An Optimum Vehicular Path Solution with Multi-heuristics

Feng Lu[1] and Yanning Guan[2]

[1] State Key Laboratory of Resources and Environmental Information System,
Institute of Geographical Sciences and Natural Resources Research,
Chinese Academy of Sciences,
Beijing 100101, P.R.China,
luf@lreis.ac.cn
[2] Laboratory of Remote Sensing Information Sciences,
Institute of Remote Sensing Applications,
Chinese Academy of Sciences,
Beijing 100101, P.R.China,
guan@irsa.irsa.ac.cn

Abstract. Heuristics have been widely used in artificial intelligence related fields including path finding. In this paper, the author argue that different heuristics can be integrated to solve the path finding problems and set forward a solution integrating greedy heuristic, directional heuristic and hierarchical heuristic. In greedy heuristic, an improved Dijkstra's algorithm based on quad heap is set forward. Then an MBR for ellipse is presented to limit the searching extent. Thirdly, hierarchical spatial reasoning is used to build another heuristic considering the hierarchical structure of road network, which makes the optimum path selection completed in higher hierarchies as much as possible. A case study is carried out with a real road network to verify the efficiency and validation of the solution integrating the above algorithms.

1 Introduction

The single source optimum path with nonnegative arc weights is one of the most natural network optimization problems. A variety of algorithms for single source optimum path has been designed and implemented[1][2]. Many of them use heuristic strategies. Heuristic indicates literally learning by experience, or more generally in the artificial intelligence literature, a heuristic is a 'rule of thumb' and as such is the approach used by almost any human in conducting a searching [3]. In the context of searching algorithms, heuristic implies simply search specific knowledge. The optimum path algorithms based on heuristics include costing algorithm[3], branch-and-bound algorithm[4], hill-climbing algorithms [4], greedy algorithms [5][6][7] and A* algorithm[8]. etc.

Among the known optimum path algorithms, many of them use a kind of heuristic named greedy searching as searching strategies and explore how to design delicate running data structures and searching algorithms, so as to improve the running efficiency of sequential optimum path algorithms under the uniform time complexity.

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3039, pp. 964–971, 2004.

Literature [1][2][7][9] have made detailed analysis and comparison for the optimum path algorithms. Because the real networks do not concern negative weights, Dijkstra's algorithm, a famous label setting algorithm adopting greedy heuristic, has got wide applications. Dijkstra's algorithm is the most mature optimum path algorithm theoretically up to date and the most robust one in practice[2][6][10]. Further work on fine-tuning Dijkstra's algorithm has been conducted by many researchers and has formed a large family of Dijkstra's algorithms.

There are many kinds of heuristics besides greedy strategy and they can be integrated to solve the optimum path problems. In this paper, the author showed the advantages of integrating the greedy heuristic, directional heuristic and hierarchical heuristic to get the optimum vehicular path in complicated road networks.

The remainder of this paper is organized as follows. Section 2 sets forward an improved Dijkstra's algorithm with quad heap priority queue. Section 3 introduces how to adopt spatial relationship between geographical objects to limit the path searching scale and speedup the searching procedure. Section 4 shows the hierarchical division of road networks and develops a hierarchy selection algorithm. Section 5 illustrates the integration of the above techniques with a case study. Finally Section 6 makes a discussion and draws some conclusions.

2 Quad Heaps and Fine-Tuned Dijkstra's Algorithm

Dijkstra's algorithm uses a heuristic based on greedy strategy. More details about Dijkstra's algorithm can be found in [11][12]. Theoretically, it is difficult to further improve the worst case time bound of Dijkstra's algorithm within the serial algorithm framework, and current research concentrates on how to improve the operational efficiency in practice.

Heap priority queues have been proved excellent data structures for Dijkstra's algorithms [2][6][7][13]. *K*-ary heaps, binomial heaps, Fabonacci heaps and radix heaps were discussed more. It has been argued that *k*-ary heaps are more suitable than binomial heaps and Fabonacci heaps to implement the priority queues for Dijkstra's algorithm in road networks [10]. A *k*-ary heap can be regarded as a complete *k*-ary tree. Every node on the corresponding complete *k*-ary tree corresponds to the element of an array that suffixed the node. The root correspond to the minimum (maximum) element of the *k*-ary heap, and the node elements of a sub tree rooted from one node are greater than (less than) the node element. The heights of *k*-ary heaps are $\lfloor \log n \rfloor$. The operation time bounds of *k*-ary heaps are proportional to the tree heights, i.e., $O(\log n)$.

Dijkstra's algorithm based on *k*-ary heaps concerns three kinds of heap operations, i.e., *heap-insert*, *heap-extract-min* and *heap-decrease-key*. Among these operations, '*heapify*' operation, which is used to keep the characteristics of *k*-ary heaps, plays a major role. Theoretically, the running time of a '*heapify*' operation depends on the *k* value. The optimal *k* value will minimize the running time of concerned heap operations.

The author had solved the optimal integer solution of k is 4. More detail can be found in [12]. The fine-tuned Dijkstra's algorithm implemented with quad heap priority queue has same time bound as that with binary heaps, namely $O((m+n)\log n)$. According to [10], the time bound is only $O((n+\log(1+m)-\log n)\log n$. For the sparse graph such as road networks that hold n<<m<<2n, the bound is $O(n\log n)$.

3 Extent Restriction for Path Searching

Most of the optimum path algorithms ignore the spatial distribution characteristics of networks, which make the searching free. Even if the procedure ends once reaching the destination nodes, there is still a lot of redundant searching. For road networks, Location information of nodes can be used to make it up.

3.1 Ellipse Restriction

In a network, approximate maximum distance or cost M_D can be determined once the source and the destination are defined. Then an extent can be constructed to limit the searching. The nodes outside the extent will not be considered during the searching for they are beyond the maximum distance or cost estimated.

If Euclid distance from a node N to the source S and to the destination T is $|SN|$ and $|NT|$ respectively, the limiting condition can be defined as $|SN|+|NT| \le M_D$. The critical points of N form an ellipse focused at S and T, with the major axis M_D, as shown in figure 1.

Ellipse limiting was firstly introduced in [14] to implement a primitive depth-first tree searching method. A suitable major axis M_D can be calculated statistically.

First, a lot of nodes can be extracted systematically from road networks to construct node sets A, B and their Cartesian product C. Every element in C can be regarded as the source and destination node between which the optimum path needs determined.

Supposing Euclid distance between them is e_{ab}, the optimum path distance is p_{ab}, a set R of ratio $r_{ab} = p_{ab}/e_{ab}$ can be formed for the extracted sample. Then a special value τ can be acquired which makes the elements in R have values not higher than τ under a special confidence. With τ as a product coefficient, the major axis M_D of the ellipse can be calculated using the coordinates of the source and the destination.

Ellipse limiting restricts the searching extent and reduces the scale of optimum path searching. But the procedure judging whether the current nodes are located in the ellipse is time consuming, which offsets the benefit brought through the reduced searching scale. When the destination is far away from the source, the ellipse limiting is sometimes worse than the free searching.

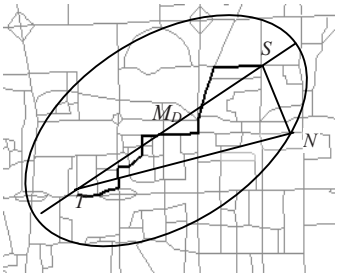

Fig. 1. Ellipse limit searching

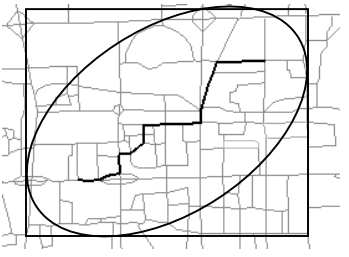

Fig. 2. MBR limit searching

3.2 MBR Restriction

To make up the deficiency of ellipse limiting, we set forward a rectangle limiting method. It avoids the great deal of product and evolution calculation. The major idea is to build the minimum bounding rectangle (MBR) for the ellipse and taking the MBR as a restricted area to reduce the searching scale.

In the ellipse method introduced above, when the product coefficient τ is acquired, the ellipse equation can be expressed as
where
After getting the ellipse equation, we can get the extremums of x and y through the differential equations

The tangents formed by the extremums of x and y compose the MBR for the ellipse, as shown in figure 2.

$$\frac{[\cos\theta(x-a)+\sin\theta(y-b)]^2}{A^2}+\frac{[-\sin\theta(x-a)+\cos\theta(y-b)]^2}{B^2}=1 \tag{1}$$

$$\theta=arctg(\frac{y_T-y_S}{x_T-x_S}) \qquad a=\frac{x_S+x_T}{2} \qquad b=\frac{y_S+y_T}{2}$$
$$A=\frac{\tau}{2}\sqrt{(y_T-y_S)^2+(x_T-x_S)^2} \qquad B=\sqrt{A^2-\frac{(y_T-y_S)^2+(x_T-x_S)^2}{4}} \tag{2}$$

((x_S,y_S), (x_T,y_T): coordinates of the source and destination nodes

$$x_m=a\pm\sqrt{A^2\cos^2\theta+B^2\sin^2\theta} \qquad y_m=b\pm\sqrt{A^2\sin^2\theta+B^2\cos^2\theta} \tag{3}$$

Taking the MBR as the restricted area, judging the nodes processed only need compare whose coordinates with the bounds of the MBR. No complex calculation is needed.

When the azimuth between the source and the destination $\theta = k\pi/2$, the MBR gets the minimum area, i.e., $4AB$. And the area ratio of the MBR to the ellipse is $4/\pi = 1.2732$. When $\theta = (2k+1)\pi/4$, the MBR gets the maximum area, i.e., $2(A^2+B^2)$. And the ratio of the area of MBR to that of the ellipse is

$$\frac{S_R}{S_E} = \frac{4\tau^2 - 2}{\pi\tau\sqrt{\tau^2 - 1}} \tag{4}$$

4 Hierarchical Path Selection

Human cognition to the world has a distinct property of spatial hierarchies, and every hierarchy contains some necessary information to solve specific problems. For example, facing a road traffic map, a person can quickly find an optimum path between two spots chosen arbitrarily on the map. Even if the path is not a shortest path, it is also a good alternate to be referenced. Here the human thinking is none other but a procedure of typical hierarchical spatial reasoning. It does not need to carry out complicated calculation on the completely unwrapped detail, but make judgement on a generalized hierarchy, and decide the optimum path in this way.

4.1 Hierarchical Structure of Road Network

Roads in the network are divided into different classes according to their grades. Hierarchies are composed of classes. Every hierarchy contains the roads within higher hierarchies. Network is divided into sub-regions with boundary formed by the road sections in lower hierarchy. A sub-region is a grid in sub-region with higher hierarchy. Adjacent sub-regions share the boundaries. Adjacent hierarchies share some common nodes, i.e. all the nodes in the higher hierarchy. These nodes are the connector between the hierarchies and lower hierarchies.

The rule of hierarchical division was set forward in [15] and get improved in this paper through keeping the shared nodes in adjacent hierarchies to facilitate backtracking from the lower to the higher hierarchies during the recursion procedure.

4.2 Hierarchy Based Optimum Vehicular Path Selection

Given the source and destination nodes S and T, supposed i and j are the highest hierarchies that contain S and T, S and T are marked as S_i and T_j, then the optimum path algorithm can run on the sub-network level decided by $\min(i,j)$. Although such a lossy algorithm can not guarantee the resulted path is all to nothing the shortest path, it is a preferable path. Furthermore, such a hierarchical spatial reasoning is coincident with the drivers' thinking manner.

Road networks can be divided into several hierarchies, e.g., trunk roads (including superhighways), ordinary road and alley. Adjacent lists are established for the nodes in the three hierarchies respectively. The path searching is performed as follows.

1) Searching the nearest node to S_i and T_j to find out the highest hierarchies that contain S_i and T_j, so as to determine the beginning hierarchy; We have $k = \min(i,j|i \quad 3 \ \& \ j \quad 3)$;

2) Confirming the hierarchical zones that S_k and T_k are located;

3) If Sk and Tk are located in the same zone of hierarchy two, then the optimum path is calculated directly on hierarchy one;

4) Otherwise, if S_k and T_k are located in the same zone of hierarchy three, we can find out the nearest nodes to S_k and T_k on hierarchy two, marked as S_2 and T_2. The optimum paths $S_k \rightarrow S_2$ and $T_2 \rightarrow T_k$ are calculated on hierarchy one. Then the optimum path $S_2 \rightarrow T_2$ is calculated on hierarchy two. $S_k \rightarrow S_2$, $S_2 \rightarrow T_2$, $T_2 \rightarrow T_k$ are linked to get the required paths;

5) Otherwise, if S_k and T_k are not located in the same zone of hierarchy three, we can first find out the nearest nodes S_2 and T_2 to S_k and T_k on hierarchy two according to the method in 4), then calculate the optimum paths $S_k \rightarrow S_2$ and $T_2 \rightarrow T_k$ on hierarchy one. Also we can find out the nearest nodes S_3 and T_3 to S_2 and T_2 on hierarchy three, calculate the optimum paths $S_2 \rightarrow S_3$ and $T_3 \rightarrow T_2$ on hierarchy two. Lastly, the optimum path $S_3 \rightarrow T_3$ is calculated directly on hierarchy three, and the all of the resulted paths, i.e., $S_k \rightarrow S_2$, $S_2 \rightarrow S_3$, $S_3 \rightarrow T_3$, $T_3 \rightarrow T_2$ and $T_2 \rightarrow T_k$ are linked to get the final result.

5 A Case Study

A real urban road network with 12,800 nodes and 17,800 lines, and a PC(with single Pentium 2.4G CPU and 256M memory) are adopted to verify the efficiency of the above algorithms and the integrated solution.

Firstly, a Dijkstra's algorithm is implemented with quad heap priority queue. It uses a greedy heuristic to find the shortest path. Secondly, a directional heuristic with MBR restriction is added onto the greedy heuristic. 400 nodes are extracted from the network systematically to form sets A, B and their Cartesian product set C. Calculating e_{ab}, p_{ab} and r_{ab} for each element in C and statistically analyzing set R formed with all r_{ab},, we get $\tau = 1.379$ under a 95% confidence. It can be resulted from section 3.2 and equation 4 that the MBR limit increases the searching extent to 27.32%~36.29%, compared with the ellipse limit. In spite of the seeming deficiency, the following analysis shows it is still worthy of such an expense.

Nodes are picked up to form five sets from the upper left (1241 nodes), lower left (1155 nodes), upper right (1366 nodes), lower right (1143 nodes) and center (1307 nodes) part of the network randomly to compare the average CPU time of calculating the shortest path between the nodes in the sets with free searching, ellipse limiting and rectangle limiting. The node sets are marked as R_{ul}, R_{ll}, R_{ur}, R_{lr} and R_{cc}, The result is shown in table 1.

It shows that MBR limit always performs better than free searching method, and at most circumstance better than ellipse limit. Moreover, MBR limit increases the confidence of resulted path for its bigger searching area.

Table 1. Average time consumption of the shortest path searching with extent restriction (millisecond)

Searching method	R_{ul}-R_{ur}	R_{ul}-R_{cc}	R_{ul}-R_{ll}	R_{ul}-R_{lr}	R_{cc}-R_{lr}	R_{cc}-R_{cc}	R_{ll}-R_{ur}
free searching	12.49	11.57	17.12	20.05	23.88	8.26	22.82
ellipse limit	6.32	9.10	17.10	20.98	9.41	4.36	25.48
MBR limit	5.86	8.95	14.65	18.51	9.64	4.13	21.87

Thirdly the hierarchical selection heuristic is added onto the greedy and directional heuristics. Figure 3a gets the shortest path only on a single hierarchy. It can be noticed half the path is made of ordinary roads. Figure 3b presents a three-hierarchy division of the network. Figure 3c shows the optimum path with hierarchical path selection. Although the resulted path may not be the shortest path, most of that is located on trunk roads. It decreases the factors difficult to expect or quantify and makes the path more robust.

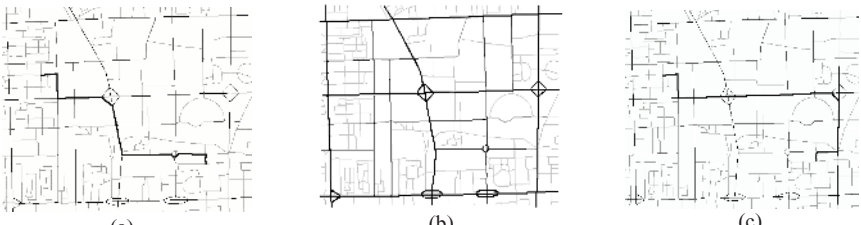

Fig. 3. Hierarchical optimum path selection

6 Conclusion

Different heuristics have different advantages and can be integrated to solve optimum path problems. Such an idea is experimented in this paper with integrating greedy heuristic, directional heuristic and hierarchical heuristic, with a real urban road network. A Dijkstra's algorithm implemented with quad heap priority queue establishes a perfect basis for the real road network optimum path searching. The spatial distribution characteristic of road networks can be used to form a directional heuristic to limit the searching scale. An MBR for ellipse limit is preferable to free searching and ellipse limit in path searching. The spatially hierarchical structure of road network can be used to build another heuristic to make the optimum path selection completed in higher hierarchies as much as possible, and improves the fault tolerance for the input data and the applicability of the resulted path. On the basis of the presented solution, other kinds of heuristics can also be integrated to contribute to the optimum path problems.

Acknowledgment. This research was supported by the National Natural Science Foundation of China under grant No. 40201043.

References

1. Deo N. and Pang C.Y., Shortest-path algorithms: taxonomy and annotation, Networks, 4(1984) 275-323.
2. Cherkassky B.V., Goldberg A.V. and Radzik T., Shortest paths algorithms: theory and experimental evaluation, Mathematical Programming, 73(1996) 129-174.
3. Fisher P. F., A primer of geographic search using artificial intelligence, Computers & Geosciences, 16(1990) 753-776.
4. Winston, P. H., Artificial Intelligence, 1st edn. Reading: Addison-Wesley, Massachusetts (1984).
5. Pallottino S., Shortest-path methods: complexity, interrelations and new propositions, Networks, 14(1984) 257-267.
6. Ahuja R.K., Mehlhorn K., Orlin J.B. and Tarjan R.E., Faster algorithms for the shortest path problem, Journal of the Association for Computing Machinery, 37(1990) 213-223.
7. Zhan F.B. and Noon C.E., Shortest path algorithms: an evaluation using real road networks, Transportation Science, 32(1998) 65-73.
8. Lester P, A* Pathfinding for Beginners, URL: http://www.policyalmanac.org/games/aStarTutorial.htm (2003).
9. Pallotlino S. and Scutella M. G., Shortest path algorithms in transportation models: classical and innovative aspects, Technical Report, Universita di Pisa (1997).
10. Goldberg A.V. and Tarjan R.E., Expected performance of Dijkstra's shortest path algorithm, Technical Report No. PRINCETONCS//TR-530-96, Princeton University (1996).
11. Tarjan R.E., Data Structures and Network Algorithms, 1st edn. Society for Industrial and Applied Mathematics Press, Philadelphia (1983).
12. Lu F., Zhou C.H. and Wan Q., An improved Dijkstra's shortest path algorithm based on quad heap priority queue and MBR searching method, Proceedings of the 9th Spatial Data Handling Symposium, (2000) 6b:3-13.
13. Fredman M.L. and Tarjan R.E., Fibonacci heaps and their uses in improved network optimization algorithms, Journal of the Association for Computing Machinery, 34(1987) 596-615.
14. Nordbeck, S. and Rystedt, B., Computer cartography --- range map, BIT, 9(1969), 157-166.
15. Car A. and Frank A., General principles of hierarchical spatial reasoning-the case of wayfinding, Proceedings of the 6th International Symposium on Spatial Data Handling, (1994) 646-664.