our MEC based architecture. "MEC server 1" "MEC server 2" and "MEC server 3" refer to the three MEC servers which are 35 km, 23.6 km, 17.7 km away, respectively.**

C. EXPERIMENTS ON LATENCY REDUCTION BY UTILIZING MEC

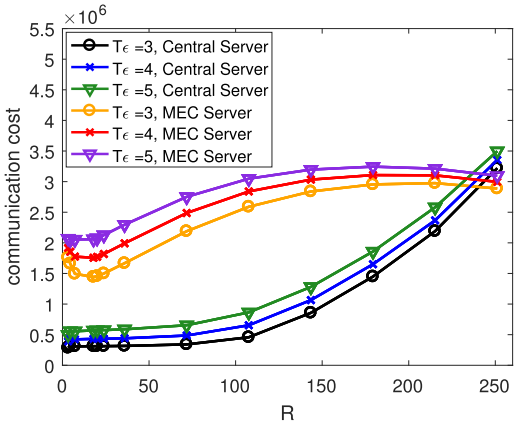
In the real-world Oldenburg, New York, and San Joaquin road networks, the side length can be more than 100 km. Thus, in a traditional client-server architecture, the longest distance from the traditional central server to a client may be at least $100\sqrt{2} \approx 141$ km. While in our MEC based architecture, as the MEC server is located at the center of each subregion of the road network, the longest distance from the MEC server to a client may be $25\sqrt{2} \approx 35$ km if we deploy MEC server according to Section VII-A.2. In fact, we can deploy more (e.g., 9, 16) MEC servers to cover the road network, and thus the longest distance from the MEC server to a client can be smaller, e.g., $\frac{100}{6}\sqrt{2} \approx 23.6$ km, $\frac{100}{8}\sqrt{2} \approx 17.7$ km.

In order to compare the latency of adopting MEC enhanced architecture with the latency of using the traditional client-server architecture, we deployed one remote Ali cloud server which is about 141 km away from the lab. we also deployed three MEC servers which are 35 km, 23.6 km, and 17.7 km away, respectively. The MEC servers are connected to users' terminals through Fiber-Wireless network. We test and plot the RTT (Round-Trip Time) latency of communicating with the traditional remote server and three MEC servers, as shown in Fig. 8.

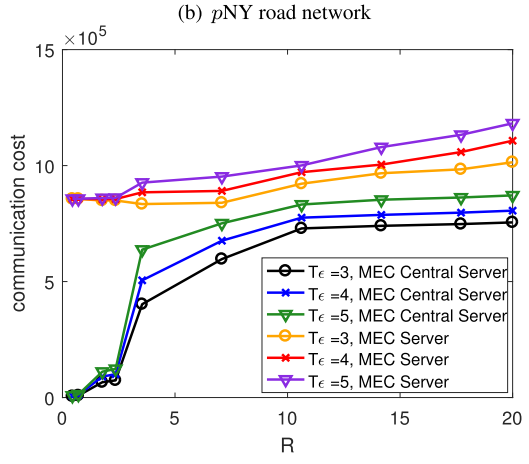
Observe that the latency of communicating with MEC servers are conspicuously lower, i.e., less than 1 ms, while the latency with respect to the remote central server can be up to



(a) Oldenburg road network

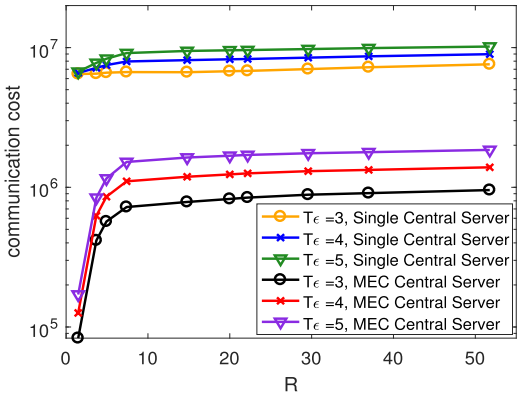


(b) *p*NY road network

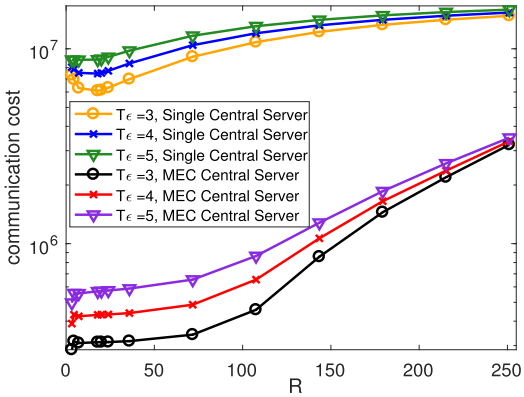


(c) San Joaquin road network

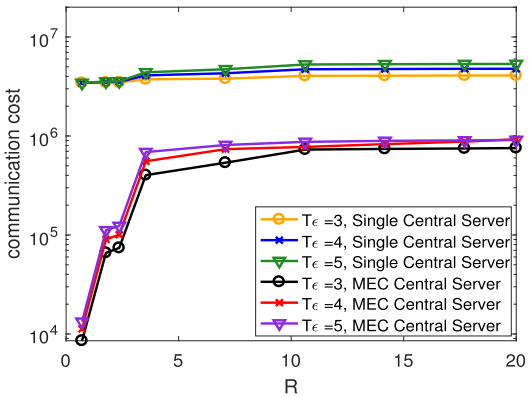
FIGURE 9. Communication cost comparison: MEC central server vs. MEC server.



(a) Oldenburg road network



(b) *p*NY road network



(c) San Joaquin road network

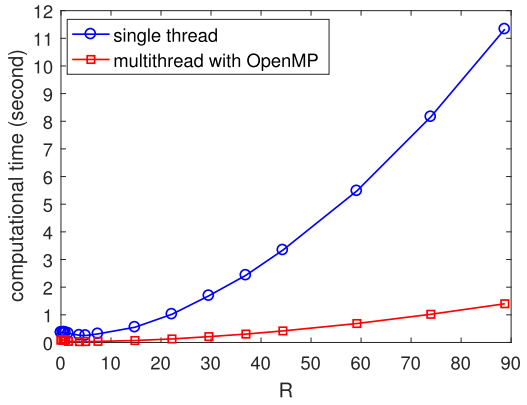
FIGURE 10. Communication cost comparison: traditional single central server vs. MEC central server.

6 times at least. Therefore, we can conclude that our proposed MEC enhanced proximity detection architecture can reduce the communication latency to a large extent and can satisfy the low-latency requirement.

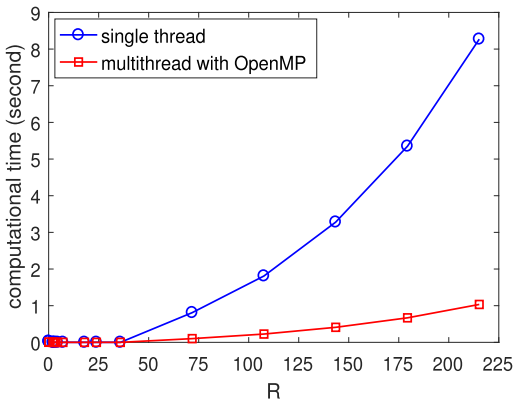
D. EXPERIMENTS ON COMMUNICATION COST INFLUENCED BY MEC

In the traditional client-server architecture, each client communicates with the central server, so the central server is involved in quite a large number of communication messages. However, in our proposed proximity detection architecture based on MEC, besides the existence of the central server, MEC servers are deployed at multiple edge clouds. In this regard, in the MEC enhanced architecture, both the central server and the MEC servers are involved in the communication messages.

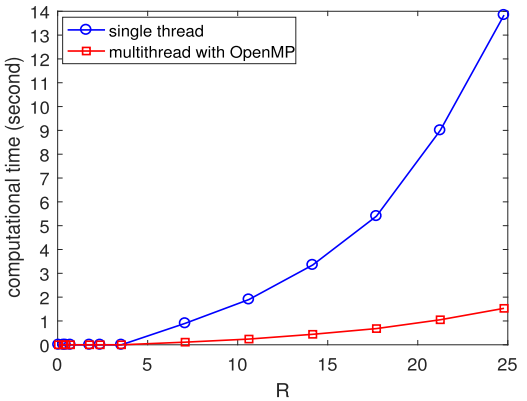
We compare the communication cost involved by central server and each MEC server under our MEC enhanced architecture, as shown in Fig. 9, where we plot the average communication cost associated with each MEC server and the communication cost associated with the MEC central server as a function of the radius R, and 'MEC central server'



(a) Oldenburg road network



(b) *p*NY road network



(c) San Joaquin road network

FIGURE 11. Running time comparison: Single-thread vs. multi-threading processing.

refers to the central server in our MEC enhanced architecture. Observe that each MEC server is involved in a large part of communication cost whereas central server is only involved in a small part of communication cost, which demonstrates that MEC servers play a more primary role in our MEC enhanced architecture.

We also implement our proximity detection algorithms in the client-server architecture in addition to our proposed MEC based architecture, to compare the communication cost involved by the central server in our MEC architecture and the central server in the traditional client-server architecture. As shown in Fig. 10, observe that when we apply proximity detection algorithms in the traditional client-server architecture, the single central server is involved in very large communication cost, whereas when we apply proximity detection algorithms in our MEC enhanced architecture, the central server in MEC architecture, i.e., MEC central server, is involved in quite small communication cost. Therefore, we can conclude that MEC architecture saves the communication cost between the clients and the central server.

E. EXPERIMENTS ON SERVER-SIDE COMPUTATIONAL COST OPTIMIZATION TECHNIQUES

By utilizing the server-side computational cost optimization techniques such as parallel computing with OpenMP, we can reduce much computational time at the server side.

As shown in Fig. 11, we compare the computational time with respect to different mobile region radii R when using one single thread and multi-thread with OpenMP, respectively, on the Oldenburg road network (Fig. 11(a)), the *p*NY road network (Fig. 11(b)), and the San Joaquin road network (Fig. 11(c)). Observe that no matter on Oldenburg road network, or *p*NY road network, or San Joaquin road network, the server-side computational time after using multithread with OpenMP, is reduced to a great extent, which is less than or around 1 second even when the mobile region radius R is quite large. Therefore, we can conclude that our computational cost optimization techniques can reduce the computational cost effectively.

VIII. CONCLUSION

In this paper, we propose a proximity detection problem in time-aware road networks and use time distance as the metric of judging whether two objects are within proximity or not. To reduce the communication latency between the server and the users, we propose a proximity detection architecture based on MEC. To reduce the communication cost (number of messages), we propose a mobile region based proximity detection method, namely, TMRBD, with client-side and server-side algorithms. To reduce the computational cost, we propose server-side computational time optimization techniques. Experimental results demonstrate that (i) our MEC enhanced proximity detection architecture can effectively reduce the communication latency, (ii) our mobile region based detection methods can effectively reduce the communication cost compared to some baseline methods, and (iii) our server-side computational time optimization methods can reduce the computational running time to a great extent.

REFERENCES

[1] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *Proc. 10th IEEE Int. Conf. Intell. Syst. Control*, Jan. 2016, pp. 1–8.
[2] M. T. Beck, S. Feld, U. Pützschler, and C. Linnhoff-Popien, "Mobile edge computing," *Informatik-Spektrum*, vol. 39, no. 2, pp. 108–114, Apr. 2016.
[3] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G, ETSI White Paper," vol. 11, no. 11, pp. 1–6, Sep. 2015.

[4] *5G Vision—The 5G Infrastructure Public Private Partnership: The Next Generation of Communication Networks and Services*. Accessed: Feb. 2015. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2015/02/5G-Vision-Brochure-v1.pdf

[5] P. K. Agarwal, L. Arge, and J. Erickson, "Indexing moving points (extended abstract)," in *Proc. 19th ACM SIGMOD-SIGACT-SIGART Symp. Princ. Database Syst.*, New York, NY, USA, May 2000, pp. 175–186. doi: 10.1145/335168.335220.

[6] X. Huang, C. S. Jensen, and S. Šaltenis, "Multiple *k* nearest neighbor query processing in spatial network databases," in *Advances in Databases and Information Systems*. Berlin, Germany: Springer, 2006, pp. 266–281.

[7] C. S. Jensen J. Kolářvr, T. B. Pedersen, and I. Timko, "Nearest neighbor queries in road networks," in *Proc. 11th ACM Int. Symp. Adv. Geographic Inf. Syst.*, Nov. 2003, pp. 1–8.

[8] M. Kolahdouzan and C. Shahabi, "Voronoi-based K nearest neighbor search for spatial network databases," in *Proc. 30th Int. Conf. Very Large Data Bases*, vol. 30, 2004, pp. 840–851.

[9] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao, "Query processing in spatial network databases," in *Proc. 29th Int. Conf. Very Large Data Bases*, vol. 29, Sep. 2003, pp. 802–813.

[10] S. Saltenis and C. S. Jensen, "Indexing of moving objects for location-based services," in *Proc. 18th Int. Conf. Data Eng.*, Feb./Mar. 2002, pp. 463–472.

[11] S. Šaltenis, C. S. Jensen, S. T. Leutenegger, and M. A. Lopez, "Indexing the positions of continuously moving objects," *SIGMOD Rec.*, vol. 29, no. 2, pp. 331–342, May 2000.

[12] Y. Tao, D. Papadias, and J. Sun, "The tpr*-tree: An optimized spatio-temporal access method for predictive queries," in *Proc. 29th Int. Conf. Very Large Data Bases (VLDB)*, Sep. 2003, pp. 790–801.

[13] J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D. L. Lee, "Location-based spatial queries," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2003, pp. 443–454.

[14] C. S. Jensen, D. Lin, and B. C. Ooi, "Query and update efficient B+−tree based indexing of moving objects," in *Proc. 13th Int. Conf. Very Large Data Bases (VLDB)*, vol. 30, Sep. 2004, pp. 768–779.

[15] A. Amir, A. Efrat, J. Myllymaki, L. Palaniappan, and K. Wampler, "Buddy tracking—Efficient proximity detection among mobile friends," *Pervas. Mobile Comput.*, vol. 3, no. 5, pp. 489–511, Oct. 2007.

[16] A. Küpper and G. Treu, "Efficient proximity and separation detection among mobile targets for supporting location-based community services," *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 10, no. 3, pp. 1–12, Jul. 2006.

[17] Y. Cai, K. A. Hua, and G. Cao, "Processing range-monitoring queries on heterogeneous mobile objects," in *Proc. IEEE Int. Conf. Mobile Data Manage.*, Jan. 2004, pp. 27–38.

[18] B. Gedik and L. Liu, "MobiEyes: Distributed processing of continuously moving queries on moving objects in a mobile system," in *Advances in Database Technology—EDBT*. Berlin, Germany: Springer, 2004, pp. 67–87.

[19] H. Hu, J. Xu, and D. L. Lee, "A generic framework for monitoring continuous spatial queries over moving objects," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2005, pp. 479–490.

[20] G. S. Iwerks, H. Samet, and K. Smith, "Continuous *k*-nearest neighbor queries for continuously moving points with updates," in *Proc. 29th Int. Conf. Very Large Data Bases (VLDB)*, vol. 29, Sep. 2003, pp. 512–523.

[21] N. Koudas, B. C. Ooi, K.-L. Tan, and R. Zhang, "Approximate NN queries on streams with guaranteed error/performance bounds," in *Proc. 13th Int. Conf. Very Large Data Bases (VLDB)*, vol. 30, Aug. 2004, pp. 804–815.

[22] M. F. Mokbel, X. Xiong, and W. G. Aref, "SINA: Scalable incremental processing of continuous queries in spatio-temporal databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2004, pp. 623–634.

[23] K. Mouratidis, D. Papadias, and M. Hadjieleftheriou, "Conceptual partitioning: An efficient method for continuous nearest neighbor monitoring," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2005, pp. 634–645.

[24] S. Prabhakar, Y. Xia, D. V. Kalashnikov, W. G. Aref, and S. E. Hambrusch, "Query indexing and velocity constrained indexing: Scalable techniques for continuous queries on moving objects," *IEEE Trans. Comput.*, vol. 51, no. 10, pp. 1124–1140, Oct. 2002.

[25] X. Xiong, M. F. Mokbel, and W. G. Aref, "SEA-CNN: Scalable processing of continuous k-nearest neighbor queries in spatio-temporal databases," in *Proc. 21st Int. Conf. Data Eng.*, Apr. 2005, pp. 643–654.

[26] X. Yu, K. Q. Pu, and N. Koudas, "Monitoring k-nearest neighbor queries over moving objects," in *Proc. 21st Int. Conf. Data Eng.*, Apr. 2005, pp. 631–642.

[27] M. R. Kolahdouzan and C. Shahabi, "Continuous K-nearest neighbor queries in spatial network databases," in *Proc. 2nd Int. Workshop STDBM*, Toronto, ON, Canada, Aug. 2004, pp. 33–40.

[28] H.-J. Cho, R. Jin, and T.-S. Chung, "A collaborative approach to moving *k*-nearest neighbor queries in directed and dynamic road networks," *Pervasive Mob. Comput.*, vol. 17, pp. 139–156, Feb. 2015.

[29] Y. Gao, B. Zheng, G. Chen, W. C. Lee, K. C. K. Lee, and Q. Li, "Visible reverse k-nearest neighbor query processing in spatial databases," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1314–1327, Sep. 2009.

[30] H. J. Cho and C. W. Chung, "An efficient and scalable approach to CNN queries in a road network," in *Proc. 31st Int. Conf. Very Large Data Bases*, K. Böhm, C. S. Jensen, L. M. Haas, M. L. Kersten, P. Larson, and B. C. Ooi, Eds. Trondheim, Norway: ACM, 2005, pp. 865–876. [Online]. Available: http://www.vldb2005.org/program/paper/fri/p865-cho.pdf

[31] K. Mouratidis, M. L. Yiu, D. Papadias, and N. Mamoulis, "Continuous nearest neighbor monitoring in road networks," in *Proc. 32nd Int. Conf. Very Large Data Bases*, U. Dayal, K. Whang, D. B. Lomet, G. Alonso, G. M. Lohman, M. L. Kersten, S. K. Cha, and Y. Kim, Eds. Seoul, South Korea: ACM, Sep. 2006, pp. 43–54. [Online]. Available: http://www.vldb.org/conf/2006/p43-mouratidis.pdf

[32] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao, "Query processing in spatial network databases," in *Proc. 29th Int. Conf. Very Large Data Bases (VLDB)*, J. C. Freytag, P. C. Lockemann, S. Abiteboul, M. J. Carey, P. G. Selinger, and A. Heuer, Eds. San Mateo, CA, USA: Morgan Kaufmann, Sep. 2003, pp. 802–813.

[33] D. Wu, M. L. Yiu, C. S. Jensen, and G. Cong, "Efficient continuously moving top-*k* spatial keyword query processing," in *Proc. IEEE 27th Int. Conf. Data Eng.*, Apr. 2011, pp. 541–552.

[34] D. Wu, M. L. Yiu, G. Cong, and C. S. Jensen, "Joint top-*k* spatial keyword query processing," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 10, pp. 1889–1903, Oct. 2012.

[35] Y. Gao, J. Zhao, B. Zheng, and G. Chen, "Efficient collective spatial keyword query processing on road networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 2, pp. 469–480, Feb. 2016.

[36] B. Yao, X. Xiao, F. Li, and Y. Wu, "Dynamic monitoring of optimal locations in road network databases," *The VLDB J.*, vol. 23, no. 5, pp. 697–720, Oct. 2014.

[37] L. Zhan, Y. Zhang, W. Zhang, and X. Lin, "Finding top *k* most influential spatial facilities over uncertain objects," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 12, pp. 3289–3303, Dec. 2015.

[38] G. Treu and A. Küpper, "Efficient proximity detection for location based services," in *Proc. Workshop Positioning, Navigat. Commun. (WPNC)*, Mar. 2005, pp. 165–173.

[39] G. Treu, T. Wilder, and A. Küpper, "Efficient proximity detection among mobile targets with dead reckoning," in *Proc. 4th ACM Int. Workshop Mobility Manage. Wireless Access*, Oct. 2006, pp. 75–83.

[40] Z. Xu and A. Jacobsen, "Adaptive location constraint processing," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2007, pp. 581–592.

[41] M. L. Yiu, S. Šaltenis, and K. Tzoumas, "Efficient proximity detection among mobile users via self-tuning policies," in *Proc. VLDB Endowment*, Sep. 2010, vol. 3, nos. 1–2, pp. 985–996.

[42] S. J. Kazemitabar, F. Banaei-Kashani, S. J. Kazemitabar, and D. McLeod, "Efficient batch processing of proximity queries by optimized probing," in *Proc. 21st ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, New York, NY, USA, Nov. 2013, pp. 84–93.

[43] Y. Xu, D. Zhang, M. Zhang, D. Li, X. Wang, and H. T. Shen, "Continuous proximity detection via predictive safe region construction," in *Proc. IEEE 34th Int. Conf. Data Eng. (ICDE)*, Apr. 2018, pp. 629–640.

[44] C. Zhang and J. Luo, "Track others if you can: Localized proximity detection for mobile networks," *Wireless Netw.*, vol. 20, no. 6, pp. 1477–1494, Aug. 2014. doi: 10.1007/s11276-014-0690-5.

[45] H. Kriegel, P. Kröger, P. Kunath, M. Renz, and T. Schmidt, "Proximity queries in large traffic networks," in *Proc. 15th Annu. ACM Int. Symp. Adv. Geographic Inf. Syst.*, Nov. 2007, p. 21.

[46] H. Kriegel, P. Kröger, and M. Renz, "Continuous proximity monitoring in road networks," in *Proc. 16th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, Nov. 2008, p. 12.

[47] Z. Xu and H. A. Jacobsen, "Processing proximity relations in road networks," in *Proc. Int. Conf. Manage. Data*, Jun. 2010, pp. 243–254.

[48] Y. Liu, H. S. Seah, and G. Cong, "Efficient proximity detection among mobile objects in road networks with self-adjustment methods," in *Proc. 21st ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, Nov. 2013, pp. 124–133.

[49] H.-P. Kriegel, P. Kröger, M. Renz, and F. D. Winter, "Proximity queries in time-dependent traffic networks using graph embeddings," in *Proc. 4th ACM SIGSPATIAL Int. Workshop Comput. Transp. Sci.*, Nov. 2011, pp. 45–54.
[50] T. Brinkhoff, "A framework for generating network-based moving objects," *GeoInformatica*, vol. 6, no. 2, pp. 153–180, 2002.

YAQIONG LIU received the bachelor's degree in computer science and technology and the second bachelor's degree in financial management from Tianjin University, China, and the Ph.D. degree in computer science and engineering from Nanyang Technological University, Singapore. She is currently a Lecturer with the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, China. Her research interests include edge computing, the IoT, spatial query processing, GIS, data mining, and image animation.

MUGEN PENG received the B.E. degree in electronics engineering from the Nanjing University of Posts and Telecommunications, China, in 2000, and the Ph.D. degree in communication and information system from the Beijing University of Posts and Telecommunications (BUPT), China, in 2005. In 2014, he was also an Academic Visiting Fellow with Princeton University, USA. He is currently a Full Professor with BUPT. He has authored/coauthored over 70 refereed IEEE journal articles and over 200 conference proceeding articles. His main research interests include wireless communication theory, radio signal processing, and convex optimizations, with particular interests in cooperative communication, radio network coding, self-organizing networks, heterogeneous networks, and cloud communication.

GUOCHU SHOU is currently a Professor with the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications. His research interests include access network and edge computing, fiber and wireless network virtualization, network construction and routing, and the mobile Internet and applications.