SOLVING QUICKEST PATH PROBLEM USING AN EXTENDED CONCEPT OF TRANSITIVE CLOSURE METHOD

Rolly Intan
Faculty of Industrial Technology, Informatics Engineering Department, Petra Christian University
e-mail: rintan@petra.ac.id

ABSTRACT: A travel company offers a transportation service by picking up customers from certain places and sending them to the other places. Concerning traffic in time domain function for every path, the problem is how to find the quickest paths in picking up the customers. The problem may be considered as the quickest path problem. We may consider the quickest path problem as an extension of the shortest path problem. The quickest path problem optimizes length of time in visiting some places. The quickest path is not always the shortest path. In the relation to the problem, this paper proposes an algorithm for solving the quickest path problem. The algorithm is constructed by modifying a concept of transitive closure method.

Keywords: transitive closure, the quickest path, the shortest path, graph.

INTRODUCTION

Today, the competition in the business world is very tight, because the number of companies working in the same field and same place are increasing. One of the methods to win the competition, a company should give a better service than the others. For example, a travel company could give a good service by arranging a schedule of picking up its customers by which the customers do not need to wait for long time to be picked up as well as waste their time in the car due to unnecessary long trip. By considering that every connecting path (street) has different degree of traffic in time domain function, the problem is how to find the quickest paths if the car departures at a certain time in picking up the customers. We may consider the problem as the quickest path problem. The quickest path is not always the shortest path. Since every path in the quickest path problem as described above is not only weighted by a single value instead it is represented by a time domain traffic function, we may consider the quickest path problem as an extension problem of the shortest path.

To solve this problem, this paper proposes an algorithm to find the quickest paths by using an extended concept of Transitive Closure Method. Simply, a weighted directed graph (digraph) can be used to represent the connections of all possible locations of customers, where vertices and edges of digraph express locations and their paths, respectively. Directions of paths are represented by the directions of edges. Every weight of each edge is a traffic time domain function.

The structure of this paper is the following. In the following section, basic concept of transitive closure method is described briefly. It is started by explaining a directed graph, and adjacency matrix as used to represent the graph. The subsequent section is devoted to propose an extended concept of transitive closure in order to solve the quickest path problem. A simple illustrative example is given to well understand the concept. Finally, a conclusion is given to wrap up the paper.

CONCEPT OF TRANSITIVE CLOSURE

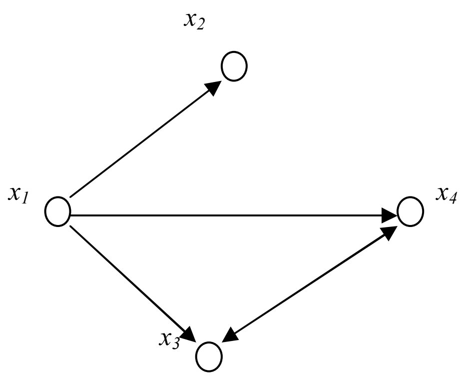

Figure 1. An Example of Directed Graph

As explained in [1], let G be a graph (directed graph). Let B be a matrix whose rows are labeled by the vertices in the graph and whose columns are labeled by the same vertices in the same order. The entry in the ith row and jth column of B, denoted by b_{ij}, is equal to 1 if there is an edge (directed edge) from the ith vertex to the jth vertex and is 0 otherwise. The matrix B is called adjacency matrix of the graph G. For instance, let G be the directed graph in Figure 1. The adjacency matrix is shown in Figure 2.

The graph has 4 vertices (x_1, x_2, x_3 and x_4) and 5 directed edges connecting x_1 to x_2, x_1 to x_3, x_1 to x_4, x_3 to x_4 and x_4 to x_3.

$$\begin{array}{c|c} & \begin{matrix} x_1 & x_2 & x_3 & x_4 \end{matrix} \\ \begin{matrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{matrix} & \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Figure 2. Adjacency matrix

In many cases the labels of the vertices are not important. Such a case we will give the matrix without the label. Thus the matrix in Figure 2 can be represented simply by:

$$B = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Suppose that there are n vertices, B^2 can be obtained by the following operation:

$$B^2 = B \circ B$$
$$= \begin{bmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nn} \end{bmatrix} \circ \begin{bmatrix} b_{11} & \cdots & b_{1n} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nn} \end{bmatrix}$$
$$= \begin{bmatrix} b^2_{11} & \cdots & b^2_{1n} \\ \vdots & \ddots & \vdots \\ b^2_{n1} & \cdots & b^2_{nn} \end{bmatrix}$$

where $b^2_{ij} = \sup_{k=1}^{n}\{\min(b_{ik}, b_{kj})\}$.

Similarly, it can be recursively proved that $B^{u+s} = B^u \circ B^s$ for $u, s \in \{1, \cdots, n-2\}$, and $2 \le u + s \le n-1$.

For example, let B be an adjacency matrix as given in Figure 2 B^2 can be calculated and obtained by operating B to B itself as given by the following operation.

$$B^2 = B \circ B$$
$$= \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \circ \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$
$$= \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Look, for example, at

$$\begin{aligned} b^2_{13} &= \sup\{\min(b_{11}, b_{13}), \min(b_{12}, b_{23}), \min(b_{13}, b_{33}), \min(b_{14}, b_{43})\} \\ &= \sup\{\min(0,1), \min(1,0), \min(1,0), \min(1,1)\} \\ &= \sup\{0,0,0,1\} \\ &= 1 \end{aligned}$$

Notice the value of b^2_{13} is 1 because b_{14} and b_{43} are both 1, which means there is an edge from vertex x_1 to vertex x_4 and from vertex x_4 to vertex x_3. Therefore, there is a 2-path from vertex x_1 to vertex x_3. In general, we can see that $b^2_{ij} = 1$ if and only if there is a k such that $\min(b_{ik}, b_{kj}) = 1$, or, in other words, there is an edge from vertex x_i to vertex x_k and from vertex x_k to vertex x_j. Similarly for b^2_{24} we have

$$\begin{aligned} b^2_{24} &= \sup\{\min(b_{21}, b_{14}), \min(b_{22}, b_{24}), \min(b_{23}, b_{34}), \min(b_{24}, b_{44})\} \\ &= \sup\{\min(0,1), \min(0,0), \min(0,1), \min(0,0)\} \\ &= \sup\{0,0,0,0\} \\ &= 0 \end{aligned}$$

so that $b^2_{24} = 0$ because there are no edges from vertex x_2 to vertex x_k and from vertex x_k to vertex x_4 for any fixed k. In other words, there are no 2-paths from vertex x_2 to vertex x_4. We conclude then that b^2_{ij}=1 if there is a 2-path from vertex x_i to vertex x_j and b^2_{ij}=0 if there is no 2-path from vertex x_i to vertex x_j. Similarly, it can be proved that there is a k-path from x_i to x_j for $1 \le k \le n$ if and only if b^k_{ij}=1.

Transitive closure of adjacency matrix B is represented by:

$$B^T = \begin{bmatrix} b^T_{11} & \cdots & b^T_{1n} \\ \vdots & \ddots & \vdots \\ b^T_{n1} & \cdots & b^T_{nn} \end{bmatrix}$$

where $b^T_{ij} = \sup_{k=1}^{n-1}\{b^k_{ij}\}$.

Here, $b^T_{ij} = 1$ if and only if there is at least one path from x_i to x_j. For example, in the relation to adjacency matrix B as shown in Figure 2, transitive closure of B is given by:

$$B^T = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Look, for example, at

$$b^T_{33} = \sup\{b_{33}, b^2_{33}, b^3_{33}\} = \sup\{0,1,0\} = 1.$$

EXTENDED CONCEPT OF TRANSITIVE CLOSURE

To solve the quickest path problem, the concept of transitive closure method as explained in Section 2 is extended as follows. Let $X = \{x_1, x_2, \cdots, x_n\}$ be a set of vertices. R is an extended adjacency matrix as defined by:

$$R = \begin{bmatrix} <\delta_{11}, r_{11}(t)> & <\delta_{12}, r_{12}(t)> & \cdots & <\delta_{1n}, r_{1n}(t)> \\ <\delta_{21}, r_{21}(t)> & <\delta_{22}, r_{22}(t)> & \cdots & <\delta_{2n}, r_{2n}(t)> \\ \vdots & \vdots & \ddots & \vdots \\ <\delta_{n1}, r_{n1}(t)> & <\delta_{n2}, r_{n2}(t)> & \cdots & <\delta_{nn}, r_{nn}(t)> \end{bmatrix}$$

where $r_{ij} : T \rightarrow Length\ of\ Time$ is a function as a maaping from $t \in T$ time to the length of time (in seconds/minutes/hours) representing length of time needed from x_i to x_j at the time *t*. $\delta_{ij} \in \{0,1\}$ is a parameter to determine whether there is a path or not from x_i to x_j. If $\delta_{ij} = 1$ then there is a path from x_i to x_j, otherwise there is no path from x_i to x_j. For example, as shown in Table 1, it takes 14 minutes to go from x_1 to x_2 at 00.00 as well as 01.00 am.

Table 1. Length of Time from x_1 to x_2

T (time)	r_{12} (t) (minutes)
00.00	14
01.00	14
02.00	13
⋮	⋮
22.00	15
23.00	15

$R^2 = R \circ R$ is a R power 2 adjacency matrix providing information of connection among every $x \in X$ that can be reached through 2 paths as given by:

$$R^2 = \begin{bmatrix} <\delta^2_{11}, r^2_{11}(t)> & <\delta^2_{12}, r^2_{12}(t)> & \cdots & <\delta^2_{1n}, r^2_{1n}(t)> \\ <\delta^2_{21}, r^2_{21}(t)> & <\delta^2_{22}, r^2_{22}(t)> & \cdots & <\delta^2_{2n}, r^2_{2n}(t)> \\ \vdots & \vdots & \ddots & \vdots \\ <\delta^2_{n1}, r^2_{n1}(t)> & <\delta^2_{n2}, r^2_{n2}(t)> & \cdots & <\delta^2_{nn}, r^2_{nn}(t)> \end{bmatrix}$$

where $\delta^2_{ij} = \sup_{k=1}^{n}\{\min(\delta_{ik}, \delta_{kj})\}$, and

$r^2_{ij}(t) = \inf_{p \in M}\{r_{ip}(t) + r_{pj}(t + r_{ip}(t))\}$ for

$M = \{p \mid \min(\delta_{ip}, \delta_{pj}) = 1\}$.

R^3 is a R power 3 adjacency matrix providing information of connection that can be reached through 3 paths, and so on. Since there are *n* vertexes, the longest possible path is *n*-1 paths. Therefore, it is necessary to look for up to R^{n-1} adjacency matrix as given by:

$$R^{n-1} = \begin{bmatrix} <\delta^{n-1}_{11}, r^{n-1}_{11}(t)> & <\delta^{n-1}_{12}, r^{n-1}_{12}(t)> & \cdots & <\delta^{n-1}_{1n}, r^{n-1}_{1n}(t)> \\ <\delta^{n-1}_{21}, r^{n-1}_{21}(t)> & <\delta^{n-1}_{22}, r^{n-1}_{22}(t)> & \cdots & <\delta^{n-1}_{2n}, r^{n-1}_{2n}(t)> \\ \vdots & \vdots & \ddots & \vdots \\ <\delta^{n-1}_{n1}, r^{n-1}_{n1}(t)> & <\delta^{n-1}_{n2}, r^{n-1}_{n2}(t)> & \cdots & <\delta^{n-1}_{nn}, r^{n-1}_{nn}(t)> \end{bmatrix}$$

where a recursive formula for δ^{n-1}_{ij} and $r^{n-1}_{ij}(t)$ can be given by:

$\delta^{n-1}_{ij} = \sup_{k=1}^{n}\{\min(\delta^{n-2}_{ik}, \delta_{kj})\}$, and

$r^{n-1}_{ij}(t) = \inf_{p \in M}\{r^{n-2}_{ip}(t) + r_{pj}(t + r^{n-2}_{ip}(t))\}$ for

$M = \{p \mid \min(\delta^{n-2}_{ip}, \delta_{pj}) = 1\}$.

In general, it can be proved that the above formulas can be represented by:

$\delta^{u+s}_{ij} = \sup_{k=1}^{n}\{\min(\delta^u_{ik}, \delta^s_{kj})\}$, and

$r^{u+s}_{ij}(t) = \inf_{p \in M}\{r^u_{ip}(t) + r^s_{pj}(t + r^u_{ip}(t))\}$,

where $M = \{p \mid \min(\delta^u_{ip}, \delta^s_{pj}) = 1\}$,

for $u, s \in \{1, \cdots, n-2\}$, and $2 \le u + s \le n - 1$.

Finally, a transitive closure adjacency matrix, R^T, can be obtained by the following formulas.

$$R^T = \begin{bmatrix} <\delta^T_{11}, r^T_{11}(t)> & <\delta^T_{12}, r^T_{12}(t)> & \cdots & <\delta^T_{1n}, r^T_{1n}(t)> \\ <\delta^T_{21}, r^T_{21}(t)> & <\delta^T_{22}, r^T_{22}(t)> & \cdots & <\delta^T_{2n}, r^T_{2n}(t)> \\ \vdots & \vdots & \ddots & \vdots \\ <\delta^T_{n1}, r^T_{n1}(t)> & <\delta^T_{n2}, r^T_{n2}(t)> & \cdots & <\delta^T_{nn}, r^T_{nn}(t)> \end{bmatrix}$$

where $\delta^T_{ij} = \sup_{k=1}^{n-1}\{\delta^k_{ij}\}$ and $r^T_{ij} = \inf_{k=1}^{n-1}\{r^k_{ij} \mid \delta^k_{ij} = 1\}$.

R^T informs all connections among all $x \in X$ that can be reached from all possiblility paths. Moreover, r^T_{ij} expresses the minimum length of time from x_i to x_j. For example, let consider directed graph G as given in Figure 1, where X = $\{x_1, x_2, x_3, x_4\}$ be the set of vertices. The length of time needed in every path (edge) is shown in Table 2.

Suppose that the car departs at 8.00 am, from Table 2, we can obtain the following adjacency matrices:

$$R = \begin{bmatrix} \langle 0,0 \rangle & \langle 1,4 \rangle & \langle 1,3 \rangle & \langle 1,8 \rangle \\ \langle 0,0 \rangle & \langle 0,0 \rangle & \langle 0,0 \rangle & \langle 0,0 \rangle \\ \langle 0,0 \rangle & \langle 0,0 \rangle & \langle 0,0 \rangle & \langle 1,2 \rangle \\ \langle 0,0 \rangle & \langle 0,0 \rangle & \langle 1,2 \rangle & \langle 0,0 \rangle \end{bmatrix}$$

$$R^2 = \begin{bmatrix} \langle 0,0 \rangle & \langle 0,0 \rangle & \langle 1,10 \rangle & \langle 1,5 \rangle \\ \langle 0,0 \rangle & \langle 0,0 \rangle & \langle 0,0 \rangle & \langle 0,0 \rangle \\ \langle 0,0 \rangle & \langle 0,0 \rangle & \langle 1,4 \rangle & \langle 0,0 \rangle \\ \langle 0,0 \rangle & \langle 0,0 \rangle & \langle 0,0 \rangle & \langle 1,4 \rangle \end{bmatrix}$$

$$R^3 = \begin{bmatrix} \langle 0,0\rangle & \langle 0,0\rangle & \langle 1,7\rangle & \langle 1,12\rangle \\ \langle 0,0\rangle & \langle 0,0\rangle & \langle 0,0\rangle & \langle 1,0\rangle \\ \langle 0,0\rangle & \langle 0,0\rangle & \langle 0,0\rangle & \langle 1,6\rangle \\ \langle 0,0\rangle & \langle 0,0\rangle & \langle 1,6\rangle & \langle 0,0\rangle \end{bmatrix}$$

Table 2. Length of time

$t_{(time)}$	x_1-x_2 (hours)	x_1-x_3 (hours)	x_1-x_4 (hours)	x_4-x_3 (hours)	x_3-x_4 (hours)
00.00	5	3	1	2	2
01.00	2	4	5	6	6
02.00	2	3	5	7	7
03.00	4	6	4	2	2
04.00	2	5	7	4	4
05.00	8	9	6	5	5
06.00	5	7	4	8	8
07.00	4	10	7	9	9
08.00	4	3	8	2	2
09.00	6	10	9	4	5
10.00	4	6	9	8	9
11.00	5	10	5	2	2
12.00	4	9	6	7	8
13.00	6	9	12	10	10
14.00	11	7	9	5	5
15.00	16	10	5	8	8
16.00	7	8	12	9	9
17.00	7	4	11	6	6
18.00	3	6	12	8	8
19.00	9	18	16	7	7
20.00	10	8	5	9	9
21.00	10	12	8	6	6
22.00	6	7	10	10	19
23.00	10	9	7	10	15

R informs connection between every two vertexes that can be reached in only one path. It can be clearly seen that there are 4 directed paths connecting x_1 to x_2, x_1 to x_3, x_1 to x_4 and x_3 to x_4 with the length of time 4, 3, 8 and 2 hours, respectively.

R^2 informs connection between every two vertexes that can be reached in two paths. In this case, there is only one path from x_1 to x_4 via x_3 with the length of time 5 hours (3 hours from x_1 to x_3 plus 2 hours from x_3 to x_4). Thus, it is faster going from x_1 to x_4 via x_3 than directly going from x_1 to x_4.

R^3 informs connection between every two vertexes that can be reached in three paths. Here, there is no connection that can be reached in three paths.

Finally, from R, R^2 and R^3, we can provide R^T as follows.

$$R^T = \begin{bmatrix} \langle 0,0\rangle & \langle 1,4\rangle & \langle 1,3\rangle & \langle 1,5\rangle \\ \langle 0,0\rangle & \langle 0,0\rangle & \langle 0,0\rangle & \langle 0,0\rangle \\ \langle 0,0\rangle & \langle 0,0\rangle & \langle 0,0\rangle & \langle 1,2\rangle \\ \langle 0,0\rangle & \langle 0,0\rangle & \langle 0,0\rangle & \langle 0,0\rangle \end{bmatrix}$$

From R^T, if the car departs at 8.00 am and starts from x_1, it takes 4 hours to reach x_2, 3 hours for going to x_3 and 5 hours to go to x_4 via x_3. Also, the car needs 2 hours for going from x_3 to x_4. Otherwise, there is no any more paths for other connections.

As explained in Section 1, the car has to pick up some customers from different places. Therefore, the problem is how to find the best order of picking up the customers in order to get the most optimum (minimum) total length of time. Let the car has to pick up m customers from m different places. In order to find the most optimum of total time, we have to examine all possible sequential orders of picking up the customers. Thus, the total combinations of sequential orders is equal to $m!$. If the car starts from x_0 and it has to pick up three customers, x_2, x_4 and x_6, then there are 6 (= 3!) combinations of sequential orders that are:

$x_0 \to x_2 \to x_4 \to x_6$, $x_0 \to x_2 \to x_6 \to x_4$, $x_0 \to x_4 \to x_2 \to x_6$, $x_0 \to x_4 \to x_6 \to x_2$, $x_0 \to x_6 \to x_2 \to x_4$ and $x_0 \to x_6 \to x_4 \to x_2$.

Total time of every combination of sequential order is calculated. The sequential order that has the most optimum (minimum) length of time will be chosen as the order of picking up the customers.

For simple example as given in Figure 1, let the car departures from x_1 and it has to pick up two customers, x_3 and x_4. There are two possible orders for picking up the customers: $x_1 \to x_3 \to x_4$ and $x_1 \to x_4 \to x_3$. Let say that the car departs at 8.00 am. As given in the previous calculations, the best sequential order is $x_1 \to x_3 \to x_4$. Total time required to complete picking up x_3 and x_4 is 5 hours. The car takes 3 hours to pick up x_3, and then 2 hours to pick up x_4.

CONCLUSION

This paper proposed an extended concept of transitive closure method in order to solve the quickest path problem. The concept was started by considering the weight of each edge of a directed graph is given by a traffic time domain function. To represent such kind of graph, an extended concept of adjacency matrix was introduced, where every cell of the matrix consists of two parameters, $\delta_{ij} \in \{0,1\}$, to determine existence of edge, and $r_{ij}(t)$, a traffic time domain function from x_i to x_j. Finally, the extended transitive closure of the adjacency matrix was proposed to solve the quickest path problem.

Implementation of the proposed concept can be found in a final project written by Suchiana Cahyono [4].

REFERENCES

1. Bakken, James A. Anderson, *Discrete Mathematics with Combination*.2 nd.

2. "Graph." National Institute of Standards and Technology (NIST). 10 Januari. 2005. <*http://www.nist.gov/dads/HTML/graph.html*>

3. Klir, George J., and Bo Yuan. Fuzzy *Sets and Fuzzy Relation. Theory and Applications*. New Jersey: Prentice Hall, 1995.

4. Shuciana Chahyono, *Perancangan Dan Pembuatan Aplikasi Pencarian Rute Jalan Tercepat Berdasarkan Tingkat Kemacetan Lalu Lintas Menggunakan Transitive Closure Method,* Skripsi, Jurusan Teknik Informatika, UK. Petra, 2005.