Application of Java - Based Optimization Dijkstra Algorithm in Parking Lot Berth Guidance

Menglong Wu
College of Electronic Information Engineering,North China University of Technology, Beijing No.5 Jinyuanzhuang Road, Shijingshan District, Beijing, 100144, China
15101153318@163.com

Bo Wu
College of Electronic Information Engineering,North China University of Technology, Beijing No.5 Jinyuanzhuang Road, Shijingshan District, Beijing, 100144, China
15101153318@163.com

Hao Song
College of Electronic Information Engineering,North China University of Technology, Beijing No.5 Jinyuanzhuang Road, Shijingshan District, Beijing, 100144, China
15101153318@163.com

Abstract—**Dijkstra algorithm is a single-source shortest path algorithm, and the result is the shortest distance from the source node to the destination node.In the parking guidance process,the shortest parking route is actually not the best parking guide route, need to consider the element of the safety of people and vehicles, shorten the time to find berths.In this paper, we use the Java programming language to implement Dijkstra algorithm with constraints, the optimum parking guidance route is obtained by comparing the berth to the sum of the exit and entrance distances. This approach allows users to more quickly and safely complete the task of parking, car park operation and management more efficient.**

Keywords-Optimization of Dijkstra Algorithm; Berth guidance; Optimal path; Java.

I. Introduction

In recent years, with the continuous increase in the possession of the entire social vehicles, driving often encounter difficult parking, parking hard to find, the parking process complicated situation and so on, these problems have long attracted wide attention, so the intelligent management of parking has become the inevitable trend of social development. Existing parking lot parking guidance system, part of the guidance system is the user based on the parking lot of the entrance signs to find their own parking spaces; The other part only stay in only consider the single element which is Short driving distance, that the shortest path is the optimal path. This guidance is single, and the degree of intelligence is not high. In view of the above problems by these parking guidance system, This article from the user point of view and the operation of the parking lot to consider the perspective of the impact of the search for parking factors,so as to establishing an optimal path of the parking guidance system, can shorten the user docked vehicle time, increase the user's safety factor, reduce the congestion inside the large parking lot, and the parking lot to run more efficient.Finally, the Dijkstra algorithm is optimized by using Java programming language.

978-1-5386-0497-7/17/$31.00 ©2017 IEEE

II. Optimal Parking Guidance Path Model

A. Parking interior structure

Figure 1 is a parking lot map, the figure has an entrance S, an outlet E.The figure is divided into "L" type crossroads and "T" type crossroads.Set up the intersection (such as n_i in the figure), driving entrance, driving exit as the vertex.The distance between the two vertices is the length of the distance between the two vertices. The distance between the two vertices is called the edge.In summary, the distribution of the parking lot can be equivalent to a weighted directed graph G = (V, E, L).V represents a nonempty set of all vertices, E represents a nonempty set of all edges, and L represents a nonempty set of edge weights.

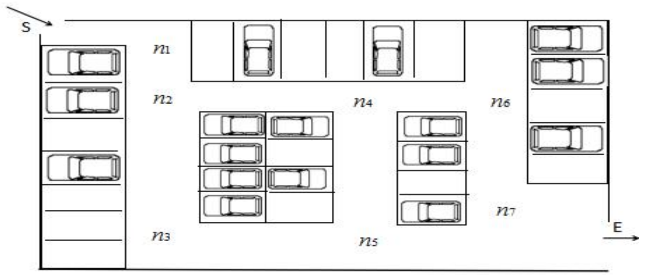


Figure 1. parking lot map

B. Establishment of optimal parking guidance path model

Different drivers have different requirements for the best way to find a parking lot.The choice of the optimal parking route will generally take into account the time spent in finding a parking space, the distance from the parking lot, the distance from the parking space to the exit, and other factors.This article considers that the way in which a user is looking for a parking space in a parking lot needs to be convenient, fast, and the shorter the walking distance required to leave the parking lot and enter the parking lot when looking for a vehicle, this will not only reduce the user stay in the parking lot of time, improve the safety factor, and can improve the efficiency of the parking lot.Therefore, this paper establishes the parking guidance model:Set the exit as a center, circle a circle at a certain distance, lock a circular area, and mark empty spaces in this area.The shortest distance between the empty spaces in the circular area and the shortest distance from the parking spaces to the outlets is calculated by comparison and the shortest distance is obtained, which is the optimal parking guide path.

III. DIJKSTRA ALGORITHM

To get the best parking guidance path, first make sure no vehicles parked in the parking area of the circle,then calculate the shortest distance between these spaces and the exit and the entrance, respectively, can be seen as a source node to a vertex between the shortest distance, that is, the shortest path.There are many ways to solve the single source shortest path,such as:Ant Colony Algorithm,Heuristic search algorithm,Genetic algorithm and so on.The ant colony algorithm belongs to the bionic algorithm, which has the characteristics of fast convergence and adaptive ability of the bionic algorithm, but it can not guarantee the exact convergence to the global optimal solution;Heuristic search algorithm is simple, fast, but can not guarantee the optimal solution.Dijkstra algorithm is relatively simple, easy to implement, but when the node is more, the calculation is large.After a comprehensive comparison, this paper uses Dijkstra algorithm to solve the shortest path problem.

A. The basic idea of Dijkstra algorithm

The Dijkstra algorithm is the shortest path algorithm from one vertex to the rest of the vertices, and solves the shortest path problem in the directed graph. Its main characteristic is that the starting point is extended to the outer layer at the starting point until it extends to the end.Assume that G = (V, E) is a weighted directed graph, and the vertex set V in the graph is divided into two groups.The first group has the set of vertices for which the shortest path has been obtained (denoted by S, and there is only one source point at the beginning of S, and then a shortest path is obtained for each of the vertices, and its vertices are added to S until all vertices are added to S , The algorithm is over);The second set is the set of vertices (denoted by U) for the rest of the shortest path, and the vertices of the second group in the order of the shortest path length are added to S.In the process of joining, the shortest path length of the vertices from the source point v to the S is less than the shortest path length from any of the vertices from the source point v to U,and the distance from the vertex in U is from the source point v to the vertex Including the current shortest path length of the vertex in S as the middle vertex.

B. Implementing Dijkstra algorithm with Java

The set of all vertices is denoted as V, and the set of vertices in the minimum spanning tree is U.At initialization, V contains all vertices, and U contains only one vertex V_0 ,and then continue to select a vertex from V, the vertex to the V_0 vertex path is the shortest and the vertex added to U, and delete the corresponding vertex in V;Each time you add a new vertex V_i to V, you need to modify the corresponding shortest path length of the vertex V_0 to the remaining vertices in V weight [j],J represents the subscript of the remaining vertex, in this case, the new shortest path weight corresponding to each vertex in weight [j] is modified to the shortest path length of the previous vertex weight [k] (k represents the subscript of the vertex before the vertex) and the path value of the remaining vertex in V The weight of the sum of GM.EdgeWeighr [k][j] is smaller than the original shortest path weight weight [k], that is, when weight[k]+GM.EdgeWeight[k][j]<weight[j], the current shortest path is weight[k]+GM.EdgeWeight[k][j]. Repeat the above steps until all vertices are added to the collection U, constantly updating the weight and path arrays,so that each vertex reaches the shortest path to the vertex of V_0 .Output the shortest path from V_0 vertex to each vertex and the distance of the shortest path.

IV. IMPLEMENTATION OF OPTIMAL PATH ALGORITHM

According to the establishment of the optimal parking guidance model, it is necessary to use the exit E vertex as the center. Began to search for empty spaces within a radius of 20 meters, if there is no qualified empty space in this range, the radius will be increased by 10 meters until a suitable empty space is found. Then use the algorithm described in the previous section to calculate the shortest distance from the exit distance and the shortest distance from the entrance, and the distances are L_O , L_i .Ultimately comparing each parking space of the sum of L_O and L_i , the optimal path is $\min(L_O + L_i)$.

Specific steps are as follows:

Assume that V is the shortest distance set to the parking lot entry S as the source node, the adjacency matrix D is distance weight value between P_i and P_j when calculating V, each element D_V (i,j) in D represents the distance length between the each nodes.M is the shortest distance set for all travel distances with exit E as the source node,D is the adjacency matrix that is assumed to be the vertex when calculating E ,each element in D_E represents the distance weight between the nodes.

Step1:The exit E vertex as the center and began to search for empty spaces within a radius of 20 meters, if there is no qualified empty space in this range, the radius will be increased by 10 meters until a suitable empty space is found. Marking the empty set of spaces as P and each parking space is marked as P_n , among n $\in$ [0,m-1], M is the total number of empty spaces searched.

Step2:Initializing the V set which is the set of the shortest path from distance entry S and place the source node a into the collection V, as V={V_1}.

Step3:Initializing D_i (i) which is the minimum path weight, as L(i)=D(0， i)， i$\in$[0,m-1].

Step4:Selecting the empty space node P_i so that P_i =min{ D_i (i)| P_j $\in$ P-V}, then P_i is the end of the shortest path from entry S that is currently obtained,when the vertex P_i is added to the set V ,the set V=V$\bigcup$ { P_i }.

Step5:Finding the minimum weight of all unused vertices (which isn't contained vertices in collection V). If a vertex K is found, the vertex K is selected in the set V. Then taking the vertex K as the middle point, re-weighting the value, as L(K)=min{L(K)， L(j)+ D_i (K)} and modifying the shortest distance from entry S to K.

Step6:Repeating Step4 and Step5 until all empty parking nodes in set P are all added to the set V.

Step7:And then using the above six steps to calculate the shortest distance from the set P in all empty parking nodes to the exit E.

Step8:Calculating the final weight of all empty parking nodes in the set P to the entrances and exits, then min{ D_i (i)+ D_o (i)} corresponding parking space P_i is the best parking spaces, the path corresponding to D_i (i) is the best path from entry S to P_i .

V. CASE RESULTS ANALYSIS

Through the implementation of the above algorithm steps, in the Eclipse environment using the Java programming language to write this algorithm, the specific preparation steps see Section II. Need to find out meet the constraints of the parking spaces through the algorithm, the length of the parking lot for the x-axis, wide for the y-axis, the entrance S for the origin of the establishment of the coordinate system [5], as shown in Figure 2. The parking lot is 6 meters long ,3 meters wide and the width of the road is 4 meters. Take each remaining parking space as a point to exit E as the center, 20 meters for the radius of the search, you can get the parking node to meet the optimal path algorithm conditions.As shown in Figure 2, two nodes P1, P2 are obtained.It is necessary to write these node data into an adjacency matrix, the matrix in the second section of the Java programming language to run the algorithm to get the optimal path.The established exit adjacency matrix is shown in Table 1, and the entry adjacency matrix is shown in Table 2(-1 means no way to pass or own to their own).

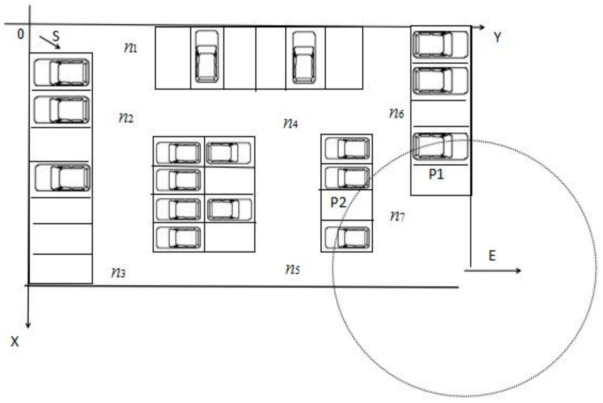


Figure 2. Parking lot map

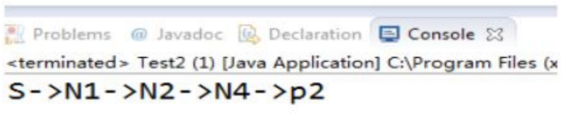
TABLE I. EXPORT ADJACENCY MATRIX

Distance	E	N7	P1	P2
E	-1	10[m]	-1	-1
N7	10[m]	-1	4[m]	3[m]
P1	-1	4[m]	-1	-1
P2	-1	3[m]	-1	-1

TABLE II. ENTRY ADJACENCY MATRIX

Distance	S	N1	N2	N3	N4	N5	N6	N7	P1	P2
S	-1	3[m]	-1	-1	-1	-1	-1	-1	-1	-1
N1	3[m]	-1	8[m]	-1	15[m]	-1	-1	-1	-1	-1
N2	-1	8[m]	-1	12[m]	12[m]	-1	-1	-1	-1	-1
N3	-1	-1	12[m]	-1	-1	12[m]	-1	-1	-1	-1
N4	-1	-1	12[m]	-1	-1	12[m]	6[m]	-1	-1	9[m]
N5	-1	-1	-1	12[m]	12[m]	-1	-1	8[m]	-1	-1
N6	-1	-1	-1	-1	6[m]	-1	-1	12[m]	7[m]	9[m]
N7	-1	-1	-1	-1	-1	8[m]	12[m]	-1	4[m]	3[m]
P1	-1	-1	-1	-1	-1	-1	7[m]	4[m]	-1	-1
P2	-1	-1	-1	-1	9[m]	-1	9[m]	3[m]	-1	-1

The experimental results show that the shortest path and its length from these nodes to the inlet S and exit E.The results of the shortest distance from the entrance are shown in Table 3.According to the data in Table 3, it can be determined that the current optimal parking space is p2.According to the results of the Java program running, as shown in Figure 3, the best parking guide route is S-> N1-> N2-> N4-> p2.



```
<terminated> Test2 (1) [Java Application] C:\Program Files (x
S->N1->N2->N4->p2
```

Figure 3. Run the results

TABLE III. DATA RESULTS

Distance	S	E
P1	36[m]	14[m]
P2	32[m]	13[m]

VI. CONCLUSION

The traditional method of providing the shortest parking path within a parking lot is usually based on the shortest path as the optimal path. In this paper, we adopt the constrained Dijkstra algorithm, that is, the path with the shortest path between the entrance and exit paths is the optimal path. Verification of this method can better provide users with parking guidance, to solve the user to the parking lot after the choice of parking spaces blindly, saving the parking time, can make the parking lot to run more efficient, more orderly and internal management, and it has a certain practical value for large parking lot. And this method only scans the parking area in the exit area of the parking node, which can greatly improve the operating efficiency of the program.

VII. REFERENCES

[1] [1]Said Broumi, Mohamed Talea,Assia Bakali.Application of Dijkstra algorithm for solving interval valued neutrosophic shortest path problem.Computational Intelligence (SSCI), 2016 IEEE Symposium Series on.

[2] [2] Xinyu Mao,Yuxin Cheng,Haige Xiang.Two Block Partitioned Dijkstra Algorithms.Vehicular Technology Conference (VTC Fall), 2013 IEEE 78th.

[3] [3] J. Anitha,M. Karpagam. Ant colony optimization using pheromone updating strategy to solve job shop scheduling.Intelligent Systems and Control (ISCO), 2013 7th International Conference on.

[4] [4] James B Orlin, Kamesh Madduri. A faster algorithm for the single source shortest path problem with few distinct positive lengths, in: Journal of Discrete Algorithms, 2010, (8)pp.189-198.

[5] [5] A.V. Goldberg, A simple shortest path algorithm with linear average time, in: Proc. 9th European Symposium on Algorithms (ESA 2001), in: Lecture Notes in Computer Science (LNCS), vol. 2161, Springer-Verlag, 2001, pp. 230-241.