IMPROVED PUBLIC TRANSIT ROUTING ALGORITHM FOR FINDING THE SHORTEST K-PATH

Inwoo Jeon, Hyunwoo Nam, Chulmin Jun*

Dept. of Geoinformatics, University of Seoul, Korea, – {yugo123, hwnam, *cmjun}@uos.ac.kr

KEY WORDS: Public transit routing, Timetable, Transfer penalty, Multiple paths, Similar paths

ABSTRACT:

Most of the existing public transit routing algorithms were developed on the basis of graph theory. Recently, algorithms are being developed that can compute for O-D public transit paths by using timetable information only, not using network structure consisting of nodes and links. The timetable-based public transit routing algorithm produces one shortest path to destination, using departure time and arrival time by stop. But it has limitations in reflecting additional factors, such as transfer penalty and alternative path selection, in the process of path calculation. In addition, since public transit passengers tend to choose one among various alternative paths, it is necessary to calculate multiple paths rather than a single path as in the existing methods. Therefore, this study proposes an improved RAPTOR algorithm that can consider transfer penalty and produce multiple paths, while it is based on RAPTOR, the existing timetable-based public transit routing algorithm. The transfer penalty was applied at the point of transfer, and differently according to transfer types. As a result of analyzing computed paths of the algorithms before and after improvement, it was found that computed paths with the improved RAPTOR algorithm proposed by this study were more similar to Seoul public transit passengers' actual travel paths than computed paths by the existing RAPTOR alone.

1. INTRODUCTION

Generally, shortest public transit routing algorithms find paths by using graph theory (Pyrga et al., 2008; Cionini et al., 2014). And as for their data structure, they use networks consisting of nodes, which represent stops, and links, which represent connection between nodes, and each link stores cost such as travel time, travel distance, etc. In addition, transfer penalty that influences path selection is also used as the cost of a link (Kim et al., 2017).

Recently, algorithms using public transit timetables are being developed, as well as graph theory-based algorithms. The timetable-based algorithms employ their own data formats that store timetables by route, not the existing data structure of node-link networks, and have the advantage that they have faster computation speed than the graph theory-based routing system (Witt, 2015; Strasser and Wagner, 2014).

But most timetable-based algorithms use in their routing only departure time and arrival time at stop, which are based on operation plans. Accordingly, they judge a path of minimum physical travel time as the fastest arrival path, regardless of how many transfers it requires. However, since public transit passengers tend not to prefer excessive transfers, it is needed to find the shortest path that takes transfer penalty into account (Park et al., 2001; Arbex and da Cunha, 2015). In addition, in the case of public transit networks that have various transportation modes, it is also necessary to search for multiple alternative paths that provide individuals with a wider range of path choices (Kim and Kim, 2009).

Hence, this study proposed a timetable-based public transit multiple routing algorithm. This algorithm is an improvement of RAPTOR, a timetable-based public transit routing algorithm, and modified RAPTOR's algorithm so that transfer penalty might be considered in the process of routing. It was so arranged that different transfer penalties might be assigned according to transfer transport types, which were classified into transfer between buses, transfer between bus and subway, and transfer between subway lines. Additionally, walking correction factor was applied to adjust transfer time during walking transfer, which allows preference for walking travel to be reflected in path calculation. Also, improvement was made so that K multiple paths without repetition from origin to destination could be computed, and in this process, similar path determination rules were applied.

To check what effects the additional functions of transfer penalty, walking correction factor, and multiple path calculation had on the results of routing, the results of path calculation by the existing RAPTOR algorithm were compared with those by the improved algorithm. Experiments were carried out to analyze the similarity between actual passenger travel paths extracted from smart card data of Seoul and the computation paths of both algorithms. It was judged that actual passengers' travel paths reflected several factors such as total travel time, travel distance, transfer penalty, preference for walking travel, and so on, and the experiments confirmed that the improved algorithm can reflect passengers' diverse path selection criteria, compared with the existing RAPTOR algorithm.

2. RELATED RESEARCH

As research related to this study, there are studies on timetable-based public transit routing, studies related to multiple-routing, and studies related to transfer penalty. First, as typical timetable-based public transit routing algorithms being recently developed, RAPTOR (Round-bAsed Public Transit Optimized Router), CSA (Connection Scanning Algorithm), and Trip-based (Delling et al., 2015; Dibbelt et al., 2013; Madkour et al., 2017) can be cited. Delling et al. (2012) proposed the RAPTOR

algorithm, in which they stored route's vehicle arrival time at each stop, searched for vehicles passing by each stop, and computed the minimum arrival time and path to the end arrival stop. According to the results of their research, the timetable-based public transit routing algorithm showed faster search time than the graph theory-based algorithm. Dibbelt et al. (2013) proposed CSA, and computed paths by searching connections that represent operation information in the form of one-dimensional array. In their study, the connection modes a one-dimensional array data structure in which the arrival time of all transports arriving at stop is represented on a one-dimensional array, and their findings showed that their algorithm has the advantage of routing speed over the graph theory-based routing algorithm, like the results of research on RAPTOR. Like this, the timetable-based search system has lower computation complexity than the existing graph theory-based search system. And thus it can find paths within a very short time, and has the feature of enabling dynamic search based on departure time.

Most studies related to the public transport multiple path search system use graph theory-based algorithms to search for multiple paths (Wang et al., 2016). Guo and Jia (2017) proposed a multiple path search system, which considers timetables, by using a graph theory-based algorithm to provide departure time to arcs in the form of links instead of nodes. The multiple path search system suggested by their study stores arrival time at node in the order of arrival by using arcs, which is repeated up to the end node. In the process of graph theory-based path search, Hu and Chiu (2015) determined a value of channel similarity to allow a certain degree of overlap among multiple paths and proposed a multiple path calculation algorithm that uses it. Their findings show that if a proper value of path similarity is applied, the difference in travel time between the shortest path and the K^{th} path is not great and no abnormal path is produced.

The last is the analysis of recent research on transfer penalty. Transfer penalty is a concept comprising temporal elements, such as transfer wait time and transfer walking travel time, and non-temporal elements, such as transfer convenience and complexity that are psychological burdens for transfer. The non-temporal elements are converted into time, which is applied to the algorithm. Yoo (2015) used smart card data to estimate the value of transfer penalty occurring when using public transit in Seoul. The study presented an average transfer penalty value of 11.24 minutes and stated that the transfer penalty value varies according to origins and destinations. Yoo (2017) considered planar distance, the number of steps, transfer time, and the existence of escalator as elements of transfer penalty, and analyzed transfer convenience by converting vertical travel distance into planar distance. The results of the study suggested a method for improvement that considers vertical travel distance as well as planar distance, for transfer penalty is also generated by the vertical travel distance. Yang (2017) surveyed passengers who used a certain section of Seoul subway lines, and computed the value of transfer penalty occurring in case of one-time transfer. The transfer penalty value between subway lines was estimated to be about 5.35 minutes per transfer, and thus the study suggested that the transfer penalty value varies with the number of transfers. Garcia-Martinez et al. (2018) analyzed with commuters the correlation between transfer penalty and the elements of walking travel time, wait time, and so on. Their study suggested that transfer penalty is influenced by temporal elements, such as transfer wait time and transfer walking time, and non-temporal elements, such as transfer to other transport modes and complexity. Like this, transfer penalty is also applied in case of transfer to the same transport modes, and higher penalty is applied in case of transfer to other transport modes (Park et al., 2012). From this, it was confirmed that the application of transfer penalty acts as an important factor in routing, and that different values should be assigned according to types of transfer transport.

Generally public transit passengers select their paths among alternative paths according to their preference (minimum transfer, minimum walking time, etc.) as well as minimum arrival time, and thus multiple routing is required. Therefore, this study intends to propose a timetable-based algorithm that allows multiple routing. In the process, the concept of path similarity, which was used in previous studies related to multiple routing, is applied with its modification to fit the timetable-based algorithm. In the case of transfer penalty, its temporal elements such as transfer wait time and transfer walking time were already reflected in the process of timetable-based routing, and thus this paper proposes a method that only the non-temporal element of psychological reactance is considered as transfer penalty, and is converted into time for path finding

Variable	Description
Tr_c	transfer penalty using transfer type c
$l(p, p_i)$	Walking time from stop $p \in S$ to stop $p_i \in S$
J^m_{p,p_i}	Path from stop $p \in S$ to stop $p_i \in S$ in m rounds
$J^{m,k}(p)$	k-th earliest path at stop $p \in S$ in m rounds
$J^m(p)$	Set of paths at stop $p \in S$ in m rounds
$t_{r,v}$	Driving information of vehicle v for the route $r \in R$
$\tau^m_{dep}(p)$	Minimum arrival time at stop $p \in S$ in m rounds
$\tau^m_{arr}(t, p)$	Arrival time from stop $p \in S$ using $t_{r,v} \in T$ in m rounds

Table 1. Description of major variables in RAPTOR algorithm

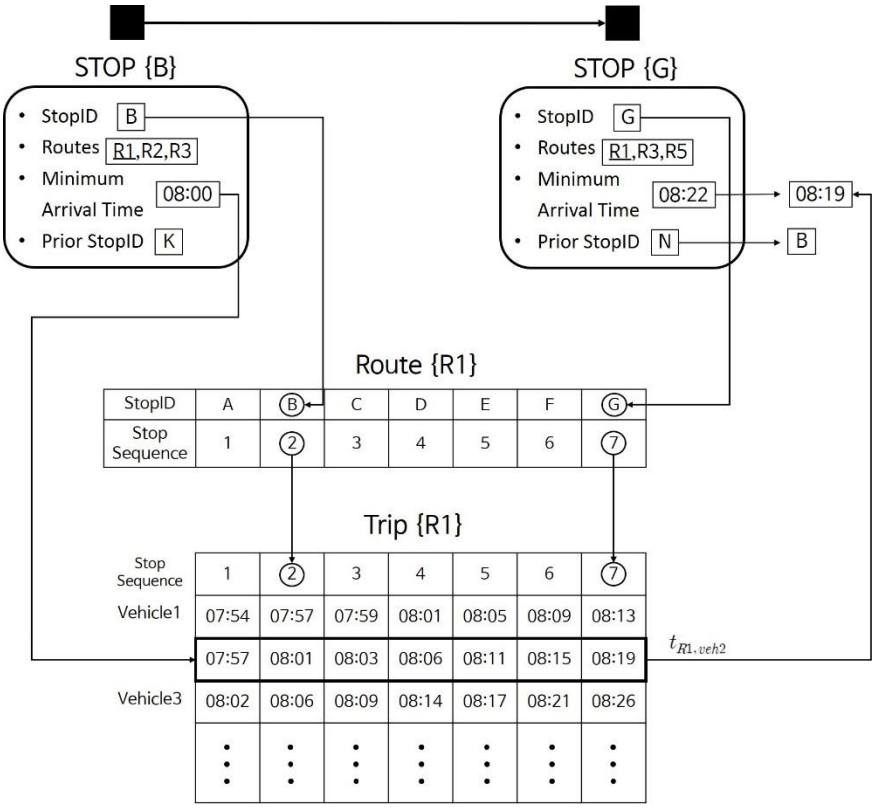

Figure 1. Example of data structure in RAPTOR algorithm

3. IMPROVED RAPTOR ALGORITHM

3.1 Existing RAPTOR algorithm

The algorithm proposed by this study is an improvement of RAPTOR, and so the existing RAPTOR is briefly explained first. Major variables used are summarized in Table 1, and the relationships between the variables and data structure used in the algorithm are shown in Figure 1. Basically, RAPTOR conducts routing by round. If a certain passenger travels from origin to destination using n transfers (or walking travels), one round refers to a stage from a transfer (origin in the case of 0th transfer) to before the next transfer (or arrival) occurs. That is, if n transfers occurred, rounds are composed of {0, 1, 2, • • • , n}. In m Round, the minimum arrival time at all stops that allow travel to from mth transfer place to m+1th transfer place (or destination) is updated.

The data structure used for the algorithm consists of several objects. They include Stop *S*, which stores stop information and $\tau_{dep}^{m}(p)$, minimum arrival time of stop p in m Round; Route *R*, which stores route information; Trip *T*, which stores driving information of vehicle for route; and *F*, which stores walking distance and time between stops. One Route consists of the IDs of stops on a route and stop sequence. One Trip refers to information on the one-time driving of a vehicle for a route from departure garage to arrival garage, and stores arrival times at stops of sequence 1 to final sequence.

Let's explain the method for searching for $t_{r,v}$ by the use of Figure 1. In Figure 1, assuming that a certain passenger arrives at Stop 'B' at 8:00, the passenger transfers to one of routes R1, R2, and R3 available at the stop. The earliest vehicle available after 8:00 among vehicles of each route is selected (in the example of Figure 1, '8:01' the time of arrival at Stop B for $t_{R1,veh2}$, the second vehicle of R1). Then, the minimum arrival time at stops passed by $t_{R1,veh2}$ is updated. For example, in case of getting off at Stop 'G,' using $t_{R1,veh2}$, if the existing minimum arrival time is 8:22, the minimum arrival time is updated to 8:19, which is the arrival time by R1. The update of minimum arrival time refers to change of $\tau_{dep}^{m}(p)$ in case that $\tau_{arr}^{m}(t, p_j)$ is earlier than $\tau_{dep}^{m}(p)$ of a stop. If $\tau_{dep}^{m}(p)$ of a stop is updated, the relevant stop is marked. All stops marked are used to search for $t_{r,v}$ or a stop reached by walking travel at the next round.

As for input values of RAPTOR algorithm, departure stop, arrival stop, and departure time are inputted. Prior to routing, the minimum arrival time of all stops is initialized to ∞, and as for the minimum arrival time of departure stop, departure time is inputted. The algorithm is run largely in three steps and starts from Round 0.

STEP 1. If not Round 0, store the minimum arrival time up to (m-1) Round in the minimum arrival time of all stops in m Round.

STEP 2. Travel all routes once from stop marked in (m-1) Round. Search for $t_{r,v}$ of a route that can be traveled after $\tau_{dep}^{m}(p)$ of the stop, and travel to all stops (p_i) passed by it. Compare $\tau_{arr}^{m}(t, p)$ and $\tau_{dep}^{m}(p_i)$ of p_i and if $\tau_{arr}^{m}(t, p)$ is earlier, update $\tau_{dep}^{m}(p_i)$ of p_i and mark the relevant stop.

STEP 3. Travel to a stops p_i that can be reached on foot from the stop marked in Step 2. Compare the time of $\tau_{dep}^{m}(p)$ plus $l(p, p_i)$ for the marked stop and $\tau_{dep}^{m}(p_i)$ of p_i, and

if the time of arrival after waking travel is earlier, update $\tau_{dep}^{m}(p_i)$ of p_i and mark the relevant stop.

STEP 4. Check whether there is any stop $\tau_{dep}^{m}(p)$ of which was updated. If there is any stop whose minimum arrival time was updated, increase m by 1 and repeat Steps 1-3. If there is no stop whose minimum arrival time was updated, the algorithm is terminated.

The RAPTOR algorithm stores only $\tau_{dep}^{m}(p)$ in arrival stop, and thus it is not possible to know a path to the arrival stop. Therefore, when $\tau_{dep}^{m}(p)$ is updated, information on a stop where $t_{r,v}$ was started to be used is stored together. The previous stop is searched by using stop information stored at the arrival stop, and this is repeated up to departure stop to determine a route. For example, in the example of Figure 1, Stop 'B,' which updated $\tau_{dep}^{m}(p)$ of Stop 'G,' is searched for, and the process of finding Stop 'K,' which updated $\tau_{dep}^{m}(p)$ of Stop 'B,' is repeated until the departure stop is found.

3.2 Improvement

3.2.1 Searching for multiple paths excluding similar paths

The existing RAPTOR algorithm searches for a route of the earliest arrival from origin to destination by tracing back from arrival stop. The algorithm proposed by this study stores *K* paths that arrive at all stops, according to the sequence of arrival time. At this time, similar paths were kept from being stored repeatedly in the *K* paths. The similar paths include cases where the same route sequence is used or the same route is travelled again (excluding subway). Two rules were applied lest similar paths be computed.

The first rule is that $t_{r,v}$ of a specific route should not be searched for at stop at which a passenger got off from the same route. (a) in Figure 2> shows a case where a passenger takes R_1 at departure stop and travels to arrival stop. The path of taking R_1 and traveling to arrival stop directly (upper path in the figure) and the path of taking R_1, getting off at middle stop 'A,' and then taking R_1 again at the same stop (lower path in the figure) differ in arrival time at arrival stop, and thus they are stored as different paths. These paths, however, may be considered the substantially same path, and thus the passenger was prevented from taking the same route, which was used before, again at the middle stop, to prevent the path from being stored in duplicate.

The second rule is that J_{p_i,p_i}^{m}, which have the same boarding sequence at a certain stop, should be stored only once in $J^{m}(p)$ of the stop. If a passenger arrives at a certain stop later, using J_{p_i,p_i}^{m} having the same boarding sequence, comparison should be made with $\tau_{dep}^{m}(p)$ of J_{p_i,p_i}^{m} stored in $J^{m}(p)$, and only the minimum arrival time should be updated. (b) in Figure 2> shows the situation of transfer from different transfer stops A and B, in case of transferring from route R_1 to route R_2. The required travel time varies according to where transfer takes place, stop A or stop B, and thus they are stored as different paths in J_{p_i,p_i}^{m}. In this study, however, the two paths were judged to be similar paths because the sequence of routes taken is the same. Therefore, paths of route sequence $R_1 \rightarrow R_2$ are stored in $J^{m}(p)$, and the minimum arrival time of $R_1 \rightarrow R_2$ is updated to the earlier $\tau_{dep}^{m}(p)$ of the two paths.

3.2.2 Application of transfer penalty and the adjustment of walking travel time

In routing, RAPTOR finds $t_{r,v}$ that a passenger can get on after arrival time at each stop, and then the passenger gets on the relevant vehicle and gets off at other stop, and arrival time at the other stop is updated. Therefore, if arrival time at each stop changes, a route and a vehicle to be taken may be changed, and RAPTOR is characterized by the dynamic change of computed path accordingly. In addition to these characteristics, this study made possible dynamic path selection that reflects transfer penalty at the time of transfer. In case of transfer at stop, $t_{r,v}$ available after waiting as much as transfer penalty time is searched for. For example, if the minimum arrival time of a stop is 15:03 and the transfer penalty value is 3 minutes, $t_{r,v}$ available after 15:06 is searched. Further, as for transfer penalty, horizontal travel and vertical travel up to transfer transport should be considered in addition to transfer frequency. Transfer from bus to bus mostly entails horizontal travel, but transfer from bus to subway or transfer between subway lines is accompanied with vertical travel by the staircase. Thus, it is assumed that different transfer penalties will be applied according to transfer types. Therefore, the transfer types were classified into transfer between buses, transfer between bus and subway, and transfer between subway lines, and different transfer penalty values were applied according to transfer types. To implement this, the transport type of previous $t_{r,v}$ arriving at transfer stop was stored, and then different transfer penalty values were applied according to transports of $t_{r,v}$ to be taken next.

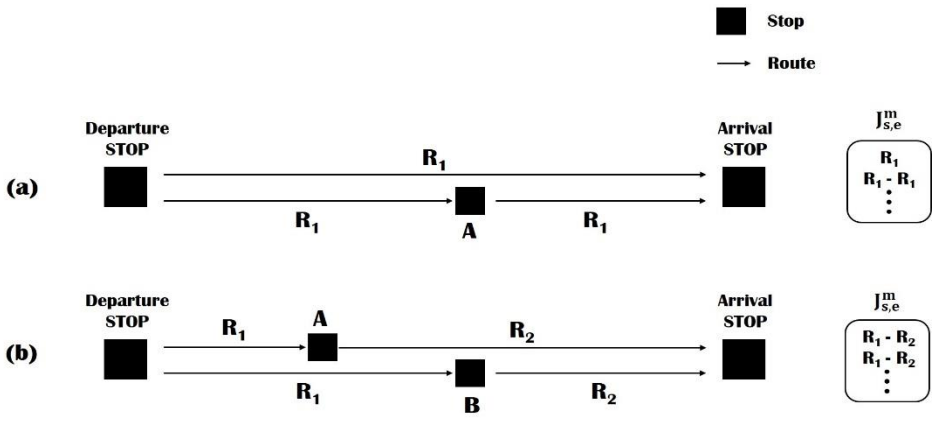

Figure 2. An example of search for similar paths

Next, the following is explanation about the application of walking correction factor. Walking speed is about 1.2 m/s based on adult gait. This study assumed that if walking travel between stops is possible, walking travel is carried out along a straight line between the stops. Thus, required walking time is computed with travel distance and pace, and arrival time at the stop reached by walking travel is updated. But since the travel distance is computed as straight-line distance, it is shorter than actual walking travel distance, and delay at crosswalk may also occur depending on walking situation. Accordingly, if pace is set to 1.2 m/s, required travel time becomes shorter than actual situation, and thus it is necessary to adjust walking travel time. Hence, this study uses the walking correction factor to make the transfer walking travel time in the algorithm similar to actual transfer walking travel time. Conceptually it is equivalent to walking travel speed, but is set to be slower than actual walking speed. For this, it is assumed that the actual public transit passenger moves from departure place to arrival place along Manhattan distance, not straight-line distance. If straight-line distance is 1, then Manhattan distance is $\sqrt{2}$. Thus, as a result of calculating speed according to the distance ratio, 0.83 m/s , a value obtained from dividing 1.2 m/s by $\sqrt{2}$, is used as the walking correction factor.

2.1.1 Routing process using the improved RAPTOR algorithm

The overall routing process of the improve RAPTOR algorithm is as follows: Prior to starting search, the minimum arrival time at departure stop is set to departure time, and the minimum arrival time at other stops is initialized to ∞. In addition, departure stop is marked, and search is started from round 0.

STEP 1. If not round 0, store $\tau_{arr}^{m-1}(t,p)$ and $J^{m-1}(p)$ in $\tau_{arr}^{m}(t,p)$ and $J^{m}(p)$ of each stop, respectively.

STEP 2. Search l_{p,p_i}^{m} and $\tau_{dep}^{m}(p)$ of stops (p_i) that can be reached on foot from stop marked in (m-1) round. If the $\tau_{arr}^{m}(t,p)$ of l_{p,p_i}^{m} is earlier than $\tau_{dep}^{m}(p_i)$, update $\tau_{dep}^{m}(p_i)$ of the stop and mark the stop. If $\mathrm{n}\left(J^{m}(p_i)\right) < K$, add l_{p,p_i}^{m} to $J^{m}(p_i)$. If $\mathrm{n}\left(J^{m}(p_i)\right) = K$, replace $J^{m,K}(p_i)$ with l_{p,p_i}^{m}. If there is change in $J^{m}(p_i)$, sort $J^{m}(p_i)$ in the order of earliest arrival time.

STEP 3. Store in *Q* all routes *R* available at stop *S* marked in (m-1) round, in the form of *(R,S)*.

STEP 4. Search $t_{r,v}$ of *R* at *S* of all *(R,S)* stored in *Q*. At this time, search for $t_{r,v}$ after waiting for Tr_c from $\tau_{dep}^{m}(p)$ of stop.

STEP 5. If a route taken last at $J^{m}(p)$ of *S* is identical to *R*, skip STEPs 6 and 7.

STEP 6. Find $t_{r,v}$ of *R*, and alight at all stops (p_i) along the route, using it. If $J^{m}(p_i)$ has no route having the route taking sequence of l_{p,p_i}^{m} at the alighting stop, proceed with the following process. Otherwise, proceed with STEP 7. If $\mathrm{n}\left(J^{m}(p_i)\right) < K$, add l_{p,p_i}^{m} to $J^{m}(p_i)$. If $\mathrm{n}\left(J^{m}(p_i)\right) = K$, compare $\tau_{dep}^{m}(p_i)$ of $J^{m,K}(p_i)$ with $\tau_{arr}^{m}(t,p)$ of l_{p,p_i}^{m}, and if update is possible, replace $J^{m,k}(p_i)$ with l_{p,p_i}^{m}. If there is change in $J^{m}(p_i)$, sort $J^{m}(p_i)$ in the order of early arrival time.

STEP 7. Find a route having the same route sequence as l_{p,p_i}^{m} among $J^{m}(p)$ of alighting stop, and update $\tau_{dep}^{m}(p)$ y comparing $\tau_{dep}^{m}(p)$ of the relevant path and $\tau_{arr}^{m}(t,p)$ of l_{p,p_i}^{m}. If update occurs, sort $J^{m}(p)$ in the order of early arrival time.

STEP 8. If there is any stop whose $\tau_{dep}^{m}(p)$ is updated, repeat from STEP 1 by increasing round. If there is no stop whose $\tau_{dep}^{m}(p)$ is updated, terminate the search.

Algorithm 1: improved RAPTOR algorithm

Input : Start and target stop p_s, p_t and departure time τ

Output : earliest *K* path at stop p_t

```
m ← 0
τ_dep^m(p) ← ∞
τ_dep^m(p_s) ← τ
mark p_s

while Count(marked stop) = 0
 if m > 0 then
  τ_dep^m(p) ← τ_dep^{m-1}(p)
  J^m(p) ← J^{m-1}(p)

 for each marked stop p do
  for each route r serving p do
    Add (r, p) to Q
  unmark p

 for each marked stop p do
  for each stop p_i walkable from p do
    if exclude similar route = 'TRUE' then
      if τ_arr^m(t,p) < τ_dep^m(J^{m,k}(p_i)) then
        J^{m,k}(p_i) ← τ_arr^m(t,p)
        mark p_i
        sort J^m(p_i)
    else
      if n(J^m(p_i)) < K then
        J^m(p_i) ← l_{p,p_i}^m
        sort J^m(p_i)
      else
       if τ_dep^m(p) + l(p,p_i) < τ_dep^m(J^{m,K}(p_i)) then
        J^{m,K}(p_i) ← τ_dep^m(p) + l(p,p_i)
        mark p_i
        sort J^m(p_i)
```

for each (r, p) in Q **do**
  $t_{r,v} \leftarrow$ trip after $\tau^m_{dep}(p) + Tr_c$
  for each stop p_i passing $t_{r,v}$ **do**
    if exclude similar route = 'TRUE' **then**
      if $\tau^m_{arr}(t,p) < \tau^m_{dep}(J^{m,k}(p_i))$ **then**
        $J^{m,k}(p_i) \leftarrow \tau^m_{arr}(t,p)$
        mark p_i
        sort $J^m(p_i)$
    else
      if $\mathrm{n}(J^m(p_i)) < K$ then
        $J^m(p_i) \leftarrow J^m_{p,p_i}$
        sort $J^m(p_i)$
      else
        if $\tau^m_{arr}(t,p) < \tau^m_{dep}(J^{m,K}(p_i))$ **then**
          $J^{m,K}(p_i) \leftarrow \tau^m_{arr}(t,p)$
          mark p_i
          sort $J^m(p_i)$
endwhile

Algorithm 2: exclude similar route

Input : $J^m(p_i)$, $t_{r,v}$, J^m_{p,p_i}
Output : TRUE or FALSE

for each $J^{m,k}(p_i)$ in $J^m(p_i)$ **do**
  if last $t_{r,v}$ of $J^{m,k}(p_i) = t_{r,v}$ **then**
    return 'FALSE'
    break
  else if $J^m_{p,p_i} = J^{m,k}(p_i)$ **then**
    return 'FALSE'
    break
  else
    return 'TRUE'

4. EXPERIMENT

This chapter will confirm the results of whether to apply transfer penalty to the existing RAPTOR algorithm. For this, paths computed before and after the application of transfer penalty were compared with actual passengers' travel paths. In addition, changes in paths computed according to transfer penalty values classified by transfer type were compared.

4.1 Region of experiments of experimental data

The region of experiments was Seoul, and its 13,419 public transit stops (659 subway stations, 12,760 bus stops), 44,880 routes, and 51,268 trips were used. As for experimental data, smart card records (52,075,973 cases) , which were provided by TOPIS, of passengers who used public transit from October 12, 2017 to October 19, 2017 were used, and timetables drawn up by using bus operation information as of October 19, 2017 were used. In addition, the range of walking transfer was restricted to within a straight-line distance of 700 m from each stop, and repeated walking transfer was allowed. As for actual public transit passengers' travel paths, paths used by 40 passengers or more for a week were selected among passenger travel paths recorded in smart cards. Among the selected paths, 500 paths of 0 transfer frequency and 500 paths of one-time transfer were randomly chosen, and consequently, 1,000 sample paths were selected. The path includes information on boarding stop, boarding time, transfer stop, transfer time, arrival stop, and arrival time.

4.2 Assumptions

Similarity modes the ratio of sample paths, among 1,000 sample paths, that coincide with paths computed by the algorithm. When checking the coincidence between sample paths and algorithm-computed paths, bus travel and subway travel were distinguished. In the case of bus travel, although a path computed by the algorithm and a sample path had different alighting stops and boarding stops, it was assumed that the paths coincided if they used the same route. In the case of subway travel, a tag record at the turnstile of boarding station and a tag record at alighting station are stored in smart card. Thus, it is not possible to identify any transfer station between boarding station and alighting station. Therefore, it was assumed that a sample path and a path computed by algorithm coincided if passenger got on at the same station and got off at the same station.

Among 1,000 sample path information, the location of departure and arrival stops and the boarding time at departure stops were used as the input of algorithm. The boarding time modes the tag time of actual public transit passenger's boarding and alighting in the case of bus, and the tag time at turnstile in the case of subway. The actual public transit passengers arrived at bus stop or subway station before the boarding time of departure place,

And thus this time is used as the departure time of the algorithm. Timetable information is the information of public transit route operation, and may be different according to dates. And the operation information may become different from schedule due to variables occurring during the actual operation of bus or subway (traffic conditions, accident, etc.). Therefore, this study assumed that although the arrival time of vehicle on schedule may be identical to boarding time, vehicle may arrive five minutes or 10 minutes earlier or later than boarding time. Thus, since fixed time was given to the arrival time of operation schedule applied to the algorithm, the departure time of algorithm was changed on the basis of passenger's boarding time instead. As for departure time, five time ranges (10 minutes before boarding time, five minutes before boarding time, boarding time, five minutes after boarding time, and 10 minutes after boarding time) were determined on the basis of boarding time, and the five departure time was set as the departure time of algorithm, respectively, and then algorithm-computed paths were searched for in relation to the 1,000 path information. Then, when routing according to all the departure time was completed, all the paths were collected, and top K paths were selected in the order of short spent time.

4.3 Result of experiments

4.3.1 Visualization results of improved algorithm

The improved algorithm proposed by this study was applied to the traffic network of Seoul for its test. Figure 3 shows the implementation of the improved algorithm. For interface development, C# was used, and for database, MS SQL was used. As for hardware, the algorithm was run in Intel Xeon E5-2667 2.90GHz and RAM 64GB. As for O-Ds randomly chosen by this algorithm, the results of calculation with K=10 show that three seconds were spent on average for departure stop → arrival stop, and that seven seconds were spent on average for departure stop → all stops.

Figure 3. Visualization of the results of routing by improved RAPTOR algorithm

Yeouido Station was selected as origin, and Lake (Seokchon Lake) as destination, and 9:30 am was inputted in departure time. The walking correction factor was set to 0.83 m/s, and transfer penalty was set to five minutes for all transfer types. The route in the darkest color in Figure 3 shows the path of the shortest arrival time found by the existing RAPTOR algorithm, and the other routes are multiple routes computed by the improved algorithm. In the case of Figure 3, 10 paths were searched, but they were represented as five paths on the map because some paths used the roads or subways of the same route. The result of the visualization of computed paths shows that available paths of O-D are variously computed according to the number of K.

4.3.2 Comparison of similarity according to whether transfer penalty is applied

Figure 4 compares the difference in path similarity according to whether transfer penalty is applied or not. In Figure 4, X-axis represents the number of K, and Y-axis similarity, respectively. In this experiment, the same value was given to transfer penalty irrespective of transfer types. Non-transfer penalty is the result computed with the walking correction factor set to 1.2 *m/s* and the transfer penalty to 0 minute, and transfer penalty shows the results computed with the walking correction factor set to 1.2 *m/s* and the transfer penalty to three minutes. In Figure 4, in case of single path (K=1), while similarity was 48.5% if transfer penalty was not applied, similarity was 58.8%, about 10%p up, if transfer penalty was applied. In case of calculating multiple paths (K=5), while similarity was 80.8% if transfer penalty was not applied, similarity was 91.4% if transfer penalty was applied. Like this, it was found that similarity was higher, irrespective of the number of K, when the improved RAPTOR algorithm was used. This modes that it became possible to search for paths of about 10% (≈ over 100 paths), which were not found by the existing RAPTOR algorithm, when the algorithm that considers transfer penalty was used. From this, it can be assumed that there are cases where during transfer, actual passengers choose paths in consideration of transfer penalty as well as the shortest required time. But it can be seen that there are also passengers' travel paths not searched even by the improved algorithm, given that similarity does not reach 100% even though K is increased to a substantial number (about 10). That is, it is judged that there are some cases where factors other than transfer penalty become criteria for path choice.

Figure 4. Comparison of similarity according to whether transfer penalty is applied

4.3.3 Comparison of the results of path calculation according to transfer penalties by transfer type

To investigate the effects of the walking correction factor and the transfer penalty by transfer type, seven cases were classified in Table 2 and their respective parameter values were set in advance. *b* is the abbreviation of bus and *s* is the abbreviation of subway in Table 2. Situation A refers to a case where transfer penalty is not considered, situation B a case where the same transfer penalty is applied irrespective of transfer types, situation C a case where transfer between buses is not preferred, situation D a case where transfer between subway lines is not preferred, situation E a case where the use of subway is not preferred, situation F a case where any of transfers is not preferred, and situation G a case where walking travel is not preferred, respectively. Table 3 shows the results of path calculation, with two Seoul subway stations (Nambu Bus Terminal Station and Isu Station) as O-D. At this point, departure time was set to 11:00 am, and K to 3, respectively. The total travel time is a value that considers all of onboard travel time, transfer time, walking travel time, and transfer penalty.

Case	CWS (*m/s*)	TP <*s-s*> (*min*)	TP<*b-b*> (*min*)	TP <*b-s*> (*min*)
A	0.8	0	0	0
B	0.8	5	5	5
C	0.8	5	15	5
D	0.8	15	5	5
E	0.8	15	5	15
F	0.8	15	15	15
G	0.3	5	5	5

Table 2. Corrected walking speed (CWS) and transfer penalties (TP) for different cases

To compare situation A and situation B, in the case of situation A, the path that requires two transfers, but spends the total travel time of 25 minutes, as in <B22 → B15 → L4>, is computed as the optimum path. In situation B, transfer penalty is applied and the required time for twice-transfer paths increases unlike situation A, and a single-travel path or a path of one-time transfer like <B461 → B16> is computed. In this process, additional wait time caused by transfer penalty was spent, and the total travel time of path <L3 → L7> increased to 29 minutes.

To compare situation B and situation C, in situation C where transfer between buses is not preferred, the total travel time of a path having transfer between buses, like <B461 → B16>, increased, and the path having transfer between subway and bus like, <L3 → B540>, was searched for as its total travel time was comparatively shortened. In addition, in situation B and in situation C, the total travel time of path having no transfer between buses is the same because it is not influenced by changed transfer

case	calculated path	total journey time	walking time
A	L3 → L7	23m 54s	3m 26s
	B22 → B15 → L4	25m 00s	5m 11s
	B22 → B17 → B06	27m 36s	6m 02s
B	L3 → L7	29m 54s	3m 26s
	B4319	32m 24s	4m 32s
	B461 → B16	32m 48s	5m 47s
C	L3 → L7	29m 54s	3m 26s
	B4319	32m 24s	4m 32s
	L3 → B540	34m 49s	5m 17s
D	B4319	32m 24s	4m 32s
	B461 → B16	32m 48s	5m 47s
	L3 → B540	34m 49s	5m 17s
E	B4319	32m 24s	4m 32s
	B461 → B16	32m 48s	5m 47s
	B461 → B643	40m 30s	8m 07s
F	B4319	32m 24s	4m 32s
	B350	40m 39s	6m 13s
	L3 → L7	41m 54s	3m 26s
G	L3 → L7	38m 50s	8m 46s
	B4319	43m 26s	11m 36s
	B4319 → L7	44m 50s	9m 57s

Table 3. Examples of paths found of an O-D (Nambu Stn. to Isu Stn. at 11:00am, K=3)
(L#: subway line number, B#: bus number)

To compare situation B and situation D, if transfer between subway lines is not preferred, path requiring the change of subway lines, like path <L3 → L7>, increased the total travel time, and failed to be found. In addition, if transfer to subway is not preferred, as in situation E, path having transfer between buses or path of single transport modes was computed. For example, although path <B461 → B643> takes approximately 10 minutes more than paths in situation B, it was found because the value of subway transfer penalty increased.

To compare situation B and situation F, as a result of searching path in situation F where no transfer is preferred, the path using single transport without change in transportation modes was found as the optimum path. At this point, like path <B350>, a path that travels on foot up to arrival stop after moving by bus up to a nearby stop, instead of getting off at the arrival stop could be computed. Path <L3 → L7> spent 29 minutes in situation B, but was searched for as a path that spends about 42 minutes in situation F because transfer penalty increased.

To see paths in situation G, the same paths as in situation B were used, but the path <L3 → L7>, which took 29 minutes in situation B, was computed as a path that takes 39 minutes because walking travel time became longer. In addition, due to the change in walking travel time, a path that was not found in situation B, like <B4319 → L7>, was found.

Overall, the experimental results showed that if different transfer penalties are given according to transfer types, different transports are chosen according to penalty values set in routing, thereby resulting in differences in paths judged to be the optimum path. In addition, it was found that different paths were computed according to the size of transfer penalty value as well as whether to apply transfer penalty.

5. CONCLUSION

This study proposed an algorithm improved by applying transfer penalty and the multiple routing technique to the existing timetable-based RAPTOR algorithm. To calculate multiple paths with no repetition, similar path determination rules were established, and methods for considering transfer penalty in the process of transfer were applied to the algorithm. Different transfer penalties were made to be set according to transfer types, and the walking correction factor was also added to adjust walking travel time. A program was implemented to test the proposed algorithm. And with it, experiments to compare similarity between actual passengers' travel paths and the results of routing were carried out. It was found that the proposed method can calculate about 10%*p* more paths similar to the actual passengers' travel paths than paths computed by the existing RAPTOR algorithm. In addition, it was found that various paths are computed whenever different transfer penalties are given.

This study has the limitation that while it differentiated between vertical travel and horizontal travel when giving transfer penalty and walking travel distance, it failed to apply their accurate and realistic values. If some parameters such as walking travel time for vertical travel is adjusted realistically, it is judged that it will be possible to estimate, by the use of the method proposed by this study, with which preference for transfer and walking travel the actual passenger decided on travel path. This study suggested routing that considers transfer penalty, but it will also need the comprehensive consideration of fare, transfer convenience and so on that affect path selection. In addition, because transfer penalty is influenced by the number of transfer and the physical distance of O-D, it is judged that the modification of the algorithm will be needed in consideration of them.

ACKNOWLEDGEMENTS

This research was supported by a grant(18CTAP-C133228-02) from Technology Advancement Research Program (TARP) funded by Ministry of Land, Infrastructure and Transport of Korean government

REFERENCES

Arbex, R. O. and da Cunha, C. B. 2015. Efficient transit network design and frequencies setting multi-objective optimization by alternating objective genetic algorithm, Transportation Research, 81, pp.355-376.

Cionini, A., D'Angelo, G., D'Emidio, M., Frigioni, D., Giannakopoulou, K., Paraskevopoulos, A. and Zaroliagis, C. 2014. Engineering graph-based models for dynamic timetable information systems, In OASIcs-OpenAccess Series in Informatics, 42, pp.46-61.

Delling, D., Dibbelt, J., Pajor, T. and Werneck, R. F. 2015. Public transit labeling, In International Symposium on Experimental Algorithms. Springer. pp.273-285.

Delling, D., Pajor, T. and Werneck, R. F. 2012. Round-based public transit routing, Proceedings of the Meeting on Algorithm Engineering & Expermiments. Kyoto, Japan. pp.130-140.

Dibbelt, J., Pajor, T., Strasser, B. and Wagner, D. 2013. In International Symposium on Experimental Algorithms. Springer. pp.43-54.

Garcia-Martinez, A., Cascajo, R., Jara-Diaz, S. R., Chowdhury, S. and Monzon, A. 2018. Transfer penalties in multimodal public transport networks, Transportation Research Part A: Policy and Practice.

Guo, J. and Jia, L. 2017. "A new algorithm for finding the K shortest paths in a time-schedule network with constraints on arcs," Journal of Algorithms & Computational Technology, 11(2), pp.170-177.

Hu, X. and Chiu, Y. C. 2015. A Constrained Time-Dependent K Shortest Paths Algorithm Addressing Overlap and Travel Time Deviation, International Journal of Transportation Science and Technology, 4(4), pp.371-394.

Kim, E.C. and Kim, T.H.(2009). K-path Algorithm for a Transfer of the Mobility Handicapped, Seoul Studies, 10(2), pp.147-159.

Kim, E.J., Lee, S.H., Cheon, C.K. and Yu, B.Y. 2017. Development of the Algorithm of a Public Transportation Route Search Considering the Resistance Value of Traffic Safety and Environmental Index, The Korea Institute of Intelligent Transport Systems, 16(1), pp.78-89.

Madkour, A., Aref, W. G., Rehman, F. U., Rahman, M. A. and Basalamah, S. 2017. A survey of shortest-path algorithms. arXiv preprint arXiv:1705.02044.

Park, B.H. and Oh, S.J. 2001. Effects of Transfer Penalty in the Public Transit Assignment, Social Economy & Policy Studies, 17(2), pp.65-92.

Park, H.C., Kim, Y.S., Kang, S.P. and Ko, S.Y. 2012. A Study of Diffenrence Between Intramodal and Intermodal Transfer Penalties, Proceedings of the KOR-KST Conference, 66, pp.555-560.

Pyrga, E., Schulz, F., Wagner, D., and Zaroliagis, C. 2008. Efficient models for timetable information in public transportation systems, Journal of Experimental Algorithmics, 12, pp.1-39

Strasser, B. and Wagner, D. 2014. Connection scan accelerated, In 2014 Proceedings of the Sixteenth Workshop on Algorithm Engineering and Experiments, pp. 125-137.

Wang, S., Yang, Y., Hu, X., Li, J. and Xu, B. 2016. Solving the K-shortest paths problem in timetable-based public transportation systems, Journal of Intelligent Transportation Systems, 20(5), pp.413-427.

Witt, S. 2015. Trip-Based Public Transit Routing. In Algorithms ESA 2015, Berlin, Heidelberg. Springer. pp.1025-1036.

Yang, S.J. 2017. A study on Route Choice Modeling in Rapid Transit Network Considering Transfer Penalty and Angular Cost, Master thesis, korea national university of transportation, Seoul, Korea

Yoo, G. S. 2015. Transfer penalty estimation with transit trips from smartcard data in Seoul, Korea, KSCE Journal of Civil Engineering, 19(4), pp.1108-1116.

Yoo, H. 2017. A Study for Improving the Convenience of Transfer in Urban Railway : Cases of the Airport Railroad, Master thesis, korea national university of transportation, Seoul, Korea

Revised August 2018