World Electric Vehicle Journal

MDPI

Article

A Lane Level Bi-Directional Hybrid Path Planning Method Based on High Definition Map

Bin Yang, Xuewei Song and Zhenhai Gao *

State Key Laboratory of Automotive Simulation and Control, College of Automotive Engineering, Jilin University, Changchun 130012, China; y17808083409@163.com (B.Y.); sxw@jlu.edu.cn (X.S.)
* Correspondence: gaozh@jlu.edu.cn

Abstract: A global reference path generated by a path search algorithm based on a road-level driving map cannot be directly used to complete the efficient autonomous path-following motion of autonomous vehicles due to the large computational load and insufficient path accuracy. To solve this problem, this paper proposes a lane-level bidirectional hybrid path planning method based on a high-definition map (HD map), which effectively completes the high-precision reference path planning task. First, the global driving environment information is extracted from the HD map, and the lane-level driving map is constructed. Real value mapping from the road network map to the driving cost is realized based on the road network information, road markings, and driving behavior data. Then, a hybrid path search method is carried out for the search space in a bidirectional search mode, where the stopping conditions of the search method are determined by the relaxation region in the two search processes. As the search process continues, the dimension of the relaxation region is updated to dynamically adjust the search scope to maintain the desired search efficiency and search effect. After the completion of the bidirectional search, the search results are evaluated and optimized to obtain the reference path with the optimal traffic cost. Finally, in an HD map based on a real scene, the path search performance of the proposed algorithm is compared with that of the simple bidirectional Dijkstra algorithm and the bidirectional BFS search algorithm. The results show that the proposed path search algorithm not only has a good optimization effect, but also has a high path search efficiency.

Keywords: high definition map; global reference path; travel cost model; bidirectional hybrid search; relaxation region

Citation: Yang, B.; Song, X.; Gao, Z. A Lane Level Bi-Directional Hybrid Path Planning Method Based on High Definition Map. *World Electr. Veh. J.* **2021**, *12*, 227. https://doi.org/10.3390/wevj12040227

Academic Editor: Joeri Van Mierlo

Received: 23 October 2021
Accepted: 8 November 2021
Published: 10 November 2021

1. Introduction

Given the source point and target point, a road-level driving map can help a vehicle select the optimal path, which can be used as a reference to guide human drivers to the destination [1]. Due to the insufficient data accuracy of road-level maps, the reference path generated from road-level driving maps is a driving task with prompts designed for drivers rather than a specific path that can be followed by an autonomous vehicle. Since the road-level reference path does not have high-precision position information, the guidance information provided for the automatic driving vehicle is too vague to be directly used for the motion control of the autonomous vehicle. In practice, autonomous vehicles mainly conduct the exploration and testing of local path planning, and lateral and longitudinal motion control based on recorded trajectories. This will make it difficult to actually deploy the algorithm in an autonomous vehicle. However, the trajectory planning method based on a high-definition map (HD map) can provide clear and delicate lane-level path guidance information for autonomous vehicles and improve the control performance [2]. HD map based environment information and trajectory planning method have also become a necessary demand for autonomous driving above L3 level [3,4].

There is still no universally accepted definition of concepts, such as lane-level maps or HD maps. However, the applications of these concepts indicate that HD driving maps

are more accurate and suitable for automatic driving tasks than traditional maps. To date, lane-level maps have attracted extensive attention from scholars. With the development of advanced data acquisition equipment, such as high-precision onboard positioning sensors, it is possible to establish an enhanced map for autonomous vehicles [5]. Liu et al. [6] proposed an HD map with 4-level data logic structure to assist the autonomous vehicle with the task of "perception-planning and policy decision-control". Guo et al. [7] proposed a lane-level map generation method based on sensors and Open Street Map (OSM). Jiang et al. [8] proposed a seven-layer automatic driving map structure model and completed the global reference path search for the structure model with the A* algorithm. Li et al. [9] also realized the global reference trajectory planning with A* algorithm based on the HD map in the park scenario, and established the motion control algorithm and control strategy of the lane keeping assist system. Currently, the research on lane-level maps mainly focuses on the high-precision geometric representation of lanes. There are relatively few studies on the path search problem of lane-level maps for automatic driving, and the path search methods used are relatively traditional.

Global path search methods based on road network map or HD map can be approximately divided into depth-first search and breadth-first search [10]. Depth-first search is a continuous detection method along the depth direction, such as the A* algorithm [11,12], screening newly discovered nodes based on greedy ideas. This method has a significant advantage in search efficiency. However, due to the Greedy Best First Search mode, the depth-first based search algorithms do not guarantee completeness (i.e., finding the optimal path). However, the global trajectory search problem based on road network or HD map has a high requirement for the global optimum. The search process should cover as much search space as possible to determine the optimal path. Therefore, the path planning method based on depth-first or heuristic search is not suitable for solving the trajectory planning problem in sparse road network environments. The breadth-first algorithm is a method that extends the boundary between the found node and the undetected vertex layer by layer in the direction of breadth. The purpose is to systematically expand and check all the nodes in the graph to find the optimal trajectory planning results (Dijkstra algorithm [13], BFS algorithm [14], etc.). The advantage of this method is that it can read and compare most of the network node information extensively and can ensure global optimization very well. Additionally, due to large-scale node search inevitably will lead to inefficient search and heavy computing load. The actual application process will bring an excellent calculation burden to the onboard controller of the autonomous vehicle.

Lane-level path planning is more challenging than road-level path planning, which manifests mainly in two aspects: the construction of a driving cost model and the optimization of the efficiency of path planning. The traditional road network structure is usually simplified as the shortest path problem in graph theory. Relevant scholars have conducted extensive research on road-level path planning algorithms. Liu et al. [15,16] constructed the popular transition graph structure based on the driving trajectory data, and realized the preference-based path selection and planning based on the popular paths and path differences. Yuan et al. [17] used the connectivity and the commuting capacity of the road as the weight of the road network, and realized the emergency route planning by the analytic hierarchy process method. However, in practical autonomous driving projects, these algorithms need to be adapted to include vehicle driving conditions in the traffic cost and take into account the specific properties of the actual road network. In addition, optimizing the search space of the path search algorithm can significantly improve the efficiency of path planning. The bidirectional search algorithm is derived from the traditional path search algorithm. Applying the path search simultaneously at the source point and the target point can greatly improve the efficiency of the path search [18]. However, the efficiency and optimality of bidirectional search are affected by the computational logic and termination conditions, so the combination of multiple algorithms is required to improve the query efficiency of the algorithm.

The goal of this research is to develop an automatic driving-oriented path search algorithm based on lane-level maps and to improve the search efficiency of the point-to-point directed path search algorithm under the premise of satisfying the global optimally requirements. This paper proposes a bidirectional hybrid path search (BHPS) method based on an HD map. A logical schematic diagram of the BHPS method is shown in Figure 1. First, the global driving environment information is extracted from the HD map, and a lane-level environment map is constructed. Then, the real value mapping from the road network map to driving cost is realized according to the road network information, road markings, and driving behavior data. This content is introduced in detail in Section 2 of this paper. Next, "forward BFS search-reverse Dijkstra search" and "forward Dijkstra search-reverse BFS search" are conducted in the search space in a bidirectional search mode. The stopping conditions of the bidirectional search are determined by the corresponding relaxation region in the search process. As the search process continues, the dimensions of the relaxation region are updated, and the search range is dynamically adjusted to maintain the ideal path search effect. After the hybrid bidirectional search is completed, the search results are evaluated and optimized to obtain the global near-optimal reference path. The detailed design process is described in Section 3 of this paper. In Section 4, we carry out comparative tests on and verify the algorithm proposed in this paper based on real HD maps, and we summarize the research in Section 5.

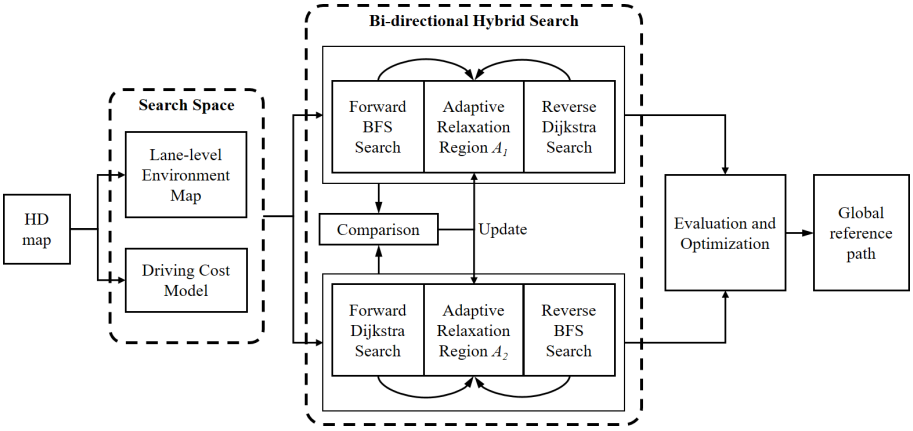


Figure 1. Logical schematic diagram of the bidirectional hybrid path search method.

2. Driving Cost Model Based on Lane-Level Maps

When road network information is very complex, the current technology cannot understand such complicated traffic scenes at the algorithm level. As a static sensor with a three-dimensional precision, comprehensive content, high precision, high robustness, and long-range, HD maps play an indispensable role in the whole process of automatic driving positioning, perception, decision-making, and motion control. HD maps are not only navigation maps but also powerful "sensors" that provide detailed environmental information in the process of vehicle driving [4]. HD maps are not specific about the accuracy but are also a more comprehensive, accurate, and clear description of the vehicle's driving environment. High-precision maps not only store detailed globally referenced road network and lane-level information but also provide a unique overall view of landmarks or the surrounding environment to assist vehicles in path planning tasks. In addition, HD maps contain considerable driving assistance information, such as the layout of intersections and the locations and details of traffic signs, and specific information, such as road speed limits, where to start steering, and specific traffic rules.

An HD map contains abundant information on the road environment. However, autonomous vehicles do not require all high-precision map content for route planning and motion control. In the process of compiling lane-level road network data, through the extraction and integration of HD map information, the lane-level road network, including

road-level road network information, road-lane connection data, and lane-level road network data, is obtained.

2.1. Lane-Level Network Structure

Road-level network information includes lane boundaries, directional lane centerline information, and other information [19]. Without loss of generality, we consider that the road network consists of a group of roads $\{r_j\}_{j=1}^{a}$ and $\{c_i\}_{i=1}^{b}$ a group of intersections, where a and b represent the number of roads and the number of intersections, respectively. The road layers can usually be expressed as:

$$W = (\{r_j\}_{j=1}^{a}, \{c_i\}_{i=1}^{b}) c_i = (P_{c,i}, E_{c,i}, T_{c,i}) \tag{1}$$

where W represents the road layer, $P_{c,i}$ represents the set of road-level nodes entering the ith intersection, $E_{c,i}$ represents the set of road-level nodes leaving the intersection, and $T_{c,i}$ represents the road-level traffic matrix between the two nodes.

$$T_i = \begin{matrix} \\ E_{c,j,1} \\ \vdots \\ E_{c,j,m} \end{matrix} \begin{matrix} \begin{matrix} P_{c,j,1} & \cdots & P_{c,j,n} \end{matrix} \\ \begin{bmatrix} t_{i,1,1} & \cdots & t_{i,n,1} \\ \vdots & \ddots & \vdots \\ t_{i,1,m} & \cdots & t_{i,n,m} \end{bmatrix} \end{matrix} \tag{2}$$

where m and n are the number of nodes leaving and entering the intersection, respectively, $t_{i,n,m} = (f_i, m_i)$ refers to whether the vehicle can reach the exit node $E_{c,j,m}$ from driving node $P_{c,j,n}$. f_i refers to whether the vehicle can arrive at node $E_{c,j,m}$ from node $P_{c,j,n}$, and m_i represents the road state, including a steering section, straight section or U-shaped bend section. The road-level network is shown as follows:

$$r_i = (P_{r,i}, E_{r,i}, Q_{r,i}) \tag{3}$$

where P_r is the set of road-level nodes entering the intersection, E_r is the set of road-level nodes leaving the intersection, and Q_r includes road category and road length.

The lane-level road network data provide lane-related information, such as high-precision road geometry, traffic rules, and road markings in sections. Combined with the connection data of roads and lanes, the optimal path at the lane level considering road curvature, traffic facilities, and traffic rules can be obtained.

The lane-level road network contains geometric and topological details of lane-level sections and intersections. Researchers have proposed a variety of lane-level map models that represent detailed lane information. The geometric information of lanes is usually represented by Bézier curves, clothoid curves, spline curves, or cubic Hermite splines [20], while road intersections are described by a topological structure.

The connection data between roads and lanes are mainly used for lane-level path optimization. The topological connection between the road-level network and lane-level network is provided by mapping the real values of network information and lane information. This maps the road-level network topology to the lane-level network. This step is the key to ensuring the flexibility and accuracy of the overall map structure. It provides a way to load HD data and retains the advantages of traditional maps.

2.2. Traffic Cost Model Based on Lane-Level Maps

To improve the vehicle traffic efficiency of the global planning path, the path search process with consideration of the traffic information and driving habits of drivers. Therefore, the driving cost model based on the high-precision map is established using real driving data.

Generally, drivers have different expected speeds on straight road sections and steering sections without considering the interference of deceleration belts, sidewalks, and other traffic facilities. On straight road sections, drivers tend to travel at a higher speed to

improve traffic efficiency. However, on a steering section, to maintain the vehicle stability and driving comfort of the vehicle in the process of driving, drivers generally travel at a lower speed. Therefore, historical records of average speeds based on driving data can be used as empirical data to estimate travel time costs and to predict future driving speeds.

In this study, we collect driver data from some sections of the road. It is unreasonable to use the average speed of the road section as the expected speed of the straight road section while collecting the driving data due to the existence of traffic facilities. Therefore, we extract the speed sets with the highest frequency for each road section and take the average value of these speed sets as the expected speed for the road section.

2.2.1. The Travel Time Cost of the Path

The weighted digraph is described by $G = (V, E)$, where V is the set of nodes and E is the set of edges in the digraph. In G, each edge has a corresponding weight, which is determined by the weighting function ω. For each edge $(u, v) \in G$, the weighting function ω: $E \rightarrow R$ represents the mapping from edge to real weight values. Assuming that the vehicle is driving at a constant speed in the lane, the travel time of the path section can be easily estimated according to the length of the path section and the expected speed:

$$\omega_{const} = \frac{\lambda(u, v)}{v_{\exp}} \tag{4}$$

where λ denotes the travel distance of the path section and v_{exp} represents the expected speed of the path section. However, in actual driving scenarios, straight lanes are not always ideally long and straight. Deceleration belts, sidewalks, and other sections requiring deceleration often appear along a straight lane. Therefore, in the path section of the HD map, the speed of vehicles is not uniform in the whole section; the road section necessitates acceleration and deceleration. In addition, it is inevitable that vehicles slow down when entering an intersection or a roundabout. Therefore, it is necessary to conduct a more scientific evaluation of the travel time cost of straight sections based on the road information contained in the HD map.

When there is a curvature change between two path segments, there is a variable-speed section with a non-zero acceleration value at the junction of the two segments due to the difference in the expected speed under the different curvature sections. A typical set of path segments is shown in Figure 2, where Path 1 and Path 3 are long, straight road sections and Path 2 is a steering segment between the two straight segments. Because of the speed difference between Path 1 and Path 2, the speed decreases on Path 1 as it approaches Path 2 to match the expected speed of Path 2. Therefore, the travel time cost of the path segment is obtained by adding the travel time cost components of the constant-speed segment and the variable-speed segment:

$$\omega_{\lambda 1} = \omega_{c1} + \omega_{t1} = \frac{2a_t\lambda_1(u, v) - (v_1 - v_0)^2}{2a_t v_1} + \frac{(v_1 - v_0)^2}{2a_t v_0} \tag{5}$$

where c_c represents the cost of travel time for the constant-speed section on Path 1, $c_{\lambda t}$ represents the time delay caused by deceleration, and at represents the expected acceleration required to meet driving comfort requirements.

Path 2 is a steering segment through which the vehicle passes at a constant speed. Therefore, the travel time cost for this segment can be directly estimated:

$$\omega_{\lambda 2} = \frac{\lambda_2(u', v')}{v_0} \tag{6}$$

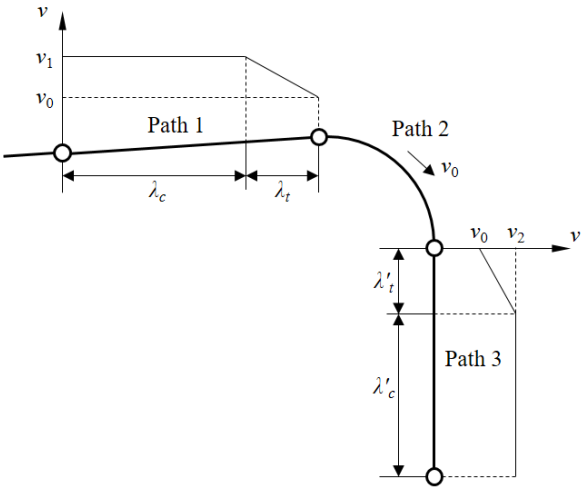


Figure 2. Schematic diagram of path segments.

Similar to Path 1, Path 3 is also composed of a constant-speed segment and a variable-speed segment. The travel time cost of the path segment can be calculated in a similar way:

$$\omega_{\lambda 3}=\omega_{t3}+\omega_{c3}=\frac{(v_2-v_0)^2}{2a_t v_0}+\frac{2a_t\lambda_3(u'',v'')-(v_2-v_0)^2}{2a_t v_2} \tag{7}$$

2.2.2. Time Cost Increases Generated by Transportation Facilities

Affected by traffic rules and vehicle passibility, a corresponding deceleration action is required when the vehicle approaches traffic facilities, such as deceleration belts and sidewalks. This process causes a delay in the transit time compared to a constant speed. The impact of deceleration belts on travel time is analyzed, as shown in Figure 3. The area affected by deceleration belts is shown in gray on HD maps.

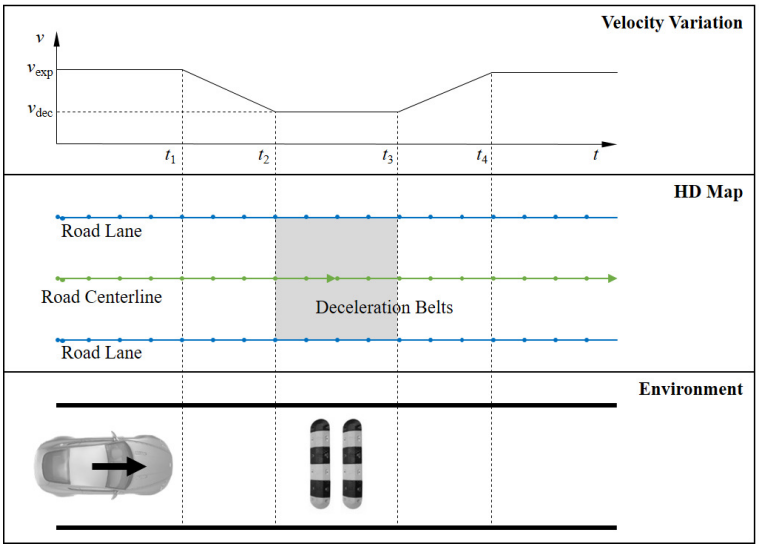


Figure 3. Schematic diagram of deceleration belt path section.

The vehicle needs to slow to a steady speed v_{dec} before entering the area affected by deceleration belts and accelerate to the desired speed in the path section after leaving the area of the acceleration area. The resulting time cost increase is as follows:

$$\Delta\omega_{dec}=\frac{\lambda_{dec}(u,v)}{v_{dec}}+2\frac{v_e-v_{dec}}{a_t}-\frac{v_e^2-v_{dec}^2+a_t\lambda_{dec}}{a_t v_e} \tag{8}$$

Therefore, in a path segment containing the deceleration belt, the travel time cost of the path is:

$$\omega_\lambda = c_\lambda + N \cdot \Delta\omega_{dec} \quad (9)$$

where N represents the number of deceleration belts contained along the path segment.

3. BHPS Algorithm Based on HD Maps

The BHPS algorithm based on HD maps consists of three parts: bidirectional hybrid search in the search space, update of the relaxation region, and global path optimization. First, the global path search is carried out in the search space simultaneously with the source point and the target point. A relaxation region is designed to adjust the path search termination condition of the hybrid search method. After that, the hybrid path search results are evaluated and optimized comprehensively, and the global path search is finally completed.

3.1. Bidirectional Hybrid Search in Search Space

Global path search has high requirements for global optimization. The search process should cover as much search space as possible to determine the optimal path. In the design of path search mode, we choose the breadth-first search method with high completeness to complete the search task of the global reference trajectory. Therefore, the algorithm proposed in this paper is based on width first and related algorithms. However, the global path of traditional Dijkstra search can not guarantee the demand for global optimality. In contrast, BFS search takes up a lot of computing resources and has low search efficiency, which brings a significant burden and test to on-board computing equipment. Therefore, to ensure the completeness of trajectory search, reasonably reduce the search space and improve the search efficiency in the global trajectory search link, we designed a two-way hybrid trajectory search method to achieve fast and near-optimal trajectory search tasks.

3.1.1. BFS Search Process

Given a digraph $G = (V, E)$ and source nodes, the breadth-first search method searches the connected edges of s in G to find all nodes that can be reached from s. The distance between s and all these accessible nodes (i.e., the minimum cost sum) is calculated. The search algorithm generates a breadth-first tree with roots s and all the reachable nodes of s. For any node v reachable from s, the path from s to V in the breadth first tree corresponds to the shortest path from s to V in digraph G, i.e., the lowest cost expected path. Since the data structure of the lane-level map is sparse, this paper uses an adjacency table to store the road data.

For breadth-first search, several additional data structures are used to store the relevant nodes. All sub-nodes resulting from the expansion of the node are added to a first-in, first-out queue. Neighbor nodes that have not been checked are placed in the open-list queue. Open-list O is used to store nodes connected to node u found by the adjacency table as the priority queue for subsequent searches. Each node $u \in V$ that completes the search is placed in the close-list queue. Close-list C can also be thought of as a breadth-first tree of completed searches. The travel time cost between the source node s and node u calculated by the BFS algorithm exists in the variable d [u]. The search process of the BFS algorithm is shown in Figure 4.

First, the source node s is taken as the root node in C. Then, the root node is used as the parent node u in the adjacency table, and the result is added to O. After that, node v is gradually extracted from O, and the cost of the path segment corresponding to node (u, v) is added to the path sequence. In this process, v is constantly added to queue Q as the parent node of the next layer search. The path search and cost calculation steps are repeated until the search reaches the target point or the search of all the nodes in the digraph is completed.

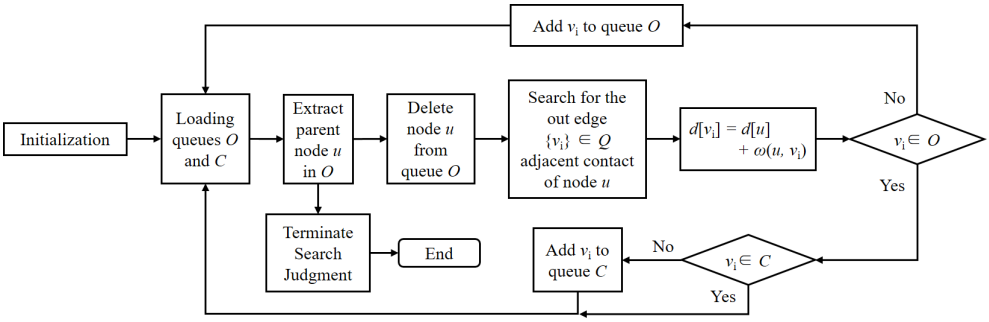


Figure 4. Logic diagram of BFS search process.

The BFS algorithm calculates the path cost from the source node to any nodes in the graph to systematically expand and check all the nodes in the graph to find the optimal result. In other words, the BFS algorithm does not consider the possibility of local search results, but thoroughly searches the entire search space. Therefore, as the search progresses, the number of parent nodes increases exponentially. The search operations therefore increase, which easily results in an expansion of the search space. Inevitably, there is a large computational load.

3.1.2. Dijkstra Search Process

The Dijkstra algorithm is a typical path search algorithm that is used to solve the shortest path problem of a single source on a digraph with weights (without a negative weight edge). Similar to the BFS algorithm, the Dijkstra algorithm also searches sub-nodes in a breadth-first manner during a local search. The difference is that the Dijkstra algorithm takes the lowest cost node as the parent node of the next-level search, compared with the BFS algorithm, which takes all nodes as the parent node for the subsequent search. The logic diagram of the Dijkstra algorithm is shown in Figure 5.

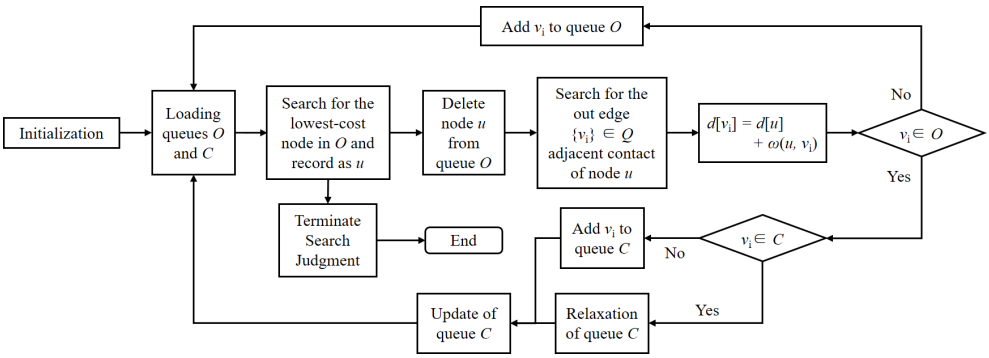


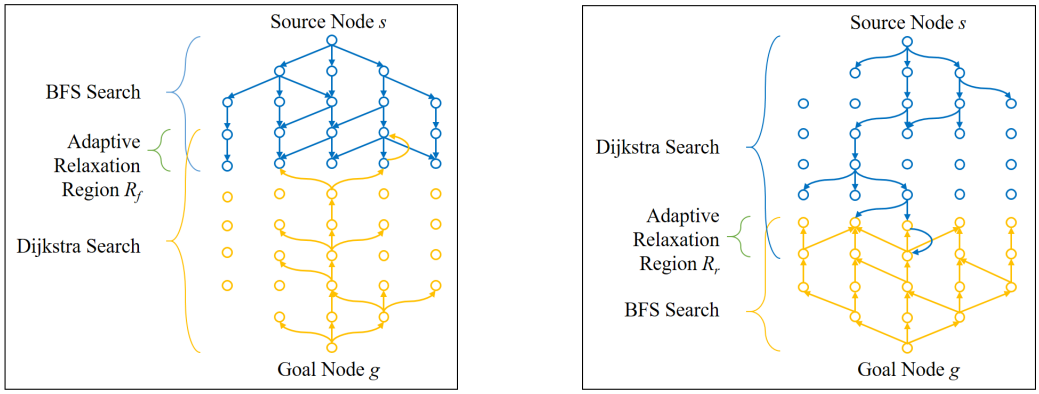
Figure 5. Logic diagram of Dijkstra search process.

When the algorithm is initialized, C contains only the source node s, and s has a travel time cost of 0. The nodes on the next level are searched using the adjacency table and recorded in queue O except for v. Then, the travel time cost $\omega(s, v)$ of the nodes is calculated and updated. All the outgoing edges v_i of u are relaxed. If the travel time cost (through node u) from s to node v_i is lower than the original travel time cost (without going through node u), $\omega(s, v_i)$ is replaced with this lower weight. After that, the node with the lowest travel cost is added to C and chosen as the parent node of the next layer search. Then, node searching and relaxation are repeated until the goal node is found.

Since the Dijkstra algorithm always selects the node with the lowest travel cost to insert into queue C during the update process of O, it can be considered that node relaxation is implemented based on a greedy strategy. Therefore, in the path search process under a complex HD map, as a result of this greedy strategy, the path search result is not always the optimal result in the global sense [21]. The Dijkstra algorithm can only find the shortest path in the local search space.

3.1.3. Bidirectional Hybrid Search Process

Bidirectional hybrid search refers to the forward and reverse search in the search space by BFS algorithm and Dijkstra algorithm, respectively. Then, the search results are fused and optimized to obtain the optimal travel cost. To avoid the redundant operation of BFS search and Dijkstra search simultaneously in the same network vertex, BFS search and Dijkstra search are not carried out completely at the same time. In the process of searching, Dijkstra search is preferred when BFS search and Dijkstra search are carried out with the same vertex, while BFS search borrows the Dijkstra search results to avoid the repeated searching and calculation of the vertex of the network. A logic diagram of the bidirectional hybrid search process is shown in Figure 6. Bidirectional hybrid search is implemented through two "BFS-Dijkstra" searches. The difference between the two search processes is that the search direction and the relaxation region are not consistent.



(**a**) Forward hybrid search (**b**) Reverse hybrid search

Figure 6. Logic diagram of bidirectional hybrid search process.

As shown in Figure 6a, a breadth-first search is carried out on the search space with the source node s as the initial search point in the forward direction from s to the goal node g. Meanwhile, a reverse Dijkstra search is carried out with g as the initial search point and in the backward direction from g to s. In the process of searching in two directions, the nodes of the search space intersect, and these intersecting nodes are recorded as the intersecting nodes p_i. When the intersection node appears during the search process, the search process is considered to have entered a relaxation phase. The search process terminates after a period of dynamic space searching in the relaxation phase. The intersecting nodes and relaxation region are designed to improve and optimize the search efficiency and effect of the globally optimal path.

After that, the search continues in both directions until p_i exceeds the range of the relaxation region. At this point, the path travel cost between nodes p_i in the relaxation region and s is found by BFS. Similarly, an optimum local path is found between the target point t and the relaxation region by Dijkstra search.

Similarly, a path search of the search space with the opposite mode in the direction from s to g is conducted, as shown in Figure 6b. That is, Dijkstra search of the search space is carried out from s to g, while reverse BFS is carried out in the direction from g to s. Since the road network information is not necessarily symmetric based on s and g, the complexity of the road network near the node s and the node g is not identical. Moreover, the search efficiency of the Dijkstra algorithm is different from that of the BFS algorithm. Therefore, when intersecting nodes occur in the bidirectional search process, two different relaxation regions are generated. The relaxation regions of the two search processes were recorded as R_f and R_r.

Through the above search process, we can obtain the following search results in the search space:

(i) Dijkstra search space and Dijkstra path from the source node s to the relaxation region R_f and from the goal node g to the relaxation region R_r;
(ii) The BFS space near the source node s and the goal node g;
(iii) Local Dijkstra search results between the relaxation regions R_f and R_r.

Therefore, in the search space of the proposed algorithm, at least one global path from s to g could be found in the expected path exists. The fact that all the above results are generated by the Dijkstra algorithm and BFS algorithm indicates that the path is globally optimal or globally near-optimal.

A comparison of the proposed algorithm with the bidirectional Dijkstra and bidirectional BFS algorithms in terms of the search scope is shown in Figure 7. The bidirectional BFS algorithm can solve the problem of a high calculation load well, and it still maintains a large search scope and a high computational complexity [22]. Although the bidirectional Dijkstra algorithm has great advantages in the search scope and search efficiency, it cannot guarantee that the path searched is globally optimal due to the limitations of the search scope and logic. The proposed algorithm is between the bidirectional Dijkstra algorithm and the bidirectional BFS algorithm in terms of the search range and algorithm complexity. The proposed algorithm can be regarded as the complement of the bidirectional Dijkstra algorithm in the search scope and the search effect, and can be regarded as the optimization of the search scope and search efficiency of the bidirectional BFS algorithm. In addition, the proposed algorithm adds relaxation regions to the search process, which can effectively avoid the severe rise in the computational load caused by the drastic increase in the number of search nodes.

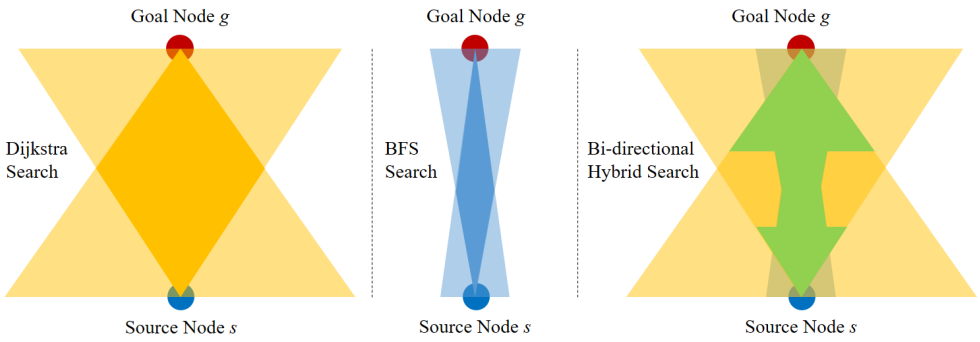


Figure 7. Comparison of search scopes.

3.2. Design and Update of Relaxation Regions

In an actual complex road network, affected by the contingency of the greedy strategy, the Dijkstra algorithm is likely to quickly find the desired path segments, resulting in BFS only completing a small range of path searches. This will result in a large search gap between the forward and reverse BFS search scopes in the hybrid search process, which greatly reduces the optimization performance of the algorithm. Therefore, the self-adaption relaxation region is introduced in the search process to improve the optimization ability of the algorithm on the premise of ensuring search efficiency.

The relaxation region is located around the intersection nodes of the bidirectional hybrid search process, which is the basis for judging the termination conditions of the forward and backward BFS and Dijkstra search processes. The application and updating process of the relaxation region is shown in Figure 8.

The initial value of the relaxation region dimension varies with the size of the search space. The larger the dimension of the map is, the larger the corresponding search space and the larger the initial value of the relaxation region dimension. This means that the relaxation region increases with an increase in the map dimension to better adapt the path search under different maps.

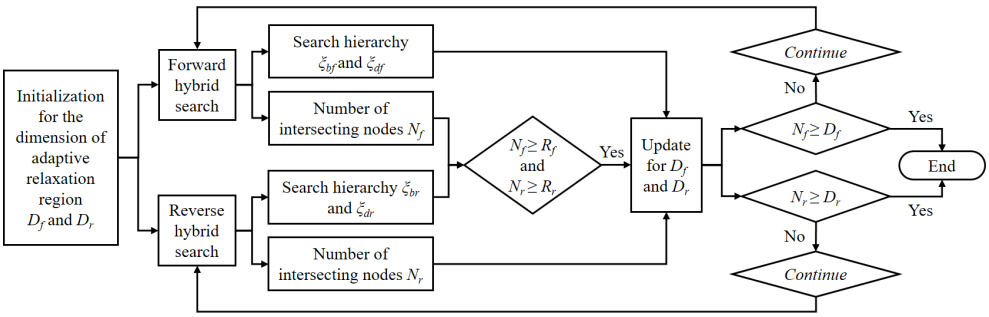


Figure 8. Relaxation region dimension updating process schematics.

As the bidirectional search process progresses, it is inevitable that the two directions of searching will intersect. As the search continues, the number of intersecting nodes increases. When the number of intersections reaches the dimension of the relaxed area, the bidirectional search related to the dimension of the initial relaxation region is completed, and the search is suspended. At this point, the ratio ϵ_i of the number of layers in the BFS scope to the number of layers of the full search can be calculated.

$$\epsilon_i = \frac{\xi_{bi}}{\xi_{bi} + \xi_{di}}, i = f, r \tag{10}$$

where ξ_{bi} and ξ_{di} represent the number of layers searched by the BFS and Dijkstra algorithms in hybrid search, respectively.

ϵ_i directly characterizes whether there exists a large search gap between the search scope sets of bidirectional BFS. A small ϵ_i value indicates that there is a large search gap between the forward and backward BFS processes, and the relaxation region dimension needs to be increased appropriately to improve the global optimally of the path. $\epsilon_i << 1$ indicates that the bidirectional BFS process only searches the search space locally. It is difficult to ensure the optimally of the path search results. Therefore, the relaxation region dimension should be increased appropriately to improve the search effect of the path. When ϵ_i approaches 1, it indicates that the bidirectional BFS process almost covers the entire search space, while the bidirectional Dijkstra search process cannot effectively find the desired path, which ensures the completeness of path search to a certain extent. After calculating and updating the dimensions of the relaxation region, the search continues based on the above bidirectional hybrid search.

4. Global Path Optimization

In the process of bidirectional hybrid search, two search processes are carried out, involving two local BFS and two Dijkstra searches. It is necessary to make combinations and logical judgments of the optimal paths in the search scope formed by the search results. In the process of bidirectional hybrid search, bidirectional Dijkstra searches intersect in the search range and inevitably pass through the BFS scope. Therefore, the Dijkstra search results are used as the reference for global paths, and the BFS search results are combined with restructuring the optimized paths. However, the proposed algorithm does not carry out a complete bidirectional Dijkstra search. In terms of the search results, two possibilities appear in the forward and reverse Dijkstra search results:

(i) The forward Dijkstra search and reverse Dijkstra search have overlapping vertices outside the BFS search scope;
(ii) The forward Dijkstra search and reverse Dijkstra search have no overlapping vertices outside the BFS search scope;

4.1. Existing Overlapping Nodes

A schematic diagram of existing overlapping nodes is shown in Figure 9a. First, the Dijkstra optimal path P_d is generated by forward and reverse Dijkstra search processes.

$$P_d = Enqueue(P_{df}, P_{dr}) \tag{11}$$

The path is obtained by direct splicing of the road section P_{df} from s to the intersection point in the forward Dijkstra search process and road section P_{dr} from the intersecting node to g in the reverse Dijkstra search process.

However, the path obtained by bidirectional Dijkstra search is not always the globally optimal path. In other words, P_d is not necessarily the optimal path in the search scope. P_d is derived from the bidirectional hybrid search results, and the search scope of the algorithm also contains BFS regions. Therefore, non-optimal path segments appear only in the BFS search scope. Considering that P_d inevitably contains the intersection nodes in the forward relaxation region and the reverse relaxation region, the globally optimal path can be found by replacing the non-optimal path segment.

Therefore, it is necessary to replace the path segment between s and the intersection node of the forward relaxation region in P_d, and the intersection node of the reverse relaxation region and g with the BFS search result. In this way, the optimal path can be found in the search scope of the proposed algorithm.

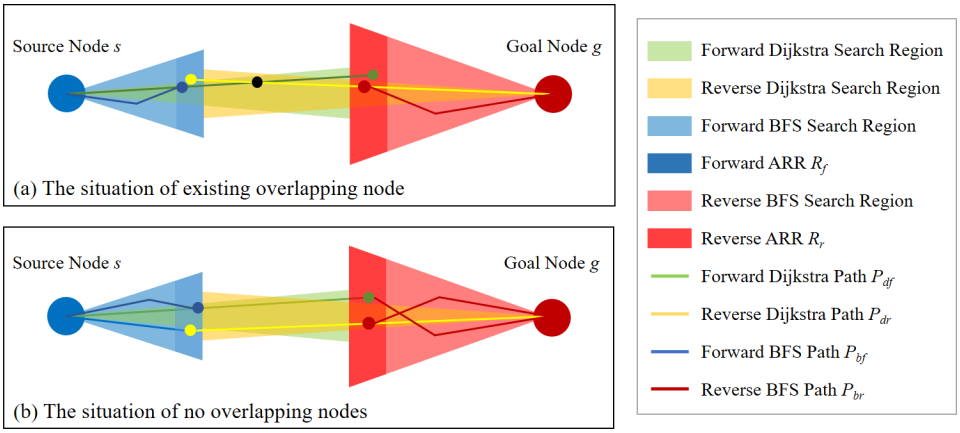


Figure 9. Schematic diagram of global path optimization.

4.2. With No Overlapping Nodes

Figure 9b shows the schematic diagram when there are no overlapping nodes outside the BFS search scope. The bidirectional Dijkstra search process generates two expected paths: forward Dijkstra search path P_{df} and reverse Dijkstra search path P_{dr}. Taking forward Dijkstra search path P_{df} as an example, the path segment does not reach g but the optimal path between source node s and the reverse relaxation region. For P_{df} to reach g, the path from the reverse relaxation region to g is supplemented by the reverse BFS optimal path. The Dijkstra optimal path from s to the forward relaxation region is also replaced with the forward BFS optimal path.

In this way, we obtain the first preferred path P_{pf} through a forward hybrid search. Similarly, based on the reverse Dijkstra search path, P_{dr} can also lead to a preferred path P_{pr}. Finally, the path travel cost of P_{pf} is compared with that of P_{pr}. The path with a lower travel cost is the optimal path in this situation.

5. Simulation Test and Analysis of Results

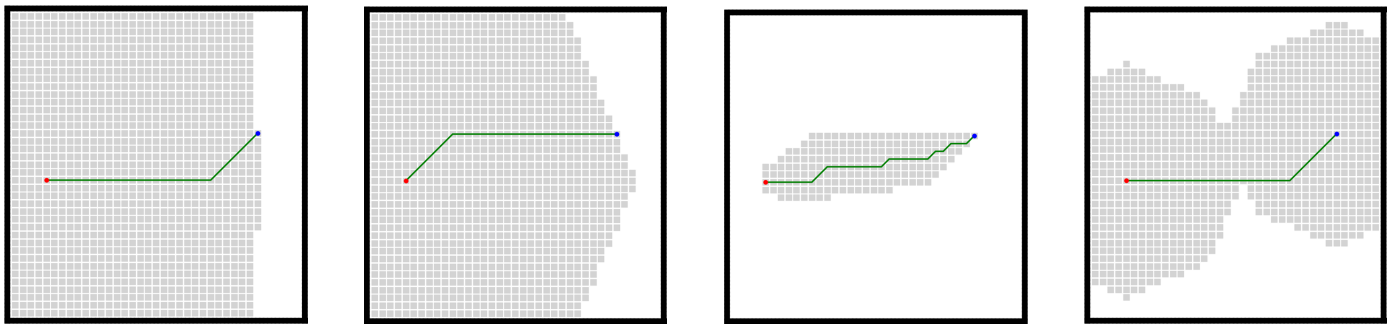
5.1. Verification of Path Search Algorithm

Firstly, we make an intuitive comparison and analysis between the trajectory search algorithm proposed in this paper and the traditional trajectory search algorithm to ver-

ify the advantages of the proposed algorithm in terms of computational efficiency and search completeness.

We conducted comparative simulation tests in undirected structured scenes to comprehensively analyze and identify the algorithms proposed in this paper from the design starting point, path search efficiency, and global search effect. For the existing standard trajectory planning algorithms, we selected the A* algorithm, BFS algorithm, and Dijkstra algorithm as a comparison. The path search effects in the same test scenario are compared and analyzed. The A* algorithm is a widely used heuristic trajectory search algorithm. Because of its efficient searchability, a star algorithm is primarily used in real-time trajectory planning scenes, such as obstacle avoidance and autonomous parking. BFS algorithm has been widely theoretical derivation and experimental analysis by relevant scholars, and its search results have been proved to be complete and optimal [22]. In the search problem of the reachable path, the BFS search result must be the optimal global solution. Therefore, the BFS search results can be used as the reference terminal of the optimal global solution to analyze the optimization ability of relevant search algorithms. The Dijkstra algorithm also has better performance in the global search effect optimality and higher search efficiency than the BFS algorithm. It is also widely used in international path planning based on road network information.

Figure 10 is a primary road network test scenario. In this scenario, the road network is evenly distributed and undirected without any scene interference. In this scenario, both BFS search and Dijkstra search have ample search space and high computational resources. The A* search is a quick way to find the ideal path. In contrast, the BHPS algorithm proposed in this paper has obvious optimization of search space compared with BFS search and Dijkstra search, although it also carries out extensive node search.



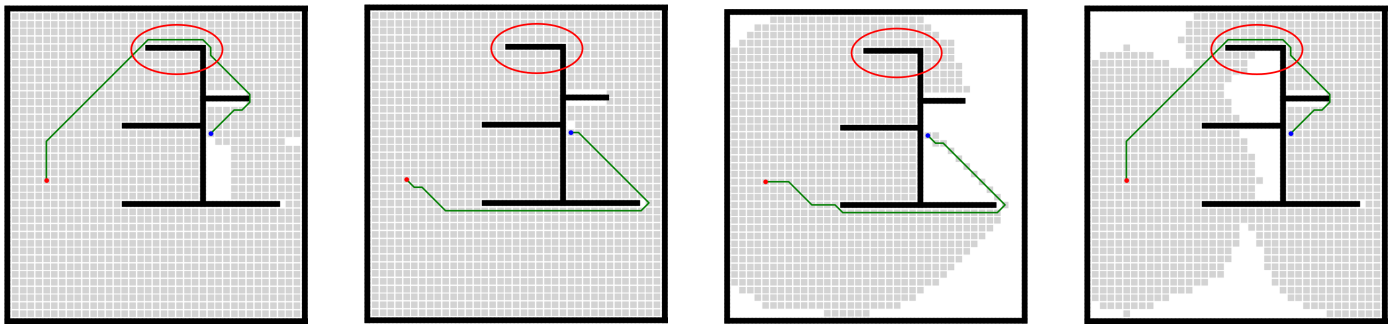
(**a**) BFS search (**b**) Dijkstra search (**c**) A* search (**d**) BHPS search

Figure 10. Search results comparison of the algorithms.

The test scenarios shown in Figures 11 and 12 are designed to simulate complex road network scenarios to verify the path search capabilities of several path search methods in complex road network environments. The difference is that there are slight differences between the two network simulation spaces. In this test scenario, both the BFS search and the Dijkstra search conducted extensive node searches for the whole search space. The A* search and BHPS search have a noticeable improvement in search space, which, to a certain extent, improves the computational efficiency of the path search algorithm. In addition, in complex road network scenarios, the advantage of the A* algorithm in search efficiency is no longer apparent. It does not produce search efficiency far beyond that of BHPS algorithm. The A* algorithm also needs to search more network nodes to obtain possible paths in complex environments. When the road network information becomes clear and straightforward, the efficient searchability of A* algorithm can be well demonstrated.

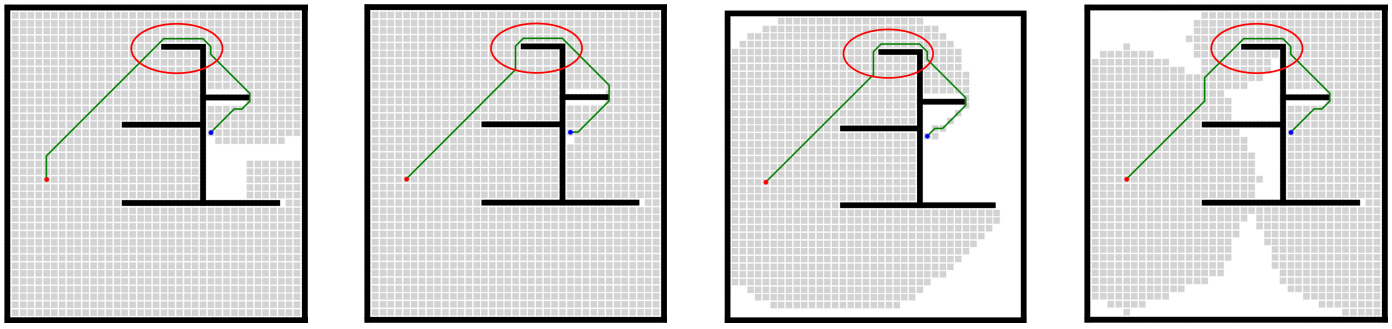
Meanwhile, due to the influence of greed strategy, minor changes in road network information will lead to entirely different search results from the A* algorithm and the Dijkstra search. The greed strategy results in a significant difference between the search result of the trajectory and the optimal global trajectory in the scenario shown in Figure 11.

In the global path planning problem, this unstable and non-optimal search result will seriously deviate from the global reference trajectory, resulting in high driving costs during vehicle following. It can be predicted that the greedy strategy of the A* algorithm in directed networks may produce unreliable search results. A greedy-based trajectory optimization strategy can achieve better trajectory search efficiency. However, such fully heuristic or depth-first search methods are unsuitable for solving the trajectory search problem under the road network information. Dijkstra search is not entirely based on greed. This unreliable situation may be alleviated due to its more expansive search space and local optimization. Therefore, the design of this paper based on BFS search and Dijkstra search is relatively reasonable. The simulation results also show that the proposed algorithm can maintain relatively reliable and stable path searching ability and improve the calculation efficiency of the trajectory searching process to a certain extent. When the road network information changes locally, the proposed algorithm maintains relatively stable and reliable search results.



(**a**) BFS search (**b**) Dijkstra search (**c**) A* search (**d**) BHPS search

Figure 11. Search results comparison of the algorithms.



(**a**) BFS search (**b**) Dijkstra search (**c**) A* search (**d**) BHPS search

Figure 12. Search results comparison of the algorithms.

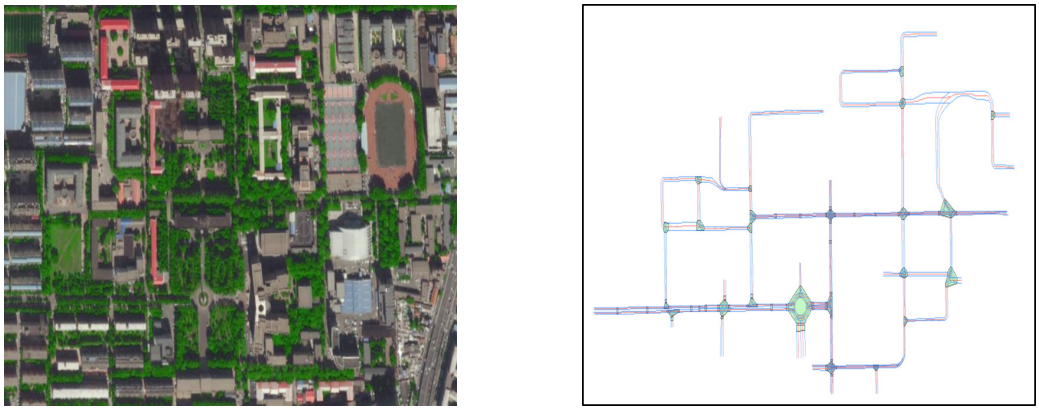
The above verification test is based on the search problem of the undirected structure graph. The driving cost of the path is simulated and calculated based on the traditional Euclidean distance. In the process of path search based on road network map or HD map, it is necessary to replace driving cost with driving cost model based on Lane-level maps described in Section 2. Next, we will conduct a complete test of the proposed algorithm under the real HD map scene.

5.2. Simulation Test Based on HD Maps

The paper tests and verifies the proposed algorithm based on the HD map of the Nanling Campus of Jilin University. The HD map used in this study is matched with the actual environment of the Nanling Campus of Jilin University (Figure 13a is the geographic environment map of Nanling Campus of Jilin University, and Figure 13b is the HD map

matched with it), which is provided by a third-party technology company cooperating with our research group. After gathering abundant static environmental information through sensors (such as RTK sensors, IMUs, lidar, and cameras) in the real world, an HD map with centimeter-level accuracy is produced after fusing the information offline.

In Figure 13b, the blue line represents the lane line, and the red line represents the centerline of the lane. Green areas indicate intersections or roundabouts, while yellow areas represent areas of transportation facilities (such as speed reduction belts and sidewalks) that affect vehicle traffic. The search space of the lane-level map produced according to the HD map is shown by the gray line in Figure 14. The search space of the lane-level map is contained in a digraph $G = (V, E)$ and the corresponding weighted function $\omega(u, v)$. The path segment is also visually expressed by edges $(u, v) \in E$.



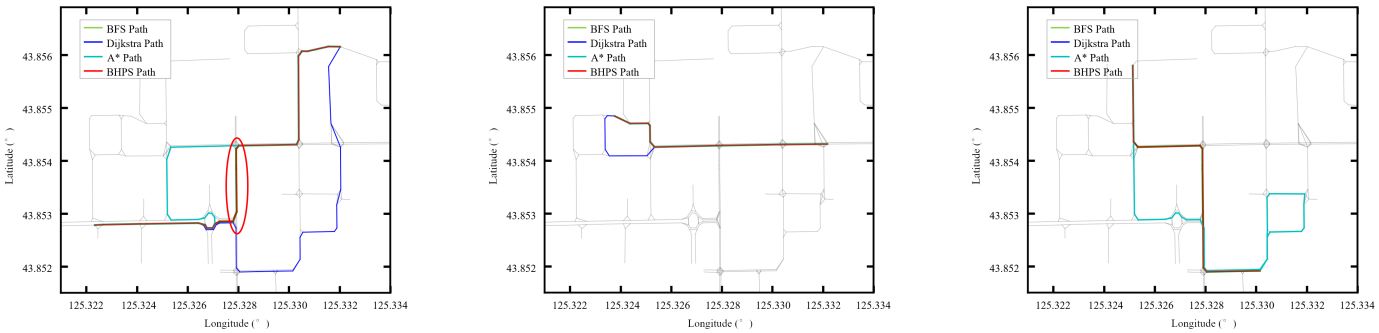
(**a**) Geographic environment (**b**) HD map

Figure 13. Test scenario.

To verify the high efficiency and near-optimal performance of the proposed BHPS algorithm, the bidirectional Dijkstra algorithm, bidirectional BFS algorithm, A* algorithm, and BHPS algorithm are used to carry out point-to-point path searches in the same scene. We find that the BHPS algorithm can generate the desired path in any specified search source node and goal node. The four search algorithms present partial differences, as shown in Figure 14. In some search scenarios, the BHPS algorithm and bidirectional BFS algorithm can find the desired path, while the bidirectional Dijkstra algorithm and the A* algorithm produce unsatisfactory search results. Although the bidirectional Dijkstra algorithm and the A* algorithm have higher search efficiency and can quickly find a passable path from the beginning to the end, the generated path does not match the desired path. During the search process, the Dijkstra algorithm carries out the next search step based on the optimal local solution in each search process. When encountering a shorter path segment with a lower travel cost (as shown in the red selection area in Figure 14a), the Dijkstra algorithm continues to search based on this shorter path segment. That is, even in relaxation operations, the Dijkstra algorithm still tends to select the local low-cost path segment as its next search solution during the search process. However, the local long-distance path segment is the required path in the optimal global solution, so the bidirectional Dijkstra algorithm cannot obtain better results.

Although the A* algorithm can quickly obtain the desired reference trajectory in some scenes, it does not obtain good search results in other scenes. In some scenarios, although the A* algorithm has the highest search efficiency, it also produces the worst search results (as shown in Figure 14a,b). Affected by the heuristic strategy, A* algorithm searches for the locally optimal future node in some road network nodes. However, these nodes cause the subsequent trajectory to deviate seriously from the desired global reference trajectory. The poor optimization ability based on the depth-first search is fully amplified in

a complex directed HD map. Compared with efficient searchability, poor trajectory search results will be more difficult to accept.



(**a**) Simulation result 1 (**b**) Simulation result 2 (**c**) Simulation result 3

Figure 14. Search results comparison of the algorithms.

The BFS algorithm is a path search method based on the complete search space and is not affected by the local cost of the path segment. As long as there is a path solution between s and g, the optimal global solution can be obtained by the BFS algorithm. The BHPS algorithm carries out path search via BFS search near s and g and uses the Dijkstra algorithm to supplement the search space in the middle segment of the search space, which greatly reduces the negative impact of local short-distance paths on the global search. Therefore, the BHPS algorithm can usually find the desired global path with less computational complexity than the bidirectional BFS search algorithm.

Tables 1–3 compare the relevant numerical results of the four algorithms in three simulation scenarios. In terms of the search range, although the A* algorithm and the Dijkstra algorithm have a small search range, corresponding to a low search time, the search effect in these scenarios is not ideal. The A* algorithm has an excellent performance in some scenes. It always maintains a more efficient trajectory search process, but it will also produce very unsatisfactory trajectory planning results in some scenes. The BHPS algorithm could complete the search task in these scenarios with a high search efficiency, while maintaining search results similar to those of the bidirectional BFS algorithm.

Table 1. Numeric comparison of simulation result 1.

	Dijkstra Algorithm	BFS Algorithm	A* Algorithm	BHPS Algorithm
Search scope dimension	102	171	38	143
Number of traffic facilities	12	16	19	16
Length of the path (m)	672.8	562.6	878.3	562.6
Travel time cost (s)	89.0285	79.5503	137.4714	79.5503
Search Time (s)	0.3022	0.5633	0.1303	0.4351

Table 2. Numeric comparison of simulation result 2.

	Dijkstra Algorithm	BFS Algorithm	A* Algorithm	BHPS Algorithm
Search scope dimension	95	171	22	129
Number of traffic facilities	9	9	9	9
Length of the path (m)	404.5	355.7	355.7	355.7
Travel time cost (s)	61.89	55.438	55.438	55.438
Search Time (s)	0.2539	0.4761	0.078	0.3624

Table 3. Numeric comparison of simulation result 3.

	Dijkstra Algorithm	BFS Algorithm	A* Algorithm	BHPS Algorithm
Search scope dimension	103	158	53	122
Number of traffic facilities	9	9	9	9
Length of the path (m)	814.8	814.8	1314.6	814.8
Travel time cost (s)	121.06	121.06	192.48	121.06
Search Time (s)	0.3283	0.4573	0.1677	0.3561

6. Conclusions

In this paper, a bidirectional hybrid global path search method based on HD maps is proposed, and this method realizes the lane-level path planning task. The proposed path search algorithm can improve the search efficiency of the point-to-point directed path search problem on the premise of satisfying the global optimally requirements and can effectively reduce the computational cost of path planning based on lane-level maps. The effectiveness of the algorithm is also proven in a simulation and real vehicle test. The main contributions of this paper focus on the following aspects:

1 This paper establishes a model of vehicle travel time costs based on lane-level maps and abundant road network information provided by HD maps;
2 The proposed algorithm uses incomplete bidirectional breadth-first search and bidirectional Dijkstra search to find the shortest path in the HD map. The algorithm not only has the high efficiency of the Dijkstra algorithm and the global optimization properties of BFS, but also greatly reduces the computational cost of the path search algorithm;
3 In this paper, the self-adaption relaxation space concept is proposed, which realizes the dynamic adjustment of the search space of the proposed algorithm. This algorithm can reduce the computational load according to the demand to satisfy the computational capacity of different computing requirements.

Author Contributions: Conceptualization, B.Y., X.S., and Z.G.; methodology, B.Y., X.S., and Z.G.; software, B.Y.; validation, B.Y., X.S., and Z.G.; formal analysis, B.Y.; investigation, B.Y.; resources, Z.G.; data curation, B.Y.; writing—original draft preparation, B.Y.; writing—review and editing, X.S. and Z.G.; visualization, B.Y.; supervision, Z.G.; project administration, Z.G.; funding acquisition, Z.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Key R&D Program of China (grant number 2017YFB0102601) and National Natural Science Foundation of China (grant numbers 51775236, 51675224, U1564214).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Skog, I. In-car positioning and navigation technologies: A survey. *IEEE Press.* **2009**, *10*, 4–21. [CrossRef]
2. Wei, J.; Snider, J.M.; Gu, T.; Dolan, J.M.; Litkouhi, B. A behavioral planning framework for autonomous driving. In Proceedings of the 2014 IEEE Intelligent Vehicle Symposium, Dearborn, MI, USA, 8–11 June 2014.
3. Chu, H.Q.; Guo, L.L.; Gao, B.Z. Predictive Cruise Control Using High-Definition Map and Real Vehicle Implementation. *IEEE Intell. Transp. Veh. Technol.* **2018**, *67*, 11377–11389. [CrossRef]
4. Liu, R.; Wang, J.L.; Zhang, B.Q. High definition map for automated driving: Overview and analysis. *J. Navig.* **2020**, *73*, 324–341. [CrossRef]
5. Joshi, A.; James, M.R. Generation of accurate lane-level maps from coarse prior maps and LiDAR. *IEEE Intell. Transp. Syst. Mag.* **2015**, *7*, 19–29. [CrossRef]
6. Liu, J.N.; Zhan, J.; Guo, C.; Ying, L.I.; Wu, H.; Huang, H. Data logic structure and key technologies on intelligent high-precision map. *Acta Geod. Cartogr. Sin.* **2019**, *48*, 939–953.
7. Guo, C.; Meguro, J.I.; Kojima, Y.; Naito, T. Automatic lane-level map generation for advanced driver assistance systems using low-cost sensors. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation, Hong Kong, China, 31 May–7 June 2014.
8. Jiang, K.; Yang, D.; Liu, C.; Zhang, T.; Xiao, Z. A flexible multilayer map model designed for lane-level route planning in autono-mous vehicles. *Engineering* **2019**, *5*, 266–295. [CrossRef]

9. Li, Z.W. *Research on Lane Keeping Control in Park Supported by High Definition Map*; Jilin University: Chngchun, China, 2019.
10. González, D.; Pérez, J.; Milanés, V.; Nashashibi, F. A review of motion planning techniques for automated vehicles. *IEEE Trans. Intell. Transp. Syst.* **2016**, *17*, 1135–1145. [CrossRef]
11. Guo, S.L.; Duan, J.; Zhu, Y.; Li, X.C.; Chen, T.W. Improved Dijkstra Algorithm Based on Fibonacci Heap for Solving the Shortest Path Problem with Specified Nodes. In Proceedings of the International Conference on Computer Science and Artificial Intelligence, Jakarta, Indonesia, 5–7 December 2017; pp. 52–61.
12. Xu, Z.; Liu, X.; Chen, Q. Application of Improved Astar Algorithm in Global Path Planning of Unmanned Vehicles. In Proceedings of the 2019 Chinese Automation Congress, Hangzhou, China, 25 November 2019.
13. Yang, B.; Ding, Z.; Yuan, L.; Yan, J.; Guo, L.; Cai, Z. A Novel Urban Emergency Path Planning Method Based on Vector Grid Map. *IEEE Access* **2020**, *8*, 154338–154353. [CrossRef]
14. Hu, X.; Jiang, Z.; Xu, C. Vehicle Path Planning Fusion Algorithm Based on Road Network. In Proceedings of the 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference, Chongqing, China, 12–14 June 2020; pp. 98–102.
15. Liu, H.; Jin, C.; Yang, B.; Zhou, A. Finding Top-k Shortest Paths with Diversity. *IEEE Trans. Knowl. Data Eng.* **2018**, *30*, 488–502. [CrossRef]
16. Liu, H.; Jin, C.; Zhou, A. Popular route planning with travel cost estimation from trajectories. *Front. Comput. Sci.* **2020**, *14*, 191–207. [CrossRef]
17. Yuan, L.; Yang, B.; Chi, Y.; Liu, Z.; Guo, L. Vehicle Emergency Route Planning Based on Grid Map. In Proceedings of the International Conference on Spatial Data and Intelligence, Virtual Event, 8–9 May 2020; pp. 122–135.
18. Broumi, S.; Bakal, A.; Talea, M.; Smarandache, F.; Vladareanu , L. Applying Dijkstra Algorithm for Solving Neutrosophic Shortest Path Problem. In Proceedings of the 2016 International Conference on Advanced Mechatronic Systems, Melbourne, Australia, 30 November–3 December 2016; pp. 412–416.
19. OpenDRIVE: Concept Document. *Association for Standardisation of Automation and Measuring Systems*; 14.01.2020; ASAM: Hoehenkirchen, Germany, 2020.
20. Liu, C. *Mission Planning and Trajectory Generation for Intelligent Vehicles Based on Lane-Level Map*; Tsinghua University: Beijing, China, 2017.
21. Jonker, R.; Volgenant, A. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing* **1987**, *38*, 325–340. [CrossRef]
22. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. *Introduction to Algorithms*, 3rd ed.; The MIT Press: London, UK, 2009.