IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 5, 2020, accepted August 13, 2020, date of publication August 24, 2020, date of current version September 2, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3018729

A Novel Urban Emergency Path Planning Method Based on Vector Grid Map

BOWEN YANG[1], ZHIMING DING[1,2,3], LEI YUAN[1], JIN YAN[2], LIMIN GUO[1], AND ZHI CAI[1]

[1]College of Computer Science, Beijing University of Technology, Beijing 100124, China
[2]Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
[3]Beijing Key Laboratory on Integration and Analysis of Large-Scale Stream Data, Beijing 100144, China

Corresponding author: Zhiming Ding (zhiming@iscas.ac.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFC0803300, in part by the Beijing Natural Science Foundation under Grant 4192004, in part by the National Natural Science of Foundation of China under Grant 61703013 and Grant 91646201, and in part by the Project of Beijing Municipal Education Commission under Grant KM201810005023 and Grant KM201810005024.

ABSTRACT When an emergency occurs in the city, a large area of road congestion usually occurs. Therefore, it is particularly important to provide an effective emergency path planning strategy for vehicles. However, existing emergency path planning methods do not take into account the connectivity characteristics of the road network and commuting capacity. For this purpose, a Grid Map Emergency Path Planning (GMEPP) framework based on a novel model Grid Road Network (GRN) is designed in this paper. First, the road network data is divided into grids under equal spacing bands, and the roads data divided into different grids and use the commuting capacity of each road as the weight of each edge in the grid. Then a Grid PageRank (GPR) algorithm will be introduced, the output value of this methodology is calculated based on the capacity and number of connected edges of all vertices pointed by the external grid in each grid. The higher value of the grid will be recommended to users first when the path is planning. According to the GRN model, an improved Bidirectional Dijkstra will be applied to query the shortest path between two points, which is called Gird Bidirectional Dijkstra (GBD). At last, GMEPP uses Reverse Contraction Hierarchies (RCH) and Multiple Reverse Contraction Hierarchies (MRCH) originality methodologies based on intersection type to speed up the query algorithm GBD. To compare the efficiency of the proposed method, this paper conducted extensive experiments to verify. The results of the test showed that the Gird Bidirectional Dijkstra Multiple Reverse Contraction Hierarchies (GBD-MRCH) is better than other methods in different grid distributions.

INDEX TERMS Emergency path, path planning, transportation network, query optimization.

I. INTRODUCTION

In recent years, as the urban road network has become more complex, the degree of road congestion has increased. If the emergency with a road occurs, the vehicles will have a large area of congestion within a period. Therefore, emergency navigation has become more and more significant [29]. For example, if an emergency occurs in a certain zone, then this zone will be unsafe so that vehicles in this area should evacuate to safe areas immediately [26]. Elbery *et al.* proposed a vehicular Ad-hoc Networks (VANETs). This method uses linear programming optimization and stochastic routing to navigate vehicle crowds in case of an emergency evacuation or after special events [27]. Furthermore other researchers have a focus on the evacuation model of crowd type [28], [30].

Recently, with the maturity of car navigation technology, more and more people will choose to rely on the path provided by the navigation device to travel when traveling [2]–[4]. In this case, the navigation system has the known origination and the specified destination. However, based emergency, there may not have a destination when start navigating [33]. The only goal is to disperse vehicles and crowds to relatively safe areas as soon as possible.

So far, some researchers have proposed related evacuation models and models of evacuation congestion in this research direction [5]–[8], [35]. Chang *et al.* advanced dual route planning for unknown scaled [1]; paper proposed that

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

every path which has the emergency zones have separated functionalities. When an emergency happens, it can effectively evacuate people. Khalid *et al.* proposed that using an Immune-based approach to solve the dynamic path planning problem [9]. Other researchers have analyzed the multi-factor conditions during evacuation as the basis for choosing the evacuation path [21]–[24]. Dulebenets *et al.* aims to fill the existing gap by investigating the impact of a various factors (including driver characteristics, evacuation path characteristics, driving conditions and traffic characteristics) on the main driving performance indicators in emergency path planning [25].

Research path planning with emergencies falls into three categories: (1) Simulation methods that model traffic flow at a single vehicle level [13]. (2) Liner Programming methods that generate time-oriented optimal evacuation plans which minimize the total evacuation time [14]. (3) Heuristic methods such as the multi-objective path search in a smart city [15]. Which, based on Linear programming approaches, use flow network methods to evaluate the paths. The road network is transformed into a time-expanded network by duplicating the original evacuation network G for each discrete time unit $t = 0, 1, \ldots, T$ [16]. Sun *et al.* studied to timely decisions and efficient planning [34]. Understanding crowd dynamics is needed to enable real-time updates of immediate threats, identify patterns or impacts, and provide a practical and effective Emergency Route Planning (ERP) approach.

The above methods are all around the weight factors of the query path and the evacuation subject to do the evacuation path planning. In this paper, road network maps are added to the study of path planning in emergency situations as variable weights. If the road network map is slightly adjusted, the query result may differ from that in the original map. To address these issues, a Grid Map Emergency Path Planning (GMEPP) framework is proposed in this paper, and this method can effectively solve vehicle evacuation from an unsafe zone to safe zones. The Gird Bidirectional Dijkstra (GBD) algorithm is divided into two parts. The first part is to query the grid with higher commuting ability; the second part is the shortest paths query within the grid. However, for a large and medium-sized city, the number of intersections may reach ten thousand. When make a path query in the original graph, the method cannot return the ideal result quickly and effectively, so it is very important to reduce the number of intersections. Therefore, the two algorithms will be introduced: Reverse Contraction Hierarchies (RCH) and Multiple Reverse Contraction Hierarchies (MRCH) to reduce the number of intersections in the road network. In short, the contributions of this study are summarized below:

1. This paper introduce a grid road network model under the background of the emergency called GRN model.

2. A novel method called GPR to sort all grids and assign different rank value to each grid.

3. A GMEPP framework based on GRN will be described, this method uses GBD algorithm for shortest path recommendation, which can effectively use the grid road network data for path planning.

4. Two graph compression algorithms are proposed based on GRN and actual network characteristics: RCH and MRCH, to speed up the path query with emergencies.

The rest of this paper is organized as follows. In Section II introduces the theoretical knowledge of emergency path planning and problem definition. In Section III presents the two new models for emergency path planning. In Section IV introduces details of grid sorting GPR algorithm based on grid map. In Section V describes a path query GBD based on grid map. In Section VI considering the existing path query acceleration algorithm,an acceleration path query algorithm RCH and MRCH based on intersection type will introduced. In Section VII presents GMEPP with a large number of experiments. The conclude this paper in Section VIII.

II. PRELIMINARIES AND OVERVIEW

Emergencies in the city are usually causing varying degrees of property damage and casualties, and often accompanied by the need for decision-makers to do reasonable planning in a short time [10]. For example when a car is driving on the road in a city, a serious traffic accident occurs at a nearby intersection, causing crossing congestion and the inability of nearby vehicles to move for a certain period. It can waste not only many resources but also delays the best time to rescue the injured in the accident. Therefore, how to effectively evacuate vehicles to safe zones in the shortest time has become the focus of current research on establishing emergency navigation in cities.

A. RELATED THEORIES AND METHODS

So far, the traditional ERP approach that an evacuee can choose an infinite number of candidate routes poses as a major complexity problem when faced with large-scale evacuation as the search space is exponentially increased [11]. Yamada used the shortest path problem and the minimum cost flow problem to study the optimization of urban evacuation planning in major disasters [12]. It is mainly used in the emergency planning phase and does not consider the dynamic characteristics of real-time evacuation. Tao *et al.* used the context of Spatial Network DataBases (SNDB) for a real-time query to recommended route the users [20]. Zhu *et al.* studied the update of disaster information, path selection and transportation time selection of emergency relief materials [6]. Yang *et al.* studied an initial assignment scheme based on the clustering algorithm, which considers both the number of informed pedestrians and the number of uninformed pedestrians [16].

The generation of the navigation technology is to carry out reasonable path planning, and the processing of this technique is given the optimal path between two known places. When considering that vehicles in emergency areas need to be evacuated to safe areas, there are two purposes: one is to ensure people's safety and property to minimize losses. The second

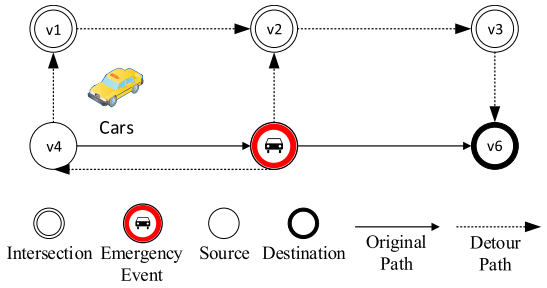


FIGURE 1. Emergency path planning instructions and detours.

is to reduce the possibility of road blockage during the evacuation process. Fig. 1 shows the basic evacuation rules.

Emergency path planning based on the navigation of the grid has higher accuracy and better efficiency [36]–[39]. Emergency navigation should have two categories: the first type is for evacuation route planning for vehicles in the emergency area; the second type is for detour route planning for vehicles that are far from the emergency area. If the vehicle far away from the place of the emergency area, when consider that the traffic jam has little influence on the first navigation path. Therefore, this paper conduct research on the first type of evacuation problems. It is worth noting that in this paper, not only the road with the shortest distance is recommended to the vehicle during the path planning, but it also ensures that the grid area passed by the vehicle has a high-rank value.

The GMEPP framework based on the grid map to solve the fast evacuation for vehicles. When considering interactive traffic conditions, evacuation routes should have a higher commuting ability as possible. Normally, the grid with more interactions indicates that it has a relatively good traffic guidance ability when happening emergencies. In this way can guarantee that the proposed method will be more effective for evacuation vehicles. In other words, the framework will give the vehicle priority to provide the area with a larger rank value as the choice for evacuation because it define that the larger the rank value that the higher the commuting capacity of its area. Thus, people who are driving the car can have more than one path to successfully arrive at the destination and minimize the impact of emergencies on navigation as much as possible.

There are some advantages to path planning on a grid map, (1) the ability to identify urban areas and streets quickly; (2) uniqueness of location; (3) convenient for path planning. But it also has some disadvantage: the efficiency of path planning is not high enough and the space could be wasted (the resolution of the grid does not depend on the complexity of the environment); accurate vehicles position estimation is needed; the effect of human-computer interaction for object recognition is not good. To address these defects, a path planning method called GBD will be given details in Section V.

B. PROBLEM DEFINITION

When the vehicles are driving on a given path by the navigation device, if ahead roads or areas conditions occur changed, the navigation device needs to switch to a reasonable other path quickly. The problem can be described as when the vehicles are driving in an emergency grid (the rank value of the grid is -1 when an emergency occurs). GRN model can provide a reasonable evacuation path for the vehicle, so that the vehicle can drive from an unsafe area to a safe area in a short time. This paper will divide the grid into two categories: one is defined as the $US - GRID$ (take place an emergency grid, i.e. unsafe grid). The rest is defined as $S - GRID$ (safe-grid, these grids are far away from the $US - GRID$ cars should carry on passing round than original paths). The vehicles in $S - GRID$ will get the navigation information about the exact location of the emergency (i.e., the grid ID) and the time of occurrence. The definition of problems is as follows.

1. The grid map, assigned according to the regional characteristics, can effectively carry out path planning. However, the current research does not have an effective method for dividing the grid map and how to assign values to the grid.

2. When an emergency occurs, the existing path planning methods cannot effectively make a reasonable evacuation path planning scheme based on the grid map.

3. Existing graph-based accelerated query algorithms are no longer suitable for computing in the grid map case.

According to the first question, input a road network map $G = < V, E >$, which contains the latitude and longitude information of the V. Calculation by GPR method, output a grid map with weight. The second problem is to input two latitude and longitude coordinates. According to the GBD method, an evacuation path SP (i.e., Shortest Path) will be returned, and the grids have a large rank value compare with adjacent grids in which the path passes. There is define the shortest path as the closest path from the current position to the higher rank grid. The input of the third problem is a grid map Gn, and a contraction grid map with a small number of vertices compared to the original grid map is obtained by the graph compression method.

The vehicles in which the $US - GRID$ once receive notification of an emergency, the first response was to choose the nearest safe area for evacuation. However, selecting an evacuation grid as a unit of the partitioned grid is important to escape information recommended by current navigation equipment. The quality of the recommended grid directly affects the efficiency and time of evacuation. For example, if the road in a selected grid is smooth and the traffic flow is fluent (i.e. more lanes), and there are more intersections and more choices, the probability of such grid being selected as evacuation is larger than other grids. Because the grid has relatively large interaction, then the evacuation routes available for vehicles will be more selective. The higher the selectivity, the greater the probability that the vehicles can reach the original destination. Which grid to recommend will be the focus of this paper research and discussion. Table 1 shows the main parameters mentioned in this article.

III. TWO NEW MODELS BASED ON EMERGENCY PATH PLANNING

In the section will describe two models: the emergency navigation road network model and the grid map model. Both

TABLE 1. Notations.

Notation	Definition
V	A set of intersection
E	A set of ordered pairs of vertices
T	Represents the average time of the vehicle passing through the road of two intersections
$Lenght$	Stands for the length of the road
S	The average speed of the road
T_q	The query time is taken into consideration
n_g	Represents the number of partitions
Gn	The whole city map in gird $n_g \times n_g$
$P(w)$	The ranking values of the web page
$P`(w)$	The ranking values of the web page without infinite loop
R_i	The GPR value of a grid
P_i	The road weight with inbound the R_i
com	Stands a collection of different vertices pointed to the same grid in one grid, the value is the number of edges R_i
V_{grid_i}	Stands a total number of vertices in ith grid
N	The number of grid, all of which contain at least one intersection
$GIDN$	Represents a grid number of In-degree
$dist[\varepsilon]$	Stores the temporary shortest path in each grid
MV	Refers to the maximum number of vertices in the grid after different grid
AIG	Stands for the average number of intersections in one grid
ARG	Stands for the average number of roads in one grid
$Variance$	Represents the distribution density of nodes
NGS	Represents the GBD searched the number of grids
NVS	Represents the GBD searched the number of vertices
CV	Represents have been contracted the number of vertices in the road network
CE	Represents have been contracted the number of edges in the road network
ART	Stands for the average runtime of the algorithm

models are improved based on the graph structure pattern. In particular, the GRN model improves the original graph structure model to a model that uses the characteristics of regional roads to represent edge weights.

A. ROAD NETWORK MODEL

A Spatial-Temporal graph based intersection can be represented as a directed Graph G $= < V, E >$, where V is a set of intersections and $E = < v_i, v_j >$ is a set of ordered pairs of vertices and indicate that there is a path between v_i to v_j, v_i and $v_j \in V$. Noted that the direction of each path is a vector and the time at point v_i is ahead of v_j time. One path $< v_s, v_d >$ means vehicles start form source point v_s and their destination is point v_d, with a weight $w := < v_i, v_j, T >$ defined as time cost for a pairs of intersection i to j, w is a non-negative number is all lengths are positive.

It is emphasized that the time T represents the average time of the vehicle passing through the road of two intersections, and the value of T is a function represented as $T = Lenght/S$, which is not fixed and $Lenght$ stands for the length of the road, S is the average speed of the road. The speed of the road network S at the query time T_q is taken into consideration in the path planning method. The purpose of T_q is to ensure the optimal path for users at this moment, because S of the road has a different value at different times.

B. GRID ROAD NETWORK MODEL

Grid map is modeled as a $Gn = < Pr, grid_{id}, (v_1, v_2, \ldots, v_i) >$, where Gn is the whole city map in gird $n_g \times n_g$ and $grid_{id}$ represents ID in the map (the value of n_g will be given in details in the experiment section), P_r indicates a ratio of the number of intersections in the grid to the number of intersections in other grids. The value of this ratio will be considered as the commuting ability of the grid. The sorting algorithm will adopt a PageRank-based grid sorting algorithm GPR (the implementation of the algorithm will be described in detail in Section IV). v_i stand for all the vertices in the grid, the more intersections in a grid, the stronger the grid's ability to communicate to other grids, the other is an out-degree set $V_{out} \in V$ that contains all the vertices in each grid.

The grid map divides the entire map into several sub-maps, $G_s \in G$. There will be several intersections and roads in each grid, and there will take these factors into account when assigning values to the grid. After dividing the map into multiple areas, the framework can assign values to each area according to the setting of parameters. These values provide an important basis for the evacuation path planning. The change of these values exactly reflects the weight change of the region, and at the same time provides a reliable basis in theory, which guarantees that the method will quantify intersections and roads into computable weight problems when algorithm search for paths later. This is exactly the advantage of doing a path search on a grid map. It is worth noting here that the irregular shape of the city map causes some grids to contain no intersection or road after our grid, so these grids ID will be deleted from the data during preprocessing.

IV. GRID RANK BASED ON ROAD NETWORK MAP

Under the influence of emergencies, the navigation path's first response should be to evacuate the existing vehicles to surrounding areas quickly. However, the intersection and road factors will directly affect the quality of the route (i.e. commuting capacity). In the following will introduce a GPR method based on a grid road network map, which assigns a rank value to the grid with intersection and road factors.

The sorting algorithm used in this paper is based on the improved GPR algorithm of the traditional PageRank method. In 1999, the paper [18] by Page *et al.* introduced an algorithm called PageRank. The main idea of this algorithm is that the more 'effective' a page is, the higher the quality of links of the Page, and the easier it is to other 'effective' pages. Therefore, the algorithm fully utilizes the relationship between web pages to calculate the importance of web pages. The uniqueness of this method is that it takes into account the correlation between all entities. The traditional value of rank

is calculated by Formula (1) and Formula (2).

$$P(w) = e \sum_{S_m}^{S_n} P(m)/N_m, S_m \in S_n \quad (1)$$

$$P'(w) = e \sum_{S_m}^{S_n} P'(m)/N_m + eE(w), S_m \in S_n \quad (2)$$

where $P(w)$ and $P'(w)$ are the ranking values of the web page. It is note that the $P(w)$ result contains the possibility that the rank is 0, which means that there is an infinite loop, but the Equation (2) plus the damping factor prevents the rank from being zero, because the page is zero will not be sorted. e is damping coefficient, m represents current web page, S_m belongs to a page linked into $P(w)$ and $P'(w)$, S_n refers to all pages linked to page m, $P(m)$ and $P'(m)$ indicate the rank value of the current page m, $E(w)$ is some vector over the web page that corresponds to a source of rank and N_m is the total number of pages. Finally, the web page will be sorted according to the value of $P'(w)$, and the high rank value will be recommended to users first. Equation (1) and Equation (2)

The grid map's purpose is to map the entire city to different grids and calculate the number of intersections and the grid's capacity as the weight of the grid. The Fig. 2 shows that the grid intersection (blue dots) is divided into different grids. According to GPR, $grid_1$ with four intersections ranks higher than $grid_2$ with two intersections, because $grid_1$ has better evacuation ability than $grid_2$ in terms of the number of intersections. However, the actual situation also need to consider road connectivity, which is the $In-degree$ and $Out-degree$ of each intersection and in the grid maps. In order to solve this problem, this article introduces a grid-based map GPR algorithm for grid rank value. In this method that intersections in each grid (vertices in the network) and specifies the number of vertices in each grid defined as n_g, the size of n_g is a factor for sorting the grid in the whole road network. The selection of n_g plays a key role in the grid sorting. Considering the number of lanes in each pair of vertices. The more lanes, the road has, the greater the road at a certain time than the road with fewer lanes.

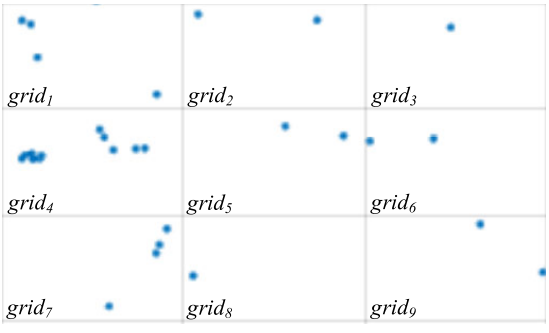


FIGURE 2. Distribution of intersection on grid map.

The rank value of the grid is iteratively calculated by the GPR method. GPR treats the number of edges as all the vertices in the grid connected to other grid vertices. The number of edges out of the grid is defined as $Grid\ \ In-Degree\ \ Number$ $(GIDN)$. This value will be added to the Equation (4) as a parameter to grid rank. The rank value of each grid can be calculated by Equation (3), Equation (4) and Equation (5).

$$GPR(grid_{ID}) = \sum_{i=1}^{k \in N} R_i * P_i + (1-e)/N \quad (3)$$

$$R_i = \sum_{i=1}^{com} GIDN_i * e * L \quad (4)$$

$$P_i = \frac{V_{grid_i}}{V} \quad (5)$$

In the Equation (3), Equation (4) and Equation (5) that the result of GPR($grid_{ID}$) $\in$ [0, 1) is the rank value of the grid (the range of values of GPR will be described in detail in the experiment section). If GPR($grid_1$)<GPR($grid_2$), then GPR($grid_2$) will be recommended because it has higher value. Where k := the number of the grid in which vertices are directly connected, it indicates that the vertices in the grid are not necessarily connected to the adjacent grids. R_i is the GPR value of ith grid, i stands for the ID of the grid, com is defined a collection of different vertices pointed to the same grid in one grid, the value is the number of edges. L is the number of lanes of the road. A damping factor e <1 is to prevent local circulation between grids and increase the rate of convergence. P_i is weight, and the larger the P_i, the larger the number of intersections in the ith grid indicates bigger weight, $grid_i$ stands ith grid, and V is vertices total number in the whole network. V_{grid_i} is a total number of vertices in ith grid, meanwhile the more likely it is to be selected when navigating. N is the number of grids, all of which contain at least one intersection.

Algorithm 1 GPR

Input: road network $G = < V, E >$
for *coordinate* **in** G **do**
  Grid = ID
  for calculation number of *grid* **and** *line* **do**
    line = weight$< v_i, v_j >$
    A_{matrix} = Edge *outdegree* $< v_i, v_j >$
  end for
end for
Initialization $R_i = 1/N$
for GPR-Iterate(*grid*) **do**
  $R_0 = 1/n$
  k=1
  for $R_{k+1} = A_{matrix} * e * P_i * L + (1-e)/N$ **do**
    $k = k$+1
    if R_{k+1}-R_k < *sum* **then**
      return R_k, G_s
    end if
  end for
end for

First, the algorithm loads the road network data, rasterizes all intersections into different grids according to given coordinates, and assigns an ID to each grid. Count the number of intersections in each grid, the number of $grid_{number}$ entrances and exits *GIDN* (*GIDN* details will be given in this section of the end part), these are taken as rank factors. According to [18] initialize the grid and the rank value of each grid is 1. At this time, the value of the edge is $1/N$ and weighted to the rank value of the connected grids, repeat until the rank value of all grids is less than one parameter, then the algorithm stops. where $N \in grid_{number}$, $e = 0.85$. A_{matrix} is a matrix for each rank calculation.

The number of map grids dominates the running time of the algorithm after rasterization. In one iteration, the algorithm needs to calculate the rank value of each grid in turn, i.e., $O(n)$. When calculating the value of a certain grid that need to consider the adjacent grid's rank value. In this analogy, when calculating the rank value of each grid, the other grids must be calculated again, i.e., $O(n-1)$ times, so the time complexity of the GPR algorithm is $O(n(n-1))$, i.e., $O(n^2)$, where n is the number of intersections in the road network.

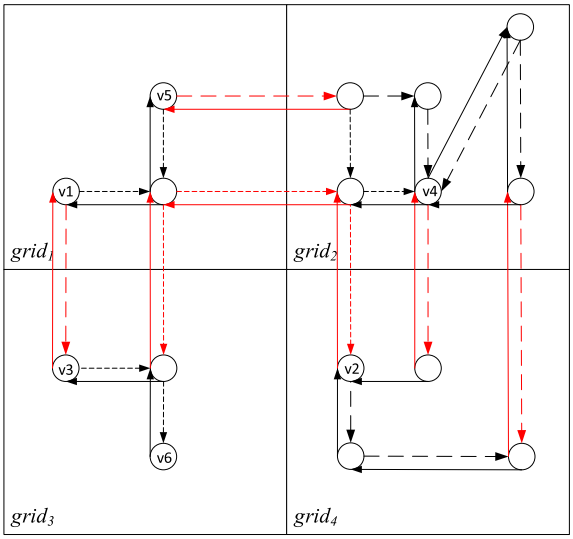


FIGURE 3. Connectivity of intersections.

In Fig. 3, the distribution of intersections with four grids and their $In-degree$ and $Out-degree$ are simulated. The points in the figure represent the intersections, the dotted lines and solid lines represent the roads in the city, and the red lines stand for the path from this grid to another grid. In Fig. 3 shows the v_4 in $grid_2$ belongs to the mentioned circular intersection in the above section. Generally, such an intersection has five to six entrances, and exits, the intersection of this type has the strongest connectivity. v_2 has a one-way path in that all are slightly less connected than v_4. *GIDN* represents one lane of a road, and $GIDN * L$ represents the commuting capacity of the road.

When GPR calculates the rank of $grid_2$ and $grid_4$, the number of junctions of $grid_2$ have six intersections and $grid_4$ have four intersections so that the $grid_2$ has higher rank value. However, when comparing $grid_1$ with $grid_3$, because the number of intersections of $grid_1$ is equal to that of $grid_3$, it can't solve the rank problem very well.

V. PATH PLANNING BASED ON GRID MAP

An effective query algorithm can achieve the goal of path planning in a short time. Traditional path recommendation methods include the Bidirectional A-star algorithm, Bidirectional Dijkstra algorithm, D-star algorithm, and so on. However, these conventional path planning methods are no longer applicable to GRN model. A new path planning method needs to be designed in conjunction with a grid map because of the need to combine secondary path planning and phased planning in emergency path planning.

A. TRADITIONAL BIDIRECTIONAL DIJKSTRA

First review the Bidirectional Dijkstra algorithm, which is a famous method for finding the shortest distance between two points. The Bidirectional Dijkstra algorithm is improved in its initial one-way stage [19]. In *FORWARD* and *BACKWARD* search that the edge with the smallest boundary weight is found from s to d. In the shortest path search process, firstly, four sets are established, namely $forwardset(fs) \in V$, $backwardset(bs) \in V$, $forwardrelaxed\ (fr) \in SP < s, d >$ and $backwardrelaxed\ (br) \in SP < s, d >$. These four sets store *Map* and *Heap* for vertices of forward and reverse searches. *Map* is defined as maintaining the set of vertices that have popped up from *Heap* sets. *Heap* is defined as maintaining the set of candidate vertices that have been processed.

To illustrate, Fig. 7 (a) shows the search process of the Bidirectional Dijkstra algorithm form source s to destination d, i.e. Bidirectional Dijkstra algorithm will try to find Shortest Path $SP < s, d >$. The algorithm begins with forward search (starting from source vertex label F) and backward (starting from destination vertex label B). Initially, $SP < s, d >= \infty$ and processed $(v \in V) = \emptyset$, and storage in the form of triple array as $< v_i, v_j, weight >$.

The forward search and the backward search should compare whether the current fr and br vertices are the same in vertex u before each search. If fr equal br that when the first candidate optimal path is obtained, otherwise the two-way search should be continued if an estimated optimal of the current vertex is larger than current potential optimal in any direction, the search in that direction stopped.

B. GRID BIDIRECTIONAL DIJKSTRA

A GBD algorithm based on the GRN model will be introduced in GMEPP framework. About the GBD algorithm, the number of intersections in each grid directly affects the ability of evacuation and connectivity. The larger the rank value that is the greater the probability of evacuation efficiency. There are two types of navigation in our method. The first one is emergency navigation, which requires two or more path queries. The second one is to navigate around the area near the emergency area.

Firstly, no matter what kind of emergency path planning methods aim to evacuate all vehicles close to emergencies orderly to safe areas. The conditions here are no longer suitable for general path planning algorithms. Because the recommended path is no longer the shortest distance problem, but how to quickly evacuate the vehicles near the emergency and maintain the smooth flow of the disaster area in a certain period. Secondly, many graph search technology needs to set emergency area as evasion area, carry out secondary navigation for existing vehicles passing through the area, first guide them into the nearby safe grid, have higher ranks, and use Bidirectional Dijkstra algorithm for secondary path planning until the destination. In the same grid, the Bidirectional Dijkstra algorithm is used to search. The starting point is the vehicle's current position, and the endpoint needs to be further subdivided.

The GBD algorithm judges the distance from the disaster area according to the starting point of the input. If the distance is within the same grid, the grid with higher rank value nearby will be searched first and the path planning will be carried out. It is noted that there need to do a further analysis of the destination point coordinates. The final evacuation destination must be different from the current vehicle itself (mainly to distinguish between $US-GRID$ and $S-GRID$). When the algorithm reads the grid adjacent to a higher rank, it calculates the ID of the nearest intersection of the grid and returns it to destination vertex d.

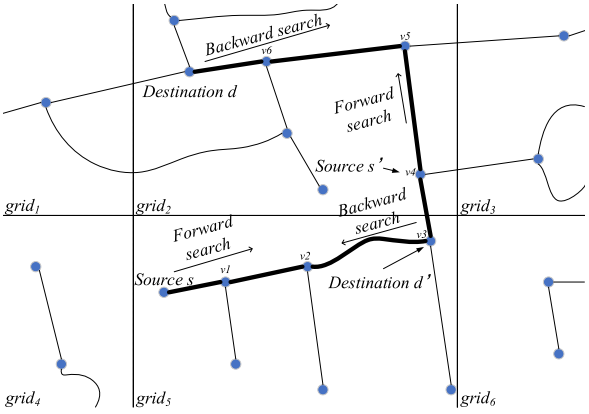


FIGURE 4. Grid bidirectional dijkstra search.

Fig. 4 shows how to do path planning from $grid_5$ to $grid_2$. s is the starting vertex and d is the target vertex. The whole $Path = |<s,v_1>,<v_1,v_2>,<v_2,v_3>,<v_3,v_4>,<v_4,v_5>,<v_5,v_6>,<v_6,d>|$, where $v_3 = d'$ and $v_4 = s'$. d' is the temporary target vertex in the same grid, and s' is the starting vertex in the adjacent grid. The edges of d' to s' belong to adjacent edges.

The GBD algorithm process is described as follows: The source vertex s is popped from *Heap*. If the next search vertex u has the same grid as vertex s, then the two-way algorithm is used to query and calculate the shortest path. However, when the next vertex is in a different grid, the rank between the grids are compared by GPR, and the grids with large values are selected to continue searching. The array $dist[\varepsilon]$ stores the temporary shortest path in each grid. Path records all the passed grid nodes ID and $dist[\varepsilon]$ for each grid. The Equation (8) shows the range of values for $dist[\varepsilon]$. The Equation (6) and Equation (7) for the shortest path with GBD is as follows:

$$SP<s,d,grid>=min|SP<s,u,grid_s>+SP<u,d,grid_d>| \tag{6}$$

$$SP<s,u,grid_s>=min|<s,u_0,t_0>+<s,u_1,t_1>+<s,u_2,t_2>+\ldots+<s,u_n,t_n>|, \quad t_0,t_1,\ldots,t_n\in T \tag{7}$$

$$dist[\varepsilon]=\begin{cases}0 & \varepsilon=s,\\ \infty & \varepsilon\neq s,<v_s,v_\varepsilon,T>nonexistent,\\ w & \varepsilon\neq s,<v_s,v_\varepsilon,T>existent,\end{cases} \tag{8}$$

where T is the time to pass v_s and v_ε. The critical part of the GBD algorithm is to find the most suitable path between and within the grid for the evacuation of the vehicle. By input a source s, a destination d and query time T_q, the algorithm will return two lists *GridSet* and *PointSet* eventually. *GridSet* and *PointSet* will store the list of result grids ID and the list of points the vehicle needs to go through, respectively. The algorithm 2 describes the processing of GBD. The above three equations are an improvement on the original equation of traditional Bidirectional Dijkstra algorithm, adding the grids search and a limiting factor T, and changing the original shortest path as the purpose to the optimal path. The optimal path here refers to the time taken by the vehicle from the emergency point to the terminal point is less than the time taken by other paths.

It is worth noting that if the query point is not within the latitude and longitude coordinates contained in the intersection, for example, on the road or at a location on a nearby road. Then our GBD will first perform the offset processing of the query point. In other words, is select the intersection point closest to the query point as the input point during query through the Euclidean distance between the two points. Here GBD will compare the latitude and longitude to select the appropriate query input point.

The time complexity required in the path search process in one direction is $O(n^2)$. From the input point, the algorithm needs to query the edge weights between adjacent points in sequence, so it needs to be calculated in sequence. At the same time the GBD algorithm uses *Heap* to store point, so the time complexity when searching in both directions is $O(2logn)$, n is the number of vertices.

The GMEPP framework stipulates that the more intersections in the grid have, the stronger the commuting ability, the better the ability to access other grids. When disasters take place that the surrounding vehicles will be required to evacuate to the grid with higher rank value at the first time.

Algorithm 2 GBD

Input: G_n **and** Source s, Destination d, Query time T_q
Initialization Matrix[i][j], Matrix[$grid_i$][$grid_j$], fs, bs, fr, br, *GridSet*, *PointSet*
if Source.$grid_x$ == Destination.$grid_y$ (Means to make a path planning within the grid) **then**
  for forward search s and backward search d **do**
    fs = s, bs = d
    relax edges must be belongs to the same $grid_{id}$
    fr = s, br = d
    $SP < s, d, grid_{id} > = \min |SP < s, u, grid_{id} > + SP < u, d, grid_{id} >|$
    PointSet = s, u, d
    if *SP* has been found **then**
      return *PointSet*
      break
    end if
  end for
end if
if Source.grid_x != Destination.grid_y **then**
  for forward search *Gs* and backward search *Gd* **do**
    choose adjacent $grid_{id}$ has a lager RANK value
    if GPR($grid_i$) > GPR($grid_j$) $i, j \in$adjacent *grid* **and** they must have link edges **then**
      $GridSet = grid_i$
    end if
    if *Gs* == *Gd* **then**
      return *GridSet*
      break
    end if
  end for
end if

It can evacuate the vehicles near the emergency site to safe areas effectively.

VI. ROAD NETWORK OPTIMIZATION

Preprocessing generates a multi-layer structure, each point in a layer. Priorities of vertices should be defined in advance, and sorting conditions can be set as needed, directly affecting the efficiency of preprocessing and searching. Contraction hierarchies algorithm uses the number of intersections from low to high as a priority in order to select the points for contraction. With more forks there are at the intersection and it has a stronger capacity for vehicles, and the ability to connect with other intersections is about smaller than the number of forks.

A. VERTICES CONTRACTION HIERARCHIES

The traditional CH algorithm contracts the vertices from a higher level to a lower level [16], [17]. For example, assume that vertices of a graph $G = < V, E >$ are named as $(1, 2, \ldots, n)$ in ascending order in terms of *importance*. This property is achieved by replacing paths of the form $< u, v, w >$ by a shortcut $< u, w >$, note that the $< u, w >$ is only required if $< u, v, w >$ is the only shortest path from u to w. The higher the capacity of the vertices, the higher the contraction level. Conversely, the lower the vertices capacity, the lower the contraction level. Therefore, contraction of the vertices in the road network by the traditional method is not required, so the order of contraction needs to be changed. While other studies relatively focus on time and turn [31], [32]. However, GMEPP will be more focused on the characteristics of the intersection in an emergency situation.

B. REVERSE VERTICES CONTRACTION HIERARCHIES

In the urban road network, the vertex is usually expressed as an intersection, known that the intersection is often divided into L-Junction intersection, T-Junction intersection, and Circular-Junction intersection (there maybe five to six forks). L-Junction is an intersection with two entrances and two exits. This kind of intersection usually appears only in the dense area of residential buildings in the old urban area. T-Junction is designed to solve the problem that the distance between the two intersections is too long; this kind of intersection also appears more in the old urban areas. The crossing is the most common way of the intersection, which usually occurs in urban areas and branches or main roads. This type of intersection will be used as the main way of intersection in busy urban areas. The last one is the Circular-Junction intersection, which usually appears for vehicles moving in different directions, and the intersection usually accompanies the emergence of turntables. When the vehicle reaches the intersection, it will go up to a circular road to circle, and the direction of the disc is fixed to counterclockwise. Fig 5 (a) and (b) have shown the distribution and type of junctions, respectively.

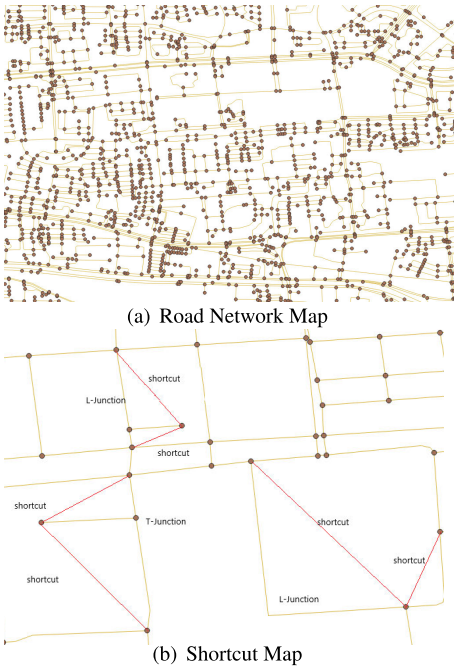
In order to address the different characteristics of intersections, this paper studies a Reverse Contraction Hierarchy (RCH) based on the grid map to reduce the computing time of the shortest path. The important factors will be discussed in traditional CH calculation before introducing RCH method.

The meaning of *importance* in traditional CH algorithm refers to the number of entries and exits vertices. The vertex with a higher degree is more important than others; the traditional CH algorithm is based on the principle that the vertices in the original network are contracted. RCH method will reverse this principle. For example, an intersection is an L-Junction intersection, $\alpha = 1$, and satisfies the intersection that must be crossed on multiple shortest paths and does not affect the value of other shortest paths. To be frank that the more important the intersection is visited, the more likely it will be contracted. RCH will be set a contraction factor α (α represents the intersection type), and the value of α will determine whether contraction should be applied to this vertex in our RCH method.

The main factor affecting the time complexity of RCH is to access the vertices in the graph. RCH is compared to the traditional CH algorithm that eliminates the shortest path search process. Each vertex in the network only needs to

Algorithm 3 RCH

Input: G_n, *IndexTable*
Initialization *IndexTable*
for $v_i \in V$ classify by "intersection-Type" **do**
  if *out* − *degree* number of $v \leq 2$ **then**
    α_i = "L-Junction", delete v_i form G and $< v_{i-1}, v_{i+1} > = < v_{i-1}, v_i, v_{i+1} >$
    weight$< v_{i-1}, v_{i+1} >$ = weight$< v_{i-1}, v_i, v_{i+1} >$
    shortcut $< u, w >$ add LinkedHashMap $< key, value >$
    shortcut $< u, w >$ store *IndexTable*
    $key = u - w$, $value$=weigth$_{uw}$
    α_i = "contracted"
  end if
  if *out* − *degree* number of $v_i == 3$ **then**
    α_i = "T-Junction"
    if $\alpha_i \in$The only way which must be passed **then**
      delete v_i form G and $< v_{i-1}, v_{i+1} > = < v_{i-1}, v_i, v_{i+1} >$
      weight$_{<v_{i-1}, v_{i+1}>}$ = weight$_{<v_{i-1}, v_i, v_{i+1}>}$
      α_i = "contracted"
    end if
  end if
  if α_i has no contracted **then**
    α_i = "Non-contracted"
  end if
end for



(a) Road Network Map

(b) Shortcut Map

FIGURE 5. (a) The road network intersection distribution (b) Contraction diagram of L-Junction and T-Junction intersections.

traverse in order, so the time complexity of RCH is $O(n)$, n is the number of vertices.

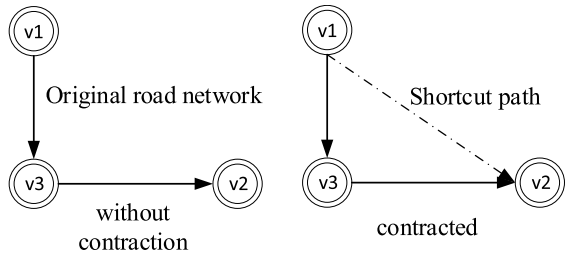
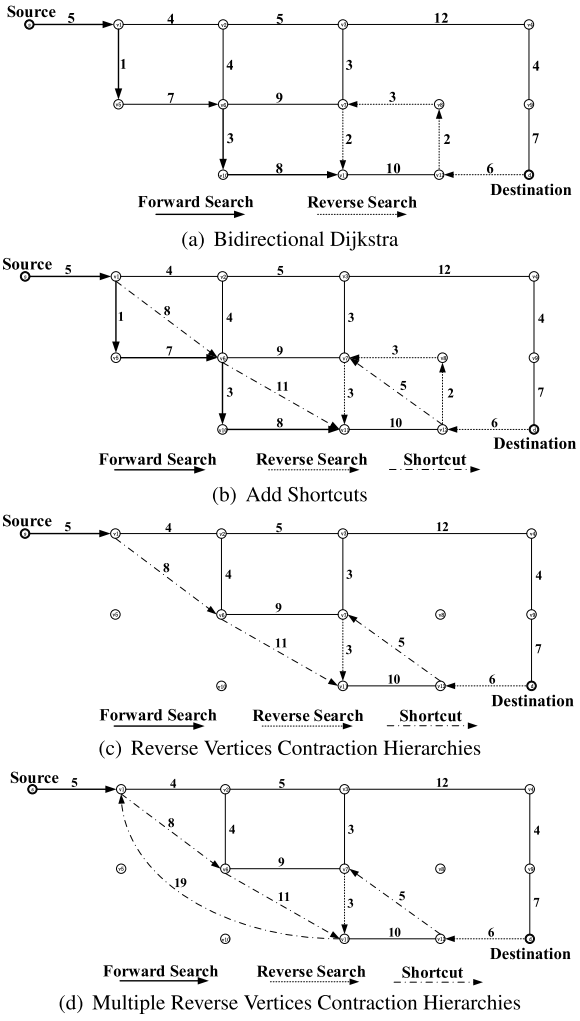


FIGURE 6. Schematic diagram of contraction at L-Junction intersection.



(a) Bidirectional Dijkstra

(b) Add Shortcuts

(c) Reverse Vertices Contraction Hierarchies

(d) Multiple Reverse Vertices Contraction Hierarchies

FIGURE 7. Shortest Path Planning Process. (a) The vertex search state without contraction (b) The vertex search with RCH (c) The vertex search with after RCH (d) The vertex search with after MRCH.

As shown in the Fig. 6, there are three vertices (v_1, v_2 and v_3) constitute a L-junction road $< v_1, v_3, v_2 >$. Form the Fig. 6 shows that vertex v_1 to vertex v_2 passes through vertex v_3, and the path is the shortest path form v_1 to v_2. At this point that can connect v_1 and v_2 with a dotted line, called shortcut between v_1 and v_2 $< v_1, v_2 >$ without v_3. RCH will store this shortcut on the original road network, where v_3 will be deleted in the road network. In other words, v_3 is stored in

the actual road network, but the vertex will be removed in the contracted road network. In searching for the shortest path and it will display v_1 to v_2 directly, while v_3 will be added directly to the feedback of users. So that the query time of the shortest path can be reduced and ensure the integrity of the shortest path information.

The actual complex urban road network has a large amount of data. The single-use of the Bidirectional Dijkstra algorithm cannot satisfy the rapid and effective evacuation and navigation of vehicles near the incident in a short time.

From the Fig. 7 (b) and (c) show that in the process of forward search from vertex v_1 to vertex v_6 and vertex v_6 to vertex v_{11}, there is only one shortest path $SP < v_1, v_5, v_6 >= 8$ and $SP < v_6, v_{10}, v_{11} >= 11$. In other words, if a car wants to get to vertex d from vertex s, it must go through vertex v_5 and v_{10} in the shortest pathway. In this case, there met the L-Junction mentioned in the query section that in the actual road network, (it is noted that there have two tips: one side may have roads under vertex v_5, but for various reasons, there is no link in the updated road network. On the other side, there are also T-Junction in the actual road network. T-Junction can be viewed as two L-Junction, which is also applicable to the RCH methodology), so in this case RCH contracted the vertex v_5 and v_{10}, then connect an edge $< v_1, v_6 >$ between v_1 and v_6, and assign the weights on $< v_1, v_5 >$ and $< v_5, v_{10} >$ edges directly add to $SP < v_1, v_6 >= 8$. This new edge $< v_1, v_6 >$ called shortcut and store it in the dataset of the road network.

C. MULTIPLE REVERSE VERTICES CONTRACTION HIERARCHIES

Algorithm 4 MRCH

```
Input: G_rch, IndexTable
Iteration RCH
for DFS G_rch do
    if contraction vertices != 0 then
        break
    else
        if shortcut < u, v >, shortcut < v, w > adjacent
        then
            shortcut < u, w > add LinkedHashMap <
            key, value >
            key = u-w, value = weigth_uw
            shortcut < u, w > store IndexTable
        else
            continue DFS
        end if
    end if
end for
```

So far, a RCH method is implemented based on the existing road network datasets. The shortest path query before and after contraction can be seen from Fig. 7 (a) and (c). There are $V = 14$ vertices and $E = 18$ edges in the original graph (because the directed graph is simulated as an undirected graph, the number of edges stored in the directed graph maybe twice as the number of undirected edges, i.e., $E \in [18,36]$). The number of vertices and edges in the contracted graph is 14 and 15, respectively. It is obvious that there are three fewer edges than without contraction in Fig. 7 (c).

After contracting the road network data, noted in Fig. 7(c) that the two shortcuts connected by vertex v_6 are $< v_1, v_6 >$ and $< v_6, v_{11} >$ respectively. If a vehicle want to go from v_1 to v_{11} then have to go through these two edges and vertex v_6. The connection black-wide dotted line of vertex v_1 and vertex v_{11} as shortcut edge $< v_1, v_{11} >$. In this case RCH method will continue to contracted vertex v_6 as shown in Fig. 7(d).

Algorithm 4 describes the calculation process of Multiple Reverse Vertices Contraction Hierarchies (MRCH) algorithm. The time complexity is related to the vertex in RCH, and the algorithm needs iterates multiple times, so the time complexity of MRCH is $O(nlogn)$, n is the number of vertices.

VII. EXPERIMENT AND VERIFICATION

A. EXPERIMENTAL SETTINGS

In this section the experimental process and parameter settings will be introduced in detail. The experiments will try different grid sizes and the number of nodes in the grid and the road density as different parameters to test the effect of path query efficiency. According to the following two reasons, our experiment did not compare the existing emergency path planning methods. (1) Different sampling frequencies datasets. (2) A different definition of road network structure. The GMEPP will be main focused on quickly importing vehicles into the grid with a higher rank value. The rest of the paper will be conducted three experiments.

1. The road network divides into a structure composed of $n_g \times n_g$ grids of the same size with different grid numbers and evaluates the effect of the division. Since the grid at equal intervals (in order to meet the priority requirements for sorting), the city center's road density is greater than that in the suburbs.

2. According to the divided grids, sort and assign the distribution of all grids. In the actual road network that the carrying capacity of the road is constant. And in the process of driving, the vehicles usually abides by the first-in-first-out principle. The capacity here refers to the maximum value of road carrying vehicles.

3. The efficiency of algorithms will conduct comparative experiments on the proposed path planning methods, including three graph compression methods. The runtime of the search path represents the contraction result.

Our experiment environment is 16G memory, 64-bit Windows 10 operation system and Intel i5 @3.30GHz CPU. The algorithm compilation language is Java and MATLAB, the visualization tool is QGIS. Datasets is Beijing city road network from OpenStreetMap[1] that contains 83884 Vertices and 222778 directed edges after processing refinement, suspension points and lines are not include (suspension points

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

and lines refer to those isolated points and edges in the graph, such points and lines usually appear in the villages on the edge). There are also about 500G of vehicle trajectory data and microwave data in Beijing.

B. DISTRIBUTION OF GRID AND GRID RANK

The Equation (9) and Equation (10) are the coordinates of the map. The first step is to find the coordinates of the starting point in the lower-left corner of the map. The vertices with XCoord ($x_i < x_{others}$) and YCoord ($y_i < y_{others}$) being the smallest, at the same time need to be screened out. However, there may also be such a point with the smallest X-axis coordinates ($x_j < x_{others}$) in the whole road network data, but its Y-axis ($y_j > y_{others}$) is larger or vice versa. To contain all intersections and these must be identified.

$$Pointrc := |(XCoord < x_i), (YCoord < y_j)| \quad (9)$$

$$Pointlc := |(XCoord > x_k), (YCoord < y_l)| \quad (10)$$

Pointrc is defined as the coordinate point of the rectangle in the lower-right corner of the map, which must be smaller than the coordinate value of all road network data. *Pointlc* is defined as the coordinates of the upper Left Corner of the rectangle, which must be larger than the coordinates of all road network data. Next, the coordinates of these two points can be calculated by searching the dataset. Fig 8 shows the distribution result of the whole Beijing city with $n_g = 18$.

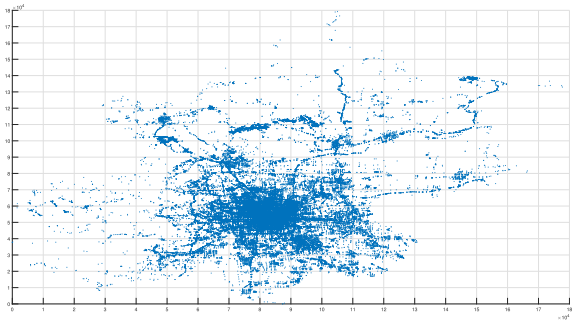


FIGURE 8. Distribution of intersections with 18 × 18 grid.

The following section will describes that how to progress with our grid program ranks all the partitioned grids. Table 2 shows the gird after rasterization status, $n_g \times n_g$ is grid quantity, *MV* refers to the maximum number of vertices in the grid after different grid. *AIG* stands for the average number of intersections in one grid, *ARG* stands for the average number of roads in one grid, and *Variance* represents the distribution density of nodes.

According to the Equation (11), as follow, N is grid number, v_i is the number of vertices in the ith grid, $\overline{v}$ is mean value the smaller the variance, the more well-distributed the segmentation results. *Variance* is a measure of dispersion in probability theory and statistical variance to measure a random variable or a set of data. Probability variance is used to measure the deviation between a random variable

TABLE 2. Distribution of vertices in n_g grid map.

$GridMap$	MV	AIG	ARG	$Variance$
n_g=18	11095	4859	10332	951.48
n_g=36	3413	1252	4581	263.71
n_g=72	1057	356	792	71.42
n_g=144	436	78	159	19.67
n_g=288	301	23	53	5.74
n_g=576	107	8	19	1.73

and its mathematical expectations, known as the mean. So, the smaller the *Variance*, the better the rasterization. The result of Table 2 shows that when all vertices are divided into $n_g = 18$ grids, the maximum number of vertices in the grids is Max $= 11095$, which means that the rank of the grid will be higher than others. It is worth noting here that when $n_g = 18$, the search strategy of GBD and Bidirectional Dijkstra algorithm is the same, because the grid range of $n_g = 18$ is large, so the path planning results in the city are the same.

$$Variance^2 = \sum_{i=1}^{N} (v_i - \overline{v})^2 / N \quad (11)$$

$n_g = 36$ and $n_g = 72$ have the same problem with $n_g = 18$ grids, although the maximum number of vertices is different. Then from $n_g = 576$ grids shows the vertices number equal to 107. In addition, from *AIG* can also find that the number of intersections contained in the 18×18 grid has reached an average of 4859 and from *ARG* can shows the number of roads has average 10332. In $n_g = 576$ has only average 8 intersections and 19 roads. The distribution of these grids will affect the efficiency of the GBD algorithm, which given detail in this section's next subsection. Fig 9 have show intersection distribution within the fifth ring after grid into $n_g = 18, 36, 72, 144, 288, 576$.

In summary, The number of grids will use $n_g = 18, 36, 72, 144, 288$ and 576 for GPR algorithm. First, the GPR method deletes the empty grids from the graph, leaves the grids with vertices, and begins to calculate the number of vertices in each grid. It is noted that the algorithm calculates the number of grids by three types of vertices. One is to connect all the other vertices in this grid, these vertices called a internal-connection. The other type is leading to other grids, and this kind of vertex called an external connection. The last one is that Points are partially outbound to connect external vertices, which called a hybrid-connection.

At the same time, the algorithm also need to consider the number of lanes L on the road. In urban roads, usually no more than one-way five lanes, so the value of L is an integer in the range from 1 to 5. When evacuating that the more lanes the road has, the more traffic flow can pass in the same time (due to assumption that all vehicles evacuation in an orderly manner, there is no overtaking or traffic jam, because if when traffic jam happen that our GMEPP framework will redeployed).

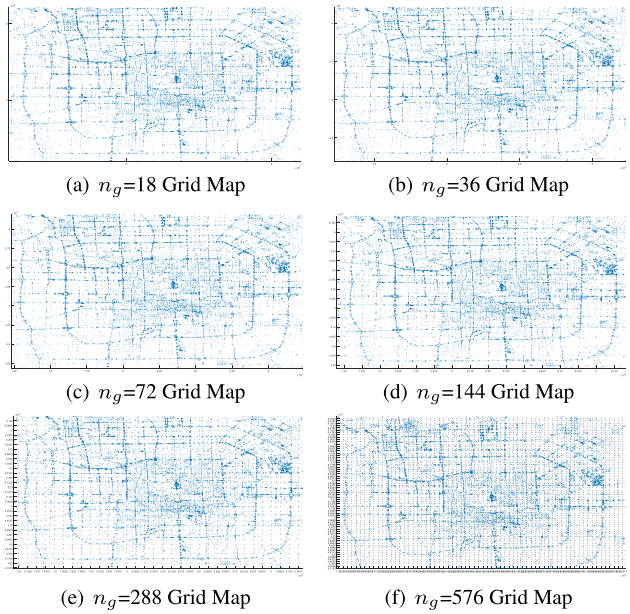


FIGURE 9. **Differently distributed grid maps within the fifth ring. (a) $n_g = 18$. (b) $n_g = 36$. (c) $n_g = 72$. (d) $n_g = 144$. (e) $n_g = 288$. (f) $n_g = 576$.**

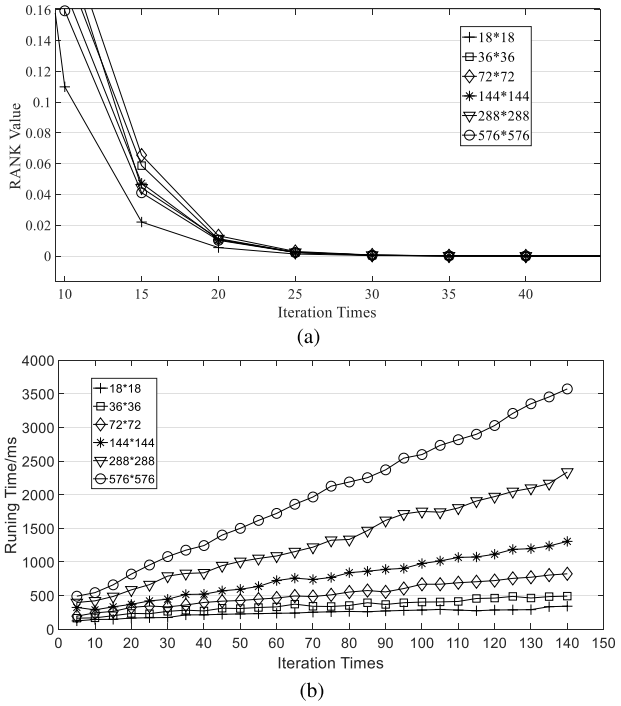


FIGURE 10. **(a) The Convergence Results of Grid after Iterative Computation. (b) With the increase of the number of iterations under different grid conditions, the running time of GPR is also continuously increasing.**

According to Equation (3), the value of R is calculated first. The initial value of R is 1 to each grid with vertices, then initializes each grid and begins to calculate the weights of $Out - degree$ edges within the grid. If the number of $Out - degree$ vertices in a grid is 3, their edges weights are L. Flat and the weights of each edge are assigned to the connected grids; each grid with external-connection repeats this calculation until the final RANK converges. Table 3 shows $n_g = 144$ and $n_g = 288$ results of the RANK with grids respectively. The theory of rank is the same and does not affect the results of the path planning, so the rank results of other grid distributions will no longer be displayed.

TABLE 3. **Partial results of descending order of the grid values with $n_g = 144$ and $n_g = 288$.**

144 $GridID$	$RANK$	288 $GridID$	$RANK$
14-46	0.02403406605964532	28-91	0.00063961350785751
20-48	0.01689090111204018	57-103	0.00035618516562940
19-47	0.01265170851251942	138-61	0.00031077176150319
39-91	0.00950338786211769	160-35	0.00030765424485485
59-86	0.00879868442408170	161-109	0.00030728136329326
15-46	0.00874443282679572	136-126	0.00030685079492260
29-52	0.00846635288234325	171-106	0.00028400636699139
21-49	0.00846595871111033	103-193	0.00028075261107912
18-48	0.00832210771747266	101-117	0.00027928511341343
52-97	0.00827585868949408	116-142	0.00027669512986079
39-23	0.00819778045965766	36-95	0.00027370420482802

From Table 3 shows the result of grid sorting, with the largest grid at the top and the smallest grid at the bottom. When the vehicle needs emergency path planning in the grid, the algorithm first searches the grid with larger RANK value in the adjacent grid for navigation, because the commuting ability of the grid is the GPR = $MaxRank$ with the adjacent grid. It should be noted that GPR updates the rank value of the grid when doing path planning.

According to Fig. 10 (a) shows that the GPR algorithm has reached a smooth state between 30 and 35 iterations. The value of grid sorting designed by us is generally smaller more than one vertices are pointing to the same grid in each cell, so it is essentially different from the traditional PageRank method. Fig. 10 (b) shows that as the number of iterations increases, the more grids, the longer the GPR running time.

C. GRID BIDIRECTIONAL DIJKSTRA WITH CONTRACTION HIERARCHIES IN DIFFERENT GRID DISTRIBUTIONS

The following experimental will tests the efficiency of emergency path planning methods based on different grid distributions. The experiment uses three paths to compare the GBD calculation effect based on different distributions of the grid. The length of the path is $P1 = 6.5$km, $p2 = 12.6$km and $p3 = 21.3$km in which all three paths are within the fifth ring, The reason why the area inside the fifth ring road is selected for testing is that the roads outside the fifth ring road are relatively sparse and the road congestion is not obvious. In Table 4 the NGS represents the GBD searched the number of grids, NVS represents the GBD searched the number of vertices, and $Runtime$ is standing the effectiveness of the algorithm. Experimental choose paths of different lengths for comparison because of the number of grids and points that GBD search under different grids will be different. Search the number of grids and points directly affect the efficiency of GBD.

TABLE 4. The effect of GBD based different distributions with three paths.

$GridMap$	$p1 = 6.5km$			$p2 = 12.6km$			$p3 = 21.3km$		
	NGS	NVS	$Runtime(ms)$	NGS	NVS	$Runtime(ms)$	NGS	NVS	$Runtime(ms)$
$n_g = 18$	8	2358	42.6	8	4581	73.8	8	11572	179.5
$n_g = 36$	15	1904	40.3	15	3358	63.6	22	7823	125.7
$n_g = 72$	22	1128	32.5	29	2874	48.2	50	5629	93.8
$n_g = 144$	43	951	21.8	50	1927	36.4	99	3041	72.1
$n_g = 288$	76	783	18.2	120	1029	26.8	213	2508	60.9
$n_g = 576$	131	374	26.4	245	712	38.5	857	2196	82.3

According to Table 4 obtains the search efficiency of GBD at different distances. Along with the number of grids increases, the number of GBD search vertices continues to decrease, but the number of search grids continues to increase. Therefore, the number of search vertices and the number of grids will directly affect the search efficiency. Below content will further discuss the four algorithms under different grid distributions.

As can be seen from the Table 4, as n_g value gets larger, the time of the algorithm gets smaller, but when $n_g = 576$, the running time of the algorithm increases. The reason is that when n_g value keeps growing, the range of the grid is decreasing, that is, the number of vertices contained in each grid is decreasing, so GBD only searches for intersections in the grid when searching each grid, so the time of the algorithm is decreasing. However, the number of grids when $n_g = 576$ is too large, and GBD needs to loosen a large number of grids when searching for them, so the running time of the algorithm should be greater than $n_g = 288$.

Fig. 11 shows the result is a comparison of the efficiency of the four algorithms in the case of different grid distribution. Experimental uses the distance of 2km as the reference for the abscissa of the path search, and the ordinate is the runtime of the algorithm. When searching in the graph has more vertices, the longer the search time. From Fig. 9 (a) and Fig. 11 (a) can find $n_g = 18$ distribution has long runtime with four algorithms, the main reason is that each grid contains many vertices. $n_g = 36$, $n_g = 72$ and $n_g = 144$ have become better because the grid size is getting smaller in Fig. 9 (b), Fig. 9 (c) and Fig. 9 (d), respectively, it affects the number of vertices in each grid.

When GBD searches for road network vertices, it needs to relax a large number of vertices because it follows the breadth first traversal condition of Bidirectional Dijkstra algorithm. However, our GBD divides the whole road network vertex into several grids. Moreover, the finer the grid is divided, the fewer vertices it contains. The advantage of GBD is that it relaxes the vertices in a grid at a time. For other grids, it is necessary to first judge the properties of the grid and then search the vertices in the grid. This reduces the time cost of the algorithm.

From Fig. 11 (a), Fig. 11 (b), Fig. 11 (c) and Fig. 11 (d) show the efficiency of the four algorithms in different distribution has little impact. Compared with other cases, Fig. 11 (e) shows the effect of $n_g = 288$ distribution is

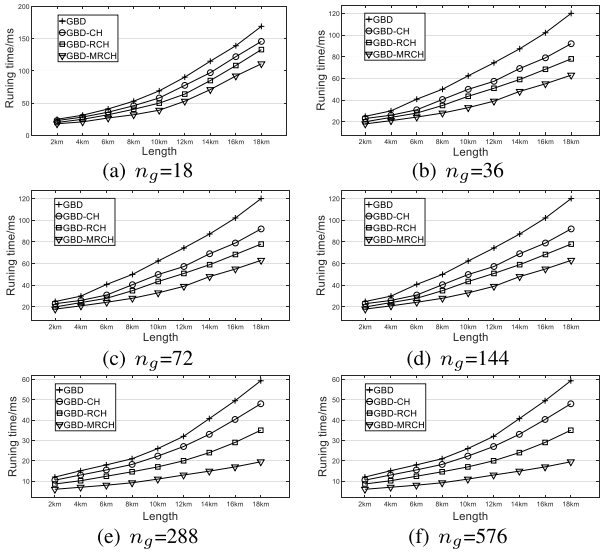


FIGURE 11. (a) Results of the average runtime of different path planning methods in $n_g = 18$ grid with different length paths. (b) Results of the average runtime of different path planning methods in $n_g = 36$ grid with different length paths. (c) Results of the average runtime of different path planning methods in $n_g = 72$ grid with different length paths. (d) Results of the average runtime of different path planning methods in $n_g = 144$ grid with different length paths. (e) Results of the average runtime of different path planning methods in $n_g = 288$ grid with different length paths. (f) Results of the average runtime of different path planning methods in $n_g = 576$ grid with different length paths.

better. Based on the distribution of $n_g = 576$ in Fig. 11 (f), the calculation time of the algorithm becomes longer because the number of vertices in each grid becomes less when the number of grids becomes larger, and more search time is required. Although the average number of vertices per grid in $n_g = 576$ is less than $n_g = 288$, the number of grids is greater than $n_g = 288$. GBD-MRCH not only searches vertices, but also compares the rank values between grids. It is emphasis that our compression algorithm is based on the intersection type of vertex compression, so it has nothing to do with the road density.

Next, experimental will combines other parameters to analyze the distribution of the grid. The result of *Variance* can indicate that the vertices can be evenly distributed in each grid. The smaller the value, the better the result. Therefore, experimental use *Variance* and the runtime of GBD to evaluate which distribution as the grid result of the GMEPP framework. From Table 4 and Fig. 9 (a) can find if an emergency

area occurs in the $n_g = 18$ grid map that our GMEPP will not be able to effectively guide the vehicle to the area, because the number of vertices searched in each grid is too large, and runtime is greater than other distributions. And when $n_g = 18$, each grid spans a size of about 10km, so when the grid is used as a $US - GRID$, it does not meet the actual situation. The vehicle needs to plan the emergency path within an area of 100 square kilometers.

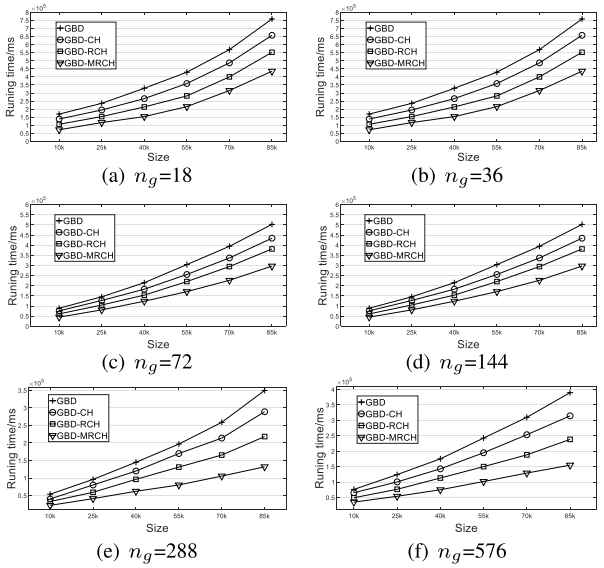


FIGURE 12. (a) Results of the average runtime of different path planning methods in $n_g = 18$ grid with different query times. (b) Results of the average runtime of different path planning methods in $n_g = 36$ grid with different query times. (c) Results of the average runtime of different path planning methods in $n_g = 72$ grid with different query times. (d) Results of the average runtime of different path planning methods in $n_g = 144$ grid with different query times. (e) Results of the average runtime of different path planning methods in $n_g = 288$ grid with different query times. (f) Results of the average runtime of different path planning methods in $n_g = 576$ grid with different query times.

The same problem exists in $n_g = 36, 72, 144$. In $n_g = 576$, although fewer vertices are searched in the grid, the number of search grids is too large, so its efficiency is not good. However, in $n_g = 288$ grid distribution is better than other distributions. And from Table 2, it can be seen that the number of intersections in the grid of $n_g = 288$ is moderate. The *Variance* has a small value, indicating that the intersections in the grid have a better uniform distribution. Although the grid sorting time of $n_g = 288$ is greater than $n_g = 144$, 72, 36, and 18, it takes no more than 1 second in its one-time path planning, because GPR has converged by 35 iterations.

The above experiments show that the four algorithms are distributed in different grids. Under different grid distributions, the GBD-MRCH algorithm runs better than other algorithms, especially the GBD-MRCH algorithm has more stable operating efficiency when $n_g = 288$. It should be emphasized here that the compression algorithm is only run once before the framework is run, not every time before path planning. Below content will further analyze the results of the graph compression algorithm. Fig. 12 shows that compares the four algorithms in different query text size. According to the size of the adjusted text, the algorithm efficiency of the algorithm when querying multiple targets can be obtained.

Fig. 12 also compare the operational efficiency of the four algorithms with multi-paths. Because the GBD algorithm is improved on the traditional Bidirectional Dijkstra algorithm, its content is to increase the grid search and the grid search restrictions, so the GBD search mode is same as Bidirectional Dijkstra. GBD-CH is a traditional CH graph compression method added on the basis of GBD, which is better than GBD. GBD-RCH and GBD-MRCH are graph compression algorithms based on intersection type. The operation effect is better than the first two algorithms. From Fig. 12 (a) shows the effect of the four algorithms on different grids, but the acceleration effect of GBD-RCH and GBD-MRCH is not obvious. Fig. 12 (b),Fig. 12 (c) and Fig. 12 (d) show runtime of the algorithms are better than $n_g = 18$, but the acceleration effect is still not significant. However, in Fig. 12 (e) shows that the acceleration effect of GBD-RCH and GBD-MRCH is significant improvement. The main reason is that the average number of nodes contained in each grid in $n_g = 288$ is 23 (in Table 3). In other words, the number of nodes relaxed in a grid is less than other distribution. However, $n_g = 576$ in Fig. 12 (f) is lower efficiency of the algorithm than $n_g = 288$, because $n_g = 576$ has too many grids.

TABLE 5. Comparison of the effects of different contraction methods to finding 500 and 1000 shortest paths with $n_g = 288$ grid.

$Method$	$Vertex$	$Edge$	CV	CE	$500ART(ms)$	$1000ART(ms)$
GBD	83884	222778	0	0	56.9	56.4
GBD-CH	83884	222778	19395	38596	45.7	48.4
GBD-RCH	83884	222778	9619	19848	32.3	34.1
GBD-MRCH	83884	222778	14455	34185	18.4	18.6

From Table 5 can obtains the average query time results of the four query methods. The experiment conducts 500 and 1000 pairs of starting and ending point queries based on the Beijing road network (the purpose of choosing many pairs of query paths is to ensure effectiveness of our GMEPP for multiple path planning in the emergency areas). The results of the algorithms' average running time show that multiple queries have little effect on the algorithms. The above four methods use the same starting point pair. *CV* and *CE* represented have been contracted the number of vertices and edges in the road network, respectively. *ART* stands for the average runtime of the algorithm. Although CH algorithm compressed more vertices than RCH and MRCH proposed by us, the acceleration effect proposed by RCH and MRCH algorithm depends on how to select the contraction factor. Traditional graph compression is usually performed based on the number of vertices in and out and the shortest path between vertices. Thus, more contracted vertices are not necessarily related to faster path planning. Moreover, in the original road network query that it is necessary to restore the shortcut edges. But our method does not directly reduce the shortcut edges, only in the source and destination to return

shortcuts. The RCH and MRCH creates an *IndexTable* for all the contraction vertices, and storing the order of vertex compression and corresponding vertex pairs in the table. In each path query, MRCH will judge whether the input is starting and ending point compression. If it does, that will start to query the corresponding vertex pairs in *IndexTable* recursively.

VIII. DISCUSSION

A. CONCLUSION

In this paper, a GMEPP framework based on grid network described, it can evacuate the vehicles in the emergency areas to the grid with a better commuting ability (rank value is greater than the grid nearby) and start from these grids to the final destination. The purpose of grid-based on the road network is to divide the whole road network, which is an entity but belongs to logical isolation in our GRN model. In other words, detour the unsafe grid. The vehicles in an unsafe space need to be evacuated to safe space immediately. Thus, our method will provide a reasonable evacuation plan for these vehicles. GMEPP framework not only consider the principle of proximity, which may cause secondary congestion to the vehicles and drivers, because the commuting ability of evacuation areas may be low. In the process of vehicle evacuation, that road congestion will occur with a high probability so that this phenomenon will affect the evacuation efficiency of vehicles. Therefore, the GMEPP framework to avoid this situation so that to evacuate vehicles more effectively.

The sizes of all the grids are arranged in descending order after sorting the grids. Ordinary Bidirectional Dijkstra algorithm is based on local optimum and searches for global optimum step by step. In our GBD method, the first consideration is to effectively guide the vehicle to the safe area in a short time, and then to search and navigate the secondary path from the safe area to the destination. Evacuation from $US-GRID$ to $S-GRID$. For vehicles in unsafe grids, priority evacuation to safer areas is very important. After grid and sort, vehicles near $US-GRID$ are recommended to those grids which have a strong commuting ability in the shortest time, and they are evacuated to the grids to navigate a globally optimal way to the destination. This paper introduced two kinds of GBD-RCH and GBD-MRCH accelerated query algorithms based on the characteristics of grid and road network. Experiments show that the grid division effect when $n_g = 288$ is suitable for GBD algorithm, and the efficiency of GBD-MRCH is better than GBD-RCH and GBD-CH.

B. FUTURE WORK

In future research work, Our main focus on two kinds of factors to add to the grid based on the road network model. At present, the weight of the road network is about the speed of the road network, but there are still other factors that will lead to the change of the weight of the road network. Conditional factors considered in current research are still insufficient. The single weight is the basis of the shortest path query, but the road network query and emergency path query under the situation condition are a few. At present, the research of adding intersection information to the acceleration algorithm is rare. Thus, based on the above two points will be our next research focus.

REFERENCES

[1] C.-H. Chang, R.-Y. Wu, and J.-L. Lin, "A dual route planning for unknown scaled emergency management," *J. Chin. Inst. Ind. Engineers*, vol. 26, no. 3, pp. 195–204, Jan. 2009.
[2] D. Zou, S. Niu, S. Chen, B. Su, X. Cheng, J. Liu, Y. Liu, and Y. Li, "A smart city used low-latency seamless positioning system based on inverse global navigation satellite system technology," *Int. J. Distrib. Sensor Netw.*, vol. 15, no. 9, Sep. 2019, Art. no. 155014771987381.
[3] X. Liu, K. N. Khan, Q. Farooq, Y. Hao, and M. S. Arshad, "Obstacle avoidance through gesture recognition: Business advancement potential in robot navigation socio-technology," *Robotica*, vol. 37, no. 10, pp. 1663–1676, Oct. 2019.
[4] T. Song, N. Capurso, X. Cheng, J. Yu, B. Chen, and W. Zhao, "Enhancing GPS with lane-level navigation to facilitate highway driving," *IEEE Trans. Veh. Technol.*, vol. 66, no. 6, pp. 4579–4591, Jun. 2017.
[5] J. E. Almeida, R. J. Rossetti, and A. L. Coelho, "Crowd simulation modeling applied to emergency and evacuation simulations using multi-agent systems," in *Proc. Comput. Sci.*, Mar. 2013, pp. 1–12.
[6] J. Zhu, S. Liu, and S. Ghosh, "Model and algorithm of routes planning for emergency relief distribution in disaster management with disaster information update," *J. Combinat. Optim.*, vol. 38, no. 1, pp. 208–223, Jul. 2019.
[7] M. Milenković and D. Kekić, "Using GIS in emergency management," in *Proc. Sinteza-Int. Sci. Conf. ICT E-Bus. Rel. Res.*, 2016.
[8] I. Lochhead and N. Hedley, "Mixed reality emergency management: Bringing virtual evacuation simulations into real-world built environments," *Int. J. Digit. Earth*, vol. 12, no. 2, pp. 190–208, Feb. 2019.
[9] M. Khalid and U. Yusof, "Immune-based approach for optimizing emergency route planning problem: Application to case studies," *Int. J. Res. Surv.*, vol. 9, pp. 3291–3298, 2015.
[10] Y.-C. Chiu, H. Zheng, J. Villalobos, and B. Gautam, "Modeling no-notice mass evacuation using a dynamic traffic flow optimization model," *IIE Trans.*, vol. 39, no. 1, pp. 83–94, Jan. 2007.
[11] P. Wang, P. B. Luh, S.-C. Chang, and K. L. Marsh, "Efficient optimization of building emergency evacuation considering social bond of evacuees," in *Proc. IEEE Int. Conf. Autom. Sci. Eng.*, Aug. 2009, pp. 250–255.
[12] T. Yamada, "A network flow approach to a city emergency evacuation planning," *Int. J. Syst. Sci.*, vol. 27, no. 10, pp. 931–936, Oct. 1996.
[13] M. Cisek and M. Kapalka, "The use of fine–coarse network model for simulating building evacuation with information system," in *Proc. Pedestrian Evacuation Dyn.*, 2011, pp. 481–491.
[14] M. Ben-Akiva, D. Mcfadden, K. Train, J. Walker, C. Bhat, and M. Bierlaire, "Development of a deployable real-time dynamic traffic assignment system: DynaMIT and DynaMIT-P User's guide," Intell. Transp. Syst. Program, Massachusetts Inst. Technol., Tech. Rep., 2002
[15] Y. Yao, Z. Peng, and B. Xiao, "Parallel hyper-heuristic algorithm for multi-objective route planning in a smart city," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 10307–10318, Nov. 2018.
[16] K. S. Yang and S. Shekhar, "Evacuation route planning," in *Proc. Spatial Netw. Big Databases*, 2017, pp. 57–72.
[17] R. Geisberger, P. Sanders, D. Schultes, and D. Delling, "Contraction hierarchies: Faster and simpler hierarchical routing in road networks," in *Proc. Exp. Algorithms*, vol. 5038, 2008, pp. 319–333.
[18] L. Page, "The PageRank citation ranking: Bringing order to the Web," Stanford InfoLab, Stanford, CA, USA, Tech. Rep., Jan. 1999, pp. 1–14.
[19] A. Sedeño-Noda and M. Colebrook, "A biobjective Dijkstra algorithm," *Eur. J. Oper. Res.*, vol. 276, no. 1, pp. 106–118, Jul. 2019.
[20] Y. Tao, C. Sheng, and J. Pei, "On k-skip shortest paths," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2011, pp. 421–432.
[21] X. Yang, X. Yang, Q. Wang, Y. Kang, and F. Pan, "Guide optimization in pedestrian emergency evacuation," *Appl. Math. Comput.*, vol. 365, Jan. 2020, Art. no. 124711.
[22] A. M. Ibrahim, I. Venkat, and P. D. Wilde, "Uncertainty in a spatial evacuation model," *Phys. A, Stat. Mech. Appl.*, vol. 479, pp. 485–497, Aug. 2017.

[23] I. Venkat, K. Subramanian, and G. Khader, "Intelligent evacuation management systems: A review," *ACM Trans. Intell. Syst. Technol.*, vol. 7, no. 3, pp. 36:1–36:27, Apr. 2016.
[24] A. M. Ibrahim, I. Venkat, and P. De Wilde, "The impact of potential crowd behaviours on emergency evacuation: An evolutionary game-theoretic approach," *J. Artif. Soc. Social Simul.*, vol. 22, Jan. 2019, doi: 10.18564/jasss.3837.
[25] M. A. Dulebenets, O. F. Abioye, E. E. Ozguven, R. Moses, W. R. Boot, and T. Sando, "Development of statistical models for improving efficiency of emergency evacuation in areas with vulnerable population," *Rel. Eng. Syst. Saf.*, vol. 182, pp. 233–249, Feb. 2019.
[26] H. Jiang and Y. Dong, "Safety evaluation for evacuation system under serious emergency from interdependent network perspective," *Int. J. Wireless Mobile Comput.*, vol. 14, no. 3, p. 250, 2018.
[27] A. Elbery, H. S. Hassanein, N. Zorba, and H. A. Rakha, "VANET-based smart navigation for emergency evacuation and special events," in *Proc. IEEE 24th Int. Workshop Comput. Aided Modeling Design Commun. Links Netw. (CAMAD)*, Limassol, Cyprus, Sep. 2019, pp. 1–6.
[28] R. Xie, Z. Yang, Y. Niu, and Y. Zhang, "Simulation of small social group behaviors in emergency evacuation," in *Proc. 29th Int. Conf. Comput. Animation Social Agents*, 2016, pp. 71–77.
[29] S. Sharma, "Simulation and modeling of group behavior during emergency evacuation," in *Proc. IEEE Symp. Intell. Agents*, Mar. 2009, pp. 122–127.
[30] C. Yang, Z. Cai, F. Xue, T. Li, and Z. Ding, "Semantic trajectory based behavior generation for groups identification," *TIIS*, vol. 12, no. 12, pp. 5782–5799, 2018.
[31] M. S. Tuin, M. Weerdt, and G. V. Batz, "Route planning with breaks and truck driving bans using time-dependent Contraction Hierarchies," in *Proc. ICAPS*, 2018, pp. 356–365.
[32] C. Nowak, F. Hahne, and K. Ambrosi, "Contraction Hierarchies with turn restrictions," in *Proc. Annu. Conf. German Oper. Res. Soc.*, 2012, pp. 569–575.
[33] S. Li, A. Zhan, X. Wu, P. Yang, and G. Chen, "Efficient emergency rescue navigation with wireless sensor networks," *J. Inf. Sci. Eng.*, vol. 27, no. 1, pp. 51–64, Jan. 2011.
[34] Y. Sun, "A reliability-based approach of fastest routes planning in dynamic traffic network under emergency management situation," *Int. J. Comput. Intell. Syst.*, vol. 4, no. 6, pp. 1224–1236, Dec. 2011.
[35] S. Yamabe, F. Hasegawa, T. Suzuki, K. Kamata, K. Hatakeyama, and O. Ito, "Driver behavior response to information presentation based on the emergency evacuation procedure of the great east Japan earthquake," *Int. J. Intell. Transp. Syst. Res.*, vol. 17, no. 3, pp. 223–231, Sep. 2019.
[36] X. Mao, S. Tang, X. Y. Li, and M. Gu, "MENs: Multi-user emergency navigation system using," *Ad Hoc Sensor Wireless Netw.*, vol. 12, nos. 1–2, pp. 23–53, 2011.
[37] E. Gelenbe and H. Bi, "Emergency navigation without an infrastructure," *Sensors*, vol. 14, no. 8, pp. 15142–15162, Aug. 2014.
[38] C. Wang, H. Lin, R. Zhang, and H. Jiang, "SEND: A situation-aware emergency navigation algorithm with sensor networks," *IEEE Trans. Mobile Comput.*, vol. 16, no. 4, pp. 1149–1162, Apr. 2017.
[39] Y. Shen, J. Lee, H. Jeong, J. Jeong, E. Lee, and D. H. C. Du, "SAINT+: Self-adaptive interactive navigation Tool+ for emergency service delivery optimization," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 4, pp. 1038–1053, Apr. 2018.

BOWEN YANG received the M.S. degree from the North China University of Water Resources and Electric Power, Zhengzhou, China, in 2018. He is currently pursuing the Ph.D. degree with the College of Computer Science, Beijing University of Technology, Beijing. His research interests include intelligent transportation systems, data mining, and path planning.

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 Institute of Software, Chinese Academy of Sciences, and with the Beijing University of Technology. Before joining the Beijing University of Technology in 2014, he was with the Institute of Scientific and Technical Information, Ministry of Communications of China from 1989 to 1993, SINOCHEM from 1996 to 1999, FernUniversität in Hagen, Germany, from 2002 to 2004, and the Institute of Software, Chinese Academy of Sciences, from 2004 to 2014. He owns five invention patents, and has published three books and about 120 articles in academic journals and conferences. His research interests include database systems, mobile and spatial-temporal data management, intelligent transportation systems, sensor data management, and information retrieval.

LEI YUAN received the bachelor's degree in engineering from the Beijing University of Technology, in 2018, where he is currently pursuing the master's degree with the School of Computer Science. His research interests include vehicle emergency navigation, data mining, and deep learning.

JIN YAN received the bachelor's degree of applied mathematics/statics from the University of Waterloo, Canada, in 2017. She is currently pursuing the master's degree in computer science with the Institute of Software, Chinese Academy of Sciences, supported by UCAS Full Scholarship. As a Research Trainee, she took part in the City of Toronto Accessibility Project and data analysis for the iDAPT Winter Laboratory, Toronto Rehab Centre, University Health Network, from 2017 to 2019. Her research interests include data analysis, data mining, emergency systems, and data management.

LIMIN GUO was born in 1984. She is currently a Lecturer with the Beijing University of Technology. She is also a Lecturer with the Beijing University of Technology. Her research interests include database research and implementation, spatial-temporal data mining, storage, query and analysis of big data, and so on.

ZHI CAI received the M.Sc. degree from the School of Computer Science, The 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.