IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 21, 2019, accepted December 16, 2019, date of publication December 26, 2019,
date of current version January 15, 2020.

Digital Object Identifier 10.1109/ACCESS.2019.2962392

A Novel Vector-Based Dynamic Path Planning Method in Urban Road Network

ZHI CAI, XUERUI CUI, XING SU, QING MI, LIMIN GUO, AND ZHIMING DING
College of Computer Science, Beijing University of Technology, Beijing 100124, China

Corresponding author: Xing Su (xingsu@bjut.edu.cn)

This work was supported by the National Key R&D Program of China (Grants No. 2017YFC0803300), the Beijing Natural Science Foundation (Grant No. 4172004, 4192004), the National Natural Science Foundation of China (Grants No. 61703013, 41971366), Beijing Municipal Education Commission Science and Technology Program (Grants No. KM201810005024, KM201810005023).

ABSTRACT The optimal path planning is one of the hot spots in the research of intelligence transportation and geographic information systems. There are many productions and applications in path planning and navigation, however due to the complexity of urban road networks, the difficulty of the traffic prediction increases. The optimal path means not only the shortest distance in geography, but also the shortest time, the lowest cost, the maximum road capacity, etc. In fast-paced modern cities, people tend to reach the destination with the shortest time. The corresponding paths are considered as the optimal paths. However, due to the high data sensing speed of GPS devices, it is different to collect or describe real traffic flows. To address this problem, we propose an innovative path planning method in this paper. Specially, we first introduce a crossroad link analysis algorithm to calculate the real-time traffic conditions of crossroads (i.e. the *CrossRank* values). Then, we adopt a *CrossRank* value based *A-Star* for the path planning by considering the real-time traffic conditions. To avoid the high volume update of *CrossRank* values, a *R-Tree* structure is proposed to dynamically update local *CrossRank* values from the multi-level subareas. In the optimization process, to achieve desired navigation results, we establish the traffic congestion coefficient to reflect different traffic congestion conditions. To verify the effectiveness of the proposed method, we use the actual traffic data of Beijing. The experimental results show that our method is able to generate the appropriate path plan in the peak and low dynamic traffic conditions as compared to online applications.

INDEX TERMS CrossRank, path planning, vector, heuristic algorithm.

I. INTRODUCTION

In the modern transportation system, the main traffic problems faced by cities are frequent traffic accidents, traffic environment deterioration, traffic congestion and so on. These problems may lead to the waste of time and resources even threaten personal safety. Most countries have similar problems and pitfalls. Even if the number of new roads is increasing, the pressure of traffic is not alleviated. It is clear that road infrastructures cannot solve traffic problems alone. Therefore, the field of intelligence transportation system research is gradually developed, which are the main way to solve the traffic problems. With the research of intelligent transportation system, there has been a vehicle navigation system and route guiding ideology.

The path planning problem is the most fundamental problem and plays a key role in the vehicle navigation system. In the road network, the path planning is widely used in areas such as the lowest cost path selection, the communication line construction and maintenance, the transportation traffic analysis, the city emergency rescue and evacuation, the urban public transportation network planning and so on. The optimal path planning can maximize the use of time and resources, alleviate traffic congestion and reduce vehicle consumption. Most importantly, can help people to find and choose the best traveling paths.

Meanwhile, some of the current traffic control and monitoring system or navigating application use GPS data representing the states of urban roads (i.e. different color e.g. red, yellow, green to distinguish the traffic conditions), based on which, path planning methods are also applied such conditions. However, real traffic flow data is very difficult to collect or well described from such few collected GPS data, this may bring a lot of misleading path, crossroad or regions that with heavy traffic or more easily to get traffic congestions.

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

The premise of path planning is to obtain road conditions from the starting point to the destination location, including road distances, road congestion, etc., and then plan the optimal path according to the real-time condition of roads. Traffic flow data can help us to obtain the road congestion condition in a timely manner, the existing methods collect traffic flow data by using geomagnetic sensors or HD video cameras. However, both methods face the issues of high cost, hard to set, or weak usage in different conditions of visibility. Probe vehicle GPS data can well represent traffic speed with very good accuracy, but may not be able to collect real traffic flow data. Therefore, we obtain the congestion condition of the road section through vector information.

In the aspect of road network conditions representation, most of the existing algorithms or applications use average speed or traffic flow volume to measure the conditions, we argue that such methods may not be suitable to describe the road network states precisely according to single or average numerical value within a certain timestamp or time period.

In view of aforementioned limitations, contributions of this paper are summarized as follows.

1) Initially, this paper introduced a continuously real-time mathematical function, for each GPS data provider, our proposed function generates its corresponding vectors according to its sampling data, which describes the current road condition according to the number of vector of real-time road condition (i.e. a better condition should have a small volume of vectors since the speed of vehicles are not changing sharply, otherwise indicating a complex traffic), in other words, we not only consider amount of speed or volume, but also their updating frequencies in traffic state representation.
2) Meanwhile, according to the topology of road network and inspired from traffic flow authority transfer rate, we proposed a crossroads link analysis algorithm which is a *PageRank* fashion approach to calculate the *CrossRank* (*CR*) values of crossroads in road network, as well as an *R-Tree* structured area splitting method is proposed to dynamically update *CR* values of local crossroads from multi-level subareas, so as to avoid the high volume of real-time updates and calculations. We use the vector based real-time traffic state representation as authority transfer weight, by analyzing the complexity of road network, greater connections is more likely to bring the traffic congestion. Thus, a real-time road network ranking can be established.
3) Moreover, we combine the real-time *CR* values as heuristic variable into *A-Star* algorithm based path planning method to achieve the dynamic calculations, while traffic congestion coefficient is established to reflect different traffic congestion conditions, so as to provide desirable path planning results. Our proposed path planning method provided a vector generation considering a group of continuous GPS data beyond the static conditions (e.g. road distances, max-speed limitations, max-volumes, number of traffic lights etc.).

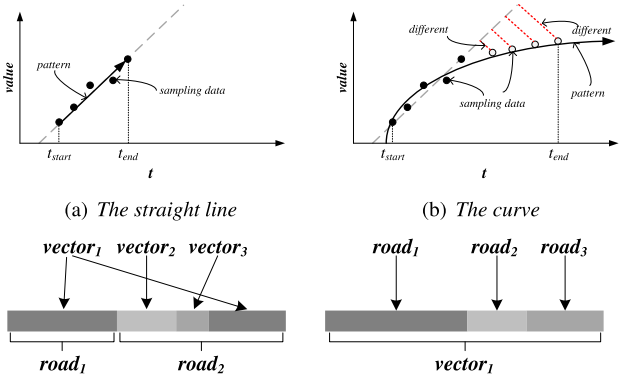
The rest parts of this paper is organized as follows. Section II reviews related work. Section III describes the problem and definitions. Section IV introduces the *CR* based *A-Star* algorithm for the path planning. Section V introduces the implementation of the path planning method and Section VI presents the experimental results and evaluations. Section VII concludes this paper.

II. RELATED WORK

In this section, we first describe the concept of vector based trajectory extraction, which is the basis of this paper. We then review prior studies related to the link analysis, navigation, etc. To the best of our knowledge, there is no previous work that focuses on the computation of vector based navigation.

A. THE EXTRACTION OF ROAD SEGMENT VECTOR BASED ON TRAJECTORY DATA

The vector extraction has been applied to many fields of research [30]. Representing the data in the form of vectors can reduce the amount of data storage, and reduce the amount of data exchanging in the query and analysis, according to our previous work [4] (e.g. In Figures 1(a) and 1(b), it can be seen that the data storage can be saved by using a vector. In other words, a vector is a kind of function that is related to *time* and *value*). We claim that if a crossroad has many link-in edges, this crossroad will be very busy during the rush hour according to the link analysis algorithm. Otherwise, it may be the choice of the path planning. In the case of the relationship between *Road* and *Vector* (e.g. Figures 1(c) and 1(d)), a *Road* can be represented by several *Vectors* denoting that this single road is busy, whereas a *Vector* can include several *Roads* (with the same vector, e.g. highway, circuit, etc) to represent a smooth traffic condition.



(a) *The straight line* (b) *The curve*

(c) *Roads include M vectors* (d) *A vector in N roads*

FIGURE 1. The relationship between roads and vectors.

Thus, vectors can be directly used to analyze and represent the traffic condition. This paper studies a navigation method based on the real-time trajectory data in an intelligent transportation system. Through the vector extraction of vehicle's

real-time trajectory data, the number of vectors in road segments can be obtained. The number of road segment vectors reflects the traffic congestion to some extent. The worse the traffic congestion is, the more number of vectors can be extracted. On the contrary, the better the traffic condition is, the less number of vectors can be extracted. Then, based on the idea of the aforementioned crossroad link analysis, the vector of road segment is introduced into the path planning of the navigation method.

B. THE LINK ANALYSIS ALGORITHM

The *PageRank* algorithm is a calculation technology based on hyperlinks between webpages, and is one of the important factors of webpage ranks. *Google* uses it to reflect the relevance and importance of webpages. In the search engine optimization operations, it is often used to evaluate the effectiveness of the webpage priority.

The link analysis algorithms have been widely used in keyword search (i.e. [2], [8], [9], [12]). As one of the most classical link analysis algorithms, the *PageRank* algorithm calculates the *PageRank* value of each webpage and ranks the importance of the webpage according to this value. The idea is to simulate a leisure Internet user, who first randomly opens a web page, spends a few minutes on the web page and then jumps to a link on the webpage. The Internet user is aimlessly opening different webpages and the *PageRank* algorithm is to estimate the probability distribution of the Internet user to access different webpages.

Therefore, in a strongly connected graph, the *PageRank* value has the following four characteristics.

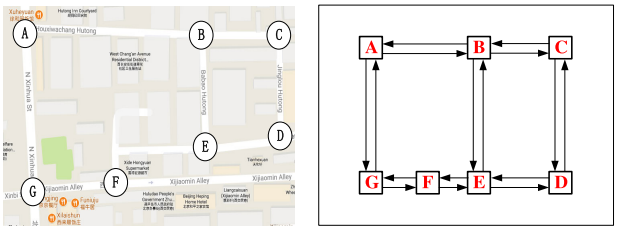
1) Regardless of other factors, the *PageRank* value can reflect the probability of a node (webpage) being randomly accessed;
2) The sum of the *PageRank* values of all nodes is a fixed value of 1;
3) The initial *PageRank* value of each node is $\frac{1}{n}$;
4) After several rounds of iteration, the *PageRank* value tends to be stable.

For a graph containing N nodes, the calculation of the *PageRank* value is described as follows.

$$PR(A) = (\frac{PR(B)}{L(B)} + \ldots) * d + \frac{(1-d)}{N}, \qquad \text{(II.1)}$$

where L is the number of edges, PR is the *PageRank* value, d is called the general damping coefficient value, which is 0.85 and N is the number of nodes in the graph.

We introduce the linking analysis algorithm into the urban road network, we name this novel method as the crossroad link analysis algorithm. The calculated value of each crossroad in the road network is referred to the *CR* value, which is applied to the path planning system. For the road network in Figure 2(a) as the example, we set each crossroad as a node, while the corresponding road segments that are connected two crossroads are considered as the edges (in most cases, two bidirectional edges in Figure 2(b)). Thus, we can conduct the link analysis algorithm on the road network graph.



(a) *The road network* (b) *The road network graph*

FIGURE 2. Crossroads and links in the road network.

C. THE NAVIGATION SYSTEMS FOR THE PATH PLANNING

With the increasing of the population in the urban area and the improvement of people's living standard, there are more and more vehicles in cities. However, the construction of roads and bridges are far behind the growth of vehicles in cities, which brings serious problems to the public transportation. As an important part of the transportation system, vehicle navigation method and the path planning algorithms can effectively solve many traffic problems, such as the traffic congestion [31], [39], the traffic accidents, the traffic control [34] and so on.

Many methods and algorithms have been proposed for the navigation and the path planning from different prespectives, such as graph based models [5], [10], *A-Star* based algorithms [18], [27], behavior based models [24], [29], self-learning based methods [29], sensor fusion based methods [38], neural network approaches [20]–[22], [33], fuzzy logic algorithms [17], [24], etc. Reference [29] developed an improved algorithm for mobile robot path planning in unknown environments, where the proposed algorithm is associated with skinner operant conditioning and a bio-inspired neural dynamics. Reference [38] provided a Dempster-Shafer inference rule to construct a 2D occupancy grid map fused with an ultrasonic sensor data, where the navigation in the unknown environment is performed by a shunting-type neural network. Reference [13] presented an improved self-organizing map neural network, which is integrated with a velocity synthesis approach for autonomous underwater vehicles. However, the aforementioned navigation methods and path planning algorithms are developed for mobile robots, unknown environments, underwater vehicles and behavior-based motivation.

In recent years, some navigation methods have been proposed for the navigation in urban road networks [6], [11], [15], [26], [37]. Some researches focus on the fastest path, [28] proposed a method to select the fastest road, records the travel time of distribution vehicles on each road, replaces the distance value with the travel time value, and uses the improved Floyd algorithm to calculate the fastest route to complete the distribution task by taking travel time, weather condition, intersection number and other factors into consideration, [16] improved the faster criterion in vehicle routing by extending the bi-delta distribution to the binormal distribution, [19] implemented comfort-based route planning.

References [7], [14], [35], [36] focus on the shortest path between specified starting and target points. Reference [25] proposed a sharing considered route assignment mechanism for fair taxi route recommendations (SCRAM), which is capable of providing taxi drivers with more efficient routes that have the least driving cost per customer. Reference [32] proposed an asynchronous and optimized method to efficiently compute shortest path over a dynamic graph. The proposed system can achieve low query latency asynchronous query processing and fast response time on query updates. Reference [3] used a multi-label A* search to optimize the problem of finding routes in road network. Reference [1] proposed an approach to multi-level shared-memory parallel graph partitioning to improve the speedups for a variety of large graphs. Reference [23] proposed a fast and simple lock-free concurrent hash table based on linear probing.

With more and more traffic vehicles in cities, the problem of traffic congestion becomes more serious. The shortest path may not be able to obtain the shortest driving time. Therefore, road traffic congestion should be considered in real time in road traffic planning. And the existing methods collect traffic flow data by using geomagnetic sensors or HD video camera. However, both of the methods are facing the issues of high cost, difficult to set, or weak usage in different conditions of visibility. Therefore, we obtain the congestion condition of the road section through vector information.

III. DEFINITIONS

In this section, we introduce the vector based *CR* value in the road network, the map-reduce framework to calculate the *CR* values and the real-time updating the *CR* values.

Definition 1 (Vector extraction): Given a moving object o and a period of time t, the vector P denote the continuous points (longitude and latitude collected from GPS device) obtained during the time t of the moving object o, the vector extraction is defined as a function i.e. Equation III.1 of the process of fitting the points P. Equation III.1 by using *bezier* curve fitting algorithm to meet the points P of curve of the tangent vector and the vector (p_0, p_1) parallel, and the terminal point of the curves of the tangent vector and vector (p_{m-1}, p_m) parallel, which is consist of three cases in the following equation. Case 1 is a simple equation, it is a straight line between two points. Case 2 is a quadratic equation, it is a curve that is determined by three points. Case 3 is a cubic equation, it is a curve that is determined by four points. Cubic *bezier* curves are specified by their endpoints (often called knots) and two control points. In our previous work [4], we have introduced the vector storage and query for intelligent transport system in detail.

$$P(t) = \begin{cases} (1-t)P_0 + tP_1 & 0 \le t \le 1 \\ (1-t)^2P_0 + 2(1-t)tP_1 + t^2P_2 & 0 \le t \le 1 \\ (1-t)^3P_0 + 3(1-t)^2tP_1 & \\ \quad +3(1-t)t^2P_2 + t^3P_3 & 0 \le t \le 1 \end{cases} \quad \text{(III.1)}$$

Thus, it is easy to use the number of extracted vectors from a single vehicle to describe such conditions of the road, e.g. even a car is moving with a high speed in average, if the vector is updating frequently, we may define that this road condition is complicated.

Definition 2 (Vector based CR value in road network): Given a road network G and the vector set V of all the objects moving in the network over a period of time t, the vector-based CR value in the road network is defined as the value of each crossroad in road network G calculated according to the crossroad link analysis algorithm. The *CR* value is used to represent the probability of a crossroad being passed. Note that it is only affected by the structure of the graph, so that, in a random situation, the *CR* value indicates the congestion probability of the crossroad. This is why the path planning algorithm requires not only the distance, but also consider the *CR* value in the path planning.

Inspired from linking analysis approaches, the crossroad link analysis algorithm is proposed to calculate the *CR* values of crossroads in the road network. In the algorithm, the webpages correspond to the crossroads in the road network and the road segments between crossroads are considered as the hyperlinks between webpages. As shown in Figure 2, the crossroads in the road network are intercepted and modeled as a directed graph.

The transition matrix of the graph in Figure 2(b) is described as follows.

$$M_1 = \begin{bmatrix} 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & \frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{3} & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{3} & 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \end{bmatrix}$$

According to the feature, the initial value of each node is $\frac{1}{n}$, the initial probability distribution vector V_0 is shown as follows.

$$V_0 = \begin{bmatrix} \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} \end{bmatrix}^T$$

By using Equation II.1, we can calculate V_1. The calculation process is described as follows.

$$V_1 = 0.85 * M * V_0 + 0.15 * V_0 = \begin{bmatrix} 0.122619 \\ 0.183333 \\ 0.122619 \\ 0.122619 \\ 0.183333 \\ 0.122619 \\ 0.142857 \end{bmatrix},$$

The iteration results are shown as below.

$$\begin{bmatrix}0.142857\\0.142857\\0.142857\\0.142857\\0.142857\\0.142857\\0.142857\end{bmatrix}\begin{bmatrix}0.122619\\0.183333\\0.122619\\0.122619\\0.183333\\0.122619\\0.142857\end{bmatrix}\begin{bmatrix}0.134087\\0.177599\\0.125486\\0.125486\\0.177599\\0.134087\\0.125655\end{bmatrix}\ldots\begin{bmatrix}0.127946\\0.180670\\0.126293\\0.126293\\0.180670\\0.127946\\0.130183\end{bmatrix}.$$

By extracting road segment vectors from the trajectory data of vehicles, the number of vectors in each road segment can be obtained. The number of extracted vectors can reflect the traffic congestion condition of the crossroad. The worse the traffic congestion is, the more vectors can be extracted.

Therefore, we introduce the number of vectors (i.e. the traffic congestion condition) to the crossroad link analysis algorithm (i.e. the *CR* value). According to the time variability of road traffic flow, the numbers of vectors that are extracted in the same road segment in different time periods are not the same, we use the congestion coefficient u ($u = 0, 1, 2, 3, 4$) to represent different traffic congestion conditions. $u = 0$ indicates the traffic congestion is the weakest. With the increase of u, the traffic congestion is more and more serious. $u = 4$ indicates the most serious traffic congestion. For the road network in Figure 2(a) as the example, in the case of $u = 2$, if the numbers of extracted vectors in the road segments of *AB*, *BC*, *CD*, *DE*, *BE*, *EF*, *FG* and *AG* are 2, 2, 4, 3, 3, 2, 1 and 3, respectively. After adding the numbers of vectors to the road network graph in Figure 2b, the weighted road network graph is illustrated in Figure 3.

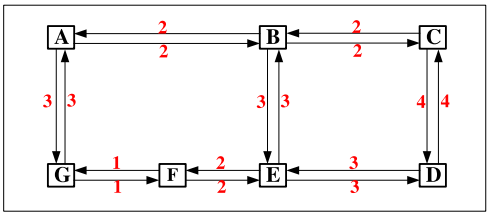


FIGURE 3. The weighted road network graph.

The transition matrix of the weighted road network graph in Figure 3 is described as follows.

$$M_2 = \begin{bmatrix}0 & \frac{2}{7} & 0 & 0 & 0 & 0 & \frac{3}{4}\\ \frac{2}{5} & 0 & \frac{1}{3} & 0 & \frac{3}{8} & 0 & 0\\ 0 & \frac{2}{7} & 0 & \frac{4}{7} & 0 & 0 & 0\\ 0 & 0 & \frac{2}{3} & 0 & \frac{3}{8} & 0 & 0\\ 0 & \frac{3}{7} & 0 & \frac{3}{7} & 0 & \frac{2}{3} & 0\\ 0 & 0 & 0 & 0 & \frac{1}{4} & 0 & \frac{1}{4}\\ \frac{3}{5} & 0 & 0 & 0 & 0 & \frac{1}{3} & 0\end{bmatrix},$$

TABLE 1. The Map process of the road network graph.

Node	Send
A	(B, CR(A)* $\frac{2}{5}$), (G, CR(A)* $\frac{3}{5}$)
B	(A, CR(B)* $\frac{2}{7}$), (C, CR(B)* $\frac{2}{7}$),(E,CR(B)* $\frac{3}{7}$)
C	(B, CR(C)* $\frac{1}{3}$), (D, CR(B)* $\frac{2}{3}$)
D	(C, CR(D)* $\frac{4}{7}$), (E, CR(D)* $\frac{3}{7}$)
E	(B, CR(E)* $\frac{3}{8}$), (D, CR(E)* $\frac{3}{8}$),(F,CR(E)* $\frac{1}{4}$)
F	(E, CR(F)* $\frac{2}{3}$), (G, CR(F)* $\frac{1}{3}$)
G	(A, CR(G)* $\frac{3}{4}$), (F, CR(G)* $\frac{1}{4}$)

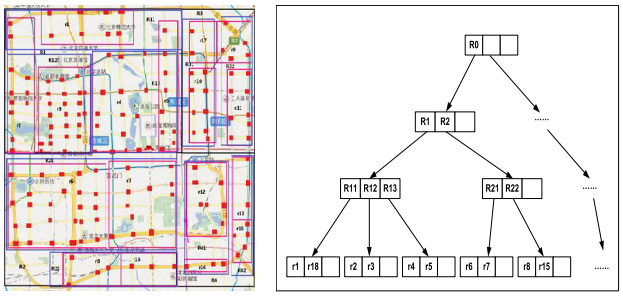
After comparing the two matrices M_1 and M_2, we can find that the order of the *CR* value of each node remains the same, but some nodes have changed. For the nodes *A* and *C* as an example, the *CR* value of the node *A* is greater than the *CR* value of the node *C* in the conventional calculation value. After the vector segmentation, the *CR* value of the node *A* is less than the *CR* value of the node *C*. From this example, it can be seen that the use of the road segment vector is necessary. For the fixed road network, the *CR* value of each crossroad is static and changes as the structure changing of the road network. Therefore, the *CR* values for a fixed road network only requires to be calculated once, which saves the calculation cost compared with the real-time *CR* values. The static *CR* values are calculated based on the structure of road network, where the *CR* value of a crossroad is related to the *CR* values of its linked crossroads and the weights of the linked road segments. To a certain extent, it can reflect the ''crowding'' level of the crossroad, that is, the higher the *CR* value is, the more likely to be congestion.

Definition 3 (Map-reduce framework to calculate the CR value): Given a transition matrix, the Map/Reduce framework to calculate the *CR* value is defined as the process by using the Map/Reduce framework to calculate the *CR* values of all nodes.

Map stage: In the Map process, each node transmits a proportion of its *CR* value to its linked nodes. If the node j transmits its *CR* value to the node i, the transmitted *CR* value is $\Delta CR(i) = CR(j) * \frac{m}{n}$, where m is the number of out links of the node j, which point to the node i and m is the number of all out links of the node j. For the road network graph in Figure 3, the Map process is shown in Table 1.

Reduce stage: Reduce process collects the *CR* values of each node, accumulates and calculates by weights. $CR(j) = a * (CR(1) + CR(2) + \ldots + CR(m)) + (1 - a) * \frac{1}{n}$, where m is the number of nodes that have links with the node j, n is the number of all nodes.

Definition 4 (The real-time updating of the CR value): In the calculation of the *CR* value, although we can use the Map/Reduce framework to speed up the calculation, it will take a long time to calculate the *CR* values for all crossroads in a large-scale road network. Even if some parts of the road network change, it will take a long time to update the *CR* values for the entire road network. To solve this problem, we introduce the *R-Tree* structure. First, the entire work network is divided into subareas and the *CR* values of



(a) *The subareas in a road network map* (b) *The R-Tree structure*

FIGURE 4. The *R-Tree* structure for the *CR* value updating.

crossroad is stored with the index of the *R-Tree*. Thus, given a road network, we firstly need to divide the road network into subareas and establish the *R-Tree* index. The *CR* values of crossroads are stored in the *R-Tree* nodes. Then, the real-time *CR* value of crossroads can be dynamically calculated and updated. The road network of Beijing and their subareas are illustrated in Figure 4(a). As shown in Figure 4(b), we can obtain the corresponding *R-Tree* structure.

In the road network, the state (i.e. the number of extracted vectors) change of road segments will only affect their adjacent crossroads and have no effect on the entire road network. If the road state in a subarea changes, we need to recalculate the *CR* values of crossroads in the subarea.

Based on the established *R-Tree* index, we can easily find, recalculate and update the *CR* values of crossroads in the subarea when the road state changes. For example, in Figure 4(b), if the road state of the node r_7 changes, we only need to recalculate the *CR* values of crossroads in subarea r_7. In this way, the congestion coefficient can be updated in time through updating real-time data and matching. If the congestion coefficient exceeds the threshold, it will be updated timely and adjust the navigation path by the *R-Tree* index, to make the path planning more accurate, efficient and effective.

IV. THE *CR* BASED *A-STAR* ALGORITHM

A-Star is one of the most effective heuristic search algorithms to find the shortest path in static network. It is also an effective approach to solve search problems. The core of *A-Star* algorithm can be described as $f(n) = g(n) + h(n)$, where $f(n)$ is the estimation function to calculate the estimated distance of the path from the start node to the end node by passing the node n, $g(n)$ represents the actual distance of the path from the start node to the node n and $h(n)$ is the heuristic function to calculate the estimated distance of the path from the node n to the end node. The closer the estimated distance is to the actual distance, the faster the heuristic search is.

For instance, we can take the Manhattan distance between two nodes as the heuristic function for a single road network, $h(n) = abs(x_d - x_n) + abs(y_d - y_n)$. When $g(n)$ is constant, the estimation function $f(n)$ will be restricted by the heuristic function $h(n)$. If the node is close to the end node, the value of $h(n)$ is small and the value of $f(n)$ is relatively small, to ensure the search direction of the shortest path is the end node.

In current *A-Star* based path planning methods, the heuristic function $h(n)$ is only calculated from the actual distance (i.e. $h(n) = F(dis)$). In this paper, the idea of adding the *CR* value to the *A-Star* algorithm is that the estimated distance of $h(n)$ is calculated by considering the *CR* value and the actual distance. Therefore, the heuristic function $h(n)$ is described by the formula: $h(n) = F(CR, dis)$, then we only need to consider the form of $F(CR, dis)$.

Based on our experiences, we assume that the form of the function is:

$$F(CR, dis) = k * CR + dis \tag{IV.1}$$

For the experimental purpose, the value of k should satisfy the following two requirements:

- The value of k can change in a certain range. We can find a suitable value of k by adjusting the weights of *CR* and *dis* to achieve different path planning results.
- The value of k can enlarge (or reduce) the final value of $F(CR, dis)$ to a certain extent so that the heuristic function $h(n)$ of the *A-Star* algorithm is still valid (the heuristic function is the Euclidean distance). Otherwise the efficiency of the algorithm will be greatly reduced (becoming the breadth-first or depth-first search) and the heuristic function $h(x)$ should be modified.

From the *CR* value and the actual distance *dis* between coordinates (i.e. latitude and longitude) in real road network data, it can be found that the difference between *CR* and *dis* is great, so Equation IV.1 is not suitable.

The formula is further improved by introducing another parameter c, which can normalize *CR* and *dis* to the same degree. In addition, the value of k is set between 0 and 1. By doing so, k can be used to adjust the weights of two different path planning objects and it satisfies the above two requirements.

After aforementioned modification, the formula is described as follows.

$$F(CR, dis) = c * k * CR + (1 - k) * dis \tag{IV.2}$$

Then, the value of the constant c should be decided. In a simple way, the value of c is the ratio of the differences between the maximum and minimum of *CR* and *dis*, which is:

$$c = \frac{max(dis) - min(dis)}{max(CR) - min(CR)} \tag{IV.3}$$

Equation IV.3 seems reasonable, but if we carefully consider the following two cases, it cannot accurately obtain c.

1) The maximum or minimum *CR* is much more or less than the average of *CR*.
2) The maximum or minimum *dis* is much more or less than the average of *dis*.

If there is a *CR* that is much more or less than the average of *CR*, c will be smaller than its true value. Through the theoretical estimation, the average *CR* should be around 583.

However, from the query of the *CR* value, we find that the largest *CR* is up to 3844, which is caused by a number of continuous unidirectional edges and there are abnormally small *CR* values as well.

The idea of Equation IV.2 is that by multiplying the *CR* value by a constant c, we can get an estimated value of *dis*. However, if the maximum or minimum *CR* is too large (or too small) compared with the average of *CR*, c and the estimated value of *dis* will become smaller, which leads to the heuristic function $h(n)$ inaccurate and affects the efficiency of the algorithm. In order to make the estimated value of *dis* more accurate, we modify Equation IV.2.

$$c = \frac{\overline{x}_{dis}}{\overline{x}_{CR}}, \tag{IV.4}$$

where $\overline{x}_{dis} = \frac{\sum x_{dis}}{n}$ and $\overline{CR} = \frac{100,000,000}{n}$, n is the number of links (edges). The expected value of c can be verified, if after adding c, the estimated value of *dis* is expected to be equal to the expected value of *dis*.

Proposition 5: We declare that $E(F(CR, dis)) = \overline{x}_{dis}$.

Proof: Calculate the expected value of Equation IV.2

$$E(F(CR, dis)) = E(c * k * x_{CR} + (1-k) * x_{dis}); \tag{IV.5}$$

Separate x_{CR} and x_{dis}:

$$E(F(CR, dis)) = E(c * k * x_{CR}) + E((1-k) * x_{dis}); \tag{IV.6}$$

Extract constants:

$$E(F(CR, dis)) = c * k * E(x_{CR}) + (1-k) * E(x_{dis}); \tag{IV.7}$$

Substitute according to Equation IV.4:

$$E(F(CR, dis)) = \frac{\overline{x}_{dis}}{\overline{x}_{CR}} * k * \overline{x}_{CR} + (1-k) * \overline{x}_{dis} \tag{IV.8}$$

Simplify the equation:

$$E(F(CR, dis)) = \overline{x}_{dis} \tag{IV.9}$$

The constant c in Equation IV.1 seems to be nearly perfect, but if we consider the specific data distribution of two samples, such as if the coefficient of variation (i.e. the ratio between the standard deviation and the average) of *CR* is big and the coefficient of variation of *dis* is small or vice versa, the estimated value will be incorrect. According to our idea, the normalized *CR* value is multiplied by the proportional coefficient k to revert to *dis*. After many attempts and calculations, the final $F(CR, dis)$ is described as follows.

$$F(CR, dis) = (\sqrt{k} * (\frac{x_{CR} - \overline{x}_{CR}}{\sigma_{CR}}) + \sqrt{(1-k)} * (\frac{x_{dis} - \overline{x}_{dis}}{\sigma_{dis}})) * \sigma_{dis} + \overline{x}_{dis}, \tag{IV.10}$$

where σ_{CR} and σ_{dis} are the standard deviations of *CR* and *dis*, respectively. The mathematical proof is performed to verify the accuracy of Equation IV.10, while its expected values can be generated as follows.

Simplify the formula:

$$F(CR, dis) = \sqrt{k} * \frac{\sigma_{dis}}{\sigma_{CR}} * (x_{CR} - \overline{x}_{CR}) + \sqrt{(1-k)} * (x_{dis} - \overline{x}_{dis}) + \overline{x}_{dis} \tag{IV.11}$$

Calculate the expected value:

$$E(F(CR, dis)) = E(\sqrt{k} * \frac{\sigma_{dis}}{\sigma_{CR}} * (x_{CR} - \overline{x}_{CR}) + \sqrt{(1-k)} * (x_{dis} - \overline{x}_{dis}) + \overline{x}_{dis}) \tag{IV.12}$$

Separate x_{CR} and x_{dis} and extract constants:

$$E(F(CR, dis)) = \sqrt{k} * \frac{\sigma_{dis}}{\sigma_{CR}} * E(x_{CR} - \overline{x}_{CR}) + \sqrt{(1-k)} * E(x_{dis} - \overline{x}_{dis}) + E(\overline{x}_{dis}) \tag{IV.13}$$

Since $E(x_{CR} - \overline{x}_{CR})$ and $E(x_{dis} - \overline{x}_{dis})$ are 0, the formula is simplified as follows.

$$E(F(CR, dis)) = E(\overline{x}_{dis}) = \overline{x}_{dis}. \tag{IV.14}$$

□

Proposition 6: $Var(F(CR, dis)) = \sigma_{dis}^2$.

Proof: Simplify the formula of IV.10:

$$F(CR, dis) = \sqrt{k} * \sigma_{dis} * (\frac{x_{CR} - \overline{x}_{CR}}{\sigma_{CR}}) + \sqrt{(1-k)} * (x_{dis} - \overline{x}_{dis}) + \overline{x}_{dis} \tag{IV.15}$$

Calculate the variance:

$$Var(F(CR, dis)) = Var(\sqrt{k} * \sigma_{dis} * (\frac{x_{CR} - \overline{x}_{CR}}{\sigma_{CR}}) + \sqrt{(1-k)} * (x_{dis} - \overline{x}_{dis}) + \overline{x}_{dis}) \tag{IV.16}$$

Simplify the formula:

$$Var(F(CR, dis)) = Var(\sqrt{k} * \sigma_{dis} * (\frac{x_{CR} - \overline{x}_{CR}}{\sigma_{CR}})) + Var(\sqrt{(1-k)} * (x_{dis} - \overline{x}_{dis})) + Var(\overline{x}_{dis}) \tag{IV.17}$$

The calculation is made and the constant is obtained:

$$Var(F(CR, dis)) = k * \sigma_{dis}^2 * Var(\frac{x_{CR} - \overline{x}_{CR}}{\sigma_{CR}}) + (1-k) * Var(x_{dis} - \overline{x}_{dis}) + Var(\overline{x}_{dis}) \tag{IV.18}$$

Based on the properties of variance, we have $Var(\overline{x}_{dis}) = 0$, $Var(\frac{x_{CR} - \overline{x}_{CR}}{\sigma_{CR}}) = 1$ and $Var(x_{dis} - \overline{x}_{dis}) = \sigma_{dis}^2$, so the equation is equal to:

$$Var(F(CR, dis)) = k * \sigma_{dis}^2 + (1-k) * \sigma_{dis}^2 = \sigma_{dis}^2. \tag{IV.19}$$

□

V. ALGORITHM DESIGN

We can obtain vectors by using the sampling data of a certain vehicle trajectory from a road segment according to Definition 1 and our previous research [4]. The number of vectors reflects the traffic congestion condition of the road to some extent, e.g. if the road is relatively congested and the vehicle's driving state frequently changes, more vectors will be extracted. Based on this concept, we introduce the crossroad link analysis algorithm considering the road segment vectors, which will influence the *CR* values of congested crossroads. In this way, we will select the path with relatively smaller *CR* values in the path planning method.

In real roads, the traffic conditions will be more crowded in the morning and evening of weekdays's rush hours. We obtain the largest number of the road segment vectors at this period of time and set the congestion coefficient as $u = 4$. On the contrary, it will be relatively smooth in the night time, so we extract the least number of vectors and set the congestion coefficient as $u = 0$. We set the corresponding congestion coefficient according to the number of extracted vectors when the state of road condition changes.

According to the theory, the *CR* value represents the probability distribution and the sum of them should be 1. However, it is unavoidable to omit the decimal part during the calculation. Therefore, we enlarge the *CR* value by 100 million times to make the result more accurate. In order to achieve the real-time updating of the *CR* value, the entire road map is divided into many subareas, and the *R-Tree* structure is used to store and index the *CR* values. When the *CR* value change of the roads in a subarea exceeds the default *threshold* (according to the parameters set for the actual roads), we only need to update the *CR* values of the corresponding roads in the subarea. Algorithm 1 and Algorithm 2 describe the process of real-time update and calculation of the *CR* value, respectively.

Algorithm 1 *CR*-Update (T, id, $Number$, θ)

Input: T: *R-Tree* (the *R-Tree* index); id: int (the road ID); $Number$: int (the number of vectors extracted from roads); θ: double; (the threshold)
Output: T: *R-Tree* (the *R-Tree* index)

1 *int Center* $\leftarrow$| *Number* $-$ *Agraph*[*id*] |;
2 **if** *(Center-Agraph[id]*θ >0)* **then**
3 $\quad$ *Agraph*[*id*] $\leftarrow$ *Number*;
4 $\quad$ *Agr* $\leftarrow$ *Search* $-$ *Agraph*(T, *id*);
5 $\quad$ *intinit*[] $\leftarrow$ *CR* $\cdot$ *Calculate*(*Agr*, *e*);
6 $\quad$ *Update*(T, *init*[]);
7 **end**
8 *Return*(T);

In Algorithm 1, the road segment vectors are extracted, when the state of roads in a subarea changes (line 1). If the number of extracted vectors exceeds the threshold θ, the *CR* values in the corresponding subarea need to be updated (line 2). First, we update the state of the roads (line 3). Then, we find the corresponding roads in the road network based on the *R-Tree* index (line 4). After that, we recalculate the *CR* values of the crossroads in the corresponding subarea (line 5). Finally, we update the *CR* values and store them in the *R-Tree* structure (line 6).

Algorithm 2 *CR*-Calculate (*Agraph*, *e*)

Input: *Agraph*: The adjacency table (the road network); *e*: double(the threshold)
Output: *init*[]: double (the *CR* value)

1 *double init*[*s*] $\leftarrow 100,000,000/n$;
2 **while** *surplus* $>$ *e* **do**
3 $\quad$ *Link*[*d*] $\leftarrow 0$;
4 $\quad$ **while** *!Links.eof* () **do**
5 $\quad\quad$ *Agraph.read*(*original*, *n*, $link_1$, $link_2$, ..., $link_n$);
6 $\quad\quad$ *Links.read*(*original*, *n*, $link_1$, $link_2$, ..., $link_n$);
7 $\quad\quad$ *Sum* $\leftarrow$ *Agraph*[$link_1$] + *Agraph*[$link_2$] + ... + *Agraph*[$link_n$];
8 $\quad\quad$ **for** $j = 1...n$ **do**
9 $\quad\quad\quad$ *Link*[$link_j$] $\leftarrow$ *Link*[$link_j$]+*init*[*original*] $*$ *Argaph*[$link_j$]/*Sum*;
10 $\quad\quad$ **end**
11 $\quad$ **end**
12 $\quad$ *Link*[*d*] $\leftarrow c * Link[d]+(1-c) * 100000000/N$;
13 $\quad$ *Surplus* $\leftarrow$| *init* $-$ *link* |;
14 $\quad$ *Init* $\leftarrow$ *Link*;
15 **end**
16 *Return*(*Init*);

In Algorithm 2, we use the array to initialize the *CR* value of each node (i.e. crossroad) (line 1), and enlarge the *CR* value by 100 million times to reduce errors. We use an adjacency table to store the basic information of the road network and store the relevant information of the node out links in the link list of each node. The end condition of the outer loop is that the change of *CR* values of all nodes are less than the threshold *e* (line 2). The inner loop is to ensure that the *CR* values of all nodes can be calculated (line 4). In the Map process, first, the weights of out links of each node is read out (lines 5 and 6). Then, the sum of weights of all out links is calculated (line 7). After that, the corresponding out links the each node are calculated and resent according to the weight proportion of the *CR* value (lines 8 and 9). In the Reduce process, first, the *CR* values of the node with the same ID are added up and the *CR* value of each node is updated (line 12). The change of *CR* values of all nodes in this loop are calculated (line 13). If the change is less than the threshold *e*, the algorithm returns all *CR* values of the road network (line 16).

The *A-Star* algorithm is expressed as: $f(n) = g(n) + h(n)$, where $h(n)$ is the heuristic function to estimate the cost of path. In this paper, we use the Euclidean distance as the heuristic function of the *A-Star* algorithm. In Section IV, the *CR* value is mixed with the distance to substitute to the *A-Star* algorithm to make the path planning more reasonable.

Algorithm 3 *CR* Based *A-Star* Path Planning (*Agraph*, *Start*, *End*)

Input: *Agraph*: The adjacency table (the road network); *Start*: int (The start node); *End*: int (the end node)
Output: *PATH*: path;

```
1  Open ← {start};
2  Close ← {};
3  while Open! = NUll do
4      n ← GetNode(Open);
5      DelectNode(Open, n);
6      if n = End then
7          Return(PATH);
8      end
9      for Y ∈ Agraph.read(n, m, link_1, link_2, ..., link_n)
       do
10         if Y ∉ Close AND Y ∉ Open then
11             V ← Get − determinedvalue(Y);
12             Inserttable(Open, Y);
13         end
14         if Y ∈ Open then
15             if V < Get − determinedvalue(Open, Y)
               then
16                 Update − determinedvalue(Open);
17             end
18         end
19         else
20             if V < Get − determindvalue(Close, Y)
               then
21                 Update − determinedvalue(Close);
22                 DelectNode(Close, Y);
23                 Inserttable(Open, Y);
24             end
25         end
26         Inserttable(n, Close);
27         Node − Sort(Open);
28     end
29 end
```

Algorithm 3 introduces how to achieve the *CR* based *A-Star* algorithm.

At the beginning of Algorithm 3, two tables are created, where the *Open* table holds all the nodes that have not been visited and the *Close* table records the visited nodes (lines 1 and 2). If the *Open* table is not empty, the node n with the minimum $f(n)$ is taken from the *Open* table (line 4) and the node n is deleted from the *Open* table (lines 4 and 5). If the node n is the end node *End*, the algorithm returns the planned path (lines 6 and 7). Otherwise, for each child node Y of the node n (line 9), if the node Y is neither in the *Open* table nor in the *Close* table (line 10), $f(Y)$ is calculated (line 11) and the node Y is inserted into the *Open* table as the successor of the node n (line 12); If the node Y is in the *Open* table (line 14) and the new calculated $f(Y)$ is less than the $f(Y)$ in the *Open* table (line 15), $f(n)$ in the *Open* table is updated (line 16); If the node Y is in the *Close* table (line 19) and the new calculated $f(Y)$ in is greater than the $f(Y)$ in the *Close* table (line 20), $f(n)$ in the *Close* table is updated (line 21), the node Y is deleted from the *Close* table (line 22) and the node Y is inserted into the *Open* table (line 23). Finally, the node n is inserted into the *Close* table (line 26) and the nodes in the *Open* table is sorted by their $f(n)$ values (line 27).

VI. EXPERIMENTAL RESULTS AND EVALUATIONS

A. EVALUATION METHOD AND EXPERIMENTAL SETTINGS

1) EVALUATION METHOD

Due to the following three main reasons, our experimental evaluation method in this paper is not verified against the existing path planning methods or algorithms in relevant research articles:

1) Inconsistent dataset, the experiment uses Beijing road network data and Beijing vehicle trajectory data, while different evaluation dataset have different properties, restrictions, i.e. sampling frequencies.
2) Inconsistent methods to generate or represent road conditions, prior studies usually use the average speed or traffic flow volume to describe the current state of the road, while in this paper, the vector quantity of real-time road conditions was used to depict the current road conditions.
3) Online application involved multiple advanced methodologies that can provide path planning results based on real-time road conditions, which is closer to our approach. Thus, we selected most used two of them (i.e. Baidu and Amap) to evaluate our proposed path planning method.

2) EXPERIMENTAL SETTINGS

In this paper, the adjacency table is build to store real road network data of Beijing. According to the trajectory data of vehicles, the road segment vectors are extracted. Since our path planning approach is designed for the dynamic road network, the path planning approaches for the static road network are not compared in our experiments. In experiments, we obtain the road network data of Beijing and sort them with different traffic congestion conditions, where the congestion coefficients are $u = 0$ and $u = 4$, and calculate the corresponding *CR* values of the road network by using the two datasets. First, we comparatively investigate the efficiency of the proposed algorithms. Then, we experimentally determine the nature of the parameter k. Finally, we verify the effectiveness of the path planning algorithm.

On the basis of the road network data of Beijing, the road network graph is established and the adjacency table is used to store them. According to the trajectory data of vehicles, the road segment vectors are extracted and the number of extracted vectors is obtained. In addition, the traffic conditions are divided according to different traffic congestion conditions and represented by the congestion coefficients u. Based on the two datasets, we conduct three experiments:

1) The efficiency of the proposed method is comparatively investigated; 2) The best value of the parameter k is experimentally determined; 3) The effectiveness of the proposed path planning method is evaluated.

Moreover, we compared the path planning results of our proposed method with two navigation systems: *Baidu* map[1] and *Amap*,[2] with the congestion coefficients u=0 and u=4. The experiments are conducted through Java on a PC with AMD Phenom 9650 2.3GHz (Quadcore) processor and 4GB of memory.

B. THE EFFICIENCY OF THE PROPOSED METHOD

As a real-time system, the efficiency of the path planning method is very important. Therefore, we evaluate the efficiency of the proposed method in this subsection, which is composed of two parts: the efficiency of the updating of the *CR* value and the value of the parameter k. By employing the *R-Tree* structure to store and index the *CR* values of the road network, the proposed method can efficiently find and update the *CR* values of corresponding crossroads, when the state of roads changes. Because the execution time of proposed method varies with respect of different values of k, it is important to determine the best value of k that enables the method to have the shortest execution time.

1) UPDATING *CR* VALUES

There are more than 172, 000 crossroads in the road network of Beijing. Whenever there is a change of the road state, it will take a lot of time and resources to recalculate and update the *CR* values of all crossroads. As shown in Figure 5, when T_1 is congested, it will affect only a few segments of roads around it, but T_2 is not affected a lot. Therefore, in this paper, we divide the road network into subareas and the *CR* values of crossroads of the whole road network is stored and indexed by the *R-Tree* structure.

With on the road network of Beijing, we delineate three rectangular areas, namely R_1, R_2 and R_3 as drawn in Figure 5. Based on the diagonal vertex coordinates of the three rectangles (i.e. P_1 to P_6 in Figure 5), we can obtain the number of crossroads in the each rectangular area through the databased query. There are 3, 457 crossroads in R_1, 18, 188 crossroads in R_2, 171, 504 crossroads in R_3, which is shown in Table 2. The diagonal vertex coordinates of the three rectangles and the experimental results are shown accordingly.

In the experiment, the system takes 772ms (28 iterations), 2, 547ms (30 iterations) and 18, 623ms (31 iterations) to update the *CR* values of crossroads in R_1, R_2 and R_3, respectively, which are shown in Table 2. From the experiment, it can be seen that with the increase of the region size, the number of crossroads and the time for updating the *CR* values increase exponentially. Therefore, if the *CR* values of the whole road network is updated, it will take a long time and updating *CR* values of crossroads in a subarea can greatly save a lot of time and resources. In the proposed method, the *R-Tree* structure is used to store and index the *CR* values of crossroads. When the state of roads changes, the *CR* values need to be updated, we only need to find the corresponding crossroads with the minimum subareas and update the *CR* value of them. By doing so, the efficiency of the proposed method can be greatly improved.

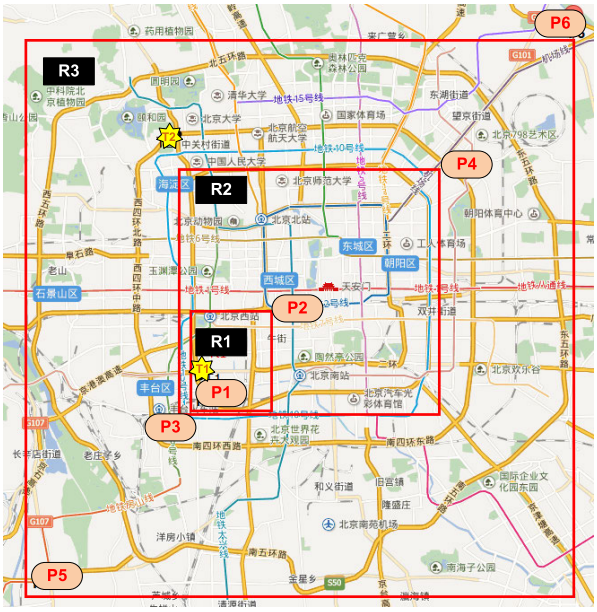


FIGURE 5. Three nodes in different levels of R-Tree in the road network of Beijing.

TABLE 2. The coordinates and the experimental results.

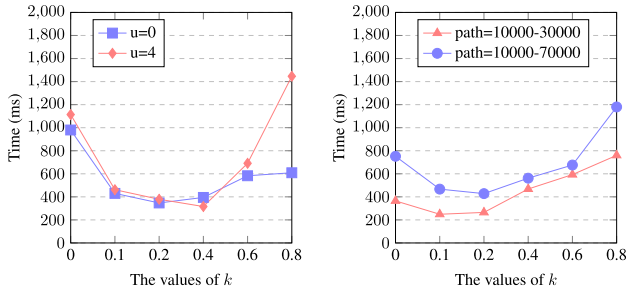
Region	Point	Coordinate		*CR*-Num	Time(ms)
		lag	lng		
R_1	P_1	39.866	116.341	3,457	772
	P_2	39.902	116.430		
R_2	P_3	39.847	116.306	18,188	2,547
	P_4	39.967	116.446		
R_3	P_5	39.756	116.204	171,504	18,623
	P_6	40.029	116.548		

2) THE DISCUSSION OF THE VALUE OF *k*

In the experiment, through adjusting the values of the parameter k (i.e. 0, 0.2, 0.4, 0.6 and 0.8), we tend to find the best value of k that enables the method to have the shortest execution time. In order to analyze the influence of k, the experiment is divided into two groups. In the first group, the navigation is conducted to find the same path (i.e. the same start and end points) with different congestion coefficients u. In the second group, the navigation is conducted to find different paths (i.e. different start and end points) with the same congestion coefficient.

[1] *Baidu* map is a network of Baidu Inc to provide search services, covering nearly 400 cities in China, thousands of districts and counties. The website of *Baidu Map* navigation is: http://map.baidu.com/.

[2] *Amap* is China's leading digital map content, navigation and location-based services solutions provider. The website of *Amap* navigation is: http://ditu.amap.com/.



(a) *Different values of u* (b) *Different paths*

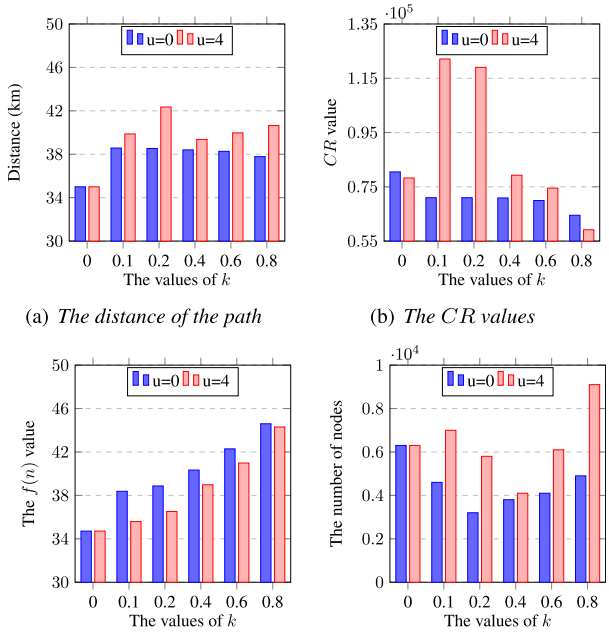
FIGURE 6. **The execution time under different values of *k*.**

Specifically, in the first group, the navigation is conducted to find the path with the start point at $vid = 10000$, the end point at $vid = 50000$ under the different congestion coefficients $u = 0$ and $u = 4$. In the second group, the navigation is conducted to find the path with the start point at $vid = 10000$, the end point at $vid = 30000$ and the path with the start point at $vid = 10000$, the end point at $vid = 70000$ under the same congestion coefficient $u = 4$. The execution time (in terms of *ms*) of the proposed method is compared under different values of k.

According to the experimental results in Figure 6, we can find the tendency of the execution time under different values of k drop at first and then increase. In the first group (the same path and different congestion coefficient), the best efficiency of the navigation method is obtained at $k = 0.2$ when $u = 0$ and $k = 0.4$ when $u = 4$. In the second group (the different paths and the same congestion coefficient), the best efficiency of the navigation method is obtained at $k = 0.1$ when the path is from $vid = 10000$ to $vid = 30000$ and $k = 0.2$ when the path is from $vid = 10000$ to $vid = 70000$. From this experiment, it can be seen that both the congestion coefficient and the path have the influence on the value determination of k. Through a large number of experiments, it can be found that in most cases, the best efficiency of the navigation method can be obtained, when the value of k is between 0.2 and 0.4. In the determination of the value of k, we consider not only the efficiency of the navigation method, but also the distance, the *CR* value and the estimated distance $f(n)$ of the planned path, etc. Therefore, it is necessary to have a deep analysis on the value of k by including other factors.

C. THE DETERMINATION OF THE VALUE OF k

The influence of the k value on the efficiency of the navigation method is analyzed in the last section. In this section, more experiments are conducted to determine the value of k based on its influence on other factors. The experiment is to use the *CR* based *A-Star* algorithm to plan the path under different values of k, and select the value of k by comparing other factors, including the distance, the *CR* value and the $f(n)$ value of the path and the number of searched nodes during the path planning.



(a) *The distance of the path* (b) *The CR values*

(c) *The estimated distance* $f(n)$ (d) *The number of searched nodes*

FIGURE 7. **The experimental results of other factors comparison.**

In the experiment, the *CR* based *A-Star* algorithm is used to plan the path with the same start and end points (i.e. from $vid = 10000$ to $vid = 50000$) under different congestion coefficients (i.e. $u = 0$ and $u = 4$). The experimental results are illustrated in Figure 7.

According to Figure 7, it can be seen that when $k = 0$, only the distance is considered in the path planning, the algorithm always finds the path with the shortest geographic distance. In this situation, the distance and the $f(n)$ value of the path are the same under two different congest coefficients (i.e. $u = 0$ and $u = 4$). Since the *CR* values are influenced by the congestion coefficient, they are different for the same path under different congestion coefficients.

When the congestion coefficient $u = 0$, with the increase of k, the *CR* values and the distance of the path decrease, while the $f(n)$ value increase. The minimum of the execution time is obtained at $k = 0.2$. During the path planning, the minimum number of searched nodes is obtained at $k = 0.2$ and $k = 0.4$. Based on the execution time and the minimum number of searched nodes, the best value of k is 0.2.

When the congestion coefficient $u = 4$, with the increase of k, the *CR* values decrease and the $f(n)$ values increase. Compared with the situation of $u = 0$, the change of the congestion coefficient does not lead the change of the *CR* values and the $f(n)$ values. The change of the distance of the path is influenced by different congestion coefficient of roads. By considering the execution time and the number of searched nodes, the best value of k is 0.4.

Through the experiment in Figure 7, it can be found that with the increase of k, the *CR* value of the planned path decreases and the $f(n)$ values increase. Therefore, the value of

k can control the proportion of the CR value and the distance dis, the higher the value of k the more influence of the CR value on the path planning.

In the experiments, it can be found that in different path planning, the difference between the best values of k is large. It can be concluded that the best value of k is greatly influenced by the structure of the road network. For example, Table 3 demonstrates the experimental results of the path planning from $vid = 10000$ to $vid = 50000$ under different values of k and Table 4 demonstrates the experimental results of the path planning from $vid = 10000$ to $vid = 70000$ under different values of k.

TABLE 3. The experimental results of the path planning for the Path:10000-50000.

k / Att	0	0.1	0.2	0.4	0.6	0.8
$dis\ (km)$	28.287	34.317	36.295	41.821	44.554	38.220
CR	77,528	93,122	92,303	88,827	77,583	55,253
$weight$	28.287	28.920	29.636	32.072	34.594	37.348
$nod\text{-}num$	3,300	2,000	2,800	4,100	5,100	7,100

TABLE 4. The experimental results of the path planning for the Path:10000-70000.

k / Att	0	0.1	0.2	0.4	0.6	0.8
$dis\ (km)$	35.120	36.835	36.037	37.698	39.366	40.814
CR	77,321	76,121	76,121	71,833	71,857	71,466
$weight$	35.110	36.380	36.037	39.417	41.681	45.297
$nod\text{-}num$	48,000	4,600	4,200	5,100	6,100	8,000

The experimental results in Table 3 and Table 4 can verify the conclusion that in different path planning situations, the best values of k are different. In Table 3, for the path planning from the start point $vid = 10000$ to the end point $vid = 50000$, the most efficient and effective path planning result is obtained at $k = 0.4$. While in Table 3, for the path planning from the start point $vid = 10000$ to the end point $vid = 70000$, the most efficient and effective path planning result is obtained at $k = 0.2$.

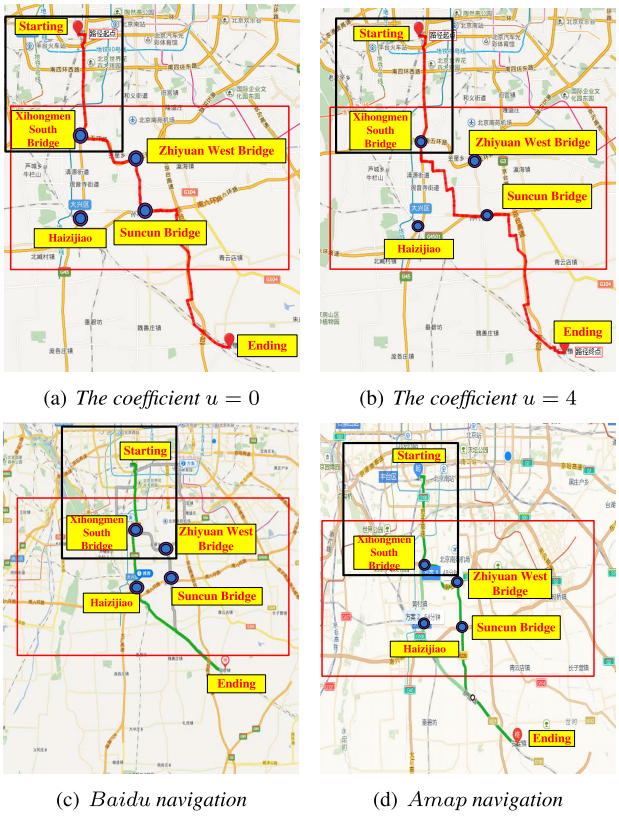
Through the analysis on a large number of experiments, we find that the best value of k varies under the same congestion coefficient and different road networks. Based on a large number of experiments, in most of road networks, the best value of k is between 0.2 and 0.4.

D. THE EFFECTIVENESS OF THE PROPOSED METHOD

In this subsection, given a starting point from city center to a ending point at the city side, we compares the recommended routes with and without congestion. Besides, we compare our approach against two existing navigation application to explored the effectiveness of the proposed method.

1) COMPARISON WITH $u = 0$ AND $u = 4$

Figure 8(a) and Figure 8(b) show the paths planned by the proposed method with $u = 0$ and $u = 4$, respectively. Moreover, we take two different parts in paths in Figure 8(a) and Figure 8(b) as the example to analyze the influence of the congestion coefficient, which are shown in Figure 9.



(a) *The coefficient* $u = 0$ (b) *The coefficient* $u = 4$

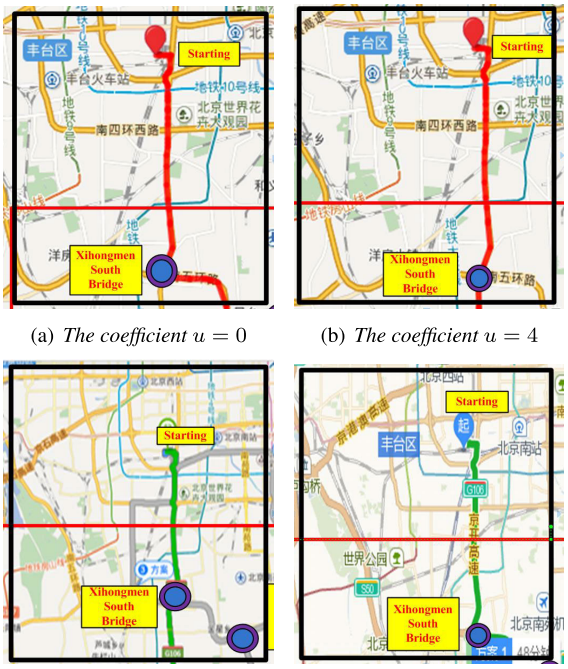
(c) *Baidu navigation* (d) *Amap navigation*

FIGURE 8. The path planning of different navigation methods and systems.

(a) *The coefficient* $u = 0$ (b) *The coefficient* $u = 4$

FIGURE 9. The difference between paths under different congestion coefficient *u*.

Figure 9 shows that under different congestion coefficients (i.e. $u = 0$ and $u = 4$), the paths planned by the proposed navigation method are different, which can be seen in Figure 8(a) and Figure 8(b). As shown in Figure 9(a), when the congestion coefficient $u = 0$, the traffic conditions of most roads are relatively smooth. When the vehicle reaches the *Xihongmen Bridge*, the proposed navigation method chooses the path of the south fifth ring road (i.e. *Path* 1 in Figure 9(a)). When the congestion coefficient $u = 4$, as shown in Figure 9(a), since the south fifth ring road is congested, the proposed method chooses to go straight in a smooth path (i.e. *Path* 2 in Figure 9(a)). In Figure 9(b), if the congestion coefficient u is 0 and the road is smooth, the vehicle is recommended to go along the *Huangxu* road (i.e. *Path* 2



(a) *The coefficient* $u = 0$ (b) *The coefficient* $u = 4$

(c) *Baidu navigation* (d) *Amap navigation*

FIGURE 10. The starting of path planning of different navigation methods and systems.

in Figure 9(a)). When the congestion coefficient $u = 4$ and the traffic condition is congested, the proposed navigation method plans a smooth path (i.e. *Path* 1 in Figure 9(b)).

It is easy to observe that our proposed variable u affect the path planning a lot comparing Figure 8(a) and 8(b), i.e. when we set $u = 0$, our proposed method generate almost the shortest paths in Figure 8(a), while when $u = 4$, the method successfully avoids the *Zhiyuan West Bridge* (with much more complicated traffic condition and easy to get traffic congestions) in Figure 8(b), but trying to keep with a shortest path fashion. This proves our previous congestion coefficient analysis.

2) COMPARISON WITH *Baidu* AND *Amap*

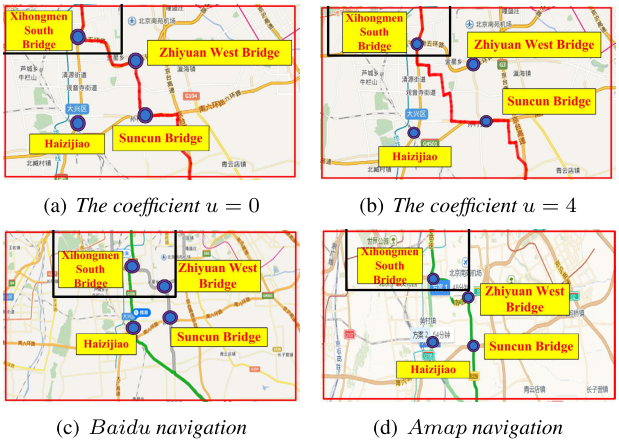
Figure 8(c) shows the path planned by the *Baidu* navigation system, while Figure 8(d) shows the path planned by the *Amap* navigation system. Note that the top 1^{st} recommended path from both two applications are in green colour, while the top 1^{st} recommended plan generally considering both the travel time cost and path distances.

The full length of the planned paths from our proposed method is very similar to that obtained from navigation applications, i.e. about 42.5km in total. Also, some of the road segments seem to be identical with *Baidu* and *Amap* according to Figure 8.

During the path planning, *Baidu* provides three path options, while *Amap* only provides two. In the experiment, the path planning method proposed in this paper is employed to generate the routes under different congestion coefficients. The paths planned by the proposed method with $u = 0$ and $u = 4$, *Baidu* and *Amap* are illustrated in Figure 8.

The navigation routes of maps are shown in Figure 10(a), 10(b),10(c),10(d) which are subgraphs in black boxes of Figure 8(a),8(b),8(c),8(d) respectively. It can be seen from the Figure 10(a),10(b),10(c),10(d) all of the navigation route from the *Starting* to the *Xihongmen South Bridge* is the same. When $u = 0$, i.e. the road is not congested, the navigation route is the closest distance from the *Starting* to the *Xihongmen South Bridge*. When $u = 4$, i.e. the road is congested, although there is congestion at this time, the navigation algorithm still considers that the road from the *Starting* to the *Xihongmen South Bridge* is the best one by calculating the distance and congestion.

By comparing the recommended routes of $u = 0$ (Figure 8(a)), $u = 4$ (Figure 8(b)) with those of *Baidu* (Figure 8(c)) and *Amap* (Figure 8(d)), we can see that the routes suggested by each navigation algorithm are different from *Xihongmen South Bridge*, and there are obvious differences in the navigation routes in the red boxes in Figure 8. Therefore, the experiment focuses on comparing the navigation routes in the red boxes. The navigation routes of maps are shown in Figure 11(a), 11(b),11(c),11(d) which are subgraphs in red boxes of Figure 8(a),8(b),8(c),8(d) respectively



(a) *The coefficient* $u = 0$ (b) *The coefficient* $u = 4$

(c) *Baidu navigation* (d) *Amap navigation*

FIGURE 11. The part of path planning of different navigation methods and systems.

When $u = 0$, i.e. the road is not congested, the navigation route is shown in Figure 11(a). The navigation route is from *Xihongmen South Bridge* to *Zhiyuan West Bridge*, from *Zhiyuan West Bridge* to *Suncun Bridge*, and the navigation route is different from that of *Baidu*, while it is the same as that of *AMap*.

When $u = 4$, i.e. the road is congested, the navigation route is shown in Figure 11(b). Since the route from *Xihongmen South Bridge* to *Zhiyuan West Bridge* and *Zhiyuan West Bridge* to *Suncun Bridge* i.e. the navigation route from $u = 0$ and *Amap* is congested, the route from *Xihongmen Bridge* to *Haizijiao* i.e. the navigation route from *Baidu* is congested. Therefore, at this time, the navigation

algorithm does not choose these two routes, rather it chooses the non-congested road i.e. ladder road in Figure 11(b), so as to avoid the very congested road. Through the comparison of $u = 0$ and $u = 4$ with *Baidu* and *Amap*, it is shown that our path planning method can effectively avoid choosing congested roads.

VII. THE CONCLUSION AND FURTHER WORK

In this paper, we focused on the problem of real-time path planning in urban road network. To overcome the limitation of traffic state representations, we proposed an innovative crossroad link analysis algorithm to calculate the *CR* value of crossroads by considering the bidirectional links and the number of vectors in road segments. Then, the *A-Star* algorithm is employed to achieve the path planning, where both the *CR* value and the Euclidean distance is used in the heuristic function during the search process. In addition, the congestion coefficient is established to describe different traffic congestion conditions of roads. We verify the effectiveness of the proposed method using the actual traffic data of Beijing, and the experimental results show that our proposed method can generate the appropriate path plan in the peak and low dynamic traffic conditions.

As further work, we plan to involve the event detection and data mining techniques into our navigation method, such as the traffic forecast, historical data mining and real demands in social traffic. Meanwhile, in order to further improve efficiency of algorithm, spatial separating methodologies will be employed, such as Voronoi or Skyline techniques.

ACKNOWLEDGMENT

This work was supported by the National Key R&D Program of China (Grants No. 2017YFC0803300), the Beijing Natural Science Foundation (Grant No. 4172004, 4192004), the National Natural Science Foundation of China (Grants No. 61703013, 41971366), Beijing Municipal Education Commission Science and Technology Program (Grants No. KM201810005024, KM201810005023).

REFERENCES

[1] Y. Akhremtsev, P. Sanders, and C. Schulz, "High-quality shared-memory graph partitioning," in *Proc. 24th Int. Conf. Parallel Distrib. Comput.*, Turin, Italy, Aug. 2018, pp. 659–671.

[2] A. Balmin, V. Hristidis, and Y. Papakonstantinou, "Objectrank: Authority-based keyword search in databases," in *Proc. 30th Int. Conf. Very Large Data Bases*, 2004, pp. 564–575.

[3] G. V. Batz and P. Sanders, "Time-dependent route planning with generalized objective functions," in *Proc. 20th Annu. Eur. Symp. Algorithms (ESA)*, Ljubljana, Slovenia, Sep. 2012, pp. 169–180.

[4] Z. Cai, F. Ren, J. Chen, and Z. Ding, "Vector-based trajectory storage and query for intelligent transport system," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 5, pp. 1508–1519, May 2018.

[5] D. Casasent, "A novel lidar-driven two-level approach for real-time unmanned ground vehicle navigation and map building," *Proc. SPIE*, vol. 9025, no. 2, p. 236, 2014.

[6] P. Chen, X. Zhang, X. Chen, and M. Liu, "Path planning strategy for vehicle navigation based on user habits," *Appl. Sci.*, vol. 8, no. 3, p. 407, Mar. 2018.

[7] P. Faizian, M. A. Mollah, X. Yuan, S. Pakin, and M. Lang, "Random regular graph and generalized de bruijn graph with K-shortest path routing," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2016, pp. 103–112.

[8] G. Fakas, Z. Cai, and N. Mamoulis, "Diverse and proportional size-l object summaries for keyword search," in *Proc. ACM SIGMOD Int. Conf.*, 2015, pp. 363–375.

[9] G. J. Fakas, Z. Cai, and N. Mamoulis, "Size-l object summaries for relational keyword search," *VLDB Endowment*, vol. 5, no. 3, pp. 229–240, 2011.

[10] Y. Fan, Q. Wang, D. Lu, and F. Jiang, "An improved dijkstra algorithm used on vehicle optimization route planning," in *Proc. 2nd Int. Conf. Comput. Eng. Technol.*, 2010, pp. 693–696.

[11] E. Fresk, G. Nikolakopoulos, and T. Gustafsson, "A generalized reduced-complexity inertial navigation system for unmanned aerial vehicles," *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 1, pp. 192–207, Jan. 2017.

[12] V. Hristidis, H. Hwang, and Y. Papakonstantinou, "Authority-based keyword search in databases," *TODSACM Trans. Database Syst.*, vol. 33, no. 1, pp. 1–40, Mar. 2008.

[13] H. Huang, D. Zhu, and F. Ding, "Dynamic task assignment and path planning for multi-AUV system in variable ocean current environment," *J. Intell. Robot. Syst.*, vol. 74, nos. 3–4, pp. 999–1012, Jun. 2014.

[14] A. Idri, M. Oukarfi, A. Boulmakoul, K. Zeitouni, and A. Masri, "A distributed approach for shortest path algorithm in dynamic multimodal transportation networks," *Transp. Res. Procedia*, vol. 27, pp. 294–300, 2017.

[15] W. Jiang, Y. Li, and C. Rizos, "A multisensor navigation system based on an adaptive fault-tolerant GOF algorithm," *IEEE Trans. Intell. Transport. Syst.*, vol. 18, no. 1, pp. 103–113, Jan. 2017.

[16] G. Jing, Y. Wu, X. Zhang, Z. Le, C. Wei, Z. Cao, Z. Lu, H. Guo, G. Jing, and Y. Wu, "Finding the 'Faster' path in vehicle routing," *IET Intell. Transp. Syst.*, vol. 11, no. 10, pp. 685–694, 2017.

[17] C.-F. Juang and Y.-C. Chang, "Evolutionary-group-based particle-swarm-optimized fuzzy controller with application to mobile-robot navigation in unknown environments," *IEEE Trans. Fuzzy Syst.*, vol. 19, no. 2, pp. 379–392, Apr. 2011.

[18] R. Kala, A. Shukla, and R. Tiwari, "Fusion of probabilistic A* algorithm and fuzzy inference system for robotic path planning," *Artif. Intell. Rev.*, vol. 33, no. 4, pp. 307–327, Apr. 2010.

[19] Z. Li, I. V. Kolmanovsky, E. M. Atkins, J. Lu, D. P. Filev, and Y. Bai, "Road disturbance estimation and cloud-aided comfort-based route planning," *IEEE Trans. Cybern.*, vol. 47, no. 11, pp. 3879–3891, Nov. 2017.

[20] C. Luo and S. X. Yang, "A bioinspired neural network for real-time concurrent map building and complete coverage robot navigation in unknown environments," *IEEE Trans. Neural Netw.*, vol. 19, no. 7, pp. 1279–1298, Jul. 2008.

[21] C. Luo, S. X. Yang, M. Krishnan, and M. Paulik, "An effective vector-driven biologically-motivated neural network algorithm to real-time autonomous robot navigation," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2014, pp. 4094–4099.

[22] C. Luo, S. Yang, D. Stacey, and J. Jofriet, "A solution to vicinity problem of obstacles in complete coverage path planning," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2002, pp. 612–617.

[23] T. Maier, P. Sanders, and R. Dementiev, "Concurrent hash tables: Fast and general(?)!," *TOPC*, vol. 5, no. 4, pp. 16:1–16:32, 2019.

[24] H. Mo, Q. Tang, and L. Meng, "Behavior-based fuzzy control for mobile robot navigation," *Math. Problems Eng.*, vol. 2013, pp. 1–10, 2013.

[25] S. Qian, C. Jian, F. M. Le, I. Sahel, and M. Li, "SCRAM: A sharing considered route assignment mechanism for fair taxi route recommendations," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2015.

[26] Y. Rasekhipour, A. Khajepour, S.-K. Chen, and B. Litkouhi, "A potential field-based model predictive path-planning controller for autonomous road vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 5, pp. 1255–1267, May 2017.

[27] X. Su, M. Zhang, and Q. Bai, "Task-based wireless mobile agents search and deployment for ad hoc network establishment in disaster environments," in *PRICAI 2014: Trends in Artificial Intelligence*. Cham, Switzerland: Springer, 2014.

[28] K. Tang, M. Qian, and L. Duan, "Choosing the fastest route for urban distribution based on big data of vehicle travel time," in *Proc. Int. Conf. Service Syst. Service Manage.*, Jun. 2017.

[29] L. Wang, S. X. Yang, and M. Biglarbegian, "Bio-inspired navigation of mobile robots," in *Autonomous and Intelligent Systems*. Berlin, Germany: Springer, 2012, pp. 59–68.

[30] Y. Xu, J. Chen, X. Li, and Z. Luo, "Vector extraction for average total power estimation," in *Proc. Asia South Pacific Design Automat. Conf. (ASP-DAC)*, 2005, pp. 1086–1089, doi: 10.1109/ASPDAC.2005.1466529.
[31] B. Yan, D. Yang, J. Ding, K. Li, and X. Lian, "An adaptive algorithm for route guidance system based on DynamicTime division traffic network model," *Automot. Eng.*, vol. 25, no. 6, pp. 606–609, 2003.
[32] D. Yang, D. Zhang, K.-L. Tan, J. Cao, and F. Le Mouëll, "CANDS: Continuous optimal navigation via distributed stream processing," *Proc. VLDB Endowment*, vol. 8, no. 2, pp. 137–148, Oct. 2014.
[33] S. Yang and C. Luo, "A neural network approach to complete coverage path planning," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 34, no. 1, pp. 718–724, Feb. 2004.
[34] Z. Yang, "Algorithm of dynamic vehicle path selection based on the urban traffic control system," *J. Highway Transp. Res. Develop.*, vol. 16, no. 1, pp. 33–36, 1999.
[35] D. Zhang, D. Yang, Y. Wang, K.-L. Tan, J. Cao, and H. T. Shen, "Distributed shortest path query processing on dynamic road networks," *VLDB J.*, vol. 26, no. 3, pp. 399–419, Jun. 2017.
[36] Y. Zhang, S. Song, Z.-J.-M. Shen, and C. Wu, "Robust shortest path problem with distributional uncertainty," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 4, pp. 1080–1090, Apr. 2018.
[37] S. Zhao, Y. Chen, and J. A. Farrell, "High-precision vehicle navigation in urban environments using an MEM's IMU and single-frequency GPS receiver," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 10, pp. 2854–2867, Oct. 2016.
[38] D. Zhu, W. Li, M. Yan, and S. X. Yang, "The path planning of AUV based on D-S information fusion map building and bio-inspired neural network in unknown dynamic environment," *Int. J. Adv. Robot. Syst.*, vol. 11, no. 3, p. 34, Mar. 2014.
[39] T. Zhu and W. Xiang, "Towards optimized routing approach for dynamic shortest path selection in traffic networks," in *Proc. Int. Conf. Adv. Comput. Theory Eng.*, Dec. 2008.

ZHI CAI received the M.Sc. degree from the School of Computer Science, University of Manchester, in 2007, and the Ph.D. degree from the Department of Computing and Mathematics, Manchester Metropolitan University, U.K., in 2011. He is currently an Associate Professor with the College of Computer Science, Beijing University of Technology, China. His research interests include information retrieval, ranking in relational databases, keyword search, and intelligent transportation systems.

XUERUI CUI received the bachelor's degree from the School of Mathematics and Information Science, Hebei Normal University, in 2017. She is currently pursuing the M.Sc. degree with the College of Computer Science, Beijing University of Technology. Her main research interests include database systems, data mining, and analysis.

XING SU received the B.Sc. degree from the School of Software Engineering, Beijing University of Technology, Beijing, China, in 2007, and the M.Sc. and Ph.D. degrees in computer science from the University of Wollongong, Australia, in 2012 and 2015, respectively. He is currently a Lecturer with the Faculty of Informatics, Beijing University of Technology. His research interests include distributed artificial intelligence, multiagent systems, disaster management, and service-oriented computing.

QING MI received the M.Sc. degree in computer science and technology from the Beijing Institute of Technology, China, in 2012, and the Ph.D. degree in software engineering from the City University of Hong Kong, in 2018. She is currently a Lecturer with the College of Computer Science, Beijing University of Technology. Her research interests primarily concentrate on code readability, data mining and analytics, deep learning, and empirical experiments.

LIMIN GUO received the bachelor's degree from the Huazhong University of Science and Technology, in 2005, and the Ph.D. degree from the Institute of Software, Chinese Academy of Sciences, in 2012. She is currently a Lecturer with the Beijing University of Technology. Her research interests include database research and implementation and spatial-temporal data mining.

ZHIMING DING received the bachelor's degree from Wuhan University, in 1989, the master's degree from the Beijing University of Technology, China, in 1996, and the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, in 2002. He is currently a Professor with the College of Computer Science, Beijing University of Technology. His main research interests include database systems, mobile and spatial-temporal data management, intelligent transportation systems, sensor data management, and information retrieval. He owns five invention patents. He has published three books and about 120 articles in academic journals and conferences.