Representation of Transportation Network and Continuous Nearest Neighbor Search

Jun FENG[♥] Naoto MUKAI[♦]
Toyohide WATANABE[♠]

To achieve efficient processes of transportation network, we propose a structure for integrating transportation information with spatial information of road network. Based on such representation, a method for searching continuous nearest neighbors for a specific route in ITS applications is proposed in this paper. Experiment shows that our method outperforms the traditional methods.

1. Introduction

In Intelligent Transportation System (ITS), a query of continuous nearest neighbors (CNN) along a predefined route should be based on the transportation information including transportation routes and current travel cost (e.g., travel time) on segments of road network. The existing work for CNN search was almost presented from the computational geometry perspective [1, 2, 3]. Their CNN search methods for line segments are effective. All the works adopted the straight-line distance between objects as a measure to compute CNN. However, in ITS applications, objects can only move on the pre-defined road network, so the distance between objects should be decided by the shortest path length or least travel cost. We have proposed a method [4] to solve CNN problem on road network. By setting up search regions in the CNN search process Dijkstra's path search algorithm [5] can find shortest path lengths from the route to its CNN effectively. However, because the travel cost is not always in direct proportion to the path length, our method cannot be applied to compute CNN using travel cost, directly. In this paper, we propose a new method for CNN search, which takes the travel junctions (or traffic constraints) and travel cost on road segments and turn corners into consideration.

This paper is organized as follows. A representation method of transportation network is proposed in Section 2. Section 3 introduces a search method for CNN search on the transportation network. Section 4 evaluates our method and Section 5 makes a conclusion on our work.

♥ 学生会員　名古屋大学大学院工学研究科博士後期課程
feng@watanabe.nuie.nagoya-u.ac.jp
♦ 学生会員　名古屋大学大学院情報科学研究科博士後期課程　naoto@watanabe.nuie.nagoya-u.ac.jp
♠ 正会員　名古屋大学大学院情報科学研究科
watanabe@is.nagoya-u.ac.jp

2. Integrated Representation Method of Transportation Network

For queries in ITS, it is important to identify certain classes of attributes that may not be presented in road maps: for example, one-way road with attributes of links, traffic constraints, information about turns between links, or access conditions from one link to another [6]. Moreover, for some important route planning problems, the turn costs are also considered [7] when we make a turn on a cross-point. A representation method for integrating traffic and spatial information about road network is proposed in this section.

A road network with nodes and links representing the crosses and road segments can be regarded as an un-directed graph G, $G = (V, L)$, where V is a set of vertices $\{v_1, v_2, \ldots, v_n\}$, and L is a collection of lines $\{l_1, l_2, \ldots, l_m\}$. Transportation network is regarded as a directed graph G', $G' = (V, A)$, where V is a set of vertices $\{v_1, v_2, \ldots, v_n\}$, and A is a collection of arcs $\{a_1, a_2, \ldots, a_p\}$.

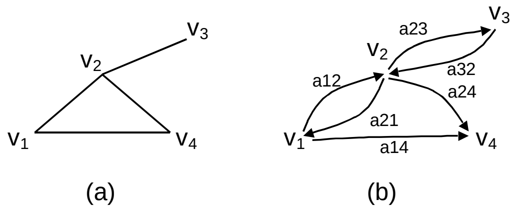

Fig.1 Road segment and traffic arc

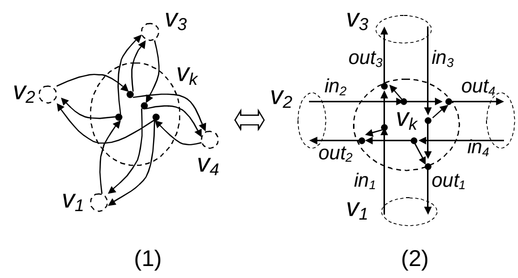

Fig.2 Cross node with constraint

Fig.1 depicts these two kinds of graphs. In the un-directed graph of Fig.1 (a), road segments are represented as lines; while in the directed graph of Fig.1 (b), two-way roads are represented as a pair of arcs. One line for a road segment in Fig.1 (a) may be corresponded to two arcs in Fig.1 (b) for transportation segments in two directions. In addition to the directions of traffic, there are usually traffic controls (constraints) on transportation network: for example, the right-turn and U-turn are forbidden on some cross-points. Fig.2 depicts such a cross-point in transportation network. In Fig.2 (1), the transportation network is depicted by the node-link method (proposed in [8]): each arc depicts a one-way road and each node corresponds to a junction. To represent the traffic constraints and turn costs, the cross-node v_k is split into four nodes and one road segment between v_k and v_i ($i=1 \ldots 4$) is replaced by three arcs. The multiplication of the nodes and links leads to a lower efficiency of processing on the network. Furthermore, this kind of representation method ignores the spatial attributes of map objects, and only the routing queries are applicable well on this model. Therefore, we propose a representation method of transportation network by using *super-node* to integrate traffic information (including traffic cost and traffic constraints) and road network. Our method is similar to that proposed in [9] in keeping traffic information on nodes of road network. However, the information of traffic constraints and turn cost on the nodes was not discussed in [9].

A *super-node* can be defined as a node in the road network: for example, v_k in Fig.2 (2). The information on the *super-node* contains the following parts:

(a) *Cost-arc*: The arc which has v_k as its final vertex is called in-arc, denoted as in_i, and similarly the arc which has v_k as its initial vertex is called out-arc, denoted as out_j. The number of those arcs is called as indegree (e.g., 4) and outdegree (e.g., 4), respectively. Every out_i is defined as a *Cost-arc* which consists of the final vertex of this arc and the traffic cost for traveling through this arc. *Cost-arc*s of v_k in Fig.2 (2) are

$$\begin{bmatrix} out_1(v_1, \mathrm{cost}_{k1}) \\ out2(v_2, \mathrm{cost}_{k2}) \\ out_3(v_3, \mathrm{cost}_{k3}) \\ out_4(v_4, \mathrm{cost}_{k4}) \end{bmatrix}.$$

(b) *Constraint-matrix*: The constraints on the *super-node* can be represented with an $n \times m$ matrix:

$$CM(v_k) = \begin{array}{c} \\ in1 \\ in2 \\ \vdots \\ in_n \end{array}\!\!\begin{array}{c} \begin{array}{cccc} out\ 1 & 2 & \cdots & m \end{array} \\ \begin{pmatrix} C_{11} & C_{12} & \cdots & C_{1m} \\ C_{21} & C_{22} & \ldots & C_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ C_{n1} & C_{n2} & \cdots & C_{nm} \end{pmatrix} \end{array}.$$

C_{ij} equals to *1* when there is a restriction from in_i to out_j, and C_{ij} equals to *0* when there is a junction from in_i to out_j. *Constraint-matrix* for v_k in Fig.2 (2) is:

$$CM(v_k) = \begin{array}{c} \\ in1 \\ in2 \\ in3 \\ in4 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} out\ 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \end{array},$$

where there are restrictions on going from in_1 to out_1 and out_4, from in_2 to out_1 and out_2, from in_3 to out_2 and out_3, and from in_4 to out_4 and out_1.

If there is no restriction for any in_i of the super-node v_k, *Constraint-matrix* of v_k is filled with *0*, and is regarded as *NULL*. Moreover, our method is able to manipulate the turn cost by extending *Constraint-matrix* to a *Turn-Cost/Constraint-matrix*. *CM* can be modified to a *Turn-Cost/Constraint-matrix:*

$$T\_CM(v_k) = \begin{array}{c} \\ in1 \\ in2 \\ \vdots \\ in_n \end{array}\!\!\begin{array}{c} \begin{array}{cccc} out\ 1 & 2 & \cdots & m \end{array} \\ \begin{pmatrix} TC_{11} & TC_{12} & \cdots & TC_{1m} \\ TC_{21} & TC_{22} & \ldots & TC_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ TC_{n1} & TC_{n2} & \cdots & TC_{nm} \end{pmatrix} \end{array}.$$

TC_{ij} ($0 \le TC_{ij} < Max$) is the turn cost from in_i to out_j. When the turn cost equals to *Max*, TC_{ij} means that there is restriction from in_i to out_j. For example, *Turn-Cost/Constraint-matrix* for v_k in Fig.2 (2) may be like this:

$$T\_CM(v_k) = \begin{array}{c} \\ in_1 \\ in_2 \\ in_3 \\ in_n \end{array}\!\!\begin{array}{c} \begin{array}{cccc} out\ 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} Max & 10 & 40 & Max \\ Max & Max & 10 & 30 \\ 10 & Max & Max & 30 \\ 10 & 40 & Max & Max \end{pmatrix} \end{array},$$

where *MAX* is defined as a large constant value.

The element TC_{ij} in this matrix with a value of MAX represents a restriction from in_i to out_j: e.g., U-turn and right-turn are forbidden in this example, so *MAX* is assigned to TC_{ii} (i = *1, 2, 3, 4*), TC_{14}, TC_{21}, TC_{32} and TC_{43}. The value of TC_{12} represents the cost 10 (e.g., 10 seconds) of making a left-turn on the cross-point (from in_1 to out_2), while the cost of crossing the point v_k from in_1 to out_3 is 40.

This method is easy to integrate the traffic information and the basic road network. In the basic road network, the additional traffic information is managed on every node. When the number of nodes and traffic arcs is unchanged, the modification of the traffic information does not injure the stability of the spatial index structure (i.e., R-tree [10]) for road network. Therefore, a kind of queries that refer to the spatial information can be solved effectively by taking advantages of the spatial index structure. Another kind of queries, which refer to traffic information, can also be solved effectively by using proper methods. We center in the following section on solving the second kind of queries with a CNN search example.

3. Continuous Nearest Neighbor Search

The transportation network used by our CNN search is managed in a dataset, denoted as *super-node* dataset, which is generated by the *super-node* representation method proposed in the previous section. The predefined route from a start point v_1 to an end point v_n is given by an array $Route(v_1, v_n) = (v_1, v_2, \ldots, v_{n-1}, v_n)$, and the target object set $\{t_a, t_b, \ldots\}$ is managed by a spatial index structure (e.g., R-tree). The detailed discussions about the processing of target objects with respect to the advantages of spatial index structure can be found in our previous papers [4, 11]. We center on the *super-node* representation method and its influence for CNN search. The *super-node* dataset consists of information about road network and traffic cost on the network. To make straightforward the explanation, we use an abstract *cost* on road network, which can be regarded as the travel time, toll of a path and so on.

3.1 Observation of super-node dataset

We make observations of the *super-node* dataset in CNN search:

(a) Every vertex in the *super-node* dataset keeps the cost information of possible out-arcs, so the travel cost from a vertex v_i on $Route(v_1, v_n)$ to the following vertex v_{i+1} along this route is kept on vertex v_i and denoted as $v_i.cost_{i+1}$. If NN of v_{i+1} is known as t_{i+1} with $cost(v_{i+1}, t_{i+1})$ the travel cost from v_i to its NN t_i is not larger than a value $Cost\text{-}limit(v_i)$ which is computed by:

$$Cost\text{-}limit(v_i) = v_i.\mathrm{cost}_{i+1} + cost(v_{i+1}, t_{i+1})$$

Cost-limit(v_i) is used to set a region for NN search of v_i (e.g., in Fig.3), and NN of v_i can be found only inside the dotted region. The region is defined as a circle with radius of *Cost-limit(v_i)* and center of v_i.

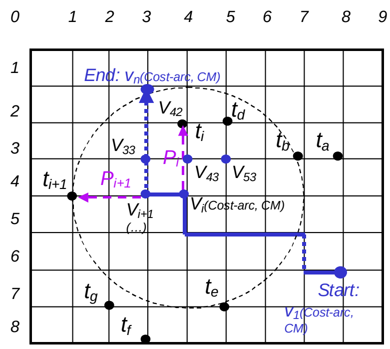

Fig.3 Predefined route and NN for v_i

(b) The nearest target object t_{i+1} of v_{i+1} is also the nearest on the possible paths from v_i via v_{i+1}. In other words, t_{i+1} is the nearest one found on a path from v_i via $v_i.out_{i+1}$. If there is any object being nearer to v_i than t_{i+1}, the shortest path from v_i to it does not pass by v_{i+1}. Certainly, it is possible that there is a path from v_i to t_{i+1} via v_j ($j \neq i+1$), which is shorter than *Cost-limit(v_i)*.

Conclusions can be driven from these observations:

- The path length from v_i to NN t_{i+1} of v_{i+1} can be set as a limit for NN search of v_i;
- NN search of v_i can be executed along the out-arcs of v_i except for $v_i.out_{i+1}$.

Here, we give a simple proof for these conclusions:

(a) t_{i+1} is a candidate of NN search of v_i. Because v_i and v_{i+1} are along the same route *Route(v_1, v_n)*, there is an out-arc of v_i leading to v_{i+1}. Being NN of v_{i+1}, t_{i+1} can also be reached from v_i via v_{i+1}. So, t_{i+1} is possible to be NN of v_i, too.

(b) *Cost-limit(v_i)* is the shortest from v_i to any object via v_{i+1}. If there is another object t' with a path shorter than that of v_{i+1} via v_{i+1}, then t' is also nearer to v_{i+1} than t_{i+1}. This contradicts to the promise that t_{i+1} is NN of v_{i+1}.

3.2 Reverse search method of CNN

We propose a method for CNN search along *Route(v_1, v_n)*. This method begins from the end vertex of this route, and searches NN for every vertex in the reverse order of this route. Our method first searches t_n for the end vertex v_n; and then generates a search limit for the next computation vertex v_{n-1} using the previous result, and checks whether there is an object nearer to v_{n-1} via the out-arcs of v_{n-1} except for $v_{n-1}.out_n$. These steps run in cycle until the computation vertex is v_1. The correctness of this method is assured by the previous observations.

NN search for every vertex can be realized by adopting a priority queue to maintain the current frontier of the search. Any vertex with a higher cost from v_i than the limit value is not inserted into the queue. By expanding the vertex on the head of queue, the algorithm ends when the head vertex connects to a target object. An example for NN search of v_i is given in Fig. 3. NN of v_i is to searched only inside the dotted region. There are assumptions that every grid represents a unit of cost; right-turn and U-turn are forbidden on v_i; and no restrictions are imposed on vertex v_{43}. The search for v_i begins from the following possible out-arcs of v_i: here, $v_i.out_{43}$. As there is no target object connecting to v_{43}, and the cost from v_i to v_{43} is not larger than *Cost-limit(v_i)*, the search expands the vertex v_{43} using the width-first method. The vertex v_{42} connecting to a target object t_i with the lowest cost is found, and the object t_i is regarded as NN of v_i.

4. Experimental Evaluation

Our prototype system was developed in Java on an SGI O2 R5000 SC 180 entry-level desktop workstation (64 M Bytes of main memory, 80 G Bytes of disk and 4 K Bytes per page for disk I/O). The system manages basic road maps in a part of Aichi Prefecture, Japan. Evaluations are made using the transportation network generated by *super-node* method (denoted as SN) and node-link method (denoted as NL). The basic road map is indexed by R-tree. The targets used in our test are point objects with uniform distribution on the road network, which is managed by another spatial index. The number of targets is varied from 3% to 12% of the nodes on the road network.

The size of a record of *super-node* (i.e., a cross node on road network, with four (v_i, *cost*) and 16 *Constraint-matrix* elements) is 72 Bytes (4 * (6 + 8) + 1 * 16), which leads to 56 records per disk page. In NL method, the record for road segment is (from-node, to-node, weight), which is 24 Bytes (8 * 3) and leads to 170 records per disk page. The total number of nodes N_{num} is 42,062 and number of links L_{num} is 60,349 in the basic road map. The average traffic arcs connecting to a node is about 2.87 (=2 L_{num} / N_{num}). When there is no traffic constraint for the basic road map, in NL method, there are 120,798 records (two times of the link numbers in road maps). As in SN method, the amount of information is related to the number of arcs in every node: here, the nodes with four, three, two and one out-arcs are about 24: 51: 13: 12. The total arcs managed in SN method are 120,798. When there are traffic constraints, Right-turn and U-turn are forbidden in about half of the cross and T-junction points. Then in NL method, there are about 142,423 nodes (more than three times of that in SN method); and 135,293 arcs (more than two times of than in SN method). The situations are depicted in Fig. 4. The datasets generated by SN method shares the same value in Fig.4, which is denoted simply as SN method, because the number of objects keeps the same. On the contrary, there are different values for different conditions of the datasets in NL method. “Constraint” means there are traffic constraints in the dataset.

CNN test results are given in Fig. 5. In the figure, x-axis represents the density of targets on the road network, which is the ratio of the targets' number (T_{num}) to the nodes' number (N_{num}) in basic road map; y-axis represents the CPU time of NN search for every computation-point on the predefined route. D represents that CNN search is done by adopting Dijkstra's algorithm for NN searches. When the traffic cost is set as the length of road segment, NN search is done inside a proper search region, which is a circle just like one generated in Section 3; and R represents the reverse search method proposed in this paper. D algorithm is executed on the datasets generated by NL method with/without constraint (the number of objects is depicted in Fig. 4); and R algorithm is done on the dataset generated by SN method with/without constraint.

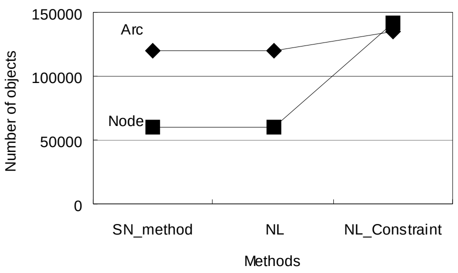

Fig.4 Numbers of arcs and nodes managed by SN and NL methods

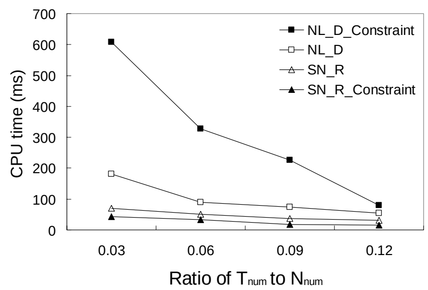

Fig.5 Average CPU time of finding one target in CNN search

In Fig. 5, we can observe that: when the searches are on the datasets without constraint, SN_R method is about 1.7 to 2.6 times faster than NL_D method; when there are traffic constraints managed in the datasets, SN_R (Constraint) method is about 5 to 14.4 times faster than NL_D (Constraint) method. This is because:

(a) In our method, NN search only expands the nodes on road map in possible directions when there is any traffic constraint on the node. With our reverse CNN search method, one or more possible directions are decreased, and the following nodes on that direction need not to be tested. The search cost is shrunken.

(b) In NL_D method, Dijkstra's algorithm is executed inside a proper search region. Therefore, NN search process includes the steps of generating search regions, creating distance matrices for networks inside the search regions and the path searches by using Dijkstra's algorithm based on the matrices. The cost of matrix creation and path search is related to the number of nodes and arcs in the search region. When there are any traffic constraints, the number of nodes and arcs is increased; and the cost is increased, accordingly. The search region can be generated only when there are direct relations between the travel cost and the length of road segment. So, the comparison is only done on this assumption. In other situations, the search region cannot be created easily for NL_D search and our method is much faster than NL_D method.

5. Conclusion

In this paper, we proposed a *super-node* structure for integrating traffic information with spatial information of road network. Because by using this structure, traffic information is embedded in the spatial structure created for road network, the queries, which refer to spatial information and traffic information, could be solved more efficiently than other methods. Experiments on CNN search confirmed it.

[文献]

[1] Y.F. Tao, D. Papadias and Q.M. Shen: "Continuous Nearest Neighbor Search", Proc. of VLDB'02, pp. 287-298 (2002).

[2] Z.X. Song and N. Roussopoulos: "K-Nearest Neighbor Search for Moving Query Point", Proc. of SSTD'01, pp. 79-96 (2001).

[3] S. Bespamyatnikh and J. Snoeyink: "Queries with Segments in Voronoi Diagrams", SODA (1999).

[4] J. Feng and T. Watanabe: "A Fast Method for Continuous Nearest Target Objects Query on Road Network", Proc. of VSMM'02, pp. 182-191 (2002).

[5] N. Christofides: "Graph Theory: An Algorithmic Approach", Academic Press Inc. (London) Ltd. (1975).

[6] M. F. Goodchild: "GIS and Transportation: Status and Challenges", GeoInformatica, Vol.4, No.2, pp. 127-139 (2000).

[7] S. Winter: "Modeling Costs of Turns in Route Planning", GeoInformatica, No.4, pp. 345-361 (2002).

[8] J.Fawcett and P.Robinson: "Adaptive Routing for Road Traffic", IEEE Computer Graphics and Applications, Vol.20, No.3, pp. 46-53 (2000).

[9] D. Papadias, J. Zhang, N. Mamoulis and Y.F. Tao: "Query Processing in Spatial Network Databases", Proc. of VLDB 2003, pp. 802-813　(2003).

[10] A. Guttman: "R-Trees: A Dynamic Index Structure for Spatial Searching", Proc. of ACM SIGMOD'84, pp. 47-57 (1984).

[11] J. Feng and T. Watanabe: "A Fast Search Method of Nearest Target Object in Road Networks", Journal of the ISCIE, Vol. 16, No. 9, pp. 484-491　(2003).

Jun FENG

名古屋大学大学院工学研究科博士後期課程在学中．1994 中国河海大学コンピュータ学院修士課程修了．地理情報システム、空間データベースシステムと空間検索の研究・開発に従事．情報処理学会学生会員．日本データベース学会学生会員．

Naoto MUKAI

名古屋大学大学院情報科学研究科博士後期課程在学中．2003 名古屋工業大学大学院工学研究科博士前期課程修了．地理情報システムと高度交通情報システムの研究・開発に従事．情報処理学会学生会員．日本データベース学会学生会員．

Toyohide WATANABE

名古屋大学大学院情報科学研究科教授．1972 京都大学理学部卒業．1975 同大学工学研究科博士課程中退。工学博士．統合化環境、分散協調環境、データベース環境、学習支援システム、文書理解に興味を持つ．情報処理学会、電子情報通信学会、日本ソフトウェア科学会、人工知能学会、システム制御情報学会、教育システム情報学会、ACM、IEEE-CS、AAAI、AACE 各会員。