

國立成功大學
資訊管理研究所
碩士論文

大眾運輸路網中最短時間及最少旅費之行程規劃研究

Optimal Paths Based on Time and Fares in Transit Networks

指導教授：王逸琳 博士

研究生：王建傑

中華民國九十七年八月

國立成功大學
碩士論文

大眾運輸路網中
最短時間及最少旅費之行程規劃研究
Optimal Paths Based on Time and Fares
in Transit Networks

研究生：王建傑

本論文業經審查及口試合格特此證明

論文考試委員： 王逸琳
李宇欣
林美貴
李耀玲
洪一蓮

指導教授：王逸琳

系(所)主管：李賢得

中華民國 97 年 5 月 30 日

大眾運輸路網中最短時間及最少旅費之行程規劃研究

國立成功大學資訊管理研究所碩士班

摘 要

在現代化的都會區中，大眾運輸是日常生活中不可或缺的工具。過去大部分乘客都利用時刻表手冊等資訊來規劃其行程，然而隨著交通逐漸繁忙，大眾運輸路網不斷擴建，現今的大眾運輸路網業已十分複雜，往往需要個人導航與行程規劃系統協助方能有效使用；然而，市面上的導航與行程規劃系統多為汽車駕駛者所量身打造，相較之下可提供大眾運輸乘客導航與行程規劃服務之工具幾乎屈指可數；由於大眾運輸路網具有路線固定、規律的營運時間、以及非線性的收費標準等等諸多特殊性，導致原來適用於一般道路路網之導航或行程規劃系統不再適用，因此發展適用於大眾運輸乘客便捷實用的行程規劃系統誠為當務之急。

在使用者輸入起訖點及其出發時間後，本論文提出數個數學模型與演算法以在具有時刻表的大眾運輸路網中規劃最短旅行時間以及最少旅費之行程。在規劃最短旅行時間行程方面，我們首先僅針對搭乘公車、捷運等大眾運輸工具之行程進行規劃，接著再進一步將步行亦列入行程考量，並提出加速方法縮小路網規模以提升行程規劃效率。在最少旅費之行程規劃方面，本論文提出數學模型與演算法來處理其非線性的旅費結構，並針對大台北地區大眾運輸路網的旅費特性提出較簡化的特殊展開網路，以更有效率的方式求解包括轉乘優惠、兩段票等情況之最少旅費行程規劃。

關鍵字：最短路徑、大眾運輸路網、行程規劃、時刻表、旅費

ABSTRACT

Optimal Paths Based on Time and Fares in Transit Networks

Michael Wang

In a public transportation oriented metropolis such as Tokyo or Taipei, transit system is a convenient means for personal trip. Historically, most transit passengers rely on printed schedules for trip planning. However, the transit network nowadays has become too complicated to navigate manually. With the advance of technology, various navigation applications have been developed for guiding private vehicles, but few are designed for public transportation. Given an origin, destination, and intended departure time, this study proposes two timetable-based algorithms to search for optimal itineraries so that the total travel time for an individual is minimized in an transit network. Itineraries showing the suggested routes with walking access and egress, bus stops, and Mass Rapid Transit (MRT) information will be generated, considering the time to wait for, to transfer between, and to stay in transit vehicles, as well as the time to walk between transit stations. In addition, this study proposes an innovative fare-based algorithm to search for the cheapest itineraries with non-linear fare structure, and an alternative fare model is specifically developed for Taipei transit system. Optimizing trip planning according to time or fare meets the common practices of passengers using transit system in a metropolitan area.

Keywords: shortest path, transit network, multimodal transportation, trip planning, timetable-based algorithm, non-linear fare structure.

ACKNOWLEDGEMENTS

本論文得以付梓，首先以及最重要的必須感謝指導教授王逸琳老師給予的細心指導，使我跨領域卻不至於惶恐。論文提案期間承蒙王泰裕老師以及翁慈宗老師對本論文提供研究方向的建議。論文口試期間則承蒙李宇欣老師、林義貴老師、黃耀廷老師以及洪一薰老師對本論文提供寶貴的意見，使本論文更加完善。謹此表達我的敬意與謝忱。

在這兩年研究所求學生涯中，我要感謝林修杰學長、戴群達學長、陳姿君學姊以及楊橙坤學長的提攜與指導；同屆的陳正楠同學、馬家宜同學、以及劉姿儀同學一起在研究室奮鬥，使得研究室充滿活力與歡笑；以及李俊賢學弟與陳志偉學弟與其他學弟妹在論文口試期間參與投影片的製作並在論文寫作期間參與文稿的審核；感謝你們使我的研究所生活多采多姿。此外，我還要特別感謝我協助指導的專題學生們—陳雅婷、張家媛、康雅淳、呂明倩—有她們的幫助，使我的論文更完善。

最後，謹將完成這份論文以及碩士學位的榮耀與喜悅獻給我的父母及家人，感謝他們長期以來的支持與包容，使我能更有自信的迎接未來的挑戰，謝謝。

Contents

List of Tables	v
List of Figures	viii
Chapter 1. INTRODUCTION	1
1.1. Background and Motivation	1
1.2. Problem Definition and Methodology	4
1.3. Database and Database Management	5
1.4. Structure of Thesis	7
Chapter 2. LITERATURE REVIEWS	8
2.1. Trip Planning Systems	8
2.2. Shortest Path Algorithms	11
2.3. Timetable Information Problems	12
2.4. Multimodal Transit Problems	16
2.5. Fare Problems	17
Chapter 3. METHODOLOGIES ON A BASIC TIMETABLE INFORMATION PROBLEM	19
3.1. Spatial Data and Temporal Data	21
3.2. Construction of a Basic Timetable-based Network	23
3.3. Query and Solution Method on the Basic Timetable-based Network	25
3.3.1. Procedures	25
3.3.2. An Illustrative Example	28
3.4. Computational Experiments	29
3.4.1. Settings and Problem Sets for the Implementation	29

3.4.2. Algorithmic Running Time Comparison	30
3.5. Speed-up Techniques	32
3.6. Summary	34

Chapter 4. METHODOLOGIES ON A MULTIMODAL TIMETABLE INFORMATION PROBLEM WITH WALKING TRANSFER	36
4.1. Spatial Data and Temporal Data	37
4.2. Construction of a Multimodal Timetable-based Network	41
4.3. Query and Solution Method on a Multimodal Timetable-based Network	44
4.3.1. Procedures	44
4.3.2. An Illustrative Example	46
4.4. Computational Experiments	47
4.4.1. Settings and Problem Sets for the Implementation	48
4.4.2. Algorithmic Running Time Comparison	49
4.5. Speed-up Techniques	50
4.6. Summary	53

Chapter 5. METHODOLOGIES ON FARE INFORMATION PROBLEMS	55
5.1. Spatial Data and Fare Data	56
5.2. Construction of a Fare-based Network	58
5.3. Query and Solution Method on Fare-based Networks	59
5.3.1. Procedures	60
5.3.2. An Illustrative Example	60
5.4. Computational Experiments	61
5.4.1. Settings and Problem Sets for the Implementation	62
5.4.2. Algorithmic Running Time Comparison	63

5.5.	Alternative Fare Models	65
5.5.1.	Fixed Fare Rate and Variable Fare Rate	65
5.5.2.	Transfer Discount	67
5.6.	Computational Experiments on Alternative Fare Models	68
5.6.1.	Settings and Problem Sets for the Implementation	69
5.6.2.	Algorithmic Running Time Comparison	69
5.7.	Summary	71
Chapter 6.	CONCLUSIONS AND FUTURE RESEARCH	72
6.1.	Summary and Contributions	72
6.2.	Applications on Different Platforms	74
6.2.1.	Platforms	74
6.2.2.	Shortest Path Algorithms	76
6.3.	Future Research	78
6.3.1.	Multiobjective Shortest Path Problem	78
6.3.2.	Dynamic Information	79
6.3.3.	Geographic Information System	80
	References	81
Appendix A.	COMPUTATIONAL EXPERIMENTS ON BASIC TIMETABLE INFORMATION PROBLEMS	86
Appendix B.	COMPUTATIONAL EXPERIMENTS ON MULTIMODAL TIMETABLE INFORMATION PROBLEMS WITH WALKING TRANSFER	88
Appendix C.	COMPUTATIONAL EXPERIMENTS ON FARE INFORMATION PROBLEMS	90

Appendix D. **COMPUTATIONAL EXPERIMENTS ON
FARE INFORMATION PROBLEMS WITH
ALTERNATIVE FARE MODEL**

92

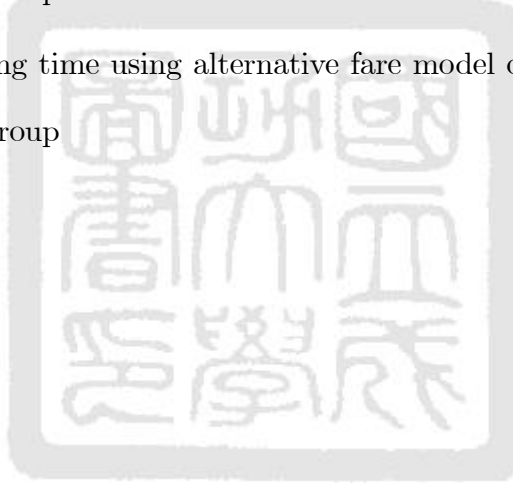


List of Tables

2.1 Some implementations on Dijkstra's algorithm	12
3.1 Nomenclature for the basic timetable information problem	20
3.2 Line route information	22
3.3 Timetable information for each route	23
3.4 Nodes information of basic timetable-based network	29
3.5 Time to construct basic timetable-based networks	31
3.6 Relative performance on five problem sets with randomly selected OD pairs	31
3.7 Relative performance on problem set 5	34
3.8 Normalization of relative performance on problem set 5	34
4.1 Nomenclature for the multimodal timetable information problem with walking transfer	38
4.2 Line route information	39
4.3 Walk distance information	40
4.4 Timetable information for each route	41
4.5 Walking time for artificial arcs	47
4.6 Nodes information of multimodal timetable-based network	47
4.7 Time to construct multimodal timetable-based network	49
4.8 Relative performance on five problem sets with randomly selected OD pairs	50
4.9 Relative performance on problem set 5	52

4.1	Normalization of relative performance on problem set 5	53
5.1	Nomenclature for fare information problem	56
5.2	Fare data	58
5.3	Node information of illustrative example	61
5.4	Fare data of illustrative example	62
5.5	Time to construct fare-based network	64
5.6	Average running time on problem set 5 for different OD group	64
5.7	Average running time on five problem set for randomly selected OD group	65
5.8	Average running time using alternative fare model on problem set 5 for different OD group	70
5.9	Average running time using alternative fare model on five problem set for randomly selected OD group	70
6.1	Comparison between single-source shortest path and all-pairs shortest path algorithms for timetable problems	77
6.2	Comparison between single-source shortest path and all-pairs shortest path algorithms for fare problem	77
A.1	Relative performance on problem set 1	86
A.2	Relative performance on problem set 2	86
A.3	Relative performance on problem set 3	87
A.4	Relative performance on problem set 4	87
B.1	Relative performance on problem set 1	88
B.2	Relative performance on problem set 2	88
B.3	Relative performance on problem set 3	89
B.4	Relative performance on problem set 4	89

C.1Average running time on problem set 1 for different OD group	90
C.2Average running time on problem set 2 for different OD group	90
C.3Average running time on problem set 3 for different OD group	91
C.4Average running time on problem set 4 for different OD group	91
D.1Average running time using alternative fare model on problem set 1 for different OD group	92
D.2Average running time using alternative fare model on problem set 2 for different OD group	92
D.3Average running time using alternative fare model on problem set 3 for different OD group	93
D.4Average running time using alternative fare model on problem set 4 for different OD group	93



List of Figures

1.1 Subway systems of Osaka City, Japan	2
1.2 An itinerary generated by MBTA	3
2.1 Illustration of TATIS	10
3.1 A simple transit network	21
3.2 A timetable-based transportation network	28
4.1 A multimodal transit network and its hierarchical display	39
4.2 An illustration of walk-transfer arc	43
4.3 A multimodal timetable-based transit network	46
5.1 A hierarchical representation of a multimodal transit network	57
5.2 A multimodal transit network	58
5.3 A fare-based multimodal transit network	61
5.4 Network construction for variable fare rate in alternative fare model	66
5.5 Network construction for fixed fare rate in alternative fare model	67
5.6 Modification on arc cost of transfer arc connecting MRT to bus	68

CHAPTER 1

INTRODUCTION

With the rapid growth of economy since industrialization, traffic congestion problem has become one of the most pressing issues of modern era. Public transportation is regarded as an efficient method to alleviate traffic jam. The key measure is to induce people to choose the public traffic vehicles but not their own cars for personal transit. As Lam et al. [23] pointed out, the commonly adopted promotion is to provide people with more contented riding conditions and quality services. In this chapter, we will first introduce the background and motivation of our study, and then we will define our problem and methodology. In addition, we will discuss the database and database management system for our study, and describe the structure of the thesis in the end.

1.1. Background and Motivation

As public transportation systems become more and more complicated along with urban expansion, public transportation users need more thorough information to help them plan journeys efficiently. Since a metropolitan transportation network usually involves a lot of bus or MRT (Mass Rapid Transit System) routes (see Figure 1.1 for an illustration of subway system), there usually exists more than one itinerary to connect any given origin-destination (OD) pair of locations at any given time.

As a result, passengers may encounter several perplexities while conducting their trips in a metropolitan area: First, an itinerary may not be easily identified without the help of a good trip planning information system. The situation becomes even worse for visitors new to the area. Second, even if several itineraries have been provided for guidance, different itineraries may have different profiles

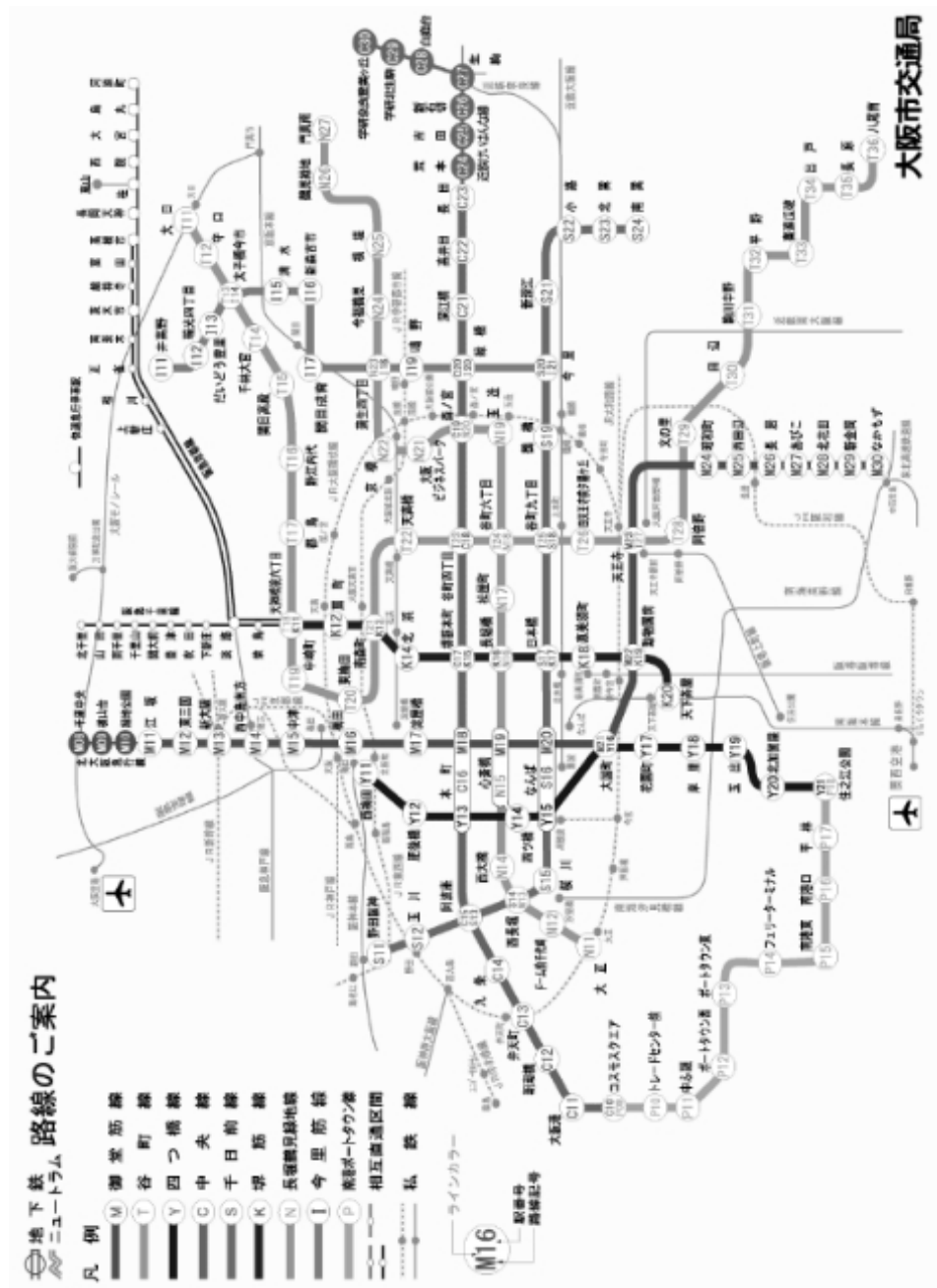


Figure 1.1. Subway systems of Osaka City, Japan

in the traveling distance, duration, and cost which make it more difficult for a passenger to select the most suitable itinerary (see Figure 1.2 for an example of multiple itineraries generated by the trip planning system of Massachusetts Bay Transportation Authority, abbreviated MBTA). Besides, the itinerary profiles may change over time due to the traffic uncertainty or occurrence of holidays. For instance, in Taipei, the problem is not always about availability of services, but

Start: Bu Central Station

End: Trinity Church


Depart at PM on 10/14/2007

Minimize Transfers and use all services

with a walking distance of 1/2 mile


☐ Trip must be accessible

Itinerary 1 - Approx. 18 mins.

 Take Green- B Line - Government Ctr To Copley Station

Approx. 3:23 PM Depart from Boston Univ Central - Inbound


Approx. 3:37 PM Arrive at Copley Station - Inbound

 Walk For 4 Mins. To Trinity Church

Walk approx. 1 block E on Boylston St.


Cost:	
Regular fare	Senior/Disabled fare
\$2.00	\$0.60

Itinerary 2 - Approx. 18 mins.

 Take Green- B Line - Government Ctr To Copley Station

Approx. 3:32 PM Depart from Boston Univ Central - Inbound

Approx. 3:46 PM Arrive at Copley Station - Inbound

 Walk For 4 Mins. To Trinity Church

Walk approx. 1 block E on Boylston St.

Cost:	
Regular fare	Senior/Disabled fare
\$2.00	\$0.60

Figure 1.2. An itinerary generated by MBTA

sometimes about finding a suitable travel plan for specific personal trip preference. Generally speaking, trip preferences that are entitled to determine the best path include travel time, access and egress time, waiting time, walking distance, the number of transfers, fares, level of comfort, etc. Among those trip preferences, the least travel time and the lowest travel fare are the most common objectives, and will consequently be the main themes of this thesis.

1.2. Problem Definition and Methodology

After reviewing previous works on public transportation trip planning, our research focuses on two objectives: finding the shortest travel time through a uniquely designed public transportation network structure in which timetable coordination is an inherent feature, and generating an itinerary of lowest fare for a trip on a transit network.

In respect of the first objective, for each stop in each bus or MRT route, suppose the timetable that records the exact schedule of each arrival and departure time is given, we are interested in devising techniques for generating an itinerary that allows a passenger to travel in minimal time, given the origin, destination, and the starting time of one's trip. For the basic problem where walking is not a means of transfer, a preprocessing algorithm is proposed to construct a specialized time-space network, so that the conventional shortest path algorithms can be directly applied to generate the itinerary of minimum travel time. Furthermore, since the timetable-based transportation network is an acyclic diagram, topological ordering algorithm, different from the variants of Dijkstra's algorithm, can be applied to find the quickest path more efficiently.

We then discuss a more complex situation: the passengers are allowed to walk from one node (bus or MRT stop) to another. To solve this problem, a hierarchical timetable-based network is introduced. In the hierarchical timetable-based network, every unique transit route is a layer itself with transfer arcs connecting each layer while the walking arcs do not exceed the designated distance. After connecting artificial links between transit stations and origin and destination, Dijkstra's algorithm and topological ordering algorithm can be applied to obtain the quickest path.

A functional transit itinerary planner should serve both computers and mobile devices. However, the natures of mobile devices are so distinct from computers that we need to develop solution techniques for them respectively. For computers,

such as application servers, a transit itinerary planner may be required to handle multiple requests at the same time. The accessibility and concurrency control of database are relatively critical. On the other hand, a transit itinerary planner may only serve one person on a mobile device while the battery life and monitor size are limited. In this situation, computational efficiency and display screen are relatively vital. The focal point of this thesis is to improve computational efficiency so that our transit itinerary planner can perform better on mobile devices. Based on this axiom, we developed two speed-up techniques to improve computational efficiency in the timetable information problems.

As for the second objective, two different models will be applied on the lowest-fare problem. For each unique transit route, we assume that the travel cost between every OD pair is known. After links between different routes, links between pseudo origin and transit stations, and links between pseudo destination and transit stations are connected, a fare-based transportation network is thus constructed. However, we use two different approaches to assign costs to arcs. The first approach constructs arcs so that every unique transit route is a complete graph itself to represent cost between every transit station. There will be no cost assigned to transfer and artificial arcs. As for the second approach, we assign different arc costs to transit routes with fixed fare rate and variable fare rate. In this model, in-route arcs only connect adjacent nodes to reduce the number of arcs. Some rules, such as the first transfer to bus from MRT is free, could also be imposed through the second approach. After connecting the arcs and assigning the costs, common shortest path algorithm, such as Dijkstra's algorithm, can be applied to obtain the cheapest path.

1.3. Database and Database Management

The amount of information on transit network available to us is massive. To get the most out of these large and complex datasets, tools that simplify the tasks

of managing the data and extracting useful information in a timely fashion are required. Otherwise, data can become a liability, with the cost of acquiring it and managing it far exceeding the value derived from it.

A database is a collection of data, typically describing the activities of one or more related organizations. A common database might contain information about entities and relationships between entities. A database management system, or DBMS, is software designed to assist in maintaining and utilizing large collections of data. The need for such systems, as well as their use, is essential nowadays. Many kinds of database management systems are in use, but our study applies relational database system (RDBMS).

From the earliest days of computers, storing and manipulating data have been a major application focus. In the late 1980s and the 1990s, advances were made in many areas of database systems. Considerable research was carried out into more powerful query languages and richer data models, with emphasis placed on supporting complex analysis of data. The data is mostly stored in a relational DBMS and the application layer can be customized to different users. Most significant, perhaps, DBMSs have entered the Internet Age. While the first generation of web-sites stored their data exclusively in operating systems files, the use of a DBMS to store data accessed through a Web browser is becoming widespread. Queries are generated through Web-accessible forms and answers are formatted using a markup language such as Hypertext Markup Language (HTML) to be easily displayed in a browser. Database developers are also adding features to their DBMS aimed at making it more suitable for deployment over mobile devices. With the emergence of database, we are now in a much friendlier environment for developing software dealing with large amount of data.

Throughout our research, we use Microsoft Access 2007 to exchange data with transit network databases on MySQL through Open Database Connectivity (ODBC).

1.4. Structure of Thesis

This thesis is organized as follows: Chapter 2 reviews fundamentals regarding algorithms, and previous works on timetable-based and fare-based public transportation trip planning; Chapter 3 defines the basic timetable-based network structures and gives solution methods to find the quickest path based on a query; Chapter 4 defines the multimodal timetable-based network structures that allow walking between nodes and gives solution methods to find the optimal choice; Chapter 5 defines the fare-based network structure and gives solution methods to find the cheapest path; Chapter 6 summarizes our thesis and contribution, and then points out some directions for future research.



CHAPTER 2

LITERATURE REVIEWS

In order to minimize travel time and reduce traffic congestion, Hall [16] pointed out that Multimodal Advanced Traveler Information System (MATIS) was developed to provide travelers with better information. Transit Advanced Traveler Information System (TATIS), part of MATIS, offers features that aid travelers to plan their journey. Koncz et al. [21] mentioned that TATIS has also been referred to by other names, such as Transit Information System (TIS), Passenger Information System (PIS), and Advanced Public Transportation System (APTS). However, TATIS was not initially designed to accommodate individual user preference but to provide the general public with static travel information. Therefore, in order to make transit itinerary planner applicable on platforms such as Internet or personal digital assistance (PDA), we need to design different computational techniques, algorithms, and itinerary display. In this chapter, we will first glance through the development of trip planning system, and then review different aspects of user preferences in transit trip planning.

2.1. Trip Planning Systems

In a highly urbanized society, such as Taipei, accurate and timely travel information can help travelers reach their destinations quickly and safely. To serve this need, Mouskos and Greenfeld [32] indicated that MATIS uses computers to provide pre-trip and en route travel information to help travelers choose the safest and most timesaving path. The history of using computers to provide transit information could be traced back to 1970s according to Cathey and Dailey [3]. Initially, MATIS was designed to assist drivers or passengers making pre-trip travel planning by learning of traffic condition to avoid congestion or constructions. However,

with the breakthrough in computer science and communication technology, the application of geographic information system (GIS) on transit trip planning was first introduced by Koncz and Greenfeld [22], and later expanded by Peng and Huang [38]. With the rapid progress in accessibility, Yang and Huang [54] suggested that transit information not only benefits travelers, but also improves the utilization of transit system. As Levinson [25] inferred, reducing travelers' exposure to congestion and confusion reduces their anxiety, and allows efficient calculation of routes.

Wu et al. [52] reckoned that Internet is one of the emerging technologies that can serve as an interactive platform for MATIS systems. Some trip planning systems have already been implemented over the Internet, such as MBTA (<http://www.mbtta.com/>), or Osaka's kotsu (<http://www.kotsu.city.osaka.jp/>). In addition to Internet, Chen et al. [5] mentioned that wireless communication technologies have made travel information more accessible to travelers on the go. Especially, in the new era of the third generation of mobile phones standards and technologies (3G), network operators could offer users a wider range of more advanced services while achieving greater network capacity. and spectrum efficiency. This improves the delivery of travel information to mobile users. For example, MBTA offers its trip planning services on PDA to subscribers of 3G network. In addition, wireless local area network (WLAN) technologies, such as 802.11g, bring easily-accessible wireless broadband access to users. Wu et al. [52] proposed that a large-scale deployment of WLAN over campus or community can serve as a channel for providing trip planning service to travelers. These emerging technologies would undoubtedly enhance future development of MATIS.

In order to help travelers utilize the massive and complicated transit network, TATIS was thus developed. TATIS is one of the facets of Advanced Traveler Information System (ATIS), and itinerary planning is one of the principal components of TATIS. In Yin et al. [55] and Taniguchi and Shimamoto [45], the purpose

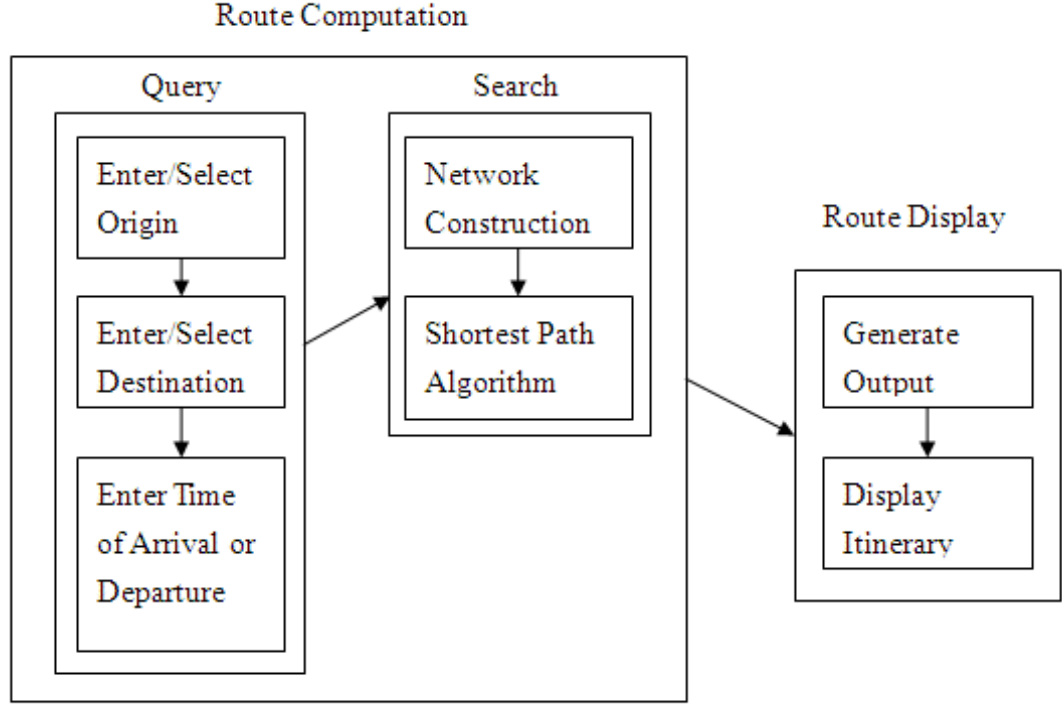


Figure 2.1. Illustration of TATIS [38]

of itinerary planning is to assist travelers in choosing the optimal path to their destinations in terms of travel distance, travel time, or other criteria. However, Rehr et al. [40] argued that TATIS lacks the ability to accommodate personalized options, such as preferred arrival or departure time. In the thesis, we will propose several methods to generate itineraries that meet user specifications. Illustrated in Figure 2.1, itinerary planning incorporates two major elements: route computation and route display. The goal of route computation is to find a connected sequence of transit route segments from a user-defined location to a destination. Route computation may be based on criteria such as the shortest travel distance, quickest travel time, or lowest fare specified by the users. According to Caulfield and O'Mahony [4], the goal of route display is to effectively present the optimal route to the traveler for guidance. Shortest path algorithms are the essence of route computation. We will review some shortest path algorithms in section 2.2.

2.2. Shortest Path Algorithms

In this section, we will introduce the basic concepts of shortest path algorithms that will be needed later throughout the thesis.

Shortest path algorithms are central to most network and transportation problems. In Zhan and Noon [56], the application of fifteen different algorithms on real road networks indicated that it is worthwhile to apply Dijkstra's algorithm to solve one-to-one or one-to-some shortest path problems. Dijkstra's algorithm, the classical single-source shortest path algorithm, is a greedy algorithm that solves the single-source shortest path problem for a weighted, directed graph where all edge weights are nonnegative. In latter section, we will see that most of the timetable-related problems could be solved by Dijkstra's algorithms or its variants.

The algorithm works by maintaining a distance label $d(i)$, which is an upper bound on the length of shortest path to each node i , and the states of *temporarily labeled* and *permanently labeled* with each node. Initially, the distance label is zero for the source node s , and infinity for all other nodes, representing the fact that we do not know any path leading to those nodes. When the algorithm terminates, the distance label will be the cost of the shortest path from the source node s to the destination node t , or infinity if no path exists between s and t .

The algorithm maintains two sets of nodes, S and Q . Set S contains all nodes whose distance labels have been permanently labeled, and set Q contains all other nodes. Set S is initially empty, and in each step, the node with the smallest distance label is moved from Q to S . As a node u is moved into S , the algorithm relaxes every outgoing arc from u to see if any improvement could be made on the shortest known path to v by first following the shortest path from the source to u , and then traversing the arc (u, v) . If the distance label of node u plus the arc length of the arc from u to v is smaller than the distance label of v , the algorithm updates the distance label $d(v)$ with the new smaller cost.

Table 2.1. Some implementations on Dijkstra's algorithm

Abbreviation	Implementation	Complexity	Reference
DIKQ	Naive implementation	$O(V ^2)$	[10]
	<i>Using buckets structure</i>		
DIKB	Basic implementation	$O(E + V C)$	[9]
	<i>Using heap structure</i>		
DIKF	Fibonacci heap	$O(E + V \log V)$	[11]
DIKH	k -array heap	$O(E \log V)$	[8]
DIKR	R -heap	$O(E + V \log(C))$	[2]

At termination, when $Q = \emptyset$, the shortest path, if one exist, is found. The running time of Dijkstra's algorithm depends on how the priority queue is implemented. There are various implementations with Dijkstra's algorithm such as Dial's implantation of buckets structure and implementation of Fibonacci heap. In Dijkstra [10], by using a naive implementation, in a graph $G = (V, E)$, the running time of Dijkstra's algorithm was $O(|V|^2)$, where V is the number of nodes and E is the number of arcs. However, there are many techniques to improve the running time of Dijkstra's algorithm, and some of the most popular implementations and their efficiencies are listed in Table 2.1. (In a graph $G = (V, E)$, and C is the maximum arc length in G .)

2.3. Timetable Information Problems

The calculation of minimum paths in a transportation network is a well-developed part of transport network modeling. In comparison to the huge number of publications devoted to the shortest path algorithms for highway networks, only a few literatures discuss issues of finding minimum paths for public transportation networks. Although some features of public transportation networks appear similar to highway networks, some are fundamentally different. For instance, transit vehicles in public transportation networks are not available at call, and hence trip planning must include a cushion for waiting time. Therefore, a route transfer waiting time must be included in the trip planning. Moreover, changing from

one route to another must be made accordingly to timetables of each route. This problem had been identified by many researchers [44, 12], who argued that algorithms developed for highway networks or road networks for private vehicles are not suited for public transportation networks because of some fundamental differences between transit network and other networks. Chriqui [7] first discussed the problem of “common bus line”, where some bus routes share common sections and a passenger must select the buses he intends to use. Huang and Peng [19] pointed out that services in a transit system are controlled by their timetables. For these reasons, a different approach must be used in calculating shortest paths in transit networks.

According to Muller-Hannemann et al. [33], one of the most important timetable problems is the earliest-arrival problem. In this problem, the goal is to find a train connection from a departure station A to an arrival station B that departs from A not earlier than a given departure time and arrives at B as early as possible. Although the earliest arrival problem has been studied, details like transfer rules and traffic days are neglected. Most studies [46, 37, 34, 41] modeled the earliest arrival problem as a shortest path problem in a static graph and solve the problem by applying variants of Dijkstra’s algorithm.

In reality, we note that any minimum path in a transportation network using distance as the arc length without considering transfers between transit routes and timetable information is not so useful. Because considering only distance implicitly assumes that the transit vehicles are ready to run on any route at any given time, and thus transfer time and waiting time for the next available vehicle would be neglected. Therefore, a timetable-based network that takes both the transit route and timetable into consideration is more practical for seeking shortest paths in a public transportation network.

Tong and Richardson [46] proposed a network file from time schedules. The file contains records of all the arcs in the network and provides the input data for

executing the algorithm. However, their network structure only records the daily frequency that a vehicle passing through an arc, which is not completely compatible with the timetable information. In Huang and Peng [19], an objected-oriented GIS data model for the transportation network was developed to simplify the network structure. Their model provides more flexibility than traditional network structure. For example, stops at a street intersection, a transit center, or across a street on a street segment can be grouped as a stop group. Although their network is designed to incorporate timetable information, some path choices are skipped during network construction, and thus the optimal choice may be lost during the search process. Lo et al. [26] proposed a state-augmented multi-modal (SAM) network. In this network, arcs are classified into two groups: transfer links and direct in-vehicle links. Because SAM network is designed to accommodate transfer rules and transit fare, the timetable information is not included. Therefore, SAM network is not suitable for timetable-based transportation network.

In addition to timetable coordination, Schulz et al. [42] suggested that computation efficiency is another important requirement for finding optimal path in public transportation networks, especially when they are implemented in Internet or mobile-device trip-planning applications. In Internet or mobile environment, Peng et al. [39] reckoned that people may not have the enough time, and it may consume too much battery power for the software application to run for more than a few seconds. Therefore, as said in Hickman [17], performance is the key in these environments.

Tong and Wong [47] pointed out that an all-or-nothing assignment procedure could be used in which all flows are assigned to the minimum generalized cost itinerary. Schulz et al. [41] demonstrated that a modified Dijkstra's algorithm can find the shortest path with realistic timetable data of the German railway company. Huang and Peng [19] developed two schedule-based optimal-path algorithms, forward search and backward search, based on traditional shortest-path

algorithms. In Wu and Hartley [53], a forward search shortest-path algorithm was developed for bi-modal (bus and walking) network. In the studies by Huang and Peng [19] and Wu and Hartley [53], the concepts of the algorithms were similar, but the network structures are different. The former includes timetable information as a node attribute, the latter transforms timetable information into cost (travel time) of arc and starting time at a station. In both situations, however, some of the paths that may have late departure from an origin but early arrival to a destination are neglected, and thus the optimal path may not be found. All of these studies employed variants of Dijkstra's algorithm, which is in fact can be further improved using the algorithm of topological ordering from Ahuja et al.[1], since a timetable-based network is an acyclic directed diagram.

In a trip-planning system, a query generally defines a set of valid connections, and an optimization criterion on that set of connections. In other words, the system is to find the optimal connection with respect to the specific criterion. Muller-Hannemann et al. [33] mentioned that the most fundamental query is referred to as the earliest arrival problem. In an earliest arrival problem, given a query that consists of an origin node o , a destination node d , and an earliest departure time t_o , one has to seek a feasible route that starts from o and is composed by valid connections in a timetable-based network such that the difference between the arrival time at d and the departure time at o is minimized. Note that connections are valid only if they do not depart before the given earliest departure time t_o .

However, Wong et al. [51] indicated that most of the studies on timetable problem had focused on single-mode public transportation. Because it is a common phenomena for a public transportation network nowadays to comprise more than one mode, Lo et al. [27] inferred that developing a multimodal route finding system has become an important issue.

2.4. Multimodal Transit Problems

Nes and Bovy [35] defined the term multimodal as the combination of distinct functional and technical modes of transportation within a trip from origin to destination. The term mode indicates different forms (in a vehicular or functional sense) of transportation. This may consist of different vehicles, such as car, bicycle, tram, bus, train, or other different services (such as taxi). McCormack and Roberts [29] mentioned that it would be difficult to model multi-modal trip planning systems, especially those which consider both transit and road networks, with information sources supplied in a great variety of different formats.

For a multimodal trip that consists of two or more journeys with different modes, a transfer by foot is sometimes necessary. In Wu and Hartley [53], walking links were added into the transportation network to address bi-modal (walking and bus) problem, but the walking links were limited to travel from origin to the nearest transit station and to walk the transit station closest to the destination to the destination. While the travel time on bus could be computed with timetable information, for traveling on foot, distance between any pair of nodes requires calculation according to their location. Therefore, data such as longitude and latitude is essential in node information to model walking problem. In Koncz et al. [21], average walking speed and Euclidean distance were used to calculate walking time (or cost) to solve the more general multimodal problem. The average walking speed was set at 3 kilometer per hour, and thus the walking time is the result of walking distance divided by average walking speed. In their study, incorporating walking between transfer nodes into published transit timetables was mentioned as a future work.

According to Vuchic [48], in a multimodal transit network, there are primarily two types of transport. One is the bus-type transport. However, Horn [18] pointed out that the bus-type transport is basically a subset of the road network, but the freedom for travel is restricted to certain fix routes and timetables. In general,

paths for bus-type transport must begin and end at stops (or stations) and trips can be made only at the times specified in timetables. Another type is the rail-type transport. As Wong et al. [51] inferred, the rail-type transport is similar to the bus-type transport but independent of the road network. McCormack and Roberts [29] suggested that one of the important considerations for multimodal trips is whether interchanges are possible within the constraints of time and distance. In some situations, these two types of transport would offer different types of service; for instance, the bus-type transport may not obey any schedule while rail-transport follows some fixed timetables. To simplify the problem, we assume that all public transportation operated accordingly to pre-determined timetables.

On the basis of transport assignment method, Le Clercq [24] mentioned that several researches [36, 13] had proposed a route choice strategy, hyperpath. Hyperpath, connecting the origin to the destination, has the property that at each stop-node, the passenger awaits only a subset of the available transit lines. In Lozano and Storchi [28], shortest viable hyperpath was designed to address to multimodal problem. In a multimodal transit network, some sequences of transportation modes are illogical for transit users. For example, it is very unlikely for a traveler to board MRT, then transfer to bus, then switch back to MRT, and then alights from MRT and board bus again. Therefore, this sequence of transportation modes could be ignored in the transit network. Hyperpath is a method for assignment models, and assignment models are based on transit service frequencies and passenger distribution. In our study, each transit route is given a fixed timetable, so hyperpath is not applicable. However, the concept of eliminating unrealistic travel patterns may be useful in reducing the number of arcs in a transit network.

2.5. Fare Problems

As Lo et al. [26] inferred, for journey planning purposes, transportation modes may consist of different fare structures, and some of them are non-linear. Sen et

al. [43] pointed out that, for routing problems with non-linear fare structures, the literature is very sparse. Horn [18] further indicated that some fare structures in public transportation are zone-based (fare will remain constant over a certain number of stops), while others may be distance-based (with the fare increase linearly or non-linearly as the travel distance increases), and some may possess provision for group concessions (such as discount for students).

McCormack and Roberts [29] suggested that traversing a journey with timetable will mostly likely not depend on factors such as distance and speed of travel, but on fares tables and concessions depending on traveler's status. In Lo et al. [26], these non-linear fare structures made route costs non-additive, and thus total travel cost cannot be determined by simply adding up the costs of individual links, making this problem non-trivial.

In some situations, composite cost comprising a weighted sum of travel costs is applied. According to several researches [30, 31, 15, 26], this method is useful in finding an optimal path accommodating two or more possibly conflicting criteria. For example, one can assign different weight to waiting time for transfer and travel time in vehicle while optimizing a trip both for time and number of transfers. However, to simplify the problem, we assume that cost, in monetary unit, has equal value under all circumstances. In other words, the weight of cost is one for any link. Furthermore, we would adopt fixed fare tables in which cost between any two stations is known. This will give us a complete graph of the fare-based network, and thereby help us find the cheapest route.

CHAPTER 3

METHODOLOGIES ON A BASIC TIMETABLE INFORMATION PROBLEM

We first consider the basic timetable information problem in which users select the origin transit station, destination transit station, and intended departure time. In this basic timetable information problem, we will not consider walking as a transfer means. The problem that takes walking as a transfer means will be discussed in the next chapter.

In order to simplify the timetable-based transit network, we made some assumptions in advance:

Assumption 3.1: *There is no congestion in the transit system.*

Since there is no congestion in the transit system, there should be no delay during all journeys. Thus we could make the following assumption:

Assumption 3.2: *All transit vehicles arrive at or depart from each station accordingly to the timetable.*

To transfer from one transit route to another transit route at the same physical location, it would take a passenger a certain amount of time to alight and board. To simplify this situation, we assume that

Assumption 3.3: *The time to alight a transit vehicle and then board another transit vehicle immediately is constant.*

Assumption 3.4: *All passengers' origins and destinations are transit stations.*

Table 3.1. Nomenclature for the basic timetable information problem

A	set of arcs in transit network G
E	set of arcs in timetable-based network \tilde{G}
ED	set of non-stop arcs in timetable-based network \tilde{G}
EX	set of route-transfer arcs in timetable-based network \tilde{G}
G	transit network; $G = (N, A)$
\tilde{G}	timetable-based network; $\tilde{G} = (V, E)$
K	the number of line routes in G
N	set of nodes in transit network G
Q	maximum number of RF_k , $k = 1, \dots, K$
R_k	the arc sequence in Route k in G , $k = 1, \dots, K$
RF_k	the frequency of dispatching transit vehicle in R_k
RS_i	set of line route segments that are associated with node i , $i \in N$
RS_i^l	the l^{th} line route segment in RS_i , $l = 1, \dots, RS_i $
RSA_i	set of line route segments that arrive at node i , $i \in N$
RSD_i	set of line route segments that depart from node i , $i \in N$
TR_i	the modification of $TRRS_i$ with some consecutive duplicated times removed
$TRRS_i$	set of times that line route segments in RS_i take place in node i , $i \in N$
$TRRS_i^l$	the time that the route segment RS_i^l take place in node i , $i \in N$
Tx_i	the time to alight a transit vehicle and then board another transit vehicle
V	set of $ N $ groups of nodes in timetable-based network \tilde{G}
V_d	a node group composed by a set of $ TR_d $ copies of destination node d , $d \in N$
V_i	a node group composed by a set of $ TR_i $ copies of node i , $i \in N$
V_j	a node group composed by a set of $ TR_j $ copies of node j , $j \in N$
V_o	a node group composed by a set of $ TR_o $ copies of origin node o , $o \in N$
d	requested destination of user query
e	an arc in \tilde{G} , $e \in E$
i, j	node in transit network G ; $i, j \in N$
k	the index of line route in transit network G
l	the ordinal number of line route segment in transit network G
m	the physical location of the origin of a line route segment RS_i^l in RS_i
n	the physical location of the destination of a line route segment
o	requested origin of user query
r	the index of the route that passes node i
t	the arrival or departure time associated with a line route segment
t_o	intended departure time of user query
u, v	node in timetable-based network \tilde{G} , $u, v \in V$
v_d	origin node in transit network, $v_d = (d, r', TRRS_d^r, Tx_d) \in V_d$
v_o	destination node in transit network, $v_o = (o, r', TRRS_o^r, Tx_o) \in V_o$
w_d	pseudo destination constructed in algorithm EAB
w_o	pseudo origin constructed in algorithm EAB
y	number of queries with unique OD pairs
α	the origin of a non-stop arc in timetable-based network \tilde{G}
$\hat{\alpha}$	the origin of a route-transfer arc in timetable-based network \tilde{G}
β	the destination of a non-stop arc in timetable-based network \tilde{G}
$\hat{\beta}$	the destination of a route-transfer arc in timetable-based network \tilde{G}

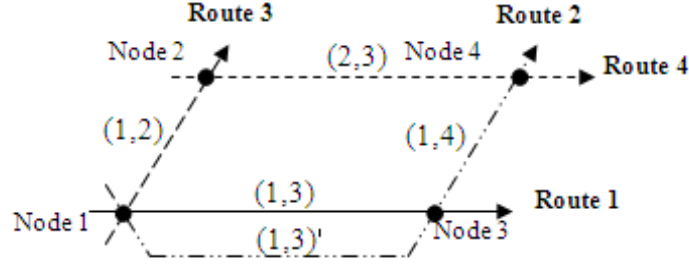


Figure 3.1. A simple transit network

3.1. Spatial Data and Temporal Data

Let digraph $G = (N, A)$ represent a transit network where N denotes the set of nodes and A denotes the set of arcs. A node in N represents some physical location where a transit vehicle can stop to pick up passengers, such as bus stop, port, or train station. Each node in N belongs to at least one line route. A line route is a set of nodes which transfer is not required to travel between them. For example, in Figure 3.1, we can take Route 1 to travel from node 1 to node 3 without transfer. Thus Route 1 is a line route. A directed arc $(i, j) \in A$ represents a direct connection (i.e. non-stop route segment) from node i to node j in a line route. Suppose there are K line routes in G , and R_k records the arc sequence in Route k for $k = 1, \dots, K$. The number of segments in R_k equals to $|R_k|$. Take Figure 3.1 for example, there are 4 nodes and 5 arcs, where Route 1 is composed by arc $(1, 3)$, Route 2 by arcs $(1, 3)'$ and $(3, 4)$, Route 3 by arc $(1, 2)$, and Route 4 by arc $(2, 4)$, respectively. Thus $R_1 = [(1, 3)]$, $R_2 = [(1, 3)', (3, 4)]$, $R_3 = [(1, 2)]$, and $R_4 = [(2, 4)]$. These topological connection relations can be stored as spatial data in a table shown as Table 3.2. Note the presence of parallel arcs (e.g. $(1, 3)$ and $(1, 3)'$ in Figure 3.1) in the transit network; there is often more than one line routes between two stops.

To calculate the quickest path, one further requires the timetable information, referred as the temporal data. If two line routes with same arc sequence have

Table 3.2. Line route information

<i>Segment No.</i>	<i>Route No.</i>	<i>Route Segment</i>
1	Route 1	: Node 1 \rightarrow Node 3
2	Route 2	: Node 1 \rightarrow Node 3
3		: Node 3 \rightarrow Node 4
4	Route 3	: Node 1 \rightarrow Node 2
5	Route 4	: Node 2 \rightarrow Node 4

different timetables, they should be taken as a different line routes. A timetable consists of data concerning nodes, line routes, and the departure and arrival times of the transit vehicles. Let RF_k denote the frequency of R_k (i.e. how often is a transit vehicle dispatched in a line route) and $RS_i = RSD_i \cup RSA_i$ be all the line route segments associated with node i , where RSD_i and RSA_i represent the sets of line route segments that depart from and arrive at node i , respectively. For each route segment RS_i^l , $l = 1, \dots, |RS_i|$ associated with node $i \in N$, we record TRS_i^l as the time that the route segment RS_i^l takes place in node i . Thus $|TRS_i| = |RS_i| = |RSD_i| + |RSA_i|$. With timetable information embedded, different arrival and departure times for a node in a line route will create different line route segments. For example, referring to Table 3.2 and Table 3.3, there are 9 line route segments associated with node 1, which are $(m, n, r, t) = [(1, 3, 1, 8:01), (1, 3, 1, 8:11), (1, 3, 1, 8:21), (1, 3, 2, 8:00), (1, 3, 2, 8:15), (1, 3, 2, 8:30), (1, 2, 3, 8:05), (1, 2, 3, 8:15), (1, 2, 3, 8:25)]$ where m is the origin of each line route segment, n is the origin of each line route segment, r is the index of line route, and t is the departure or arrival associated with the line route segment.

To further simplify the problem, we assume that for any consecutive line route segments (\check{i}, i) and (i, \hat{i}) on R_k , the arrival of (\check{i}, i) and departure of (i, \hat{i}) happens at the same time at node i . As a result, we can remove some consecutive duplicated times in the vector TRS_i when node i is an intermediate node in R_k , and save the modified vector as TR_i . Note that $|TRS_i| = |TR_i| + \sum_{k: \text{node } i \text{ is an intermediate node in } R_k} RF_k$, since each time when a route passes through

Table 3.3. Timetable information for each route

	Route 1		Route 2			Route 3		Route 4	
	Node 1	Node 3	Node 1	Node 3	Node 4	Node 1	Node 2	Node 2	Node 4
1	8:01	8:21	8:00	8:23	8:30	8:05	8:15	8:00	8:15
2	8:11	8:31	8:15	8:38	8:45	8:15	8:25	8:14	8:29
3	8:21	8:41	8:30	8:53	9:00	8:25	8:35	8:28	8:43

node i , either its arrival or departure time will be doubly counted in $|TRS_i|$. Furthermore, the departure time of each line route segment is always later than or equal to the preceding arrival time (except the first one, which has no preceding arrival time); that is $TRS_i^l \leq TRS_i^{l'}$ if $l < l'$. Thus any passenger alight a line route at some intermediate node will not be able to take any earlier dispatch at the same node. To comprehend more thoroughly, let's look at the following example.

Table 3.3 is the timetable for the simple transit network in Table 3.2, where all routes have three departures (i.e. $RF_k = 3$ for $k = 1, 2, 3, 4$). Node 1 has 9 departure line route segments and 0 arrival line route segments, thus $|RSD_1| = 9$, $|RSA_1| = 0$, and $|TRS_1| = |TR_1| = [8:00, 8:01, 8:05, 8:11, 8:15, 8:15, 8:21, 8:25, 8:30]$. Similarly, $|RSD_2| = 3$, $|RSA_2| = 3$, and $|TRS_2| = |TR_2| = [8:00, 8:14, 8:15, 8:25, 8:28, 8:35]$; $|RSD_3| = 3$, $|RSA_3| = 6$, $|TRS_3| = [8:21, 8:23, 8:23, 8:31, 8:38, 8:38, 8:41, 8:53, 8:53]$, and $|TR_3| = [8:21, 8:23, 8:31, 8:38, 8:41, 8:53]$; $|RSD_4| = 0$, $|RSA_4| = 6$, and $|TRS_4| = [8:15, 8:29, 8:30, 8:43, 8:45, 9:00]$. Since $RF_k = 3$ and node 3 is an intermediate node in R_2 , 8:23, 8:38, and 8:53 are duplicated in TRS_3 .

3.2. Construction of a Basic Timetable-based Network

To calculate the itinerary with the shortest travel time in a transit network, one requires incorporating the temporal information into the transit network. To this end, here we propose a preprocessing algorithm to construct a time-space network called the basic timetable-based network, denoted by $\tilde{G} = (V, E)$, where V is a set of $|N|$ groups of nodes and $E = ED \cup EX$ consists of the non-stop arcs

(ED , arcs that are in the same line route) and route-transfer arcs (EX , arcs that connect different line routes).

A node group $V_i \subseteq V$ for each $i \in N$ is composed by a set of $|TR_i|$ copies of node $i \in N$, where each copy corresponds to a specific time associated with a line route segment connecting i . Thus $|V| = \sum_{i \in N} |TR_i|$. For our convenience, four tuples, (i, r, TRS_i^l, Tx_i) , are used to describe each node $v \in V$ in the basic timetable-based transportation network, where i corresponds to a physical location of node $i \in N$, r represents the index of the route that passes i , TRS_i^l records the time of the l^{th} route segment in RS_i arriving at or departing from node i (e.g. 8:15), and Tx_i is the time to alight a transit vehicle and then board another transit vehicle (e.g. 2 minutes) set by the users.

Each non-stop arc $(\alpha, \beta) \in ED$ represents each line route segment in $\bigcup_{i \in N} RS_i$, and its end nodes correspond to the end nodes of that line route segment at a specific time. That is, if $\alpha \in V_i$ and $\beta \in V_j$, then $i \neq j$. Furthermore, $|ED| = \sum_{i \in N} |RS_i|$. The orientation for a non-stop arc follows the sequence of the nodes appeared in its corresponding line route segment. Thus a non-stop arc is directed from a node of earlier time to a node of later time, and we set its length to be the duration of its corresponding line route segment, or in other words, the difference in time on its end nodes. On the other hand, each route-transfer arc $(\hat{\alpha}, \hat{\beta}) \in EX$ connects two nodes of the same physical location (i.e. if $\hat{\alpha} \in V_i$ and $\hat{\beta} \in V_j$, then $i = j$) but different times, as long as the time difference of end nodes does not exceed the time for route change. In particular, for each route-transfer arc $(\hat{\alpha}, \hat{\beta}) \in EX$ where $\hat{\alpha} = (i, r_1, TRS_i^{l_1}, Tx_i)$ and $\hat{\beta} = (i, r_2, TRS_i^{l_2}, Tx_i)$ correspond to some physical location $i \in N$, its orientation directs from u to v and its length can be set as $TRS_i^{l_2} - TRS_i^{l_1}$, which is greater than or equal to Tx_i . Therefore, the number of route-transfer arcs connecting nodes in $V_i \subseteq V$ is at most $O(|V_i|^2)$, and thus $|EX| = O(\sum_{i \in N} |V_i|^2)$.

Now we give steps of our preprocessing algorithm **CBTBN** (Constructing the Basic Timetable-based Network) as follows:

Step 1: Read the spatial data (e.g. Table 3.2) and temporal data (e.g. Table 3.3), store data in $N, A, RS, RSD, RSA, TRS, TR$.

Step 2: For each node $i \in V$, we store each node the defined four tuples (i, r, TRS_i^l, Tx_i) .

Step 3: Construct each non-stop arc $(\alpha, \beta) \in ED$, using V and RS .

Step 4: Construct each route-transfer arc $(\hat{\alpha}, \hat{\beta}) \in EX$, using V .

Since $\sum_{i \in N} |TR_i| = O(\sum_{i \in N} |RS_i|) = O(\sum_{k=1, \dots, K} RF_k |R_k|) = O(KQ|N|)$, where $Q = \max_{k=1, \dots, K} RF_k$, we can derive that $|V| = \sum_{i \in N} |TR_i| = O(KQ|N|)$ and $|E| = |ED| + |EX| = \sum_{i \in N} |RS_i| + O(\sum_{i \in N} |V_i|^2) = O(KQ|N| + |N|^2)$, which means the size of \tilde{G} is a polynomial function of the size of G (i.e. $|N|$) and the input temporal data (i.e. K and Q). In other words, algorithm CBTBN can construct a basic timetable-based network in time polynomial to the input data sizes $|N|$. Moreover, each arc in \tilde{G} always directs from a node of earlier time to a node of later time, which means \tilde{G} is an acyclic diagram.

3.3. Query and Solution Method on the Basic Timetable-based Network

In this section, we design an algorithm to solve the earliest arrival problem on the basic timetable-based network, and then demonstrate this procedure with the simple transit network shown in Figure 3.1. Since the timetable-based network is acyclic, we can apply topological-ordering-based algorithms to solve the shortest path problem.

3.3.1. Procedures

Depending on the locations of the requested origin o and destination d , as well as the starting time t_o of the query, we would like to identify a feasible itinerary

of the minimum travel time from o to d in G . Here we give an algorithm called **EAB** (stands for Earliest Arrival for Basic timetable-based network) with steps as follows:

- Step 1:** Read the basic timetable-based network \tilde{G} , as well as all the related data structures.
- Step 2:** Read the indices of the requested origin o and destination d , and the starting time t_o .
- Step 3:** For each $(u, v) \in E$, where $u = (i, r, TRS_i^l, Tx_i)$ with $TRS_i^l < t_o$, remove it from E .
- Step 4:** Remove those route-transfer arcs whose both end nodes are in V_o or V_d .
- Step 5:** Construct a pseudo origin w_o , a pseudo destination w_d , artificial arcs (w_o, v_o) for each node $v_o = (o, r', TRS_o^{r'}, Tx_o) \in V_o$ with nonnegative length equal to $TRS_o^{r'} < t_o$, and artificial arcs (v_d, w_d) with zero length for each node $v_d = (d, r', TRS_d^{r'}, Tx_d) \in V_d$ with $TRS_d^{r'} \geq t_o$.
- Step 6:** Solve a shortest path from w_o to w_d in \tilde{G} using any shortest path algorithm.
- Step 7:** Output the calculated shortest path, which corresponds to the quickest itinerary as requested.

In particular, based on the basic timetable-based network \tilde{G} , Step 3 first eliminates arcs that are too early for the user; Step 4 removes those route-transfer arcs inside the node groups corresponding to o and d since one can only change line routes at intermediate nodes but not the origin and destination; Step 5 connects the pseudo origin and pseudo destination to their corresponding nodes that serve as the start and end of the request itinerary. At this moment, algorithm EAB has already incorporated the query information into the modified basic timetable-based network. Since the length of each arc in the modified basic timetable-based network either represents the duration of a line route segment (for a non-stop arc),

the time to transfer between line routes (for a route-transfer arc), or the waiting time before boarding a transit vehicle (for an artificial arc connecting to nodes in V_o), a shortest path in the modified basic timetable-based network is thus an itinerary with the shortest travel time from o to d starting from time t_o .

Since the artificial arcs added to \tilde{G} also direct from nodes of earlier time to nodes of later time, the modified basic timetable-based graph is still acyclic. For seeking a shortest path in an acyclic diagram of $|V|$ nodes and $|E|$ arcs, the topological ordering algorithm [1] can be modified to solve a shortest path in $O(|E|)$ time, which is theoretically the most efficient since any shortest path algorithm takes at least $\Omega(|E|)$ time to read the input network. To our surprise, the point of using a topological ordering algorithm to seek a shortest path in a time-space network seems to have been neglected in the literatures. Most of the related researches exploit variants of the Dijkstra's algorithm, and that takes $O(|E| + |V| \log |V|)$ time by Fredman and Tarjan [11]. Empirically speaking, the results of the computational experiments conducted by Cherkassky et al. [6] and Wang [49] also indicate that topological-ordering-based algorithms such as the algorithms by Goldberg and Radzik [14] and Wang et al. [50] are more efficient than variants of Dijkstra's algorithm to solve shortest paths in an acyclic diagram.

For solving the shortest path itinerary problems for multiple (say, $y > 1$) OD pairs, we only need to construct the basic timetable-based network once and repeats Step 2 through Step 7 for y times, rather than constructing the basic timetable-based network from the scratch for y times. Furthermore, such a problem is also a good practice for applying the multiple pairs shortest path algorithm by Wang et al. [50], which is designed for solving shortest paths for multiple OD pairs on a network of fixed topology (in our case, the basic timetable-based network).

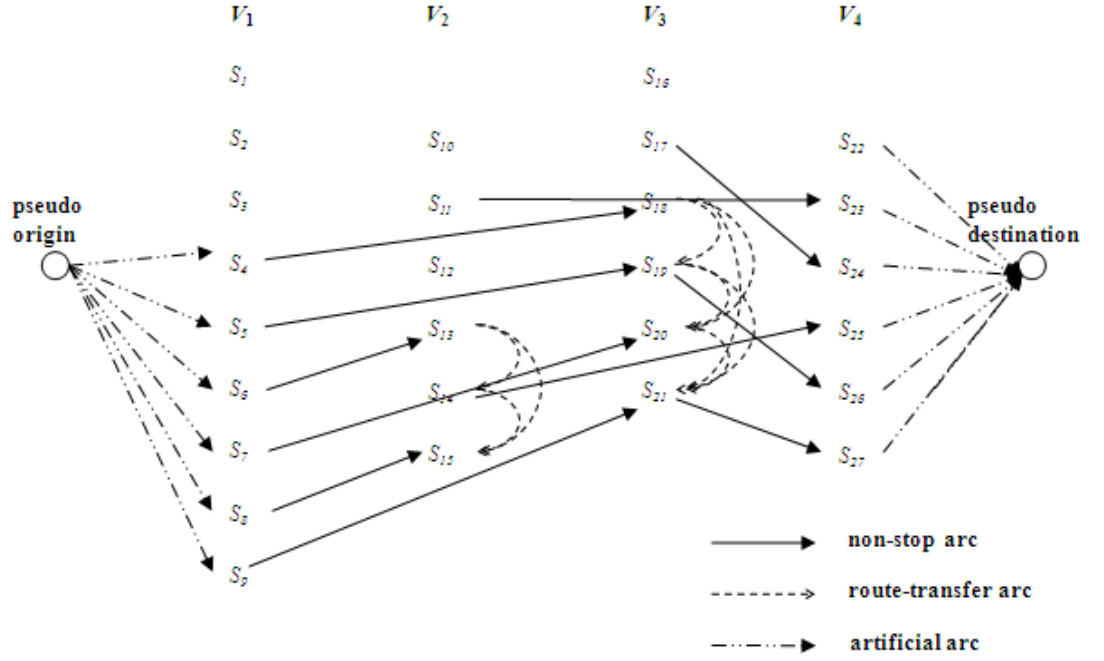


Figure 3.2. A timetable-based transportation network

3.3.2. An Illustrative Example

To demonstrate the formulation of our basic timetable-based transportation network, consider a transit network consisted of four transit routes and four nodes as shown in Figure 3.1 and Table 3.2. Table 3.3 gives the timetable of this transit system.

In this example, if a transfer is needed at any node, a two-minute time period is required. In other words, $Tx_i = 2$ for each node $i \in N$. Suppose that a passenger plans to travel from node 1 to node 4 with a planned departure time of 8:10. Then, our algorithm EAB eliminates unqualified arcs from the basic timetable-based network that has been stored in the memory, adds pseudo nodes and artificial arcs to construct a modified basic timetable-based network as shown in Figure 3.2, and solve a shortest path from the pseudo origin to the pseudo destination. In this case, the itinerary with the minimum travel time will be starting from node 1, taking the bus of route 3 on 8:15, getting off the bus at node 2 on 8:25, taking

Table 3.4. Nodes information of basic timetable-based network

S_1	(1, 2, 8:00, 2)	S_{10}	(2, 4, 8:00, 2)	S_{16}	(3, 1, 8:21, 2)	S_{22}	(4, 4, 8:15,2)
S_2	(1, 1, 8:01, 2)	S_{11}	(2, 4, 8:14, 2)	S_{17}	(3, 2, 8:31, 2)	S_{23}	(4, 4, 8:29,2)
S_3	(1, 3, 8:05, 2)	S_{12}	(2, 3, 8:15, 2)	S_{18}	(3, 1, 8:31, 2)	S_{24}	(4, 2, 8:30,2)
S_4	(1, 1, 8:11, 2)	S_{13}	(2, 3, 8:25, 2)	S_{19}	(3, 2, 8:38, 2)	S_{25}	(4, 4, 8:43,2)
S_5	(1, 2, 8:15, 2)	S_{14}	(2, 4, 8:28, 2)	S_{20}	(3, 1, 8:41, 2)	S_{26}	(4, 2, 8:45,2)
S_6	(1, 3, 8:15, 2)	S_{15}	(2, 3, 8:35, 2)	S_{21}	(3, 2, 8:53, 2)	S_{27}	(4, 2, 9:00,2)
S_7	(1, 1, 8:21, 2)						
S_8	(1, 3, 8:25, 2)						
S_9	(1, 2, 8:30, 2)						

the bus of route 4 on 8:28, and arriving at node 4 on 8:43. Table 3.4 demonstrates the information stored in each node.

3.4. Computational Experiments

This section summarizes our computational results of basic timetable information problem. After introducing the implementation settings, datasets from Taipei transit system will be used for the implementations. We will compare two implementation of shortest path algorithms: topological ordering algorithm, and Dijkstra's algorithm.

3.4.1. Settings and Problem Sets for the Implementation

All of our computational experiments are conducted using Microsoft Visual Studio 2005 on a Acer Aspire machine with an Intel Core 2 Duo processor at 1866 MHz and 2039 MB RAM running Microsoft Windows XP SP2.

We test our implementation on problem sets based on the bus datasets obtained from China Engineering Consultants Incorporation (CECI) and MRT datasets obtained from Taipei Rapid Transit Corporation (i.e. Taipei Metro). The bus datasets consists of bus route information and bus stop locations. The MRT datasets consists of MRT station location. With the locations of bus stops and MRT stations, we could estimate the distance between stops and stations. According to Institute of Transportation, MOTC [20], the average speed for a bus in Taipei is 22.88 kilometer per hour, and the average walking speed for a Taipei

pedestrian is 4 kilometer per hour. With the average travel time between MRT station provided by Taipei Metro, we could estimate the length (travel time) of arcs in Taipei transit system.

Because Taipei transit system does not obey fixed time schedules in reality, we generate timetables for each bus stop or MRT stations. We assume that Taipei transit system operates from 8 a.m. to 10 p.m.; in other words, the earliest available time for bus or MRT service is 8 a.m. and the last possible time for a passenger to board a bus or MRT train is 10 p.m.. The service intervals for each line route are randomly-generated values between 10 minutes and 30 minutes.

3.4.2. Algorithmic Running Time Comparison

We select 5 problem sets for computational experiments. Problem set P_1 consists of 139 line routes and 3392 transit stations. Problem set P_2 consists of 160 line routes and 3732 transit stations. Problem set P_3 consists of 187 line routes and 4117 transit stations. Problem sets P_4 consists of 202 line routes and 4240 transit stations. Problem set P_5 , the complete network of Taipei transit system, consists of 336 line routes and 5451 transit stations. The number of nodes for the original transit network, the number of nodes for the basic timetable-based network, and the time to construct the basic timetable-based network are summarized in Table 3.5. The time to construct the network increases as the network size increases.

For each problem set, we select 10 groups of OD pairs. In each OD groups, there are 100 OD pairs with approximately same Euclidean distance. In addition, we conduct experiment on 5000 OD pairs with origins and destinations randomly selected. Each OD pair was checked by Depth-First Search (DFS) to assure feasible solutions before applying topological ordering algorithm. The unit of time for performance comparison is central processing unit time (CPU time). CPU time is the amount of time that a computer program consumes in processing central

Table 3.5. Time to construct basic timetable-based networks

Set	Routes	Nodes/Arcs (original)	Nodes/Arcs(expanded)	Time
P ₁	139	3392 / 11001	152317 / 1061154	22391
P ₂	160	3732 / 13183	159627 / 1122708	25610
P ₃	187	4117 / 14892	177180 / 1397623	27847
P ₄	202	4240 / 15854	180333 / 1650558	30359
P ₅	336	5451 / 25736	232613 / 2033405	41908

Table 3.6. Relative performance on five problem sets with randomly selected OD pairs

Set	DIJ	TO	SPD (=DFS+TOAD)	ΔT_D	SPB (=BFS+TOAB)	ΔT_B
P ₁	3938	2159	1057 (=610+447)	51.0%	1043 (=617+426)	51.7%
P ₂	4257	2438	1150 (=658+492)	52.6%	1132 (=674+458)	53.6%
P ₃	4405	2557	1265 (=736+529)	50.5%	1226 (=743+483)	52.1%
P ₄	4352	2612	1277 (=735+542)	51.1%	1255 (=755+500)	52.0%
P ₅	5270	2963	1425 (=812+613)	51.9%	1383 (=814+568)	53.3%
time unit in CPU click $\Delta T_D = (TO - SPD)/TO \times 100\%$ $\Delta T_B = (TO - SPB)/TO \times 100\%$						

processing unit (CPU) instructions. The CPU time is often measured in clock ticks and is thereby used as a point of comparison for CPU usage of a program.

Two shortest path algorithms are implemented for computational experiments. The first algorithm is Dijkstra’s algorithm with binary heap, denoted as *DIJ*. The second algorithm is a topological-ordering-based algorithm, denoted as *TO*. The results for problem set 5, the complete Taipei transit system, are summarized in Table 3.7 (*SPD*, *DFS*, *TOAD*, *SPB*, *BFS*, *TOAB*, ΔT_D and ΔT_B will be discussed in section 3.5). As for the computational performance on 5000 randomly selected OD pairs, the results are presented in Table 3.6. Results for other problem sets are shown in Appendix A.

We notice that topological-ordering-based algorithm performs better than Dijkstra’s algorithm on basic timetable-based network. The reason why topological-ordering-based algorithm performs better than Dijkstra’s algorithm is that the topological ordering algorithm [1] can be modified to solve a shortest path in $O(|E|)$ time while Dijkstra’s algorithm takes $O(|E| + |V| \log |V|)$ time at best

[11]. Generally, the result could be obtained within seconds in real time for topological-ordering-based algorithm.

3.5. Speed-up Techniques

We have successfully implemented two algorithms, topological-ordering-based algorithm and Dijkstra's algorithm, to solve basic timetable information problem. The computational efficiency is reasonable on a personal computer (PC). However, in order to operate on a mobile device, such as cellular phone or PDA, the performance of our current solution may not be efficient enough due to inferior CPU or less memory space equipped with mobile device. Facing these limitations, some speed-up techniques may be necessary.

One proposed speed-up technique is to find a feasible solution through DFS before applying shortest path algorithm, denoted as SPD. By finding a feasible, but not necessarily the shortest, solution, we could set a boundary on our searching. Since the shortest path should have a cost lower than or equal to any feasible solution, we could eliminate the nodes that operate later the sum of intended departure time and the cost of feasible solution obtained through DFS. The complexity is the same, but the searched network size could be potentially much smaller. However, based on the nature of the timetable-based network, we can use Breadth-first Search (BFS) to find a feasible solution slightly more efficiently. Since a passenger will not transfer from one line route to another more than three times often according to our computational results, the feasible solution obtained from BFS could potentially be closer to the optimal solution than the feasible solution obtained from DFS. Theoretically, BFS may take longer to find a feasible solution than DFS but may eliminate more nodes through feasible solution closer to optimal solution. The size of network is smaller after applying BFS and thus the shortest path algorithm can find the optimal solution faster. The speed-up technique using

BFS to find a feasible solution before applying shortest path algorithm is denoted as SPB.

Another proposed speed-up technique is finding the solution with longest travel time in the multimodal timetable-based network, denoted as SPL. If the longest possible travel time is known, we could eliminate the nodes that operate later than the sum of intended departure time and the longest possible travel time. By applying these techniques, the complexity remains the same, but the searched network size could be smaller.

We hereby apply SPD and SPB on our 5 problem sets. Table 3.7 summarizes the implementation results of 10 groups of OD pairs in problem set 5. Results on other problem sets are shown in Appendix A. In Table 3.7, *DIJ* is the computational time for Dijkstra's algorithm, *TO* is the average computational time for topological ordering algorithm, *DFS* is the computational time to conduct DFS for an feasible solution and then set the boundary accordingly, *TOAD* is the computational time for topological ordering algorithm after the speed-up technique is applied, and *SPD* is the total computational time with speed-up technique, i.e. the total of *DFS* and *TOAD*. *BFS* is the computational time to conduct BFS for an feasible solution and then set the boundary accordingly, *TOAB* is the computational time for topological ordering algorithm after the speed-up technique is applied, and *SPB* is the total computational time with speed-up technique, i.e. the total of *BFS* and *TOAB*. In addition, we list ΔT_D , the percentage of time improvement from plain topological ordering algorithm to topological ordering algorithm with speed up technique using DFS (i.e. $\Delta T\% = (TO - SPD)/TO \times 100\%$) and ΔT_B , the percentage of time improvement from plain topological ordering algorithm to topological ordering algorithm with speed up technique using BFS (i.e. $\Delta T\% = (TO - SPB)/TO \times 100\%$) in the table.

Table 3.7. Relative performance on problem set 5

OD Group	DIJ	TO	SPD (=DFS+TOAD)	ΔT_D	SPB (=BFS+TOAB)	ΔT_B
1	4436	2635	1251 (=711+540)	52.5%	1224 (=725+499)	53.5%
2	4390	2599	1241 (=708+533)	52.3%	1233 (=711+522)	52.5%
3	4552	2724	1286 (=728+588)	52.8%	1310 (=736+574)	51.9%
4	4678	2882	1366 (=777+589)	52.6%	1336 (=778+558)	53.6%
5	4973	3169	1480 (=837+643)	53.3%	1458 (=860+598)	54.0%
6	5314	3506	1637 (=924+713)	53.3%	1602 (=938+664)	54.3%
7	5296	3486	1634 (=922+712)	53.1%	1639 (=948+691)	53.0%
8	5311	3518	1643 (=927+716)	53.3%	1616 (=942+674)	54.1%
9	5668	3862	1803 (=1019+784)	53.3%	1742 (=1022+720)	54.9%
10	6239	4430	2058 (=1161+897)	53.5%	2048 (=1192+856)	53.8%
time unit in CPU click $\Delta T_D = (TO - SPD)/TO \times 100\%$ $\Delta T_B = (TO - SPB)/TO \times 100\%$						

Table 3.8. Normalization of relative performance on problem set 5

OD Group	DIJ	TO	SPD (=DFS + TOAD)	SPB (=BFS + TOAB)
1	3.54	2.11	1.00 (=0.57 + 0.43)	0.98 (=0.58 + 0.40)
2	3.54	2.10	1.00 (=0.57 + 0.43)	0.99 (=0.57 + 0.42)
3	3.54	2.12	1.00 (=0.57 + 0.43)	1.02 (=0.57 + 0.45)
4	3.42	2.11	1.00 (=0.57 + 0.43)	0.98 (=0.57 + 0.41)
5	3.36	2.14	1.00 (=0.57 + 0.43)	0.99 (=0.57 + 0.40)
6	3.25	2.14	1.00 (=0.56 + 0.44)	0.98 (=0.57 + 0.41)
7	3.24	2.13	1.00 (=0.56 + 0.44)	1.00 (=0.58 + 0.42)
8	3.23	2.14	1.00 (=0.56 + 0.44)	0.98 (=0.57 + 0.41)
9	3.14	2.14	1.00 (=0.57 + 0.43)	0.97 (=0.57 + 0.40)
10	3.03	2.15	1.00 (=0.56 + 0.44)	1.00 (=0.58 + 0.42)

By applying our first proposed speed-up technique, the average running time improves approximately 53% on problem set 5. By applying our speed-up technique using BFS, the average running time improves approximately 55%. We further normalize the results for each OD group in problem set 5 on the basis of SPD and summarize in Table 3.8. The ratio between time usage of DFS and that of TOAD or BFS and that of TOAB remains approximately constant throughout all problem sets.

3.6. Summary

In this chapter, we solve the basic timetable information problem. In addition to the line route information of a transit system, our problem takes timetable

into consideration. In order to incorporate timetable information, a new network, basic timetable-based network, is constructed through algorithm CBTBN. After constructing the basic timetable-based network, we then apply algorithm EAB with implementations of topological ordering algorithm and Dijkstra’s algorithm.

By conducting computational experiments on datasets from Taipei transit network, the results have shown reasonable efficiency on PC. Our computational experiments also support the theoretical projection that topological ordering algorithm should perform better than Dijkstra’s algorithm. Furthermore, we propose some speed-up techniques to improve computational efficiency. The results of applying our first proposed speed-up technique, SPD, show a 53% running time improvement. and the results of applying speed-up technique using BFS, SPB, show a 54% running time improvement. Although the computational results indicate that SPB has a better efficiency than SPD, the improvement attributes the nature of Taipei transit system. Whether SPB has better computational efficiency than SPD or not requires further validation. The actual performance of mobile device remains unknown.

CHAPTER 4

METHODOLOGIES ON A MULTIMODAL TIMETABLE INFORMATION PROBLEM WITH WALKING TRANSFER

We then consider a more complicated timetable problem in which users select the origin transit station, destination transit station, and intended departure time. The origin and destination are not limited to transit stations, and walking is permitted as a means of transportation.

To simplify the multimodal timetable information problem with walking transfer, some assumptions are made:

Assumption 4.1: *There is no congestion in the transit system.*

Since there is no congestion in the transit system, there should be no delay during all journeys. Thus we could make the following assumption:

Assumption 4.2: *All transit vehicles arrive at or depart from each node accordingly to the timetables.*

To transfer from one transit route to another transit route at the same physical location, it would take a passenger a certain amount of time to alight and board. To simplify this situation, we assume that

Assumption 4.3: *Waiting time for route transfer at the same physical location is constant.*

Since walking is allowed in this problem, we further assume that

Assumption 4.4: *The walking speed remains the same for every traveler.*

However, the nature of private road network (for walking) is very different from that of transit network. In order to prevent details of private road network from increasing computation complexity, we assume that

Assumption 4.5: *The walking distance from one node to another is defined as the Euclidean distance between those two nodes.*

4.1. Spatial Data and Temporal Data

Let digraph $G = (N, A)$ represent a transit network where N denotes the set of nodes and A denotes the set of arcs. A node in N represents some physical location where a transit vehicle can stop to pick up passengers, such as bus stop, port, or train station. Each node in N belongs to at least one line route. A line route is a set of nodes which transfer is not required to travel between them. For example, in Figure 4.1, we can take Route 1 to travel from node 1 to node 3 without transfer. Thus Route 1 is a line route. A directed arc $(i, j) \in A$ represents a direct connection (i.e. non-stop route segment) from node i to node j in a line route. Suppose there are K line routes in G , and R_k records the arc sequence in Route k for $k = 1, \dots, K$. The number of segments in R_k equals to $|R_k|$. Take Figure 4.1 for example, there are 6 nodes and 4 arcs, where Route 1 is composed by arc $(1, 2)$ and $(2, 3)$, Route 2 by arcs $(4, 5)$ and $(5, 6)$, respectively. Thus $R_1 = [(1, 2), (2, 3)]$ and $R_2 = [(4, 5), (5, 6)]$. These topological connection relations can be stored as part of spatial data in a table shown as Table 4.2.

Notice that in Figure 4.1, we demonstrate that a transit network with multiple line routes can be decomposed into multiple layers of network, in which each layer consists of one line route. In other words, if there are K line routes in the transit network G , then G can be decomposed into K layers (for the transit network in Figure 4.1, there are two layers, G_{BR1} and G_{BR2}). This decomposition

Table 4.1. Nomenclature for the multimodal timetable information problem with walking transfer

A	set of arcs in transit network G
C	set of walking distance in timetable-based network \tilde{G}
E	set of arcs in timetable-based network \tilde{G}
ED	set of non-stop arcs in timetable-based network \tilde{G}
EX	set of route-transfer arcs in timetable-based network \tilde{G}
EW	set of walk-transfer arcs in timetable-based network \tilde{G}
G	transit network; $G = (N, A)$
\tilde{G}	timetable-based network; $\tilde{G} = (V, E)$
K	the number of line routes in G
N	set of nodes in transit network G
R_k	the arc sequence in Route k in G , $k = 1, \dots, K$
RF_k	the frequency of dispatching transit vehicle in R_k
RS_i	set of line route segments that are associated with node i , $i \in N$
RS_i^l	the l^{th} line route segment in RS_i , $l = 1, \dots, RS_i $
RSA_i	set of line route segments that arrive at node i , $i \in N$
RSD_i	set of line route segments that depart from node i , $i \in N$
TR_i	the modification of TRS_i with some consecutive duplicated times removed
TRS_i	set of times that line route segments in RS_i take place in node i , $i \in N$
TRS_i^l	the time that the route segment RS_i^l take place in node i , $i \in N$
TW_{ij}	walk time between node i and j , $i, j \in N$
Tx_i	the time to alight a transit vehicle and then board another transit vehicle
V	set of $ N $ groups of nodes in timetable-based network \tilde{G}
V_d	a node group composed by a set of $ TR_d $ copies of destination node d , $d \in N$
V_i	a node group composed by a set of $ TR_i $ copies of node i , $i \in N$
V_o	a node group composed by a set of $ TR_o $ copies of origin node o , $o \in N$
c_{ij}	the walking distance between node i and j , $i, j \in N$
a, b	the index of line route in transit network G
d	requested destination of user query
i, j	node in transit network G ; $i, j \in N$
k	the index of line route in transit network G
l	the ordinal number of line route segment in transit network G
m	the physical location of the origin of a line route segment RS_i^l in RS_i
n	the physical location of the destination of a line route segment
o	requested origin of user query
r	the index of the route that passes node i and j , $i, j \in N$
s	the walking speed in transit network G
t	the arrival or departure time associated with a line route segment
t_o	intended departure time of user query
t_s	specified acceptable walking range
u, v	node in timetable-based network \tilde{G} , $u, v \in V$
v_d	origin node in transit network, $v_d = (d, r', TRS_d^r, Tx_d) \in V_d$
v_o	destination node in transit network, $v_o = (o, r', TRS_o^r, Tx_o) \in V_o$
w_d	pseudo destination constructed in algorithm EAM
w_o	pseudo origin constructed in algorithm EAM
α, β	the origin/destination of a non-stop arc in timetable-based network \tilde{G}
$\hat{\alpha}, \hat{\beta}$	the origin/destination of a route-transfer arc in timetable-based network \tilde{G}
$\acute{\alpha}, \acute{\beta}$	the origin/destination of a walk-transfer arc in timetable-based network \tilde{G}

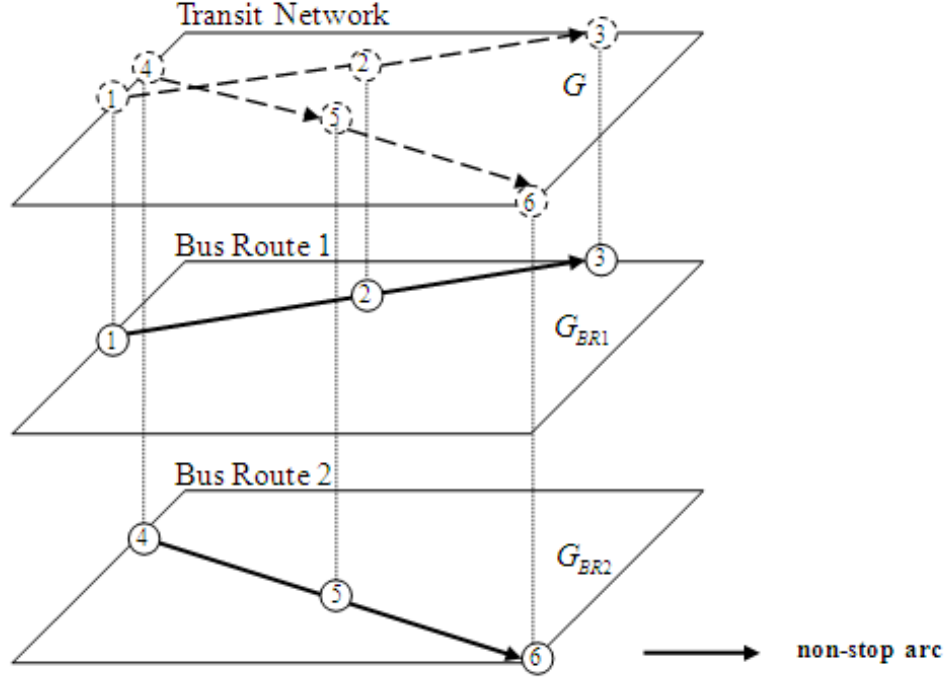


Figure 4.1. A multimodal transit network and its hierarchical display

Table 4.2. Line route information

<i>Segment No.</i>	<i>Route No.</i>	<i>Route Segment</i>
1	Route 1	: Node 1 \rightarrow Node 2
2		: Node 2 \rightarrow Node 3
3	Route 2	: Node 4 \rightarrow Node 5
4		: Node 5 \rightarrow Node 6

process would create a hierarchical transit network, and thus named Hierarchical Decomposition Process (HDP).

Another part of spatial data is the walking distance between each node. Based on Assumption 4.5, the walking distance between two nodes is defined as the Euclidean distance between them. Therefore, the walking distance between each node can be denoted as c_{ij} , $i, j \in N$, $i \neq j$. These walking distances can be stored as part of spatial data using a table as shown in Table 4.3. Then by setting the walking speed, the time consumed for each walk segment can be obtained.

To calculate the quickest path, one further requires the timetable information, referred as the temporal data. If two line routes with the same arc sequence have

Table 4.3. Walk distance information

<i>Walk No.</i>	<i>Walk Segment</i>	<i>Distance</i>
1	Node 1 \rightarrow Node 2 :	c_{12}
2	Node 1 \rightarrow Node 3 :	c_{13}
3	Node 1 \rightarrow Node 4 :	c_{14}
4	Node 1 \rightarrow Node 5 :	c_{15}
5	Node 1 \rightarrow Node 6 :	c_{16}
6	Node 2 \rightarrow Node 3 :	c_{23}
7	Node 2 \rightarrow Node 4 :	c_{24}
8	Node 2 \rightarrow Node 5 :	c_{25}
9	Node 2 \rightarrow Node 6 :	c_{26}
10	Node 3 \rightarrow Node 4 :	c_{34}
11	Node 3 \rightarrow Node 5 :	c_{35}
12	Node 3 \rightarrow Node 6 :	c_{36}
13	Node 4 \rightarrow Node 5 :	c_{45}
14	Node 4 \rightarrow Node 6 :	c_{46}
15	Node 5 \rightarrow Node 6 :	c_{56}

different timetables, they should be taken as different line routes. A timetable consists of data concerning nodes, line routes, and the departure and arrival times of the transit vehicles. Let RF_k denote the frequency of R_k (i.e. how often a transit vehicle is dispatched in a line route) and $RS_i = RSD_i \cup RSA_i$ be all the line route segments associated with node i , where RSD_i and RSA_i represent the sets of line route segments that depart from and arrive at node i , respectively. For each route segment RS_i^l , $l = 1, \dots, |RS_i|$ associated with node $i \in N$, we record TRS_i^l as the time that the route segment RS_i^l takes place in node i . Thus $|TRS_i| = |RS_i| = |RSD_i| + |RSA_i|$. With timetable information embedded, different arrival and departure times for a node in a line route will create different line route segments. For example, referring to Table 4.2 and Table 4.4, there are 3 line route segments associated with node 1, which are $(m, n, r, t) = [(1, 2, 1, 8:01), (1, 2, 1, 8:11), (1, 2, 1, 8:21)]$ where m is the origin of each line route segment, n is the origin of each line route segment, r is the index of line route, and t is the departure or arrival associated with the line route segment.

To further simplify the problem, we assume that for any consecutive line route segments (i', i) and (i, i') on R_k , the arrival of (i', i) and departure of (i, i')

Table 4.4. Timetable information for each route

	Route 1			Route 2		
	Node 1	Node 2	Node 3	Node 4	Node 5	Node 6
1	8:01	8:21	8:30	8:00	8:23	8:30
2	8:11	8:31	8:40	8:15	8:38	8:45
3	8:21	8:41	8:50	8:30	8:53	9:00

happens at the same time at node i . As a result, we can remove some consecutive duplicated times in the vector TRS_i when node i is an intermediate node in R_k , and save the modified vector as TR_i . Note that $|TRS_i| = |TR_i| + \sum_{k: \text{node } i \text{ is an intermediate node in } R_k} RF_k$, since each time when a route passes through node i , either its arrival or departure time will be doubly counted in $|TRS_i|$. Furthermore, the departure time of each line route segment is always later than or equal to the preceding arrival time (except the first one, which has no preceding arrival time); that is $TRS_i^l \leq TRS_i^{l'}$ if $l < l'$. Thus any passenger alight a line route at some intermediate node will not be able to take any earlier dispatch at the same node.

4.2. Construction of a Multimodal Timetable-based Network

To calculate the itinerary with the shortest travel time in a transit network, one requires incorporating the temporal information into the transit network. To this end, here we propose a preprocessing algorithm to construct a time-space network called the multimodal timetable-based network, denoted by $\tilde{G} = (V, E)$, where V is a set of $|N|$ groups of nodes and $E = ED \cup EX \cup EW$ consists of the non-stop arcs (ED , arcs that are in the same line route), route-transfer arcs (EX , arcs that connect different line routes), and walk-transfer arcs (EW , arcs that allow travelers to walk from one node to another).

A node group $V_i \subseteq V$ for each $i \in N$ is composed by a set of $|TR_i|$ copies of node $i \in N$, where each copy corresponds to a specific time associated with a line route segment connecting i . Thus $|V| = \sum_{i \in N} |TR_i|$. For our convenience, four tuples, (i, r, TRS_i^l, Tx_i) , are used to describe each node $v \in V$ in the multimodal

timetable-based transportation network, where i corresponds to a physical location of node $i \in N$, r represents the index of the route that passes i , TRS_i^l records the time of the l^{th} route segment in RS_i arriving at or departing from node i (e.g. 8:15), and Tx_i is the time to alight a transit vehicle and then board another transit vehicle (e.g. 2 minutes) set by the users.

Each non-stop arc $(\alpha, \beta) \in ED$ represents each line route segment in $\bigcup_{i \in N} RS_i$, and its end nodes correspond to the end nodes of that line route segment at a specific time. That is, if $\alpha \in V_i$ and $\beta \in V_j$, then $i \neq j$. Furthermore, $|ED| = \sum_{i \in N} |RS_i|$. The orientation for a non-stop arc follows the sequence of the nodes appeared in its corresponding line route segment. Thus a non-stop arc is directed from a node of earlier time to a node of later time, and we set its length to be the duration of its corresponding line route segment, or in other words, the difference in time on its end nodes. On the other hand, each route-transfer arc $(\hat{\alpha}, \hat{\beta}) \in EX$ connects two nodes of the same physical location (i.e. if $\hat{\alpha} \in V_i$ and $\hat{\beta} \in V_j$, then $i = j$) but different times, as long as the time difference of end nodes does not exceed the time for route change. In particular, for each route-transfer arc $(\hat{\alpha}, \hat{\beta}) \in EX$ where $\hat{\alpha} = (i, r_1, TRS_i^{l_1}, Tx_i)$ and $\hat{\beta} = (i, r_2, TRS_i^{l_2}, Tx_i)$ correspond to some physical location $i \in N$, its orientation directs from u to v and $TRS_i^{l_1} + Tx_i \leq TRS_i^{l_2}$ and its length can be set as $TRS_i^{l_2} - TRS_i^{l_1}$. Therefore, the number of route-transfer arcs connecting nodes in $V_i \subseteq V$ is at most $O(|V_i|^2)$, and thus $|EX| = O(\sum_{i \in N} |V_i|^2)$.

The major difference between the basic timetable-based network introduced in previous chapter and the multimodal timetable-based network is the addition of walk-transfer arcs. In the multimodal timetable-based network, travelers are allowed to walk from one transit station to all other transit stations, and thus there are walk-transfer arcs between every two nodes with different physical locations. Take Figure 4.2 for example. There are two bus routes, Bus Route 1 and Bus Route 2; from station 1 on Bus Route 1, a passenger has the options to walk

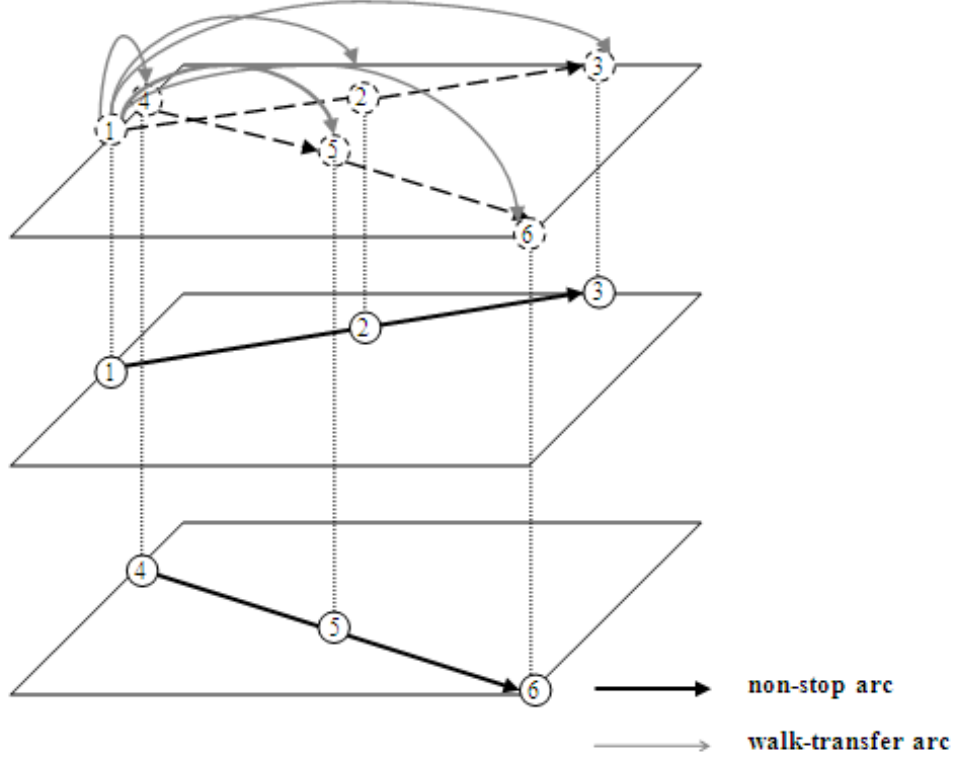


Figure 4.2. An illustration of walk-transfer arc

from station 1 to station 2, 3, 4, and 5, which are within the specified walking range. So each walk-transfer arc $(\acute{\alpha}, \acute{\beta}) \in EW$ connects two nodes of different physical locations (i.e. if $\acute{\alpha} \in V_i$ and $\acute{\beta} \in V_j$, then $i \neq j$.) and different times where the time difference of end node does not exceed the time required to walk between end nodes, and the length between each node does not exceed the specified walking range t_s . In particular, for each walk-transfer arc $(\acute{\alpha}, \acute{\beta}) \in EW$ where $\acute{\alpha} = (i, r_1, TRS_i^{l_a}, Tx_i)$ and $\acute{\beta} = (j, r_2, TRS_j^{l_b}, Tx_j)$ correspond to physical locations i and j , where $i, j \in N$ and $i \neq j$, and route number a and b , where $a, b \in K$, the distance between locations i and j can be denoted as c_{ij} , the speed of walking can be denoted as s (defined as a constant value according to assumption 4.4), the travel time between i and j by foot can be denoted as TW_{ij} where $TW_{ij} = \frac{|c_{ij}|}{|s|}$, and its orientation directs from u to v and $TRS_i^{l_a} + TW_{ij} \leq TRS_j^{l_b}$.

Now we give steps of our preprocessing algorithm CMTBN (Constructing the Multimodal Timetable-based Network) as follows:

Step 1: Read the spatial data (e.g. Table 4.2 and Table 4.3) and temporal data (e.g. Table 4.4), store data in $N, A, C, RS, RSD, RSA, TRS, TR$.

Step 2: Set acceptable walking range t_s .

Step 3: For each node $i \in V$, we store each node the defined four tuples (i, r, TRS_i^l, Tx_i) .

Step 4: Construct each non-stop arc $(\alpha, \beta) \in ED$, using V and RS .

Step 5: Construct each direct-transfer arc $(\hat{\alpha}, \hat{\beta}) \in EX$, using V .

Step 6: Construct each walk-transfer arc $(\acute{\alpha}, \acute{\beta}) \in EW$, using V, C , and RS .

4.3. Query and Solution Method on a Multimodal Timetable-based Network

In this section, we design an algorithm to solve the earliest arrival problem on the multimodal timetable-based network, and then demonstrate this procedure with the transit network shown in Figure 4.1. Since the timetable-based network is acyclic, we can apply topological-ordering-based algorithms to solve the shortest path problem.

4.3.1. Procedures

Depending on the locations of the requested origin o and destination d , as well as the starting time t_o of the query, we would like to identify a feasible itinerary of the minimum travel time from o to d in G . Here we give an algorithm called EAM (stands for Early Arrival for Multimodal timetable-based network) with steps as follows:

Step 1: Read the multimodal timetable-based network \tilde{G} , as well as all the related data structures.

Step 2: Read the indices of the requested origin o and destination d , and the starting time t_o .

Step 3: For each $(u, v) \in E$, where $u = (i, r, TRS_i^l, Tx_i)$ with $TRS_i^l < t_o$, remove it from E .

Step 4: Remove those route-transfer arcs whose both end nodes are in V_o or V_d .

Step 5: Construct a pseudo origin w_o , a pseudo destination w_d , artificial arcs (w_o, v_o) for each node $v_o = (o, r', TRS_o^l, Tx_o) \in V_o$ with nonnegative length equal to $(TRS_o^l - t_o)$, where $(TRS_o^l - t_o) \geq \frac{c_{w_o v_o}}{s}$, and artificial arcs (v_d, w_d) with nonnegative length equals to $\frac{c_{v_d w_d}}{s}$ for each node $v_d = (d, r', TRS_d^l, Tx_d) \in V_d$ with $TRS_d^l \geq t_o$.

Step 6: Solve a shortest path from w_o to w_d in \tilde{G} using any shortest path algorithm.

Step 7: Output the calculated shortest path, which corresponds to the quickest itinerary as requested.

In particular, based on the multimodal timetable-based network \tilde{G} , Step 3 first eliminates arcs that are too early for the user; Step 4 removes those route-transfer arcs inside the node groups corresponding to o and d since one can only changes line routes at intermediate nodes but not the origin and destination; Step 5 connects the pseudo origin and pseudo destination to their corresponding nodes that serve as the start and end of the request itinerary. Through these five steps, algorithm EAM has already incorporated the query information into the modified multimodal timetable-based network. Since the length of each arc in the modified multimodal timetable-based network represents either the duration of a route segment (for a non-stop arc), the waiting time to transfer routes (for a direct-transfer arc), the walking time to transfer routes (for a walk-transfer arc), or the waiting time and walking time to start the itinerary (for an artificial arc connecting to nodes in V_o), a shortest path in the modified multimodal timetable-based network thus corresponds to an itinerary with the minimal travel time from o to d starting from time t_o .

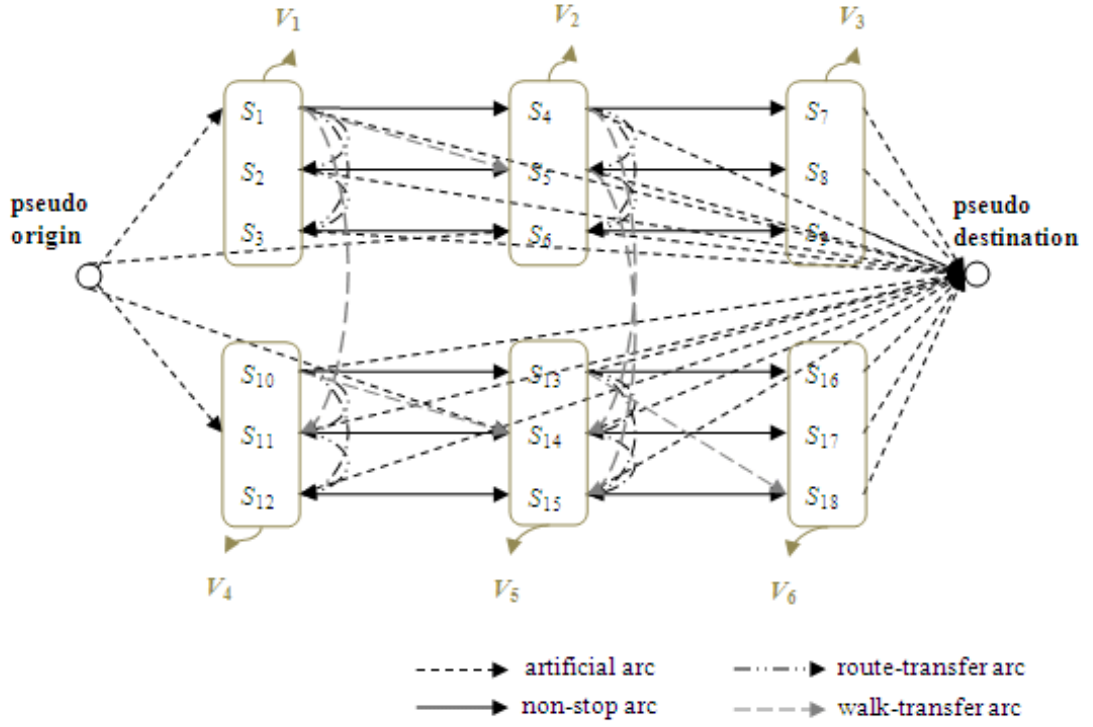


Figure 4.3. A multimodal timetable-based transit network

Since the artificial arcs added to \tilde{G} also direct from nodes of earlier time to nodes of later time, the modified multimodal timetable-based network remains acyclic, same as the basic timetable-based network. For seeking a shortest path in an acyclic diagram of $|V|$ nodes and $|E|$ arcs, the topological ordering algorithm [1] can be modified to solve a shortest path in $O(|E|)$ time, which is theoretically the most efficient since any shortest path algorithm takes at least $\Omega(|E|)$ time to read the input network.

4.3.2. An Illustrative Example

To demonstrate the formulation of our multimodal timetable-based transportation network, consider a transit network consisted of two transit routes and six nodes as shown in Figure 4.1 and Table 4.2. Table 4.3 gives the walking time between each node, and Table 4.4 gives the timetable of this transit system.

Table 4.5. Walking time for artificial arcs

<i>Segment</i>	<i>Time</i> (mins)	<i>Segment</i>	<i>Time</i> (mins)	<i>Segment</i>	<i>Time</i> (mins)
$o \rightarrow V_1$	2	$o \rightarrow V_3$	121	$o \rightarrow V_5$	71
$o \rightarrow V_2$	61	$o \rightarrow V_4$	11	$o \rightarrow V_6$	131
$V_1 \rightarrow d$	131	$V_3 \rightarrow d$	51	$V_5 \rightarrow d$	61
$V_2 \rightarrow d$	71	$V_4 \rightarrow d$	121	$V_6 \rightarrow d$	2

Table 4.6. Nodes information of multimodal timetable-based network

S_1	(1, 1, 8:01, 2)	S_7	(3, 1, 8:30, 2)	S_{13}	(5, 2, 8:23, 2)
S_2	(1, 1, 8:11, 2)	S_8	(3, 1, 8:40, 2)	S_{14}	(5, 2, 8:38, 2)
S_3	(1, 1, 8:21, 2)	S_9	(3, 1, 8:50, 2)	S_{15}	(5, 2, 8:53, 2)
S_4	(2, 1, 8:21, 2)	S_{10}	(4, 2, 8:00, 2)	S_{16}	(6, 2, 8:30, 2)
S_5	(2, 1, 8:31, 2)	S_{11}	(4, 2, 8:15, 2)	S_{17}	(6, 2, 8:45, 2)
S_6	(2, 1, 8:41, 2)	S_{12}	(4, 2, 8:30, 2)	S_{18}	(6, 2, 9:00, 2)

In this example, if a transfer is needed at any node, a two-minute waiting time is required. In other words, $Tx_i = 2$ for each node $i \in N$. Suppose a passenger plans to travel from location A to location B with a planned departure time of 8:00. Then, our algorithm EAM eliminates unqualified arcs from the multimodal timetable-based network that has been stored in the memory, adds pseudo nodes and artificial arcs to construct a modified multimodal timetable-based network as shown in Figure 4.3. The cost (time consumed) between pseudo node and transit stations are shown in Table 4.5. Table 4.6 demonstrates the information stored in each node. The algorithm then solves a shortest path from the pseudo origin to the pseudo destination. In this case, the itinerary with the minimum travel time will start from origin o on time 8:00, walk for 11 minutes before reaching station 4, wait for 4 minutes at the station, take the bus of route 2 on 8:15, get off the bus at node 6 on 8:45, walk for another 2 minutes, and then arrive at destination on 8:47. The total travel time is 47 minutes.

4.4. Computational Experiments

This section summarizes our computational results of multimodal timetable information problem with walking transfer. After introducing the implementation

settings, datasets from Taipei transit system will be used for the implementations. We will compare two implementation of shortest path algorithms: topological ordering algorithm, and Dijkstra's algorithm.

4.4.1. Settings and Problem Sets for the Implementation

All of our computational experiments are conducted using Microsoft Visual Studio 2005 on a Acer Aspire machine with an Intel Core 2 Duo processor at 1866 MHz and 2039 MB RAM running Microsoft Windows XP SP3.

We test our implementation on problem sets based on the bus datasets obtained from China Engineering Consultants Incorporation (CECI) and MRT datasets obtained from Taipei Rapid Transit Corporation (i.e. Taipei Metro). The bus datasets consists of bus route information and bus stop locations. The MRT datasets consists of MRT station location. With the locations of bus stops and MRT stations, we could estimate the distance between stops and stations. According to Institute of Transportation, MOTC [20], the average speed for a bus in Taipei is 22.88 kilometer per hour, and the average walking speed for a Taipei pedestrian is 4 kilometer per hour. With the average travel time between MRT station provided by Taipei Metro, we could estimate the length (travel time) of arcs in Taipei transit system.

Because Taipei transit system does not obey fixed time schedules in reality, we generate timetables for each bus stop or MRT station. We assume that Taipei transit system operates from 8 a.m. to 10 p.m.; in other words, the earliest available time for bus or MRT service is 8 a.m. and the last possible time for a passenger to board a bus or MRT train is 10 p.m.. The service intervals for each line route are randomly-generated values between 10 minutes and 30 minutes.

Table 4.7. Time to construct multimodal timetable-based network

Set	Routes	Nodes/Arcs (original)	Nodes/Arcs (expanded)	Time
P ₁	139	3392 / 11001	152317 / 3085427	73653
P ₂	160	3732 / 13183	159627 / 3820659	93015
P ₃	187	4117 / 14892	177180 / 4511358	101298
P ₄	202	4240 / 15854	180333 / 5096710	110960
P ₅	336	5451 / 25736	232613 / 6551773	138359

4.4.2. Algorithmic Running Time Comparison

We select 5 problem sets for computational experiments. Problem set P₁ consists of 139 line routes and 3392 transit stations. Problem set P₂ consists of 160 line routes and 3732 transit stations. Problem set P₃ consists of 187 line routes and 4117 transit stations. Problem sets P₄ consists of 202 line routes and 4240 transit stations. Problem set P₅, the complete network of Taipei transit system, consists of 336 line routes and 5451 transit stations. The number of nodes for the original transit network, the number of nodes for the basic timetable-based network, and the time to construct the basic timetable-based network are summarized in Table 4.7. The time to construct the network increases as the network size increases.

For each problem set, we select 10 groups of OD pairs. In each OD groups, there are 100 OD pairs with approximately same Euclidean distance. In addition, we conduct experiments on 5000 OD pairs with origins and destinations randomly selected. Each OD pair was checked by Depth-First Search (DFS) to assure feasible solutions before applying topological ordering algorithm. The unit of time for performance comparison is central processing unit time (CPU time). CPU time is the amount of time that a computer program consumes in processing central processing unit (CPU) instructions. The CPU time is often measured in clock ticks and is thereby used as a point of comparison for CPU usage of a program.

Two shortest path algorithms are implemented for computational experiments. The first algorithm is Dijkstra's algorithm with binary heap, denoted as *DIJ*. The second algorithm is a topological-ordering-based algorithm, denoted as *TO*. The

Table 4.8. Relative performance on five problem sets with randomly selected OD pairs

Set	DIJ	TO	SPD (=DFS+TOAD)	ΔT_D	SPB (=BFS+TOAB)	ΔT_B
P ₁	10569	7503	3495 (=1976+1519)	53.4%	3393 (=1993+1400)	54.7%
P ₂	10928	7750	3592 (=2026+1566)	53.6%	3598 (=2051+1547)	53.6%
P ₃	11169	8082	3750 (=2116+1634)	53.6%	3749 (=2153+1596)	53.6%
P ₄	11321	8294	3748 (=2084+1664)	54.8%	3636 (=2113+1523)	56.1%
P ₅	11973	8960	4096 (=2281+1815)	54.3%	4007 (=2301+1706)	55.3%
time unit in CPU click $\Delta T_D = (TO - SPD)/TO \times 100\%$ $\Delta T_B = (TO - SPB)/TO \times 100\%$						

results for problem set 5, the complete Taipei transit system, are summarized in Table 4.9 (SPD , DFS , $TOAD$, SPB , BFS , $TOAB$, ΔT_D and ΔT_B will be discussed in section 4.5). As for the computational performance on 5000 randomly selected OD pairs, the results are presented in Table 4.8. Results for other problem sets are shown in Appendix B.

We notice that topological-ordering-based algorithm performs better than Dijkstra's algorithm on basic timetable-based network. The reason why topological-ordering-based algorithm performs better than Dijkstra's algorithm is that the topological ordering algorithm [1] can be modified to solve a shortest path in $O(|E|)$ time while Dijkstra's algorithm takes $O(|E| + |V| \log |V|)$ time at best [11]. Generally, the result could be obtained within seconds in real time for topological-ordering-based algorithm.

4.5. Speed-up Techniques

We have successfully implemented two algorithms, topological-ordering-based algorithm and Dijkstra's algorithm, to solve multimodal timetable information problem. However, in order to operate on a mobile device, such as cellular phone or PDA, the performance of our current solution may not be efficient enough due to inferior CPU or less memory space equipped with mobile device. Facing these limitations, some speed-up techniques may be necessary.

One proposed speed-up technique is to find a feasible solution through DFS before applying shortest path algorithm, denoted as SPD. By finding a feasible, but

not necessarily the shortest, solution, we could set a boundary on our searching. Since the shortest path should have a cost lower than or equal to any feasible solution, we could eliminate the nodes that operate later the sum of intended departure time and the cost of feasible solution obtained through DFS. The complexity is the same, but the searched network size could be potentially much smaller. However, based on the nature of the timetable-based network, we can use Breadth-first Search (BFS) to find a feasible solution slightly more efficiently. Since a passenger will not transfer from one line route to another more than three times often according to our computational results, the feasible solution obtained from BFS could potentially be closer to the optimal solution than the feasible solution obtained from DFS. Theoretically, BFS may take longer to find a feasible solution than DFS but may eliminate more nodes through feasible solution closer to optimal solution. The size of network is smaller after applying BFS and thus the shortest path algorithm can find the optimal solution faster. The speed-up technique using BFS to find a feasible solution before applying shortest path algorithm is denoted as SPB.

Another proposed speed-up technique is finding the solution with longest travel time in the multimodal timetable-based network, denoted as SPL. If the longest possible travel time is known, we could eliminate the nodes that operate later than the sum of intended departure time and the longest possible travel time. The complexity is the same, but the searched network size could be smaller.

We hereby apply SPD and SPB on our 5 problem sets. Table 4.9 summarizes the implementation results of 10 groups of OD pairs in problem set 5. Results on other problem sets are shown in Appendix B. In Table 4.9, *DIJ* is the computational time for Dijkstra's algorithm, *TO* is the average computational time for topological ordering algorithm, *DFS* is the computational time to conduct DFS for an feasible solution and then set the boundary accordingly, *TOAD* is the computational time for topological ordering algorithm after the speed-up technique is

Table 4.9. Relative performance on problem set 5

OD Group	DIJ	TO	SPD (=DFS+TOAD)	ΔT_D	SPB (=BFS+TOAB)	ΔT_B
1	10126	7032	3223 (=1804+1419)	54.2%	3216 (=1839+1377)	54.3%
2	10478	7377	3387 (=1899+1488)	54.1%	3342 (=1903+1439)	54.7%
3	10748	7647	3499 (=1957+1542)	54.2%	3519 (=1991+1528)	54.0%
4	10994	7899	3614 (=2022+1593)	54.2%	3556 (=2045+1511)	55.0%
5	11345	8244	3775 (=2113+1661)	54.2%	3767 (=2123+1664)	54.3%
6	11773	8672	3966 (=2219+1747)	54.3%	3955 (=2246+1710)	54.4%
7	11850	8758	4001 (=2238+1764)	54.3%	3934 (=2264+1670)	55.1%
8	12235	9135	4169 (=2330+1840)	54.4%	4058 (=2336+1723)	55.6%
9	12562	9465	4327 (=2421+1906)	54.3%	4381 (=2476+1905)	53.7%
10	12908	9807	4474 (=2500+1974)	54.4%	4430 (=2540+1890)	54.8%
time unit in CPU click $\Delta T_D = (TO - SPD)/TO \times 100\%$ $\Delta T_B = (TO - SPB)/TO \times 100\%$						

applied, and *SPD* is the total computational time with speed-up technique, i.e. the total of *DFS* and *TOAD*. *BFS* is the computational time to conduct BFS for an feasible solution and then set the boundary accordingly, *TOAB* is the computational time for topological ordering algorithm after the speed-up technique is applied, and *SPB* is the total computational time with speed-up technique, i.e. the total of *BFS* and *TOAB*. In addition, we list ΔT_D , the percentage of time improvement from plain topological ordering algorithm to topological ordering algorithm with speed up technique using DFS (i.e. $\Delta T\% = (TO - SPD)/TO \times 100\%$) and ΔT_B , the percentage of time improvement from plain topological ordering algorithm to topological ordering algorithm with speed up technique using BFS (i.e. $\Delta T\% = (TO - SPB)/TO \times 100\%$) in the table.

By applying our first proposed speed-up technique using DFS, the average running time improves approximately 53% on problem set 5. By applying our speed-up technique using BFS, the average running time improves approximately 55%. We further normalize the results for each OD group in problem set 5 on the basis of SPD and summarize in Table 4.10. The ratio between time usage of DFS and that of TOAD remains approximately constant throughout all problem sets.

Table 4.10. Normalization of relative performance on problem set 5

OD Group	DIJ	TO	SPD (=DFS + TOAD)	SPB (=BFS + TOAB)
1	3.14	2.18	1.00 (=0.56 + 0.44)	1.00 (=0.57 + 0.43)
2	3.09	2.18	1.00 (=0.56 + 0.44)	0.99 (=0.56 + 0.42)
3	3.07	2.19	1.00 (=0.56 + 0.44)	1.01 (=0.57 + 0.44)
4	3.04	2.19	1.00 (=0.56 + 0.44)	0.98 (=0.57 + 0.42)
5	3.01	2.18	1.00 (=0.56 + 0.44)	1.00 (=0.56 + 0.44)
6	2.97	2.19	1.00 (=0.56 + 0.44)	1.00 (=0.57 + 0.43)
7	2.96	2.19	1.00 (=0.56 + 0.44)	0.98 (=0.57 + 0.42)
8	2.93	2.19	1.00 (=0.56 + 0.44)	0.97 (=0.56 + 0.41)
9	2.90	2.19	1.00 (=0.56 + 0.44)	1.01 (=0.57 + 0.44)
10	2.89	2.19	1.00 (=0.56 + 0.44)	0.99 (=0.57 + 0.42)

4.6. Summary

In this chapter, we solve the multimodal timetable information problem with walking transfer. In addition to the line route information of a transit system, our problem takes timetable and walking transfer into consideration. In order to incorporate timetable information and walking transfer, a new network, multimodal timetable-based network, is constructed through algorithm CMTBN. After constructing the multimodal timetable-based network, we then apply algorithm EAM with implementations of topological ordering algorithm and Dijkstra's algorithm. For each problem set, the number of arcs of its timetable-based network is much greater with walking transfer as a transportation means.

By conducting computational experiments on datasets from Taipei transit network, the results have shown reasonable efficiency on PC. Our computational experiments also support the theoretical projection that topological ordering algorithm should perform better than Dijkstra's algorithm. Furthermore, we propose two speed-up techniques to improve computational efficiency. The results of applying our first proposed speed-up technique, SPD, show a 53% running time improvement. and the results of applying speed-up technique using BFS, SPB, show a 55% running time improvement. Although the computational results indicate that SPB has a better efficiency than SPD, the improvement attributes the nature of Taipei transit system. Whether SPB has better computational efficiency

than SPD or not requires further validation. Compared with basic timetable information problem, it takes more time to solve multimodal timetable information problem with walking transfer. However, the actual performance of mobile device remains unknown.



CHAPTER 5

METHODOLOGIES ON FARE INFORMATION PROBLEMS

The nature of fare is very different from travel time, especially when its structure is non-linear. In order to find the cheapest route in a transit network, we need to construct a modified transit network with fare information embedded before applying shortest path algorithms. In order to simplify the problem, we need to make some assumptions:

Assumption 5.1: *In the transit network, the length (cost) between any two nodes is known and fixed.*

Assumption 5.2: *The length (cost) is exclusively defined as the travel fare in appropriate monetary units.*

Since walking is a costless transportation option, a trip planning system would generate itineraries composed of all walking with respect to lowest travel fare if walking is a transportation alternative. Therefore, we make a further assumption to avoid this scenario:

Assumption 5.3: *Walking is restricted to traversals between origin and transit stations and those between transit stations and destination without exceeding a certain range limitation. Traversing by foot is not a transportation means between transit stations in finding an optimal trip itinerary with lowest travel fare.*

Table 5.1. Nomenclature for fare information problem

A	set of arcs in transit network G
E	set of arcs in fare-based network \tilde{G}
ED	set of non-stop arcs in fare-based network \tilde{G}
EX	set of route-transfer arcs in fare-based network \tilde{G}
G	transit network; $G = (N, A)$
\tilde{G}	fare-based network; $\tilde{G} = (V, E)$
K	the number of line routes in G
N	set of nodes in transit network G
V	set of $ N $ groups of nodes in fare-based network \tilde{G}
V_i	a node group composed by a set of copies of node i , $i \in N$
V_l	a node group composed by a set of copies of node l , $l \in N$
V_n	a node group composed by a set of copies of node n , $n \in N$
c_{ij}	the travel cost between node i and node j , $i, j \in N$ and $i \neq j$
i	a node in transit network G ; $i \in N$
j	a node in transit network G ; $j \in N$
k	the index of line route in transit network G
n	number of nodes
v_d	origin node in transit network, $v_d = (d, r', m) \in V_d$
v_o	destination node in transit network, $v_o = (o, r', m) \in V_o$
w_d	pseudo destination constructed in algorithm LTF
w_o	pseudo origin constructed in algorithm LTF
α	the origin of a non-stop arc in fare-based network \tilde{G}
$\hat{\alpha}$	the origin of a route-transfer arc in fare-based network \tilde{G}
β	the destination of a non-stop arc in fare-based network \tilde{G}
$\hat{\beta}$	the destination of a route-transfer arc in fare-based network \tilde{G}

5.1. Spatial Data and Fare Data

Let digraph $G = (N, A)$ represent the transit network where N denotes the set of nodes and A denotes the set of arcs. A node in N represents some physical location where a transit vehicle can stop to pick up passengers, such as a bus stop, port, or train station. Each node in N belongs to at least one line route. A line route is a set of nodes which transfer is not required to travel between them.

A directed arc $(i, j) \in A$ represents a direct connection from node i to node j in a line route. Suppose that there are K line routes. Each transit line route k for $k = 1, \dots, K$ can be presented as an individual layer through HDP (see Figure 5.1, a transit network with MRT and Bus Route can be decomposed into two layers).

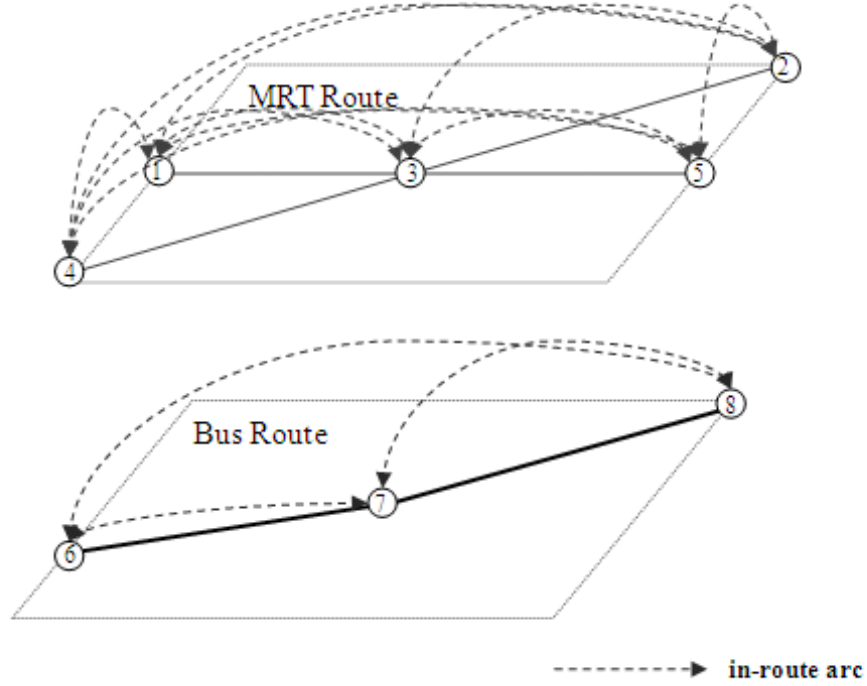


Figure 5.1. A hierarchical representation of a multimodal transit network

Take the multimodal transit network in Figure 5.2 for example, there are 6 physical locations and 2 transit line routes in different transit modes. The decomposed network is shown in Figure 5.1. For MRT (denoted as line route 1), it is composed by in-route arcs, $A_{MRT} = \{(i, j) \mid i, j = 1, 2, 3, 4, 5 \text{ and } i \neq j\}$, which make it a complete graph. For Bus Route (denoted as line route 2), it is composed by in-route arcs, $A_{BUS} = \{(i, j) \mid i, j = 6, 7, 8 \text{ and } i \neq j\}$, which make it a complete graph. These topological connection relations can be stored as spatial data. In comparison with a timetable-based network, we note that parallel arcs basically do not exist in each individual layer of a fare-based network. Since each transit route resides in its own layer, it is unlikely to have more than one route between two stops in each individual layer.

To calculate the cheapest path, one further requires the fare information referred as the fare data. The fare data consists of information concerning nodes, routes, and fare between nodes. Table 5.2 lists the fare data for the transit network

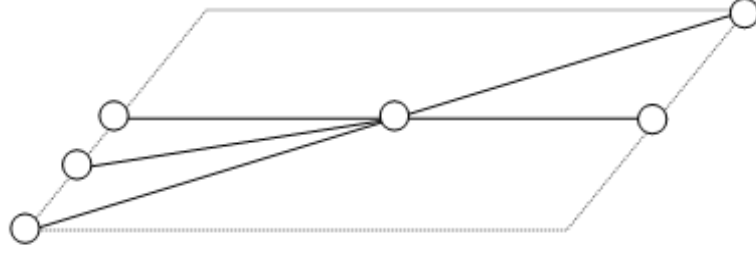


Figure 5.2. A multimodal transit network

Table 5.2. Fare data

Route No.	Route Segment	Fare	Route No.	Route Segment	Fare
Route 1	: Node 1 \rightarrow Node 2	: c_{12}	Route 1	: Node 2 \rightarrow Node 1	: c_{21}
Route 1	: Node 1 \rightarrow Node 3	: c_{13}	Route 1	: Node 2 \rightarrow Node 3	: c_{22}
Route 1	: Node 1 \rightarrow Node 4	: c_{14}	Route 1	: Node 2 \rightarrow Node 4	: c_{24}
Route 1	: Node 1 \rightarrow Node 5	: c_{15}	Route 1	: Node 2 \rightarrow Node 5	: c_{25}

Route No.	Route Segment	Fare	Route No.	Route Segment	Fare
Route 1	: Node 3 \rightarrow Node 1	: c_{31}	Route 1	: Node 4 \rightarrow Node 1	: c_{41}
Route 1	: Node 3 \rightarrow Node 2	: c_{32}	Route 1	: Node 4 \rightarrow Node 2	: c_{42}
Route 1	: Node 3 \rightarrow Node 4	: c_{34}	Route 1	: Node 4 \rightarrow Node 3	: c_{43}
Route 1	: Node 3 \rightarrow Node 5	: c_{35}	Route 1	: Node 4 \rightarrow Node 5	: c_{45}

Route No.	Route Segment	Fare	Route No.	Route Segment	Fare
Route 1	: Node 5 \rightarrow Node 1	: c_{51}	Route 2	: Node 6 \rightarrow Node 7	: c_{67}
Route 1	: Node 5 \rightarrow Node 2	: c_{52}	Route 2	: Node 6 \rightarrow Node 8	: c_{68}
Route 1	: Node 5 \rightarrow Node 3	: c_{53}	Route 2	: Node 7 \rightarrow Node 6	: c_{76}
Route 1	: Node 5 \rightarrow Node 4	: c_{54}	Route 2	: Node 7 \rightarrow Node 8	: c_{78}
			Route 2	: Node 8 \rightarrow Node 6	: c_{86}
			Route 2	: Node 8 \rightarrow Node 7	: c_{87}

in Figure 5.1. From assumption 2, the travel cost c_{ij} between node i and node j , $i, j \in N$ and $i \neq j$, are all known and fixed.

5.2. Construction of a Fare-based Network

To calculate the itinerary with the lowest travel cost in a transit network, one requires incorporating the fare information into the transit network. To this end, here we propose a preprocessing algorithm to construct a hierarchical network called the fare-based network, denoted by $\tilde{G} = (V, E)$, where V is a set of $|N|$ groups of nodes and $E = ED \cup EX$ consists of the non-stop arcs (ED , arcs that

are in the same line route) and route-transfer arcs (EX , arcs that connect different line routes).

A node group $V_i \subseteq V$, for each $i \in N$ is composed by a set of copies of node $i \in N$, where each copy corresponds to the same physical location of node i . For our convenience, three tuples, denoted by (i, r, m) , are used to describe each node $v \in V$ in the fare-based transportation network, where i corresponds to a physical location of node $i \in N$, r represents the index of the route that passes i , and m indicates the transit mode.

For each in-route arc $(\alpha, \beta) \in ED$, $\alpha \in V_l$ and $\beta \in V_n$ where $l \neq n$. While an in-route arc is directed from a node to another within a transit route, transfer arc $(\hat{\alpha}, \hat{\beta}) \in EX$ connects two nodes of the same physical location but on different transit routes. That is, if $\hat{\alpha} \in V_l$ and $\hat{\beta} \in V_n$, then $l = n$.

Now we give steps of our preprocessing algorithm **CTFBN** (Constructing the Fare-Based Network) as follows:

Step 1: Read the spatial data and fare data.

Step 2: For each node $i \in V$, we store each node the defined tuples (i, r, m) .

Step 3: Construct each in-route arc $(\alpha, \beta) \in ED$.

Step 4: Construct each transfer arc $(\hat{\alpha}, \hat{\beta}) \in EX$.

5.3. Query and Solution Method on Fare-based Networks

In this section, we design an algorithm to solve the lowest fare problem on the fare-based network, and then demonstrate this procedure with the simple transit network shown in Figure 5.1. In contrast with timetable-based network, the fare-based network is not acyclic, thus we will apply variants of Dijkstra's algorithms to solve the shortest path problem.

5.3.1. Procedures

Depending on the locations of the requested origin o and destination d of the query, we would like to identify a feasible itinerary of the minimum travel fare from o to d in \tilde{G} . Here we give an algorithm called **LTF** (stands for Lowest Travel Fare) with steps as follows:

- Step 1:** Read the fare-based network \tilde{G} , as well as all the related data structures.
- Step 2:** Read the indices of the requested origin o and destination d .
- Step 3:** Construct a pseudo origin w_o , a pseudo destination w_d , artificial arcs (w_o, v_o) for each node $v_o = (o, r', m) \in V_o$ and artificial arcs (v_d, w_d) for each node $v_d = (d, r', m) \in V_d$.
- Step 4:** Remove the artificial arcs whose lengths (costs) exceed the user-specified value.
- Step 5:** Solve a shortest path from w_o to w_d in \tilde{G} using any shortest path algorithm.
- Step 6:** Output the calculated shortest path, which corresponds to the cheapest itinerary as requested

5.3.2. An Illustrative Example

In order to demonstrate the formulation of our fare-based transportation network, consider a transit network made up of two transit routes and eight nodes as shown in Figure 5.1. Table 5.2 gives the fare data of this transit system.

In this example, a transfer can only occur at transit stations with same physical locations. In other words, transfer can only occur between node 3 and node 7 in this example. Suppose that node 4 and node 6 are within the user-specified range from origin, and node 2 and node 8 are within the user-specified range from destination. Our algorithm LTF reads in the fare-based transit network and adds pseudo nodes and artificial arcs to construct a modified fare-based transit network

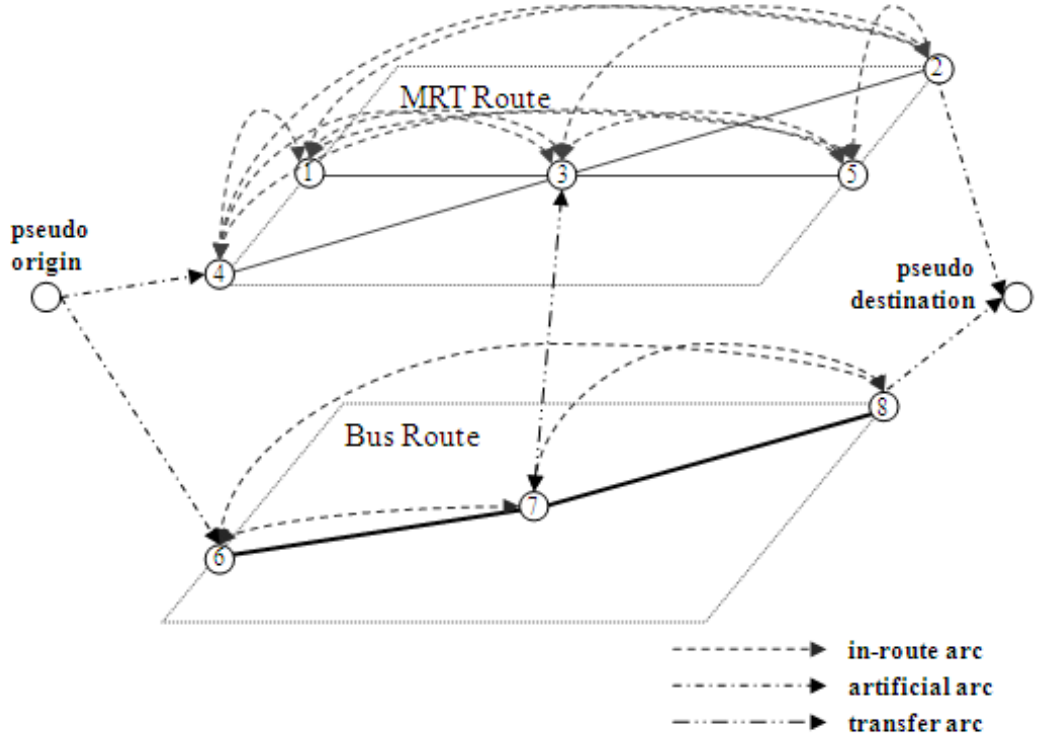


Figure 5.3. A fare-based multimodal transit network

Table 5.3. Node information of illustrative example

N_1	(1, 1, 1)	N_5	(5, 1, 1)
N_2	(2, 1, 1)	N_6	(6, 2, 2)
N_3	(3, 1, 1)	N_7	(7, 2, 2)
N_4	(4, 1, 1)	N_8	(8, 2, 2)

as shown in Figure 5.3, and solve a shortest path from the pseudo origin to the pseudo destination. In this case, the itinerary with the lowest travel fare will be starting from node 6, taking the bus route, and arriving at node 8. The total cost is \$3. Table 5.3 demonstrates the information stored in each node. Table 5.4 lists the fare data for this transit network.

5.4. Computational Experiments

This section summarizes our computational results of fare information problem. After introducing the implementation settings, datasets from Taipei transit system

Table 5.4. Fare data of illustrative example

in-route arcs of MRT

Route No.	Route Segment	Fare
Route 1	: Node 1 \rightarrow Node 2	: 10
Route 1	: Node 1 \rightarrow Node 3	: 5
Route 1	: Node 1 \rightarrow Node 4	: 10
Route 1	: Node 1 \rightarrow Node 5	: 10
Route 1	: Node 2 \rightarrow Node 1	: 10
Route 1	: Node 2 \rightarrow Node 3	: 5
Route 1	: Node 2 \rightarrow Node 4	: 10
Route 1	: Node 2 \rightarrow Node 5	: 10
Route 1	: Node 3 \rightarrow Node 1	: 5
Route 1	: Node 3 \rightarrow Node 2	: 5

Route No.	Route Segment	Fare
Route 1	: Node 3 \rightarrow Node 4	: 5
Route 1	: Node 3 \rightarrow Node 5	: 5
Route 1	: Node 4 \rightarrow Node 1	: 10
Route 1	: Node 4 \rightarrow Node 2	: 10
Route 1	: Node 4 \rightarrow Node 3	: 5
Route 1	: Node 4 \rightarrow Node 5	: 10
Route 1	: Node 5 \rightarrow Node 1	: 10
Route 1	: Node 5 \rightarrow Node 2	: 10
Route 1	: Node 5 \rightarrow Node 3	: 5
Route 1	: Node 5 \rightarrow Node 4	: 10

in-route arcs of Bus Route

Route No.	Route Segment	Fare
Route 2	: Node 6 \rightarrow Node 7	: 3
Route 2	: Node 6 \rightarrow Node 8	: 3
Route 2	: Node 7 \rightarrow Node 6	: 3

Route No.	Route Segment	Fare
Route 2	: Node 7 \rightarrow Node 8	: 3
Route 2	: Node 8 \rightarrow Node 6	: 3
Route 2	: Node 8 \rightarrow Node 7	: 3

transfer arcs from MRT to bus

Route Segment	Fare
Node 3 \rightarrow Node 7	: 0

transfer arcs from bus to MRT

Route Segment	Fare
Node 7 \rightarrow Node 3	: 0

:

artificial arcs from pseudo origin to nodes

Route Segment	Fare
$o \rightarrow$ Node 4	: 0

Route Segment	Fare
$o \rightarrow$ Node 6	: 0

artificial arcs from nodes to pseudo destination

Route Segment	Fare
Node 2 $\rightarrow d$: 0

Route Segment	Fare
Node 8 $\rightarrow d$: 0

will be used for the implementations. We will conduct our optimization process with Dijkstra's algorithm.

5.4.1. Settings and Problem Sets for the Implementation

All of our computational experiments are conducted using Microsoft Visual Studio 2005 on a Acer Aspire machine with an Intel Core 2 Duo processor at 1866 MHz and 2039 MB RAM running Microsoft Windows XP SP2.

We test our implementation on problem sets based on the bus datasets obtained from China Engineering Consultants Incorporation (CECI) and MRT datasets obtained from Taipei Rapid Transit Corporation (i.e. Taipei Metro). The bus datasets consists of bus route information and bus stop locations. The MRT datasets consists of MRT station location and the travel fare between each MRT station. Since the bus datasets do not include fare information, we assume that the travel fare between every bus stop on the same line route is a fixed number; in our case, for any journey on the same line route, the travel fare is set at \$15.

5.4.2. Algorithmic Running Time Comparison

We select 5 problem sets for computational experiments. Problem set P_1 consists of 139 line routes, 3392 transit stations, and 11001 arcs. Problem set P_2 consists of 160 line routes, 3732 transit stations, and 13183 arcs. Problem set P_3 consists of 187 line routes, 4117 transit stations, and 14892 arcs. Problem sets P_4 consists of 202 line routes, 4240 transit stations, and 15854 arcs. Problem set P_5 , the complete network of Taipei transit system, consists of 336 line routes, 5451 transit stations, and 25736 arcs. The number of arcs for the original transit network, the number of arcs for the fare-based network, and the time to construct the fare-based network are summarized in Table 5.5. The time to construct the network increases as the network size increases.

For each problem set, we select 10 groups of OD pairs. In each OD groups, there are 100 OD pairs with approximately same Euclidean distance. In addition, we conduct experiment on 5000 OD pairs with origins and destinations randomly selected. Each OD pair was checked by Depth-First Search (DFS) to assure feasible solutions before applying topological ordering algorithm. The unit of time for performance comparison is central processing unit time (CPU time). CPU time is the amount of time that a computer program consumes in processing central

Table 5.5. Time to construct fare-based network

Set	Routes	Nodes/Arcs (original)	Nodes/Arcs (expanded)	Time (in CPU time)
P ₁	139	3392 / 11001	3392 / 472772	14432
P ₂	160	3732 / 13183	3732 / 603696	19792
P ₃	187	4117 / 14892	4117 / 659275	20807
P ₄	202	4240 / 15854	4240 / 695337	21729
P ₅	336	5451 / 25736	5451 / 1002431	36274

Table 5.6. Average running time on problem set 5 for different OD group

<i>Problem Set</i>	<i>OD Group</i>	<i>DIJF</i>
P ₅	1	7098
P ₅	2	7306
P ₅	3	7624
P ₅	4	7797
P ₅	5	7977
P ₅	6	8218
P ₅	7	8511
P ₅	8	8619
P ₅	9	8773
P ₅	10	9033
time unit in CPU click		

processing unit (CPU) instructions. The CPU time is often measured in clock ticks and is thereby used as a point of comparison for CPU usage of a program.

Dijkstra's algorithm with binary heap, denoted as *DIJF*, is implemented for computational experiments. A binary heap data structure requires $O(\log n)$ time to perform insert, decrease-key and delete-min, and it requires $O(1)$ time for the other heap operations. Consequently, the binary heap version of Dijkstra's algorithm runs in $O(m \log n)$ time. The results for problem set 5, the complete Taipei transit system, are summarized in Table 5.6. As for the computational performance on 5000 randomly selected OD pairs, the results are presented in Table 5.7. Results for other problem sets are shown in Appendix C. We notice that the running time increases as the network size increases.

Table 5.7. Average running time on five problem set for randomly selected OD group

<i>Problem Set</i>	<i>DIJF</i>
P ₁	4796
P ₂	5009
P ₃	5108
P ₄	5252
P ₅	7589
time unit in CPU click	

5.5. Alternative Fare Models

In section 5.2, we propose a fare-based network to solve all fare problems. In this general fare-based network, arcs are created so that nodes within a line route are strongly connected. However, enumerating all possible costs between each node is not the most efficient technique to find cheapest route in Taipei transit system. For Taipei transit system, we develop an alternative fare model to cope with two different situations and thereby reduce the number of arcs and reassign arc costs to improve computational efficiency.

5.5.1. Fixed Fare Rate and Variable Fare Rate

In Taipei transit system, there are two major categories of fare rates. The first major category is fixed fare rate. Buses in Taipei transit system charge passengers a fixed fare rate. According to Department of Transportation of Taipei City Government, the fare for single leg journey is \$15 for general public, \$12 for students, \$8 for the disabled and children. The second major category is variable fare rate. MRT in Taipei transit system charges passengers a variable fare rate. For example, if a MRT passenger decides to travel from Taipei Main Station of Danshui Line to Taipei City Hall of Nangang Line, it will cost him \$20, but if a MRT passenger decides to travel from Taipei Main Station of Danshui Line to Qilian of Danshui Line, it will cost him \$30.

Therefore, the alternative fare model handles line routes with variable fare rate and fixed fare rate separately. For variable fare rate, we still use the same

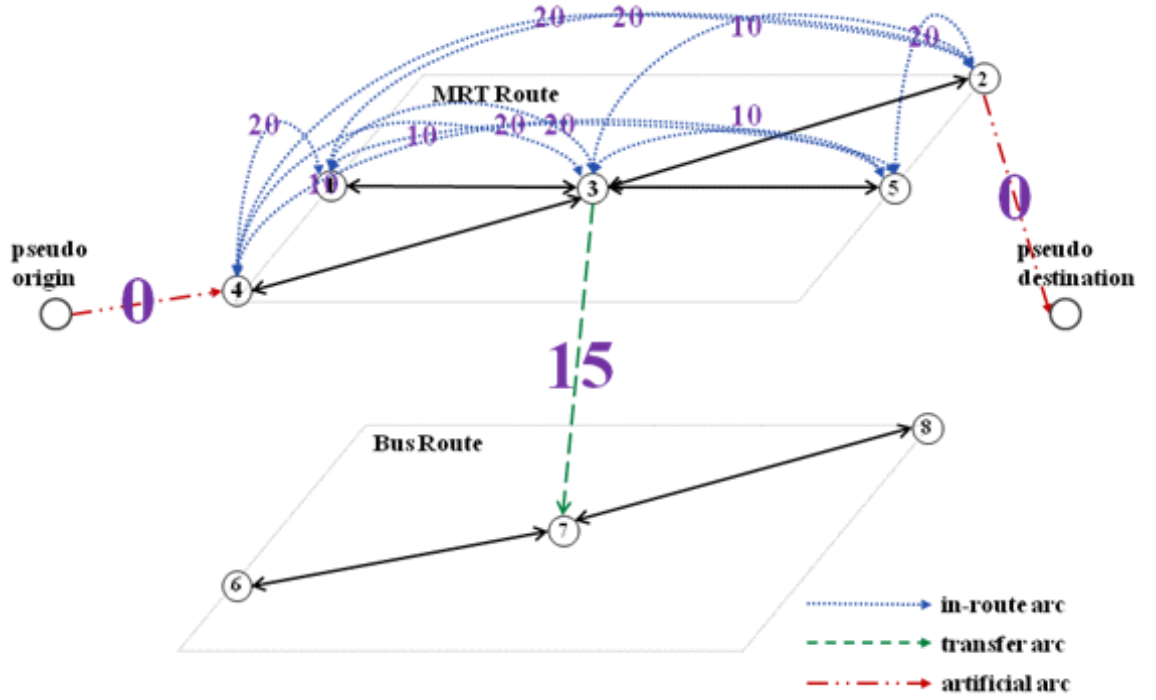


Figure 5.4. Network construction for variable fare rate in alternative fare model

procedures in CTFBN to construct in-route arcs and artificial arcs. But there will be costs assigned on transfer arcs which a traveler leaves a variable-fare-rate line route for a fixed-fare rate line route. On the other hand, for the transfer arcs where a traveler leaves a variable-fare-rate line route for another variable-fare-rate line route, there will be no cost assigned. Figure 5.4 demonstrates this construction with the assumption that MRT has a variable fixed rate and bus has a fixed fare rate of \$15.

For fixed fare rate, the costs are assigned on artificial arcs, and there will be no cost on in-route arcs. However, if a traveler leaves a fixed-fare-rate line route for a variable-fare-rate line route, there will be no cost assigned on the transfer arcs. On the other hand, for transfer arcs where a traveler leave a fixed-fare-rate line route for another fixed-fare-rate line route, there will be costs assigned on them. Figure 5.5 demonstrates this construction with the assumption that MRT has a variable fixed rate and bus has a fixed fare rate of \$15.

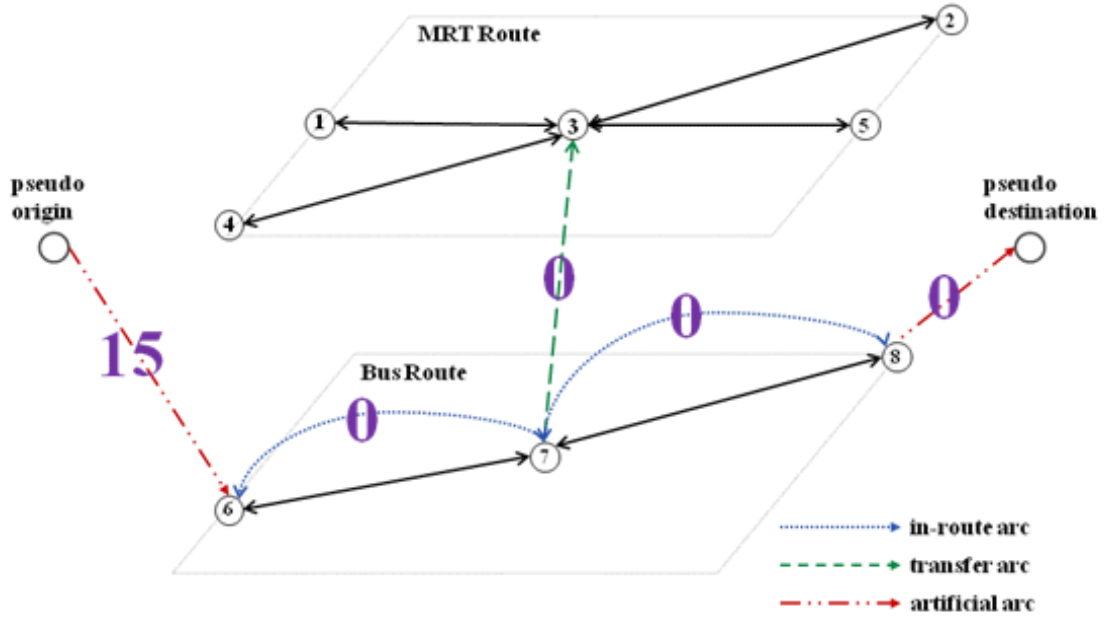


Figure 5.5. Network construction for fixed fare rate in alternative fare model

Through this approach, in-route arcs connect only adjacent nodes within the same line route in contrast with forming a complete graph. If a line route has n nodes, there will be $n - 1$ arcs in the alternative fare model and there will be $\frac{n(n-1)}{2}$ in the general fare model. The number of nodes and the number of transfers arcs and artificial arcs are the same in both general fare model and alternative fare model. The number of arcs thus reduces significantly. In addition, since most of the arcs belong to line routes with fixed fare rate such as bus, a great number of in-route arcs will have zero cost in the alternative fare model. With the implementation of Dijkstra's shortest path algorithm, the running time decreases as the number of zero-cost arcs increases.

5.5.2. Transfer Discount

In Taipei transit system, the first bus transfer after a MRT trip within an hour is free. For example, if you take MRT to travel from Taipei Main Station of Danshui Line to Taipei City Hall of Nangang Line, and then switch to a bus to travel from

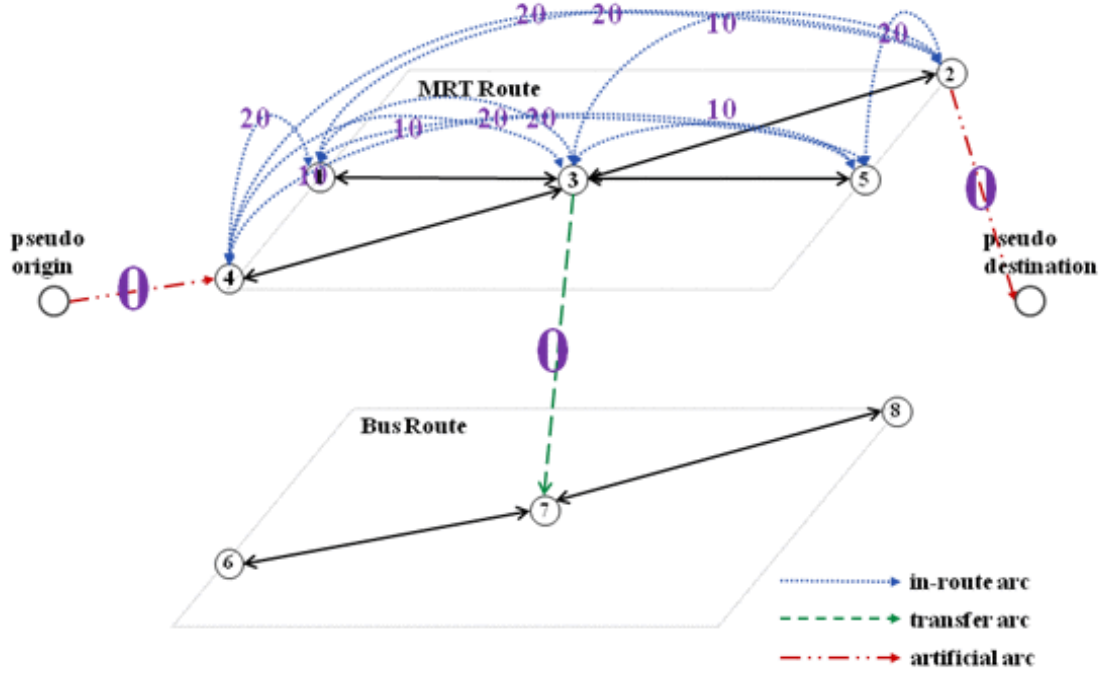


Figure 5.6. Modification on arc cost of transfer arc connecting MRT to bus

Taipei City Hall to National Palace Museum, the bus ride will be free as long as the line route consists of single leg. In this case, we will change the costs to zero on arcs connecting a MRT line route to bus line route. Figure 5.6 illustrates this modification.

Similarly, we can apply this technique to any transfer discount.

5.6. Computational Experiments on Alternative Fare Models

This section summarizes our computational results of fare information problem using the alternative fare model. After introducing the implementation settings, datasets from Taipei transit system will be used for the implementations. We will conduct our optimization process with Dijkstra's algorithm.

5.6.1. Settings and Problem Sets for the Implementation

All of our computational experiments are conducted using Microsoft Visual Studio 2005 on a Acer Aspire machine with an Intel Core 2 Duo processor at 1866 MHz and 2039 MB RAM running Microsoft Windows XP SP2.

We test our implementation on problem sets based on the bus datasets obtained from China Engineering Consultants Incorporation (CECI) and MRT datasets obtained from Taipei Rapid Transit Corporation (i.e. Taipei Metro). The bus datasets consists of bus route information and bus stop locations. The MRT datasets consists of MRT station location and the travel fare between each MRT station. Since the bus datasets do not include fare information, we assume that the travel fare between every bus stop on the same line route is a fixed number; in our case, for any journey on the same line route, the travel fare is set at \$15.

5.6.2. Algorithmic Running Time Comparison

We select 5 problem sets for computational experiments. Problem set P_1 consists of 139 line routes, 3392 transit stations, and 11001 arcs. Problem set P_2 consists of 160 line routes, 3732 transit stations, and 13183 arcs. Problem set P_3 consists of 187 line routes, 4117 transit stations, and 14892 arcs. Problem sets P_4 consists of 202 line routes, 4240 transit stations, and 15854 arcs. Problem set P_5 , the complete network of Taipei transit system, consists of 336 line routes, 5451 transit stations, and 25736 arcs.

For each problem set, we select 10 groups of OD pairs. In each OD groups, there are 100 OD pairs with approximately same Euclidean distance. In addition, we conduct experiment on 5000 OD pairs with origins and destinations randomly selected. Each OD pair was checked by Depth-First Search (DFS) to assure feasible solutions before applying topological ordering algorithm. The unit of time for performance comparison is central processing unit time (CPU time). CPU time is the amount of time that a computer program consumes in processing central

Table 5.8. Average running time using alternative fare model on problem set 5 for different OD group

<i>Problem Set</i>	<i>OD Group</i>	<i>DIAT</i>
P ₅	1	3069
P ₅	2	3258
P ₅	3	3418
P ₅	4	3503
P ₅	5	3621
P ₅	6	3792
P ₅	7	3886
P ₅	8	3927
P ₅	9	4072
P ₅	10	4163
time unit in CPU click		

Table 5.9. Average running time using alternative fare model on five problem set for randomly selected OD group

<i>Problem Set</i>	<i>DIAT</i>
P ₁	2710
P ₂	2796
P ₃	2851
P ₄	2961
P ₅	3204
time unit in CPU click	

processing unit (CPU) instructions. The CPU time is often measured in clock ticks and is thereby used as a point of comparison for CPU usage of a program.

Dijkstra’s algorithm with binary heap, denoted as *DIAT*, is implemented for computational experiments. A binary heap data structure requires $O(\log n)$ time to perform insert, decrease-key and delete-min, and it requires $O(1)$ time for the other heap operations. Consequently, the binary heap version of Dijkstra’s algorithm runs in $O(m \log n)$ time. The results for problem set 5, the complete Taipei transit system, are summarized in Table 5.8. As for the computational performance on 5000 randomly selected OD pairs, the results are presented in Table 5.9. Results for other problem sets are shown in Appendix D. We notice that the running time reduces significantly in comparison with the general fare model.

5.7. Summary

In this chapter, we solve the fare information problem. In addition to the line route information of a transit system, our problem takes fare information into consideration. In order to incorporate fare information, a new network, fare-based network, is constructed through algorithm CTFBN. After constructing the basic timetable-based network, we then apply algorithm LTF with implementations of Dijkstra's shortest path algorithm with binary heap. We also propose an alternative fare model to serve Taipei transit system more efficiently. Our alternative model handles line routes with variable fare rate and fixed fare rate separately, and thus reduces the number of arcs and reassigns some arcs with zero costs. Moreover, we can implant transfer discounts in the alternative fare model by changing the arc costs on transfer arcs..

By conducting computational experiments on datasets from Taipei transit network, the results have shown reasonable efficiency on PC. Furthermore, we reduces the running time by using the alternative fare model. The reduction attributes to the reduction of arcs and reassignment of arc costs. However, our method may not be efficient enough to implement on a application server, which is a software engine that delivers applications to client computers or devices, typically through the Internet and using the HyperText Transfer Protocol (HTTP), let alone mobile devices. Application servers are distinguished from web servers by the extensive use of server-side dynamic content and frequent integration with database engines. To expand the utilization of our method, we can research into more speed-up techniques in the future.

CHAPTER 6

CONCLUSIONS AND FUTURE RESEARCH

This chapter concludes the thesis by highlighting our contributions in Section 6.1, and then suggesting some potential directions for future research in computational efficiency, system building and their related problems in Section 6.2.

6.1. Summary and Contributions

This thesis contains frameworks for generating itinerary for passengers in a transit network when timetable information or fare information are available. During the past five decades, many new methods and applications related to transportation and transit system have been proposed and researched, but few were in regard to timetable information and fare information.

Assuming that a passenger's objective is to minimize the total travel time to the intended destination in a transit system, we consider two situations" basic timetable information problem, and multimodal timetable information problem with walking transfer.

In Chapter 3, we discuss the basic timetable information problem. In this problem, a passenger can only travel from one transit station to another by transit vehicle. In other words, our proposed solution needs to generate an itinerary with minimal travel time while walking is not a transportation means. Through our solution method, we first construct a timetable-based network. Since the timetable-based network is acyclic, we then use a topological-ordering-based algorithm inspired by Ahuja et al. [1] to find the shortest path. Although shortest path problem for transit system has been researched for many years, to the best of our knowledge, there exists no topological-ordering-based algorithm designed specifically for solving timetable information problems.

Numerical results are presented to illustrate the efficiency of our proposed method; the results have shown reasonable efficiency on PC. In addition, by showing the numerical results of classical shortest path algorithm, Dijkstra's algorithm, we demonstrate the disparity in efficiency of Dijkstra's algorithm and topological ordering algorithm. The computational results match the theoretical prediction that topological ordering algorithm performs more efficiently than Dijkstra's algorithm. For application servers or mobile devices, the CPU and memory may be inferior to those of PC. We thus proposed two speed-up techniques to improve the efficiency of our method. The computational results of our first speed-up technique, which applied Depth-first Search to find a feasible solution and set an upper boundary to the posterior topological ordering algorithm, indicate improvement of more than 50% on running time in comparison to the plain topological ordering algorithm.

In Chapter 4, we discuss the multimodal timetable information problem with walking transfer. In this problem, a passengers is allowed to walk from one transit station to another. Compared with basic timetable information problem, the number of arcs greatly increases in this problem. Analogous to basic timetable information problem, our proposed solution method first constructs an acyclic timetable-based network. Then we apply a topological-ordering-based algorithm to solve the shortest path problem.

Finally, in Chapter 5 we discuss the fare information problem where a passenger's objective is to minimize the total travel cost to the intended destination in a transit system. Since listing walking as a transportation means will result in all-walking itinerary, we exclude walking from our problem. Through our solution method, we first construct a fare-based network. With no timetable information embedding, the fare-based network is not acyclic. Therefore, we implement Dijkstra's algorithm with binary heap to solve the shortest path problem. The numerical results indicate reasonable efficiency on PC. However, we also propose

an alternative fare model to further improve the computational efficiency. Our alternative fare model reduces the number of in-route arcs while the number of transfer arcs and artificial arcs remain the same.

Since travel distance, travel time, and travel cost are some of the most important concerns for a transit passenger, designing solution methods for timetable information problem and fare information problem, as we have researched in Chapter 3, 4, and 5, is rather crucial.

6.2. Applications on Different Platforms

In this section, we summarize the differences between two platforms, computers and mobile devices. We then indicate how to implement our model to generate itineraries on different platforms.

6.2.1. Platforms

On a sparse network, such as the timetable-based network or fare-based network in the thesis, single-source shortest path algorithm is more suitable than all-pairs shortest path. Therefore, our research focuses on improving computational efficiency of single-source shortest path algorithm. However, different platforms may require different types of shortest path algorithms in correspondence with platform environments.

A mobile device, also known converged device, handheld device, or simply handheld, is a pocket-sized computing device, typically having a display screen with touch input or a miniature keyboard. Smartphones and PDAs are most common among those who require the assistance and convenience of a conventional computer in environments where carrying one would not be practical. Although most smartphones and PDAs have both color screens, audio capabilities, and internet connection, most of them are equipped with CPU and memory inferior to computers. Many smartphones and PDAs run using a variation of the ARM

architecture. The ARM architecture is a 32-bit RISC (Reduced instruction set computer) microprocessors that are also widely used in mobile devices and embedded systems. For example, one of the most advanced smartphones, iPhone by Apple Inc., runs a 667MHz ARM1176JZF-based CPU made by Samsung, which is a SIMD (Single instruction, multiple data), high perf integer CPU with 8-stage pipeline, 675 Dhryston, and 2.1 MIPS (Million instructions per second). The CPU for iPhone clocks much slower than the CPU for PC, such as a typical Intel Core 2 Duo Processor E8500 which clocks at 3.16 GHz. With slower CPU, smaller memory space, and limited battery life, a mobile device thus focuses on computational efficiency so that each operation will not consume too much energy. In other words, we can consume less energy by finishing the operation quicker.

On the other hand, a personal computer (PC) may be a desktop computer, a laptop computer, or a tablet computer. The most common operation systems are Microsoft Windows, Mac OS, and Linux, while the most common microprocessors are the x86 and PowerPC CPUs. The distinguishing characteristics of PC are that the computer is primarily used, interactively, by one person at a time. This is contrast to the batch-processing or time-sharing servers which allow the systems to be used by many people, usually at the same time.

In information technology, a server is a device that performs for connected clients as part of the client-server architecture. Server computers are devices designed to run server application that accepts connections in order to service requests by sending back responses, often for extended periods of time with minimal human direction and maintenance. Under light loading, every server application can run concurrently on a single server computer and under heavy loading, multiple server computers may be required for each application. Although servers can be built from commodity computer components, dedicated, high-load, mission-critical servers use specialized hardware that is optimized for the needs of servers. CPU speeds are far less critical for many servers than they are for many PCs.

Not only are typical server tasks likely to be delayed more by Input/Output (I/O) requests than processor requirements, but also the lack of any graphical user interface (GUI) in many servers frees up great amount of processing power for other tasks, making the overall processor power requirement lower. The lack of GUI in a server makes it unnecessary to install expensive video adapters. Similarly, elaborate audio interfaces, joystick connections, USB (universal serial bus) peripherals, and the like are usually unnecessary. Typical servers include heavy-duty network connections in order to allow them to handle the large amounts of traffic that they typically receive and generate as they receive and reply to client requests. In brief, the key characteristic of servers is to handle I/O requests as soon as possible.

6.2.2. Shortest Path Algorithms

On a mobile device, the memory space is limited. Therefore, it is not practical to use an all-pairs shortest path algorithm to obtain every possible result and store them in the database so that the software application can simply display a stored result upon user query. With searching for shortest path upon every user query as the only option, improving computational efficiency to conserve battery life with limited memory space is the key factor. We thus propose several speed-up techniques to reduce application running time. However, instead of single-source shortest path algorithm, denoted as SSSPA, we can apply all-pairs shortest path algorithm, denoted as APSPA, on a different platform, such as computers.

Computers usually have larger disk space for data storage, and thus we can use an all-pairs shortest path algorithm to find all results and store them into database if the network has moderate size, such as Taiwan railway network. The application will seize results from database when a user request an itinerary. Under this architecture, the application does not need to compute shortest paths upon query, and thus the running time for each query is a constant time $O(1)$ while the running time for each query is $O(E_T)$, where $\tilde{G}_T = (V_T, E_T)$ represents an

Table 6.1. Comparison between single-source shortest path and all-pairs shortest path algorithms for timetable problems

	SSSPA	APSPA
Storage space	$O(V_T + E_T)$	$O(FV_T^2)$
Preprocessing time	0	$O(FV_TE_T)$
Query time	$O(E_T)$	$O(1)$

Table 6.2. Comparison between single-source shortest path and all-pairs shortest path algorithms for fare problem

	SSSPA	APSPA
Storage space	$O(V_F + E_F)$	$O(V_F^2)$
Preprocessing time	0	$O(V_F(E_F + V_F \log V_F))$
Query time	$O(E_F + V_F \log V_F)$	$O(1)$

acyclic timetable-based network, by using the topological ordering algorithm for timetable information problems. However, the preprocessing time for SSSPA and APSPA are different.

We can solve an all-pairs shortest-paths problem by running a single-source shortest-path algorithm V_T times, once for each vertex as the source. If all edge weights are nonnegative, we can use a single-origin-all-destinations algorithm separately for each origin. For all-pairs problem, topological ordering algorithm yields a running time of $O(V_TE_T)$. In addition, we have to take departure time into consideration for timetable information problem. Since the passenger departure time is unknown, we need to enumerate itineraries for all possible departure time. Let F be the collection of time when a transit vehicle leaves a transit station, and thus the total preprocessing time to solve the all-pairs shortest path problem for timetable information problems is $O(FV_TE_T)$ while SSSPA requires no preprocessing time.

The storage space for data is $O(V_T + E_T)$ for SSSPA since we only need to store the node and arc information of the timetable-based network. On the other hand, the space required to store all possible itineraries is $O(FV_T^2)$ because there are V_T^2 OD pairs for each possible departure time. The comparison is concluded in Table 6.1.

As for fare information problem, timetable information has been excluded. Therefore, the preprocessing time for fare information problem by using APSPA is $O(V_F(E_F + V_F \log V_F))$, where $\tilde{G}_F = (V_F, E_F)$ represents a fare-based network, while SSSPA requires no preprocessing time. The running time for each query for fare information problem remains a constant time $O(1)$ by using APSPA and $O(E_F + V_F \log V_F)$ by using a Fibonacci heap implementation of Dijkstra's algorithm since the topological ordering algorithm is not applicable to fare information problem. The space required to store all possible itineraries by using APSPA is $O(V_F^2)$ while using SSSPA only takes $O(V_F + E_F)$. The comparison is concluded in Table 6.2.

6.3. Future Research

In this section, we propose some interesting directions for future research.

6.3.1. Multiobjective Shortest Path Problem

Due to the multiobjective nature of many optimization problems, mainly in the area of transportation problem, in recent years there has been an increase in research on multiobjective shortest path problem, with goals of relevant interest, like the minimization of cost, time, unreliability, etc. When a budget of various resources is given, some objectives related to the corresponding resources can be treated as constraints for the problem.

It is usually assumed that in multiobjective analysis, the objectives are in conflict; therefore, in general, there is no single optimal solution, but rather a set of nondominated or noninferior solutions, denoted Pareto optimal solutions, from which the decision maker must select the most preferred one, or the best compromise solution. There are several approaches used for exploring the Pareto optimal solution set. More specifically, we can distinguish the following three categories:

generating methods, methods based on utility functions, and interactive methods. In these methods, the multiobjective shortest path problem becomes a single objective shortest path problem.

Although the multiobjective shortest path problem fits the reality that a passenger may be concerned for both travel time and travel cost, it is difficult to, taking methods based on utility functions for example, assign the coefficients for utility functions individually. Nevertheless, displaying an itinerary fulfilling multiple objectives may be useful in the real world.

6.3.2. Dynamic Information

Automatic vehicle location (AVL) systems based on global positioning systems (GPS) have been widely adopted by many transit systems to monitor the movements of transit vehicles on a real-time basis. In our research, we assume that transit vehicles travel in the absence of congestion. As a result, when predicting bus arrival times at subsequent bus stops, the delay incurred at one stop will be carried forward in the downstream direction. In reality, skilled bus operators sometimes adjust their speeds in order to keep their buses on schedule. Therefore, prediction accuracy incorporating real-time information for an itinerary planning system is an issue that should be considered in the context of specific requirements from the perspectives of system users and system suppliers.

It would be useful to provide a passenger with an itinerary using both static information and dynamic information. Static information refers to bus schedule information, historical information of traffic conditions, etc. Dynamic information includes real-time bus location data, delay at bus stops, weather condition, current traffic condition, etc. However, the amount of information could be too enormous for a mobile device or an application server to process. In reality, it is unrealistic to carry a laptop equipped with Internet access all the time; therefore,

the computational efficiency needs further improvement to incorporate dynamic information on a mobile device.

6.3.3. Geographic Information System

One of the recent development in GIS technology is to deliver GIS data and analysis functions on the Web through the Internet. Internet GIS, an emerging technology to serve GIS data and GIS functionality on the Web, is designed to integrate the Web and GIS in order to manipulate, visualize, and analyze GIS data on the Web. A transit information system could be composed of the Web browser, Web server, and one or more application servers. The Web browser is a user interface to collect user input. The Web server serves as a middleware to handle user's request and transfer the request to an application server. The application server is used to process user requests. The application server could be composed of three different components, a map server, a network analysis server, and a database server. The map server is designed for map rendering and spatial analysis; the network analysis server is used to provide network analysis functions, such as shortest path algorithms; and the database server is used to handle data management via DBMSs.

A Web-based transit information system could integrate Internet GIS into it so that the user interface is map-based. The user can thus interact with the transit network and street maps, conducting query, search, and map rendering. The interactive map-based user interface also allows the system to incorporate other information, such as shops, theaters, parks, and other local attractions. This is very important for visitors who may want to explore these sites around their destinations. However, building a transit information system is beyond the scope of this thesis, and can be exploited as a future research.

References

- [1] Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (1993). *Network flows: theory, algorithms and applications*. Englewood Cliffs, NJ: Prentice Hall.
- [2] Ahuja, R. K., Mehlhorn, K., Orlin, J. B., & Tarjan, R. E. (1990). Faster algorithms for the shortest path problem. *Journal of the ACM*, 37(2), 213-223.
- [3] Cathey, F. W., & Dailey, D. J. (2003). A prescription for transit arrival/departure prediction using automatic vehicle location data. *Transportation Research Part C*, 11(3-4), 241-264.
- [4] Caulfield, B., & O'Mahony, M. (2007). An examination of the public transport information requirements of users. *IEEE Transactions on Intelligent Transportation Systems*, 8(1), 21-30.
- [5] Cheng, S.-T., Liu, J.-P., Kao, J.-L., & Chen, C.-M. (2002). *A new framework for mobile web services*. Paper presented at the Applications and the Internet (SAINT) Workshops, Nara City, Japan.
- [6] Cherkassky, B. V., Goldberg, A. V., & Radzik, T. (1996). Shortest paths algorithms: theory and experimental evaluation. *Mathematical Programming*, 73(2), 129-174.
- [7] Chriqui, C. (1975). Common bus lines. *Transportation Science*, 9(2), 115-121.
- [8] Corman, T. H., Leiserson, C. E., & Rivest, R. L. (1990). *Introduction to algorithms*. Cambridge, MA: MIT Press.
- [9] Dial, R. B. (1969). Algorithm 360: shortest-path forest with topological ordering. *Communications of the ACM*, 12(11), 632-633.
- [10] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1), 269-271.
- [11] Fredman, M. L., & Tarjan, R. E. (1987). Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3), 596-615.
- [12] Friedrich, M., Hofsaess, I., & Wekeck, S. (2001). Timetable-based transit assignment using brand and bound techniques. *Journal of the Transportation Research Board: Transportation Research Records*, 1752, 100-107.

- [13] Gentile, G., Nguyen, S., & Pallottino, S. (2005). Route choice on transit networks with online information at stops. *Transportation Science*, 39(3), 289-297.
- [14] Goldberg, A. V., & Radzik, T. (1993). A heuristic improvement of the Bellman-Ford algorithm. *Applied Mathematical Letters*, 6(3), 3-6.
- [15] Granat, J., & Guerriero, F. (2003). The interactive analysis of the multicriteria shortest path problem by the reference point method. *European Journal of Operational Research*, 151(1), 103-118.
- [16] Hall, R. W. (1996). Route choice and advanced traveler information systems on a capacitated and dynamic network. *Transportation Research Part C*, 4(5), 289-306.
- [17] Hickman, M. (2002). *Robust passenger itinerary planning using transit AVL data*. Paper presented at the IEEE 5th International Conference on Intelligent Transportation Systems, Singapore.
- [18] Horn, M. E. T. (2004). Procedures for planning multi-leg journeys with fixed-route and demand-responsive passenger transport services. *Transportation Research Part C*, 12(3-4), 33-55.
- [19] Huang, R., & Peng, Z.-R. (2002). Schedule-based path-finding algorithms for transit trip-planning systems. *Journal of the Transportation Research Board: Transportation Research Records*, 1783, 142-148.
- [20] Institute of Transportation. (2001). *Conditions of Transportation and Communications in Taiwan Area*. Taipei, Taiwan: Institute of Transportation, Ministry of Transportation and Communications.
- [21] Koncz, N., Greenfeld, J., & Mouskos, K. (1996). A strategy for solving static multiple-optimal-path transit network problems. *Journal of Transportation Engineering*, 122(3), 218-225.
- [22] Koncz, N., & Greenfeld, O. (1995). *The development of a GIS-based transit advanced traveler information system*. Paper presented at the Annual Conference of the Urban and Regional Information Systems Association, Washington, D.C.
- [23] Lam, W. H. K., Cheung, C. Y., & Poon, Y. F. (1999). A study of passenger discomfort measures at Hong Kong mass transit railway system. *Journal of Advanced Transportation*, 33(3), 389-399.
- [24] Le Clercq, F. (1972). A public transport assignment method. *Traffic Engineering and Control*, 14(2), 91-96.
- [25] Levinson, D. (2003). The value of advanced traveler information systems for route choice. *Transportation Research Part C*, 11(1), 75-87.

- [26] Lo, H. K., Yip, C. W., & Wan, K. H. (2003). Modeling transfer and non-linear fare structure in multi-modal network. *Transportation Research Part B*, 37(2), 149-170.
- [27] Lo, H. K., Yip, C.-W., & Wan, Q. K. (2004). Modeling competitive multi-modal transit services: a nested logit approach. *Transportation Research Part C*, 12(3-4), 251-272.
- [28] Lozano, A., & Storchi, G. (2002). Shortest viable hyperpath in multimodal networks. *Transportation Research Part B*, 36(10), 853-874.
- [29] McCormack, J. E., & Roberts, S. A. (1996). Exploiting object oriented methods for multi-modal trip planning systems. *Information and Software Technology*, 38(6), 409-417.
- [30] Mirchandani, P. B., & Wiecek, M. M. (1993). Routing with nonlinear multiattribute cost functions. *Applied Mathematics and Computation*, 54(2-3), 215-239.
- [31] Modesti, P., & Sciomachen, A. (1998). A utility measure for finding multiobjective shortest paths in urban multimodal transportation networks. *European Journal of Operational Research*, 111(3), 495-508.
- [32] Mouskos, K. C., & Greenfeld, J. (1999). A GIS-based multimodal advanced traveler information system. *Computer-Aided Civil and Infrastructure Engineering*, 14(4), 267-279.
- [33] Muller-Hannemann, M., Schulz, F., Wagner, D., & Zaroliagis, C. (2007). *Timetable information: models and algorithms*. In Algorithmic Methods for Railway Optimization. Berlin, Germany: Springer.
- [34] Nachtigal, K. (1995). Time depending shortest-path problems with applications to railway networks. *European Journal of Operational Research*, 83(1), 154-166.
- [35] Nes, R. V., & Bovy, P. (2004). Multimodal traveling and its impact on urban transit network design. *Journal of Advanced Transportation*, 38(3), 225-241.
- [36] Nguyen, S., & Pallottino, S. (1989). *Hyperpaths and shortest hyperpaths*. Paper presented at the Lectures given at the third session of the Centro Internazionale Matematico Estivo (C.I.M.E.) on Combinatorial optimization.
- [37] Orda, A., & Rom, R. (1990). Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. *Journal of the ACM*, 37(3), 607-625.
- [38] Peng, Z.-R., & Huang, R. (2000). Design and development of interactive trip planning for web-based transit information systems. *Transportation Research Part C*, 8(1-6), 409-425.

- [39] Peng, Z.-R., Kim, E., & Weng, Y. (2006). *Performance enhancement for online transit trip planning systems: a dynamic reduction of transit networks*. Paper presented at the Annual Meeting of Transportation Research Board.
- [40] Rehrl, K., Brunsch, S., & Mentz, H.-J. (2007). Assisting multimodal travelers: design and prototypical implementation of a personal travel companion. *IEEE Transactions on Intelligent Transportation Systems*, 8(1), 31-42.
- [41] Schulz, F., Wagner, D., & Weihe, K. (2000). Dijkstra's algorithm on-line: an empirical case study from public railroad transport. *Journal of Experimental Algorithmics*, 5, Article 12.
- [42] F. Schulz, D. Wagner, and C. Zaroliagis. (2002) *Using multi-level graphs for timetable information in railway systems*. Paper presented at the 4th Workshop on Algorithm Engineering and Experiments.
- [43] Sen, S., Pillai, R., Joshi, S., & Rathi, A. K. (2001). A mean-variance model for route guidance in advanced traveler information systems. *Transportation Science*, 35(1), 37-49.
- [44] Spiess, H., & Florian, M. (1989). Optimal strategies: a new assignment model for transit networks. *Transportation Research Part B*, 23(2), 83-102.
- [45] Taniguchi, E., & Shimamoto, H. (2004). Intelligent transportation system based dynamic vehicle routing and scheduling with variable travel times. *Transportation Research Part C*, 12(3-4), 235-250.
- [46] Tong, C. O., & Richardson, A. J. (1984). A computer model for finding the time-dependent minimum path in a transit system with fixed schedules. *Journal of Advanced Transportation*, 18(2), 145-161.
- [47] Tong, C. O., & Wong, S. C. (1999). A stochastic transit assignment model using a dynamic schedule-based network. *Transportation Research Part B*, 33(2), 107-121.
- [48] Vuchic, V. (1981). *Urban Public Transportation Systems and Technology*. Englewood Cliffs, NJ: Prentice-Hall.
- [49] Wang, I.-L. (2003). Shortest paths and multicommodity network flows. *Dissertation Abstracts International*, 64(03), 1472B. (AAT 3085008)
- [50] Wang, I.-L., Johnson, E. L., & Sokol, J. S. (2005). A multiple pairs shortest path algorithm. *Transportation Science*, 39(4), 465-476.
- [51] Wong, K. I., Wong, S. C., Tong, C. O., Lam, W. H. K., Lo, H. K., Yang, H., et al. (2005). Estimation of origin-destination matrices for a multimodal public transit network. *Journal of Advanced Transportation*, 39(2), 139-168.
- [52] Wu, C. H., Su, D. C., Chang, J., Wei, C. C., Lin, K. J., & Ho, J. M. (2003). *The design and implementation of intelligent transportation web services*. Paper presented at the IEEE Conference on Electronic Commerce, Newport Beach, CA.

- [53] Wu, Q., & Hartley, J. (2004). Accommodating user preferences in the optimization of public transport travel. *International Journal of Simulation*, 5(4), 12-25.
- [54] Yang, H., & Huang, H.-J. (2004). Modeling user adoption of advanced traveler information systems: a control theoretic approach for optimal endogenous growth. *Transportation Research Part C*, 12(3-4), 193-207.
- [55] Yin, Y., Lam, W. H. K., & Miller, M. A. (2004). A simulation-based reliability assessment approach for congested transit network. *Journal of Advanced Transportation*, 38(1), 27 - 44.
- [56] Zhan, F. B., & Noon, C. E. (1998). Shortest path algorithms: an evaluation using real road networks. *Transportation Science*, 32(1), 65-73.



APPENDIX A

COMPUTATIONAL EXPERIMENTS ON BASIC TIMETABLE INFORMATION PROBLEMS

This appendix lists computation results of different shortest path algorithms, DIJ, TO, and SPT, on datasets derived from Taipei transit system and the improvement of efficiency by using our proposed speed-up technique.

Table A.1. Relative performance on problem set 1

OD Group	DIJ	TO	SPD (=DFS+TOAD)	ΔT_D	SPB (=BFS+TOAB)	ΔT_B
1	3895	2144	1032 (=591+441)	54.1%	1031 (=594+437)	51.9%
2	4039	2270	1082 (=615+466)	54.0%	1086 (=623+463)	52.1%
3	4096	2320	1109 (=632+477)	54.1%	1106 (=640+466)	52.3%
4	4250	2457	1175 (=669+506)	54.1%	1155 (=670+485)	53.0%
5	4351	2550	1216 (=694+522)	54.2%	1181 (=700+481)	53.7%
6	4419	2632	1246 (=705+539)	54.2%	1211 (=719+493)	54.0%
7	4378	2609	1237 (=705+532)	54.3%	1214 (=714+500)	53.4%
8	4586	2822	1335 (=757+578)	54.3%	1305 (=779+526)	53.7%
9	4645	2878	1363 (=774+589)	54.3%	1362 (=779+583)	52.6%
10	4811	3041	1435 (=814+621)	54.3%	1419 (=824+595)	53.3%
time unit in CPU click $\Delta T_D = (TO - SPD)/TO \times 100\%$ $\Delta T_B = (TO - SPB)/TO \times 100\%$						

Table A.2. Relative performance on problem set 2

OD Group	DIJ	TO	SPD (=DFS+TOAD)	ΔT_D	SPB (=BFS+TOAB)	ΔT_B
1	4096	2296	1078 (=605+471)	53.1%	1047 (=609+438)	54.4%
2	4220	2413	1151 (=655+496)	52.3%	1127 (=662+465)	53.3%
3	4208	2423	1153 (=653+498)	52.5%	1141 (=671+470)	52.9%
4	4301	2516	1188 (=672+515)	52.8%	1201 (=690+511)	52.3%
5	4675	2896	1373 (=783+589)	52.6%	1375 (=786+589)	52.5%
6	4827	3024	1418 (=799+617)	53.2%	1372 (=805+567)	54.6%
7	4925	3109	1455 (=821+632)	53.3%	1414 (=830+584)	54.5%
8	5161	3368	1583 (=894+687)	53.0%	1572 (=913+659)	53.3%
9	5226	3456	1623 (=918+703)	53.1%	1632 (=939+693)	52.8%
10	5286	3461	1622 (=920+702)	53.1%	1595 (=922+673)	53.9%
time unit in CPU click $\Delta T_D = (TO - SPD)/TO \times 100\%$ $\Delta T_B = (TO - SPB)/TO \times 100\%$						

APPENDIX B

**COMPUTATIONAL EXPERIMENTS ON
MULTIMODAL TIMETABLE INFORMATION
PROBLEMS WITH WALKING TRANSFER**

Table B.1. Relative performance on problem set 1

OD Group	DIJ	TO	SPD (=DFS+TOAD)	ΔT_D	SPB (=BFS+TOAB)	ΔT_B
1	9478	6372	2923 (=1637+1286)	54.1%	2858 (=1665+1193)	55.1%
2	9777	6675	3070 (=1722+1348)	54.0%	3069 (=1731+1338)	54.0%
3	10050	6948	3789 (=1785+1404)	54.1%	3148 (=1827+1321)	54.7%
4	10363	7253	3326 (=1864+1463)	54.1%	3236 (=1890+1346)	55.4%
5	10655	7553	3461 (=1937+1524)	54.2%	3411 (=1963+1448)	54.8%
6	10919	7817	3583 (=2006+1576)	54.2%	3515 (=2044+1471)	55.0%
7	11295	8191	3747 (=2097+1650)	54.3%	3749 (=2102+1647)	54.2%
8	11543	8442	3856 (=2155+1701)	54.3%	3851 (=2201+1650)	54.4%
9	11862	8762	4004 (=2240+1765)	54.3%	3889 (=2268+1621)	55.6%
10	12245	9152	4181 (=2337+1843)	54.3%	4114 (=2375+1739)	55.0%
time unit in CPU click $\Delta T_D = (TO - SPD)/TO \times 100\%$ $\Delta T_B = (TO - SPB)/TO \times 100\%$						

Table B.2. Relative performance on problem set 2

OD Group	DIJ	TO	SPD (=DFS+TOAD)	ΔT_D	SPB (=BFS+TOAB)	ΔT_B
1	9533	6425	2957 (=1661+1296)	54.2%	2864 (=1661+1203)	55.4%
2	9847	6751	3103 (=1741+1362)	54.1%	3053 (=1780+1273)	54.8%
3	10157	7063	3241 (=1816+1425)	54.2%	3223 (=1823+1400)	54.4%
4	10425	7327	3362 (=1883+1478)	54.2%	3360 (=1905+1455)	54.1%
5	10708	7615	3489 (=1951+1538)	54.2%	3449 (=1961+1488)	54.7%
6	11031	7933	3628 (=2029+1599)	54.3%	3587 (=2038+1549)	54.8%
7	11363	8258	3780 (=2117+1663)	54.3%	3735 (=2133+1602)	54.8%
8	11586	8485	3882 (=2173+1710)	54.4%	3924 (=2227+1697)	53.8%
9	11960	8862	4052 (=2267+1785)	54.3%	3950 (=2300+1650)	55.4%
10	12188	9079	4151 (=2323+1828)	54.4%	4030 (=2347+1683)	55.6%
time unit in CPU click $\Delta T_D = (TO - SPD)/TO \times 100\%$ $\Delta T_B = (TO - SPB)/TO \times 100\%$						

Table B.3. Relative performance on problem set 3

[illegible]

Table B.4. Relative performance on problem set 4

[illegible]

APPENDIX C

COMPUTATIONAL EXPERIMENTS ON FARE INFORMATION PROBLEMS

This appendix lists computation results of Dijkstra's algorithm with binary heap on datasets derived from Taipei transit system.

Table C.1. Average running time on problem set 1 for different OD group

<i>Problem Set</i>	<i>OD Groups</i>	<i>DIJF</i>
P ₁	1	4500
P ₁	2	4742
P ₁	3	5039
P ₁	4	5199
P ₁	5	5432
P ₁	6	5717
P ₁	7	5796
P ₁	8	6002
P ₁	9	6218
P ₁	10	6375
time unit in CPU click		

Table C.2. Average running time on problem set 2 for different OD group

<i>Problem Set</i>	<i>OD Groups</i>	<i>DIJF</i>
P ₂	1	4748
P ₂	2	5021
P ₂	3	5282
P ₂	4	5388
P ₂	5	5587
P ₂	6	5851
P ₂	7	6064
P ₂	8	6116
P ₂	9	6328
P ₂	10	6615
time unit in CPU click		

Table C.3. Average running time on problem set 3 for different OD group

<i>Problem Set</i>	<i>OD Groups</i>	<i>DIJF</i>
P ₃	1	4800
P ₃	2	5034
P ₃	3	5349
P ₃	4	5543
P ₃	5	5764
P ₃	6	5934
P ₃	7	6135
P ₃	8	6344
P ₃	9	6304
P ₃	10	6644
time unit in CPU click		

Table C.4. Average running time on problem set 4 for different OD group

<i>Problem Set</i>	<i>OD Groups</i>	<i>DIJF</i>
P ₄	1	4933
P ₄	2	5156
P ₄	3	5415
P ₄	4	5593
P ₄	5	5745
P ₄	6	6080
P ₄	7	6173
P ₄	8	6338
P ₄	9	6529
P ₄	10	6708
time unit in CPU click		

APPENDIX D

COMPUTATIONAL EXPERIMENTS ON FARE INFORMATION PROBLEMS WITH ALTERNATIVE FARE MODEL

This appendix lists computation results of Dijkstra's algorithm with binary heap using alternative fare model on datasets derived from Taipei transit system.

Table D.1. Average running time using alternative fare model on problem set 1 for different OD group

<i>Problem Set</i>	<i>OD Groups</i>	<i>DIJF</i>
P ₁	1	2702
P ₁	2	2876
P ₁	3	3002
P ₁	4	3117
P ₁	5	3249
P ₁	6	3457
P ₁	7	3548
P ₁	8	3591
P ₁	9	3698
P ₁	10	3812
time unit in CPU click		

Table D.2. Average running time using alternative fare model on problem set 2 for different OD group

<i>Problem Set</i>	<i>OD Groups</i>	<i>DIJF</i>
P ₂	1	2837
P ₂	2	3004
P ₂	3	3147
P ₂	4	3271
P ₂	5	3397
P ₂	6	3580
P ₂	7	3613
P ₂	8	3730
P ₂	9	3851
P ₂	10	3916
time unit in CPU click		

Table D.3. Average running time using alternative fare model on problem set 3 for different OD group

<i>Problem Set</i>	<i>OD Groups</i>	<i>DIJF</i>
P ₃	1	2914
P ₃	2	3041
P ₃	3	3186
P ₃	4	3278
P ₃	5	3445
P ₃	6	3620
P ₃	7	3720
P ₃	8	3809
P ₃	9	3956
P ₃	10	3977
time unit in CPU click		

Table D.4. Average running time using alternative fare model on problem set 4 for different OD group

<i>Problem Set</i>	<i>OD Groups</i>	<i>DIJF</i>
P ₄	1	2907
P ₄	2	3021
P ₄	3	3235
P ₄	4	3336
P ₄	5	3451
P ₄	6	3679
P ₄	7	3713
P ₄	8	3863
P ₄	9	3949
P ₄	10	3957
time unit in CPU click		