IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 14, 2020, accepted January 4, 2021, date of publication January 20, 2021, date of current version February 3, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3053169

A New Algorithm Based on Dijkstra for Vehicle Path Planning Considering Intersection Attribute

DAN-DAN ZHU AND JUN-QING SUN

Tianjin Key Laboratory of Intelligence Computing and Novel Software Technology, Tianjin University of Technology, Tianjin 300384, China

Corresponding author: Dan-Dan Zhu (ddzhu0121@163.com)

This work was supported in part by the National Nature Science Foundation under Grant 71501141, and in part by the Science and Technology Ministry's Special Foundation of Developing Important and Large Equipment under Grant 2014YQ120351.

ABSTRACT Vehicle path planning is a key issue for car navigation systems. When path planning, considering the time spent at intersections is more in line with the actual situation, so it is of practical significance to study the path planning problem take account intersection attributes. In this article, we study the problem in a deterministic network, taking the minimization of travel time from the origin to the destination as the optimization goal. For this purpose, we construct a mathematical model for the problem. This paper proposes a reverse labeling Dijkstra algorithm (RLDA) based on traditional Dijkstra algorithm to solve the problem, it is proved that the correctness of the RLDA algorithm theoretically, and analyze that the RLDA algorithm has a lower polynomial time complexity. Finally, we selected the actual road network as the simulation experiment object to verify the effectiveness of the algorithm searching for the optimal path. And select 10 groups of networks of different sizes and conduct extensive experiments to compare the convergence efficiency and calculation speed between RLDA and PSO, GA, ACO, NNA, OPABRL. The statistical results show that the convergence rate of the RLDA algorithm is better than that of ACO, NNA, and GA. When the number of network nodes is less than 350, the algorithm has the smallest running time.

INDEX TERMS Path planning, intersection attribute, reverse labeling Dijkstra algorithm, deterministic network.

I. INTRODUCTION

Urban traffic is closely related to people's production and life, with the progress of society, the pressure on transportation is increasing, and the problems of traffic jams, traffic accidents and environmental pollution caused by traffic are also becoming more and more serious. In order to solve the problems faced by the current transportation, the Intelligent Transportation System (ITS) came into being. Intelligent transportation system effectively integrates advanced information technology, data communication transmission technology, electronic control technology, sensor technology and computer processing technology to the entire transportation system, thus establishing a real-time, accurate, and efficient comprehensive transportation management system [1]. Intelligent transportation systems use modern science and technology to establish intelligent connections between roads, vehicles, and drivers and passengers. Greatly improve transportation efficiency, fully guarantee traffic safety, improve environmental quality, and increase energy efficiency. It is currently internationally recognized as the best way to solve urban and highway traffic congestion, improve driving safety, increase operating efficiency, and reduce air pollution. It is also a frontier subject in the field of transportation research around the world [2], [3].

The concept of intelligent transportation system was put forward by the American Intelligent Transportation Society in the 1990s, and it was vigorously promoted in countries all over the world. However, the germination of intelligent transportation system thought can be traced back to the 1960s [4]. The static route guidance and computer traffic control technology that appeared in the 1960s can be regarded as embryonic forms. The importance at that time has not been highly valued. After entering the 1990s, it has developed at an alarming speed. In order to solve the common traffic problems, many developed countries have rushed to invest a lot of funds and manpower to conduct large-scale research experiments on intelligent road transportation [5]. Japan began to pay attention to ITS in the 1970s, and the Japanese government has invested a lot of funds and policies in the field of ITS

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

in order to form an industry to promote Japan's economic development. In the past few years, nearly ten thousand in-car navigation systems have been applied in the market. ITS applications in Japan are mainly in traffic information provision, electronic toll collection, public transportation, commercial vehicle management, and emergency vehicle priority [6]. The progress of ITS application in Europe lies between Japan and the United States. Due to the decentralized investment of European governments and the inconsistency of the needs of various countries, it is difficult to establish a unified traffic information service system throughout Europe. However, in the development of advanced travel information systems (ATIS), advanced vehicle control systems (AVCS), advanced commercial vehicle operating systems (ACVOS), and advanced electronic toll collection systems (AETCS), the prospects are very attractive [7].

The Dynamic Route Guidance System (DRGS) is an important branch and one of the key technologies of the ITS [8]. It uses modern technologies such as Global Positioning System (GPS), computers and communications to obtain a large number of timely and accurate traffic network data, and plans the corresponding optimal route for drivers to choose. The route optimization algorithm is the key technology of DRGS. The optimization algorithm can improve the efficiency of the route guidance system to meet the real-time traffic environment. In the massive and rapidly changing traffic information, whether the dynamic guidance can be completed in time after the traffic information is updated in the traffic system, and whether the effective information can be quickly extracted and efficiently fed back to the pedestrian is crucial [9]. Therefore, optimization algorithms become the key to solving the problem.

In the existing research on the path planning of the car navigation, usually only consider the time spent on the arc, and the time spent at the node is usually attributed to the corresponding arc or ignored. In the urban road network, there are a large number of intersections and road sections. The length of the road section between two intersections is usually short. In addition, due to the influence of traffic lights and traffic rules, the time spent at the intersection may be longer than pass through a road segment, the time it takes for vehicles to go straight, turn right, turn left or U-turn through the intersection also varies, so the time spent at the intersection can no longer be ignored. Therefore, when planning a route for vehicles in urban traffic, it is necessary to consider both the time spent passing through the road section and the intersection [27].

In this paper, we focus on the path planning problem of car navigation systems considering intersection attributes, taking the minimum total travel time as the optimization goal, a reverse labeling dijkstra algorithm (RLDA) is proposed to solve this problem. The rest of this paper is organized as follows: In section II, we will provide a survey of the relevant literature. In Section III, we will give a detailed description of the issues discussed, the mathematical models. And present our optimization algorithm in Section IV. Section V will perform a large number of experiments on the algorithm (RLDA) proposed in this paper and other five commonly used algorithms. Section VI will summarize the work of this paper.

II. RELATED WORKS

The optimization algorithms studied and applied in DRGS mainly include traditional graph theory shortest path algorithm and intelligent optimization algorithm [10]. The former includes: Dijkstra [11], [12], A* algorithm [13], Floyd algorithm [14], and hierarchical heuristic search algorithm, etc.; the latter includes: genetic algorithm, ant colony algorithm, neural network algorithm, particle swarm optimization algorithm and simulated annealing algorithm, machine learning algorithm, etc..

The shortest path problem (SP) is a classic algorithm problem in graph theory research, which aims to find the shortest path between two nodes in a graph (composed of nodes and paths). Many practical problems can be transformed into shortest path problems [15]. Dijkstra algorithm was first proposed in 1959 to solve the shortest path problem. Dijkstra algorithm is a common algorithm for solving single source shortest path problems. It uses an adjacency matrix storage structure to store the relationship between vertices and vertices, and traverses all vertices one by one. The time complexity of the algorithm is $O\left(n^2\right)$ [16]. In order to solve the problem of poor availability of the traditional Dijkstra algorithm in expressway emergency evacuation planning paths, Liu *et al.* considered the characteristics of nodes and sections in the expressway network with traffic capacity and conditional restrictions, and improved the dijkstra algorithm [17]. A* algorithm is also an earlier traditional shortest path optimization algorithm, which was first put forward by Raphael *et al.* in 1968 [18]. The A* algorithm is the most effective direct search method for solving the shortest path in a static road network, it is also a common heuristic algorithm for many other problems. The A* algorithm is a complete algorithm, its advantage lies in its ability to capture the solution completely, but its disadvantage is that the algorithm is more complex. Therefore, the improved A* algorithm has been extensively studied. For the robot path planning problem, Song *et al.* proposed a hybrid algorithm of particle swarm optimization algorithm (PSO) and A* algorithm to obtain the solution [19].

The application research of intelligent optimization algorithm in solving the shortest path problem has also received extensive attention from researchers. Liu *et al.* improved the machine learning algorithm to solve the path planning problem of intelligent driving vehicles with restricted road sections [4]. Ant colony optimization (ACO) has the characteristics of parallelism, robustness, and positive feedback, and has been widely used to solve combinatorial optimization problems, such as the famous traveling salesman problem (TSP), secondary allocation problems (QAP), job shop scheduling problems (JSP) and many other complex combinatorial optimization problems. Due to the huge number of

intersections and road sections in the urban road network, the ant colony algorithm is easy to fall into the local optimum when solving the path planning problem, and has the disadvantages of high number of iterations and high time complexity. Aiming at the above shortcomings of the ant colony algorithm, Chen *et al.* combined with the A* algorithm to propose an efficient and fast improved algorithm to improve the efficiency and accuracy of the search [20]. In order to solve the Capacitated Vehicle Routing Problem (CVRP), Shang *et al.* aimed to develop a multi-constraint path optimization method with simple structure, strong versatility, and easy coupling and nesting, and proposed a path optimization method with specific capacity constraints. They proposed an improved simulated annealing algorithm (ISA) with tempering operation. An overall optimization framework based on simulated annealing operations was constructed, and for the improvement of global search capabilities, initial solution generation, better solution acceptance rules and neighborhood transformation strategies were designed respectively. Afterwards, through comparative experiments, the effectiveness and robustness of ISA were verified [21]. For the Capacitated Vehicle Routing Problem (CVRP), Mohammed *et al.* proposed an improved genetic algorithm [22], Yang put forward a hybrid algorithm of genetic algorithm and particle swarm optimization (PSO) [23], and Chen *et al.* put forward a hybrid algorithm of ant colony optimization (ACO) and particle swarm optimization (PSO) [24]. Regarding the pick-up and delivery vehicle routing problem with time windows, Yan *et al.* put forward an improved particle swarm optimization algorithm [25]. Zhang *et al.* put forward a new discrete particle swarm optimization based on route-segment to solve the vehicle routing problem with time windows (VRPTW), they setting the route-segment as the velocity of particles and in which the velocity and position updating rules are designed based on the concept of ''ruin and recreate'' [26].

III. MATHMATICAL FORMULATIONS

A. PROBLEM STATEMENT

In this paper, we conduct research on vehicle path planning considering intersections time consumption in a deterministic network. In such a network, we assumed that the travel time along an arc and the waiting time at a node that has different subsequent arcs do not vary with time. In Figure 1, nodes and arcs correspond respectively to intersections and road sections in the actual road network, S is the starting node and D is the destination. The number marked on a directed arc is the time required to pass through the arc. The number marked next to the node i respectively represent the time spent at the node i when turning from the node i to different subsequent node. When only considering the time spent on arcs, the minimum time path from the starting node S to the destination D is $S \rightarrow A \rightarrow C \rightarrow D$ with the time spent $2+3+2=7$. When the time cost of the node and the arc are considered at the same time, the time cost of the path $S \rightarrow A \rightarrow C \rightarrow D$ is $2+4+3+4+2=15$. However, the time spent on path $S \rightarrow B \rightarrow G \rightarrow D$ is

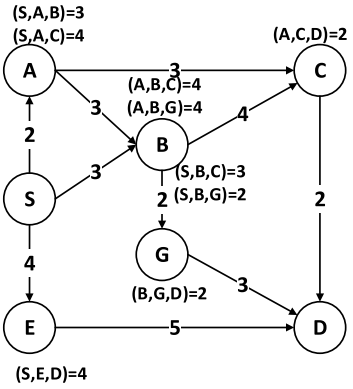

FIGURE 1. Network graph considering node cost.

$3+2+2+2+3=12$, which becomes the minimum time path from the starting node S to the destination D instead of the path $S \rightarrow A \rightarrow C \rightarrow D$. It can be seen that we can get different paths under the two conditions, and considering the time spent at the intersection is more realistic, so it is necessary to study the optimization algorithm of vehicle path planning considering node attributes in this paper.

B. PROBLEM ASSUMPTIONS

Combining the characteristics of the problem addressed in this paper, give the following assumptions:

- Assume that the road network is a deterministic directed graph network in which the attribute values of nodes and edges do not change over time, and follow the first-in first-out (FIFO) rule.
- Within a certain time period, the time it takes for a vehicle to pass through any section and intersection of the road network does not rely on the departure time.
- Within the same time period, the time it takes for a vehicle to pass the intersection in any way (including left turn, right turn, go straight, and U-turn) is independent and is independent of the time it arrive at the intersection.
- Assume that the time spent by the vehicle traveling on any segment of the road network is a fixed constant.
- Assumed that the time it takes a vehicle to go through an intersection in any given way is a fixed constant.

C. MATHEMATICAL MODEL

We adopt the four-tuple directed graph $G = (\mathbb{V}, \mathbb{E}, \Omega, \Phi)$ to describe the road network we are studying.

$\mathbb{E}$: The collection of all road segments between intersections in the road network (corresponding to the arcs in the directed graph).

$|\mathbb{E}|$: The number of arcs in the directed graph.

$\mathbb{V}$: The collection of all intersections in the road network (corresponding to the nodes in the directed graph).

$|\mathbb{V}|$: The number of node in the directed graph.

$\mathbb{S}^+(i) = \{j|(i,j) \in \mathbb{E}$: The successor nodes collection of node $i(i \in \mathbb{V})$.

$\mathbb{S}_n^+(i)$: The subset of $\mathbb{S}^+(i)$, for $\forall j \in \mathbb{S}_n^+(i)$, arc (i,j) is not labeled B.

$\mathbb{P}^-(i) = \{j|(i,j) \in \mathbb{E}$: The predecessor nodes collection of node $i(i \in \mathbb{V})$.

$\mathbb{P}_n^-(i)$: The subset of $\mathbb{P}^-(i)$, for $\forall h \in \mathbb{P}_n^-(i)$, arc (h, i) is not labeled B.

$\Omega = \{\mathcal{L}(i,j)|i,j \in \mathbb{V}, (i,j) \in \mathbb{E}$: The collection of each arc attribute values, where, $\mathcal{L}(i,j)$ is the attribute value of the arc segment (i,j), i.e., the time it takes for a vehicle to pass through road segment (i,j).

$\Phi = \left\{\Lambda(i,j,k) \,|\, j \in \mathbb{V}, i \in \mathbb{P}^-(j), k \in \mathbb{S}^+(j)\right\}$: Node attribute values collection in the directed graph, where $\Lambda(i,j,k)$ is the attribute value of node j (weight or time cost), it represents the time it takes for the vehicle to travel from road segment (i,j) to road segment (j,k) at intersection j.

Assuming that D is a certain intersection in the road network, find the path $\{j_0, j_1, j_2, \cdots, j_{n-1}, j_n\}$ ($j_0 = S$ and $j_n = D$) with the smallest sum of attribute values from any other intersection S to D in the network. That is, the path with the smallest sum of the time spent by the vehicle traveling on the road segment and the time spent passing through the intersection, which is called the shortest path problem in deterministic network considering the intersection attributes. There are the following assumptions:

- When the vehicle travels along the path $\left\{j_0, j_1, j_2, \cdots, j_{n-1}, j_n\right\}$, the vehicle departs from intersection j_k or the vehicle arrives at intersection j_k means that the vehicle is at the starting node of road segment (j_k, j_{k+1}).
- When the vehicle arrives at the destination node D, we assume that vehicle passes the road segment (j_{n-1}, j_n) and, through the node $j_n = D$, arrives at the starting node of the road segment (j_n, j_{n+1}), where $j_{n+1} = D$, $(j_n, j_{n+1}) = (D, D)$ is a virtual road segment, and $\Lambda(j_{n-1}, D, D) = 0$.

The mathematical model of the problem can be described as follows, where $\Gamma(S, D)$ is the minimum travel time from starting node S to the destination D.

$$\Gamma(S,D) = \min_{(j_0,\cdots,j_{n-1},j_n)} \sum_{k=1}^{n} [\mathcal{L}(j_{k-1},j_k) + \Lambda(j_{k-1},j_k,j_{k+1})] \tag{1}$$

D. ANALYSIS OF MATHEMATICAL MODEL

Theorem 1: Assume the path $(j_0, j_1, j_2, j_3 \ldots, j_{n-1}, j_n)$ is the optimal solution of the shortest path problem in the time-invariant network from the starting node $j_0 = S$ to the destination node $j_n = D$ considering the attributes of the intersection, it is easy to obtain that any sub-path $(j_{k+1}, j_{k+2}, j_{k+3}, \ldots, j_{k+l-1}, j_{k+l})$ of the path $(j_0, j_1, j_2, j_3 \ldots, j_{n-1}, j_n)$ is the optimal path from node j_{k+1} to node j_{k+l}. In other words, in the time-invariant network, there is an optimal substructure for the optimal solution of the shortest path problem that considering the time consumption of the intersection.

Proof: We use the cut-and-paste method to prove the correctness of the theorem. Suppose the sub-path $(j_{k+1}, j_{k+2}, j_{k+3}, \ldots, j_{k+s-1}, j_{k+s})$ is not the shortest path from the starting node j_{k+1} to destination node j_{k+s} in the directed graph. Then there must exist another sub-path $(j_{k+1}, j_{k+2}, j'_{k+3}, \ldots, j'_{k+q-1}, j'_{k+q})$ that satisfies:

$$\beta(j_{k+1}, j_{k+2}, j'_{k+3}, \ldots, j'_{k+q-1}, j'_{k+q}) < \beta(j_{k+1}, j_{k+2}, j_{k+3}, \ldots, j_{k+s-1}, j_{k+s}) \tag{2}$$

where:

$$\beta(j_{k+1}, j_{k+2}, j'_{k+3}, \ldots, j'_{k+q-1}, j'_{k+q}) = \sum_{t=1}^{q-1} [\mathcal{L}(j'_{k+t}, j'_{k+t+1}) + \Lambda(j'_{k+t}, j'_{k+t+1}, j'_{k+t+2})] \tag{3}$$

where j'_{k+1} is the beginning of road segment (j_{k+1}, j_{k+2}), j'_{k+2} is the beginning of road segment (j_{k+2}, j'_{k+3}), j'_{k+q} is the beginning of road segment (j'_{k+q}, j_{k+s+1}), and node j'_{k+q+1} is the beginning of road segment (j_{k+s+1}, j_{k+s+2}); and:

$$\beta(j_{k+1}, j_{k+2}, j_{k+3}, \ldots, j_{k+s}) = \sum_{t=1}^{s-1} [\mathcal{L}(j_{k+t}, j_{k+t+1}) + \Lambda(j_{k+t}, j_{k+t+1}, j_{k+t+2})] \tag{4}$$

where j_{k+1} is the beginning of road segment (j_{k+1}, j_{k+2}), j_{k+s} is the beginning of road segment (j_{k+s}, j_{k+s+1}). Therefore, we have:

$$\begin{aligned}\beta(j_0, j_0, j_0 \ldots, j_k) + \beta(j_{k+1}, j_{k+2}, j'_{k+3}, j'_{k+4}, \ldots, j'_{k+s}) \\ + \beta(j_{k+s+1}, j_{k+s+2} \ldots, j_n) < \beta(j_0, j_0, j_0 \ldots, j_k) \\ + \beta(j_{k+1}, j_{k+2}, j_{k+3}, j_{k+4} \ldots, j_{k+s}) \\ + \beta(j_{k+s+1}, j_{k+s+2}, \ldots, j_n)\end{aligned} \tag{5}$$

This is incompatible with the assumption that path $(j_0, j_1, j_2, j_3 \ldots, j_{n-1}, j_n)$ is the shortest path from the starting node $j_0 = S$ to the destination node $j_n = D$ in the directed graph, so the assumption does not hold. Therefore, theorem 1 is correct.

Theorem 2: There is optimal substructure for the optimal solution of the shortest path problem in the time-invariant network considering intersection attributes.

Theorem 3: There is repeated sub-problems for the shortest-path problem in the time-invariant network considering intersection attributes, so recursive algorithms can be used to solve the problem.

IV. RLDA ALGORITHM DESIGN

Based on the previous analysis, we can use the dynamic programming method to design the solving algorithm. Next, we will introduce the basic formulas involved in the algorithm.

A. BASIC FORMULA

$\Gamma(j,k,l)$ is the minimum travel time starting from the first node j of the road segment (j,k) and passing through this segment, then through intersection k and the segment (k,l) to the end point D. Then, there is the following recursion

formula:

$$\Gamma(j,k,l) = \min_{m\in\mathbb{S}^+(l)} \{\mathcal{L}(j,k) + \Lambda(j,k,l) + \Gamma(k,l,m)\} \quad (6)$$

where

$$\Gamma(j,D,D) = \mathcal{L}(j,D), \quad \forall j \in \mathbb{P}^-(D) \quad (7)$$

From the above analysis and Theorem 1 and Theorem 2, there are the following equations:

$$\Gamma(j,D) = \min_{m\in\mathbb{S}^+(j), l\in\mathbb{S}^+(k)} \Gamma(j,k,l)\} \quad (8)$$

$\rho(j,k)$ is the minimum travel time from the first node i of the road section (i,j) passing through this section (i,j) and through intersection j to the destination D. Then: $\rho(j,k)$ is the shortest travel time from the starting node of the road segment (j,k) through the road segment (j,k) and the intersection k to the end node D, obtain the following formula:

$$\rho(j,k) = \min_{l\in\mathbb{S}^+(k)} \Gamma(j,k,l)\} \quad (9)$$

or

$$\rho(j,k) = \min_{l\in\mathbb{S}^+(k)} \{\mathcal{L}(j,k) + \Lambda(j,k,l) + \rho(k,l)\} \quad (10)$$

where

$$\mathcal{L}(j,k) \in \Omega,\ \Lambda(j,k,l) \in \Phi,\ k \in \mathbb{S}^+(j),\ l \in \mathbb{S}^+(k) \quad (11)$$

$$\rho(j,D) = \mathcal{L}(j,D), \quad (12)$$

Let $(k, \Pi_\rho(j,k))$ as the succeeding road segment of road segment (j,k) on the minimum travel time path via road segment (j,k) to end point D:

$$\Pi_\rho(j,k) = \underset{l\in\mathbb{S}^+(k)}{argmin} \{\mathcal{L}(j,k) + \Lambda(j,k,l) + \rho(k,l)\} \quad (13)$$

We denote the minimum travel time from the starting point S to the end point D by $\psi(\mathrm{S},\mathrm{D})$, and we record $\Pi_\psi(j)$ as the successor node of node i for this path. From the above analysis, it follows that: Define $\psi(\mathrm{S},\mathrm{D})$ as the shortest travel time from the starting node S to the destination D, define $\Pi_\psi(i)$ as the successor node of node j in the shortest path.

$$\psi(S,D) = \underset{k\in\mathbb{S}^+(S)}{\operatorname{argmin}} \{\mu(S,k)\} \quad (14)$$

$$\Pi_\psi(j) = \underset{k\in\mathbb{S}^+(S)}{\operatorname{argmin}} \{\rho(j,k)\} \quad (15)$$

B. OVERVIEW OF RLDA ALGORITHM

For the solution of the discussed problem, we propose a Reverse Labeling Dijkstra Algorithm (RLDA) based on traditional Dijkstra. Its basic idea is: Beginning with the destination, gradually explore the shortest path that through one road segment (j,k) (with node j as its starting node) to the destination D in the directed graph. In the process, after the minimum travel time $\rho(j,k)$ from the given arc (j,k) to the end node D is calculated, mark the arc (j,k) with B label (Black arc). Then calculate the minimum travel time from all arcs (i,j) with the node j as the end node to the destination D.

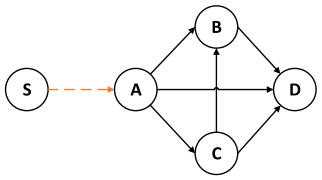

FIGURE 2. Case 1.

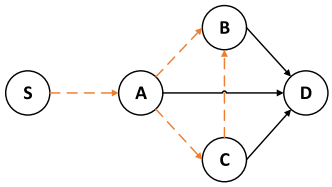

FIGURE 3. Case 2.

During this process, if all arcs with j as the starting node have the B label (that is, the minimum travel time through the arc to the destination D already be obtained), the recursive equation (5) can be used to calculate the minimum travel time from the arc (i,j) to the end D. If an arc starting at node j does not have a label B, then node j is taken as the tree root, arc segments without label B are taken as the search objects. Find out all the arcs that start from node j and connected with node j without the B label, and construct a breadth-first search tree (BFS-tree), all arcs in this tree are marked with G label (green arcs). Then, the minimum travel time from any arc in the tree to the destination D is calculated, and the arcs with label G in the tree are changed to B label. At the end, calculate the minimum travel time from the arc (i,j) the destination D. It can be seen that each step of the algorithm is to change a certain (or some) road segment without B label to B label, as a result, the number of arcs marked with B in the directed graph G increases. In this way, at most $|\mathbb{V}|$ iterations, the shortest path from any road segments to the destination can be found, the shortest path from the starting node S to the destination D is also found. In each iterative process, divide the road segments set in the directed graph into three subsets: the set of road sections marked with B ($\mathbb{E}_B$), the set of road sections marked with G ($\mathbb{E}_G$), and the complement set $\mathbb{E}_O$ (marked by Orange) of sets $\mathbb{E}_B$ and $\mathbb{E}_G$, $\mathbb{E}_O = \mathbb{E} - (\mathbb{E}_B \cup \mathbb{E}_G)$. Because the RLDA algorithm is executed in reverse, it will first calculate the optimal path from the arc segment that is close to the end point to the end point. The obtained data for the current arc segment can be used in the next iteration to calculate the optimal path from its predecessor arc segment to the end point, making full use of existing data to reduce the calculation time of the algorithm.

We use an example to show the performed process. As shown in Figure 2 and Figure 3, the black arc (j,k) indicates that the minimum travel time $\rho(j,k)$ from the arc (j,k) to the end point has been calculated, that is, arc (j,k) already marked by B. In Figure 2, select node A as the current node, we need to calculate the minimum travel time $\rho(S,A)$ from arc (S,A) (orange dotted line) to the destination

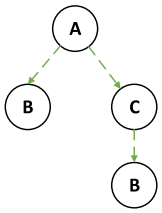

FIGURE 4. Breadth-first search tree.

node D. Since the arc segments $(A, k)(\forall k \in \mathbb{S}^+(A))$ are already marked by B, $\rho(S, A)$ can be computed directly by equation (5). In Figure 3, calculate the minimum travel time $\rho(S, A)$ from the arc (S, A) to the destination node D, since the arc segments (A, B) and (A, C) that take A as starting node are not marked by B, a breadth-first search tree with node A as the root node need to be constructed, then put all the arcs with node A as the starting node that are not marked with the B label on the tree and mark with the G label (green dotted line), as shown in FIGURE 4. After $\rho(A, B), \rho(A, C)$, and $\rho(C, B)$ are all calculated, then the minimum travel time $\rho(S, A)$ from the arc (S, A) to the destination node D can be obtained.

C. SPECIFIC STEPS OF RLDA ALGORITHM

Given a directed graph $G = (\mathbb{V}, \mathbb{E}, \Omega, \Phi)$, node collection $\mathbb{V}$, arc collection $\mathbb{E}$, node attribute value collection Φ, arc attribute value collection Ω, and destination D:

Step0. Initialization: $\mathbb{E}_B = \emptyset, \mathbb{E}_G = \emptyset, \mathbb{V}_G = \{w | w \in D$; for any $j \in \mathbb{V}$, let $S_n^+(j) = \mathbb{S}^+(j), \mathbb{P}_n^-(j) = \mathbb{P}^-(j)$; $n = 1$.

Step 1. If $\mathbb{E}_B = \mathbb{E}$, jump to Step 5; or else, jump to Step 2.

Step 2. Select one node $w \in \mathbb{V}_G$. If $\mathbb{S}_n^+(w) = \emptyset$, jump to Step 3; Else if $\mathbb{S}_n^+(w) \neq \emptyset$, that is, at least one node $k \in \mathbb{S}^+(w)$, make $(w, k) \notin \mathbb{E}_B$ [it means the arc (w, k) is not marked by B], do:

Step 2.1 Let w be the root node, take the arc segment that do not have the B label as the search target, build a BFS-tree. Select the arcs (j, k) that satisfy $k \in \mathbb{S}_n^+(j)$ and not in the tree [that is, arcs (j, k) don't have B or G label], and put them into the constructed BFS-tree, and insert such arcs into the stack $\mathbb{E}_G$ in order, remove the node k from the collection $\mathbb{S}_n^+(j)$, until the search process no longer able to find an arc segment that doesn't have B or G label.

Step2.2 Take arcs (j, k) from the stack $\mathbb{E}_G$ in turn, put each node from the node set $\mathbb{P}_n^-(k) - \mathbb{V}_G$ into the queue $\mathbb{V}_G$ (the node is labeled G), and for each arc (j, k), calculate:

$$\rho(j, k) = \min_{l \in \mathbb{S}^+(k)} \{\mathcal{L}(j, k) + \Lambda(j, k, l) + \rho(k, l)\}$$

and record

$$\Pi_\rho(j, k) = \underset{l \in \mathbb{S}^+(k)}{argmin} \{\mathcal{L}(j, k) + \Lambda(j, k, l) + \rho(k, l)\}$$
$$\gamma((j, k)) = ((k, \Pi_\rho(j, k))$$

Insert (j, k) into $\mathbb{E}_B$ and delete node j from $\mathbb{P}_n^-(k)$. Repeat the above operation until $\mathbb{E}_G = \emptyset$, then jump to Step 3.

Step3. If $\mathbb{P}_n^-(w) = \emptyset$, then remove node w from queue $\mathbb{V}_G$, jump to Step 4; Else if $\mathbb{P}_n^-(w) \neq \emptyset$, for arbitrary $j \in \mathbb{P}_n^-(w)$, if $j \notin \mathbb{V}_G$, insert node j into queue $\mathbb{V}_G$, insert arc (j, w) into stack $\mathbb{E}_B$, remove w from $\mathbb{S}_n^+(j)$, remove j from $\mathbb{P}_n^-(w)$, remove w from $\mathbb{V}_G$, calculate:

$$\rho(j, w) = min\{\mathcal{L}(j, w) + \Lambda(j, w, z) + \rho(w, z)\}$$

Record:

$$\Pi_\rho(j, w) = \underset{z \in \mathbb{S}^+(w)}{argmin} \{\mathcal{L}(j, w) + \Lambda(j, w, z) + \rho(w, z)\}$$
$$\gamma((j, w)) = ((w, \Pi_\rho(j, w))$$

Jump to Step 4.

Step4. $n = n + 1$, jump to Step 1.

Step5. Calculate

$$\psi(S, D) = \underset{k \in \mathbb{S}^+(s)}{\text{argmin}} \{\mu(S, k)\}$$

Obtain the shortest time from the starting node S to the end point D.

Step6. Construct the optimal path from the start node S to the end node D by $\Pi_\rho(j, w)$,

$$\Pi_\psi(S) = \underset{k \in \mathbb{S}^+(s)}{\text{argmin}}\{\mu(s, k)\}, \text{ and } \gamma((j, w)) \text{ END.}$$

The steps for building BFS-tree in the above subroutine (step 2.1 to step 2.2) as follows:

Step0. BEGIN. $\mathbb{E}_Y = \{(w, w)\}$.

Step1. If $\mathbb{E}_Y = \emptyset$ jump to Step 3; Else if, $\mathbb{E}_Y \neq \emptyset$, take arc $(k, l) \in \mathbb{E}_Y$, when $\mathbb{S}_n^+(l) \neq \emptyset$, take out $m \in S_n^+(l)$ in turn, insert arc segments (l, m) into the stack $\mathbb{E}_G$, and record $F(l, m) = (k, l)$ [means that the sub-arc of the arc (k, l) on the BFS-tree is (l, m), until $\mathbb{S}_n^+(j) = \emptyset$. Jump to Step 2.

Step2. Delete (k, l) from $\mathbb{E}_Y$, jump to Step 1.

Step3. Output stack $\mathbb{E}_G$. END.

D. MODULAR PSEUDOCODE OF RLDA ALGORITHM

The operation of the proposed RLDA algorithm mainly include: build the breadth-first search tree (BFS-tree), queue, and stack. The BFS algorithm is one of the basic graph search algorithms and the prototype of many important graph search algorithms. Dijkstra algorithm adopted the similar idea to BFS. It is a blind search method that systematically checks all nodes in the graph to find the optimal solution. Stacks and queues are widely used in program design, they are both linear data structures, stack obeys the rule "last in first out" [Insert(L, m+1, y), Delete(L, m)], and queue obeys the rule "first in first out" rule [Insert (L, m+1, y),

Algorithm 1: BFS-Tree(Module1)

BFS(G, w)
% Given directed graph $G = (\mathbb{V}, \mathbb{E}, \Omega, \Phi)$, w is root node of the tree, build a BFS-Tree using arcs without the B label.
for *each* $(j, m) \notin \mathbb{E}_B \cup \mathbb{E}_G$ **do**
    Color $[(j, m)] \leftarrow$*green*
end

Algorithm 2: Stack Operation(Module2)

Push (stack E_G, (j, m));
do
    $\mathbb{E}_B = \mathbb{E}_G \cup \{(j, m)\}$;
    $\mathbb{S}_n^+(j) = \mathbb{S}_n^+(j) - \{m\}$;
    // Delete node m from successor nodes collection of j.
Until $(j, m) \notin \mathbb{E}_B \&\& (j, m) \notin \mathbb{E}_G$;
// BFS-tree completed
Pop(stack, $\mathbb{E}_G$);
While $\mathbb{E}_G \neq \emptyset$ **do**
    Color$[(j, m)] \leftarrow$*Black;*
    $\mathbb{E}_G = \mathbb{E}_G - \{(j, m)\}$;
    $\rho(j, m) = \min_{s \in \mathbb{S}^+(m)} \{\mathcal{L}(j, m) + \Lambda(j, m, s) + \rho(m, s)\}$
    $\Pi_\rho(j, m) = \operatorname{argmin}_{s \in \mathbb{S}^+(m)} \{\mathcal{L}(j, m) + \Lambda(j, m, s) + \rho(m, s)\}$
    $\gamma((j, m)) = ((m, \Pi_\rho(j, m))$
    //Compute the shortest time from each arc segment on the BFS-tree to the end point according to the stacking order.
then
    $\mathbb{P}_n^-(m) = \mathbb{P}_n^-(m) - \{j\}$
    //Remove j from the predecessor node collection of node m
End

Delete (L, 1)]. According to the above operations with different characteristics, we can decompose the RLDA algorithm into four modules that can be called mutually: build BFS-tree operation (Module 1), stacking and popping operation for arcs (Module 2), enqueue and dequeue operation for nodes (Module 3), optimal solution output operation (Module 4). Pseudo code corresponding to each module shown in the Algorithm 1 to 4.

There is mutual calling relationship between the four modules, we use Algorithm 5 to describe. It can be seen that each module has interface connected with other modules. After initialization, according to $\mathbb{E}_B = \mathbb{E}$ or $\mathbb{E}_B \neq \mathbb{E}$ respectively perform module 1 or module 4. Module 2 follows module 1 means that Module 2 is performed after Module 1 finished. In the same way, Module 3 follows Module 2. After Module 3 has been executed, according to whether $\mathbb{E}_B = \mathbb{E}$ or $\mathbb{E}_B \neq \mathbb{E}$, the algorithm decides to go to the next iteration or to exit the program after executing Module 4.

Algorithm 3: Queue Operation(Module3)

DeQueue($\mathbb{V}_G$)
// Select each arc (j, m) that meets the condition $m \in \mathbb{S}_n^+(j)$ and is not on the BFS-tree, put it into the BFS-tree, and put arcs (j, m) into $\mathbb{E}_G$.
for w $\in \mathbb{V}_G \&\& \mathbb{S}_n^+(w) = \emptyset$ **do**
if $\mathbb{P}_n^-(w) = \emptyset$
    $\mathbb{V}_G = \mathbb{V}_G - \{w\}$;
    // If w is isolated, remove node w from V_G
else
    for *each* $j \in \mathbb{P}_n^-(w)$;
        // For each predecessor node of w, do the following.
        if $j \notin \mathbb{V}_G$ **do**
        EnQueue $(\mathbb{V}_G, j)$;
        $\mathbb{V}_G = \mathbb{V}_G \cup \{j\}$; *// put node j into $\mathbb{V}_G$.*
        $\mathbb{E}_B = \mathbb{E}_B \cup (j, w)$;
        // Put (j, w) into the collection $\mathbb{E}_B$ and give it B label.
        $\rho(j, w) = \min_{x \in \mathbb{S}^+(w)} \{\mathcal{L}(j, w) + \Lambda(j, w, x) + \rho(w, x)\}$
        $\Pi_\rho(j, w) = \operatorname{argmin}_{x \in \mathbb{S}^+(w)} \{\mathcal{L}(j, w) + \Lambda(j, w, x) + \rho(w, x)\}$
        $\gamma((j, w)) = ((w, \Pi_\rho(j, w))$
        then
        $\mathbb{S}_n^+(j) = \mathbb{S}_n^+(j) - \{w\}$; *//Remove w from $\mathbb{S}_n^+(j)$.*
        $\mathbb{P}_n^-(w) = \mathbb{P}_n^-(w) - \{j\}$; *//Delete j from $\mathbb{P}_n^-(w)$*
        $\mathbb{V}_G = \mathbb{V}_G - \{w\}$;*% Delete w from $\mathbb{V}_G$*
        Update $\mathbb{E}_B$
end

E. THEORETICAL ANALYSIS OF RLDA ALGORITHM

1) CORRECTNESS OF RLDA ALGORITHM

Proposition 1: Before the nth iteration of reverse label Dijkstra algorithm (RLDA), if a certain node w already in the queue $\mathbb{V}_G$, then in the nth iteration, it is impossible for the node w to be put into the queue $\mathbb{V}_G$ again.

Proof: From the above execution process of the algorithm, the proposition is obviously correct.

Proposition 2: If a certain node$j$ in the queue $\mathbb{V}_G$ ($j \in \mathbb{V}_G$) is removed from the queue $\mathbb{V}_G$ in the nth iteration, there must be $S_n^+(i) = \emptyset$ after the n-th iteration of the RLDA algorithm, so for any m-th iteration (m>n), it is impossible for the node j to be put into the queue $\mathbb{V}_G$.

Proof: In the n-th iteration, assume one node j that in the queue $\mathbb{V}_G$ is removed from the queue $\mathbb{V}_G$, it can be known from the above perform process of RADA algorithm, when the Step2 and Step3 finished, $\mathbb{S}_n^+(i)$ must be empty ($\mathbb{S}_n^+(i) = \emptyset$), then before the m-th (m>n) iteration performed, $\mathbb{S}_n^+(i) = \emptyset$ is also established. However, in the m-th iteration, before a certain node j is put into the queue $\mathbb{V}_G$, $\mathbb{S}_n^+(i) \neq \emptyset$ must be established, which contradicts with $\mathbb{S}_n^+(i) = \emptyset$. So it is impossible for the node j to be put into the queue $\mathbb{V}_G$, and the proposition 2 is set up.

Algorithm 4: Optimal Path Output(Module4)

Output ( )
$\psi(S, D) = \min_{k \in \mathbb{S}^+(S)} \{\rho(S, k)\}$;
//Got the minimum time from starting node S to target node D.
$\Pi_\psi(S) = \operatorname{argmin}_{k \in \mathbb{S}^+(S)} \{\rho(S, k)\}$;
// Got the subsequent node of S on the minimum time path from the starting node S to the target node D.
$\Pi_\rho(j, w) = \operatorname{argmin}_{x \in \mathbb{S}^+(w)} \{\mathcal{L}(j, w) + \Lambda(j, w, x) + \rho(w, x)\}$;
// Got the subsequent arc (w, x) of arc (j, w) on the minimum time path from arc (j, w) to the target node D.
$\gamma((j, w)) = (w, \Pi_\rho(j, w))$;
// Record the subsequent arc of arc (j, w) on the optimal path from starting node S to target node D as $(w, \Pi_\rho(j, w))$.
$(S, \Pi_\rho(S) \to \gamma((S, \Pi_\rho(S)) \to \gamma(\gamma(((S, \Pi_\rho(S))), \cdots, (D, D)$;
// Output the optimal path from starting node S to target node D.
End

Algorithm 5: Call Relationship Among Modules

Initialization: $G = (\mathbb{V}, \mathbb{E}, \Omega, \Phi)$, $n = 1$, $\mathbb{E}_G = \emptyset$, $\mathbb{V}_G = \{w \mid w = D\}$, $\mathbb{E}_B = \emptyset$, *for* $\forall j \in \mathbb{V}$, $\mathbb{S}_n^+(j) = \mathbb{S}^+(j)$, $\mathbb{P}_n^-(j) = \mathbb{P}^-(j)$
Result: *A Sequence* $\lambda := (S, j), (j, k), \cdots, (D, D)$
Main procedure
if $\mathbb{E}_B \neq \mathbb{E}$
  for $\forall w \in \mathbb{V}_R$
  if $\mathbb{S}_n^+(w) = \emptyset$
    perform Module3;
      if $\mathbb{E}_B \neq \mathbb{E}$
        perform Module1;
      else
        perform Module4;
  else
    perform Module1;
    // build BFS-tree
    perform Module2;
    perform Module3
      if $\mathbb{E}_B \neq \mathbb{E}$
        perform Module1;
      else
        perform Module4;
  else *perform Module*4;
End

Proposition 3: When a certain iteration of the RLDA algorithm is completed, if $\mathbb{E}_B \neq \mathbb{E}$, then $\mathbb{V}_G \neq \emptyset$ must hold.

Proof: When a certain iteration of the RLDA algorithm is completed, if $\mathbb{E}_B \neq \mathbb{E}$, then there must exist a certain arc segment (k, l) that is not marked by B, and $m \in \mathbb{S}^+(l)$, $(l, m) \in \mathbb{E}_B$, $\mathbb{S}_n^+(k) = \emptyset$ and $\mathbb{P}^-(l) = \emptyset$ are also hold. From Proposition 2, there are two possibilities: node k is in the queue $\mathbb{V}_G$ or is out of the queue $\mathbb{V}_G$ and has never entered into $\mathbb{V}_G$. Assuming the latter holds, since the arc (l, m) has been marked by B, when the arc (l, m) is inserted into $\mathbb{E}_B$, the node k must be inserted into the queue $\mathbb{V}_G$, which is contradictory, so the hypothesis does not hold. Therefore, proposition 3 is proved.

Theorem 4: In the iterative process of the RLDA algorithm, for any node $j \in \mathbb{V}$, it enters and exits the queue $\mathbb{V}_G$ at most once, so the number of iterations of the algorithm does not exceed $|\mathbb{V}|$ times at most.

Proposition 4: The proposed reverse labeling Dijkstra algorithm (RLDA) has cycle invariance: before the start of the n-th iteration, if all the arcs in the set $\mathbb{E}_B$ have label B, then all the arcs in $\mathbb{E}_B$ still have label B before the start of the (n+1)-th iteration.

The theorem 5 can be derived by the above Theorem 4, Proposition 3, and Proposition 4.

Theorem 5: When the algorithm is iterated at most $|\mathrm{V}|$ times, $\mathbb{E}_B \neq \mathbb{E}$ holds, that is, for any arc $(j, k) \in \mathbb{E}$, the minimum travel time $\rho(j, k)$ from the arc (j, k) to the target node D has been obtained.

Proposition 5: In the iterative process of the RLDA algorithm that does not exceed $|\mathbb{V}|$ times at most, for any $(j, k) \in \mathbb{E}$, it enters the built BFS-tree at most once, and therefore it enters and exits the set $\mathbb{E}_G$ at most once.

Proof: If a certain arc segment (j, k) appears in the BFS-tree in a certain iteration, after the iteration is completed, the arc segment (j, k) must already be marked by B (means that (j, k) entered into $\mathbb{E}_B$), and in each subsequent iteration, any arc (j, k) inserted into the BFS tree was not marked by B before it is inserted, it is impossible that arc (j, k) be inserted into the BFS-tree again. Proposition 5 is corrected.

2) TIME COMPLEXITY OF RLDA ALGORITHM

The previous analysis shows that RLDA algorithm has at most $|\mathbb{V}|$ iterations, and the time complexity cost in the algorithm iteration process is mainly composed of three parts:

1) The computational time complexity cost of all nodes entering and leaving the queue $\mathbb{V}_G$.

2) The computational time complexity cost of all arc segment entering and leaving the stack $\mathbb{E}_G$ when building the BFS tree.

3) The computational time complexity cost to compute the least travel time $\rho(j, k)$ through each arcs segment (j, k) to the target node.

Let:

$$\mathbb{N} = \max_j \{|\mathbb{S}^+(j)|, |\mathbb{P}^-(j)|\} \tag{16}$$

Theorem 6: In the whole iterative process of the RLDA algorithm, in the process of building the BFS-tree, the computational time complexity cost of all nodes entering and leaving the queue $\mathbb{V}_G$ and all arcs entering and leaving the collection stack $\mathbb{E}_G$ is $O(|\mathbb{V}| + \mathbb{N} \cdot |\mathbb{E}|)$.

Proof: The proof of the theorem consists of three steps.

Firstly, prove that the computational time complexity cost of all arc segments entering and leaving the stack $\mathbb{E}_G$ is $O(N \cdot |\mathbb{E}|)$ in the entire iteration of the RLDA algorithm when building the BFS-tree. Based on Proposition 5, each arc $(j, k) \in \mathbb{E}$ in the directed graph entering and leaving the stack $\mathbb{E}_G$ at most one time. From the specific steps of the RLDA algorithm, the time complexity cost for each arc to enter the collection stack $\mathbb{E}_G$ is $O(1)$. Therefore, the total time complexity cost for all arcs to enter the stack $\mathbb{E}_G$ is $O(|\mathbb{E}|)$. For each arc segment $(j, k) \in \mathbb{E}_G$ that already in the stack $\mathbb{E}_G$, the process of it is deleted from the stack $\mathbb{E}_G$ includes two operations: 1) Moving the one arc segment (j, k) from the stack $\mathbb{E}_G$ to the set $\mathbb{E}_B$, the corresponding time complexity cost is $O(1)$; 2) Finding a node j in set $\mathbb{P}_n^-(k)$ and removing the node j from $\mathbb{P}_n^-(k)$, the corresponding time complexity cost is $O(\mathbb{N})$. So the time complexity cost of all arc (j, k) being deleted from the stack $\mathbb{E}_G$ is $O(\mathbb{N} \cdot |\mathbb{E}|)$. To sum up, the computational time complexity cost of all arc segments entering and leaving the stack $\mathbb{E}_G$ is $O(\mathbb{N} \cdot |\mathbb{E}|)$.

In the second step, we prove that the time complexity cost of removing all the nodes $j \in \mathbb{V}_G$ from the queue $\mathbb{V}_G$ is $O(|\mathbb{V}|)$. From Theorem 4, that in the entire iteration of the RLDA algorithm, in the entire iteration of the algorithm, for any node $j \in \mathbb{V}$, the node j enters and exits the queue $\mathbb{V}_G$ at most one time. Based on the specific steps of the RLDA algorithm, the time complexity cost of each node being removed from the queue $\mathbb{V}_G$ is $O(1)$.So the time complexity cost for all nodes to be removed from the queue $\mathbb{V}_G$ is $O(|\mathbb{V}|)$.

The third step is to prove that the time complexity cost for all nodes enter into the queue $\mathbb{V}_G$ is $O(|\mathbb{E}|)$. For any node j that enters and exits queue $\mathbb{V}_G$, it is either the predecessor node $((j \in \mathbb{P}_n^-(w))$ of a certain node w that will be deleted from the queue $\mathbb{V}_G$, or it is the predecessor node $((j \in \mathbb{P}_n^-(l))$ of the tail node l of the arc (k, l) in the BFS tree. (1) If it is the former, when node j enters into the queue $\mathbb{V}_G$, all nodes $v \in \mathbb{P}_n^-(w)$ and are not in queue $\mathbb{V}_G$ will enter into the queue $\mathbb{V}_G$, that is starting from node w, find all arc segments (j, w) $(j \in \mathbb{P}_n^-(w), j \notin \mathbb{V}_G)$, then all the starting node j of arc (j, w) will enter into the queue $\mathbb{V}_G$, so in the entire iteration of the algorithm, the corresponding time complexity in this case is $O(|\mathbb{E}|)$. (2) If it is the latter case, when node j enters into enter into the queue $\mathbb{V}_G$, all nodes $v \in \mathbb{P}_n^-(l)$ and are not in queue $\mathbb{V}_G$ will enter into the queue $\mathbb{V}_G$, that is starting from node l, find all arc segments (j, l) $(j \in \mathbb{P}_n^-(l), j \notin \mathbb{V}_G)$, then all the starting node j of arc (j, w) will enter into the queue $\mathbb{V}_G$, then $\mathbb{P}_n^-(l) = \emptyset$ established. Therefore, in later iterations, if an arc segment (h, l) with the same tail node as the arc segment (k, l) is found in $\mathbb{E}_G$, since $\mathbb{P}_n^-(l)$, there is no need to scan the previous founded arc (j, l) again; and it can be seen from Proposition 5 that in no more than $|\mathbb{V}|$ iterations of the RLDA algorithm, for any arc $(k, l) \in \mathbb{E}$, the arc (k, l) entering and leaving the BFS tree at most once. So, for any node $l \in V$, arc (j, l) $(j \in \mathbb{P}_n^-(l))$ needs to be scanned at most once, and the founded starting node j of the arc (j, l) enters into the queue $\mathbb{V}_G$. The corresponding time complexity in this case is also $O(|\mathbb{E}|)$.

TABLE 1. Arc attribute value.

$\mathcal{L}(S,A)$	$\mathcal{L}(S,H)$	$\mathcal{L}(S,E)$	$\mathcal{L}(A,B)$	$\mathcal{L}(A,I)$	$\mathcal{L}(H,A)$
3	4	4	3	3	3
$\mathcal{L}(H,G)$	$\mathcal{L}(E,G)$	$\mathcal{L}(E,F)$	$\mathcal{L}(B,C)$	$\mathcal{L}(I,C)$	$\mathcal{L}(I,J)$
4	5	6	3	4	3
$\mathcal{L}(G,J)$	$\mathcal{L}(F,J)$	$\mathcal{L}(F,D)$	$\mathcal{L}(C,D)$	$\mathcal{L}(J,C)$	$\mathcal{L}(J,D)$
5	6	6	7	5	6

To sum up, in the entire iterative process of the RLDA algorithm, in the process of building the BFS-tree, the computational time complexity cost of all nodes entering and leaving the queue $\mathbb{V}_G$ and all arcs entering and leaving the collection stack $\mathbb{E}_G$ is $O(|\mathbb{V}| + N \cdot |\mathbb{E}|)$. Theorem 6 is correct.

Theorem 7: In the iterative process of the RLDA algorithm, the time complexity cost required to computed the least time from all arc segments $((j, k) \in \mathbb{E})$ in the directed graph to the target node D is $O(N \cdot |\mathbb{E}|)$.

Proof: For any $(j, k) \in \mathbb{E})$, when compute the minimum time $\rho(j, k)$ from arc (j, k) to the destination node D based on the equation (10), the equation (17) need to be computed $|\mathbb{S}^+(k)|$ times, and do $\left|\mathbb{S}^+(k)\right| - 1$ times comparison. Each computation of this formula requires 2 addition operations, so computed the minimum time $\rho(j, k)$ from an arc (j, k) to the destination, the cost of time complexity is $O(\mathbb{N})$. So, compute the minimum time from all arcs to the target node, the required time complexity cost is $O(\mathbb{N} \cdot |\mathbb{E}|)$. Theorem 7 is correct.

$$\mathcal{L}(j, k) + \Lambda(j, k, l) + \rho(k, l) \tag{17}$$

From Theorem 6 and Theorem 7, the Theorem 8 can obviously be obtained.

Theorem 8: The time complexity of the proposed RLDA algorithm in the deterministic network considering node attributes is $O(|\mathbb{V}| + \mathbb{N} \cdot |\mathbb{E}|)$.

The above theoretical analysis shows that the RLDA algorithm proposed in this paper has lower time complexity.

F. COMPUTATIONAL EXAMPLE

In this section, we give a computational example to verify the algorithm proposed in this article. Use directed graph (Figure 5) to represent the road network graph, which contains 11 intersections and 18 directed arcs. The starting node is S and the destination node is D. Travel time of each road segment As shown in Table 1 (arc attribute $\mathcal{L}(j, k)$), the time spent on different turns at all intersections is shown in Table 2 (node attribute $\Lambda(i, j, k)$). The goal is to find the minimum time path from intersection S to destination node D.

When the node attribute (weight or time cost) is not considered, obtain the minimum time path from the start node S to the destination node D is S → A → I → J → D by Dijkstra algorithm, and the corresponding minimum time is 15.

When considering the node attribute, using the proposed RLDA algorithm proposed in this paper to compute the minimum time path from node S to destination node D. Omit the computation process, 9 iterations is 9 is required, the data

TABLE 2. Node attribute value.

$\Lambda(S,A,B)$ 2	$\Lambda(S,A,I)$ 4	$\Lambda(S,H,A)$ 2	$\Lambda(S,H,G)$ 3	$\Lambda(S,E,G)$ 4	$\Lambda(S,E,F)$ 3
$\Lambda(A,B,C)$ 2	$\Lambda(A,I,C)$ 4	$\Lambda(A,I,J)$ 4	$\Lambda(H,A,B)$ 3	$\Lambda(H,A,I)$ 2	$\Lambda(H,G,J)$ 5
$\Lambda(E,G,J)$ 3	$\Lambda(E,F,J)$ 3	$\Lambda(E,F,D)$ 4	$\Lambda(B,C,D)$ 3	$\Lambda(I,C,D)$ 3	$\Lambda(I,J,C)$ 4
$\Lambda(I,J,D)$ 5	$\Lambda(G,J,C)$ 5	$\Lambda(G,J,D)$ 4	$\Lambda(F,J,C)$ 4	$\Lambda(F,J,D)$ 3	$\Lambda(J,C,D)$ 2

TABLE 3. Computation result.

$\rho(\mathrm{J},\mathrm{D}) = 6, \gamma((\mathrm{J},\mathrm{D})) = (\mathrm{D},\mathrm{D})$	$\rho(\mathrm{J},\mathrm{C}) = 14, \gamma((\mathrm{J},\mathrm{C})) = (\mathrm{C},\mathrm{D})$
$\rho(\mathrm{C},\mathrm{D}) = 7, \gamma((\mathrm{C},\mathrm{D})) = (\mathrm{D},\mathrm{D})$	$\rho(\mathrm{F},\mathrm{D}) = 6, \gamma((\mathrm{F},\mathrm{D})) = (\mathrm{D},\mathrm{D})$
$\rho(\mathrm{F},\mathrm{J}) = 15, \gamma((\mathrm{F},\mathrm{J})) = (\mathrm{J},\mathrm{D})$	$\rho(\mathrm{G},\mathrm{J}) = 15, \gamma((\mathrm{G},\mathrm{J})) = (\mathrm{J},\mathrm{D})$
$\rho(\mathrm{I},\mathrm{J}) = 14, \gamma((\mathrm{I},\mathrm{J})) = (\mathrm{J},\mathrm{D})$	$\rho(\mathrm{I},\mathrm{C}) = 14, \gamma((\mathrm{I},\mathrm{C})) = (\mathrm{C},\mathrm{D})$
$\rho(\mathrm{B},\mathrm{C}) = 13, \gamma((\mathrm{B},\mathrm{C})) = (\mathrm{C},\mathrm{D})$	$\rho(\mathrm{E},\mathrm{F}) = 16, \gamma((\mathrm{E},\mathrm{F})) = (\mathrm{F},\mathrm{D})$
$\rho(\mathrm{E},\mathrm{G}) = 23, \gamma((\mathrm{E},\mathrm{G})) = (\mathrm{G},\mathrm{J})$	$\rho(\mathrm{H},\mathrm{G}) = 24, \gamma((\mathrm{H},\mathrm{G})) = (\mathrm{G},\mathrm{J})$
$\rho(\mathrm{H},\mathrm{A}) = 24, \gamma((\mathrm{H},\mathrm{A})) = (\mathrm{A},\mathrm{B})$	$\rho(\mathrm{A},\mathrm{I}) = 21, \gamma((\mathrm{A},\mathrm{I})) = (\mathrm{I},\mathrm{J})$
$\rho(\mathrm{A},\mathrm{B}) = 18, \gamma((\mathrm{A},\mathrm{B})) = (\mathrm{B},\mathrm{C})$	$\rho(\mathrm{S},\mathrm{E}) = 23, \gamma((\mathrm{S},\mathrm{E})) = (\mathrm{E},\mathrm{F})$
$\rho(\mathrm{S},\mathrm{H}) = 31, \gamma((\mathrm{S},\mathrm{H})) = (\mathrm{H},\mathrm{G})$	$\rho(\mathrm{S},\mathrm{A}) = 23, \gamma((\mathrm{S},\mathrm{A})) = (\mathrm{A},\mathrm{B})$

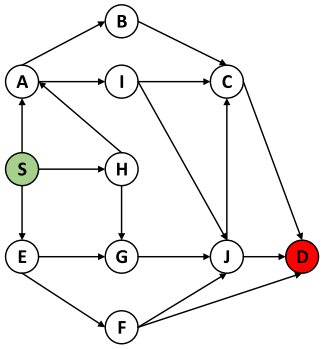

FIGURE 5. Directed graph for computational example.

obtained is shown in Table 3. Based on equation (14), the minimum time from node S to destination node D is obtained, as following equation:

$$\psi(\mathrm{S},\mathrm{D}) = \min_{k \in \mathbb{S}^+(S)} \{\mu(1,S)\} = \rho(S,A) = 23 \qquad (18)$$

According to $\gamma((\mathrm{k},\mathrm{l}))$, the minimum travel time path from node S to destination node D is S $\rightarrow$ A $\rightarrow$ B $\rightarrow$ C $\rightarrow$ D, and the corresponding time is 23.

V. EXPERIMENT

A. SIMULATION OF ROLA APPLIED IN URBAN NETWORK

This section provide a simulation of applying the proposed Reserve Labeling Dijkstra algorithm (RLDA) to obtain the optimal path in an actual urban road network. We perform the proposed RLDA algorithm in MATLAB on an Intel Core(TN) i7-6700HQ processor at 3.20 GHz with 16 GB RAM on a Windows 10 64-bit operating system. The data of the travel time of each road section and the times spent for different turns at the intersections in the selected area is randomly generated by the simulation software.

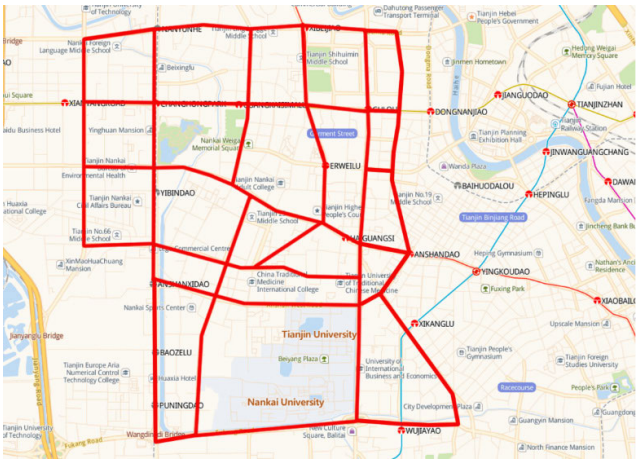

FIGURE 6. Target road network.

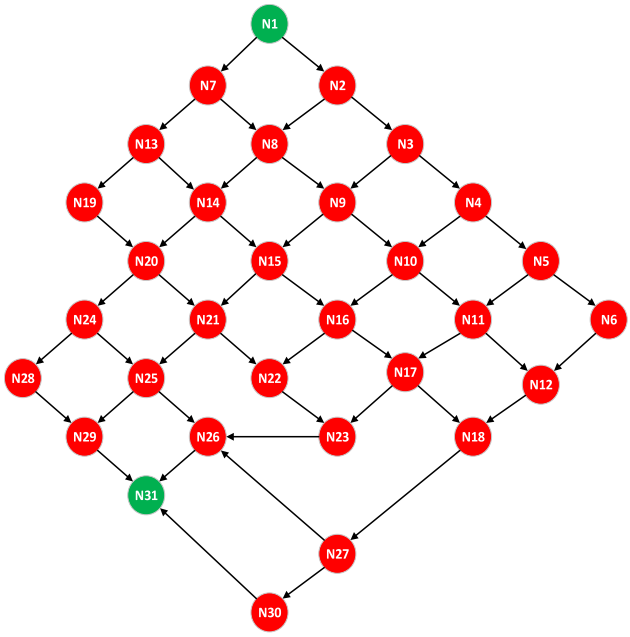

FIGURE 7. The simulation directed graph for the selected network.

1) ROAD NETWORK SELECTION AND SIMULATION DIAGRAM

Intercept the road network in a certain area of Tianjin map as the simulation object, shown in Figure 6. Circle the main road sections and intersections with red solid line, and generate the corresponding simulated directed graph in MATLAB, as shown in Figure 7. There are 49 road sections (arcs) and 31 intersections (node N1 to N31). Let the time $L(i,j)$ to pass through each road segment (i,j) and the time $\Lambda(i,j,k)$ spent on different turning at each intersection j obey uniform distribution U[1, 20]. Among them, node N1 is the starting node, and node N31 is the destination node, and the optimization goal is to obtain the least travel time path from node N1 to node N31 in the network.

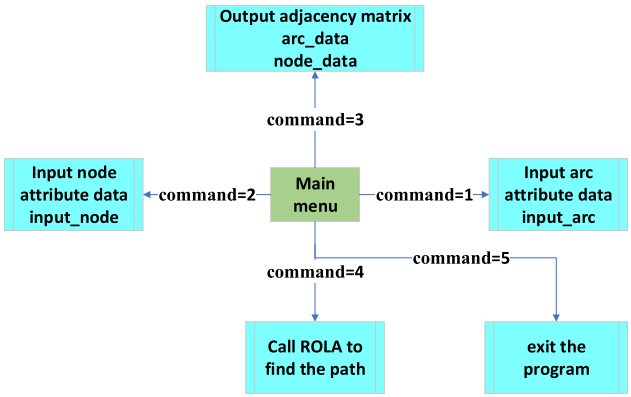

FIGURE 8. The connections relationship of all.m file.

2) SIMULATION PROGRAM DESIGN

The experiment program menu is the entrance to the program and calls other.m files according to different commands entered, includes five parts: (1) Input interface of arcs attribute values; (2) Input interface of node attribute values; (3) The output interface of the adjacency matrix of the inputted node and arc attribute values; (4) The main program file of the RLDA algorithm; (5) Exit the program. The connections between them are illustrated in Figure 8.

In the simulation procedure, the travel time of road segment (k, l) represented by $arc\_data(k, l)$, the turning time spent of road segment (k, l) at junction l to road segment (l, m) is represented by $node\_data(k, l, m)$. Because the simulated road network is a directed graph, generally: $node_{data}(k, l, m) \neq node\_data(m, l, k)$, $arc\_data(k, l) \neq arc\_data(l, k)$, if there is no road segment between node k and node l, then $arc\_data(k, l) = \infty$, $node\_data(j, k, l) = \infty$. For the road network with n intersections, arc_data is a $n * n$ two-dimensional matrix, $node\_data$ is a $n * n * n$ three-dimensional matrix. The pseudo code of the simulation program shown in Algorithm 6

3) RESULT

Because there are 31 nodes in the network, according to proposition 5, the RLDA algorithm iterates at most 31 times to get the optimal path from the start node N1 to the destination node N31 is N1 → N2 → N8 → N14 → N15 → N16 → N17 → N23 → N26 → N31, as shown in Figure 9, and the path on the selected map is shown in Figure10, with the least travel time is 81.5, and the CPU running time of the simulation is 0.0347s..

If the time cost of the intersection is not considered, for the same starting node and destination node, the corresponding optimal path is N1 → N2 → N8 → N9 → N15 → N21 → N25 → N29 → N31 by traditional Dijkstra algorithm, and the shortest time is 58.8, which is different from the path obtained by RLDA algorithm under considering intersection.

Algorithm 6: Pseudo of Simulation Program

```
Function menu()
command = 1;
arc_data=[ ];
node_data= [] ;
while command ~=5
    command -input('Welcome to the RLDA system for
    finding the shortest path! \n Please select: \n 1. Enter
    arc data \n 2. Enter turn data at the node \n 3. View
    data \n 4. Find the path \n5. Exit the program \n')
    ifcommand ==1
      arc_data=input_dat();
    elseif command == 2
      node_data=input_dat2()
    elseifcommand == 3
      disp (arc_data);
      disp (node_data);
    elseif command == 4
        arc = input('Please enter the target arc:');
        sta = input('Please enter the starting node:');
        dst = input('Please enter destination:');
          if sta == dst
           fprintf ('The starting node and the
           destination node cannot be the same! \r\n');
          else
            [Leng_arc, path_arc, Leng_sta, path_sta] =
            RLDA ( arc_data, node_data, arc, sta, dst);
            fprintf ('The minimum expected dst time
            from arc "arc"to destination "dst"is:
            %d \n',leng_arc);
            k = length(path_arc);
            fprintf('The passing path is:');
            for i = 1:1:k-1
              fprintf('%d→' , path_arc(i));
            end
              fprintf('%d\n' , path_arc(k));
            fprintf( 'The minimum travel time from
            starting node "sta" to destination
            "dst"is: %d\n',leng_sta);
              m = length(path_sta);
              fprintf('The passing path is:');
              for i =1:1:m-1
                fprintf('%d→' , path_arc(i));
              end
              fprintf('%d\n', path_sta(m));
          end
        end
    end
```

commonly used and latest path planning algorithms [28], including particle swarm optimization (PSO), ant colony algorithm (ACO), genetic algorithm (GA), neural network algorithm (NNA), OPABRL [4]. Analyze the performance and advantages of the RLDA algorithm compared with other

B. COMPARATIVE SIMULATION EXPERIMENT

1) EXPERIMENT ENVIRONMENT AND FACTOR SETTINGS

In this section, we will conduct simulation comparison experiments between the proposed RLDA algorithm and several

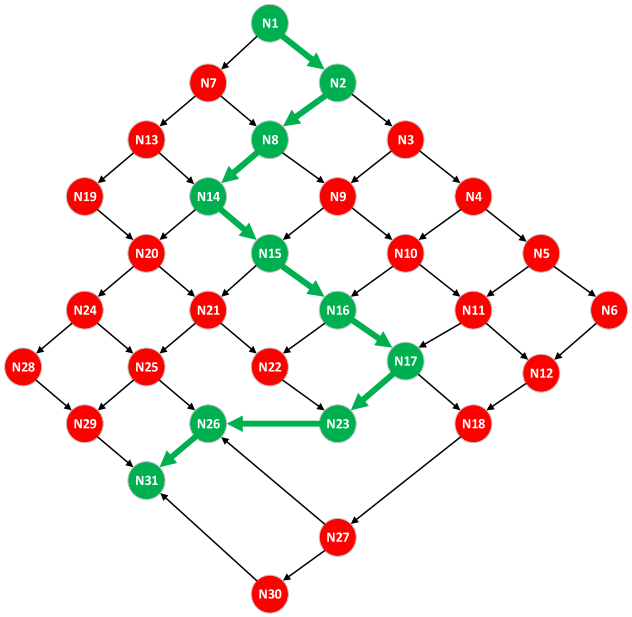

FIGURE 9. The outputted optimal path of simulation program.

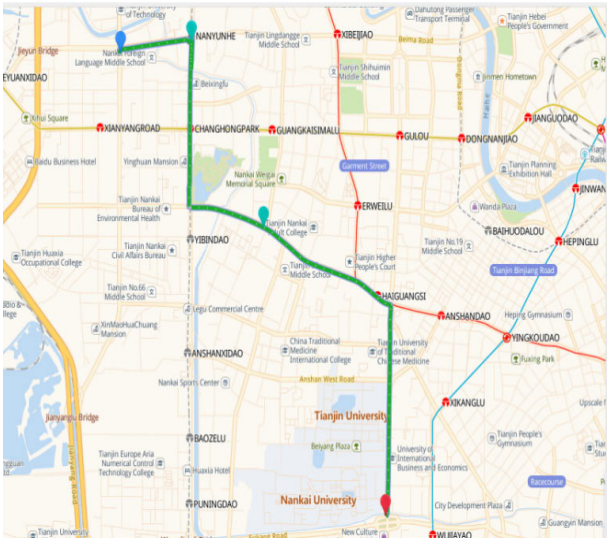

FIGURE 10. Optimal path on the selected map.

five algorithm. The parameter settings of each algorithm are shown in Table 4. Perform these simulation experiments on the MATLAB platform (he simulation environment configuration is the same as the previous first experiment). The comparative performance mainly verified by the simulation experiment includes: (1) Algorithm convergence rate; (2) Algorithm running time.

We select 10 sets of networks with different numbers of nodes and arcs, as shown in Table 5. The connection between nodes in each set of networks is similar to Figure 7. Assuming that each node has one or more other nodes connected to it, no more than six nodes. Randomly select a node as the starting node as the starting node, each node is numbered from near to far, and the node with the highest number as the

TABLE 4. Parameters setting.

Method	Factors	Value
ACO	Ant size	100
	Heuristic factor a	1
	Pheromone volatility ρ	0.2
GA	Population size	100
	Crossover probability P_1	0.6
	Mutation probability p_2	0.01
NNA	Network layers	4
	Number of input nodes	5
	Number of output nodes	1
	Number of hidden layer nodes	10
PSO	Particle swarm size	90
	Learning factor c_1	1.5
	Learning factor c_2	2.0
	Inertia weight parameter ω	0.8
OPABRL	Learning factor α	0.6
	Discount factor θ	1
	Greedy strategy ε	0.6

TABLE 5. Network scale of simulation experiment.

	Node_Number	Arc_Number
SET1	50	100
SET2	100	300
SET 3	150	500
SET 4	200	700
SET5	250	900
SET6	300	1100
SET 7	350	1300
SET 8	400	1500
SET 9	450	1700
SET10	500	1900

destination node. Same as before, for the network with n nodes, the attribute values $node\_data\,(k, l, m)$ of all nodes form a three-dimensional matrix of $n * n * n$, and the attribute values $arc\_data(k, l)$ of all arc segments form a two-dimensional matrix of $n * n$. Let the time $\mathcal{L}(i, j)$ to pass through each road segment (i, j) and the time $\Lambda(i, j, k)$ spent on different turning at each intersection j obey uniform distribution U[1, 5]. For the simulation experiment of each scale of road network, we need to run each algorithm to find the minimum time route from the selected starting node to the destination node, each algorithm runs ten times, and the average of the results is taken as the statistics.

2) COMPARATIVE EXPERIMENT RESULTS AND ANALYSIS

Experiment Result 1:Running the proposed RLDA algorithm, PSO algorithm, ACO algorithm, GA algorithm, OPABRL [4], and NNA, to find the optimal path between the same starting node and destination node in each scale network. Each algorithm runs ten times, the average number of iterations, path length, running time used for statistical analysis of algorithm performance are shown in Table 6.

Experiment Result 2: For the convergence rate of the algorithms involved in the experiment, taking a network with 100 nodes as an example, Figure 11-16 shows the average convergence rate of each algorithm running ten times. The x-axis is the number of iterations, and the y-axis is the optimal path time change obtained during the operation

TABLE 6. Statistics of experimental results.

Statistical Indicators	Method	SET 1	SET 2	SET 3	SET 4	SET 5	SET 6	SET 7	SET 8	SET 9	SET10
Iterations	RLDA	36	88	104	155	202	267	349	452	495	522
	PSO	43	93	136	203	301	399	452	469	475	479
	ACO	51	102	165	255	342	415	455	487	492	513
	GA	60	105	134	234	331	408	464	501	512	527
	NNA	57	110	173	265	358	422	456	478	501	526
	OPABRL	39	92	125	187	258	312	341	363	371	390
Path Length	ROLA	97	303	514	675	816	1023	1225	1421	1732	2003
	PSO	106	308	522	692	824	1035	1243	1436	1732	2014
	ACO	112	310	527	679	828	1044	1218	1421	1747	2005
	GA	117	310	532	712	833	1029	1228	1431	1729	2011
	NNA	114	314	533	704	819	1023	1237	1440	1744	2025
	OPABRL	105	306	514	683	816	1031	1234	1426	1753	2047
Running Time(ms)	ROLA	8	62	181	369	708	914	1251	1625	1844	1893
	PSO	14	190	442	1004	1213	1567	1617	1639	1652	1666
	ACO	21	168	607	1298	1591	1760	1792	1828	1834	1829
	GA	30	153	524	1158	1539	1822	1910	1926	1934	1927
	NNA	154	273	573	1067	1330	1548	1667	1746	1762	1769
	OPABRL	11	92	381	887	991	1162	1159	1213	1231	1242

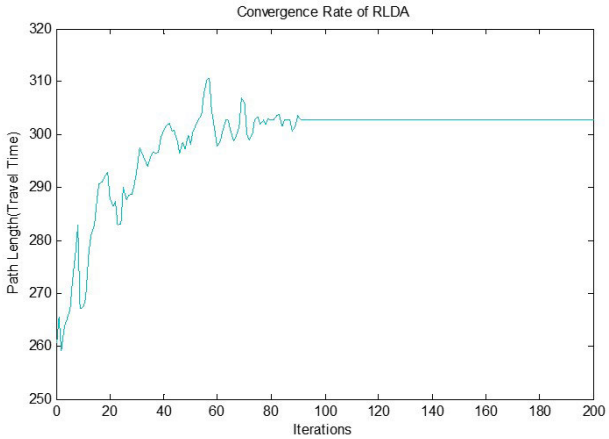

FIGURE 11. Convergence rate of RLDA (NODE_NUMBER = 100).

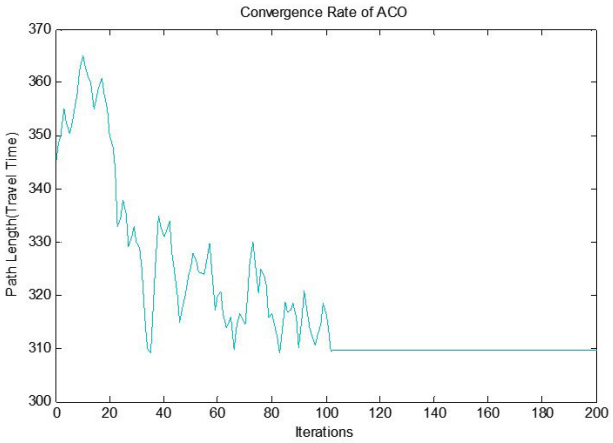

FIGURE 13. Convergence rate of ACO (NODE_ NUMBER = 100).

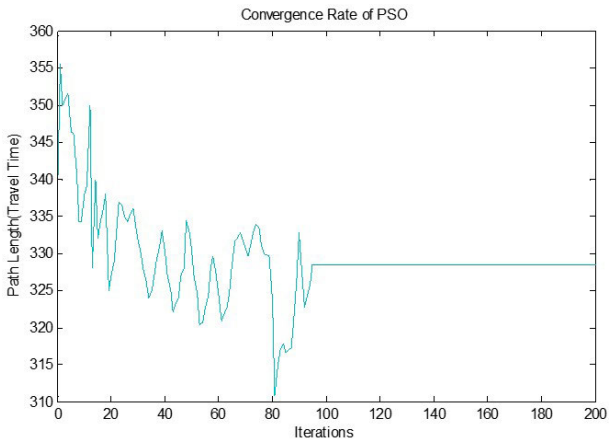

FIGURE 12. Convergence rate of PSO (NODE_ NUMBER = 100.

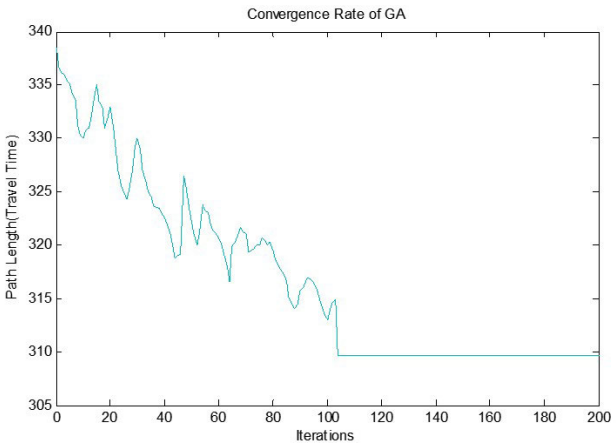

FIGURE 14. Convergence rate of GA (NODE_ NUMBER = 100).

of the algorithm that varies with the number of iterations. It can be found that all the compared algorithms have similar convergence trends. Although the optimal path obtained by each algorithm fluctuates in the previous iteration process, as the number of iterations increases, the optimal path of all algorithms gradually stabilizes. In terms of the number of iterations to reach convergence, compared with other algorithms, the RLDA algorithm proposed in this paper is close

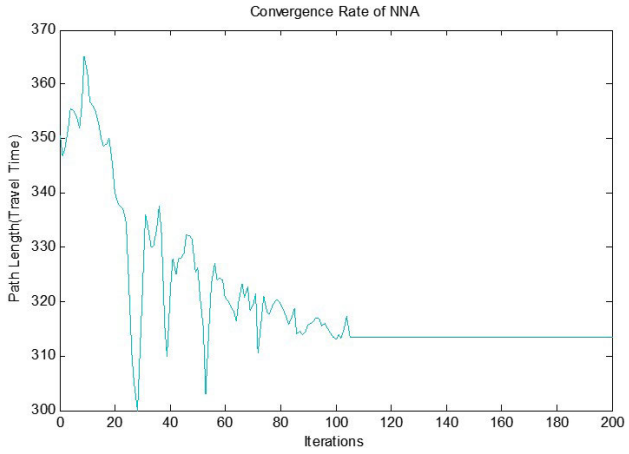

FIGURE 15. Convergence rate of NNA (NODE_ NUMBER = 100).

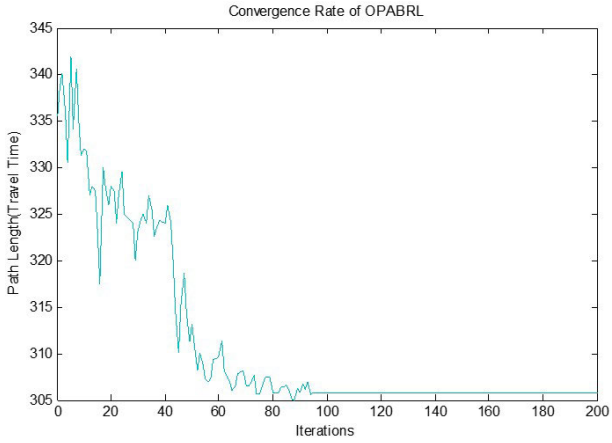

FIGURE 16. Convergence rate of OPABRL (NODE_ NUMBER = 100).

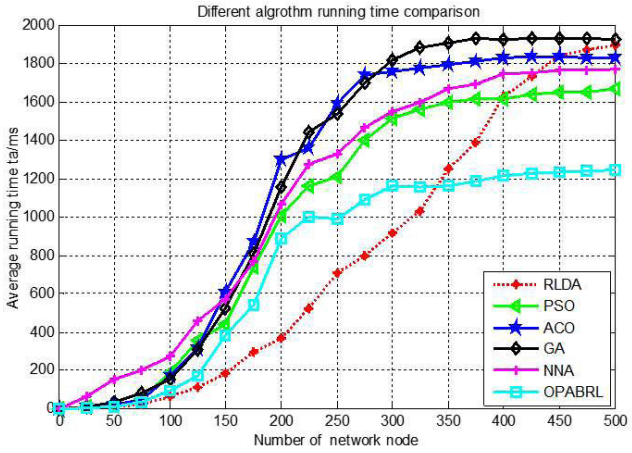

FIGURE 17. Running time of different algorithms.

to the particle swarm optimization (PSO) and the OPABRL algorithm based on machine learning methods [4], but significantly smaller than the other three algorithms. In addition, the ant colony optimization shows a transient fluctuation after convergence. Therefore, under the same experimental configuration, the RLDA algorithm can converge to the optimal path earlier and has better stability.

Experiment Result 3: In terms of the computational time of these algorithms, the average running time (in millisecond) of each algorithm under each network scale, as shown in Figure 17, it is observed that: 1) As the network scale increases, the growth rate of the running time of all considered algorithms all show a trend of fast first and then slow, and finally tend to stay the same. 2) After the network scale reaches 300, the running time of the other five algorithms (PSO, ACO, NNA, GA, OPABRL) changes little with the growth of the network scale, while the calculation time of RLDA algorithm changes very little after the network size reaches 450. 3) When the number of network nodes is less than 350, the RLDA algorithm proposed in this paper has the smallest running time among all algorithms. When the network size is greater than 350, the OPABAL algorithm is outperform other algorithms. So, in terms of computation effort, the proposed RLDA algorithm in this paper is obviously better than other five compared algorithms when the network size is less than 350.

VI. CONCLUSION

There are many intersections and road segment in the urban traffic network. The time spent at intersections accounts for a large proportion of the entire travel time from the starting node to the destination, and should not be ignored. In this paper, we discuss the vehicle path planning problem considering the attributes of the intersection for the car navigation system. The consideration of node attributes is the main innovation of this paper. For the addressed problem, this paper proposes the reverse labeling Dijkstra algorithm (RLDA) to solve this problem based on the traditional Dijkstra algorithm, combined with breadth first search, stack and queue data structures. Regarding the RLDA algorithm, we give the specific steps of the algorithm, the modular pseudo-code of the algorithm, and conduct a lot of theoretical analysis, and analyze that the RLDA algorithm has a lower polynomial time complexity. Finally, we conducted two kinds of experiments. One is to intercept the actual traffic network to verify the effectiveness of the algorithm in searching for the optimal path. The other is to select ten groups of networks of different sizes and conduct extensive experiments to compare the convergence efficiency and calculation speed between RLDA and PSO, GA, ACO, NNA, OPABRL. The statistical results show that the convergence rate of the RLDA algorithm is better than that of ACO, NNA, and GA. When the number of network nodes is less than 350, the algorithm has the smallest running time.

REFERENCES

[1] B. W. Yang, "Research status and development trend analysis of intelligent transportation system," *China Plant Eng.*, vol. 2, no. 2, pp. 121–122, 2019.

[2] Q. T. Geng, "Study on the key technology of intelligent transportation system based on the theory of image recognition," Ph.D. dissertation, Dept. Comput. Sci. Eng., Jilin Univ., Changchun, Chin, 2016.

[3] B. Wei, "Modeling and simulation research on vehicular ad hoc networks of intelligent transportation system based on Internet of Things," Ph.D. dissertation, Dept. Comput. Sci. Eng., Lanzhou Jiaotong Univ., Lanzhou, Chin, 2017.
[4] X.-H. Liu, D.-G. Zhang, H.-R. Yan, Y.-Y. Cui, and L. Chen, "A new algorithm of the best path selection based on machine learning," *IEEE Access*, vol. 7, pp. 126913–126928, 2019.
[5] H. Zhang and X. Lu, "Vehicle communication network in intelligent transportation system based on Internet of Things," *Comput. Commun.*, vol. 160, pp. 799–806, Jul. 2020.
[6] Y. S. Chen, "Research on decision-making method of driving behavior in urban complex traffic situation of intelligent automobile," Ph.D. dissertation, Dept. Auto. Eng., Jilin Univ., Changchun, Chin, 2019.
[7] F. Zong, M. Zeng, W. Zhong, and F. Lu, "Hybrid path selection modeling by considering habits and traffic conditions," *IEEE Access*, vol. 7, pp. 43781–43794, 2019.
[8] F. Hong, "Research on comprehensive cognition of traffic vehicles and virtual test for intelligent vehicle," Ph.D. dissertation, Dept. Auto. Eng., Jilin Univ., Changchun, Chin, 2018.
[9] H. Geng, "Research on path optimization of distributed dynamic route guidance system of intelligent transportation system," M.S. thesis, Dept. Mech. Eng., Lanzhou Jiaotong Univ., Lanzhou, Chin, 2016.
[10] P. Liu, "Research on modeling and optimization algorithm of dynamic route guidance system in intelligent transportation system," Ph.D. dissertation, Dept. Auto. Eng., Jilin Univ., Changchun, Chin, 2017.
[11] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Math.*, vol. 1, no. 1, pp. 171–269, 1959.
[12] Y. L. Chen, L. Y. Zhuang, L. B. Zhu, X. J. Shao, and H. Wang, "Research on path planning of parking system based on the improved Dijkstra algorithm," *Modern Manuf. Eng.*, vol. 6, no. 7, pp. 63–67, 2017.
[13] Y. W. Liu and H. Q. Wu, "Path planning based on theoretical shortest distance variable weight A* algorithm," *Comput. Meas. Control*, vol. 26, no. 4, pp. 175–178, 2018.
[14] X. F. Zuo and W. J. Shen, "Improved algorithm about muti-shortest path problem based on floyd algorithm," *Comput. Sci.*, vol. 44, no. 5, pp. 232–267, 2017.
[15] H. Kim, S.-H. Kim, M. Jeon, J. Kim, S. Song, and K.-J. Paik, "A study on path optimization method of an unmanned surface vehicle under environmental loads using genetic algorithm," *Ocean Eng.*, vol. 142, pp. 616–624, Sep. 2017.
[16] S. X. Wang and Z. X. Wu, "Improved Dijkstra shortest path algorithm and its application," *Comput. Sci.*, vol. 39, no. 5, pp. 223–228, 2012.
[17] Y. L. Liu, Y. Li, J. L. Wu, and X. Meng, "Route planning of expressway emergency evacuation based on improved Dijkstra algorithm," *Transp. Res.*, vol. 2, no. 6, pp. 54–66, 2016.
[18] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968.
[19] Y. Song and Z. M. Wang, "Path planning simulation based on improved A* algorithm," *J. Changchun Univ. Technol.*, vol. 40, no. 2, pp. 138–141, 2019.
[20] Y. F. Ge, T. Chen, X. Y. Kong, and L. Q. Gao, "Application of improved ant colony algorithm in car navigation," *Control Eng. China*, vol. 23, no. 1, pp. 133–137, 2016.
[21] Z. Y. Shang, J. N. Gu, and J. P. Wang, "An improved simulated annealing algorithm for the capacitated vehicle routing problem," *Comput. Integr. Manuf. Syst.*, vol. 7, no. 5, pp. 87–102, 2020.
[22] M. A. Mohammed, M. K. Abd Ghani, R. I. Hamed, S. A. Mostafa, M. S. Ahmad, and D. A. Ibrahim, "Solving vehicle routing problem by using improved genetic algorithm for optimal solution," *J. Comput. Sci.*, vol. 21, pp. 255–262, Jul. 2017.
[23] W. Y. Zhi, "Improved GA with particle swarm's evolutionary strategy for solving constrained optimization problems," *Control Decis.*, vol. 16, no. 4, pp. 135–147, 2012.
[24] Y. Kao, M.-H. Chen, and Y.-T. Huang, "A hybrid algorithm based on ACO and PSO for capacitated vehicle routing problems," *Math. Problems Eng.*, vol. 2012, pp. 1–17, Jan. 2012.
[25] Y. F. Ma, F. Yan, and K. Kang, "An improved particle swarm optimization for simultaneous pickup and delivery vehicle routing problems with time windows under a fuzzy random environment," *Oper. Res. Manage. Sci.*, vol. 27, no. 12, pp. 73–83, 2018.
[26] J. Zhang, F. Yang, and X. Weng, "An evolutionary scatter search particle swarm optimization algorithm for the vehicle routing problem with time windows," *IEEE Access*, vol. 6, pp. 63468–63485, 2018.
[27] S. H. Li, J. Q. Sun, and Z. Yang, "Key technologies of car navigation service recomendation system based on context awareness," *Proc. 31st China Control Conf.*, 2012, pp. 7298–7303.
[28] P. A. Mohammad, H. A. Mohammad, and B. Mehdi, "Application of GA, PSO, and ACO algorithms to path planning of autonomous underwater vehicles," *J. Mar. Sci. Appl.*, vol. 11, no. 3, pp. 492–496, 2012.

DAN-DAN ZHU was born in 1988. She is currently pursuing the Ph.D. degree with the Tianjin University of Technology, Tianjin, China. She is also a Researcher with the Tianjin University of Technology. Her research interests include path planning and modeling and simulation of complex systems.

JUN-QING SUN was born in 1964. He received the Ph.D. degree from Nankai University, Tianjin, China. He is currently a Professor with the Tianjin Key Laboratory of Intelligence Computing and Novel Software Technology, Tianjin University of Technology. His research interests include the modeling and simulation of complex systems, optimization of supply chains and modern logistics, service systems, and service computing.