A Map Database System for Route Navigation with Multiple Transit Points and Destination Points

Kunihiko Kaneko
Department of Computer Science
Fukuyama University
Fukuyama-Shi, Japan
kaneko@fuip.fukuyama-u.ac.jp

Shinya Honda
Graduate School of Information Science and Electrical Engineering
Kyushu University
Fukuoka-Shi, Japan
2ie14075w@s.kyushu-u.ac.jp

***Abstract*—This paper presents a map database system for route navigation. The system contains database describing roads, interest points (such as a bus stop, store, etc), and route images. The system also includes an index for efficient processing of the shortest path query. Given a set of candidate destination points and a set of candidate transit points, the system generates an index for shortest path query dynamically. Then, a computer system gets a starting point, and the system makes the shortest path that includes one of the candidate destination points and one of the candidate transit points. The k-shortest path (k-SPT) method is used to evaluate the shortest path query. The original version of k-SPT does not consider multiple number of destination points, then the implementation of k-SPT is modified.**

Keywords- map database; shortest path navigation; one-way road; shortest path query; k-shortest path (k-SPT)

I. Introduction

Map databases contain various types of data. They are roads, building, signals and so on. They also often contain color images of the real world. One of the main applications of map databases is route navigation. A system can recommend suitable routes for users using a map database. In this paper, we present a route navigation system utilizing a map database. The system obtains a set of candidate transit points and a set of candidate destination points from users in advance. Then, a user specifies a starting point, and the system will return a shortest path that contains one of the transit points, and one of the destination points. We employed the map database from the OpenStreetMap project [1] in the system.

In this paper, we employed a k-shortest path (k-SPT) method [2] to evaluate shortest path query. Here, the variable k stands for the number of transit points that are contained in the query result. For the problem of high-speed evaluation of k-shortest path query, k-SPT method was proposed. In the method, an index is generated dynamically using a set of candidate transit points and a set of candidate destination points. The system evaluates shorted path quickly using the index. Any user can change starting point. The same index is employed for the same set of candidate transit points and a set of candidate destination points.

TABLE I. Map Database Structure

Entity Set	Table Definition
Node	Node(node_id, lat, lon, classification, feature)

Node List	NodeList(way_id, classification, feature)
	NodeListElement(way_id, seq, node_id)

Element Set	ElemenetSet(relation_id, type, ref, role)
Edge	Edge (way_id, node1, node2)

In this paper, database structure of the map database is explained briefly in Section 2. Evaluation results of shortest path query using k-SPT are explained in Section 3. The results show that k-SPT method works for a single destination point. In Section 4, a route navigation system is explained. The original version of k-SPT does not take into consideration multiple number of destination points, then the implementation of k-SPT is modified.

II. Map Database Structure

There are four types of entity sets to describe a map in map databases as follows.

- **Node**
  Each node represents a particular point in a map.
- **Node list**
  A node list is an ordered list of nodes. It represents a polyline or polygon.
- **Element Set**
  An element set is a set of nodes and node lists. This type of object is employed to represent relationships between nodes and node lists.
- **Edge**
  An edge is a relationship between two nodes.

A map may be described using the above sets. Both a node set and an edge set are used in k-SPT method. Structures of map databases may be different. For example, a map in OpenStreetMap dataset only contains three sets; node set, anode list set and element list set.

We employed a relational database management system to manage map databases. To implement the above sets on the top of relational database management system, we

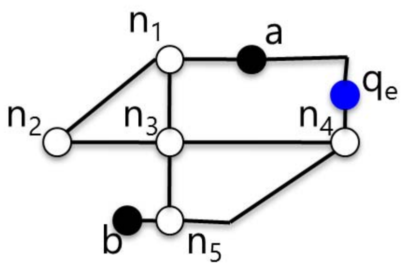


Figure 1. A map example. The destination point is **q$_e$**, and the candidate set of transit points is {**a**, **b**}.

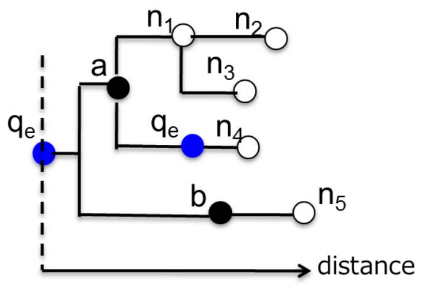


Figure 2. Resulting shortest path tree for the map in Fig. 1. The destination point is **q$_e$**, and the candidate set of transit points is {**a**, **b**}.

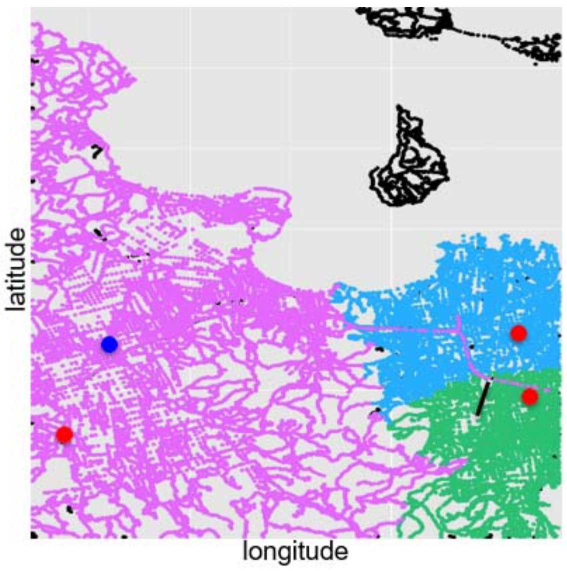


Figure 3. Example of visualization of a shortest path tree. The blue point is the destination point. The three red points are candidate transit points. The three colored areas (magenta, blue and green) in the map show the choice of transit points.

defined the five tables listed in Table 1. The four tables 'Node', 'NodeList', 'ElementSet' and 'Edge' are used to store the four sets node, node list, element set and edge, respectively. We defined the table named 'NodeListElement' to store the order of elements in node lists.

III. K-SPT METHOD

After one starting point, one destination point, and one candidate set of transit points is provided by users, k-SPT method will produce a shorted path. The shortest path is described using a data structure for intermediate data. The data structure is a list, and each element describes the transit point, total distance, and the next node. For example, For the map in Fig. 1, the shortest path from the node **n$_4$** to **q$_e$** is [(**a**, 11, **n$_1$**), (**a**, 11, **n$_3$**), (**a**, 11, **n$_4$**), (**a**, 11, **q$_e$**)].

To evaluate shorted path query efficiently, k-SPT method [2] was proposed. In the method, a shortest path tree (SPT) is created from one destination point and one candidate set of transit points. Note that the same shortest path tree can be used to evaluate a shortest path query when a starting point is changed. Fig. 2 illustrates the shortest path tree for the map in Fig. 1. In the tree, the destination point q$_e$ is the root of the tree.

- **First level of shortest path tree**
  The two edges (**q$_e$**, **a**) and (**q$_e$**, **b**) in the tree represent the distance from **a** to **q$_e$** and from **b** to **q$_e$**, respectively. The distance is co,[iyrf using Dijkstra's algorithm.
- **Other levels of shortest path tree**
  The more-than-one levels of the tree are sub-trees, and the roots of the sub-trees are transit points **a** and **b**. The sub-trees spanning trees. The minimum distance s from transit points to all other nodes are calculated, and each of the other nodes belongs to the spanning tree whose root in the nearest to the node.

The shortest path tree includes all the nodes in the map. Once, a starting point is specified by a user, the starting point is searched using the shortest tree. The path from the root node of the shortest path tree to the node representing the starting point is the shortest path.

Fig. 3 illustrates a shortest path tree. Here, we utilized the OpenStreetMap road dataset. We specified latitude range as from 33.50447 to 33.66863, and longitude range as from 130.21066 to 130.34180 when utilizing a dataset. The range of latitude and longitude covers Fukuoka city in Japan. As showed in Fig. 3, a shorted path tree indicates the relationship between starting points and transit points.

IV. A ROUTE NAVIGATION SYSTEM

We implemented a route navigation system as follows.

A. *Color Image for Transit Points*

In our map database implementation, all transit points have color images. Once the transit point is determined, the color image of the transit point is displayed for route navigation. We add the following new table to manage color image in our map database.

image_url(image_name, url, node_id)

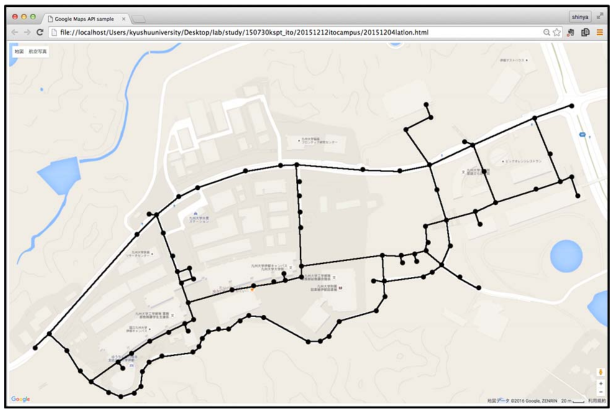
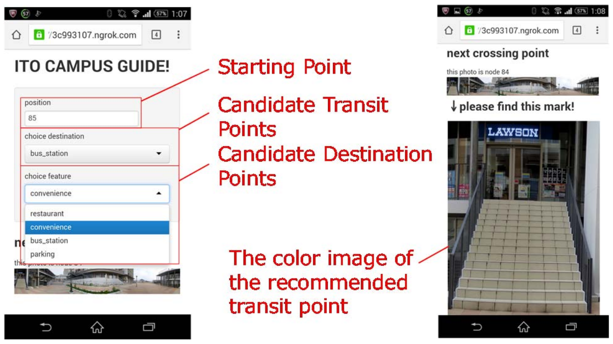


Figure 4. A map database in order to verify our route navigation system.



(a) form (b) image display

Figure 5. Two informative displays of our implementation of a route navigation system.

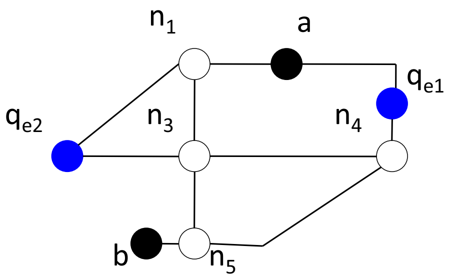


Figure 6. A map example. The candidate destination points are {$\mathbf{q}_{e1}$, $\mathbf{q}_{e2}$} and the candidate transit points are {**a**, **b**}.

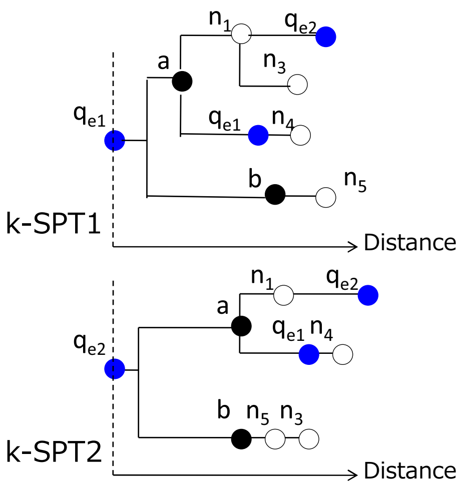


Figure 7. The resulting shortest path trees. There are two different shortest path trees for two transit points.

B. Map Database Population

We employed the OpenStreetMap dataset. All objects in the OpenStreetMap dataset have 'feature' and 'classification' values. We used the objects whose feature value is either 'road' or'footway' to build the road part of a map database. The classification value of road and footway may be 'oneway' or not. For an oneway road, we populate one directed edge in the map database. For other classifications, we populated two directed edges between the same two nodes to simplify system implementation. The black lines in Fig. 4 show the road part of the map database.

We also used the objects whose feature value is either 'bus_stop', 'convenience', 'crossing' or 'parking' to build the point part of a map database. Users can select the point part objects as candidate transit points and candidate destination points. The black points in Fig. 4 show the point part of the map database.

C. Form and Image Display

There is a Web form to specify candidate transit points and candidate destination points by user as Fig. 5 (a). In the form, the current position gotten by GPS is displayed. There is a different image display as Fig. 5 (b). It displays the color image of the recommend transit point in the candidate transit points.

D. Multiple Destination Points

K-SPT method is a way for a single destination point. Users may have multiple candidate destination points, and wish to tour one of the candidate destination points. The original k-SPT method does not take into consideration such situation. Then, we implement in order to select the best destination point among candidate destination points as follows.

*a) **Generation of shortest path trees***

First, shortest path trees are obtained using k-SPT method. For example, two shortest path trees are generated as showed in Fig. 7 for the map in Fig. 6. Here, there are two candidate destination points, then, two shortest path trees are generated.

*b) **Choice of shortest path tree***

As explained in Section 3, all the nodes in the shortest path trees have a field describing the distance to the destination point. For example, the node $\mathbf{n}_4$ in k-SPT1 in Fig 7 has '11' value as distance, and the node $\mathbf{n}_4$ in k-SPT2 in Fig 7 has '12' value as distance. We employed the distance values

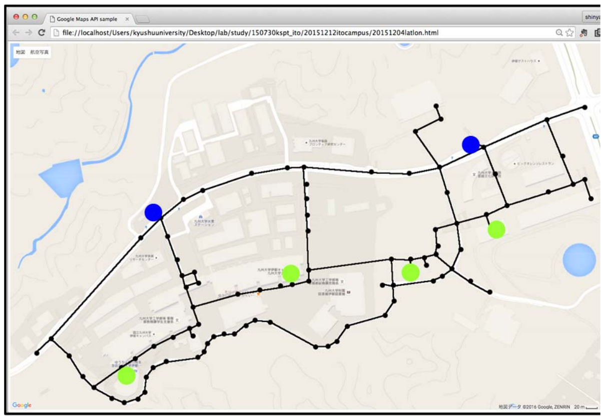


Figure 8. The bus stops and convenience stores in the map. The blue circles are bus stops, and green circles are convenience stores.

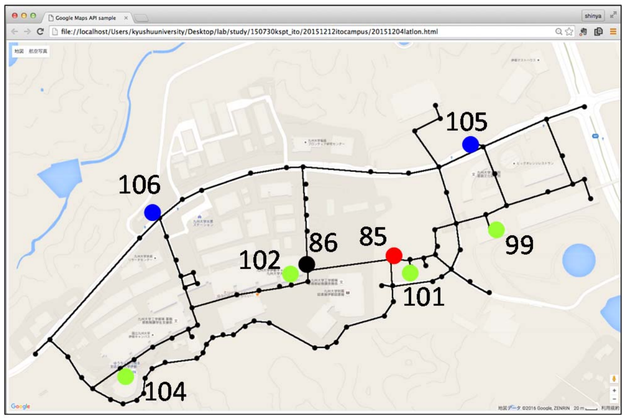


Figure 9. The starting point is 85. All the bus stops and convenience stores have unique numbers.

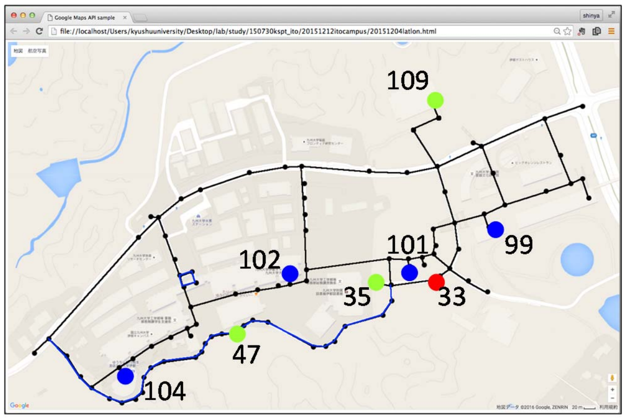


Figure 10. The starting point is 33. The candidate transit points are {35, 47} (parking areas). The candidate destination points are {99, 102, 102, 104} (convenience stores).

to select the best destination point. Given a starting point, the shortest path tree in which the node describing the starting point has the minimum distance value is selected. Note that a starting point may change. As the result, the destination point may change as a starting point may change.

E. Route Navigation Example

There are two bus stops, and four convenience stores in the map database as showed in Fig. 8. We suppose that a user wants to reach one of any bus stops via one of any convenience stores.

Here, we assume that the current position of a user is '85' in Fig. 9. The system generates two shortest path trees named 'KSPT105' and 'KSPT106' for two bus stops '105' and '106', respectively. For the node '85' in the two shortest path trees has the following values, respectively.

KSPT105: [99, 487.13]

KSTP106: [102, 541.17]

It means that if the destination point '105' is selected, the total distance from '85' to '105' is '487.13', and the transit point should be '99'. It also means that if the destination point '106' is selected, the total distance from '85' to '106' is '541.17', and the transit point should be '102'. In this case, the destination point '105' is nearer than '106'. Then, the destination point '105', and the transit point '99' are recommended to the user.

If the starting point is changed to be '86', the result is different as follows.

KSPT105: [99, 616.63]

KSTP106: [102, 411.67]

In this case, the destination point '106', and the transit point '102' are recommended to the user.

Fig. 10 is another example. We suppose that a user wants to reach one of any convenience stores via one of any parking areas. The blue points are convenience stores, and the green points are parking areas in Fig. 10. We also suppose that the current position of the user is '33'. The system will generate the following results.

KSPT99: [35, 354.60]

KSPT101: [35, 185.16]

KSPT102: [35, 325.43]

KSPT104: [35, 565.44]

In this case, both the destination point '101' and the transit point '35' are recommended to the user.

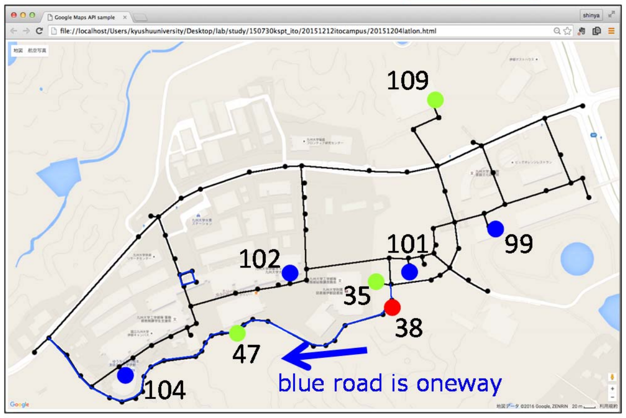


Figure 11. The starting point is changed to be 38. The blue road in the map is an oneway.

If the starting point is changed to be '38' in Fig. 11, the result is as follows.

KSPT99: [47, 882.20]

KSPT101: [47, 712.76]

KSPT102: [47, 853.03]

KSPT104: [47, 633.42]

In this case, both the destination point '104' and the transit point '101' are recommended to the user. The system considers an oneway road correctly.

ACKNOWLEDGMENT

This work is partially supported by JSPS KAKENHI Grant Number 15H05708.

REFERENCES

[1] OpenStreetMap Project Web page: https://www.openstreetmap.org

[2] Sarana Nutanong, Egemen Tanin, Jie Shao, Rui Zhang and Kotagiri Ramamohanarao, "Continuous Detour Queries in Spatial Networks," IEEE Transactions on Knowledge and Data Engineering, vol. 24, no. 7, pp. 1201-1215, 2012.