Journal of Southeast University (English Edition) Vol. 26, No. 4, pp. 558 – 562 Dec. 2010 ISSN 1003—7985

Optimal path finding algorithms based on SLSD road network model

Zhang Xiaoguo Wang Qing Gong Fuxiang

(School of Instrument Science and Engineering, Southeast University, Nanjing 210096, China)

Abstract: A solution to compute the optimal path based on a single-line-single-directional (SLSD) road network model is proposed. Unlike the traditional road network model, in the SLSD conceptual model, being single-directional and single-line style, a road is no longer a linkage of road nodes but abstracted as a network node. Similarly, a road node is abstracted as the linkage of two ordered single-directional roads. This model can describe turn restrictions, circular roads, and other real scenarios usually described using a super-graph. Then a computing framework for optimal path finding (OPF) is presented. It is proved that classical Dijkstra and A* algorithms can be directly used for OPF computing of any real-world road networks by transferring a super-graph to an SLSD network. Finally, using Singapore road network data, the proposed conceptual model and its corresponding optimal path finding algorithms are validated using a two-step optimal path finding algorithm with a pre-computing strategy based on the SLSD road network.

Key words: optimal path finding; road network model; conceptual model; digital map; vehicle navigation system; A* algorithm; Dijkstra algorithm

Optimal path finding (OPF) functionality is helpful for a driver to make a quick decision on how to get to the destination with the least cost, such as time, oil, distance, etc[1-3].

So far, road networks in digital maps are generally in single-line mode style[2, 4-5], which means that a road is abstracted as a single line, and road crossings are abstracted as nodes. Most existing road network conceptual models meet difficulties in supporting kinds of real-world road network cases, such as a circular road and turn restrictions[3]. A super-graph has the aforementioned ability; however, it is too complex to be used in an embedded system in its data model, data storage, data update, and spatial operation.

First, a single-line-single-directional (SLSD) road network model is presented. It is proved that any super-graph can be transferred to a normal SLSD road network. So the proposed SLSD road network is able to describe such real-world road scenarios: road directions, turn restrictions, circular roads, etc.

Secondly, it is proved that classical Dijkstra and A* algorithms can be used without any modifications in OPF computing by transferring the corresponding super-graph to an SLSD road network.

Finally, tests using a two-step optimal path finding algorithm are conducted. From test results, with the SLSD road network model, classical A* and Dijksta algorithms can be directly used for optimal path finding without any modification.

Received 2010-05-26.

Biography: Zhang Xiaoguo (1973—), male, doctor, lecturer, zxg519@sina.com.

Foundation item: The National Key Technology R& D Program of China during the 11th Five Year Plan Period (No. 2008BAJ11B01).

Citation: Zhang Xiaoguo, Wang Qing, Gong Fuxiang. Optimal path finding algorithms based on SLSD road network model [J]. Journal of Southeast University (English Edition), 2010, 26(4): 558 – 562.

1 Conceptual Model of SLSD Road Network

A road network model is usually based on road connection, and it can be described as

$$\begin{aligned}&\text{RoadWork} = (N, R)\\ &N = \{x \mid x \in \text{NodeSet}\}\\ &R = \{\text{NR}\}\\ &\text{NR} = \{\langle x, y\rangle \mid L(x, y)\wedge(x, y \in N)\}\end{aligned}$$

where N is the set of all the road nodes; $L(x, y)$ means a link between road nodes x and y, and its value is affected by geometrical length between two nodes (the road segment's length) and other parameters, such as road lanes, road classification, etc.

This model has some disadvantages in describing a super-graph:

1) Difficulty in describing circular roads (see Fig. 1(a)) and multiple roads between two nodes (see Fig. 1(b)).

2) Difficulty in describing turn restrictions (see Fig. 1 (c)).

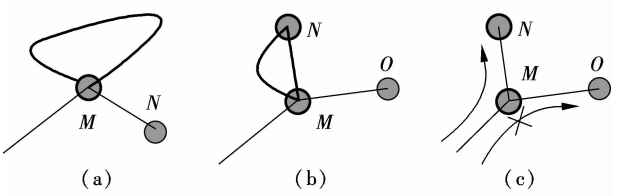

Fig. 1 Two simplified cases of super-graphs and turn restriction. (a) Circular road; (b) Multiple roads between two nodes; (c) Turn restriction

To overcome the above disadvantages, a node-connected SLSD road network model is presented. The principle of this model can be described as follows:

1) All roads are single-directional;

2) A road node is regarded as the linkage of directional roads; therefore, they are "roads" in the new model instead of "nodes" in the traditional model.

3) The term traverse is used to describe an ordered directional road pair, through which a vehicle can run from the first road to the next one.

The model can be defined as

$$\begin{aligned}&\text{RoadWork} = (N, R)\\ &N = \{x \mid x \in \text{DirectRoadSet}\}\\ &R = \{\text{NR}\}\\ &\text{NR} = \{\langle x, y\rangle \mid L(x, y)\wedge(x, y \in N)\}\end{aligned}$$

where N is a directional road. If a real-world road is bi-directional, it will be divided into two different single-directional roads. NR describes topological relationships set among directional roads.

In this model, if a driver can go through directional road x to y, there must be a link $L(x, y)$ between the two roads, and $L(x, y)$ is the road node connecting the two roads.

The core elements in the SLSD road network model are shown in Fig. 2. In the figure, a node represents an endpoint of a road, and it can be presented with 2-D or 3-D coordinates. The term traverse is defined to describe link $L(x, y)$.

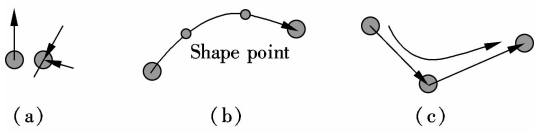

Fig. 2 Core elements in SLSD road network. (a) Node; (b) Road; (c) Traverse

Sometimes, a vehicle is allowed to make a turn inside a road. In Fig. 3(a), single-directional roads AB and $B'A'$ are reversed to each other in the same physical road. A driver in this road can make a turn at almost any point, so we define the two single-directional roads as being "all-connected". A virtual road node (see Fig. 3(b)) is abstracted to describe such connectivity information. Now, the special case in Fig. 3(a) is consistent with the aforementioned model.

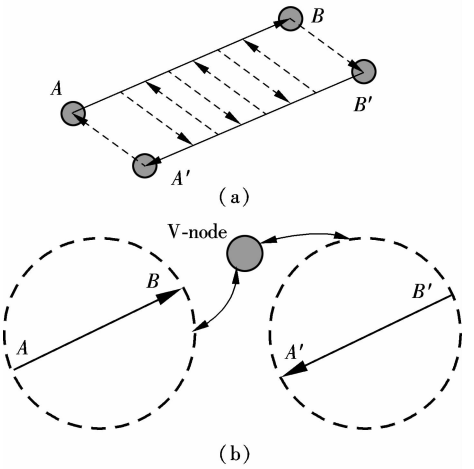

Fig. 3 All-connected road and virtual node

Fig. 4 shows a simplified road network. *CO-OC* is an all-connected traverse. It can be described by Tab. 1 and Tab. 2. In Tab. 1, road node topological information is listed. InLink is used to describe which directional roads can go to this node directly, and OutLink is used to present information regarding which directional roads can run from the node directly. In Tab. 2, topological information of directional roads is listed. InNode means the node through which a car can go to the current road. Similarly, OutNode is the node that the current directional road can run to. InLink is used to give information regarding which directional roads can run to the current road through InNode; OutLink is used to present information regarding which roads the current road can run to through OutNode. AllCon is used to store "all-connected" traverses.

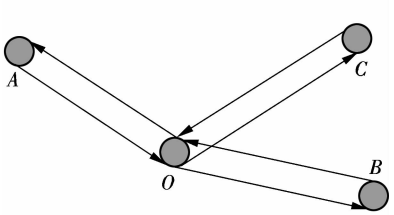

Fig. 4 A simplified road network based on SLSD model

Tab. 1 Road node topological information

ID	InLink	OutLink
1(A)	OA	
2(O)	AO, CO, BO	OA, OC, OB

Note: only for node A and O, others are not listed.

Tab. 2 Directional road topological information

ID	InNode	OutNode	InLink	OutLink	AllCon
1(AO)	A	O	OA	OB, OC OA	
2(OA)	O	A	BO, CO AO	AO	
3(OC)	O	C	AO, BO		CO

Note: only for road AO, OA and OC, others are not listed.

The model has the following advantages:

1) Being able to represent any real-world roads, single-directional or not;

2) Being able to express turn restrictions;

3) Being able to describe any styles of super-graph.

In modern metropolises, the widths of some roads are even more than 30 m, and there are 8 lanes or more. To resolve this problem, the multi-lane model is used. A physical road can be divided into more than 2 or even 5 to 6 SLSD roads. Fig. 5 shows a real-world physical road with 12 lanes in Hong Kong, and to describe it better, two non-overlapped directional lines are used (see Fig. 6).

Fig. 5 A real-world broad street in Hong Kong

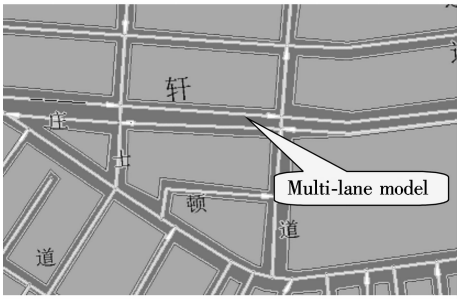

Fig. 6 Multi-lane model for broad streets or roads

2 Optimal Path Computing Framework Based on SLSD Road Network Model

2.1 Dijkstra algorithm

Most OPF algorithms, precise or quasi-precise, hierarchical[6-10] or non-hierarchical[11-14], pre-computed[7-8, 10] or real-time[11], can be roughly divided into two kinds: 1) Dijkstra or Dijkstra-like algorithms[8, 13]; 2) A* or A*-like algorithms, such as the width-first searching algorithm, the Dequeue algorithm, etc[8, 11].

Given a road-connected graph (see Fig. 7), if all the weights are non-negative, to find a shortest path, usually the Dijkstra algorithm can be used.

The Dijkstra algorithm, initially proposed by E. W. Dijkstra, depends on the Dijkstra matrix, which is used to store weight values between pairs of nodes. Tab. 3 shows the Dijkstra matrix of a graph as shown in Fig. 7, and the value in each cell is the weight between two corresponding nodes. After the Dijkstra matrix is obtained, the algorithm can be described using the process in Ref. [15].

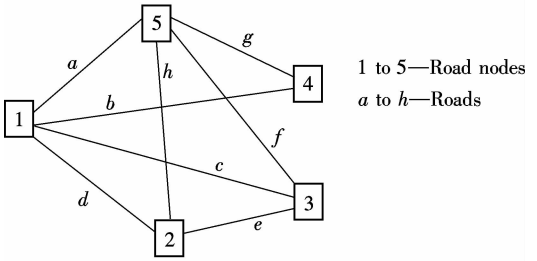

Fig. 7 A simple road-connected network

Tab. 3 Dijkstra matrix for traditional road network

Node	Node				
	1	2	3	4	5
1		*d*	*c*	*b*	*a*
2	*d*		*e*		*h*
3	*c*	*e*			*f*
4	*b*				*g*
5	*a*	*h*	*f*	*g*	

Limited by the disadvantages of the traditional road network model, sometimes the Dijkstra algorithm cannot be directly used. In the SLSD road network model, being single-directional, each road owns the road information it connects, so turn restriction information is stored in the Dijkstra matrix. Therefore, with the node-connected SLSD road network model, the Dijkstra algorithm can be directly used without any modification. In Fig. 8, a simple road network is described; the corresponding Dijkstra matrix is shown in Tab. 4. In Tab. 4, matrix element (m, n) means whether a vehicle

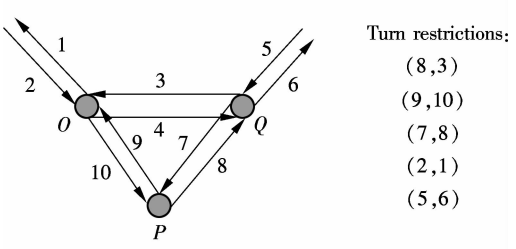

Fig. 8 A simple SLSD road network

Tab. 4 Dijkstra matrix for road network in Fig. 8

FromRoad	ToRoad									
	1	2	3	4	5	6	7	8	9	10
1										
2				*O*						*O*
3	*O*			*O*						*O*
4			*Q*			*Q*	*Q*			
5			*Q*				*Q*			
6										
7									*P*	
8						*Q*				
9	*O*			*O*						
10								*P*	*P*	

can go from directional road m to n or not. For example, the value of $(3, 1)$ is O, which means that a vehicle can go from road 3 to 1 through node O. So being different from the traditional Dijkstra matrix, the Dijkstra matrix of the SLSD road network has the following characteristics:

1) The value of the matrix element means the road node through which a vehicle can go from FromRoad to ToRoad. If the value is empty, it shows that FromRoad and ToRoad are not topologically connected.

2) The cost of a vehicle going through a road is not stored in the matrix. Instead, it can be obtained from the road property in the road database.

2.2 A* algorithm

A* algorithm is a heuristic width-first searching algorithm for OPF computing. Generally, there are two kinds of searching strategies: 1) Single-directional; 2) Bi-directional.

Furthermore, the single-directional A* searching algorithm can be divided into two kinds: 1) Forward searching; 2) Reverse searching. To implement the first one, for a given node, information regarding which succeeding nodes a car can arrive at should be provided. And in the second searching process, we should know through which nodes a car can arrive at the current node. In bi-directional searching, both of them are needed.

In the A* algorithm, usually searching cost $f(x)$ is used to guide heuristic searching.

$$f(x) = \alpha g(x) + \beta h(x) \tag{1}$$

where α, β are the weight coefficients; x is the current cost; $h(x)$ is the estimated future cost. The cost can be distance, time consumption, oil consumption, etc. Take distance as an example. Suppose that we have

$$\alpha = \beta = 1 \tag{2}$$

The real cost is always more than that of Eq. (1). According to Fig. 8, a data structure using C++ can be used to aid the searching process:

```
struct anode_t
{
id_t idNode; //current node
cost_t costCurrent; //current cost
cost_t costFull; //full cost
anode_t* nodeParent; //preceding node
};
```

We need to maintain the track of all visited nodes by putting them in a CLOSE list. Likewise, we will maintain the track of all the nodes we want to visit in an OPEN list. If only non-negative weight is legal for each linkage, a forward A* algorithm can be described as the following process:

1) Initiate the OPEN and CLOSE lists.

2) Put the original node in the OPEN list, and set it as the current extending node.

3) Extend the current node, sort all its sequential nodes according to its heuristic cost in Eq. (1).

4) For each sequential node,

a) If it has been in the CLOSE list, ignore it;

b) If it is already in the OPEN list with an equal or lower cost, ignore it; if with a higher cost, remove the old one, and reinsert the new node and make sure that the OPEN list is continue to be sorted.

5) Insert the extended node to the CLOSE list, and make sure the CLOSE list is sorted.

6) Go to step 4), until the destination is found, or there is no node in the OPEN list which means that it fails to find any path.

7) If the destination is found, trace the path from destination to origin using data structure anode_*t*.

For a reverse searching process, a similar algorithm can be used.

Clearly, if there are circular roads (see Fig. 9(a)) or multiple roads between two nodes (see Fig. 9(b)), both single-directional and bi-directional searching processes will have the same problem as that of the Dijkstra algorithm because the system does not know how to remember through which road to get to the next node (see Figs. 9(c) and (d)).

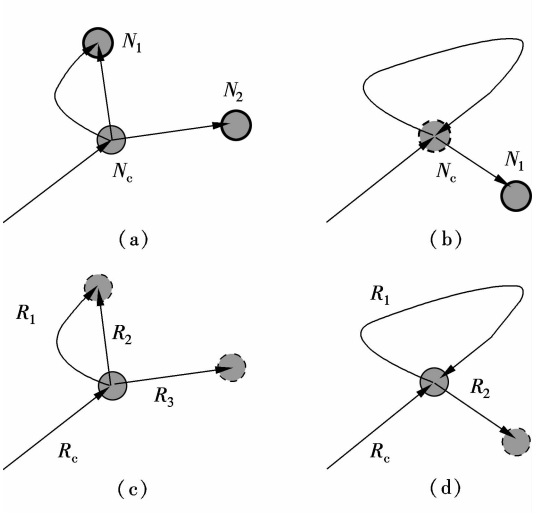

Fig. 9 Problem of traditional A* algorithm and A* search process in an SLSD road network model

With the SLSD road network, each searching step can be considered as the process of finding a directional road whose whole cost is the minimum. The cost can be written as

$$f(x) = \alpha g(x) + \beta h(x) + \gamma m(x) \tag{3}$$

where γ is the weight coefficient; $m(x)$ is the transmission cost of a vehicle from the preceding single-directional road to the next one in each traverse. The value of $m(x)$ depends on the traffic rule (run on the left side or the right side), road crossing complexity, road class, and real-time traffic current, etc.

For retrieving the computed result, the structure anode_slsd_*t* which is similar to anode_*t* is used:

```
struct anode_slsd_t
{
id_t idRoad; //current node
cost_t costCurrent; //current cost
cost_t costFull; //full cost
anode_slsd_t* roadParent; //preceding node
};
```

For path planning algorithms based on tree searching techniques, a basic means to decrease searching cost is tree-trim.

In the classical A* algorithm, if not considering turn restrictions, a basic tree truncation rule can be written as follows: An optimal path will not go through a given node more than twice. If turn restriction is considered, the algorithm will be very complex. Likewise, considering turn restrictions in the SLSD road network, the tree-trim rule of the A* algorithm can be described as follows: An optimal path will not go through the same single-directional road twice or more than twice. Clearly, it is very straightforward since a single-directional road in the SLSD road network corresponds to a "node" in classical road networks.

Therefore, it is proved that with the SLSD road network we can implement path-planning using classical Dijkstra and A* algorithms directly with supporting turn restrictions, road direction, and some other special real-world road network cases.

2.3 Algorithm complexity

For a digital map, the scale of a road network can be described using the number of nodes. Given a simplified road network in Fig. 10, we can use Eq. (4) to evaluate the complexity.

$$S = N^2 \tag{4}$$

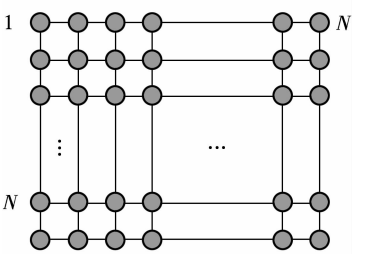

Fig. 10 A simplified road network

In the SLSD roadwork model, the road network complexity evaluated using directional roads can be roughly written as

$$S = 2(2(N(N-1))) = 4N^2 - 4 \tag{5}$$

It seems that more time and memory will be consumed in path finding in the SLSD road network model. However, considering that the classical A* algorithm and the Dijkstra algorithm cannot directly handle scenarios like turn restric-

tions and other cases, the SLSD road network model and its A* and Dijkstra algorithms are valuable because of their simplicity.

3 Experimental Results

Based on the research, we implement a two-step quasi-optimal path finding algorithm. In the system, pre-computing technology is used to calculate optimal path data in a stem road network (high-grade highway), and the A* algorithm is used. Usually we just find the route to a highway, and obtain the best path by searching existing computing results. Fig. 11 shows OPF computing results.

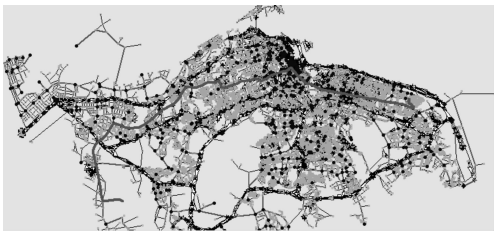

Fig. 11 Test results from a two-step algorithm in Singapore map

4 Conclusion

By investigating various kinds of real road network cases, an SLSD road network model is presented. And it is proved by transferring a super-graph to an SLSD road network that traditional A* and Dijkstra algorithms can be directly used to obtain optimal path finding in any real-world cases without any algorithm modification.

In the future, we will adopt the presented network model and the OPF algorithm in large-scale digital maps for better performance.

References

[1] Zhan F B, Noon C E. Shortest path algorithms: an evaluation using real road networks [J]. *Transportation Science*, 1998, **32**(1): 65 - 73.

[2] Zhao Y L. *Vehicle location and navigation system* [M]. London: Artech House, Inc, 1999.

[3] Zhang X G. The theory and application of vehicle navigation oriented GIS and its map-matching techniques[D]. Nanjing: School of Instrument Science and Engineering of Southeast University, 2001. (in Chinese)

[4] Ness S, Lee T. Single line street network: the foundation of mobile GIS[C]//*Proc of IEEE-IEE Vehicle Navigation & Information Systems Conference*. Ottawa, Canada, 1993: 34 - 37.

[5] Christopher K H. The potential of precision maps in intelligent vehicles[C]//*Proc of IEEE International Conference on Inteilligent Vehicles*. Stuttgart, Germany, 1998: 419 - 422.

[6] Car A, Taylor G, Brunsdon C. An analysis of the performance of a hierarchical wayfinding computational model using synthetic graphs [J]. *Computers, Environment and Urban Systems*, 2001, **25**(1): 69 - 88.

[7] Hisao N, Yuuji Y, Teruo F. Path finding algorithms based on the hierarchical representation of a road map and its application to a map information system [J]. *IPSJ Journal Contents*, 2001, **31**(5): 659 - 665.

[8] Holzer M. Hierarchical speed-up techniques for shortest-path algorithms[D]. Konstanz, Germany: University of Konstanz, 2003.

[9] Li Q Q, Zeng Z, Yang B S. Hierarchical model of road network for route planning in vehicle navigation systems [J]. *IEEE Intelligent Transportation Systems Magazine*, 2009, **1**(2): 20 - 24.

[10] Jing N, Huang Y W, Rundensteiner E A. Hierarchical encoded path views for path query processing: an optimal model and its performance evaluation [J]. *IEEE Transactions on Knowledge and Data Engineering*, 1998, **10**(3): 371 - 388.

[11] Thorup M. Undirected single-source shortest paths with positive integer weights in linear time [J]. *Journal of the Association for Computing Machinery*, 1999, **46**(3): 362 - 394.

[12] Muhammad A A. Integrating OO road network database, cases and knowledge for route finding[C]//*Proc of the* 2001 *ACM Symposium on Applied Computing*. Las Vegas, Nevada, USA, 2001: 215 - 219.

[13] Wagner D, Willhalm T. Geometric speed-up techniques for finding shortest paths in large sparse graphs[C]//*Proc of European Symposium on Algorithms*. Budapest, Hungary, 2003: 776 - 787.

[14] Möhring R H, Schilling H, Schütz B, et al. Partitioning graphs to speed up Dijkstra's algorithm [J]. *The ACM Journal of Experimental Algorithm*, 2006, **11**: 1 - 29.

[15] Morris J. Data structures and algorithms: Dijkstra algorithm [EB/OL]. (1998-12-30) [2010-09-25]. http://www.cs.auckland.ac.nz/software/AlgAnim/dijkstra.html.

基于 SLSD 道路网络模型的最优路径算法

张小国　王　庆　龚福祥

(东南大学仪器科学与工程学院,南京 210096)

摘要:提出了基于单线单向(SLSD)道路网络的最优路径算法. 不同于传统网络,在 SLSD 网络中,路元素被抽象成网络的节点,且都是单向单线的;而道路节点被抽象成网络的链接. 该网络模型可以很好地表述拐弯限制、回路以及多条道路存在于 2 个路口等只有超图模型才能很好表示的真实路网情形. 基于此网络模型,给出了相关的最优路径算法,并且证明了将超图转化为 SLSD 道路网络后,A* 及 Diskstra 算法可以不加修改直接用于计算任何真实路网的最优路径. 最后,结合新加坡道路网络数据,给出了一个预先计算的两步法最优路径算法及其计算结果,验证了所提出的模型和算法.

关键词:最优路径寻找;道路网络模型;概念模型;数字地图;车辆导航系统;A* 算法;Dijkstra 算法

中图分类号:TP208;U666.1