algorithms

MDPI

Article

Application of Angle Related Cost Function Optimization for Dynamic Path Planning Algorithm

Mingbin Zeng [1], Xu Yang [2,*], Mengxing Wang [2] and Bangjiang Xu [2]

[1] School of Public Policy and Management, University of Chinese Academy of Sciences, Beijing 100049, China; dgzmb@163.com

[2] School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China; 2220160645@bit.edu.cn (M.W.); xbjtdcq@gmail.com (B.X.)

* Correspondence: yangxu@tsinghua.edu.cn; Tel.: +86-010-68913467

Received: 19 July 2018; Accepted: 9 August 2018; Published: 15 August 2018

Abstract: In recent years, Intelligent Transportation Systems (ITS) have developed a lot. More and more sensors and communication technologies (e.g., cloud computing) are being integrated into cars, which opens up a new design space for vehicular-based applications. In this paper, we present the Spatial Optimized Dynamic Path Planning algorithm. Our contributions are, firstly, to enhance the effective of loading mechanism for road maps by dividing the connected sub-net, and building a spatial index; and secondly, to enhance the effect of the dynamic path planning by optimizing the search direction. We use the real road network and real-time traffic flow data of Karamay city to simulate the effect of our algorithm. Experiments show that our Spatial Optimized Dynamic Path Planning algorithm can significantly reduce the time complexity, and is better suited for use as a real-time navigation system. The algorithm can achieve superior real-time performance and obtain the optimal solution in dynamic path planning.

Keywords: Intelligent Transportation System; advanced driver assistance systems; dynamic path planning; optimization

1. Introduction

Traffic problems are common problems faced by cities all over the world. Common traffic problems include traffic congestion, exhaust pollution, traffic accidents, parking difficulties and so on [1]. At present, the main traffic problems in urban traffic are traffic congestion, the increase of energy consumption and the aggravation of environmental pollution. Traffic jams not only cause a huge waste of time and environmental damage, but also hinder the development of urbanization and become the blocking factor to economic development. In the face of these severe urban traffic problems, how to solve the traffic congestion problem better under the condition of limited traffic resources has become a hot topic.

Researchers are beginning to try to use modern science and technology to solve traffic problems, integrating sensing, information, communication, computer technology, and intelligent transportation systems [2]. Intelligent transportation system technology, represented by the car network, can help users plan the optimal route and reduce vehicle travel time by strengthening the traffic distribution of the urban road network and conducting traffic guidance in real-time. It is an effective method to optimize the capacity of the urban road network and to reduce traffic congestion and traffic accidents.

As the core technology of intelligent transportation system, the development of dynamic path planning technology is becoming more and more mature and perfect [3]. In this paper, we present the Spatial Optimized Dynamic Path Planning (SODPP) algorithm which is based on the road-map loading mechanism and search direction optimization.

Our contributions are as follows:

(1) We enhance the effect of the loading mechanism for road maps by dividing the connected sub-net and building an spatial index;
(2) We enhance the effect of dynamic path planning by optimizing the search direction.

The following is organized as follows: In Section 2, we discuss related works, while in Section 3, we discuss the motive behind our algorithm; the whole algorithm is discussed in Section 4; Section 5 gives experiment results and discussions; the conclusions are given in Section 6.

2. Related Works

A number of works have been published on the topic of path searching in congested systems. Cascetta provided a comprehensive and systematic presentation of the mathematical models for the simulation of transportation systems and the methodologies for the analysis and design of these systems. Theoretical and operational aspects were presented in a rigorous and exhaustive framework, addressing a broad range of applications performed by researchers and practitioners [4]. Another study about transport modeling and its practical applications was presented by Ortuzar in 2011 [5].

Path search and route choice are the main components for any path searching algorithm. Ben Akiva et al. developed and tested innovative models of driver's route choice behavior [6]. The approach they exploited was to define choice sets of "labelled" paths to transform a large number of physical routes into a smaller number of routes, each representing a specific "label". A labelled path is defined as the optimal physical path with respect to some criterion function. The criteria that might be relevant to route choice include travel time, distance, scenery, congestion, signposting, etc. The label functions are estimated by maximizing the proportion of observed routes included in the sets of labelled paths.

De Maio et al. explored route choice on road networks [7]. The route choice model is divided into three levels: the generation of alternatives, the perception of alternatives and choice set, and finally, the choice of alternatives belonging to the choice set. A deterministic, selective, multi-criteria approach is used to generate the routes. A covering measure is calculated by comparing observed and generated paths to take into account which routes are currently chosen.

Vitetta presented the Quantum Utility Model (QUM) [8], which is derived from quantum mechanics models. In QUM, it is possible to simulate the sequence of decisions in cases of unique or not-unique pre-trip decisions at intermediate levels.

In addition, some researchers are starting to take into account the multi-dimensional attributes of the road network. Mcginty et al. first put forward the concept of "path quality" and thought that path quality should be considered in the optimal path selection [9]. Nie et al. presented the concept of "reliable path" which holds that the reliable path should be one that can guarantee the arrival on time [10]. Choi et al. introduced the concept of "driver orientation" and chose the appropriate path as the optimal path [11].

The existing research is focused on transforming different road attributes into unified dimensionless attributes, transforming the problem into an optimal path solution problem, and then using various optimal path algorithms to solve the problem. In fact, the optimal path search algorithm can be summarized as finding the path that meets the minimum target value of an attribute in a weighted road network. So, in this work, we aim to find a more efficient way to find the optimal path, and to enhance the real-time performance of the path searching process.

3. Problem Analysis

The design motive behind our SODPP algorithm is presented in this section. We chose three commonly used and representative algorithms, namely the Dijkstra algorithm, A* algorithm and ant colony algorithm, to illustrate the motive.

The Dijkstra algorithm was proposed by Professor E. W. Dijkstra to solve the shortest path problem [12]. It can be used to solve the shortest path from source point to other nodes in directed graph. It is also the theoretical basis of the routing algorithm and optimal path algorithm in many systems at present [13]. The main idea of the Dijkstra algorithm is to carry out a breadth-first search from the source point and expand to the outer layer from the starting point to the end point. The Dijkstra algorithm is guaranteed to find the optimal path in the given network, but for larger scale road networks, the processing efficiency is lower [14], and thus, it is not suitable for real-time and high accuracy dynamic path planning.

There are two main issues for the Dijkstra algorithm, as follows:

(1) Because of the strategy of global search, it is necessary to search all the nodes in the network. The search space is shown in Figure 1—it takes a lot of computation time and the time efficiency of the algorithm is low;
(2) Because of the need to store intermediate results, when the network is large, it uses a lot of storage space.

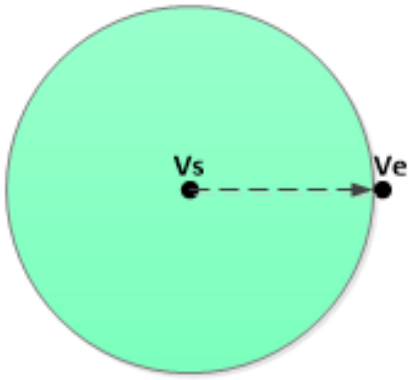

Figure 1. Search space of the Dijkstra algorithm.

The ant colony algorithm is also a heuristic algorithm that takes advantage of nature law. The ant colony algorithm is a kind of bionic algorithm which is proposed to simulate the behavior of an ant colony searching for optimal path after studying the real behavior of ant colony foraging [15]. The ant colony algorithm is often used to solve complex combinatorial optimization problems, such as the traveling salesman problem and the optimal path problem. In solving the optimal path problem, the ant colony algorithm (ACA) leads the behavior of searching for the optimal path by pheromones when simulating the foraging process of an ant colony in the biological world. The ant colony algorithm is an iterative algorithm based on empirical control. Its search space decreases with the number of iterations (increasing or decreasing), as shown in Figure 2. The search process of the ant colony algorithm does not depend on any mathematical theory, and it has a strong ability to find the global optimal solution. In the ant colony algorithm, ants tend to choose the path with more pheromones, and the selection of a path by ants will enhance the pheromone concentration of these paths so that the positive feedback of information can be realized. Although ants will choose the path with high pheromone concentration, the probability-based state transfer strategy still gives ants a certain probability of choosing a path a with low pheromone concentration so that the algorithm can try more paths to find the shortest path. However, the ant colony algorithm is still an iterative algorithm based on experience, which is prone to local convergence and slow convergence speed [16]. At the same time, it takes a long time to construct the solution which results in a long search time. It is not suitable for systems with high real-time requirements.

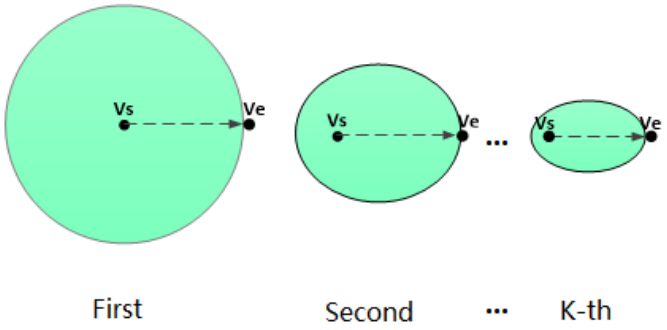

Figure 2. Search space of the ant colony algorithm.

The A* algorithm is a simple and practical heuristic search method which was proposed by Hart, Nilssno, and Raphael in 1968. The so-called heuristic search is a knowledge-based search strategy which refers to the evaluation of all the nodes that can be searched in a search tree by evaluating the best nodes and then searching for the target nodes from the best nodes. The heuristic search can effectively reduce the search area, avoid a large number of unnecessary search paths, and improve the efficiency. Compared with blind search methods, like the Dijkstra algorithm, the heuristic search, in principle, only needs part of the search space to get the solution of the problem, and the efficiency of the search can be improved effectively [17,18]. In heuristic searches, it is very important to evaluate the location. The evaluation method can control the size of search space and search time which directly affects the efficiency and accuracy of the algorithm. Different evaluation methods will achieve different results. For the A* algorithm, according to the control of its evaluation function, the search space is as shown in the blue part of Figure 3, and the search space is much smaller than the space of Dijkstra algorithm, as shown in Figure 1.

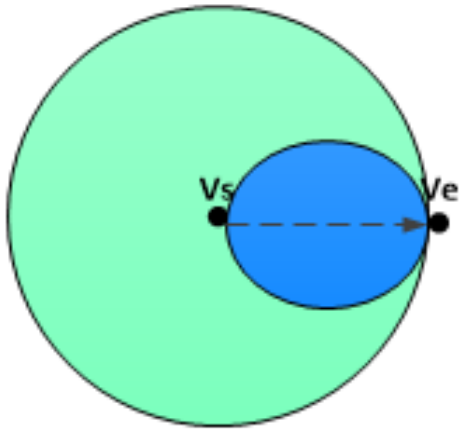

Figure 3. Search space of the A* algorithm.

According to our analysis, we summarize the three algorithms in Table 1. It can be concluded from Table 1 that the A* algorithm can obtain the optimal solution, and the search space for A* is the smallest. Also, A* has the lowest time complexity, with better robustness and convergence, and can better meet the needs of real-time systems [19].

Table 1. Comparison of the three algorithms.

	Time Efficiency	Complexity	Convergence	Robustness	Optimal Solution
Dijkstra	Wose	Easy	No	Good	Yes
A*	Good	Easy	Yes	Good	Yes
ACA	Worst	Complex	Yes	Good	Not

So, in this paper, we present the Spatial Optimized Dynamic Path Planning algorithm based on the optimization of the A* algorithm.

4. The Spatial Optimized Dynamic Path Planning Algorithm

The optimization of the Spatial Optimized Dynamic Path Planning algorithm was mainly concentrated on two aspects:

(1) The scale of the road network was controlled. The urban road network was decomposed into different sub-nets. According to the location of the starting point, different sub-nets were chosen as the search space to avoid searching the whole road network, thus reducing the scale of network information updated in real time;
(2) The search direction wwas restricted. The evaluation function was optimized, considering both the distance and search direction. More heuristic information was used to reduce the search area and search time to speed up the convergence of the algorithm.

4.1. Control the Scale of the Road Network

A common method for controlling the scale of the road network is to frame a search area in the road network by ellipse, circle, and polygon according to the coordinates of the starting node, which can be regarded as the search area of the algorithm. However, there is no guarantee that all nodes in the designated area are connected to each other for those methods [20], so a common shortcoming of these methods is that they may not find an optimal path in the selected search area. For example, as the A* algorithm is heuristic, it cannot consider some of the best paths that, to avoid congestion, move to an area not explored by A*.

In order to solve this problem, this paper proposes taking the connected sub-graph of the divided road network to be the search area when controlling the scale of the road network to ensure that there is at least one path in the search area. Therefore, the efficiency of the algorithm is guaranteed, and the robustness of the algorithm is ensured at the same time.

4.1.1. Road Sub-Net Partition Considering Connectivity

According to graph theory, the urban road network can be regarded as a directed graph (G), and the sub-net of the urban road network is a sub-graph of the directed graph (G). Because the urban road network system itself is a connected system, there must be a path between every two nodes in the road network. According to the definition of the strongly connected graph, we know that urban road network is a strongly connected graph. For a sub-net of a network, if there is a path between any two nodes in the sub-net, the sub-net is a connected sub-net. The problem of dividing a complete network into a connected sub-net can be transformed into solving the strongly connected components in a directed graph. The digraph is the complete network, and the strong connected component is the connected sub-net of the network.

The Kosaraju algorithm or Tarjan algorithm is generally used to solve the strongly connected components of digraphs. The principle of the Kosaraju algorithm is simple. In order to obtain all strong connected components of graph G, we first need a Depth First Search (DFS) of graph G. Then, the inverse graph (G^T) of graph G is calculated, and then another DFS is carried out for the inverse graph (G^T); thus, the connected component of graph G is obtained. Although the Kosaraju algorithm has a simple process, it has a high time complexity due to the need for two DFS searches.

The Tarjan algorithm is a DFS algorithm implemented by recursion. The theoretical basis of the Tarjan algorithm is to perform a DFS search for a directed graph (G) from any node. Every strongly connected component in a directed graph is a sub-tree in a search tree. The core idea of the algorithm is to add the unprocessed node in the current search tree to the stack when DFS and to determine whether the node is a strongly connected component when backtracking. Compared with the Kosaraju algorithm, the DFS is performed only once in the Tarjan algorithm, and the inverse graph is not needed, so the time complexity of Tarjan algorithm is lower.

Therefore, in this paper, the Tarjan algorithm was selected to solve the connected sub-net of road network, and the appropriate sub-net was selected to be used as the search area.

4.1.2. Sub-Net Query Optimization Based on the Spatial Index

In order to optimize and control the size of the network, an effective method is also needed to quickly match the appropriate sub-nets from all sub-nets once the location of the initial point of navigation has been determined. If the matching time of sub-net is too long, it will directly affect the solution time of the algorithm. Since all sub-nets in this study were stored in spatial databases, the sub-net matching problem was how to query spatial objects efficiently in spatial databases.

In spatial databases, spatial query efficiency is an important indicator to measure their performance [21]. Because of the huge amount of data processed by spatial databases and the complexity and diversity of spatial objects processed, in order to deal with these complex data more efficiently, queries are often optimized by using spatial indexes.

In this paper, we used the R-Tree index as the spatial index. The R-Tree index is an object-based indexing method. The R-Tree index divides the geographic space according to the Minimum Bounding Rectangle of the spatial object. The spatial index information of the object is organized into a tree structure, and the index information of the spatial object is managed by the index tree.

4.2. Restrict Search Direction

4.2.1. Analysis of the A* Algorithm Evaluation Method

The evaluation function of the A* algorithm ($f(i)$) is defined as

$$f(i) = g(i) + h(i), \tag{1}$$

where $g(i)$ is the actual cost of searching from the source to the current node (i), while $h(i)$ is the estimated cost from the current node (i) to the destination. As a heuristic search based on empirical information judgment, the more heuristic information included in the valuation function ($h(i)$), the smaller the search space is, and the more efficient the search algorithm is. On the one hand, the more information contained in the evaluation function, the more nodes that can be excluded and the smaller the search scale of the algorithm is. On the other hand, an increase in heuristic information will increase the amount of computation required and reduce the time efficiency of the algorithm. Thus, the design of the evaluation function is of critical importance [22].

In the A* algorithm, the evaluation function is designed to estimate the distance between the current node and the target node, as shown in Figure 4.

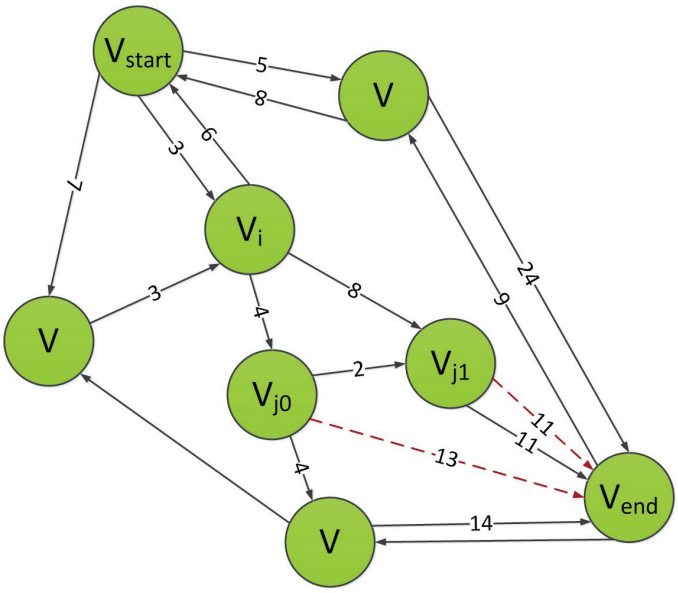

Figure 4. Illustration of the A* evaluation function.

If the current node is V_i, when picking next node, the evaluation function needs to be performed for nodes V_{j0} and V_{j1}. If it is assumed that the Euclidean distance is used as a measurement for distance, then the evaluation functions for those two nodes are calculated with

$$f(V_{j0}) = g(V_{j0}) + h(V_{j0}) = (3+4) + (2+11) = 20 \tag{2}$$

$$f(V_{j1}) = g(V_{j1}) + h(V_{j1}) = (3+8) + 11 = 22. \tag{3}$$

In this case , the algorithm would pick V_{j0} as next node, thus V_{j1} and all its successors would be excluded from searching list, thus restricting the search range. However, this method only takes into account the distance relationships between nodes, and does not take into account the directional relationships between nodes. According to actual life experience, if the chosen path is closer to the direction of the ray formed by the starting point and the end point, the smaller the angle is, the shorter the distance of the final path is. We can prove this.

As shown in Figure 5, the current node is V_i. Suppose the angle between the rays from V_i to V_{j0} and the rays from V_i to V_{end} is β, and the angle between the rays from V_i to V_{j1} and the rays from V_i to V_{end} is α. The distance from V_i to V_{j0} is g_0, while the distance from V_i to V_{j1} is g_1. The estimated distance from V_i to V_{end} is p.

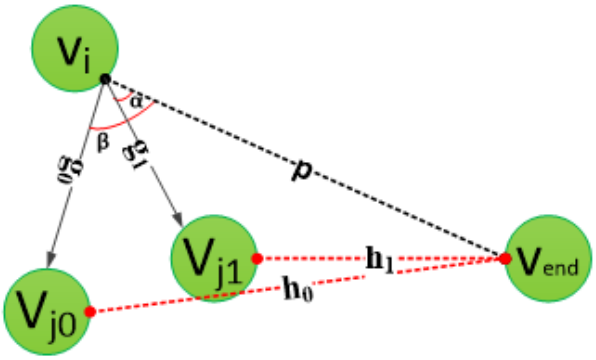

Figure 5. The influence of the search direction on the evaluation function.

According to the cosine theorem, for the following triangles (shown in Figure 6), there are

$$b^2 = a^2 + c^2 - 2ac \times \cos(B). \tag{4}$$

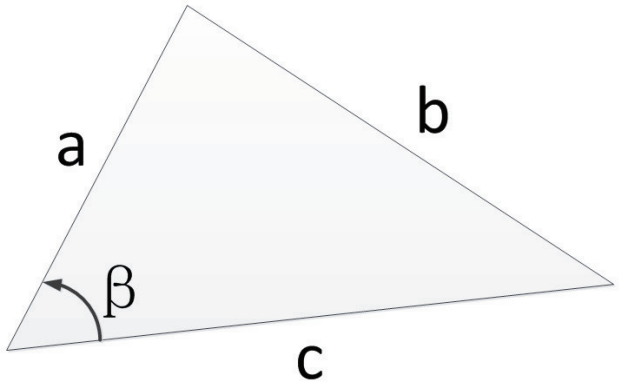

Figure 6. Cosine theorem.

Thus, we have

$$F(\beta) = h_0^2 = g_0^2 + p^2 - 2g_0 p \times \cos(\beta). \tag{5}$$

The derivation of function $F(\beta)$ is

$$F'(\beta) = 2g_0 p \times \sin(\beta). \tag{6}$$

Since the value of function $\sin(\beta)$ is $[0, 1]$ in the $[0, \pi]$ interval, the value of derivative $F'(\beta)$ in the $[0, \pi]$ interval is also greater than or equal to 0. That is,

$$F'(\beta) = 2g_0 p \times \sin(\beta) \geq 0. \tag{7}$$

From the nature of the derivative of the function, we can conclude that function $F'(\beta)$ in the interval [0, Π] monotonically increases, and we can prove that when there is a common edge (p) and $\alpha \leq \beta$, there must be a third side ($h_1 \leq h_0$) which is proved as follows:

$$g_1 + h_1 \leq g_0 + h_0. \tag{8}$$

It can be concluded that when the cost is equal for g_0 and g_1, the search can find a shorter path by choosing the direction closer to the current node and the end line. At the same time, selecting a small angle in the search can reduce the number of nodes that need to be searched and the search range, thus further reducing the solution time of the algorithm and improving the efficiency of the algorithm.

4.2.2. Search Direction Restricted Evaluation Method

According to previous analyses, in this paper, both the distance and search direction were used as heuristic information to design the valuation function as follows:

$$h(i) = h_r(i) \times h_d(i), \tag{9}$$

where $h_r(i)$ evaluates the influence of search direction, while $h_d(i)$ evaluates the influence of distance. Here, distance is measured as the Euclidean distance. If the coordinate of current node is (x, y), and the coordinate of the destination is (x_{end}, y_{end}), then $h_d(i)$ is calculated as

$$h_d(i) = \sqrt{(x - x_{end})^2 + (y - y_{end})^2}. \tag{10}$$

In order to design $h_r(i)$, two conditions need to be satisfied:

(1) $h_r(i)$ needs to be a function increasing monotonously with the angle. According to the previous discussion, if we choose a search direction with a smaller angle to the current node and the end point (set the angle to α), we can find the shortest path as soon as possible. In addition, the smaller the angle (α), the smaller the search area is. Therefore, it is necessary to design the $h_r(i)$ function as a function that increases monotonously with α. Thus, the less α is, less the $h_r(i)$ is, and the less h is. The algorithm can select a smaller search area with a greater opportunity to limit the direction of search;

(2) The value of the evaluation function for the current node must be less than or equal to the actual distance from the current node to the destination which can be expressed as $h_r(i) \times h_d(i) \leq h_d(i)$. That is, the value of $h_r(i)$ must be less than or equal to 1, and since the estimated distance cannot be negative, the range of value of $h_r(i)$ must fall within the interval $[0, 1]$.

If we set the value of the angle in $[0, \pi]$ interval, we can choose the monotone increasing trigonometric function in that interval:

$$f(\alpha) = -\cos(\alpha). \tag{11}$$

This function satisfies the first condition of monotone increment on $[0, \pi]$, and its value range is $[-1, 1]$, which is shown in Figure 7. However, this function does not satisfy the second condition.

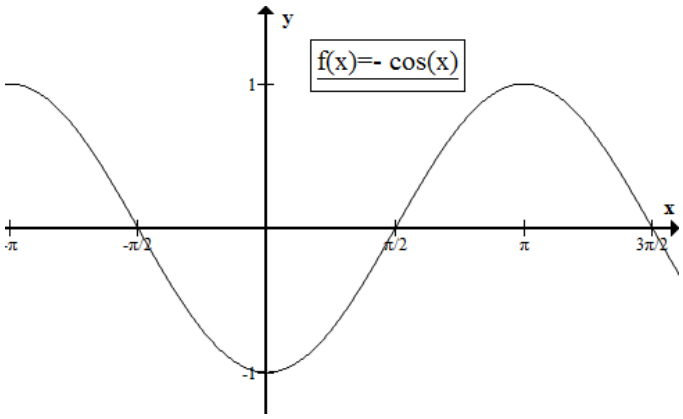

Figure 7. $-\cos(\alpha)$.

So, we use min-max normalization method to map $f(\alpha)$ onto the interval of $[0, 1]$:

$$\begin{aligned} g(\alpha) = & \frac{f(\alpha) - min(f(\alpha))}{max(f(\alpha)) - min(f(\alpha))} \\ & \times [max(g(\alpha) - min(g(\alpha)) + min(g(\alpha)] \\ = & \frac{-\cos(\alpha) - \cos(0)}{\cos(\pi) - \cos(0)} \times 1 + 0 \\ = & \frac{1 - \cos(\alpha)}{2} \end{aligned}. \tag{12}$$

This function is shown in Figure 8. It can be seen from Figure 8 that the $g(\alpha)$ function does not only satisfy the monotone increments in the $[0, \pi]$ interval, but also satisfies the requirement that the value in this interval be located in $[0, 1]$ interval. Therefore, this function can be used as $h_r(i)$.

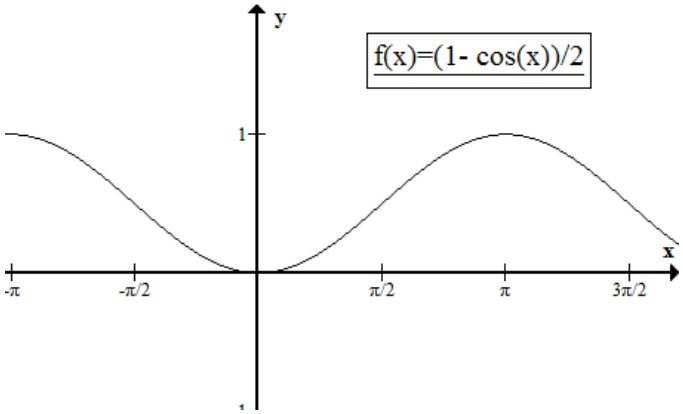

Figure 8. Image for $h_r(i)$.

Thus, the evaluation function for our algorithm is

$$h(i) = \frac{1-\cos(\alpha)}{2}\sqrt{(x-x_{end})^2+(y-y_{end})^2}. \tag{13}$$

5. Experiments

5.1. Experiment Framework

In this paper, Eclipse was used to realize the coding and development of the system, SUMO was used to realize the path drawing and analysis of navigation, and MySQL (including spatial extension Spatial) was used for database management. SUMO is a road traffic simulation software for continuous traffic conditions. The software was originally developed by the German Aerospace Center in 2000. Because of its perfect function and open source features, it has been widely used in the field of intelligent transportation research [23]. SUMO is well suited for traffic simulation and supports the analysis and evaluation of traffic related algorithms. The graphical interface of SUMO is shown in Figure 9.

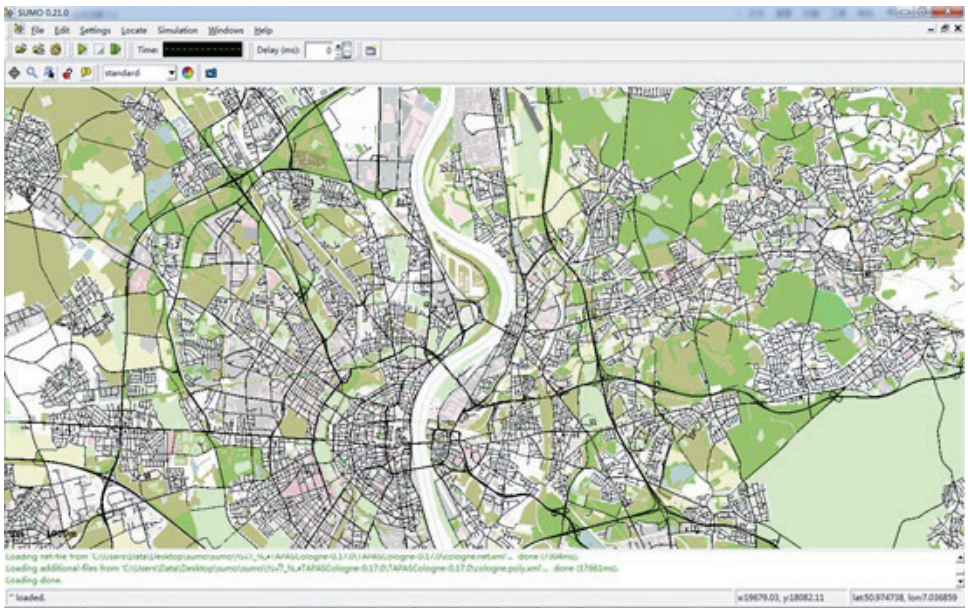

Figure 9. Interface for SUMO.

We chose the Dijkstra algorithm, A* algorithm and ant colony algorithm to compare our Spatial Optimized Dynamic Path Planning (SODPP) algorithm with. In addition, we chose the urban road network of Karamay. The network contains 846 road nodes, 1928 roads, and 5375 sets of road connectivity data. The urban road network of Karamay is shown in Figure 10.

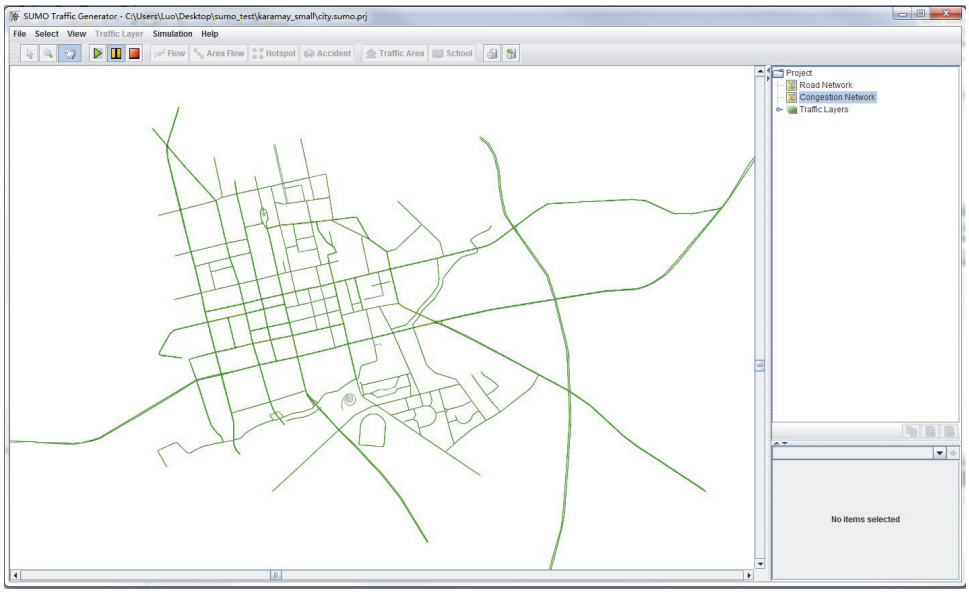

Figure 10. Karamay city.

5.2. Results and Discussion

In this paper, multi-time experiments were carried out, and the number of nodes, running times, and path information of different algorithms were recorded and analyzed in detail.

In order to select the representative experimental data for analysis, this paper selected the results of three experiments with the theoretical shortest path lengths, which had different numbers of search nodes and increments. The detailed data recorded in these three experiments were as follows.

In the first experiment, the node ID 34,091 was the starting point, the node ID 34,201 was the end point, the four algorithms sought solutions, and the shortest path length was 2290.95 m, The data records of the algorithm solution are shown in Table 2.

Table 2. Comparison of results for Experiment 1.

Evaluation Standard	Dijkstra	A*	ACA	SODPP
Start Point	34,091	34,091	34,091	34,091
End Point	34,201	34,201	34,201	34,201
Total Points	826	826	826	826
Load Points	826	826	826	42
Load Time	851 ms	836 ms	801 ms	151 ms
Construct Time	1162 ms	1167 ms	1087 ms	126 ms
Search Points	76	37	82	23
Total Time	17 ms	9 ms	63 ms	5 ms
Path Length	2290.95 m	2290.95 m	2295.08 m	2290.95 m

In the second experiment, node ID was the 33,898 is starting point, node ID 34,100 was the end point, the four algorithms sought solutions, and the shortest path length is 2832.47 m, The data records of the algorithm solution are shown in Table 3.

Table 3. Comparison of results for Experiment 2.

Evaluation Standard	Dijkstra	A*	ACA	SODPP
Start Point	33,898	33,898	33,898	33,898
End Point	34,100	34,100	34,100	34,100
Total Points	826	826	826	826
Load Points	826	826	826	156
Load Time	890 ms	860 ms	801 ms	210 ms
Construct Time	1208 ms	1162 ms	1093 ms	172 ms
Search Points	204	147	224	92
Total Time	176 ms	152 ms	2438 ms	85 ms
Path Length	2832.47 m	2832.47 m	3086.84 m	2832.47 m

In the third experiment, the node ID was 33,766 for the starting point, the node ID was 33,966 for the end point, the four algorithms sought solutions, and the shortest path length was 21415.83 m, The data records of the algorithm solution are shown in Table 4.

Table 4. Comparison of results for Experiment 3.

Evaluation Standard	Dijkstra	A*	ACA	SODPP
Start Point	33,766	33,766	33,766	33,766
End Point	33,966	33,966	33,966	33,966
Total Points	826	826	826	826
Load Points	826	826	826	257
Load Time	860 ms	875 ms	801 ms	410 ms
Construct Time	1186 ms	1167 ms	1109 ms	264 ms
Search Points	348	212	394	134
Total Time	2564 ms	2110 ms	18,057 ms	1142 ms
Path Length	21,415.83 m	21,415.83 m	22,365.50 m	21,415.83 m

Compared with the experimental data, the ant colony algorithm had the lowest time efficiency, but our algorithm was the best. In the road network load scale, besides the SODPP algorithm, the other three algorithms need to load all of the nodes. Loading nodes is time consuming and the time taken to build a road network is relatively large. In terms of the number of algorithm search nodes, the number of nodes searched by the Dijkstra algorithm is larger, and the number of nodes searched by the A* algorithm and SODPP algorithm are less. In regard to the path quality, apart from ant colony algorithm, the other algorithms are able to find the shortest path.

Based on the analysis of the principle of the algorithm and the experimental data, we found that when the SODPP algorithm was used to load the nodes and build the network, the time was greatly reduced, indicating that the improvement of the sub-net is effective. In the analysis of the experimental data of the SODPP algorithm with three experimental datasets, we found that the size of the road network, the number of search nodes, and the time-consumption required to seek the shortest path were significantly reduced. In contrast, compared with the Dijkstra algorithm, A* algorithm and ant colony algorithm, the SODPP algorithm was shown to effectively improve the time required for map loading and choosing the optimal path to solve the real-time requirements of real-time navigation systems.

5.3. Extension of Our Method

Although distance was used as the weight of paths in this work, actually, any characteristics (i.e., slope, width, capacity, disturbances, ...) and traffic flow, even the user behaviors could be modeled into weight of different paths to adopt our SODPP algorithm to find out the optimum path under certain situations. In Figure 11, when no congestion happens, the path searching result for a request from the place is denoted as a dot to the place denoted with a star.

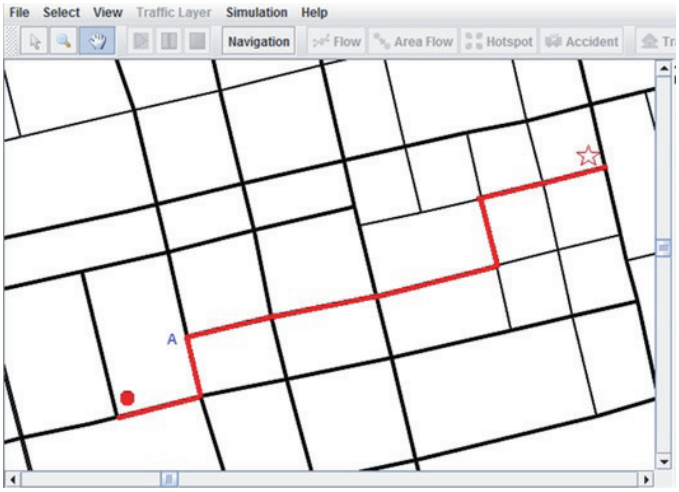

Figure 11. Path search result under normal situation.

If we include the traffic flow characteristics into the weight of any path, we can take the impact of congestion into account when looking for the best path. Figure 12 shows the path searching result for the same request when there is congestion detected in places denoted by *A* through analyzing real-time traffic data.

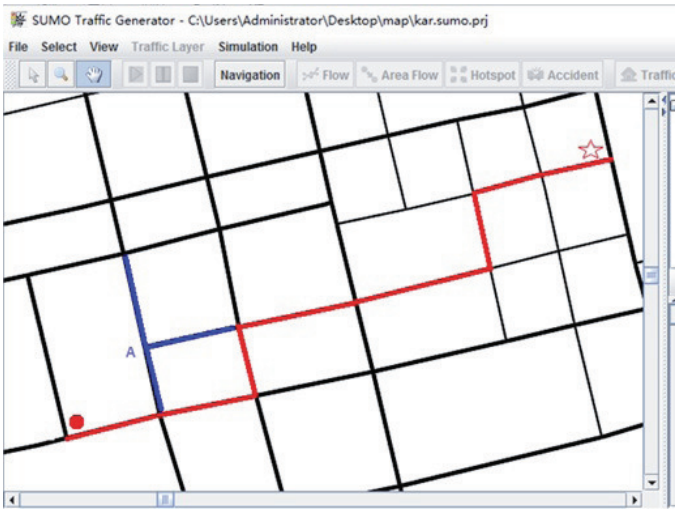

Figure 12. Path search result under congestion situation.

6. Conclusions and Future Work

The innovative feature of this paper was the proposal of the SODPP algorithm which can improve the time efficiency of current algorithms by using the connected sub-net instead of dividing the network area of the traditional navigation that cannot find the path. Experiments showed that our spatially optimized dynamic path planning algorithm can significantly reduce the time complexity and is better suited for real-time navigation systems. The algorithm can achieve superior real-time performance and obtain the optimal solution in dynamic path planning.

Although distance was used as the weights of paths in this work, actually, any characteristics (i.e., slope, width, capacity, disturbances, ...) and traffic flow, even the user behaviors, could be modeled as the weight of different pathes to adopt our SODPP algorithm to find the optimum path under certain situations. In fact, in future work, we will generate a path searching algorithm based on the SODPP algorithm with respect to the minimization of carbon dioxide emission.

Also, a traffic forecasting method was presented in our previous work [24]. We aim to integrate it into our SODPP algorithm as a future work to solve dynamic instability problems considering that the same information is shared by users and they travel in the same network area independently from the present or future congestion. By considering history data and real data at the same time, we could get more accurate predictions about the road situation to provide more optimized path searching results with consideration of personal experience.

Author Contributions: Conceptualization, X.Y.; Methodology, B.X.; Software, B.X.; Validation, B.X.; Formal Analysis, B.X.; Writing-Original Draft Preparation, M.W.; Writing-Review & Editing, M.Z.; Visualization, M.W.; Supervision, X.Y.; Funding Acquisition, X.Y.

Funding: This research was funded by the National Key R&D Program of China No. 2016YFC0904902, the National Natural Science Foundation of China under Grant No. 61502032, the Core Electronic Devices, High-End General Purpose Processor, and Fundamental System Software of China under Grant No. 2012ZX01034-001-002, Tsinghua National Laboratory for Information Science and Technology (TNList), and Samsung Tsinghua Joint Laboratory.

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

References

1. Tang, B.; Chen, Z.; Hefferman, G.; Pei, S.; Wei, T.; He, H.; Yang, Q. Incorporating intelligence in fog computing for big data analysis in smart cities. *IEEE Trans. Ind. Inf.* **2017**, *13*, 2140–2150. [CrossRef]
2. Ball, J.E.; Anderson, D.T.; Chan, C.S. A comprehensive survey of deep learning in remote sensing: theories, tools and challenges for the community. *J. Appl. Remote Sens.* **2017**, *11*, 042609. [CrossRef]
3. Hu, X.; Chen, L.; Tang, B.; Cao, D.; He, H. Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles. *Mech. Syst. Signal Process.* **2017**, *100*, 482–500. [CrossRef]
4. Cascetta, E. *Transportation Systems Analysis: Models and Applications*, 2nd ed.; Springer: New York, NY, USA, 2009.
5. De Dios OrtÃozar, J.; Willumsen, L.G. *Modelling Transport*, 4th ed.; John Wiley Sons: Hoboken, NJ, USA, 2011.
6. Ben-Akiva, M.; Bergman, M.J.; Daly, A.J.; Ramaswamy, R. Modeling inter-urban route choice behaviour. In Proceedings of the 9th International Symposium on Transportation and Traffic Theory, Delft, The Netherlands, 11–13 July 1984; pp. 299–330.
7. De Maio, M.L.; Vitetta, A. Route choice on road transport system: A fuzzy approach. *J. Intell. Fuzzy Syst.* **2015**, *28*, 2015–2027. [CrossRef]
8. Vitetta, A. A quantum utility model for route choice in transport systems. *Travel Behav. Soc.* **2016**, *3*, 29–37. [CrossRef]
9. Mcginty, L.; Smyth, B. Personalised route planning: A case-based approach. In Proceedings of the 5th European Workshop on Advances in Case-Based Reasoning, Trento, Italy, 6–9 September 2000; pp. 431–442.
10. Nie, Y.M.; Wu, X. Shortest path problem considering on-time arrival probability. *Transport. Res. Part B Method* **2009**, *43*, 597–613. [CrossRef]
11. Choi, W.K.; Kim, S.J.; Kang, T.G.; Jeon, H.T. Study on method of route choice problem based on user preference. In Proceedings of the International Conference of Knowledge-Based Intelligent Information and Engineering Systems, Vietri sul Mare, Italy, 12–14 September 2007; pp. 645–652.
12. Gass, S.I.; Fu, M.C. Dijkstra's Algorithm. In *Encyclopedia of Operations Research & Management Science*; Springer: New York, NY, USA, 2002; pp. 273–315.
13. Tan, G.Z.; He, H.; Aaron, S. Global optimal path planning for mobile robot based on improved Dijkstra algorithm and ant system algorithm. *J. Cent. South Univ.* **2006**, *13*, 80–86. [CrossRef]
14. Johnson, D.B. A note on dijkstra's shortest path algorithm. *J. ACM* **1973**, *20*, 385–388. [CrossRef]
15. Dorigo, M.; Gambardella, L.M.; Birattari, M.; Martinoli, A.; Poli, R.; Stützle, T. *Ant Colony Optimization and Swarm Intelligence*; Springer: New York, NY, USA, 2006.
16. Xie, J.; Cai, C. An ant colony algorithm on continuous searching space. *Int. Symp. Multispectral Image Process. Pattern Recognit.* **2015**, *9814*, 981402.

17. Siregar, B.; Gunawan, D.; Andayani, U.; Lubis, E.S.; Fahmi, F. Food delivery system with the utilization of vehicle using Geographical Information System (GIS) and A Star algorithm. *J. Phys. Conf. Ser.* **2017**, *801*, 012038. [CrossRef]
18. Zhang, Z.; Zhao, Z. A multiple mobile robots path planning algorithm Based on A-star and Dijkstra algorithm. *Int. J. Smart Home* **2014**, *8*, 75–86. [CrossRef]
19. Cheng, L.P.; Liu, C.X.; Yan, B. Improved hierarchical A-star algorithm for optimal parking path planning of the large parking lot. In Proceedings of the IEEE International Conference on Information and Automation, Hailar, China, 28–30 July 2014; pp. 695–698.
20. Bast, H.; Delling, D.; Goldberg, A.; Müller-Hannemann, M.; Pajor, T.; Peter, S.; Wagner, D.; Werneck, R.F. *Algorithm Engineering;* Springer: New York, NY, USA, 2016; pp. 541–550.
21. Samet, H.; Sankaranarayanan, J.; Alborzi, H. Scalable network distance browsing in spatial databases. In Proceedings of the ACM SIGMOD International Conference on Management of Data, Vancouver, Canada, 9–12 June 2008; pp. 43–54.
22. Xu, Z.P.; Lin, K. An algorithm based on improved A* restrictions on the path to search regional planning approach. *Comput. Knowl. Tech.* **2008**, *21*, 055.
23. Mainali, M.K.; Mabu, S.; Yu, S.; Eto, S.; Hirasawa, K. Dynamic optimal route search algorithm for car navigation systems with preferences by dynamic programming. *Trans. Electr. Electron. Eng.* **2011**, *6*, 14–22. [CrossRef]
24. Tang, H.; Liang, Y.; Huang, Z.; Wang, T.; He, L.; Du, Y.; Ding, G. Key technology of real-time road navigation method based on intelligent data research. *Comput. Intell. Neurosci.* **2016**, *2016*, 4. [CrossRef] [PubMed]