FINDING LEAST-TRANSFER PATH IN MULTI-ROUTE / MODAL PUBLIC TRANSIT SYSTEM

by

YUAN GAO

(Under the Direction of Xiaobai Yao)

ABSTRACT

Demand of using public transit has been tremendously increasing during the past decade. As the public transit system become more and more complicated both in volume and dimension, the need for trip planning assistance, especially trips involving transfers, is emerging. Traditional shortest path algorithms are not very suitable for trip planning tasks involving transfers. A new network representation is introduced inspired by the traditional connectivity network, only with multiple layers. Each layer represents one route in the network. An algorithm is designed and implemented to accommodate the network representation with relatively high computational efficiency. A series of matrices are introduced to store the information of least transfer paths between each pair of nodes in the network. A NYC subway case study is conducted to validate the algorithm and examine the efficiency of the algorithm. Potential application of the least transfer path matrices in planning supporting is discussed.

INDEX WORDS: Least Transfer Path, Network Representation, D^{LTP} matrix, R matrix, NL matrix, RL matrix

FINDING LEAST-TRANSFER PATH IN MULTI-ROUTE / MODAL PUBLIC TRANSIT SYSTEM

by

YUAN GAO

BS, Nanjing University, China 2006

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2008

FINDING LEAST-TRANSFER PATH IN MULTI-ROUTE / MODAL PUBLIC TRANSIT SYSTEM

by

YUAN GAO

Major Professor:	Xiaobai Yao
Committee:	Lan Mu Thomas R. Jordan

TABLE OF CONTENTS

BACKGROUND AND INTRODUCTION

To alleviate traffic congestion problem and high expense on gases from using private vehicles, public transportation becomes an important alternative in the high-density urban areas as it provides a relative efficient and affordable way to deal with the enormous amount and complex daily needs to commute in cities. However, due to highly dispersed travel patterns (home, work, entertainment, shopping and etc.) and the continuing urban sprawl (both in size and complexity), it is financially and operationally infeasible to provide adequate direct transit services between all pairs of origins and destinations. A compromised and pragmatic solution is to transfer at certain connection points from one route to another or from one mode of transportation to another to go from their origin to the destination indirectly. Sometimes, to reach a remote destination might involve multiple transfers eventually.

By connecting different routes and modes of public transit network, transfers can greatly expand transportation system and provide more flexibility for passengers to choose by their own preferences. Moreover, transfers also help to establish stratified networks to concentrate passengers on the main routes and then distribute them to secondary routes to improve the operational efficiency and reduce the system wide economic cost. However, transfer also has disadvantages because of its own characteristics.

Knoppers and Muller argues that “A transfer reduces the attractiveness of public transport as an alternative to private cars” [Knoppers and Muller, 1995]. Several attributes contribute to the inconvenience of transfers. Among these are the physical effort and time consumption to alight from one vehicle to aboard the other, extra transfer fees and chances to miss the transfer vehicles due to various reasons. Furthermore, it’s difficult for a newcomer to get familiar with the complex public transportation network and choose the right connection points to optimize their trips. All those factors reduce passengers’ willingness to use public transit network and at a higher level, reduces the efficiency of the whole transportation system.

Generally speaking, the deficiency of transfers comes from two major sources: systematic and non-systematic deficiencies.

1. Systematic Deficiency of transfers

Systematic Deficiency refers to the deficiency that cannot be avoided by public transit users themselves. Those deficiencies can only be removed or reduced by the improvement of the system itself. More specifically, among these are:

1) Non-optimized design and planning of public transit network

The design of public transit network involves connection point set up, route planning, coverage frequency, and etc. Connection points are sometimes not very well designed that they are not at the right place to provide best connectivity between different routes to extend the service coverage, or passengers have to cover a long distance to aboard the connection vehicles which costs lots of physical effort and time, as well. Route planning can be another issue. Shortsighted route planning would greatly draw back the total efficiency of the public transit network. Due to the quick development of urban area and difficulty to reconstruction, the current routes sometimes does not serve the passengers' need well and cost them redundant time and fare to travel. Furthermore, to maintain a affordable operational cost and not to exceed the capability of network infrastructure, agencies have to keep a reasonably lower frequency of services which leads to more waiting time for those passengers.

2) Instability of Services

Public transportation users might constantly notice that the public transportation vehicles do not always follow the time table strictly. Statistically, vehicles' arriving times fall into a range centered at the schedule time and the deviation of which depends on the mode, traffic condition and other various uncontrollable factors such as drivers' experience, total flux of passengers, vehicle malfunction, etc. All those random or non-random factors contribute to instability of public transportation services in terms of punctuality which is supposed to be one of the most important characteristics of public transportation. The direct result of instability is longer waiting time for passengers. On the system-wide level, it hurts the time efficiency of the whole network. Research on dynamic timing control of public transportation has been done by in the recent years (I will elaborate in the literature review part), however, the practicability is still open to doubt considering the immaturity of theories, cost, and etc.

2. Non-Systematic Deficiency of transfers

Non-systematic deficiency refers to those inefficiencies caused by the users of public transportation for different reasons instead of the system itself. More specifically, among these are:

1) Lack or Unavailability of Information

Considering the complexity and size of modern public transportation networks, it is nearly impossible for passengers to well understand the whole network. Passengers' motivation to study the network is driven by their needs to use the public transportation system. Their familiarity of the network depends on how long they have been using the network and their capability to acquire information about the network. Furthermore, passengers' knowledge is often restricted by their daily activity territories. Trips involving transfers especially demand higher familiarity of the network. Passengers without enough information and understanding of the network might make non-optimized decisions when dealing with trips to remote sites involving transfers.

2) Lack of Expertise in route Planning

Even under the assumption that passengers have had enough information about the network, they sometimes still cannot make optimized decisions due to lack of expertise in route planning for their trips which might involve high volume of computation and theory foundation.

Lots of work has been done to improve the efficiency of public transportation network in every direction that I mentioned above. Practices and development/redevelopment projected have been conducted based on the research findings such as public transportation infrastructure design, dynamic and intelligent public transportation management. However, less or not enough efforts have been made dedicated to solve the intractable problems arose from transfer-involving trips from public transportation users. This situation motivates me to conduct research in this area by searching for an efficient and reliable way to solve the problem. In the following part of the thesis, I will start with literature review to look back into the history of the development of shortest path analysis algorithm and solutions, and also their merits and demerits for the trip planning involving transfers for public transportation users. Then, I will demonstrate the inspiration of my research, methodology, pilot project results, and significance of my study.

LITERATURE REVIEW

1. Classic and innovative shortest path problems

Before we start to examine the shortest-path problems involving transfers, it's necessary for us to briefly review the history of development of more classical shortest-path problems. Shortest path problems have been broadly studied in various fields especially in transportation science in the past decades. However, practices and analysis have proven that there were still no universal optimal protocols to deal with all kinds of network structures and practice scenarios. Stuart pointed out that for different shortest path algorithms, some network structures are better fit for a certain algorithm among which. And to choose a certain algorithm, users should consider problem structure, hardware configuration and programming language [Stuart 1969]. However, some studies fail to appreciate Stuart's opinion and previous results. They made insufficient statements to judge the previous algorithms based on their specific setting of problems. In the following part, I will briefly review some efficient algorithms for specific setting of problems.

1) One source shortest-path problem

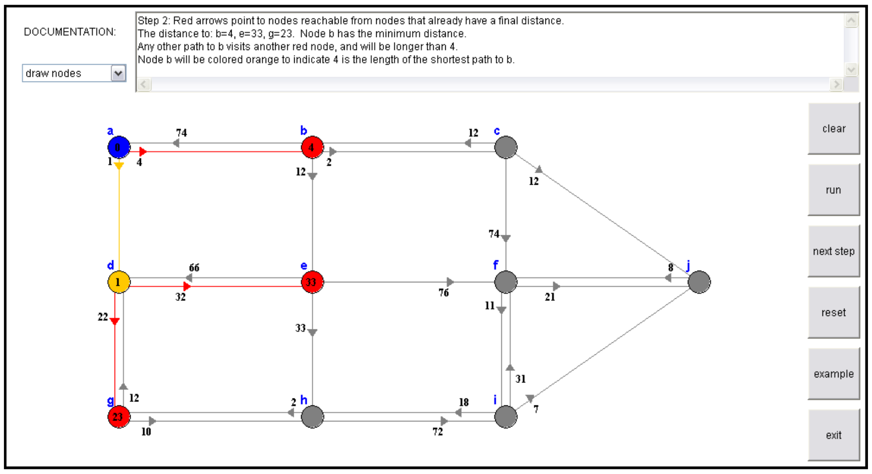
Dijkstra introduced one of the most popular shortest-path algorithms in 1959 [Dijkstra, 1959]. Let's consider a network with N nodes, each assigned a number from 1 to N arbitrarily. A matrix can be constructed so that each element shows the cost to go from one node to the other according to the row and column number. The cost should be equal to or greater than zero. In Dijkstra's algorithm, by applying certain labeling techniques and systematical expansion of the route tree following certain criteria, he was able to construct the final route tree. Please look at the following steps:

I. Mark the starting node with 0 and all the other nodes with infinity.

II. Begin from the starting node and search for all the direct connected nodes and temporarily mark with the cost value in the matrix

III. Choose the smallest temporal mark, since any other path to this node will be greater than the marked cost, (Figure 1), we will make the temporal marked value permanent.

IV. Search for all the direct connected nodes of the permanently marked nodes, mark with the cost value in the matrix plus the permanently marked value. Then, repeat step III until all the nodes are marked permanently.


DOCUMENTATION:

draw nodes

Step 2: Red arrows point to nodes reachable from nodes that already have a final distance.
The distance to: b=4, e=33, g=23. Node b has the minimum distance.
Any other path to b visits another red node, and will be longer than 4.
Node b will be colored orange to indicate 4 is the length of the shortest path to b.

Figure 1 Flow of constructing Route tree in Dijstra's algorithm

(Source: http://www.dgp.toronto.edu/people/JamesStewart/270/9798s/Laffra/DijkstraApplet.html)

By following this protocol, we can find the shortest path from a specific node to all other nodes efficiently, and this algorithm is proofed to be robust as soon as all the cost value is equal or greater than zero. If we only want to find the route from one node to another, we can stop if the destination node is marked permanently in step 4 to increase the efficiency. Another algorithm is Shortest-Path Tree(SPT) which can be viewed as a variation of the Dijkstra algorithm. The deficiency of of SPT is that every time there is any change in the network structure even only minor one. We need to re-proceed the SPT algorithm again to create the tree, however, Paolo Narváez, Kai-Yeung Siu, Hong-Yi Tzeng managed to introduce a rich set of dynamic update procedures to remedy the drawback of the static SPT algorithm[Paolo Narváez, Kai-Yeung Siu, Hong-Yi Tzeng, 2000]. The approximate computational complexity for the alogirithm is $O(2.5N^2)$.

2) Multi-sources shortest-path problem

Base on the algorithms we discussed from the single-source shortest-path problem, we can simply repeat the steps I to IV by choosing different starting node to solve the problem. However, considering the computation cost(approximate computational complexity would be $O(2.5N^3)$, it's usually not the most efficient way to accomplish this task. In 1962, FLOYD first introduced another algorithm to solve the multi-sources shortest- path problems. Later, this algorithm has been improved and generalized into a procedure featuring steps showing below:

I. Start from generalized the network (N nodes) into a N by N matrix showing distance (or cost, time) between each pair of nodes with direct links. (L^1 matrix). If there is no direct link between a certain pair of nodes, mark that cell with infinity or a large-enough number. All the cells can be referred as $L^1_{x,y}$. At the same time, instead of only recording the total distance, we will also create a R matrix to record all those intermediate points. To initiate the R matrix, we mark all those pairs with direct link by the index of the destination node. Leave all the other cells blank.

II. Then we can start the iteration process:

$$L^k_{x,y} = \min\left(L^{k-1}_{x,y}, L^{k-1}_{x,i} + L^{k-1}_{i,y}\right) \qquad \text{IF } \left(L^{k-1}_{x,i} + L^{k-1}_{i,y} < L^{k-1}_{x,y}\right),\ R_{x,y} = i$$

Here, k initiate from 2, increased by 1 for each iteration, and terminate at N. i=1, 2, 3, 4, … N.

After the two steps, we will have two matrices: L^N, which shows the shortest distance between each pair of nodes, and R matrix, a matrix showing all the intermediate nodes of the shortest-path. The approximate computational complexity for this algorithm is $O(N^3)$.

3) Shortest-path passing specific nodes problem

We can define this problem in the following way: Given a network with N nodes, we want to find the shortest path to go from node A to node B visiting a set of k nodes in the path. One of the most classical problem in this class would be the travelling salesman problem which is related to early works and discussions from William Rowan Hamilton and Thomas Kirkman in 17th century, and formally posed by Hassler Whitney in a talk at Princeton University in 1930th. This problem is a sub-set of the shortest-path-passing-specific-nodes-problem class. The most significant characteristic of this set of problems is that they feature a circular path, the starting and destination nodes are the same.

The shortest-path passing specific node problems are classified as NP-hard (nondeterministic polynomial-time hard) in computational complexity theory. The simplest way to solve this problem would be trying all the permutations of the k intermediate nodes to find the shortest-path. The numbers of permutation would be: $P(N,k) = \frac{N!}{(N-k)!}$, not including the computation we needed to find the shortest-path between each pairs of nodes in the k nodes set which would be $C(k,2) \times 2.5N^2 = \frac{2.5k!N^2}{2!(k-2!)}$. According to the formula, we can find that as the N or k increase, the computation needed to solve this problem will increase tremendously. S. E. Dreyfus proposed a progressive framework of solving this class of problems(S. E. Dreyfus, 1969) while he criticized a simple but erroneous solution introduced by Saksena and Kumar. However, he stated that the framework he proposed is not-significantly efficient himself. According to the computation-intense nature of this class of problem, we do not have a universal and efficient solution to all the problems. Instead, different methods and algorithms have been developed to solve specific problems. Different efforts have been made in various directions. For example, instead of finding the optimal path, we can search for sub-optimal solutions; we can also use heuristic or randomized approaches to solve the problems.

4) Shortest-path problem involving penalties and prohibitions

The first three classes of shortest-path problems can all be understood and solved in a topological context. However, if we bring these problems to a geographic and realistic context, more factors should be taken into account. For example, because of traffic flow control, for vehicles passing (passing straight or making turns) there is certain amount of time cost for the traffic lights, yielding and prohibitions for left turns. To solve the shortest-path problems more accurately, those costs are very significant and need to be considered carefully. Lots of studies have been done in this field, S.Palottino and M. G. Scutella summarized that two major approaches has been used to deal with the problem [S.Palottino and M. G. Scutella, 1998]. The first approach is called "dual network", "i.e. the graph obtained by considering the original arcs as nodes and introducing one arc for each pair of subsequent arcs of the original graph, with a cost equal to the sum of the cost of the first arc and of the penalty." The other approach is the "expended network", movements are highlighted in the intersections by means of "dummy nodes and arcs" which including the cost of turn penalty or prohibitions.

5) Time dependant shortest-path problem

In the previous discussed classes of the shortest-path problems, we assume that the cost to go from one node to another is static. However, in real life applications, the situation is not exactly the same. More dynamic factors are involved. For example, one of the most significant factor is time. More specifically, if the passenger travels by their own vehicles, the time cost to go from one node to another will decided by the time he made the travel. If he travels in busy hours, the time he spent would be much more significant than mid-night. Or if the passenger is using public transit systems, his travel time would be affected by the time he reaches the starting node or intermediate transfer nodes, since there is a time-interval that the public transit passes/starts at that node. If we formulate a similar matrix representation T as we did for the multi-source shortest path problems, $T_{x,y}$ will not be static. Instead, $T_{x,y} = f(t)$. Specifically, we can define that the starting point of time counting is 0, So the previous two situation we discussed can be formulated as followed:

I. Passenger travelled by own vehicles:

$$\mathrm{f(t)} = \mathrm{g(t\%c)}$$

C is the cycle period of how the traffic condition changes. g()is the function to simulate how the change is in a cycle period.

II. Passenger travelled by public transits:

$$\mathrm{f(t)} = \begin{cases} \mathrm{cost}_{\mathrm{x,y}} + \mathrm{t\%I}, & t \neq 0 \\ \mathrm{cost}_{\mathrm{x,y}}, & \mathrm{t} = 0 \end{cases}$$

In this function, we assumed that first transit leaves at time 0, the interval of transit is I, and the $\mathrm{cost}_{x,y}$ is that the travel time cost when the passenger arrives at the station exactly when the transit starts/leaves, and wait no time.

Similar formulation of the problem is first introduced by Ariel Orda , Raphael Rom[Ariel Orda , Raphael Rom, 1990]. Algorithm researches and application explorations have been made by several other scholars. C. Malandraki and M.S. Daskin explored the formulations, properties and heuristic algorithms in time dependent vehicle routing problems, which is the first situation we introduced [C. Malandraki and M.S. Daskin, 1992]. K. Nachtigall introduced the application of time-dependent shortest-path problems in railway networks using modified Dijkstra algorithm which is the second situation we introduced [K. Nachtigall, 1995]. Further application of the time-dependent shortest-path formulation can also be integrated into any of the classes of shortest-path problems to make them more dynamic.

6) Shortest path in multi-route/modal network

Finding shortest path in multimodal network is a relatively new area in the shortest path problem family. In recent year, related studies have been done both for planning and routing purposes. In 1986, Crainic and Rousseau introduced their solution to find the optimal freight path in a bi-modal network by inserting delay variables in transfer terminal [Crainic and Rousseau, 1986]. In 1987, Spiess and Florian proposed their research for public transit user to make their path choices which includes transfers. They formulate the optimization model by studying the "optimization strategies" from transit riders, and the authors are able to control their algorithm to a polynomial level which is relatively efficient for large scale system applications[Spiess and Florian,1989]. Nguyen et al [1995] incorporate time-dependence factors to solve a dynamic passenger assignment problem in transit networks. Knoppers and Muller [1995] investigate the possibilities and limitations of coordinated transfers in public transit. They used probability theories to demonstrate the time distribution of transit operations and waiting time to suggest methods to minimize waiting time by configure the transit operation schedule. In 2000, Ziliaskopoulos and Wardell [2000] proposed a comprehensive intermodal optimal path algorithm accounting for waiting time, switching delays. The formulation of the network is well-designed to incorporate various factors which would affect the shortest-path decision, and the computational comeplexity of the algorithm is very prominent since it is independent of the of number of modes, and has been tested in realistic size networks. The shortcoming of his algorithm is that it requires a very complex dataset and the implementation is computationally intensive.

Some software packages already start to provide support for multimodal networks. ESRI updated their network analyst in ARCGIS 9.1 to incorporate an advanced connectivity model which can be used to represent complex scenarios such as multi-modal transportation networks. This enables users to efficiently model multiple forms of transportation across a single data set by using points of coincidence, such as rail stations or bus stops, which form the linkages between several different forms of transportation.

2. Current Practices in major public transit systems

To improve the system-wide and per-passenger efficiency of transit network, different measures have been taken by major metropolitan transit authorities. The following step of the study chooses two major public subway transit systems (New York and Washington) as examples to demonstrate different natures of each transit network and corresponding practices taken by local transit authorities in effort of advancing network efficiency. Efforts have been made to remedy the systematic deficiency and non-systematic deficiency. More specifically, on the system-wide side, public transit authorities are conducting analysis of the defects and disadvantages of the network and planning for reconstruction and extension projects. On the other side, those authorities try to provide better trip-planning assistance through various media to the public including route map, schedule, real time operation status update, and online trip planners and etc. Transfer-related efficiency issues are particularly addressed. Moreover, from the private sector side, the effort has been mainly focused on trip planning and assistance. Many web-based APIs are available to public and transit agents, among which the package provided by Google is the most prominent one. Detailed introduction will also be provided in the following part.

1) New York City Subway

New York City Subway (NYC subway) is the third largest subway system in the world and the largest in North American in terms of annual total boarding passengers (6,432,700 average weekday passengers in 2007). It is also one of few transit systems in North America operating 24 hours a day, 365 day a year. The system spreads throughout Manhattan, Brooklyn, Queens and Bronx and is consisted of 26 routes, all of which passes Manhattan except for the Brooklyn–Queens Crosstown Local (G) which connects Brooklyn and Queens directly. The following Figure is an overview of NYC subway system.

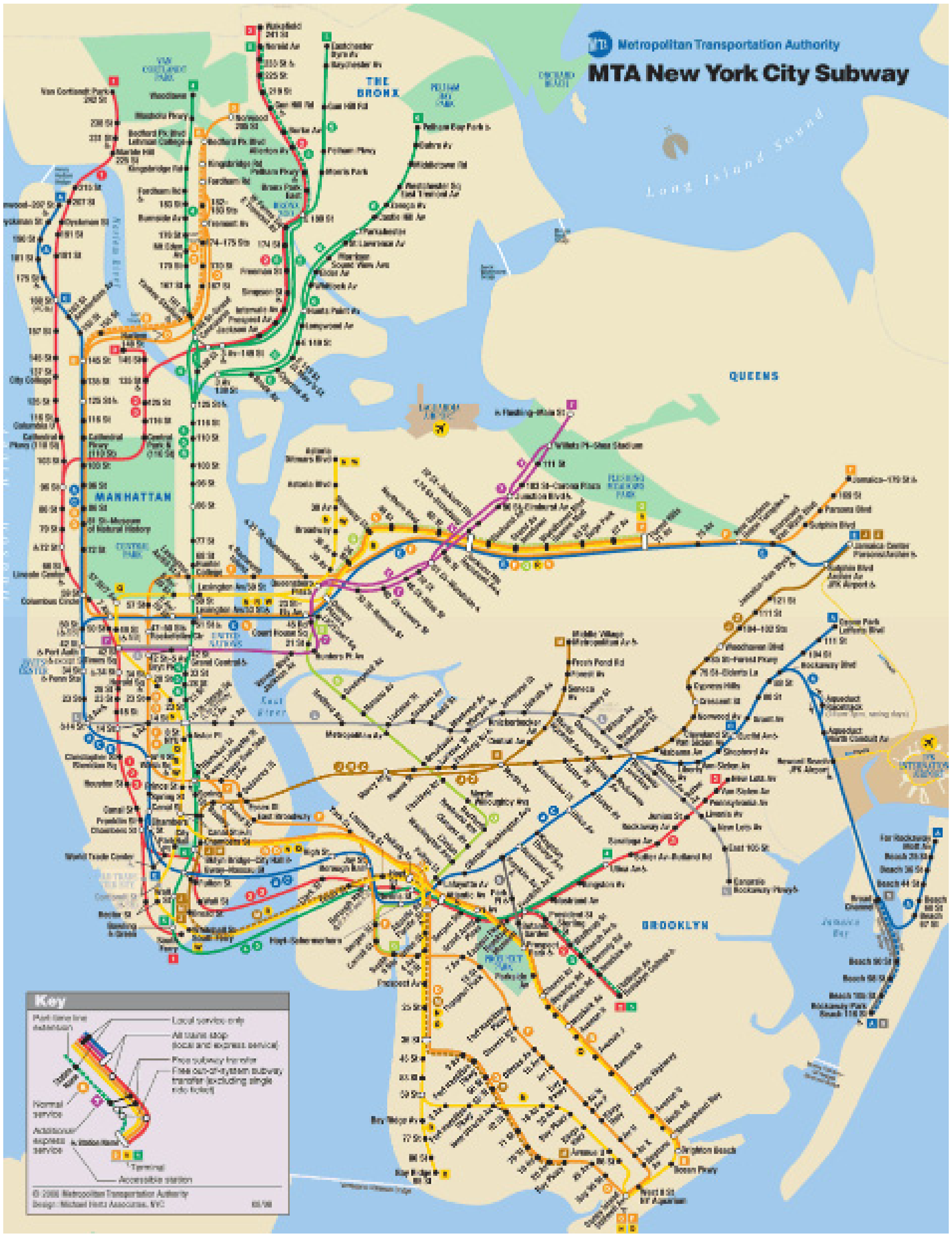


Figure 2 New York City Subway Map [Wikipedia, 2008]

(Source: http://www.mta.info/nyct/maps/submap.htm)

Comparing to other major transit networks in the nation, most of which adopted a hub and spoke paradigm, NYC subway system is more irregular in shape to accommodate with its unique land form and passengers' complex daily travel pattern. Topologically, lower Manhattan is the most connected part since great amount of travel demand is taking place here. West Brooklyn is also well connected to serve as a transfer center to connect Manhattan, lower Brooklyn and West Brooklyn.

Due to the complexity of the network and extensive demand of travelling, certain measures are adopted by Metropolitan Transit Authority (MTA) to reduce passengers' travel time and unnecessary transfers.

First of all, NYC subway features a dynamic route scheme both temporally and spatially. Temporally, some of the routes have both express and local services. Express services ensure the efficiency of transporting passengers from major stops especially in rush hours while local services provides greater flexibility to passengers. Spatially, some routes have an extra extension or alternative route during the rush hour to provide better flexibility. For example, during the afternoon rush hours, passengers going from Manhattan back to Brooklyn will have the option of taking the extended Route #5 subway to Brooklyn directly instead of making time-consuming transfers in Manhattan.

Secondly, NYC subway is multi-dimensioned, which enables multiple trains of different routes pass one stop at the same time to advance the instant capability. Meanwhile, it also raises the problem that making transfers at such stops become energy consuming and sometimes confusing. MTA has developed a sophisticated system to assist passengers to make transfers at those stops. Instructions and signs are carefully designed to guide passengers through all the way. Elevators and escalators are installed where vertical movement is needed. There are also some shuttle services to provide faster access to nearby stations. For example, the 42th Street Shuttle connects the most extensively used Time Square station and Grand Central Station.

Moreover, MTA provides online Trip planner for passengers to plan their trips with various customized options. The trip planner provides passenger the options to choose either to minimize transfers or minimize travel time. It can also recognize descriptive address which is more flexible and tolerant than using only physical addresses. Moreover, the trip planner can incorporate temporal information to accommodate the dynamic schedule of NYC subways as well as some other user preference such as route preference, mode preference (Subway, local bus and express bus) and accessibility issues. Recently, the planner has become available on popular mobile platforms. Some of the latest mobile platform's location-based service ability makes the planner much more convenient to use.

2) Washington Metropolitan Area Subway

Washington Metropolitan Area Subway system (Washington Metro) is the second largest subway system in USA in terms of number of annually boarding passengers (727,684 trips per weekday). The subway system is owned and operated by Washington Metropolitan Area Transit Authority (WMATA). Washington is mainly located in District of Columbia, and extended into Maryland and Virginia. There are 5 existing lines (Red line, blue line, orange line, yellow line, and green line) and 2 planned lines (Silver line and Purple line) in the metro network. The following Figure is an overview of Washington Metro.

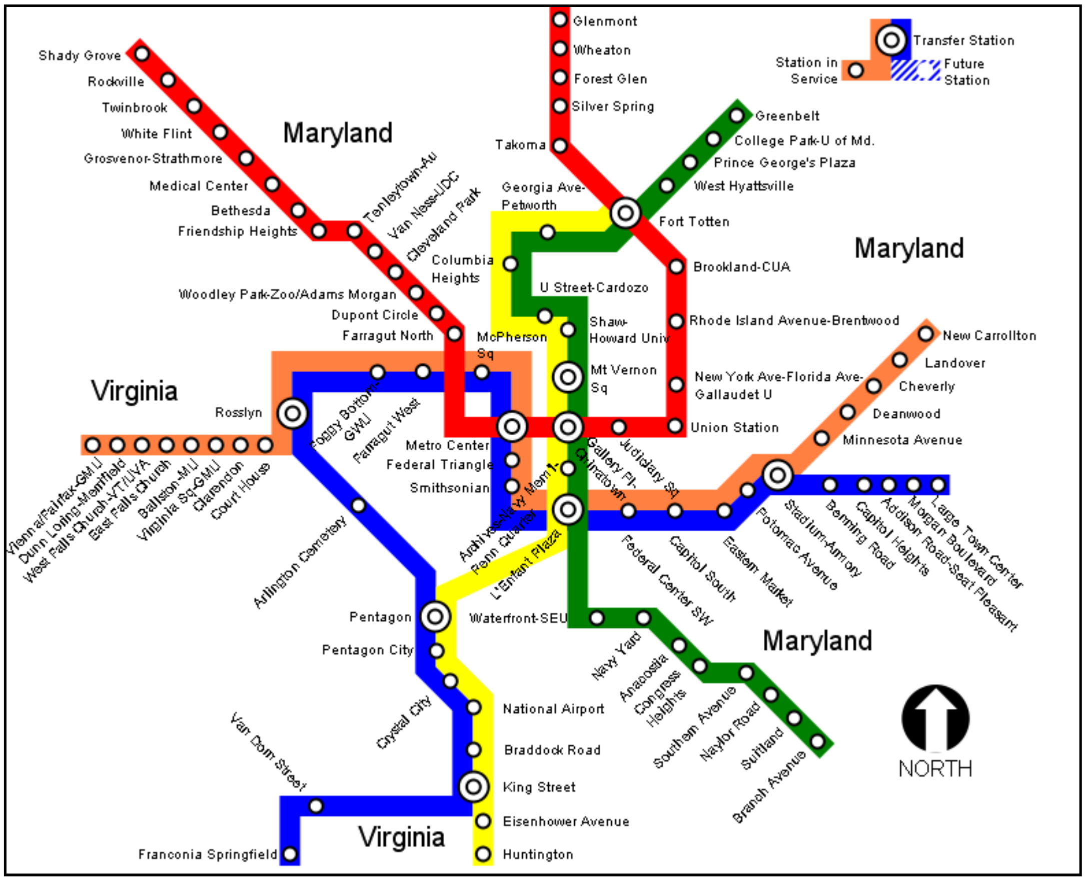


Figure 3 Washington Metropolitan Area Subway Map

(Source: http://en.wikipedia.org/wiki/Image:WMATA_system_map.svg)

Comparing to NYC subway, the Washington metro is designed following a spoke-hub distribution paradigm. There are both advantages and disadvantages of using such paradigm. Topologically, the spoke-hub structure is a relative economic-efficient way to connect all the nodes in a network comparing to a point-to-point model. The network structure is more intuitive for a user to understand and adapted to. It is an ideal structure for daily suburb to city travels. However, the structure also has its own drawbacks. Hubs in the network are very critical. Change or malfunction in the hubs might lead to paralysis of the whole or part of the network. Moreover, hubs are also the bottleneck of the network, the capacity of the hubs limits the total capacity of the network. However, Washington metro did not adopt a single hub mode, multiple hubs are designed in the center part of the service area, which reduce the burden of each hub and make the network more tolerant to unexpected situations. Last but not least, the spoke-hub paradigm is not efficient for suburb to suburb travels.

WMATA also took certain measures to remedy the disadvantages of the network. On the network structure level, two new lines are proposed. The silver line is more regular expansion of the network which connects Dulles International Airport and points west passing the center of city. The purple line has greater impact in terms of network structure. It serves as a rapid bypass for direct commuting from suburbs. Passengers can avoid unnecessary transfers at the center of city, and enjoy a shorter travel time. Meanwhile, it also increases the tolerance of the network by provide an alternative passage way from the city perimeter. Once there is unexpected situation happened in the center hubs, passengers can still use the purple line to access the rest of the network. The following Figure demonstrates how the purple line connects red line, green line, orange line and suburbs.

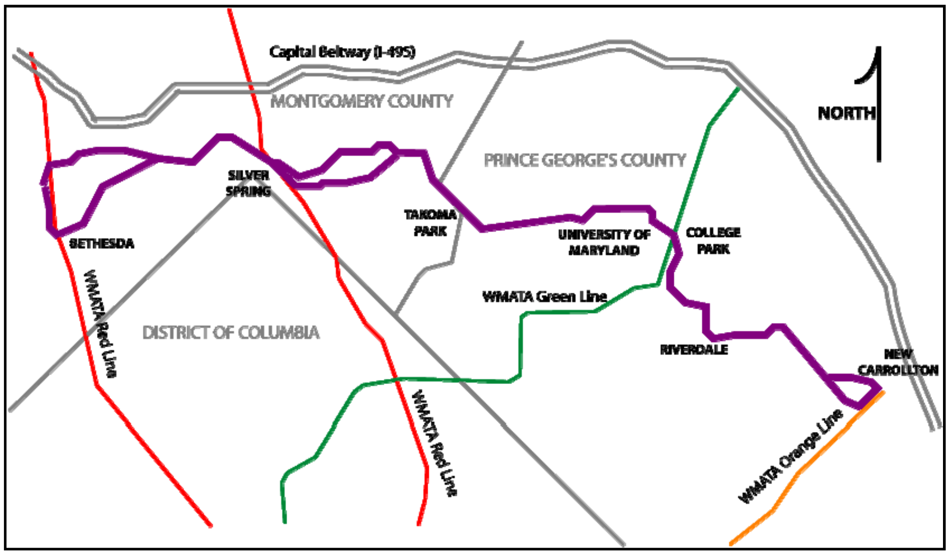


Figure 4 Proposed Purple line Map in Washington Metro Network

(Source: http://en.wikipedia.org/wiki/Image:Washington_Purple_Line.png)

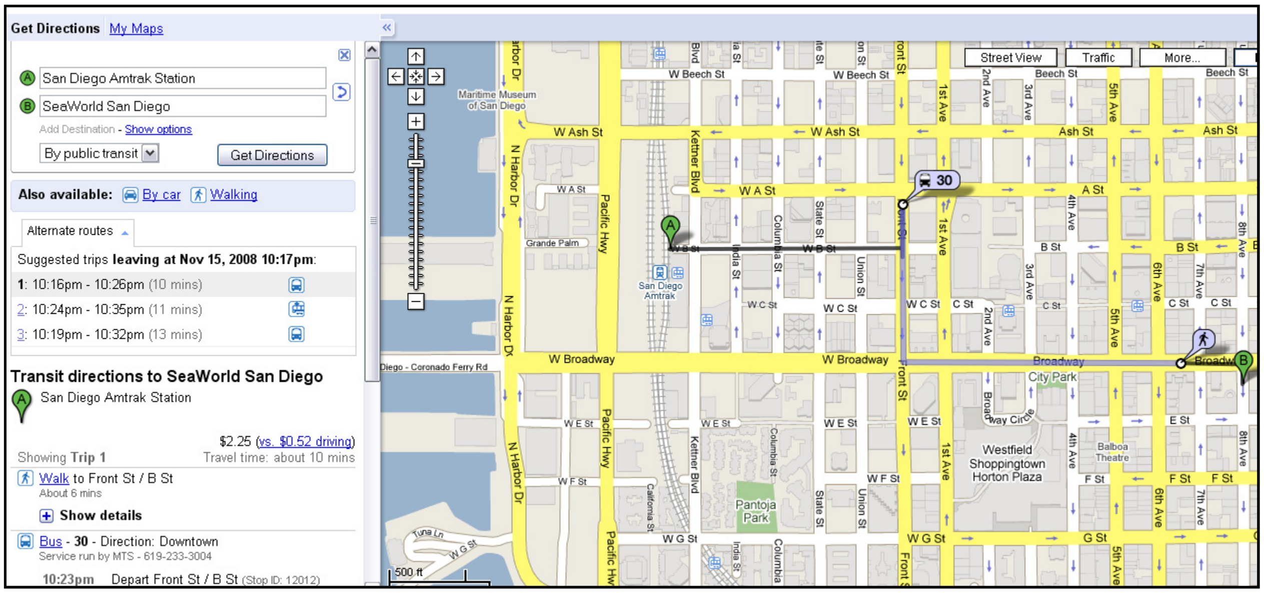
WMATA also developed a similar online trip planning system for passengers. It's also available on mobile platforms. Besides providing similar trip planning assistance info as the NYC subway trip planner does, the new feature as part of the online passenger service system is that the authority provides real time metrorail service disruption reports that describe current Metrorail operating status, including any delays of 10 minutes or more by using “eAlerts” (a custom e-mail service notifying you of disruptions affecting the lines and times of day you wish to monitor.), and also the real time update of elevator and escalator status online or by “ELLEN” (a custom elevator status e-mail notification service).

Other major transit agencies also have their own strategies to enhance the system-wide performance of their transit networks. For instance, Bay Area Rapid Transit (BART) developed a sophisticated automatic transit operation system to dynamically control the flow of all cars. Since BART is more an inter-city transit service, rail to local transit connection facilities are well established to reduce passengers’ transfer time. Further extension of routes is proposed to connect more cities adjacent to bay area. Chicago Transit Authority (known as “L” for its shape) focused their energy on the renovation and reorganization of the L loop where most of the routes converge. New lines are constructed or proposed to enhance the robustness of the center hub in within the L loop.

3) Google Transit Planner and Transit Feed Specification

Following the emerging mass transit grows in metropolitan regions, there is increasing demand for information related to transit trip planning from commuters. The trend also attracts private companies to provide information for commuters as part of their services. Various web-service providers offer different APIs for individual developers and public agents to store regional public transit related data and provide customized trip planning functionalities, among which, the package provided by Google seems to be one of the most prominent one.

The package is consisted of two parts. The first part is Google Transit Planner, the graphic user interface directly available to end users. Passengers can access the interface directly from their browsers. The use of the application is very straightforward. Passengers choose the metropolitan or city where they are in. A descriptive address for the starting point and destination are needed. Then, the application will automatically generate a "best" path for passengers with detailed instruction of the transit they are going to take, transfer points, path to walk to the bus stop or subway station, and other necessary information. Passengers also have the option to input departure time. If departure time is not given, the system will automatically suggest suitable departure time according to transit operation schedule. However, the system did not give specific information about how to define "best". Passengers can not choose their own priority, which could be a drawback for the system. However, the interface is very intuitive and the visual instruction is clear and easy to read. It is very suitable for passengers not familiar with the transit network. Fee information is also included in the planner. If the trip involves transfers, the planner will show the price for each section of the trip. The following Figure is a demonstration of the transit planner user interface.


Get Directions My Maps

San Diego Amtrak Station

SeaWorld San Diego

Add Destination - Show options

By public transit — Get Directions

Also available: By car Walking

Alternate routes

Suggested trips **leaving at Nov 15, 2008 10:17pm**:

1: 10:16pm - 10:26pm (10 mins)

2: 10:24pm - 10:35pm (11 mins)

3: 10:19pm - 10:32pm (13 mins)

Transit directions to SeaWorld San Diego

San Diego Amtrak Station

$2.25 (vs. $0.52 driving)

Showing **Trip 1** — Travel time: about 10 mins

Walk to Front St / B St

About 6 mins

Show details

Bus - **30** - Direction: Downtown

Service run by MTS - 619-233-3004

10:23pm Depart Front St / B St (Stop ID: 12012)

Figure 5 Google Transit Planner graphic user interface

(Source: http://maps.google.com/maps?hl=en&tab=wl)

Google also provides trip planner API to individual developers and public transit agents, which allows them to develop more customized web-based applications to suit their specific needs and aims. The system also shares most of the functionality and libraries from Google map. So, the new API is also compatible with some existing Google map API-based applications. Chicago Transit Authority has successfully developed a series of web applications based on Google map and Google transit APIs. Those APIs also offer interface to other databases, which allows “mash up” developing to include other peripheral information.

The second part is Google Transit Feed Specification (GTFS). It enables public transit agencies to convert transit related data (including route layout, schedule, fee, and etc) to utilize the Google Transit Planner to provide trip planning services instead of developing their own trip planning packages.

The first attempt of using GTFS was made by TriMet of Portland, OR transit agency after Google Transit’s initial launch in 2006, following by Bay Area Rapid Transit (BART) in bay area, and Tram et Bus de la CUB (TBC) in France. The number of agencies using GTFS has grown tremendously in the past two years. Until recently, there have been 90 participating agencies across the world, and with some other more joining soon. Those agencies include 77 from United States, 5 from Canada, 1 from Japan, 6 from Taiwan, China, 1 from Australia, and 11 from Europe.

The GTFS feeds from agencies should contain the following information: agency.txt, stops.txt, routes.txt, trips.txt, stop_times.txt, calendar.txt, calendar_dates.txt, fare_attributes.txt, fare_rules.txt, shapes.txt, frequencies.txt, and transfers.txt. All the routes should be given a unique identification number, stops can be shared by different routes. Other information should be recorded following the identification numbers accordingly. The information needed to run the Google Transit is very extensive and detailed, including route schedule and operating hours, stop types, and etc. Several projects are conducted to create applications to help public agents to convert or modify their own data into GTFS format more conveniently, those projects include: GoogleTransitDataFeed Project, Transit Timetable Publishing Application and etc. They are either programmed by Google, or by other third party groups.

The launch of Google Transit provides a great platform for public agencies to provide trip planning services to the public. The cost to prepare GTFS data is relatively low comparing to developing their own planning system. On the other hand, the standardized user interface makes it easier for passengers to use the same application for different cities or regions. Another advantage of GTFS is that it serves as a standard format for public transit from different agents. It's a great source to conduct scientific research using data from different agents without converting different format of data back and forth, and to support decisions from planning departments. Transit data of 17 agencies have already been available to download for scientific purposes, with more coming in the future. However, the potential drawback of Google Transit is that all the algorithms about the trip planning are sealed and not available to public. Thus, how the priority of passengers' preference is organized is unknown, and users can not define their own preference as the trip planners provided by NYC subway and Washington Metro do.

PROBLEMS AND RESEARCH

To deal with the high-standing oil price, traffic congestion problems and other social, environmental problems raised by private vehicles as the major commuting tool in metropolitan areas, the demand of public transit has been tremendously increasing during the last decades. As public transportation systems become more complex both in volume and dimension (multi-modal, multi-routes, potential transfers), there are increasing needs to provide passengers an efficient decision supporting system to help them deal with large volume of information and make efficient choice of route. Although the shortest path problems have been extensively studies during the past years, existing data structure and algorithms for shortest path problems could not fulfill the challenge brought by the new characteristics of the modern transit network.

First of all, one of the most significant issues in public transit trip planning is how to deal with transfers. Due to nature of traditional shortest path algorithms, transfers are often treated as penalties. However, as being discussed in the first part of the thesis, the impact of transfers to network structure and passengers' choices of routes is obviously underestimated. Topologically, it is difficult to describe transfers in the traditional network representation. Psychologically, it is imprudent to insert transfers as a numerical penalty since the impact is complex and volatile for different people. So, transfer deserves special attention when being dealt with in shortest path algorithms.

Secondly, according to the review of the shortest path solutions in multi-route/modal network, those solutions are still very immature. Due to the complexity and huge volume of public transit network, they are somehow either not prominent in computation efficiency, or the implementation is difficult considering the extra information needed and the complexity of the algorithms.

Therefore, it is necessary to design a new data structure which is suitable for transfer-involved trips in a multi-route/modal network, and corresponding algorithms which can conduct the shortest path analysis efficiently. Although traditional network representation and algorithms have their drawbacks when dealing with transfer-related shortest path problems, they can still provide very valuable knowledge and inspiration to be utilized.

Generally speaking, the objectives of this research is to creatively utilize traditional network representation to create a new suitable network representation and design relatively efficient algorithm to conduct analysis to find the shortest path with the first priority to minimize transfers in passengers' trips. More specifically, my research will include the following steps:

I. Network representation design and data retrieval

In this step, a new network representation will be designed. The idea is borrowed from the classic connectivity matrix. However, instead of having only one layer to contain all the route information, multiple layers will be introduced. Each layer will only contain the one route's information. Data from public transit agencies will be converted into the proposed network representation format.

II. Least transfer path algorithm design and implementation

In this part, a least transfer path algorithm will be designed utilizing the matrices created in the previous step. A series of matrices will be carefully designed to store all the detailed results from the algorithm. A graphic user interface will be designed to advance user's experience to input data, acquire least transfer path information and output results.

III. Least transfer path analysis in NYC subway system

A least transfer path analysis will be conducted in selected routes of the NYC subway system to validate the network representation and algorithm in a practical scale. The suggested path will be examined by comparing to real life experiences. Results of the analysis will be explored to provide more insightful knowledge about the transit network itself. All the data and algorithms will be retrieved and implemented on ArcGIS platforms using VBA and ArcObjects. More detailed methodology will be elaborated in the following part.

METHODOLOGY AND IMPLEMENTATION

1. Algorithm Design

1) Network Representation

To simplify the demonstration of the network representation proposed. A very simple multi-route/mode network is used for example (Figure 6). The traditional network representation views the network as a whole (Figure 6a). In this network, each node (for public transportation, bus stops are represented as nodes) is indicated by a unique number. The linkage between specific pair of nodes shows that they are connected directly. It presents the connectivity of the network very clearly. However, the limitation of this network representation is that it did not contain any route information. All the routes are assumed connected. Transfers between each pair of routes do not result in any kind of cost. Therefore, we cannot perform least-transfer path analysis base on this network representation without additional information inserted. To break through this limitation, some small changes is made to the original network representation. As you can see in the right part of Figure 6b, if we assume that there are two bus routes in the network, we can split the network into two parts, each part of the network represents one route. The modified representation only made a simple change to the original one; however, it did provide the foundation for further least-transfer path analysis.

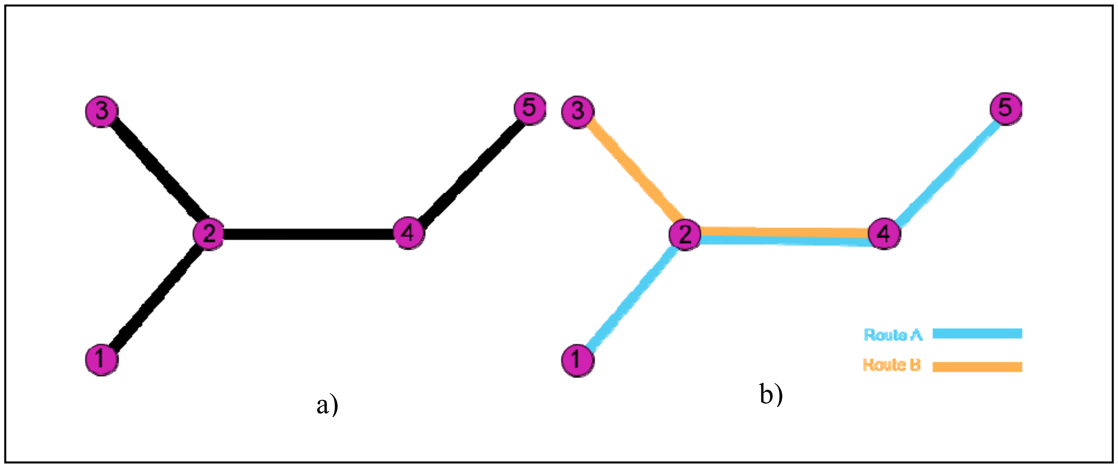


Figure 6 the traditional network and the multi-path network

2) C_x^i Matrices

Beyond the theoretical modified representation of the network. It is necessary to connect the network into a matrix form to facilitate our analysis. Inspired by the classic matrix representation, C matrix (Connectivity matrix), we will create an array of C matrices to represent the network (Figure 7)

$$C_A^1 \qquad\qquad C_B^1$$

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 1 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 & 1 \\ 5 & 0 & 0 & 0 & 1 & 0 \end{array} \qquad \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 1 & 0 \\ 3 & 0 & 1 & 0 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 \end{array}$$

Figure 7 C matrices of route A and route B

As you can see, each of these matrices shows the connectivity in different routes. If two nodes **a** and **b** (bus stops) are connected directly in route **X**, then the cell $\mathrm{C}_{\mathrm{x}}^{1}(\mathrm{a},\mathrm{b})$ will be marked 1. Otherwise, if they are not connected, it will be marked 0. For all the cells on the main diagonal, they will all be marked as 0. Direction of connectivity matters in this matrix. If a → b is allowed and b→ a is not allowed, then $\mathrm{C}_{\mathrm{x}}^{1}(\mathrm{a},\mathrm{b}) = 1$ and $\mathrm{C}_{\mathrm{x}}^{1}(\mathrm{b},\mathrm{a}) = 0$. If all the connections are bi-directional, then the matrix is transposable.

3) C_x^iMatrices → D_{x} Matrices

For each of those C-matrices, we can create a Shimbel Distance matrix (D matrix) respectively. The Shimbel Distance matrix finds the shortest path between each pair of nodes in terms of number of linkages of a path. Operationally, the D matrix is computed by successively squaring the connectivity matrix (C matrix), and noting after each iteration whether or not any new non-zero elements occur. If so, the square of that C matrix is entered into the appropriate cell in the D matrix. The procedure stops when the square of the C matrix reaches the diameter of the network or all the elements are filled in with non-zero values except for the elements on the main diagonal. Please look at the following example of how to create a D matrix for $\mathrm{C}_{\mathrm{A}}^{1}$ (Figure 8).

$$C_A^1 \quad \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 1 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 1 & 0 & 0 & 1 \\ 5 & 0 & 0 & 0 & 1 & 0 \end{array} \rightarrow \quad D_A \quad \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & & & \\ 2 & 1 & 0 & & 1 & \\ 3 & & & 0 & & \\ 4 & & 1 & & 0 & 1 \\ 5 & & & & 1 & 0 \end{array}$$

$$C_A^2 = C_A^1 * C_A^1 \quad \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 1 & 0 & 0 & 1 & 0 \\ 2 & 0 & 2 & 0 & 0 & 1 \\ 3 & 0 & 0 & 0 & 0 & 0 \\ 4 & 1 & 0 & 0 & 2 & 0 \\ 5 & 0 & 1 & 0 & 0 & 1 \end{array} \rightarrow \quad D_A \quad \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & & 2 & \\ 2 & 1 & 0 & & 1 & 2 \\ 3 & & & 0 & & \\ 4 & 2 & 1 & & 0 & 1 \\ 5 & & 2 & & 1 & 0 \end{array}$$

$$C_A^3 = C_A^1 * C_A^2 \quad \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 2 & 0 & 0 & 1 \\ 2 & 2 & 0 & 0 & 3 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 3 & 0 & 0 & 2 \\ 5 & 1 & 0 & 0 & 2 & 0 \end{array} \rightarrow \quad D_A \quad \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & \backslash & 2 & 3 \\ 2 & 1 & 0 & \backslash & 1 & 2 \\ 3 & \backslash & \backslash & 0 & \backslash & \backslash \\ 4 & 2 & 1 & \backslash & 0 & 1 \\ 5 & 3 & 2 & \backslash & 1 & 0 \end{array}$$

Figure 8 Construction of D matrix for Route A

As you can see, rather than enumerating the total number of paths between two nodes, D matrix determines the least number of linkages between two nodes. In the matrix procedure, the power of C matrix is recorded in the appropriate cell of the D matrix when non-zero products occur for the first time. For example, if we want to find the shortest path from node 5 to node 1. The cell value for D(5,1) equals to 3, if we look at Figure 1, we can see that the shortest path there would be 5→4→2→1, exactly 3 linkages are used. However, network of route A is not a "connected" network, which means that you cannot visit all pairs of the nodes with route A. The result is that in the D_A Matrix, some of the cell values are not filled. The same procedures can be done to create D_B matrix. So, we will have two D matrices for each of the routes (Figure 9).

$$D_A \qquad \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & \backslash & 2 & 3 \\ 2 & 1 & 0 & \backslash & 1 & 2 \\ 3 & \backslash & \backslash & \mathbf{0} & \backslash & \backslash \\ 4 & 2 & 1 & \backslash & 0 & 1 \\ 5 & 3 & 2 & \backslash & 1 & 0 \end{array}$$

$$D_B \qquad \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & \backslash & \backslash & \backslash & \backslash \\ 2 & \backslash & 0 & 1 & 1 & \backslash \\ 3 & \backslash & 1 & 0 & 2 & \backslash \\ 4 & \backslash & 1 & 2 & 0 & \backslash \\ 5 & \backslash & \backslash & \backslash & \backslash & 0 \end{array}$$

Figure 9 D matrices for route A and B

4) D_i Matrices → D^{LTP} Matrices

As I've discussed just now, each of the networks of route A and route B is not a completely connected network. So, to visit certain pairs of nodes, it is necessary to make a transfer. So, in the following part, I will show how to develop a matrix, to record the least transfer path (LTP) for each of pair of nodes. I called this matrix D^{LTP} Matrix. For a pair of nodes a and b, the corresponding cell in D^{LTP} matrix D^{LTP}(a, b) shows how many routes is needed to go from node a to node b.

For D_0^{LTP} matrix, it records the path using only 1 route (Figure 10):

$$\begin{array}{c|ccccc} D_A & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \backslash & 1 & \backslash & 2 & 3 \\ 2 & 1 & \backslash & \backslash & 1 & 2 \\ 3 & \backslash & \backslash & \backslash & \backslash & \backslash \\ 4 & 2 & 1 & \backslash & \backslash & 1 \\ 5 & 3 & 2 & \backslash & 1 & \backslash \end{array} \oplus \begin{array}{c|ccccc} D_B & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \backslash & \backslash & \backslash & \backslash & \backslash \\ 2 & \backslash & \backslash & 1 & 1 & \backslash \\ 3 & \backslash & 1 & \backslash & 2 & \backslash \\ 4 & \backslash & 1 & 2 & \backslash & \backslash \\ 5 & \backslash & \backslash & \backslash & \backslash & \backslash \end{array} \rightarrow \begin{array}{c|ccccc} D_0^{LTP} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \backslash & 1 & & 1 & 1 \\ 2 & 1 & \backslash & 1 & 1 & 1 \\ 3 & & 1 & \backslash & 1 & \\ 4 & 1 & 1 & 1 & \backslash & 1 \\ 5 & 1 & 1 & & 1 & \backslash \end{array}$$

Figure 10 Construction of D_0^{LTP} matrix

In the procedure above, we can see that we combine D_A and D_B to create D_0^{LTP}. To combine them, we are not simply adding two matrices together. Instead, we mark all the non-zero values in D_A and D_B as 1 in D_0^{LTP} matrix, which means that for those pair of nodes in the matrix D_0^{LTP} marked by 1, we only use one route to accomplish to path. In other words, it means no transfer is needed. The other values are left blank for further procedures. The cells on the main diagonal are left blank on purpose.

To createD_1^{LTP}, multiple procedures are performed. Please look at the Figure 11 below:

$$\begin{array}{c|ccccc} D_0^{LTP} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \backslash & 1 & & 1 & 1 \\ 2 & 1 & \backslash & 1 & 1 & 1 \\ 3 & & 1 & \backslash & 1 & \\ 4 & 1 & 1 & 1 & \backslash & 1 \\ 5 & 1 & 1 & & 1 & \backslash \end{array} \rightarrow \begin{array}{c|ccccc} D_0^{LTP} * D_0^{LTP} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 3 & 2 & 2 & 2 & 2 \\ 2 & 2 & 4 & 1 & 3 & 2 \\ 3 & 2 & 1 & 2 & 1 & 2 \\ 4 & 2 & 3 & 1 & 4 & 2 \\ 5 & 2 & 2 & 2 & 2 & 3 \end{array} \rightarrow \begin{array}{c|ccccc} D_1^{LTP} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \backslash & 1 & 2 & 1 & 1 \\ 2 & 1 & \backslash & 1 & 1 & 1 \\ 3 & 2 & 1 & \backslash & 1 & 2 \\ 4 & 1 & 1 & 1 & \backslash & 1 \\ 5 & 1 & 1 & 2 & 1 & \backslash \end{array}$$

Figure 11 Construction of D_1^{LTP} matrix

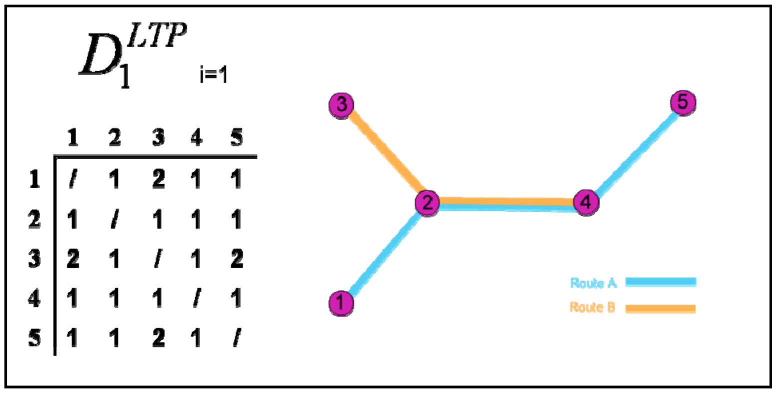
From Figure 11, we can see that, the first procedure we have done is to multiple D_0^{LTP} by D_0^{LTP}. The product shows how many possible routes to go from one node to another by make only one transfer. Then, the second procedure we have done is to update the new occurred non-zero products to the previous D_0^{LTP} matrix to create D_1^{LTP} as 2. So, for a specific new updated cell value in $D_1^{LTP}(a,b)$ shows that to go from node a to b, at most one transfer need to be done to accomplish the path.

Now, the D^{LTP} matrix is finished, which means that now we find paths between all pairs of nodes making at most 1 transfer. Therefore, we can stop our procedure. D_1^{LTP} would be the final D^{LTP} matrix. However, for a more complex network involving more nodes and routes, more D_i^{LTP} might be needed. But the procedure is similar as I described above. If we define the operation of creating D_i^{LTP} as "$\otimes$ ", we will come up with the following flow:

$$D_0^{LTP}=D_A \oplus D_B$$

$$D_i^{LTP}=D_0^{LTP} \otimes D_{i-1}^{LTP} \ (i>=1)$$

The algorithm stops when D_i^{LTP} is completed, and the D_i^{LTP} will be the final D^{LTP} matrix. For our simple network, please look at Figure 12 to understand D^{LTP} matrix better.



D_1^{LTP} i=1

	1	2	3	4	5
1	/	1	2	1	1
2	1	/	1	1	1
3	2	1	/	1	2
4	1	1	1	/	1
5	1	1	2	1	/

Figure 12 Final D^{LTP} matrix and the network representation

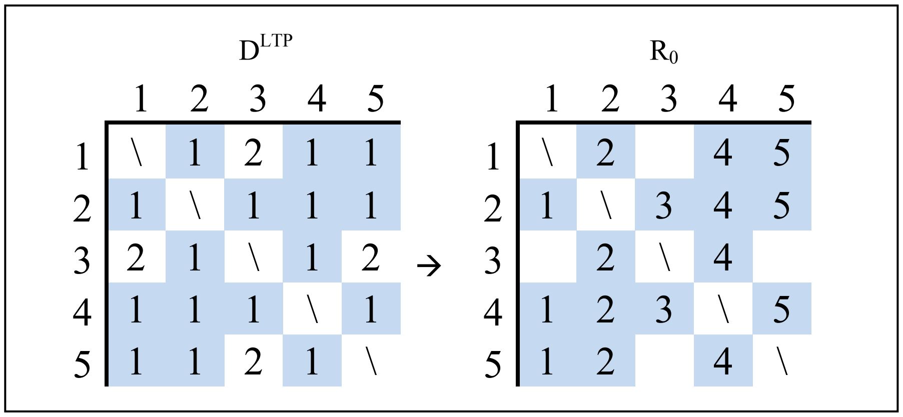
So, for example, if we want to go from node 1 to node 3, the path we will choose is $1 \rightarrow 2 \rightarrow 3$. So, we need to make a transfer at node 2. If we look back to the cell D^{LTP} (1,3) =2. It means that two routes are needed, which means that we need to make 1 transfer.

5) D^{LTP} Matrices → R, NL, RL Matrices

One shortage of D^{LTP} matrix is that it only shows how many transfers we need to make to accomplish to path. Some necessary information of the path is still missing for passengers such as transfer points, which route to take and etc. So, it is necessary to construct some other supplementary matrices to store the detailed path information.

Three matrices will be constructed. The first matrix stores the information of the transfer points (R matrix). The second matrix stores the information that how many linkages are passed from the origin node to the end node (NL matrix). The third matrix stores the information of that what is the route we are going to transfer from and to (RL matrix). The following part will demonstrate how we create R, NL, and RL matrix step by step.

For the first iteration, three matrices are initialized. To create R matrix, we can look at the D^{LTP} matrix. For all the cells with value 1 in the D^{LTP} matrix, we mark the corresponding cells in the R with the destination node index, since there are no intermediate nodes. Please look at Figure 13 below for details:



$$\overset{D^{LTP}}{\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \backslash & 1 & 2 & 1 & 1 \\ 2 & 1 & \backslash & 1 & 1 & 1 \\ 3 & 2 & 1 & \backslash & 1 & 2 \\ 4 & 1 & 1 & 1 & \backslash & 1 \\ 5 & 1 & 1 & 2 & 1 & \backslash \end{array}} \rightarrow \overset{R_0}{\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \backslash & 2 & & 4 & 5 \\ 2 & 1 & \backslash & 3 & 4 & 5 \\ 3 & & 2 & \backslash & 4 & \\ 4 & 1 & 2 & 3 & \backslash & 5 \\ 5 & 1 & 2 & & 4 & \backslash \end{array}}$$

Figure 13 Create R_0 Matrix from D^{LTP}

To create NL and RL layer, we can simply compare all the corresponding non-zero cell values in the both D_A and D_B, and updated the smallest value to the TP layer. If the smallest value is in D_A, we mark the corresponding cell in RL layer with A; if the smallest value is in D_B, we mark it with B. If there are more than one D matrix has the smallest value, then we will choose the D matrix base on alphabetical order (The criteria to choose may change the route we use for certain step, but will not change the total number of linkages that have to be covered and transfer points.) The result is shown in Figure 14 below:

DA

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & \backslash & 2 & 3 \\ 2 & 1 & 0 & \backslash & 1 & 2 \\ 3 & \backslash & \backslash & 0 & \backslash & \backslash \\ 4 & 2 & 1 & \backslash & 0 & 1 \\ 5 & 3 & 2 & \backslash & 1 & 0 \end{array}$$

DB

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & \backslash & \backslash & \backslash & \backslash \\ 2 & \backslash & 0 & 1 & 1 & \backslash \\ 3 & \backslash & 1 & 0 & 2 & \backslash \\ 4 & \backslash & 1 & 2 & 0 & \backslash \\ 5 & \backslash & \backslash & \backslash & \backslash & 0 \end{array}$$

$$\downarrow$$

NL_0

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \backslash & 1 & & 2 & 3 \\ 2 & 1 & \backslash & 1 & 1 & 2 \\ 3 & & 1 & \backslash & 2 & \\ 4 & 2 & 1 & 2 & \backslash & 1 \\ 5 & 3 & 2 & & 1 & \backslash \end{array}$$

RL_0

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \backslash & A & & A & A \\ 2 & A & \backslash & B & A & A \\ 3 & & B & \backslash & B & \\ 4 & A & A & B & \backslash & A \\ 5 & A & A & & A & \backslash \end{array}$$

Figure 14 Create NL_0 Matrix from D_A and D_B

For the second iteration, let us first look at the cells in D^{LTP} matrix with value 2. According to the definition of D^{LTP} matrix, if we want to go from node to another with value 2, the least transfer number to accomplish it is 1. Therefore, the corresponding cells in R matrix must be still empty by now. Let's look at the route from node 1 to 3 for example. Let us look at Figure 15 for details:

NL

	1	2	3	4	5
1	\	1		2	3
2	1	\	1	1	2
3		1	\	2	
4	2	1	2	\	1
5	3	2		1	\

\	+	\	=	\
1	+	1	=	2
\	+	\	=	\
2	+	2	=	4
3	+	\	=	\

R

	1	2	3	4	5
1	\	2	2	4	5
2	1	\	3	4	5
3		2	\	4	
4	1	2	3	\	4
5	1	2		4	\

NL

	1	2	3	4	5
1	\	1	2	2	3
2	1	\	1	1	2
3		1	\	2	
4	2	1	2	\	1
5	3	2		1	\

RL

	1	2	3	4	5
1	\	A	AB	A	A
2	A	\	B	A	A
3		B	\	B	
4	A	A	B	\	A
5	A	A		A	\

Figure 15 Create R, NL, RL matrices from initial NL and D^{LTP}

In the NL layer of first iteration, row 1 shows the number of linkages of the shortest path to go from node 1 to all other nodes with no transfers, while column 3 shows all the number of linkages of the shortest path to go from all other nodes to node 3. We can find that the shortest path to go from node 1 to node 3 making one transfer would be making transfer at node 2, total linkages in this path would be two comparing making transfer at node 4, which need 4 linkages to complete the path. So, we can update the R(1,3)=2, which means that to go from node 1 to 3, the transfer point should be node 2. We can also update the NL(1,3)=2, which shows the number of linkages in that shortest path. For RL matrix, we knew that the least transfer path from 1 h 3 is starting from node 1, make a transfer at node 2, then arrived at Node 3. So, we can refer to the RL matrix we just initiated. R(1,2)=A, R(2,3)=B. So, in RL matrix, we will update cell RL(1,3) with AB. The other three blank values can also be updated by following the same procedures.

From the D^{LTP} matrix, we can see that there is no cell value greater than 2, which means that we can find the path between any pair of nodes making at most 1 transfer. So, we do not need to further iterate to update more matrices. However, as the network becomes more complicated, there is chance that we might need to have further iterations to complete the matrix. The third or further iterations are the same as the second iteration. Please look at Figure 16 for the completed R, NL, RL matrices.

R

	1	2	3	4	5
1	\	2	2	4	5
2	1	\	3	4	5
3		2	\	4	
4	1	2	3	\	4
5	1	2		4	\

NL

	1	2	3	4	5
1	\	1	2	2	3
2	1	\	1	1	2
3		1	\	2	
4	2	1	2	\	1
5	3	2		1	\

RL

	1	2	3	4	5
1	\	A	AB	A	A
2	A	\	B	A	A
3	BA	B	\	B	AB
4	A	A	B	\	A
5	A	A	BA	A	\

Figure 16 Completed R, NL, RL matrices

To read those matrices, I will use the path from node 5 to node 3 for example. Firstly, look at the TP layer, the cell value of R (5, 3), which is 2. It means that we need to make transfer at 3 to complete the path. Then, we want to know how we can go from node 2 to node 3. This time, we check the cell value of R(2, 3), which is 3. It means that we can go directly from node 2 to node 3 making no transfer. So, we know that the path would be 5→2→3. The transfer point is 2. Now let's look at the cell value of NL (5, 2) and NL (2, 3), which are 2 and 1 respectively. It means that to go from node 5 to node 2, the shortest path takes 2 linkages, while the path from 2 to 3 takes 1. So, the total linkages from 5 to 3 would be 3. This can also be found by checking the NL(5,3)=3. Last, let us look at the cell value of RL (5, 3), which is "AB". That means we will transfer from route A to route B.

So, we can summarize the Least-Transfer Path from node 5 to node 3:

Path: 5→ 2 → 3

Transfer Node: 2

Linkages: 2+1=3

Routes: A → B

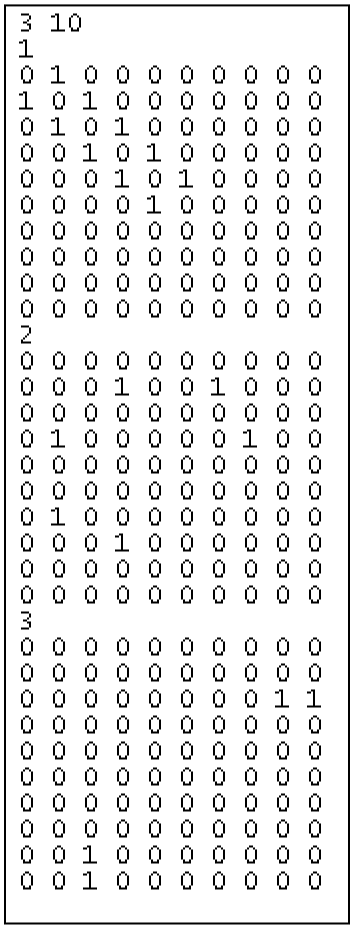
So, as we can see, these matrices provide us all the information we need for a public transit user to plan their path from a specific node to another in great details.

2. Implementation

The previous part explained the algorithms and flow to find the shortest path theoretically. In the following part, I will explain the implementation of my algorithms. Since ArcGIS is one the most popular GIS platforms currently in use, I will use ArcGIS as an example for implementation taking advantage of the flexible extensibility in ArcGIS by using VBA and ArcObjects. The programming code can be found in the Appendix.

1) Data Preparation

In order to implement the algorithm, the necessary data is the connectivity matrices for each route in your public transit network. All the nodes (stops) should be indexed with a unique ID (from 1 to X). Routes should be indexed (from 1 to X), too. This procedure can be done either manually or by writing scripts to construct Connectivity matrices. Connectivity matrices should be formulated and recorded in a text file to be loaded. An example of the text file is shown in Figure 17:



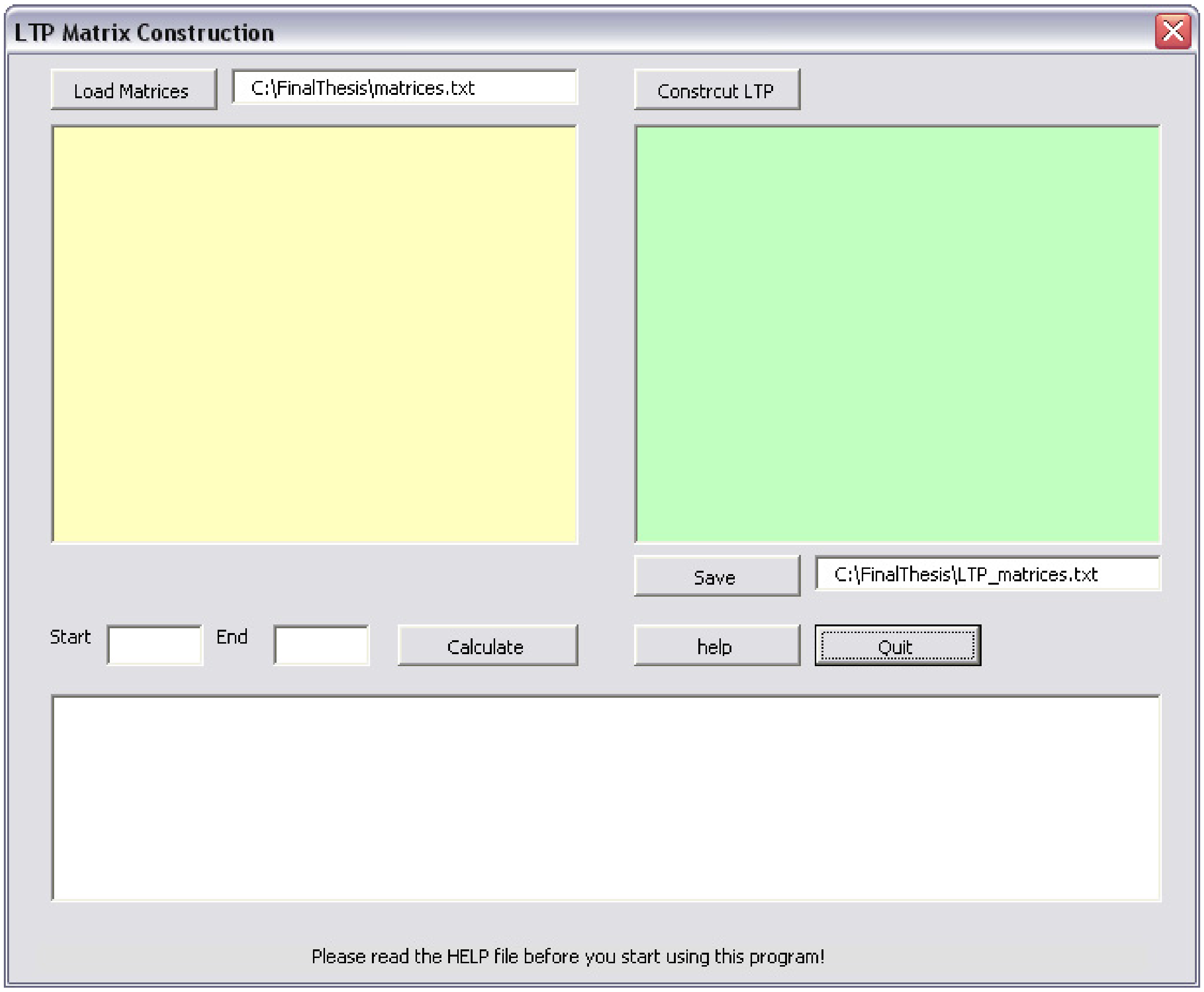
```
3 10
1
0 1 0 0 0 0 0 0 0 0
1 0 1 0 0 0 0 0 0 0
0 1 0 1 0 0 0 0 0 0
0 0 1 0 1 0 0 0 0 0
0 0 0 1 0 1 0 0 0 0
0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
2
0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
3
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0
```

Figure 17 Example of connectivity matrix text file

The First line shows the number of routes and number of nodes in the transit network. The every connectivity matrix of a certain route is shown in a space-delimited matrix with the index of route at the beginning.

2) Interface Design

Utilizing the integrated VBA development Environment in ArcGIS desktop 9.3, I designed a concise graphic user interface. Necessary functionalities are built in for the ease of use (Figure 18).



C:\FinalThesis\matrices.txt

C:\FinalThesis\LTP_matrices.txt

Please read the HELP file before you start using this program!

Figure 18 graphic user interface of the Least Transfer Path program in ArcGIS platform

Steps to use the program:

I. Type in the full path of the text file containing the network information in the left-upper textbox, click “Load Matrices” to proceed. In the left yellow textbox, the C matrices of each route will show up.

II. Click “Construct LTP” to implement the core algorithms which I demonstrated in the Algorithm design part. D matrices, D^{LTP} matrices, R matrices, NL matrices, and RL matrices will show up in the green textbox.

III. Input the starting and ending node index, click "Calculate". The detailed instructions of the least transfer path will show up in the white textbox below.

IV. Click "Save" to save the result in a text file for reference.

V. You can also click "Help" to load instructions to use the program, and click "Quit" to end the program.

3. Test result

A small scale network was created to test the correctness of program. In the test network, there are three routes (yellow, blue, and red) and 10 nodes. Each of the nodes is given a sequential identification number from 1 to 10. The C matrix of each route is created accordingly. The following Figure gives an overview of the small scale network. It is worth mentioning that the red route does not have a stop at node 3.

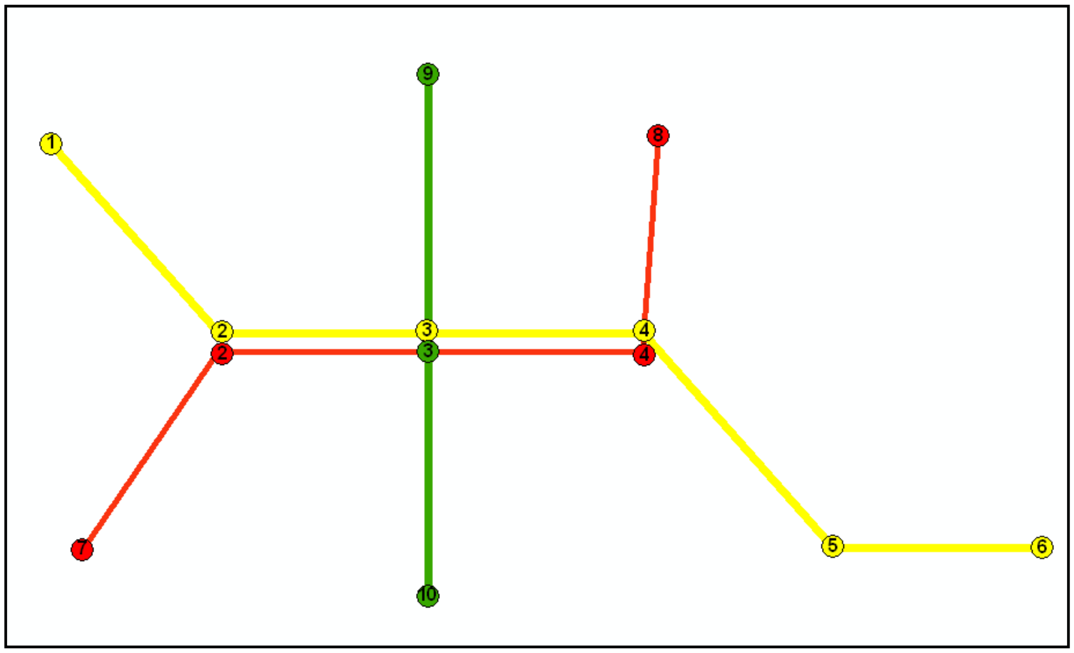


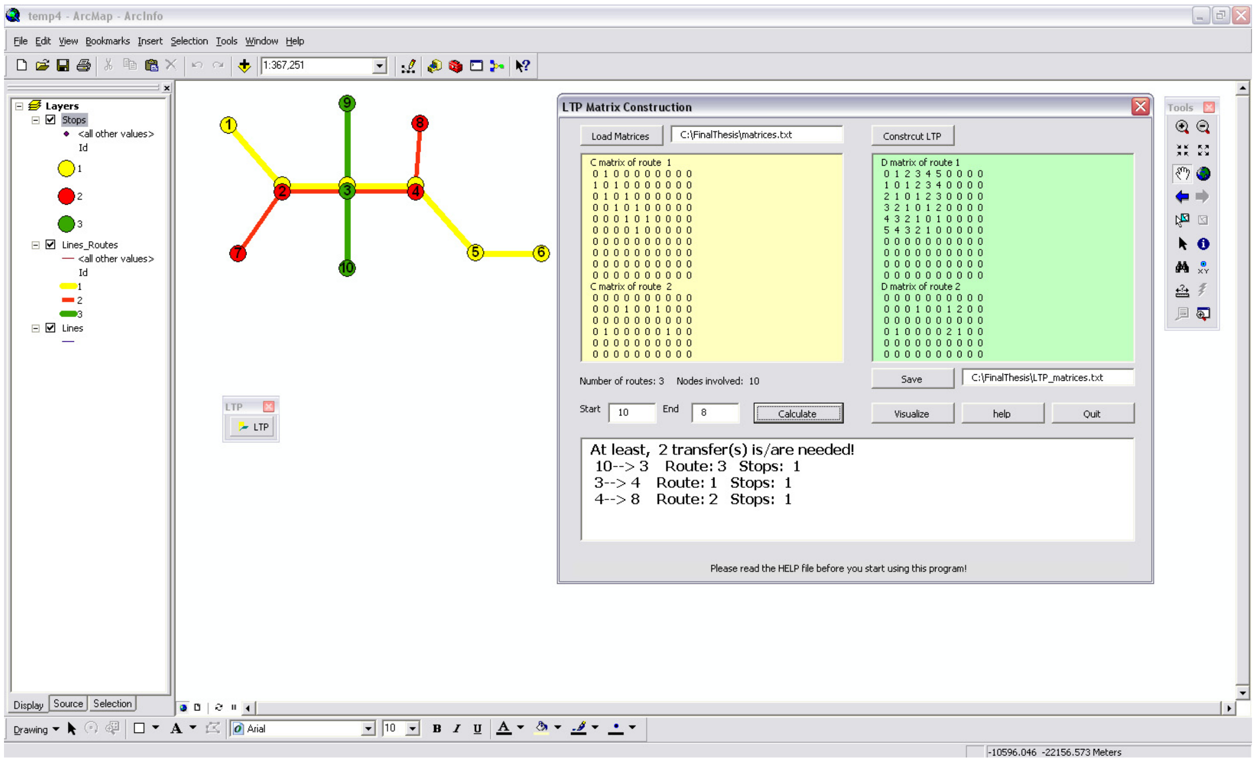
Figure 19 a small scale network with 3 routes and 10 nodes

After the data are inputted into the application, the following results are provided:

```
                 Final DTLP Matrix
 0    1    1    1    1    1    2    2    2    2
 1    0    1    1    1    1    1    1    2    2
 1    1    0    1    1    1    2    2    1    1
 1    1    1    0    1    1    1    1    2    2
 1    1    1    1    0    1    2    2    2    2
 1    1    1    1    1    0    2    2    2    2
 2    1    2    1    2    2    0    1    3    3
 2    1    2    1    2    2    1    0    3    3
 2    2    1    2    2    2    3    3    0    1
 2    2    1    2    2    2    3    3    1    0
                  Final R matrix
 0    2    3    4    5    6    2    2    3    3
 1    0    3    4    5    6    7    8    3    3
 1    2    0    4    5    6    2    4    9   10
 1    2    3    0    5    6    7    8    3    3
 1    2    3    4    0    6    4    4    3    3
 1    2    3    4    5    0    4    4    3    3
 2    2    2    4    4    4    0    8    2    2
 2    2    4    4    4    4    7    0    4    4
 3    3    3    3    3    3    3    3    0   10
 3    3    3    3    3    3    3    3    9    0
                  Final NL matrix
 0    1    2    3    4    5    2    3    3    3
 1    0    1    1    3    4    1    2    2    2
 2    1    0    1    2    3    2    2    1    1
 3    1    1    0    1    2    2    1    2    2
 4    3    2    1    0    1    3    2    3    3
 5    4    3    2    1    0    4    3    4    4
 2    1    2    2    3    4    0    3    3    3
 3    2    2    1    2    3    3    0    3    3
 3    2    1    2    3    4    3    3    0    2
 3    2    1    2    3    4    3    3    2    0
                  Final RL matrix
 /    1    1    1    1    1   12   12   13   13
 1    /    1    2    1    1    2    2   13   13
 1    1    /    1    1    1   12   12    3    3
 1    2    1    /    1    1    2    2   13   13
 1    1    1    1    /    1   12   12   13   13
 1    1    1    1    1    /   12   12   13   13
21    2   21    2   21   21    /    2  213  213
21    2   21    2   21   21    2    /  213  213
31   31    3   31   31   31  312  312    /    3
31   31    3   31   31   31  312  312    3    /
```

Figure 20 Test results

All results are thoroughly examined, the suggested path between each pair of nodes are compared with the actual least transfer routes. All of the results are correct. Taking the path from Node 10 to Node 8 as an example (Figure 21), the suggested least transfer path is 10 → 3 (route 3), then 3 → 4 (route 1), then 4→ 8 (route 2), which is exactly the same as the supposed least transfer path.



```
C:\FinalThesis\matrices.txt

C matrix of route  1
 0 1 0 0 0 0 0 0 0 0
 1 0 1 0 0 0 0 0 0 0
 0 1 0 1 0 0 0 0 0 0
 0 0 1 0 1 0 0 0 0 0
 0 0 0 1 0 1 0 0 0 0
 0 0 0 0 1 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0
C matrix of route  2
 0 0 0 0 0 0 0 0 0 0
 0 0 0 1 0 0 1 0 0 0
 0 0 0 0 0 0 0 0 0 0
 0 1 0 0 0 0 0 1 0 0
 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0

D matrix of route 1
 0 1 2 3 4 5 0 0 0 0
 1 0 1 2 3 4 0 0 0 0
 2 1 0 1 2 3 0 0 0 0
 3 2 1 0 1 2 0 0 0 0
 4 3 2 1 0 1 0 0 0 0
 5 4 3 2 1 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0
D matrix of route 2
 0 0 0 0 0 0 0 0 0 0
 0 0 0 1 0 0 1 2 0 0
 0 0 0 0 0 0 0 0 0 0
 0 1 0 0 0 0 2 1 0 0
 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0

C:\FinalThesis\LTP_matrices.txt

Number of routes: 3    Nodes involved:  10

Start  10    End  8

At least,  2 transfer(s) is/are needed!
 10--> 3    Route: 3   Stops:  1
 3--> 4    Route: 1   Stops:  1
 4--> 8    Route: 2   Stops:  1

Please read the HELP file before you start using this program!
```

Figure 21 Implementation result of LTP program

CASE STUDY

1. Case study site introduction

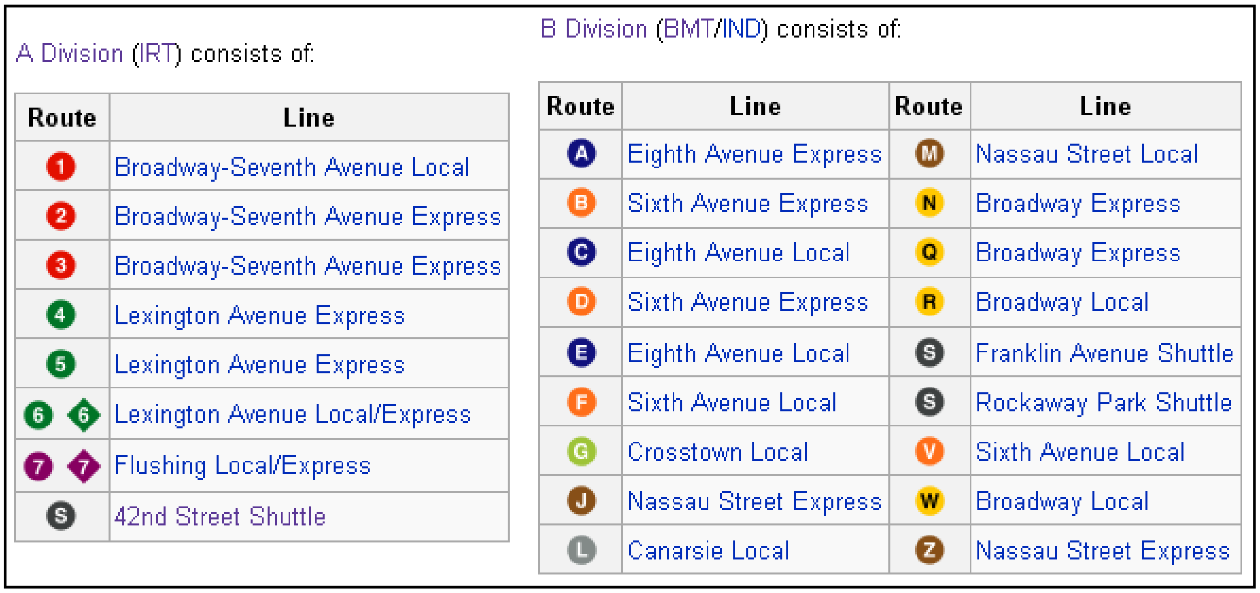
In this part of the research, a case study of my network representation and algorithm will be conducted using NYC subway system. There are several reasons that urge me to make this choice.

First of all, as being introduced in the previous part, NYC subway system is the largest subway system in the United States both in terms of number of annual boarding passengers and total route miles. NYC subway takes a much larger proportion of passengers' daily trip in the city comparing to other cities and metropolitan areas due to the congestion problem and high cost of using private vehicle. Many people living in NYC even do not own a private vehicle. Therefore, their travel patterns by subway are more diverse. Passengers do not only use subway as their major work-home commuting mean, but also other personal activities. Many trips happen within the urban area instead of only from suburbs to urban. Since travels within the urban areas are often relatively short, transfer will take up a large proportion of the travel time. Therefore, avoiding transfer is one of the most important mean to advance the efficiency of short daily trips. The least transfer path will be a very important reference for people to make trips in NYC.

Secondly, the NYC subway's network structure does not follow the spoke-hub paradigm. The chance that transfers would happen is much greater than a hub-spoke network system, such as Washington metro and Chicago subway system. In such systems, transfers are often made at the center hub or a series of central hubs in the middle of the city. Travel patterns are relatively simple in those networks, and transfers often cannot be avoided.

2. Route selection and data preparation

There are 26 routes in the NYC subway network. The 26 routes are operated by two individual companies (Interborough Rapid Transit Company and Brooklyn-Manhattan Transit Corporation). Routes owned by Interborough Rapid Transit Companies consists Division A, while routes owned by Brooklyn-Manhattan Transit Corporation consists Division B. (Figure 22) In this case study, all the routes in Division A (Figure 23) is chosen to do the analysis so that the network representation and algorithm can be examined in a practical application scale, and meanwhile, the amount of computation is controlled within the capability of a personal computer (2.4GHz Intel Core 2 Duo processor, 1066MHz FSB, and 2 GB 1066MHz DDR3 SDRAM).



A Division (IRT) consists of:

Route	Line
1	Broadway-Seventh Avenue Local
2	Broadway-Seventh Avenue Express
3	Broadway-Seventh Avenue Express
4	Lexington Avenue Express
5	Lexington Avenue Express
6 6	Lexington Avenue Local/Express
7 7	Flushing Local/Express
S	42nd Street Shuttle

B Division (BMT/IND) consists of:

Route	Line	Route	Line
A	Eighth Avenue Express	M	Nassau Street Local
B	Sixth Avenue Express	N	Broadway Express
C	Eighth Avenue Local	Q	Broadway Express
D	Sixth Avenue Express	R	Broadway Local
E	Eighth Avenue Local	S	Franklin Avenue Shuttle
F	Sixth Avenue Local	S	Rockaway Park Shuttle
G	Crosstown Local	V	Sixth Avenue Local
J	Nassau Street Express	W	Broadway Local
L	Canarsie Local	Z	Nassau Street Express

Figure 22 Divisions of routes in NYC subway network

(Source: http://en.wikipedia.org/wiki/New_York_City_Subway)

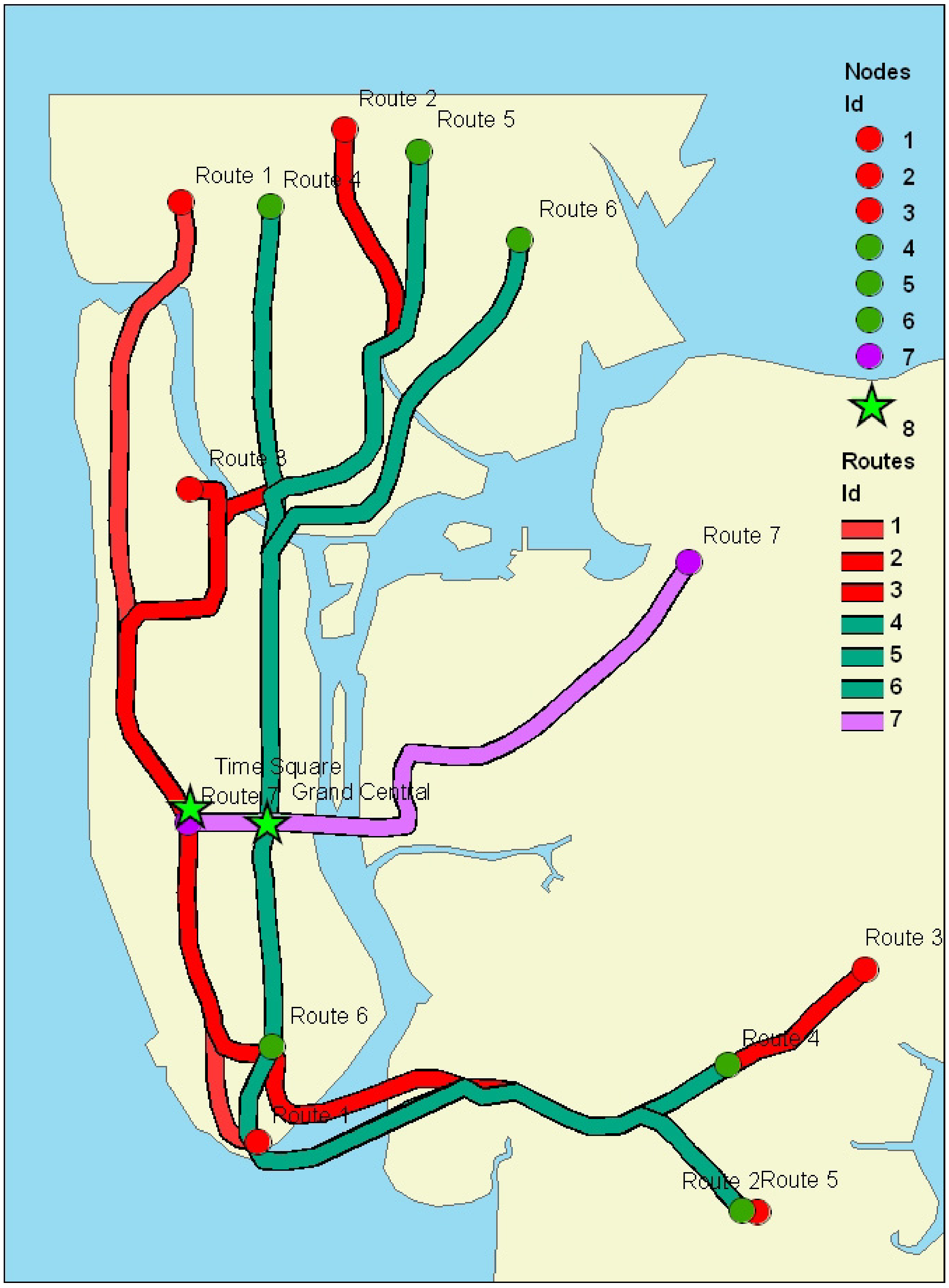


Figure 23 Selected IRT Routes

All the stations involved in the 7 routes are named with a unique ID in a sequential order. The priority is route 1 to route 7, north to south, west to east. However, the sequence of the node IDs can be arranged in any order or even randomly. Stations shared by different routes and nearby stations within a certain range are recognized as one station, giving only one ID. In sum, there are 170 nodes in the network with ID 1 to 170. A table containing the ID number, station name, and passing through routes information is created for reference. Then, 7 connectivity matrices of every route are created. Each matrix is 170 by 170 in size.

3. Program result

Approximately, it took the computer 1 hour and 45 minutes to finish the calculation of least transfer path between each pair of nodes (28730 pairs in total). 4 170 by 170 matrices are created (D^{LTP}, R, NL, and RL), containing all the necessary information of the least transfer paths. Random pairs of nodes are selected, the suggested least transfer paths of which are all correct comparing to online trip planners provided by MTA. After the initial calculation, no more further calculation is needed to find the least transfer path between each pair of nodes. Detailed path information can be queried instantly.

Generally speaking, starting from any station within the division A, passengers can arrive at any other station within 2 transfers. According to Table 1, approximately 25% of the stations can be connected without transfers. 52% percent of stations can be connected with 1 transfer. And the rest of the stations can be connected with 2 transfers.

Table 1. Counts of minimal transfers to connect two random stations in division A

# of transfers	counts	percent
0	7282	25.34633
1	14984	52.15454
2	6464	22.49913
total	28730	100

4. Exploratory Analysis

The 4 matrices created by the application not only can be used to for trip planning for individual passengers, some of them also can shows the characteristics of the transit network on system-wide level which might provide useful information for planning department and transit agencies to construction plans or operation improvements. Figure 24 is the D^{LTP} matrix of NYC subway, division A. Each cell in the matrix representing how many transfers is needed to go from one station to another. Each cell is colored in a way that the warmer the color is, the more transfers is needed. The dark blue diagonal is the group of pairs of same stations. So, those values are meaningless. The yellow strip marked by 1 is the track left by Grand Central station, which means that grand central is the one of few stations on route 4,5,6 that can go to all stations on route 1 with only 1 transfer (from route 7). Similarly, the blue strip marked by 2 means that Time Square station is one of few stations on a 1,2,3 that can go to all the stations on 4,5,6 with only 1 transfer. Generally speaking, abnormality (strips, dark spots) in the matrix suggests this station is the potential major transfer points. The continuous appearances of strips suggest that two or more routes share a section of their routes instead of only intersect at certain points. The area marked by 3 and 4 fits the description. Area 3 is the shared section by route 1,2,3. Area 4 is the shared section by route 4,5,6. On the other hand, the red rectangles in the corner indicate that to transfer from route 1 to most stations in route 4,5,6 need two transfers.

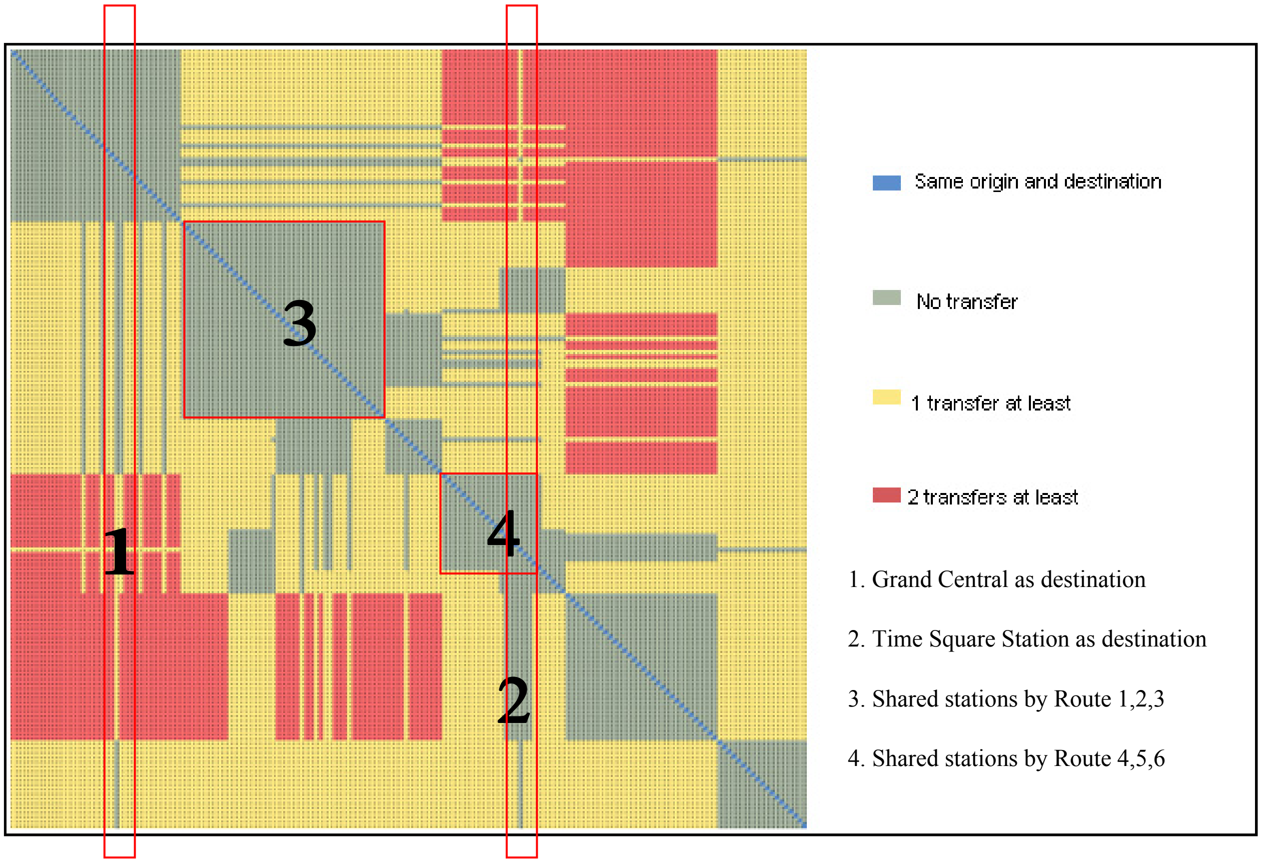


Figure 24 D^{LTP} matrix of NYC Subway division A

The above figure reveals several characteristics of the transit network can be addressed. Firstly, Grand central and Time Square stations have the advantages to serve as transfer points in the network. Secondly, the difficulty to transfer from route 1 to route 4, 5, 6 indicate that it is not easy to access upper east Bronx from upper west Manhattan and west Bronx. Since the division B subway is not included in the case study, there is no ground to criticize or to make planning suggestion to the NYC subway agencies. However, the ease and intuitiveness of D^{LTP} matrix interpretation do provide some information about the network structure in certain perspective.

CONCLUSION

The objective of this study is to design a suitable network representation for a transit network consisted of multiple routes or modes, and to develop a corresponding algorithm which can compute the least transfer path between each pair of nodes in the network. Detailed information about the path should be provided to assist passengers to choose their paths wisely.

The study first summarized a number of existing classical shortest path algorithms. Their strengths and weaknesses of possible applications in the field of public transit network were specially discussed. It is found that the nature of those algorithms and the way they organize network related data might not be very efficient and effective to deal with shortest transfer path problems in public transit networks. Different practices of major transit agencies in attempt of advancing the system-wide efficiency of transit networks were also discussed.

Based on the literature review in the field of classical shortest path problems, this study designed an innovative approach to dealing with the key issue , transfers, which distinguishes this set of problems from other shortest path problems. Several points were specially addressed. First of all, based on the nature of public transit system, a simple and new network representation was introduced. Since transit routes are usually linear or circular in shape, finding a path within the route is hardly a problem. More attention was paid on inter-route trip planning. In the new network representation, nodes are unique and can be shared by different routes. The entangled network is subdivided into different layers with each layer representing one route. This network representation is appropriate because it resembles the nature and structure of public transit network. It also makes further calculation of least transfer paths easier.

Secondly, a progressive least transfer path algorithm was carefully designed. Starting from C matrices to D matrices, D^{LTP}, R, NL, RL matrices are generated sequentially.~~;~~ ach step is simple and efficient. The result matrices not only work well together to reconstruct the least-transfer paths, but also provide abundant extra information to study the structure of the transit system itself.

Several possible applications of the proposed network representation are speculated. First of all, major public transit agencies can develop trip-planning assistance systems based on my network representation and the proposed algorithm. Network matrices and the series of result matrices can be updated periodically. The path information can be provided through web-based applications. The query of path information from passengers would be very efficient since no further calculation is needed. Secondly, further exploratory analysis of the result matrices can be done to reveal the characteristics (advantages and disadvantages) of the network structure. Transit agencies and planning departments can utilize those exploratory analysis based on the result of my proposed method to assist their development and re-development of public transit networks.

However, the network representation and the algorithm are still in their infancies, further research can be done to improve their robustness. For instance, in addition to numbers of linkages, more details information such as physical distance or various types of cost between each pair of nodes can also be incorporated in the method. Transit operation schedules can be considered for time estimation. Furthermore, instead of only giving one least transfer path, some other equally good least transfer paths or even second-best least transfer paths can be included to give passengers more choices, and served as alternative choices when the first least transfer paths are not viable due to possible emergencies or delays.

REFERENCE

Lozano, A., & Storchi G. (2002 December). Shortest viable hyperpath in multimodal networks, *Transportation Research Part B: Methodologic*,*36*(10), 853-874.

Ziliaskopoulos, A. K., & Mahmassan, H. S. (1996). A note on least time path computation considering delays and prohibitions for intersection movements, *Transportation Research B*, 30B (5), 359-367.

Orda, Ariel., & Rom, R. (1990 July). Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length, *Journal of the ACM (JACM*), 37 (3), 607-625.

Laffra C., Dijkstra's algorithm demonstration Applet, Pace University. Retrieved March 27, 2008, from Dijkstra's Shortest Path Algorithm Website: http://www.dgp.toronto.edu/people/JamesStewart/270/9798s/Laffra/DijkstraApplet.html

Malandraki C., & Daskin M.S. (1992). Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms. *Transportation Science 26*(3), 185–200.

Dijkstra's algorithm. (2008, March 23). Retrieved March 27, 2008, from Wikipedia, The Free Encyclopedia Web site : http://en.wikipedia.org/w/index.php?title=Dijkstra%27s_algorithm&oldid=200224965

DIJKSTRA E. W. (1959). A Note on Two Problems in Connexion with Graphs, *Numerische Mathematik 1*, (pp.269-271).

Handler, G. Y., &Zang, I,(1980). Dual Algorithm for the Constrained Shortest Path Problem. *NETWORKS*, *10*(4), 293-310.

Spiess H., & Florian M. (1989). Optimal strategies: A new assignment model for transit networks, *Transportation Research. Part B 23,* 83-102.

Nachtigall, K(1995). Time-Dependent shortest-path problems with applications to railway networks. *European Journal of Operational Research 83*, 154–166.

Knoppers, P., & Muller, T. (1995). Optimized transfer opportunities in public transport Source: *Transportation Science [0041-1655] KNOPPERS 29* (1), p.101.

Knoppers, P. and Muller, T.(1995). Optimized transfer opportunities in public transport. *Transportation Science 29* (1), 101–105.

Narváez P., Siu, K.Y., Tzeng H.Y. (2000 December). New dynamic algorithms for shortest path tree computation. *IEEE/ACM Transactions on Networking (TON)*, *8* (6), 734-746.

Dunphy, T. R., & Kimberly F. (1996). Transportation, Congestion, and Density: New Insights,. *Transportation Research Record*, 1552, 89-96.

Dreyfus, S. E. (1969). An appraisal of some shortest path algorithms, *Oper. Res.*, 17, 395-412.

Nguyen,S., Pallotino, S., & Malucelli F. (1995). A modeling framework for the passenger assignment on a transport network with timetables, (CRT-94-47) Quebec, CA: University of Montreal Press.

Pallottino S., Scutellá M.G., Marcotte, P., Nguyen, S. (1998). Shortest path algorithms in transportation models: Classical and innovative aspects, in: P. Marcotte, S. Nguyen (Ed.), *Equilibrium and Advanced Transportation Modelling* (pp. 245–281). Dordrecht: Kluwer Academic Publishers.

Crainic T., &Rousseau, J.M. (1986). Multicommodity, multimode freight transportation: A general modeling and algorithmic framework for the service network design problem, *Transportation Research, Part B 20,* 225-242.

Ziliaskopoulos, A., & Wardell, W.(2000). Intermodal optimum path algorithm for multimodal networks with dynamic arc travel times and switching delays, *European Journal of Operational Research, 125*, 486–502.

APPENDICES

A. Code of the LTP program

```
' Create index of all the C1 Matrices

Dim c() As Variant

' Create index of all the C_final Matrices

Dim c_c() As Variant

' Create index of all the D Matrices

Dim d() As Variant


' Variables to store the number of routes, and size of the C matrices

Dim num, size As Integer

Dim tempResult() As Integer

Dim DLTP() As Integer

Dim R() As Integer

Dim NL() As Integer

Dim RL() As String


' Subroutine to multiply two matrices

Private Sub Mult(a As Variant, b As Variant)

    Dim size, i, j, k, temp As Integer

    size = UBound(a, 1)

    ReDim tempResult(1 To size, 1 To size)

    For i = 1 To size
```

```
        For j = 1 To size
            temp = 0
            For k = 1 To size
                temp = a(i, k) * b(k, j) + temp
            Next k
            tempResult(i, j) = temp
        Next j
    Next i
End Sub

' subrotine to create a series of matrices containing least transfer path information for each pair of nodes
in the network
Private Sub CmdCal_Click()
    Dim x, y As Integer
    x = Val(TextBox1.Text)
    y = Val(TextBox2.Text)
    Dim temp1, temp2 As Integer
    Dim tmpstring As String

    txtSolution.Text = ""

    If x = y Then
        txtSolution.Text = "Same starting and ending point. Please reinput!!"
    Else
        txtSolution.Text = txtSolution.Text & "At least, " & Str(DLTP(x, y) - 1) & " transfer(s) is/are
needed!" & Chr(13)
```

```
        temp1 = x
        temp2 = x
        While temp2 <> y
           temp2 = R(temp1, y)
           txtSolution.Text = txtSolution.Text & Str(temp1) & "-->" & Str(temp2) & "      Route: " &
RL(temp1, temp2) & "   Stops: " & Str(NL(temp1, temp2)) & Chr(13)
           temp1 = temp2
        Wend

    End If

End Sub

' click display the help window
Private Sub cmdhelp_Click()
    Form2.Show
End Sub

' Calculate DLTP, R, NL, RL matrices from the loaded C matrices
Private Sub cmdLTP_Click()
    Form1.txtResult.Text = ""

    'variables to control loops
    Dim i, j, k, l, m, x, y As Long
    Dim temp As Long
    Dim tmpSize() As Variant
    Dim tmpMatrix() As Long

    Dim tmpMatrix2() As Long

    ReDim c_c(1 To num) As Variant

    ReDim d(1 To num) As Variant


'Calculate D matrices and display

    For i = 1 To num

        ReDim tmpSize(1 To size) As Variant

        c_c(i) = tmpSize

        For j = 1 To size

            ReDim tmpMatrix(1 To size, 1 To size)

            c_c(i)(j) = tmpMatrix

        Next j

    Next i


    For i = 1 To num

        c_c(i)(1) = c(i)

    Next i


    For i = 1 To num

        For j = 2 To size

            For k = 1 To size

                For l = 1 To size

                    temp = 0

                    For m = 1 To size

                        'temp = c_c(i)(1)(k, m) * c_c(i)(j - 1)(m, l) + temp
                If (c_c(i)(1)(k, m) > 0 And c_c(i)(j - 1)(m, l) > 0) Then
                    temp = 1
                End If
            Next m
            c_c(i)(j)(k, l) = temp
        Next l
    Next k
  Next j
Next i

For i = 1 To num
  ReDim tmpMatrix(1 To size, 1 To size)
  d(i) = tmpMatrix
Next i

For i = 1 To num
  For j = 1 To size
    For k = 1 To size
        For l = 1 To size
            If ((c_c(i)(j)(k, l) > 0) And (Not (k = l Or d(i)(k, l) > 0))) Then
                d(i)(k, l) = j
            End If
        Next l
    Next k
  Next j
Next i
```

```
'Calculate DLTP Matrix and Display
    ReDim DLTP(1 To size, 1 To size)
    ReDim tmpMatrix(1 To size, 1 To size)
    Dim tmpNode As Long
    Dim tmpRoute As Long
    Dim tmpLink As Long
    Dim tmpTotalLink As Long
    Dim tmpstring As String
    Dim temp1 As Long
    Dim temp2 As Long

    ReDim R(1 To size, 1 To size) As Integer
    ReDim NL(1 To size, 1 To size) As Integer
    ReDim RL(1 To size, 1 To size) As String

'no transfer

    For j = 1 To size
        For k = 1 To size
            tmpLink = 9999
            tmpRoute = 0
            For i = 1 To num
                If (d(i)(j, k) > 0 And (j <> k)) Then
                    DLTP(j, k) = 1
                    If d(i)(j, k) < tmpLink Then
                        tmpLink = d(i)(j, k)
                        tmpRoute = i
                    End If
                End If
            Next i
            If tmpLink <> 9999 Then
                NL(j, k) = tmpLink
                R(j, k) = k
                RL(j, k) = CStr(tmpRoute)
            End If
        Next k
    Next j

'more transfers
    ReDim tmpMatrix(1 To size, 1 To size)
    For x = 1 To size
        For y = 1 To size
            tmpMatrix(x, y) = DLTP(x, y)
        Next y
    Next x

    For i = 2 To num
        Call Mult(tmpMatrix, DLTP)
        For x = 1 To size
            For y = 1 To size
                If (DLTP(x, y) = 0 And tempResult(x, y) > 0 And (Not x = y)) Then
                    DLTP(x, y) = i
```

```
            End If
        Next y
    Next x
Next i

For x = 1 To size
    For y = 1 To size
        tmpMatrix(x, y) = NL(x, y)
    Next y
Next x

For i = 2 To num
    For x = 1 To size
        For y = 1 To size

            tmpLink = 9999
            tmpNode = 0
            For k = 1 To size
                If (tmpMatrix(x, k) > 0 And NL(k, y) > 0 And tmpLink > NL(x, k) + NL(k, y)) Then
                    tmpLink = NL(x, k) + NL(k, y)
                    tmpNode = k
                End If
            Next k
            If (R(x, y) = 0 And x <> y) Then
                R(x, y) = tmpNode
                NL(x, y) = tmpLink
            End If
         Next y
      Next x
   Next i

   For x = 1 To size
      For y = 1 To size
         If (RL(x, y) = "" And x <> y) Then
            temp1 = x
            temp2 = x
            tmpstring = ""
            While (temp2 <> y)
               temp2 = R(temp1, y)
               tmpstring = tmpstring & CStr(RL(temp1, temp2))
               temp1 = temp2
            Wend
            RL(x, y) = tmpstring
         End If
      Next y
   Next x

   For x = 1 To size
      RL(x, x) = "/"
   Next x
End Sub
```

```
'save the least transfer path results to a txt file
Private Sub cmdSave_Click()
    Dim fileSysObj As New FileSystemObject
    Dim txtStream As TextStream
    Dim tmpLine() As String
    Dim tmpMatrix() As Integer
    Dim i, j, k As Integer

    Set fileSysObj = New FileSystemObject
    Set txtStream = fileSysObj.OpenTextFile(Form1.txtSavePath.Text, ForWriting, True)
    For i = 1 To num
        txtStream.WriteLine "D matrix of Route " & Str(i)
        For j = 1 To size
            For k = 1 To size
                If (k <> size) Then
                    txtStream.Write Str(d(i)(j, k)) & " "
                Else
                    txtStream.Write Str(d(i)(j, k))
                End If
            Next k
            txtStream.WriteLine
        Next j
    Next i

    txtStream.WriteLine "Final DLTP matrix"
    For j = 1 To size
    For k = 1 To size
        If (k <> size) Then
            txtStream.Write Str(DLTP(j, k)) & " "
        Else
            txtStream.Write Str(DLTP(j, k))
        End If
    Next k
    txtStream.WriteLine
Next j

txtStream.WriteLine "Final R matrix"
For j = 1 To size
    For k = 1 To size
        If (k <> size) Then
            txtStream.Write Str(R(j, k)) & " "
        Else
            txtStream.Write Str(R(j, k))
        End If
    Next k
    txtStream.WriteLine
Next j

txtStream.WriteLine "Final NL m atrix"

For j = 1 To size
    For k = 1 To size
```

```
            If (k <> size) Then
                txtStream.Write Str(NL(j, k)) & " "
            Else
                txtStream.Write Str(NL(j, k))
            End If
        Next k
        txtStream.WriteLine
    Next j

    txtStream.WriteLine "Final RL matrix"

    For j = 1 To size
        For k = 1 To size
            If (k <> size) Then
                txtStream.Write RL(j, k) & " "
            Else
                txtStream.Write RL(j, k)
            End If
        Next k
        txtStream.WriteLine
    Next j
    txtStream.Close

End Sub

' initialize the graphic user interface
```

```
Private Sub UserForm_Initialize()

    Form1.txtPath.Text = "C:\FinalThesis\matrices.txt"

    Form1.txtSavePath.Text = "C:\FinalThesis\LTP_matrices.txt"

    Form1.TxtDisplay.MultiLine = True

    Form1.TxtDisplay.ScrollBars = fmScrollBarsBoth

    Form1.Label4.Caption = "                                        Please read the HELP file before you start
using this program!"

End Sub


Private Sub CmdLoadMatrix_Click()

    Form1.TxtDisplay.Text = ""

    Dim fileSysObj As New FileSystemObject

    Dim txtStream As TextStream

    Dim tmpLine() As String

    Dim tmpMatrix() As Integer

    Dim i, j, k As Integer


    Set fileSysObj = New FileSystemObject

    Set txtStream = fileSysObj.OpenTextFile(Form1.txtPath.Text)

    tmpLine() = Split(txtStream.ReadLine, " ")

    num = Val(tmpLine(0))

    size = Val(tmpLine(1))

    Form1.Label1.Caption = "Number of routes:" & Str(num) & "     " & "Nodes involved: " & Str(size)


    ReDim c(1 To num) As Variant

    For i = 1 To num
```

```
        ReDim tmpMatrix(1 To size, 1 To size)
        'Form1.TxtDisplay.Text = Form1.TxtDisplay.Text & "C matrix of route " & Str(i) & vbCrLf
        txtStream.ReadLine
        For j = 1 To size
            tmpLine() = Split(txtStream.ReadLine, " ")
            For k = 1 To size
                tmpMatrix(j, k) = Val(tmpLine(k - 1))
       '         Form1.TxtDisplay.Text = Form1.TxtDisplay.Text & Str(tmpMatrix(j, k)) & " "
            Next k
        '    Form1.TxtDisplay.Text = Form1.TxtDisplay.Text & vbCrLf
        Next j
        c(i) = tmpMatrix
    Next i
    txtStream.Close
    MsgBox ("load matrices done!")
End Sub


' quit the pplication
Private Sub CmdQuit_Click()
    Unload Form1
    Form1.TxtDisplay.Text = ""
End Sub
```