Methods of Searching Information Using Legacy Data and Variable Weighted Graph

JUBAEG KIM, JUNGHO PARK, DONGJOON CHA, DEOKMO KU
Department of Computer and Information Science
Sunmoon University
Tangjung-myeon, Asan, Chungnam
KOREA

Abstract: Currently, various information retrieval methods have been implemented and are used on the Internet. For example, the existing shortest-path system obtains the information from the shortest-path database (DB), which is constructed in advance using Dijkstra's algorithm. However, employing a subway, if an accident disrupts the path between any two stations, the shortest-path DB does not remain intact, and must be reconstructed from scratch using Dijkstra's algorithm. In this paper, we propose a more efficient shortest-path searching method that does not require reconstructing the entire path from scratch.

And, in the case of the shortest path, the transit process may have a great influence on the shortest-path search. Existing methods, such as Dijkstra's algorithm, might offer unavailable shortest-path information because they do not consider the condition of transit passengers. In this paper, we propose a variable weighted graph constructed from information about the transit passenger's condition. We propose a method that obtains the available shortest-path information using a variable weighted graph. In addition, we describe our retrieval system for offering the available shortest-path information in the light of the transit passenger's condition.

Key-Words: Dijkstra's algorithm, legacy data, variable weighted graph, shortest path, transit

1 Introduction

The Internet has become an integral part of our lives, and most of us use the Web to send and receive e-mails and to find information. A Web site that searches for this information is called a retrieval engine, and various ways exist to look for information, such as by subject, keywords, or natural language.

Of the existing retrieval methods used to obtain the shortest-path information, Dijkstra's algorithm is the most efficient. Existing systems using Dijkstra's algorithm obtain the shortest-path information from the shortest-path database (DB), which is constructed in advance by Dijkstra's algorithm. However, consider a railroad in the real world. An accident, such as electrical trouble, can disrupt the path between any two stations. Then, the existing shortest-path DB cannot be used as is. Consequently, after the shortest-path DB is reconstructed from scratch, the shortest path is then retrieved and offered from the DB.

When an accident occurs, the original graph changes and the new graph contain the shortest path of the original graph, *i.e.*, the graph before the accident. This shortest path is the old solution. In the new graph, the old solution cannot be used intact. In this paper, we propose an algorithm that efficiently computes the shortest path using the old solution, which is referred to as the legacy data. Our algorithm does not reconstruct the entire path for the graph after the change from scratch. By using the legacy data, we omit the uninjured part of the shortest path and reconstruct only the part that was affected by the change.

Consider the case of the shortest path in a subway. The transit process might have a great influence on the shortest path search. For example, pregnant women or old people need more time in subway transit. The previous methods, such as Dijkstra's algorithm, might offer unavailable shortest-path information because they do not consider the condition of a transit passenger, for example, the passenger's health.

In order to search the shortest-path information efficiently according to the transit passenger's condition, we suggest a variable weighted graph that uses the information on the passenger's condition and the

This study was supported by the Ministry of Information and Communication of Korea under the Information Technology Research Center Support Program supervised by the Institute of Information Technology Assessment (IITA-2006-C1090-0603-0020).

original weighted graph. The information of the original weighted graph consists of the transfer time between any two stations and the transit time at any transit station. In addition, we propose a method that offers the available shortest-path information using the variable weighted graph.

Moreover, we developed a system that offers the shortest-path information in the light of the transit passenger's condition. We describe our system, which is a mobile version, but can also be used, on a PC.

2 A Method of Searching Information Utilizing the Legacy Data

If a starting point and destination are given, the existing system for finding the shortest path offers the information from the shortest-path DB, which is constructed in advance using Dijkstra's algorithm. If a problem exists on the path between any two stations, the existing shortest-path DB cannot be used in the graph after the change. Therefore, the shortest path for the graph after the change must be reconstructed from scratch using Dijkstra's algorithm.

In this section, we present a more efficient algorithm for computing the shortest path that uses the old solution, which is referred to as the legacy data. In our algorithm, we do not reconstruct the entire path for the graph after the change from scratch. By using the legacy data, we reconstruct only the part that was affected by the change.

2.1 Dijkstra's algorithm and the weighted graph

The existing shortest-path methods, such as Dijkstra's algorithm, are applied to weighted graphs, as shown in Fig. 1. For example, the numbers assigned to each edge in Fig. 1 indicate the transfer time between any two stations. Dijkstra's algorithm determines the shortest path between the start node (node A) and the destination node (node B) by repeatedly deciding the shortest path for the node having the smallest sum of edge weights among the nodes that do not form the shortest path.

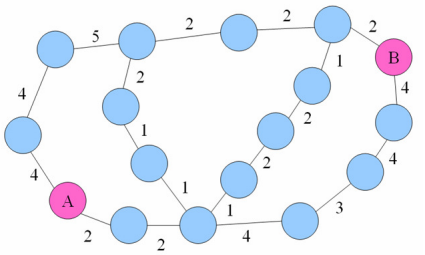
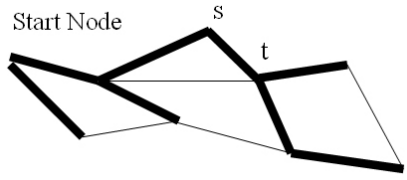


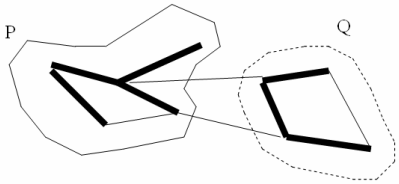
Fig. 1 An example of a weighted graph G

2.2 Our algorithm for searching information using legacy data

When weighted graph G becomes G' as the result of a modification to its edges or nodes, the shortest path of G remains in G'. It is called the old solution. At this time, the shortest-path DB of the original weighted graph G has already been constructed. That is, the bold lines in Fig. 2(a) show the shortest path between the start node and the other nodes. Suppose that edge (s, t) is deleted. Figure 2(b) shows the weighted graph G' that arises from G. Even if weighted graph G has been changed into G', the shortest path information for G remains in G' (Fig. 2(b)). We call the shortest path of G that remains in G' the legacy data.



(a) G and its shortest path



(b) The new weighted graph G' and the legacy data

Fig. 2 An example of legacy data

The idea is that our algorithm promotes efficiency by using the legacy data. In graph G', the legacy data are divided into P and Q, and the shortest paths of the nodes within part P are not changed, although the shortest paths involving the nodes in part Q might change. Our algorithm utilizes this property.

In our algorithm, we omit the uninjured part of the shortest path (*i.e.*, part P) and reconstruct only the shortest path that might have been injured by the change. That is, we reconstruct only the path for the nodes contained in part Q by applying Dijkstra's algorithm.

2.3 The complexity of our algorithm

The time complexity of Dijkstra's algorithm on a weighted graph G containing n nodes and e edges is $O(n^2)$. Suppose that there are n' nodes in G'. Therefore, the time complexity of the algorithm that reconstructs the shortest path from scratch becomes $O(n'^2)$.

Our algorithm repeatedly iterates the procedure that decides only the shortest path for the nodes in part Q by applying Dijkstra's algorithm. This procedure has O(n') time complexity. If the number of nodes in part Q is q ($0 \leqq q \leqq n'$), this procedure is repeated q times. Therefore, the time complexity of our algorithm becomes O($n'q$). It is obvious that our algorithm is more efficient than the method that reconstructs the entire shortest path of G' from scratch.

3 A Method of Searching Information Using the Variable Weighted Graph

When we use the subway, we encounter transit stations. Depending on the circumstances, no single transit station is fastest. For example, pregnant women or old people need more transit time, while a young student does not. A method such as Dijkstra's algorithm might offer unavailable shortest-path information because it does not consider that transit time can vary with each passenger.

Therefore, we propose a new graph: the variable weighted graph, which considers the transit time according to the passenger's condition. The variable weighted graph is an extension of a weighted graph.

In this section, we explain the variable weighted graph and our method that obtains the available shortest-path information using a variable weighted graph.

3.1 The variable weighted graph

Suppose we have a weighted graph and the transit time information is given (Table 1(a)). In the table, a dash (–) indicates that there is no railroad between the two stations, while the number indicates the basic transit time. Since A1→A2 and A2→A1 generally differ, the numbers in the two cells differ.

The left side in Table 1(b) represents the basic transit times from node K to nodes L, M, and N. The right side in Table 1(b) represents the transit times according to the passenger's condition. In this case, it indicates the passenger's condition is good and the transit time is halved. In the reverse case, the transit time is doubled.

Now, suppose that the path between the start node A and node K has been decided, and the passenger's condition is good. Then, nodes L, M, and N become candidates to be included in the shortest path as it is extended. At this moment, using the information on the passenger's condition, the transfer times for K→L, K→M, and K→N are changed (Table 1(b)).

Table 1 An example of transfer time

	A1	A2	…	ZZ
A1	-	2	…	5
A2	3	-	…	-
…	…	…	…	…
ZZZ	2	-	…	-

(a) Transfer time information

	Before Change	After Change
Node L	6	3
Node M	10	5
Node N	0	0

(b) User condition and transfer time

As a result, the variable weighted graph in Fig. 3 is built. The weights of edges (L, K), (M, K), and (N, K) are changed. For example, in the new variable weighted graph, the weight of edge (L, K) becomes 4 since the original weight is 1 and the transit time is 3. In this manner, the variable weighted graph continues to be changed and regenerated depending on the execution of the algorithm.

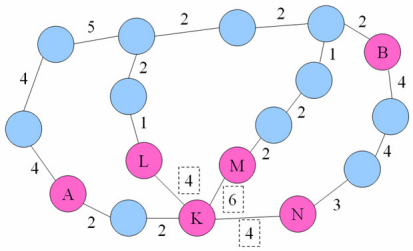


Fig. 3 An example of a variable weighted graph

3.2 Our algorithm searches information using the variable weighted graph

First, the user enters the departure site and destination, and the user's condition information (good, average, or bad). The basic concept behind our algorithm is the same as that of Dijkstra's algorithm without considering passenger condition information. That is, Dijkstra's algorithm repeatedly decides the nodes that are included in the shortest path from among the candidate nodes.

In our algorithm, if a transit station is not included in the candidate nodes, Dijkstra's algorithm is applied on the existing variable weighted graph. If a transit station is included, the information on transit time is renewed using the passenger's condition. Consequently, the variable weighted graph is renewed, and the shortest path keeps being extended by applying Dijkstra's algorithm to the renewed variable weighted graph.

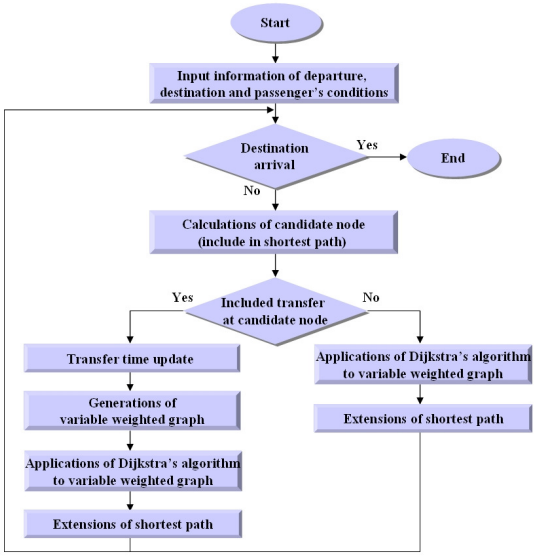


Fig. 4 Our algorithm using the variable weighted graph

3.3 The complexity of our algorithm

The time complexity of Dijkstra's algorithm on a weighted graph of n nodes and e edges is $O(n^2)$. Our algorithm checks whether transit nodes are included in the candidate nodes at each repetition of Dijkstra's algorithm. If they are included, only the information on transit time is added to generate a variable weighted graph. To solve this problem, we need a few extra steps. Therefore, the time complexity of our algorithm is also $O(n^2)$, which is the same as that of Dijkstra's algorithm in Order notation.

4 Implementation of the Information Searching System

We implemented our algorithm search system using a variable weighted graph to obtain the shortest-path information. To achieve this, our system consists of five modules: user-matching DNT (Datum Need Time information for each subway path) extraction modules can selectively extract user-matching DNT, including the departure point and destination given by users from the standard DNT, which has already been stored; a module to analyze user input data that analyzes individual facts (sex, age, baggage, disability, or health index) entered by users; the RNT (Real Need Time information for each subway path) generation module, which reflects the user's features and changes some of the items in the user matching DNT according to the peculiarities of individual users, while also generating the RNT that reflects user's peculiarities; a module for selecting the subway path based on the shortest time that deciphers the RNT, which selects the subway path taking the shortest time from among all the subway paths included in the RNT; and a module that provides the optimal information on the subway paths and selects a subway path based on the shortest time for the user.

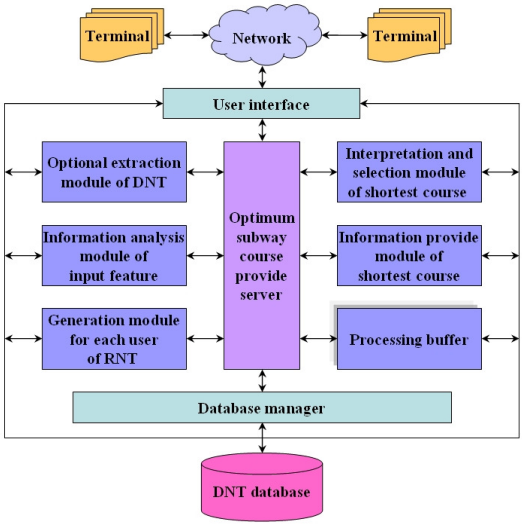


Fig. 5 System overview

5 Conclusions

In this paper, we proposed a more efficient shortest-path search method that does not reconstruct the shortest path from scratch. In addition, as in the case of the shortest subway path, the transit process might have a great influence on the shortest-path search. Existing methods such as Dijkstra's algorithm do not offer the user suitable shortest-path information because they do not consider the condition of the transit passenger.

And, we presented a variable weighted graph that is constructed from information concerning the transit passenger's condition. In addition, we propose a method that searches the shortest-path information in the light of the transit passenger's conditions using a variable weighted graph. We also describe our retrieval system, which provides the shortest-path information in light of the transit passenger's condition.

References:

[1] Jungho Park, Yoonyoung Park, and Sunghee Choi, Distributed Algorithms Solving the Updating Problems, *Korea Journal of Computational and Applied Mathematics*, Vol. 9, No. 2, 2002, pp. 437–450.

[2] Christopher J. Van Wyk, *Algorithms in C++*, Addison Wesley, 2001.