A Study of Transportation Network Design Problems

by

Mingyuan Chen

A Thesis Presented to

The University of Manitoba

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in Industrial Engineering

Winnipeg, Manitoba

August 29, 1991

ISBN 0-315-76632-8

A STUDY OF TRANSPORTATION NETWORK DESIGN PROBLEMS

by

MINGYUAN CHEN

A thesis submitted to the Faculty of Graduate Studies of
the University of Manitoba in partial fulfillment of the requirements
of the degree of

DOCTOR OF PHILOSOPHY

Abstract

Transportation network design involves deciding which roads to build to form the transportation infrastructure and which roads to upgrade to efficiently increase the capacities of the network. Traffic assignment models are needed as an essential part of network design procedures. Most network design models ignore the stochastic and dynamic aspects of traffic demand in the assignment part. These aspects are considered in this thesis.

A logit-based stochastic traffic assignment model is explored and new algorithms for solving stochastic static traffic assignment problems are developed. A new branch and bound approach for solving discrete network design problems with stochastic static traffic assignment is developed. This is the first time that stochastic traffic assignment is included in a network design model.

A new traffic assignment algorithm, using the method of successive averages, is developed for dynamic traffic assignment. Theoretical analysis of this algorithm in a simplified network is provided. A branch and bound approach for solving network design problems with dynamic traffic assignment is developed. By incorporating the dynamic traffic assignment model into a network design model, one is able to consider peak hour traffic situation in solving network design problems.

Traffic assignment and network design models with time-varying demands are extended to formulate a parts routing problem and a system design problem in a manufacturing system. The manufacturing problems are solved by the techniques developed in this transportation study.

Acknowledgements

The author wishes to express his heartfelt appreciation and gratitude to his advisor, Dr. A.S. Alfa for his guidance and advice throughout this study. His understanding, patience and suggestions have always been a source of inspiration. His help to the author in all aspects in completing this Ph.D. program is greatly appreciated.

The author also wishes to express his appreciation to other members of the Advisory Committee for their valuable suggestions and advice. Their help is also greatly appreciated.

Funding provided by the University of Manitoba Research Development Fund and the University of Manitoba Graduate Fellowship is gratefully acknowledged. Funding provided by the Natural Sciences and Engineering Research Council of Canada through a grant to Dr. A.S. Alfa is also gratefully acknowledged.

The author is obliged to his wife, Qing-Qing, for her support and sacrifice during the course of this work.

Contents

Chapter 1

Introduction

1.1 The Transportation Network Design (ND) Problem

1.1.1 Background

Transportation network design deals with planning transportation networks. This includes deciding which roads to build to form the transportation infrastructure and which roads to upgrade to increase the capacities of the network. Transportation network planning is an important practical work since it is a routine task of transportation agencies and significantly affects the daily life of the people. It is also an important research problem and attracts much attention from transportation studies because many practical problems are very complicated and remain unsolved. The motivation of this research is to develop models and algorithms that can be used to solve complicated practical network design problems.

A transportation network refers to the infrastructure of a transportation system such as a city road network, a highway transportation network, a rail road system, or air service connections of an airline company. The phrase "network design" is usually used to refer to two different but related types of transportation planning problems:

i) Making decision whether to add/delete connections (links) in an existing transportation network; and/or, whether to improve/reduce capacities of certain existing links in the network. A typical example is of a city transportation agency deciding which road(s) in the city should be upgraded or which new road(s) should be built. The objective of the planning work is to maximize the benefits from the upgrading and constructions of the road(s) within the city's budget constraint.

ii) Planning of a new transportation network where none was in existence previously; such as planning a transportation network in a new city. A good example is planning of the road system for the City of Brasilia.

The first type of network design problem is more common than the second one, and it needs to be solved much more often than the latter. In an ever changing society, there is always a need to establish new connections among residential or business areas even for a newly constructed city. In this research, therefore, we will mainly study the first type of network design problem. The phrase "network design (ND) problems" in this thesis will refer to the first type of network design problem only.

As discussed above, solving ND problems deals with the issues of adding, deleting and improving links in an existing network according to given criteria. Practical ND examples can be found in our daily life. Traffic roads in a city needs to be maintained and upgraded every year. Once a while, the whole transportation system of a city may need to be improved. Making decisions regarding the selection of certain roads for upgrading and/or selecting new roads between two residential areas for building is a typical network design problem. Another example can be found in today's railway systems. Higher operating cost and reduced demand of railway travellers have forced railway companies to stop part of their services. The decisions of which part of

the current services should be discontinued while keeping the remaining service at a maximum possible level is also a network design problem. City public transit systems may face a similar network design problem. Airline companies also need to solve ND problems. Economic recession may bring airline companies to reduce or stop part of their services. Making decisions as to which air routes to disconnect or determine the level of services to be provided is an ND problem similar to the one that railway companies have. When the economy starts booming, however, airline companies may face another ND. They may need to make decisions regarding the restoration of some of the old connections or establish new connections in order to meet the increasing demands. For different ND problems, major considerations in solving the problems may change. For example, construction cost will be a major factor in solving an ND problem in a road network, while in a airline system, operating cost will be the main consideration.

In this thesis research, we mainly discuss ND problems in highway or urban transportation networks. The main considerations in solving such network design problems are:

- total travel time that all the travellers need to spend in the concerned networks in order to complete journeys from their origins to destinations; and
- construction cost needed to modify the existing networks.

In this era of ever decreasing budgets, the need to optimize the network design process is becoming more critical. The methods presently employed in solving practical ND problems are combinations of experience and some simple heuristic procedures. Mathematical optimization models and algorithms have also been developed and applied in solving ND problems. Advantages and limitations of the

different approaches in solving ND problems will be discussed later in this chapter. In this research, we mainly discuss the development of optimization or heuristic models and algorithms that can be used to solve realistic ND problems.

1.1.2 The Importance of ND Problems

Properly solving ND problems is very important due to the following reasons:

- Solution of an ND problem has great social and political impact on a local community or even on the whole society. To construct a highway or discontinue a part of railway services may affect the daily lives of hundreds of people. A wrong solution of a large scale ND problem may result in social and environmental disasters.

- A large amount of manpower and capital investment is involved in most ND problems. Millions of dollars may be needed to build a short segment of highway. Dismantling a portion of a railway service or deleting an airline connection may cost hundreds of jobs. Optimal or sub-optimal solutions of ND problems must be searched for because of the possibility of wasting precious manpower and huge capital investment if a wrong solution is implemented.

1.1.3 The Complexity of ND Problems

ND problems are usually difficult to solve because of their complexities. To properly solve an ND problem, one needs to consider many complicated factors. To solve an ND problem normally involves one or more of the following factors:

- Traffic assignment component: Solving different traffic assignment problems is needed in order to solve different ND problems. For example, if one is to solve an ND problem in a highway or urban road network, traffic assignment

should be generated according to the user equilibrium principle (Wardrop's First Principle); if, however, the ND problem is raised from a railway system or airline services, traffic assignment should be generated according to the system optimal principle (Wardrop's Second Principle). Traffic assignments based on the two different principles are generated by different models and algorithms.

- Uncontrollable factors: In solving an ND problem in a road network, behavior of the motorists in the network is assumed to be predictable but uncontrollable. The number of vehicles on the roads in the network are the main variables in solving the ND problem. These variables are descriptive variables because of the uncontrollable behavior of the motorists. It could be very complicated to formulate an optimization model with those uncontrollable variables in the ND problem.

- Time-varying aspect: In solving an ND problem in an urban road network, one should consider the time-varying aspects of traffic demands. Solving ND problems in an urban network normally aims at reducing traffic congestions during peak hours when traffic demands are time-varying. Models and algorithms will not generate realistic solutions for such dynamic ND problems if the time-varying aspect is ignored. However, ND problems with the additional time variable become very complicated.

- Multiple stages: The procedure of solving an ND problem can normally be divided into two stages: i) traffic assignment; ii) selection of a proper set of candidate links.

- Integer variables: In formulating an ND model, integer variables may be involved. It is often difficult to develop an efficient algorithm to solve an optimization model with integer variables.

- Social and political factors: They are often involved in solving many realistic ND problems. It is also difficult to use any mathematical models or to develop optimization models to deal with such qualitative information.

1.2 Techniques Used for Solving ND Problems

As discussed above, the implementation of an ND problem solution may require large scale construction. To construct a segment of highway or to stop certain part of railway services may have certain social, political or environmental impacts. In most cases, it is impossible to conduct real scale experiments in searching solutions of an ND problem. Or, at least, such experiments are normally not affordable. Approaches for solving ND problems then rely on:

i) knowledge of well trained and highly skilled experts with long time experience in transportation network planning; these experts can make good decisions using their experience; and

ii) optimization or simulation models to solve various ND problems. ND problem solutions can be generated by computer programs based on scientific procedures. These solutions can be tested by computer programs using realistic examples before they are implemented.

Qualitative knowledge and long time domain experience of transportation planning experts can never be replaced by computer programs. Their experience must be

utilized in solving realistic ND problems. In fact, many complicated ND problems are solved mainly by such knowledge and experience.

On the other hand, mathematical models and other scientific procedures are playing a more important role in solving today's ND problems. Scientific research of ND problems is very active. But realistic large scale ND problems are very difficult to solve by mathematical optimization methods and they require a large amount of computational effort. In recent years, many researchers have focused on finding efficient methods to solve realistic large-scale ND problems.

Techniques used in this research for solving ND problems are mainly mathematical optimization models and efficient algorithms. Some of the algorithms are exact methods and some of them are only heuristic methods. The effectiveness and efficiency of the algorithms are our main considerations. All the algorithms developed in this thesis research have been coded in computer programming language. The correctness and efficiency of these algorithms can be tested by computer using small or large network examples.

1.3 Major Contributions of This Research

Traffic assignment and ND problems have been studied by many researchers. Many useful results have been reported and published in the literature. However, the two major weaknesses in the existing models are that many of them i) ignore the stochastic element, and ii) ignore the time-varying aspect. These two important aspects are included in different models developed in the research presented in this thesis. The major contributions made in this research can be listed as follows:

- For static transportation networks in which traffic demands are not time-varying:

- A stochastic traffic assignment model is explored and new algorithms for solving the stochastic traffic assignment problems are developed.
- A new branch and bound approach for solving discrete ND problems with stochastic traffic assignment is developed. To our knowledge, this is the first time that the stochastic traffic assignment is included in an ND model.

- For dynamic transportation networks in which traffic demands are time-varying:
  - A new traffic assignment algorithm is developed and theoretical analysis of the algorithm in a simplified network is provided.
  - A branch and bound approach for solving discrete ND problems with dynamic traffic assignment is developed. This is the first time that a dynamic traffic assignment model is incorporated into an ND model.

- Other contributions:
  - ND solutions based on static and dynamic traffic demands are compared for different types of performance functions. It concludes that time-varying aspect of traffic demands should be considered in solving ND problems.
  - Traffic assignment and ND models with time-varying demands are extended to formulate a parts routing problem and a system design problem in a manufacturing system. The manufacturing problems are very similar in nature to the dynamic traffic assignment and ND problems, and are solved by the techniques developed in this transportation study.

1.4 Organization of This Thesis

The rest of this thesis is organized as follows:

Chapter Two is a literature review. Related works reported in recently published papers and books are briefly discussed in this chapter.

Chapter Three is a discussion of traffic assignment and ND problems with static traffic demand. A number of traffic assignment models and algorithms, including the ones developed in this research, are discussed extensively. A discrete model to formulate the static ND problems and a branch and bound algorithm for solving the model are discussed in this chapter. A number of numerical examples are presented to compare different traffic assignment models and algorithms. Network design examples, including one based on a realistic city road network, are provided for testing the models and algorithms presented in this chapter.

Chapter Four is a discussion of traffic assignment and ND problems with time-varying traffic demands. A new algorithm for dynamic traffic assignment is presented with a theoretical analysis in a simple two-link network. An ND problem with dynamic traffic demands is discussed and a branch and bound procedure for solving the dynamic ND problem is presented. Numerical examples are provided. Different versions of dynamic ND models are compared in this chapter. ND problems with static and dynamic traffic demands are solved by these different versions of the dynamic ND model. The comparison study is conducted using a number of randomly selected numerical examples.

Chapter Five is an extension of the traffic assignment and ND models and algorithms. They are extended to solve a parts routing problem in manufacturing systems using a method similar to the one for dynamic traffic assignment. Solving manufacturing system design problems by transportation ND technique is also dis-

cussed in this chapter.

Chapter Six presents the conclusion and discussion of this thesis. Approaches and methods developed or applied in this thesis research are discussed. Computational results of examples and observations made in this thesis research are also discussed. Topics of future research are proposed.

All computer programs developed in this research are coded in FORTRAN-77 on the AMDAHL-5870 mainframe computer at the University of Manitoba.

Chapter 2

Literature Review

2.1 Introduction

In the area of traffic assignment and transportation network design (ND), there are a comparably large number of research reports for solving traffic assignment and ND problems with static traffic demands. There are also a number of papers in the literature discussing dynamic traffic assignment problems, that is, the assignment problem with time-varying demands. However, research reports of ND problems with time-varying demands are very few.

2.2 Static Traffic Assignment and Network Design Problems

2.2.1 Traffic Assignment Models and Algorithms

In solving traffic assignment and ND problems, traffic assignment can be performed based on two different principles. These are called Wardrop's First and Second Principles. Traffic assignment based on Wardrop's First Principle is the user equilibrium assignment; whereas traffic assignment based on Wardrop's Second Principle is the system optimal assignment; (formal definitions of user equilibrium and system op-

timal assignments are given in Chapter Three). System optimal traffic assignment can be formulated into a nonlinear mathematical programming model and it can be efficiently solved by optimization techniques. The system optimal assignment model can be found in Sheffi (1985). The model and algorithms will be very useful when the traffic pattern in the system is controllable. Examples are public transit systems and railway systems. Network design models based on the system optimal traffic assignment are also useful in solving the network design problems in a controllable system. In this thesis research, we mainly consider the traffic assignment and network design problems in urban or highway transport systems where in which the behaviors of the motorists are normally uncontrollable. User equilibrium based on Wardrop's First Principle has to be used.

User equilibrium traffic assignments can be further classified into deterministic user equilibrium (DUE) assignment and stochastic user equilibrium (SUE) assignment. Traffic assignment based on deterministic user equilibrium principle can also be formulated into a nonlinear mathematical programming model. Very extensive discussion of the DUE model and its various versions can be found in the book by Sheffi (1985). Convex Combinations method, an efficient algorithm for solving nonlinear mathematical programming problems with linear constraints, is used to solve this model. Applications of Convex Combinations method for solving various versions and extensions of the DUE traffic assignment model are also presented in Sheffi (1985).

However, traffic assignment with deterministic user equilibrium has been criticized as not being very realistic. Stochastic user equilibrium assignment models are thus suggested as more realistic approaches, because they recognize that travellers do not have perfect knowledge of the system and that there is some distortion in

how they perceive travel times in the system. To perform stochastic traffic loadings, Dial (1971) presented the STOCH (logit-based) approach and Daganzo and Sheffi (1977) suggested a simulation (probit-based) approach. Stochastic traffic loading is the first step for SUE assignments. A general optimization model is proposed in Daganzo and Sheffi (1977) for solving stochastic traffic assignment problems. The objective function of the general SUE model has a complicated form and it is difficult to evaluate. A related model for economic equilibria with discrete choices is suggested by Daganzo (1979). Fisk (1980) proposed an optimization model with a logit-based traffic loading approach. The objective function of Fisk's model is simpler and the model is easier to solve. Powell and Sheffi (1982) introduced the method of successive averages (MSA) for solving these SUE assignment models and the convergence of MSA in solving the SUE assignment problems was proved. The speed of convergence of MSA is normally slower than that of Convex Combinations method.

2.2.2 Network Design Models and Algorithms

When the capacity of each road (link) in a transportation network is unlimited, traffic assignments based on the two different principles, that is, the user equilibrium principle and system optimal principle, become identical. A discrete network design problem in a network with unlimited capacities becomes a zero-one mixed integer linear programming problem. Taha (1975) discussed three special enumeration methods (Benders' decomposition, penalty method, and modified partitioning method) for optimally solving this 0-1 mixed integer programming problem. In their extensive literature review of ND problems, Magnanti and Wong (1984) referred to several papers using Benders' decomposition to solve this type of ND problem.

Branch and bound approach is also used for solving this problem when construction cost is used as a budget constraint. Detailed discussion of the approaches for solving this ND problem can be found in Magnanti and Wong (1984). Since it is an NP hard integer programming problem, heuristic algorithms are also developed for solving this problem. The heuristic methods include those of Janson and Husaini (1987), and Boffey and Hinxman (1979). It is interesting to note that Boffey and Hinxman used an operator-search technique, a typical artificial intelligence technique, to implement their heuristics. Multi-objective programming techniques have also been proposed for solving this ND problem. Examples are Current et al. (1985) and Current et al. (1987).

In a road network with limited link capacities, travel time on each link is a function of the traffic volume on the link. Under this circumstance, ND models based on system optimal assignment or user equilibrium assignment will normally yield different results. Dantzig and Maier (1979) applied Steenbrink's (1974) decomposition approach and solved a continuous system optimal ND problem. They also discussed a continuous system optimal ND model with budget constraint and solved it using Lagrange multiplier technique. LeBlanc and Abdulaal (1979) also used Lagrange multiplier technique to solve a continuous system optimal ND problem. LeBlanc (1976) formulated a railway ND problem by a system optimal model. Hoang (1982) applied generalized Bender's decomposition approach developed by Geoffrion (1972) to solve a discrete system optimal ND problem.

For solving ND problems with the user equilibrium principle in congested networks, LeBlanc (1975) proposed a branch and bound approach for solving a discrete budget constraint problem. He used total travel cost from a corresponding system optimal model as the lower bound to narrow the search for optimal solutions.

Poorzahedy and Turnquist (1982) developed a heuristic approach to solve a budget constraint discrete ND problem. A sensitivity analysis mechanism is provided by Poorzahedy and Turnquist. The two papers of LeBlanc (1975) and Poorzahedy and Turnquist (1982) are the only ones we have found that reported solution approaches for solving discrete user equilibrium ND problems. Abdulaal and LeBlanc (1979) used two optimization methods, Powell algorithm and Hooke-Jeeves algorithm to solve continuous user equilibrium ND problems. Marcotte (1983) presented an algorithm for solving a continuous user equilibrium ND problem. His approach is based on the assumption that the user equilibrium can be achieved by solving a series of system-optimal traffic assignment problems with a finite number of extra constraints. Suwansirikul et al. (1987) proposed a heuristic approach for solving a continuous ND problem. Their method is based on a decomposition approach. LeBlanc and Abdulaal (1984) conducted a comparison study of using system optimal and user equilibrium models to solve a user equilibrium ND problems.

2.3 Dynamic Traffic Assignment and Network Design Problems

Traffic assignment and ND problems become much more complicated when time-varying aspect of traffic demands is included. The problems now are called dynamic traffic assignment problem and dynamic ND problem. There are a number of papers in the literature addressing dynamic traffic assignment problems. In their pioneering work, Merchant and Nemhauser (1978) developed a system optimal dynamic assignment model and provided a piecewise linear programming approach to solve it. A global optimum solution can be obtained with certain assumptions of the objective function. Carey (1987) reformulated the system optimal model by Merchant and

Nemhauser into a convex non-linear programming model. Mahmassani and Herman (1984) presented a user equilibrium dynamic assignment model for joint selection of departure time and path. Mathematical analysis for this model was also presented. Both the model and the analysis of Mahmassani and Herman (1984) are for a simplified network of two links with one origin-destination (O-D) pair. It may be difficult to extend their approach and results to a general network with multiple O-D pairs. Friesz et al.(1989) presented continuous time formulations for both system optimal and user equilibrium dynamic traffic assignments in a general network with multiple O-D pairs. Comprehensive theoretical analyses for the system optimal and user equilibrium assignment models were also provided. Their formulation of the user equilibrium model assumes that each traveller in the network will re-calculate his/her minimum travel time path and shift to a new shortest path for the rest of his/her trip at each node as he/she arrives at the nodes. This model will be useful in real time analysis, but it may be difficult to develop an algorithm to solve such an assignment model.

Because the time-varying aspect is involed, it is very difficult to develop suitable models for the dynamic user equilibrium traffic assignment. Attempts have been made to develop effective and efficient algorithms to solve the problem without a formal formulation. These algorithms are based on common logic and commonly accepted assumptions of travellers' behavior. For example, Alfa (1987) developed two algorithms to solve the user equilibrium dynamic assignment problems. Both algorithms can be applied to a general network with multiple O-D pairs. The basic idea of the two algorithms is to assign the traffic volume on the shortest paths between O-D pairs iteratively. Traffic congestions encountered in the previous iterations will be avoided by the traffic loadings in the later iterations. After large

number of iterations, equilibrium can be expected. Another algorithm developed without formal formulation is CONTRAM reported in Leonard et al. (1989). CONTRAM is developed for solving user equilibrium traffic assignment problems with time-varying demands in a general network with multiple O-D pairs. CONTRAM contains many well-developed functions and it is a very good traffic management tool. Traffic assignment procedure in CONTRAM is based on an iterative algorithm. This algorithm allocates traffic demands to the minimum travel time routes by portions (packets). It can be considered as an incremental algorithm. Other research work for solving dynamic traffic assignment problems include Janson (1990) and Drissi-Kaitouni (1990).

Unfortunately, we cannot find any report in the literature for solving ND problems with time-varying traffic demands. It seems that the research work on ND problems with time-varying demands presented in this thesis is the only work explicitly addressing this topic. We believe that some other available models and algorithms, in addition to the approach developed in this research, can be applied to solve dynamic ND problems. For example, it is possible to construct a ND model based on the assignment model by Merchant and Nemhauser (1978) to solve a system optimal network design problem with time-varying demands.

2.4 Summary

From the above discussion, it can be seen that various network design models and algorithms have been developed for solving different ND problems. Most of the formulations are based on optimization models but many of the algorithms are only heuristic. In many cases, the complexity and difficulty of the problems prevent them from being solved optimally. It also can be seen that many difficult problems,

such as discrete user equilibrium ND problems with static traffic demands, dynamic traffic assignment and ND problems need to be further explored. More mathematical models and efficient algorithms for solving these difficult problems need to be developed.

Chapter 3

Network Design Problems with Static Traffic Assignment

3.1 Introduction

Transportation assignment models and solving approaches are the basis for the study of transportation network design problems. Any network design model must be based on a stable traffic pattern which needs to be generated by appropriate traffic assignment models and algorithms. In the next section, we will discuss various traffic assignment models and algorithms with static demand, emphasizing stochastic user equilibrium problems. The static demand means that in the transportation network, the number of vehicles for each origin-destination (O-D) pair is a constant in the considered time period. Some development of a logit-based stochastic user equilibrium model will be presented. In Section 3.3, we will discuss discrete transportation network design problems and present a branch and bound algorithm for solving a user equilibrium network design problem using an incremental stochastic assignment approach. Traffic assignment and network design examples are presented to illustrate these approaches and algorithms.

3.2 Traffic Assignment Models

Consider a transportation network $G = \{V, L\}$, where V and L are the set of nodes and the set of links of G, respectively. Define W as the set of origin or destination nodes in the network with $W \subseteq V$. For network G, travel time t_{ij} spent by vehicles on link ij, is a function ("the performance function") of the number of the vehicles (the traffic flow) x_{ij} on link ij, where ij is a one way link from node i to node j, $i \in N, j \in N$. The performance function of each link is assumed, for the static traffic assignment and network design problems, to be of the BPR (Bureau of Public Roads) type curve, i.e.:

$$t_{ij}(x_{ij}) = a_{ij} + b_{ij}x_{ij}^4$$

where a_{ij} and b_{ij} are parameters independent of traffic volume on link ij. a_{ij} and b_{ij} reflect the length, capacity, quality of pavement, and other features of the road represented by link ij in network G.

Let $X = [x_{12}, ..., x_{ij}, ..., x_{|L|}]$ and $|L|$ denote the cardinality of L. We define $X \in \mathbf{X}(\mathbf{F}) \Longleftrightarrow$

$$\sum_k f_k^{rs} = d_{rs}, \quad \forall r, s \in W; \tag{3.1}$$

$$x_{ij} = \sum_{rs} \sum_k f_k^{rs} \delta_{ij,k}^{rs}, \quad \forall ij \in L; \tag{3.2}$$

$$f_k^{rs} \geq 0, \quad \forall k, r, s; \tag{3.3}$$

$$x_{ij} \geq 0, \quad \forall ij \in L; \tag{3.4}$$

where:

f_k^{rs} is the traffic flow on route k from origin r to destination s,

d_{rs} is the traffic demand (number of vehicles) from origin r to destination s, and

$$\delta_{ij,k}^{rs} = \begin{cases} 1 & \text{if link } ij \text{ is on route } k \text{ from } r \text{ to } s, \\ 0 & \text{otherwise.} \end{cases}$$

With above definitions and illustrations, we will discuss traffic assignment problems in network G. More symbols and definitions may be given in the following text whenever it is necessary.

3.2.1 System Optimal Traffic Assignment

Some transportation systems are controllable. A controllable system is the one in which the movement of vehicles can be controlled by a higher level authority. Examples are railway systems, airline companies, or an automated guided vehicle (AGV) system in an advanced manufacturing system. For such controllable transportation systems, optimization of the whole system output is normally considered the most important task.

A formal definition of system optimal traffic assignment can be stated as (Sheffi (1985)):

At system optimal, the total travel time spent by all vehicles will increase if any motorist for any O-D pair shifts his/her travel route in the network.

From the definition, it is easy to develop a mathematical programming model to generate the system optimal traffic assignment. A typical mathematical model is:

(SOA)

$$\min \sum_{ij} x_{ij} t_{ij}(x_{ij}) \tag{3.5}$$

subject to:

$$X \in \mathbf{X}(\mathbf{F}) \tag{3.6}$$

It has been proven that there exists a unique solution for Eqs(3.5) and (3.6) if the performance functions $t_{ij}(x_{ij})$ are convex (see Sheffi (1985)).

Our focus in this research is on user equilibrium assignment and network design problems. Therefore, the system optimal traffic assignment problems will not be discussed in detail.

3.2.2 User Equilibrium Traffic Assignments

System optimal assignment is a good approach for transportation analysis in a controllable system. However, it cannot be applied to a traffic system where the main traffic flow is composed of individual vehicles. A typical example is the movement of motorists in an urban or a highway traffic network. In such networks, each motorist selects his/her own route along his/her O-D. Motorists may have different considerations in selecting their routes such as sightseeing, avoiding heavy traffic, or driving on the roads with good pavement. In spite of the different considerations, a dominating and widely accepted criterion for each travelling individual is the minimum travel time. It is reasonable, and consistent with practical situations (Florian and Nguyen (1976)), to assume that each motorist in an urban or highway transportation network will select the route with minimum travel time for his/her O-D. After a large number of trips through the same network, travellers can acquire knowledge of the network and avoid traffic congestions. A stable traffic pattern, called user equilibrium, can be expected and be eventually achieved by the travellers in the network. The formal definition of the user equilibrium traffic assignment is (Sheffi (1985)):

For each O-D pair, at user equilibrium, the travel time on all used routes is equal, and less than or equal to the travel time that would be experienced by a single vehicle on any unused routes.

The system optimal traffic assignment is not, except in some special cases, a user equilibrium assignment. One special case, for example, is that each link in the network has unlimited capacity. In such a situation, system optimal assignment becomes identical to user equilibrium assignment.

In an urban or highway transportation network, traffic assignment will not be stable unless it has reached a user equilibrium pattern. The reason is that if the assignment is not in a user equilibrium pattern, there will always be at least one traveller who will shift his/her route unilaterally and reduce his/her travel time.

It is clear that in studying an urban or highway transportation network, system optimal traffic assignment model should not be applied. The two main reasons are as follows:

- the system optimal traffic pattern is normally not identical to the user equilibrium traffic pattern;
- the system optimal traffic assignment in an urban or highway network with limited capacities will result in unstable traffic flows.

3.2.3 Deterministic User Equilibrium (DUE) Assignment

There are several possible types of user equilibrium assignment models. The term "deterministic" is used to distinguish a particular user equilibrium pattern from the "stochastic" assignment model. Deterministic and stochastic models have a major

difference in the assumption of the travellers' behaviors. In the deterministic model, it is assumed that the path selected by each traveller in the network is the exact shortest path from his/her origin to destination. In the stochastic models, however, it is assumed that the path selected by each traveller may not be the exact shortest path between the O-D pair, but the actually selected path can be very close to the exact one. There are parameters in the stochastic models to reflect such "closeness". Stochastic models are more general and more flexible than the deterministic one. Detailed discussion of stochastic models and algorithms will be presented later in this chapter.

Deterministic user equilibrium traffic assignment problem can be formulated into a non-linear mathematical programming model as shown below:

(DUEA)

$$\min \sum_{ij} \int_0^{x_{ij}} t_{ij}(\omega) d\omega \tag{3.7}$$

subject to:

$$X \in \mathbf{X}(\mathbf{F}) \tag{3.8}$$

It has been proven that there exists a unique solution for Eqs(3.7) and (3.8) if the perfomance functions are convex (see Sheffi (1985)). The performance functions used in all of the static models in this chapter are BPR type functions (as discussed in the first paragraph in this section). It can be seen such functions are convex functions.

As stated in Sheffi (1985), the objective function of this formulation does not have any intuitive economic or behavioral interpretation. It can be viewed as a mathematical construct that is used to solve the user equilibrium assignment problem.

Solving the DUE Assignment Problem

The most suitable algorithm for solving user equilibrium assignment problem formulated in Eqs(3.7) and (3.8) is the Convex Combinations method. This method is also called Frank-Wolfe algorithm. The step by step procedure of the algorithm is:

Algorithm 3.1

Step 0. Let $n = 0$ and $x_{ij}^{(n)} = x_{ij}^{(0)} = 0, \quad ij \in L.$

Step 1. Calculate $t_{ij}^{(n)} = t_{ij}(x_{ij}^{(n)}), \quad ij \in L.$

Step 2. Using a shortest route algorithm to perform the all-or-nothing traffic loading based on the current travel time $t_{ij}^{(n)}$, find an auxiliary link flow $y_{ij}^{(n)}$ (the search direction):
$y_{ij}^{(n)} = \sum_{rs} \sum_k d_{rs} \delta_{ij,k}^{rs,(n)} \quad , \quad ij \in L$

Step 3. Compute: $x_{ij}^{(n+1)} = x_{ij}^{(n)} + \zeta_n (y_{ij}^{(n)} - x_{ij}^{(n)}), \quad ij \in L,$
where ζ_n is found by solving: $\min_{\zeta \in (0,1)} \sum_{ij \in L} \int_0^{x_{ij}^{(n)} + \zeta_n (y_{ij}^{(n)} - x_{ij}^{(n)})} t_{ij}(\omega) d\omega.$

Step 4. If: $S_v = \frac{\sum_{ij} [\frac{1}{I} \sum_{j=n-I+1}^{n} (x_{ij}^{(j)} - \bar{x}_{ij}^{n,I})^2]^{1/2}}{\sum_{ij} \bar{x}_{ij}^{n,I}} \leq \epsilon,$ stop; otherwise,
set $n = n + 1$, go to Step 1;
where $\bar{x}_{ij}^{n,I} = \frac{1}{I} \sum_{j=n-I+1}^{n} x_{ij}^{(j)}$ with $I \geq 1$ being an integer,
and ϵ is a pre-determined small real number.

Remarks of the Convex Combinations Algorithm

- In the algorithm, $*^{(l)}$ is the value of variable $*$ at iteration l.
- Step 0 is the initialization step.
- In Step 2, the all-or-nothing traffic loading can be performed by Dijkstra's

shortest route algorithm (see Ravindran et al. (1987)) to obtain the descent directions.

- In Step 3, the line search distance ζ_n can be found by the golden section method.

- Convergence of the algorithm is mathematically proved. Details can be found in Sheffi (1985).

The Convex Combinations algorithm presented above is an optimization method for solving mathematical programming problems with non-linear objectives and linear constraints. From a practical point of view, it can be explained as below.

Since, by definition, the DUE assignment model assumes that each traveller will select minimum travel time route for him/herself, the assignment can be generated by loading the traffic demands to the corresponding minimum travel time route for each O-D pair. Without loss of generality, consider a network with only one O-D pair. The assignment procedure can be started by applying a shortest route algorithm to find the minimum travel time route for the O-D pair. At this stage, all the motorists are loaded to this shortest route. Due to limited capacity of each link in the network, links on the shortest route become congested and travel time on this route may increase. Other routes with unused links may become new minimum travel time routes. Path shifting of the motorists from the old minimum travel time route to the new ones will be observed. After some motorists have been reloaded to the new routes, the shortest route algorithm will be used again to calculate a new minimum travel time route and the motorists will be reloaded between all the minimum travel time routes found in the network. After a number of iterations of this loading and reloading process, the user equilibrium travel pattern can be

reached, at which no motorist can reduce his/her travel time by unilaterally shifting route.

The Convex Combinations method is also an efficient algorithm. It takes a few seconds on the mainframe computer to generate DUE assignment for a network with about 30 nodes and 70 links.

Comments on the DUE Assignment Model

Deterministic user equilibrium assignment model together with the Convex Combinations algorithm has been used by transportation practitioners for real traffic analysis (Sheffi 1985). It is also widely used in studying network design models by researchers. Most urban transportation network design models use DUE assignment in their assignment component. Whilst the DUE approach is computationally convenient and simple to use, it is not believed to be very representative of the travellers' behavior. DUE model implies that every motorist travelling from his/her origin to destination has perfect knowledge of travel time in the network. In other words, it assumes that every motorist travelling in the network knows exactly which route has the minimum travel time. This assumption is not necessarily true in realistic situations. To eliminate this unrealistic assumption, stochastic user equilibrium assignment models have been developed. Although they normally need more computational effort in order to find a solution, stochastic models are more representative and more flexible than DUE model to generate traffic assignments. This will be discussed in detail in the next subsection.

3.2.4 Stochastic User Equilibrium (SUE) Assignment Models

A formal definition of the stochastic user equilibrium, modified after Sheffi (1985), can be stated as follows:

For each O-D pair, at stochastic user equilibrium, each motorist believes that travel time on all used routes is equal, and less than or equal to the travel time that would be experienced by a single vehicle on any unused routes.

The uncertainty factor in the SUE models can be represented by the probability of selecting minimum travel time routes by the motorists.

In the deterministic user equilibrium (DUE) model discussed previously, loading and reloading of traffic demands are carried out by the all-or-nothing shortest route algorithm in each iteration. In the stochastic user equilibrium (SUE) assignment, there is a probability that a "shortest route" selected by the motorists is not the real shortest route. In other words, some routes with travel time longer than the real minimum travel time route could be selected by motorists due to their incomplete knowledge of the network or some other distortion. The route selection errors will result in a number of "reasonable routes", instead of a single minimum travel time route, for each O-D pair.

The procedure used for the SUE assignment is similar to that for the DUE assignment. At each iteration of the SUE assignment, traffic demands will be divided into several portions and loaded to all of these "reasonable routes". The size of the portion of the demand loaded to a particular route is proportional to the probability that this route has been selected as a "minimum travel time route". In the next

iteration of the stochastic loading, new travel time on each link in the network will be calculated and the demands will be reloaded to a new set of "reasonable routes". Stochastic user equilibrium can be achieved by a large number of iterations of this loading and reloading process.

Recall that in Convex Combinations algorithm, Dijkstra's shortest route algorithm is utilized to perform deterministic (all-or-nothing) traffic loading. The shortest route algorithm is also a critical part in stochastic traffic loading. But there are two different ways of applying the shortest route algorithm when stochastic traffic loading is performed. These are probit-based and logit-based traffic loading models.

Probit-Based Traffic Loading Model

In the probit-based model, it is assumed that there is an error in perceiving the travel time on each link in the network. The error is a random variable and assumed to be normally distributed with the actual travel time as its mean. The variation of the distribution can be based on the actual travel time on the link. The probit-based loading process is a simulation based iterative approach. It can be explained as follows. In the first iteration, the sampled travel times (with errors) are used to generate a minimum travel time route for each O-D pair. Traffic demand will be loaded on the generated routes. In the next iteration, new sampled travel times will be calculated and the demand is then averaged among the minimum travel time routes obtained in previous iterations. The loading process is completed after a number of such loading (not assignment) iterations. The algorithm is given in Sheffi (1985) and the step by step procedure is presented below:

Algorithm 3.2

Step 0. Let $l = 1$

Step 1. Sample the traffic time $\tilde{t}_{ij}^{(l)}$ from $\tilde{t}_{ij} \sim N(t_{ij}, \beta t_{ij})$ $ij \in L$,

where: in Step 1, $N(t_{ij}, \beta t_{ij})$ denotes the normal distribution with mean of t_{ij} and variation of βt_{ij}, β is a given constant.

Step 2. Based on $\{\tilde{t}_{ij}^{(l)}\}$, assign d_{rs} to the minimum travel time route for each O-D pair.

Step 3. Let $\bar{x}_{ij}^{(l)} = [(l-1)\bar{x}_{ij}^{(l-1)} + x_{ij}^{(l)}]/l \quad ij \in L$, where $\bar{x}_{ij}^{(l)} = \frac{1}{l}\sum_{m=1}^{l} x_{ij}^{(m)}$.

Step 4. If $\max_{ij}\{\sigma_{ij}^{(l)}/\bar{x}_{ij}^{(l)}\} \leq \epsilon$, stop, $\{x_{ij}^{(l)}\} = \{\bar{x}_{ij}^{(l)}\}$ is the loading generated; otherwise, let $l = l + 1$, go to Step 1.

In this step, ϵ is a given constant and

$$\sigma_{ij}^{(l)} = \sqrt{\frac{1}{l-1}\sum_{m=1}^{l}(x_{ij}^{(m)} - \bar{x}_{ij}^{(l)})^2}, \quad ij \in L.$$

A different stochastic traffic loading model, the logit-based model, is presented next.

The Logit-Based Traffic Loading Model

Logit-based traffic loading model assumes that the errors in selecting the minimum travel time routes occur in the travel time of the whole route rather than on each link, as assumed by probit-based model. Dial (1971) developed an algorithm, called STOCH algorithm, to implement the logit-based loading process. In the logit-based traffic loading, STOCH simultaneously generates a set of "reasonable routes" for each O-D pair. A portion of the traffic demand will be directly loaded on each of these routes. Unlike Algorithm 3.2 for solving the probit-based model, the STOCH algorithm is not iterative. The step by step procedure of STOCH is presented as follows:

Algorithm 3.3

Step 1. For each O-D pair rs and each link $ij \in L$, calculate:

$$v_{ij} = \begin{cases} e^{\theta[r(j)-r(i)-t_{ij}]}, & \text{if } r(i) < r(j) \text{ and } s(i) > s(j), \\ 0, & \text{otherwise;} \end{cases}$$

where $r(i)$ is the travel time from origin node r to node i along the minimum travel time path; $s(i)$ is the travel time from node i to destination node s along the minimum path; θ is a parameter;

Step 2. Calculate link weight w_{ij}:

$$w_{ij} = \begin{cases} v_{ij}, & \text{if } i = r, \ r \text{ is the origin,} \\ v_{ij} \sum_{m \in F_i} w_{mj}, & \text{otherwise;} \end{cases}$$

where F_i is the set of upstream nodes of all links arriving at node i.

Step 3. Load the traffic demands:

$$x_{ij} = \begin{cases} d_{rs} \dfrac{w_{ij}}{\sum_{m \in F_j} w_{mj}} & \text{if } j = s, \ s \text{ is the destination,} \\ \sum_{m \in O_j} x_{jm} \dfrac{w_{ij}}{\sum_{m \in F_j} w_{mj}} & \text{for all other links } ij, \end{cases}$$

where O_i is the set of downstream nodes of all links leaving node i.

Discussion of the Two Stochastic Loading Models

The two algorithms presented above can be used to carry out stochastic traffic loading without path enumerations. Therefore, there is no computational barrier for both of them to be used in realistic large networks. In Algorithm 3.2 for solving the

probit-based model, the parameter β in the normal distribution $N(t_{ij}, \beta t_{ij})$ reflects the level of distortions of the travellers. The larger the β, the larger the probability that a non-minimum travel time route is believed to be the minimum travel time route. In Algorithm 3.3, θ is the parameter. But in this case, the smaller the θ, the larger the set of the "reasonable routes".

Under certain circumstances, logit-based model may generate unrealistic traffic loading results which can be avoided by the probit-based loading model (Sheffi (1985)). However, there are two major advantages of the logit-based model and the STOCH algorithm over the probit-based model and the simulation based algorithm (Algorithm 3.2):

- Only a general SUE model is available to formulate the probit-based SUE assignment problem. It can be seen that the objective function of the general model (SUEA-G, presented next) is very complicated. However, there is a simpler mathematical formulation for logit-based SUE assignment. It is less difficult to further explore the logit-based model.

- Logit-based traffic loading needs much less computational time than probit-based loading. To generate SUE assignment, traffic loadings need to be performed for a large number of iterations. Computational cost is a very significant factor in selecting the traffic loading algorithm.

We have conducted a number of computational experiments to compare traffic assignments generated by the two stochastic loading models. The difference in the assignment results between the two assignment models is very small. This can be seen in an example later presented in this chapter. The difference in computational times required by the two algorithms, however, is quite large. Because of the advantages of the logit-based model, we mainly use the logit-based loading model and

the STOCH algorithm for SUE assignment.

Formulations and Approaches for Solving SUE Assignment

A general mathematical formulation for SUE assignment is developed by Sheffi and Powell (1981) and presented in Sheffi (1985). Below is the formulation:
(SUEA-G)

$$\min\{-\sum_{rs} d_{rs} S_{rs}[\mathbf{T}^{rs}(X)] + \sum_{ij} t_{ij}(x_{ij}) - \sum_{ij} \int_0^{x_{ij}} t_{ij}(\omega) d\omega\} \tag{3.9}$$

where:

$$S_{rs}[\mathbf{T}^{rs}(X)] = E[\min_{k \in K_{rs}} \{C_k^{rs}\} | \mathbf{T}^{rs}(X)] \tag{3.10}$$

subject to:

$$X \in \mathbf{X}(\mathbf{F}) \tag{3.11}$$

where in Eqs(3.10) and (3.11), $\mathbf{T}^{rs} = [..., T_k^{rs}, ...]$ with T_k^{rs} being the travel time along the k-th route for O-D pair rs; C_k^{rs} is a random variable with $E[C_k^{rs}] = T_k^{rs}$ and K_{rs} is the set of all possible routes for O-D pair rs.

The model presented in Eqs(3.9) to (3.11) is a general formulation. There is no any other specific requirement for the distribution of the random variable C_k^{rs} except its mean value should be equal to the actual travel time on the path k. So it can be used to formulate both probit-based and logit-based stochastic assignment models.

It can be seen that the objective function, Eq(3.9), in the SUEA-G model is very complicated. It is difficult to develop an efficient method to evaluate this function. If this function is not evaluated, then the Convex Combinations algorithm cannot be applied. Because objective function of the model needs to be evaluated in Step 3

of Algorithm 3.1 when the Convex Combinations method is used to solve the DUE assignment problems.

One method that will converge for solving this model is the method of successive averages (MSA). The MSA method is based on the Robbins-Monro approximation approach (Robbins and Monro, 1951) for searching the solution of a single variable function in which the nature of the function is unknown. The method was extended by Blum (1955) for solving multiple variable function problems. MSA is discussed in detail in Wilde (1966) and Sheffi (1985). It is a similar method to the Convex Combinations algorithm in solving traffic assignment problems. Recall the explanation of the iterative procedure to achieve the deterministic traffic assignment given in the second paragraph of Section 3.2.3. The basic idea of MSA is the same as explained there. The only difference is in the traffic reloading process. In the optimization procedure of the Convex Combinations method, at each iteration, travellers will be reloaded to the shortest paths found in the previous iterations, in the way that the objective function (the total travel cost) is minimized. Thus the objective function needs to be evaluated in every iteration. While in the MSA procedure, at each iteration, the number of travellers are evenly distributed among all the shortest paths found in the previous iterations and hence there is no need to evaluate the objective function. In the optimization terminology, Convex Combinations and MSA use the same procedure to search the descent direction (find the shortest path in the network) in solving a non-linear minimization problem. But the two methods are different at finding the step length along the descent direction. In the Convex Combinations method, the step length is determined by minimizing the objective function at each iteration. In MSA, the step length is predetermined to be $1/n$ and is not optimized. Using MSA, one does not need to evaluate the objective function

in each iteration. But achieving convergence by MSA will normally need a large number of iterations compared to the Convex Combinations method. When the objective function is too complicated to evaluate or is not clearly specified, MSA will be a good method to generate a convergent assignment solution. The MSA algorithm to solve the (SUEA-G) model is presented as follows:

Algorithm 3.4

Step 0. Let $n = 0$ and $x_{ij}^{(n)} = x_{ij}^{(0)} = 0, \quad ij \in L.$

Step 1. Generate $t_{ij}^{(n)} = t_{ij}(x_{ij}^{(n)}), \quad ij \in L.$

Step 2. Use Algorithm 3.2 (Algorithm 3.3) to perform probit-based (logit-based) traffic loading based on current travel time $t_{ij}^{(n)}$, find an auxiliary link flow $y_{ij}^{(n)}$ (the search direction).

Step 3. Compute: $x_{ij}^{(n+1)} = x_{ij}^{(n)} + \frac{1}{n+1}(y_{ij}^{(n)} - x_{ij}^{(n)}), \quad ij \in L.$

Step 4. If: $S_v = \frac{\sum_{ij}[\frac{1}{I}\sum_{j=n-I+1}^{n}(x_{ij}^{(j)} - \bar{x}_{ij}^{n,I})^2]^{1/2}}{\sum_{ij}\bar{x}_{ij}^{n,I}} \leq \epsilon$, stop; otherwise, set $n = n + 1$, go to Step 1 (where: $\bar{x}_{ij}^{n,I} = \frac{1}{I}\sum_{j=n-I+1}^{n} x_{ij}^{(j)}$).

Convergence of the MSA algorithm has been proven (Powell and Sheffi (1982)) for both probit-based and logit-based traffic loading methods.

Another algorithm based on an incremental approach can also be used to generate SUE assignment. The algorithm is presented as follows:

Algorithm 3.5

Step 0. Let $n = 0$ and $x_{ij}^{(n)} = x_{ij}^{(0)} = 0, \quad ij \in L.$

Let $\bar{d}_{rs} = d_{rs}/K$, where K is a predetermined large integer.

Step 1. Generate $t_{ij}^{(n)} = t_{ij}(x_{ij}^{(n)}), \quad ij \in L.$

Step 2. Use Algorithm 3.2 (Algorithm 3.3) to perform probit-based (logit-based) traffic loading based on the current travel time $t_{ij}^{(n)}$.

Find $\bar{x}_{ij}^{(n)}$ based on $\bar{d}_{rs}$.

Step 3. Let $x_{ij}^{(n+1)} = x_{ij}^{(n)} + \bar{x}_{ij}^{(n)} \quad ij \in L.$

If $n = K$, stop; otherwise, $n = n + 1$, go to Step 1.

The incremental algorithm is useful in solving certain traffic assignment problems. However, convergence of the algorithm is not guaranteed.

We have mentioned that there is no simple formulation for the probit-based SUE assignment model. But there is a comparably simple mathematical programming model for the logit-based SUE assignment formulated by Fisk (1980):

(SUEA-L)

$$\min_x \{\frac{1}{\theta} \sum_{rs} \sum_k f_k^{rs} \ln(f_k^{rs}) + \sum_{ij \in L} \int_0^{x_{ij}} t_{ij}(\omega) d\omega\} \tag{3.12}$$

subject to:

$$X \in \mathbf{X}(\mathbf{F}) \tag{3.13}$$

It can be noted that Eq(3.12) is much simpler than the objective function of the SUEA-G model (Eqs(3.9) to (3.11)). More efficient algorithms have been developed in addition to the MSA method. In order to present these new algorithms, the linear constraints Eq(3.2) in defining the feasible region of the traffic flow will be expressed in the following matrix format:

$$X = F\Delta \tag{3.14}$$

or

$$X^T = \Delta^T F^T; \tag{3.15}$$

where:

$X = \{..., x_{ij}, ...\}$,

$F = \{..., f_k^{rs}, ...\}$

$(\Delta^{(rs)})_{ij,k} = \delta_{ij,k}^{rs}$, and

$\{*\}^T$ is the transpose of a matrix or a vector $\{*\}$.

As discussed above, the logit-based SUE model can be solved by Algorithm 3.4 (MSA) or Algorithm 3.5 (Incremental). The advantage of using the MSA method is the guarantee of the convergence without evaluating the objective function Eq(3.12). Since the step length in the MSA search process is fixed, a rather large number of iterations is normally required to achieve equilibrium. In this section, we present two new algorithms (Algorithms 3.6 and 3.7) for solving the (SUEA-L) model. Because the objective function of this model is not too complicated to evaluate, evaluation of Eq(3.12) is carried out in Algorithms 3.6 and 3.7, and an optimized step length is used.

In Algorithm 3.6, we evaluate the second term of Eq(3.12) to get a better step length ζ_n in each iteration instead of the fixed value $1/n$. The second term in Eq(3.12) is selected first because it is much easier to evaluate than the whole function. The step by step procedure of this algorithm is the same as that of Algorithm 3.4, except in Step 3. In Step 3 of Algorithm 3.6, the golden section method is used to get the step length ζ_n. Algorithm 3.6 is presented below.

Algorithm 3.6

Step 0. Same as Step 0 of Algorithm 3.4.

Step 1. Same as Step 1 of Algorithm 3.4.

Step 2. Using the STOCH algorithm to perform the logit-based traffic

assignment based on the current travel time $t_{ij}^{(n)}$, find and auxiliary link flow $y_{ij}^{(n)}$ (the search direction):

$y_{ij}^{(n)} = \sum_{rs} \sum_k d_{rs} P_{(n),k}^{rs} \delta_{ij,k}^{rs,(n)}$ $,ij \in L$, where

$P_{(n),k}^{rs}$ is the probability generated by logit-based loading approach.

Step 3. Compute: $x_{ij}^{(n+1)} = x_{ij}^{(n)} + \zeta_n (y_{ij}^{(n)} - x_{ij}^{(n)}), \quad ij \in L$, where

ζ_n is found by solving: $\min_{\zeta_n \in (0,1)} \sum_{ij \in L} \int_0^{x_{ij}^{(n)} + \zeta_n (y_{ij}^{(n)} - x_{ij}^{(n)})} t_{ij}(\omega) d\omega$.

Step 4. Same as Step 4 of Algorithm 3.4.

Convergence of Algorithm 3.6 is not proved but it can be seen in a numerical example presented later.

In Algorithm 3.7 presented next, the evaluation of the whole objective function, Eq(3.12), is carried out to obtain a step length closer to the optimal one at each iteration of the search process. Since the first term of Eq(3.12) is expressed by the route volume f_k^{rs} rather than link volume x_{ij}, a transformation of the expression is required. The inverse or the generalized inverse of the incidence matrix Δ is used to allow us to express variable f_k^{rs} in terms of x_{ij}. Since the solution of the linear equations in Eq(3.15) is basically a least square solution, the step length based on this transformation is not necessarily optimal. However, the step length, also obtained by the golden section method, is the best one that can be expected from the formulation of Eqs(3.12)-(3.13). The transformation is as follows.

i) If Δ is an invertable square matrix and the inverse of its transpose is Γ_1:

$$\Gamma_1 = (\Delta^T)^{-1},$$

then from Eq(3.15):

$$F^T = \Gamma_1 X^T, \tag{3.16}$$

and the transformation is completed.

ii) The condition that Δ is an invertable square matrix does not always hold. When the matrix Δ is not square, the set of the linear equations (Eq(3.15)) does not have a unique solution. One approach for solving this type of linear equations is to obtain a least square solution of the system. Assume that Δ is not a square matrix but its rank is equal to the number of its rows. Then the matrix $(\Delta\Delta^T)$ is invertable, and an approximate solution of F can be achieved. Let:

$$\Gamma_2 = (\Delta\Delta^T)^{-1}\Delta,$$

then:

$$\bar{F}^T = \Gamma_2 X^T. \tag{3.17}$$

In fact, Γ_2 is a special type of generalized inverse matrix of Δ^T and $\bar{F}$ is the least square solution of F. For more detailed discussion of the generalized inverse of matrices, one can refer to Strang (1980).

iii) In the most general case, even the inverse of matrix $(\Delta\Delta^T)$ does not exist. Then the generalized inverse of the incidence matrix can be used. Assume that the row dimension and column dimension of Δ^T are k and l, respectively. Also assume that the rank of this matrix is r. Then the "$\bar{L} - \bar{U}$" factorization method can be used to obtain:

$$\Delta^T = \bar{L}\bar{U},$$

where $\bar{L}$ and $\bar{U}$ are k by r and r by l matrices, respectively. From $\bar{L}$ and $\bar{U}$, it is easy to construct the generalized inverse of Δ^T:

$$\Gamma_3 = \bar{U}^T(\bar{U}\bar{U}^T)^{-1}(\bar{L}^T\bar{L})^{-1}\bar{L}^T,$$

and then:

$$\bar{F}^T = \Gamma_3 X^T.$$

The solution $\bar{F}$ obtained from Γ_3 is also a least square solution of Eq(3.15) (Strang (1980)).

Using the above transformation, Algorithm 3.7 has been developed. The step by step procedure of the algorithm is presented below. In the algorithm, Γ represents Γ_1, Γ_2 or Γ_3 and is not distinguished. $\gamma_{k,i}^{rs}$ are the elements of Γ, $\forall r, s, k, i$.

Algorithm 3.7

Step 0. Same as Step 0 of Algorithm 3.6.

Step 1. Same as Step 1 of Algorithm 3.6.

Step 2. Same as Step 2 of Algorithm 3.6.

Step 3. From $y_{ij}^{(n)}$, generate $\Delta^{(n)}$ and calculate $\Gamma^{(n)}$.

Step 4. Compute: $x_{ij}^{(n+1)} = x_{ij}^{(n)} + \zeta_n(y_{ij}^{(n)} - x_{ij}^{(n)})$, $ij \in L$, where

ζ_n is found by solving:

$\min_{\zeta_n \in (0,1)} \{\frac{1}{\theta} \sum_{rs} \sum_k [\bar{f}_{k,(n)}^{rs}(\zeta_n) \ln \bar{f}_{k,(n)}^{rs}(\zeta_n)] +$

$+ \sum_{ij \in L} \int_0^{x_{ij}^{(n)} + \zeta_n (y_{ij}^{(n)} - x_{ij}^{(n)})} t_{ij}(\omega) d\omega\}$,

where: $\bar{f}_{k,(n)}^{rs}(\zeta_n) = \sum_{ij} \gamma_{k,i}^{(n),rs} [x_{ij}^{(n)} + \zeta_n (y_{ij}^{(n)} - x_{ij}^{(n)})]$, and

$\gamma_{k,i}^{(n),rs} = (\Gamma^{(n)})_{k,i}^{rs}$.

Step 5. Same as Step 4 of Algorithm 3.6.

It should be noted that, although there is no need for path enumeration in using the STOCH method nor in generating the incidence matrix Δ, the generation of Δ requires certain computational effort. The amount of computation of Γ is a function of the parameter θ in Eq(3.12). The smaller the θ, the larger the incidence matrix. This is because a small θ is associated with a large number of "efficient routes". In a

realistic large network, if a very small θ is used, the number of the "efficient routes" for each O-D pair could be very large and the generation of Δ will require a longer computational time. However, it may not be realistic that a very small θ be used in a stochastic assignment. Computational results and comparison of Algorithms 3.4, 3.6, and 3.7 are illustrated by Example 3.3. The development of the new algorithms presented in this section (Section 3.2) will appear in Chen and Alfa (1991a).

A Brief Summary of the Algorithms

In these two subsections (Sections 3.2.3 and 3.2.4) we have discussed five traffic assignment algorithms (Algorithms 3.1, 3.4, 3.5, 3.6 and 3.7) and two stochastic traffic loading algorithms (Algorithms 3.2 and 3.3). Algorithm 3.1 is the Convex Combinations method which can efficiently solve DUE traffic assignment problems. Algorithms 3.2 and 3.3 are stochastic traffic loading methods where one is probit-based and the other is logit-based. These two algorithms are the components of the probit-based and the logit-based SUE assignment procedures, respectively. Algorithm 3.4 is the MSA algorithm. It can generate convergent solutions for DUE or SUE (probit-based or logit-based) traffic assignments without evaluating the objective function. Algorithm 3.5 is an incremental algorithm for DUE or SUE assignments and is a heuristic method. Convergence of this algorithm cannot be guaranteed, but it is useful in a network design method (presented in the next section) for reducing the computational burden. Algorithms 3.6 and 3.7 were developed for solving logit-based SUE traffic assignment model, SUEA-L. They cannot be used to solve a probit-based model. These two algorithms are heuristic but they need a smaller number of iterations to generate logit-based SUE traffic assignments than the MSA method.

3.2.5 Discussion of DUE and SUE Assignment Models

The stochastic user equilibrium (SUE) approach provides a more realistic and better representation of the travellers' route selection process than the DUE. However, SUE based assignment algorithms require more computational effort.

Sheffi and Powell (1981) compared DUE and SUE approaches for traffic assignment. They found that as the level of congestion of the network increases, the two approaches tend to produce about the same assignment results. However, Sheffi and Powell (1981) noticed that, at low demand levels, SUE models overestimated (DUE models underestimated) the flows.

The value of the parameters (β in the probit-based model and θ in the logit-based model) used for the variance may have varying effects on assignment results. In addition to being more realistic, the SUE approach is more flexible for calibration to a network. Once the link travel time functions have been calibrated for the links of a particular network, the DUE can only produce one assignment result. But because of the possible incorrect assumption made in the DUE model, assignment generated by DUE algorithms may not be representative of the travellers' behavior in selecting of their routes. Such results might lead to wrong decisions regarding network improvements. On the other hand, with a logit-based SUE, the parameter θ in Eq(3.12) can be calibrated to reflect the imperfection of travellers' perception of the travel time in the system and thereby reproduces the assignment more closely.

To further emphasize the importance of this parameter in calibration, consider two hypothetical cities which have exactly identical networks and origin-destination trip matrices. All the matching links of both networks have the same characteristics. When a traffic assignment is carried out for both cities using the DUE approach the resulting traffic flows will be the same for the matching links. If however the

travellers in the two cities behave differently and have different knowledge of travel time in the system, their route selection processes could be different which might result in different traffic flows on the matching links. Such differences cannot be reflected by the DUE. But by using the SUE, the θ can be varied to reflect the differences. This idea is further illustrated in a network design example, Example 3.4.

3.2.6 Assignment Examples

Numerical examples are presented for illustration of the assignment models. Example 3.1 gives a comparison of the probit-based and logit-based SUE assignments. Example 3.2 gives a comparison of the incremental method and the method of successive averages; Example 3.3 compares the results by Algorithms 3.4, 3.6 and 3.7.

Example 3.1.

Consider network G_0 shown in Figure 3.1. The parameters of the links in G_0 are given in Table 3.1. It is assumed that there are four O-D pairs in the network and the demands are shown in Table 3.2. Probit-based and logit-based stochastic traffic loadings are performed by Algorithms 3.2 and 3.3, respectively. The computational results of the SUE assignments using Algorithm 3.4 are presented in Table 3.3. CPU times for the probit-based and logit-based assignments are 4.06 and 0.30 seconds, respectively. It can be seen from Table 3.3 that the differences in the traffic volumes generated by probit-based and logit-based assignments are very small. The difference in the computational time of the assignments based on the two loading methods is quite significant. Based on this observation, we will only consider using the logit-based assignment method for solving a stochastic network design problem.

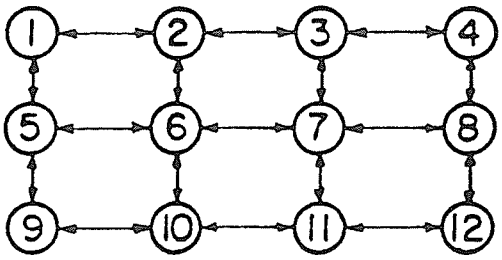

Figure 3.1: Network G_0

Table 3.1: Data for Network G_0

(i,j)	$a_{i,j}$	$b_{i,j}$	(i,j)	$a_{i,j}$	$b_{i,j}$	(i,j)	$a_{i,j}$	$b_{i,j}$	(i,j)	$a_{i,j}$	$b_{i,j}$
1,2	20.0	0.008	1,5	18.0	0.008	2,1	20.0	0.008	2,3	23.0	0.008
2,6	19.0	0.008	3,2	23.0	0.008	3,4	17.0	0.008	3,7	16.0	0.008
4,3	17.0	0.008	4,8	22.0	0.008	5,1	18.0	0.008	5,6	14.0	0.008
5,9	24.0	0.008	6,2	19.0	0.008	6,5	14.0	0.008	6,7	17.0	0.008
6,10	20.0	0.008	7,3	16.0	0.008	7,6	17.0	0.008	7,8	13.0	0.008
7,11	26.0	0.008	8,4	22.0	0.008	8,7	13.0	0.008	8,12	19.0	0.008
9,5	24.0	0.008	9,10	7.0	0.008	10,6	20.0	0.008	10,9	7.0	0.008
10,11	18.0	0.008	11,7	26.0	0.008	11,10	18.0	0.008	11,12	17.0	0.008
12,8	19.0	0.008	12,11	17.0	0.008						

Table 3.2: Demands of 4 Origin-Destination Pairs of Example 3.1

O-D Pairs	(1,12)	(4,9)	(9,4)	(12,1)
Demand	5.0	8.0	6.0	10.0

Table 3.3: Volume Comparison of Probit- and Logit-based Methods

(i,j)	$x^l_{i,j}$	$x^p_{i,j}$	(i,j)	$x^l_{i,j}$	$x^p_{i,j}$	(i,j)	$x^l_{i,j}$	$x^p_{i,j}$	(i,j)	$x^l_{i,j}$	$x^p_{i,j}$
1, 2	2.70	2.72	1, 5	3.17	3.32	2, 1	5.90	5.81	2, 3	2.52	2.57
2, 6	3.01	3.07	3, 2	5.33	5.30	3, 4	3.74	3.82	3, 7	3.39	3.22
4, 3	5.79	5.69	4, 8	3.43	3.64	5, 1	4.98	5.23	5, 6	3.45	3.33
5, 9	3.39	3.66	6, 2	3.39	3.44	6, 5	5.58	5.71	6, 7	4.78	4.45
6,10	3.84	3.73	7, 3	4.16	4.08	7, 6	6.70	6.58	7, 8	3.88	3.78
7,11	2.72	2.60	8, 4	3.47	3.51	8, 7	5.92	6.06	8,12	3.06	3.18
9, 5	3.06	3.19	9,10	4.85	4.97	10, 6	4.44	4.33	10, 9	6.52	6.50
10,11	3.71	3.98	11, 7	3.37	3.31	11,10	5.98	6.12	11,12	3.38	3.45
12, 8	5.13	5.33	12,11	6.30	6.30		$sd_{(p,l)}$:	0.139			

where: $sd_{(p,l)} = \frac{\sqrt{\sum_{ij}^{L}(x^p_{ij}-x^l_{ij})^2}}{L-1}$, x^p_{ij} and x^l_{ij} are flows on link ij generated by probit-based and logit-based assignments, respectively. L is the number of total links in G_0.

Example 3.2.

This example is used to compare the stochastic MSA (Algorithm 3.4) and the incremental assignment method (INC, Algorithm 3.5). Traffic assignments were performed on network G_0 shown in Figure 3.1. The associated data of network G_0 are given in Table 3.1. The demands of four origin-destination pairs are shown in Table 3.2. Three values of θ, $\theta = 1.0, 0.01$, and 0.001 were used in this example. Table 3.4 shows the traffic volumes generated by the MSA and INC methods for various values of θ. Table 3.5 shows the standard deviation (SD) of the link volumes from Table 3.4. Table 3.6 shows the other computational results of the two assignment methods. As shown in Table 3.5, deviations between the MSA method and the incremental method are 0.535, 0.080, and 0.009 for $\theta = 1.0., 0.01$ and 0.001, respectively. It shows that the incremental method can produce traffic assignments similar to the ones produced by the MSA method, especially when the inperfection of travellers' knowledge is more significant (θ is smaller).

Table 3.4: Volume Comparison of Stochastic MSA and INC Methods

Assig. Method	MSA*	INC*	MSA†	INC †	MSA‡	INC ‡
Link						
1,2	1.18	0.43	3.40	3.40	3.58	3.58
1,5	4.28	4.69	2.82	2,81	2.80	2.80
2,1	5.52	4.75	6.48	6.46	6.76	6.76
2,3	1.80	1.16	3.10	3.08	3.28	3.28
2,6	2.63	2.23	3.11	3.02	3.10	3.09
3,2	5.25	4.87	5.32	5.10	5.39	5.37
3,4	3.47	3.86	3.99	3.99	4.09	4.09
3,7	3.82	3.96	3.40	3.44	3.40	3.40
4,3	5.77	5.89	5.76	5.66	5.79	5.78
4,8	2.27	2.12	3.67	3.72	3.70	3.70
5,1	4.94	5.37	4.75	4.75	4.62	4.62
5,6	4.25	4.28	3.32	3.30	3.30	3.30
5,9	3.59	3.42	3.55	3.52	3.68	3.68
6,2	3.51	2.84	3.95	4.06	4.17	4.19
6,5	5.82	5.89	5.44	5.44	5.41	5.40
6,7	4.92	5.81	4.50	4.52	4.41	4.41
6,10	4.66	4.50	3.50	3.56	3.41	3.42
7,3	4.98	5.64	3.86	3.80	3.80	3.80
7,6	6.93	6.46	7.01	7.31	7.18	7.21
7,8	4.79	4.28	3.43	3.42	3.31	3.31
7,11	0.83	0.56	3.11	3.06	3.10	3.10
8,4	2.57	2.15	3.44	3.39	3.40	3.40
8,7	6.46	5.89	5.57	5.59	5.42	5.42
8,12	3.92	4.52	2.84	2.86	2.81	2.81
9,5	2.68	2.49	3.35	3.32	3.40	3.39
9,10	5.57	6.29	4.28	4.30	4.12	4.12
10,6	5.10	6.07	3.96	3.95	3.82	3.82
10,9	6.66	7.36	6.09	6.10	5.84	5.83
10,11	4.28	4.03	3.40	3.40	3.31	3.31
11,7	2.33	1.29	3.92	4.02	4.17	4.18
11,10	5.81	6.67	5.67	5.59	5.44	5.42
11,12	2.74	2.86	3.58	3.59	3.60	3.60
12,8	5.89	6.15	4.76	4.70	4.62	4.61
12,11	5.77	6.22	6.66	6.75	6.79	6.80

Table 3.5: Standard Deviations of Traffic Volumes

Assig. Method	MSA$*$	INC$*$	MSA$\dagger$	INC$\dagger$	MSA$\ddagger$	INC$\ddagger$
MSA$*$	0.0	0.535	1.027	1.034	1.117	1.119
INC$*$		0.0	1.386	1.396	1.486	1.488
MSA$\dagger$			0.0	0.080	0.132	0.137
INC$\dagger$				0.0	0.134	0.134
MSA$\ddagger$					0.0	0.009
INC$\ddagger$						0.0

$sd_{(u,v)} = \frac{\sqrt{\sum_{ij}^{L}(x_{ij}^{u}-x_{ij}^{v})^2}}{L-1}$
where: $sd_{(u,v)}$ is the entry in Table 3.5,
x_{ij}^{u} and x_{ij}^{v} are flows on link ij generated by different algorithms (or θ) u and v,
L is the number of total links in network G_0.

Table 3.6: Other Computational Results of Example 3.2

Assig. Method	Travel Cost	$K_{\epsilon\leq 0.05}$	CPU Time
MSA$*$	0.339	15	2.20 sec.
INC$*$	0.357	(NK=15)	2.01 sec.
MSA$\dagger$	0.336	7	1.04 sec.
INC$\dagger$	0.338	(NK= 7)	0.96 sec.
MSA$\ddagger$	0.339	7	1.04 sec.
INC$\ddagger$	0.339	(NK= 7)	0.95 sec.

Note in Tables 3.4,3.5 and 3.6:
C-C: Convex combinations method,
MSA: Method of successive average,
INC: Incremental method,
$*$: $\theta = 1.0$,
$\dagger$: $\theta = 0.01$,
$\ddagger$: $\theta = 0.001$,
NK: Number of portions by which the traffic demands are divided.
The scale of Column 2 in Table 3.8 is 10,000.

Table 3.7: Data for Network G

(i,j)	$a_{i,j}$	$b_{i,j}$	(i,j)	$a_{i,j}$	$b_{(i,j)}$
(1, 2)	20.0	0.008	(1, 5)	18.0	0.008
(2, 3)	23.0	0.008	(2, 6)	19.0	0.008
(3, 4)	17.0	0.008	(3, 7)	16.0	0.008
(4, 8)	22.0	0.008	(5, 6)	14.0	0.008
(5, 9)	24.0	0.008	(6, 7)	17.0	0.008
(6,10)	20.0	0.008	(7, 8)	13.0	0.008
(7,11)	26.0	0.008	(8,12)	19.0	0.008
(9,10)	7.0	0.008	(10,11)	18.0	0.008
(11,12)	17.0	0.008	-	—	—

Example 3.3

Example 3.3 is designed to compare Algorithms 3.4, 3.6 and 3.7 with the logit-based traffic loading in Algorithm 3.4. It is assumed that there is only one origin-destination pair from node 1 to node 12 in network G as shown in Figure 3.2. Stochastic assignments were performed by the three algorithms with a demand of 100. Data for the network G are shown in Table 3.7. Variations on parameter θ is shown in Table 3.8. Comparisons of the convergence behaviors of the algorithms with $\theta = 0.01$ are presented in Figures 3.3, 3.4 and 3.5. In these three figures, the horizontal axis represents the iteration number while the vertical axis represents total travel time. The unit on the vertical axis in each of the figures is 100,000.0.

Other computational results are also given in Table 3.8. It can be seen that the convergence behavior of the MSA method is improved by the optimization of the step length ζ into the procedure for generating the logit-based SUE assignment.

The number I in the last step of Algorithms 3.4, 3.6 and 3.7 was (arbitrarily) set to 7. The convergence criterion ε for Examples 3.1, and 3.3 was 0.001. ε for

Table 3.8: Computational Results of Example 3.3

θ	Algorithm #	Travel cost	$N_{\epsilon<0.05}$	CPU Time
0.01	1	126.83	24	0.92 sec
	2	126.04	9	0.50 sec
	3	126.04	9	0.54 sec
0.05	1	126.56	25	0.95 sec
	2	126.88	7	0.40 sec
	3	126.88	7	0.43 sec
0.1	1	128.21	23	0.88 sec
	2	126.86	7	0.40 sec
	3	126.86	7	0.43 sec
1.0	1	126.09	23	0.90 sec
	2	128.63	12	0.66 sec
	3	128.63	12	0.72 sec

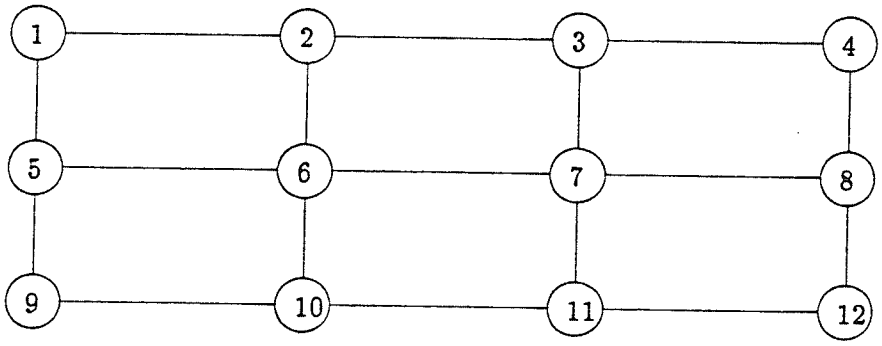

Figure 3.2: Network G (Example 3.3)

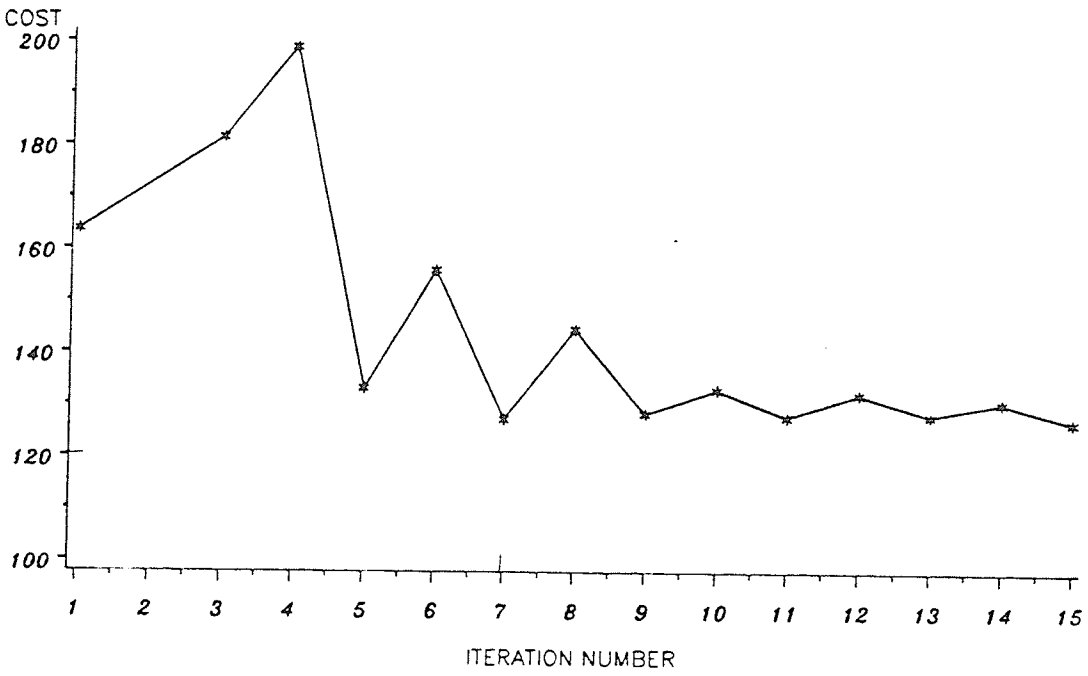

Figure 3.3: Travel Cost of Each Iteration (Algorithm 3.4)

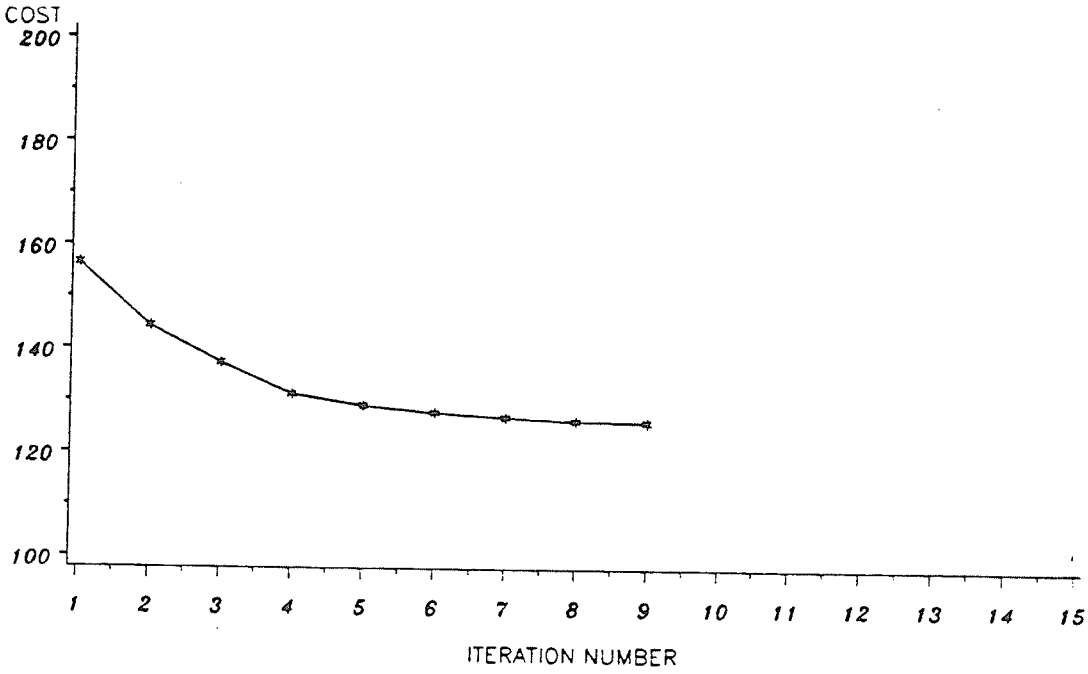

Figure 3.4: Travel Cost of Each Iteration (Algorithm 3.6)

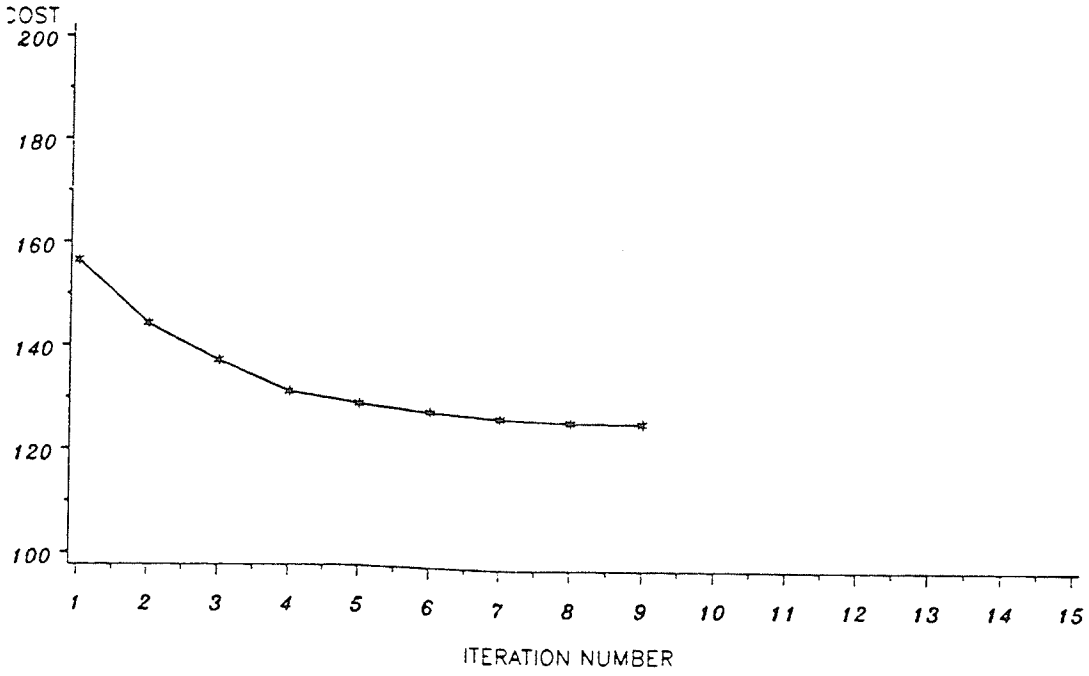

Figure 3.5: Travel Cost of Each Iteration (Algorithm 3.7)

Example 3.2 is shown in Table 3.6.

3.3 Network Design Models

Various transportation network design (ND) models and algorithms have been developed to solve different practical ND problems. These models can be distinguished by the following features:

- those that have continuous or discrete decision variables;
- those in which the construction cost is treated as part of the objective function or as a budget constraint; and
- those that are based on system optimal or user equilibrium assignment.

ND models with continuous decision variables may generate non-integer results which need to be further rounded off to obtain the final decision. So the results obtained by continuous optimization models and algorithms may not be directly

implemented. The results generated by discrete ND models and algorithms, on the other hand, normally can be directly implemented. This is the main reason that we will, in this section, concentrate our effort on solving discrete ND problems with SUE assignment and with the construction cost being part of the objective function.

Before the network design problems are discussed, we will recall the definitions given at the beginning of Section 3.2. Some of the definitions may be modified and some new definitions will be given.

Define a network $G_0 = \{V, L_0\}$, where V and L_0 are the set of nodes and set of links of G_0. Assume that, in order to improve the capability of a transportation network, there are a number of new roads (links) that have been proposed to be constructed into the network G_0, and/or, there are a number of old roads (links) in the network G_0 that have been proposed for improvement.

We call all these proposed links (for construction or improvement) candidate links. For the problems discussed in this section, we assume that the total number of candidate links is M. Let C_l denote the construction cost of building a new link or improving an old link l. We define a candidate project as the construction of a set of candidate links. Therefore, for M candidate links, we have 2^M different candidate projects, including the "do-nothing project" in which no candidate links will be constructed/improved.

The decisions of implementing different candidate projects are represented by the decision variables U_h, $h = 0, 1, 2, ..., D$ $(D = 2^M - 1)$:

$U_0 = (0, 0, ..., 0, 0, 0)$

$U_1 = (0, 0, ..., 0, 0, 1)$

$U_2 = (0, 0, ..., 0, 1, 0)$

$\vdots$

$U_h = (u_{j,1}, u_{j,2}, ..., u_{j,l}, ..., u_{j,M})$

$\vdots$

$U_D = (1, 1, ..., 1, 1, 1)$,

where U_0 represents the "do-nothing project", U_1 represents the project in which only the M-th candidate link will be constructed, ..., and U_D represents the project where all the candidate links will be constructed. The vectors $U_h, h = 0, ..., D$ are generated by the following procedure:

i). Let $U_0 = (0, 0, ..., 0, 0, 0)$ and $U_1 = (0, 0, ..., 0, 0, 1)$.

ii). For $w = 1, ..., M-1$, $v = 0, ..., 2^w - 1$,

$u_{v+2^w,l} = u_{v,l}$, $l = 1, ..., M$, and $u_{v+2^w,M-w} = 1$.

The construction cost of each candidate project is denoted by C_h^P, $h = 0, 1, ..., D$. We define C_h^P to be the sum of the construction cost of the links belonging to the candidate project h. The objective of the ND problems is to select a decision variable U^* (accordingly, a network G^*) from U_h, $h = 0, 1, ..., D$, in order that the total cost of the transportation system is minimized. The total cost is the sum of the converted construction cost and the total travel time spent by motorists in the network $G_h = (N, L_h)$, where L_h is the set of links including the links in L_0 and those links in the candidate project h, $h = 0, 1, ..., D$. Note that all the networks $G_h = (N, L_h)$, $h = 0, 1, ..., D$, have the same set of node N.

Similar to the definitions given in Section 3.2, let $X = [x_1, x_2, ..., x_{ij}, ..., x_{|L_h|}]$. We define $X \in \mathbf{X}(\mathbf{F})_h \Longleftrightarrow$

$$\sum_k f_k^{rs} = d_{rs}, \quad \forall r, s; \tag{3.1'}$$

$$x_{ij} = \sum_{rs} \sum_k f_k^{rs} \delta_{ij,k}^{rs}, \quad \forall ij \in L_h; \tag{3.2'}$$

$$f_k^{rs} \geq 0, \quad \forall k, r, s; \tag{3.3'}$$

$$x_{ij} \geq 0, \quad \forall ij \in L_h; \tag{3.4'}$$

where f_k^{rs}, d_{rs} and $\delta_{ij,k}^{rs}$ have the same definitions as in Eqs(3.1) to (3.4).

3.3.1 A System Optimal Network Design Model

For a controllable transportation system, a system optimal ND model can be used for selecting the optimal candidate project. The purpose of developing and solving such a model is to improve the capabilities of the system while the total cost to construct the project and the total travel time by the motorists in the network is minimized.

The mathematical programming model for a discrete system optimal ND problem can be formulated as follows:

(SOND)

$$\min \sum_{ij \in L_h} x_{ij} t_{ij}(x_{ij}) + \lambda_s \sum_{l=1}^{M} C_l v_l \tag{3.18}$$

subject to:

$$X \in \mathbf{X}(\mathbf{F})_h \tag{3.19}$$

where:

$$v_l = \begin{cases} 1 & \text{if link } l \text{ is constructed,} \\ 0 & \text{otherwise;} \end{cases} \tag{3.20}$$

where λ_s expresses the conversion between construction cost and travel time, and x_{ij} is the traffic flow on link $ij, ij \in L_h$.

The above system optimal model can be solved by using a classical non-linear mathematical programming method and branch and bound approach. Various approaches for solving different system optimal ND models can be found in the extensive survey paper by Magnanti and Wong (1984). Since the main interest of this

research is in the user equilibrium ND models, the system optimal ND problems will not be discussed in more detail in this research.

3.3.2 User Equilibrium Network Design Models

Before the mathematical formulations of the ND models are presented, a phenomenon called the Braess' Paradox, that could happen in solving user equilibrium ND problems needs to be addressed.

It is widely accepted that if a new link has been added to a transportation network, or the capacity of an existing link in the network has been improved, total travel time spent by the travellers at the equilibrium traffic pattern should be reduced. However, an example has been established that shows the opposite. This phenomenon contradicts the common belief and is called the Braess' Paradox. The paradox can be illustrated by applying the DUE assignment or the logit-based SUE assignment algorithm to Braess' hypothetical sample networks (see Sheffi (1985)). Thus ignoring Braess' paradox in any of the approaches for solving user equilibrium ND problems reduces the method to a pure heuristic. On the other hand, it should be pointed out that the parameters used in the performance functions for the Braess' sample network are very unrealistic. It would be difficult to find real networks that have some of the types of the parameters used. The occurrences of the paradox need to be further explored. In the algorithm presented later in this section for solving ND problems, the Braess' Paradox will be ignored.

The mathematical formulations of the ND problems can be developed by minimizing the sum of the total travel time and the (converted) construction cost subject to the user equilibrium traffic patterns in the concerned transportation networks. A mathematical programming model of the ND problem with DUE assignment can

be presented as follows:

(DUEND)

$$\min_{U_h} \sum_{ij \in L_h} x_{ij}^{*} t_{ij}(x_{ij}^{*}) + \lambda_u \sum_{l=1}^{M} C_l v_l \tag{3.21}$$

where v_l are the 0-1 integer variables as in Eq(3.20) and λ_u expresses the conversion between construction cost and travel time, and x_{ij}^{*} is the deterministic user equilibrium traffic flow on link ij which satisfies:

$$(3.7) \text{ and } (3.8). \tag{3.22}$$

Note in Eq(3.22) that $ij \in L_h$.

Various versions of the user equilibrium ND model can have the same objective function as presented in Eq(3.21). Therefore different ND models can be formulated by combining Eq(3.21) with different assignment models. For example, an ND model with a general SUE assignment formulation can be constructed by using Eq(3.21) as the objective function subject to the constraints expressed by Eqs(3.9) to (3.11).

It can be seen that the DUEND model is a two-level non-linear integer programming model. An algorithm based on a a branch and bound method for has been developed in this thesis research for solving this type of ND problems. This method is discussed next.

3.3.3 A Branch and Bound Method for Solving an ND Problem

In this subsection we will focus on an ND problem with logit-based SUE assignment. The reasons for considering this model are:

- SUE assignment is more realistic compared to DUE assignment as discussed earlier;

- there is no sufficient support to justify that the probit-based loading approach is superior to the logit-based loading approach, or vise versa;
- the logit-based loading approach is more efficient than the probit-based loading approach and experiments have shown that the difference in the assignments generated by these two approaches is negligible (one of the examples is presented in Example 3.1).

The detailed mathematical formulation of this model is:

(SUEND-L)

$$\min_{U_h} \sum_{ij \in L_h} x_{ij}^{\times} t_{ij}(x_{ij}^{\times}) + \lambda_u \sum_{l=1}^{M} C_l v_l \tag{3.23}$$

where v_l are the 0-1 integer variables as in Eq(3.20) and λ_u expresses the conversion between construction cost and travel time, and $x_{ij}^{\times}$ is the logit-based SUE traffic flow on link ij which satisfies:

(SUEA-L)

$$\min_{x} \{ \frac{1}{\theta} \sum_{rs} \sum_{k} f_k^{rs} \ln(f_k^{rs}) + \sum_{ij \in L_h} \int_0^{x_{ij}} t_{ij}(\omega) d\omega \} \tag{3.24}$$

subject to:

$$X \in \mathbf{X}(\mathbf{F})_h \tag{3.25}$$

Note that Eq(3.23) is same as Eq(3.21). Eqs(3.24) and (3.25) are the same as Eqs(3.12) and (3.13), respectively. They are re-numbered in this subsection only for the citing purpose.

Next we present a heuristic algorithm which applies the MSA algorithm (Algorithm 3.4) and the incremental method (Algorithm 3.5) for traffic assignment. The algorithm uses a branch and bound method for selecting optimal or sub-optimal network design decisions. The step by step procedure of the branch and bound

algorithm for solving Eqs(3.23) to (3.25) is given below:

Algorithm 3.8

The algorithm is divided into two phases.

Phase 1.

Step 1.0. Let $n = 0$ and $x_{ij}^{(n)} = x_{ij}^{(0)} = 0, \quad ij \in L_0$.

Step 1.1 Generate $t_{ij}^{(n)} = t_{ij}(x_{ij}^{(n)})$, $ij \in L_0$.

Step 1.2. Using the STOCH algorithm to perform a logit traffic assignment based on the current travel time $t_{ij}^{(n)}$, find an auxiliary link flow $y_{ij}^{(n)}$ (the search direction) :

$y_{ij}^{(n)} = \sum_{rs} \sum_k d_{rs} P_{(n),k}^{rs} \delta_{ij,k}^{rs} \quad , \quad ij \in L_0$, where:

$P_{(n),k}^{rs} = \frac{exp(-\theta T_{(n),k}^{rs})}{\sum_l exp(-\theta T_{(n),l}^{rs})} \geq 0$, $\sum_k P_{(n),k}^{rs} = 1$ and $T_{(n),k}^{rs} = \sum_{ij \in L_0} t_{ij}^{(n)} \delta_{ij,k}^{rs}$.

Step 1.3. Set $n = n + 1$. Compute: $x_{ij}^{(n)} = x_{ij}^{(n-1)} + \frac{1}{n}(y_{ij}^{(n-1)} - x_{ij}^{(n-1)}), \quad ij \in L_0$.

Step 1.4. If:

$S_v = \frac{\sum_{ij} [\frac{1}{I} \sum_{j=n-I+1}^{n} (x_{ij}^{(j)} - \bar{x}_{ij}^{n,I})^2]^{1/2}}{\sum_{ij} \bar{x}_{ij}^{n,I}} \leq \varepsilon$,

go to Step 2.0; otherwise, go to Step 1.1.

Where: $\bar{x}_{ij}^{n,I} = \frac{1}{I} \sum_{j=n-I+1}^{n} x_{ij}^{(j)}$.

Phase 2.

Step 2.0. Set $NK = n$. $LB = \sum_{ij \in L_0} x_{ij}^{(NK)} t_{ij}^{(NK)}$.

Set current optimal solution $U^* = U_0 = (0, 0, ..., 0, 0)$.

Set $\Delta d_{rs} = d_{rs}/NK$.

Set $h = D$, where $D = 2^M - 1$.

Step 2.1. For $G_h = \{V, L_h\}$ perform the incremental logit traffic assignment:

Step 2.1.0. Let $m = 0, x_{ij}^{(m)} = x_{ij}^{(0)} = 0$, $ij \in L_h$.

Step 2.1.1. Compute $t_{ij}^{(m)} = t_{ij}(x_{ij}^{(m)})$.

If $TC_h = \sum_{ij \in L_h} x_{ij}^{(m)} t_{ij}^{(m)} + \lambda_u C_h^P > LB$,

and there is no any k, $1 < k < h$, such that $U_h \succ U_k$,

go to Step 2.3; otherwise, go to Step 2.1.2.

(The expression of "$U_h \succ U_k$" means that all the candidate links in project k are included in project h.)

Step 2.1.2. Using the STOCH algorithm, generate:

$\Delta x_{ij}^{(m)} = \sum_{rs} \sum_k \Delta d_{rs} P_{(m),k}^{rs} \delta_{ij,k}^{rs}$,

where $P_{(m),k}^{rs}$ is same as in Phase 1, except here $ij \in L_h$.

Step 2.1.3. Set $m = m + 1$. Compute: $x_{ij}^{(m)} = x_{ij}^{(m-1)} + \Delta x_{ij}^{(m-1)}$, $ij \in L_h$.

If $m > NK$, go to Step 2.2; otherwise, go to Step 2.1.1.

Step 2.2. If $TC_h < LB$, set $LB = TC_h$ and $U^* = U_h$.

For all $k, k < h$ and $U_h \succ U_k$,

if: $\sum_{ij \in L_h} x_{ij}^{(NK)} t_{ij}^{(NK)} + \lambda_u C_k^P \geq LB$,

then eliminate project k from further consideration, go to Step 2.3.

Step 2.3. If $h = 1$, output U^* as the optimal solution, stop;

otherwise, set $h = h - 1$, go to Step 2.1.

Remarks

i) In Algorithm 3.8 presented above, Phase 1 is the logit-based MSA traffic assignment for network G_0. Phase 2 is the branch and bound method for selecting G^* from G_h, $h = 0, 1, ..., 2^M - 1$. The assignment method used in Phase 2 is a logit-based incremental method.

ii) The branch and bound method is used to solve the network design problem without considering Braess' Paradox. That is, we assume that the total travel cost at the user equilibrium pattern in a network with more links will be equal

to or less than the total travel cost in the same network with less links. For an illustration of the branch and bound algorithm, consider two candidate projects, A and B. If:

- project A has more candidate links than project B;
- traffic assignment in the network with project A constructed has been performed and the associated total traffic cost is known; and
- the sum of construction cost of project B and the travel cost in the network with project A constructed is equal to or greater than the sum of the construction cost of project A and the travel cost in the network with project A constructed (assume that this sum is the current lower bound),

then the traffic assignment in the network with project B constructed will not be performed, because the sum of construction cost of project B and the travel cost in the network with project B constructed will be equal to or greater than the current lower bound.

iii) In general, the branch and bound method discussed in **ii)** is presented in Step 2.2 of Algorithm 3.8. In that step, candidate project h is the current project and any of the other candidate projects, project k, will be eliminated if:

- the sum of the converted construction cost $\lambda_u C_k^P$ and the total travel cost associated with any G_h, $h \neq k$, is greater than the current lower bound, and
- all the candidate links of project k are included in the above project h.

However, ignoring the Braess' Paradox is one of the two factors that reduce Algorithm 3.8 to a heuristic method in solving the model.

iv) The logit-based incremental traffic assignment algorithm is used for the assignment on G_h, $h = 1, ..., D$. The purpose for applying the incremental approach is to reduce the number of iterations in the assignment process. If the total cost corresponding to network $G_h = \{V, L_h\}$ $h = 1, ..., D$, exceeds the current lower bound even when not all of the portions have been assigned, then it is possible to eliminate the further assignment iterations and go to the next candidate project.

As mentioned earlier, MSA can generate convergent solutions for solving the assignment problem of Eqs(3.24) and (3.25). The number of iterations NK corresponding to the MSA convergence deviation ε is used for the incremental approach in Phase 2. It is expected that the number of iterations required for the MSA method to converge would also be sufficient for the incremental method to achieve a solution close to the stochastic equilibrium. Normally, such an achievement cannot be guaranteed. In discussing the deterministic assignment method, Ferland et al.(1975) presented a simple network to show that the incremental method might generate an assignment not necessarily of the user equilibrium traffic pattern. The results of their example still apply to the stochastic assignment method used in the above algorithm. This is another reason that makes the algorithm heuristic. However, from the test results of Example 3.2, the travel costs generated by the MSA method and the incremental method vary within a small range, and hence the incremental method is used based on the computational experience. In summary, the algorithm is a heuristic method because:

- Braess' Paradox is ignored, and

- convergence of the incremental method cannot be guaranteed.

3.3.4 Discussion

In this section, we presented a number of transportation ND models and a heuristic algorithm for solving an ND problem with logit-based SUE assignment. We concentrated our research on the discrete ND problems. We believe that this type of problem is more realistic. Since integer variables are involved, the discrete ND problems are more difficult to solve than continuous ND problems.

The stochastic traffic assignment approach has been accepted as being more flexible and more representative for real life route selection. However, due to computational difficulties associated with it, very little effort had been directed to incorporating it into network design problems.

A major achievement in developing Algorithm 3.8 is that the stochastic assignment based MSA and the logit-based incremental method have been incorporated into a network design model. It also shows that:

i) The stochastic traffic assignment model can be incorporated into the ND problems thereby representing the real life route selection process more closely with more flexibility. Therefore, more realistic ND solutions can be expected.

ii) The logit-based model can be used for traffic assignment in solving ND problems although it does require more computational time than the deterministic assignment method.

iii) The incremental assignment method can be used to reduce the number of iterations of traffic assignment.

The results of this section (Section 3.3) will appear in Chen and Alfa (1991b).

3.3.5 Network Design Examples

Example 3.4 illustrates the advantage of using a stochastic traffic assignment model in solving network design problems.

Example 3.4.

A small network g_0 shown in Figure 3.6 is used for this example. The parameters a_{ij} and b_{ij} are shown in Table 3.9. This example compares the differences that could result in a network design problem when the deterministic and stochastic assignment methods are used. In this example, the Convex Combinations method (Algorithm 3.1) and the logit-based MSA (Algorithm 3.4) were used to perform deterministic and stochastic traffic assignments, respectively. Assume that the demand from node 1 to node 4 of g_0 is 10 units. Various values of parameter θ were used in the stochastic assignments. Computational results presented in Table 3.10 are illustrative of the advantage and the flexibility of the SUE approach. Let us assume, for an ND problem, that any link in g_0 with flow larger than or equal to 5.0 units will be improved. According to the DUE approach, links (1,3) and (3,4) will be improved. If, however, we use the SUE assignment, links (1,3) and (3,4) will be improved when $\theta \geq 10.0$, where $\theta \leq 1.0$, links (1,3) and (2,4) will be the ones to be improved. Our decision regarding network improvement will thus depend on θ. This example demonstrates the advantage of using the SUE assignment. Rather than ignoring the appropriateness of using the SUE because of the computational requirements, more effort should be devoted to improving its computational efficiency so it can be used in solving various traffic assignment and ND problems. The number I in Algorithms 3.1 and 3.4 for this example was set to 7. The convergence criterion ε was 0.001.

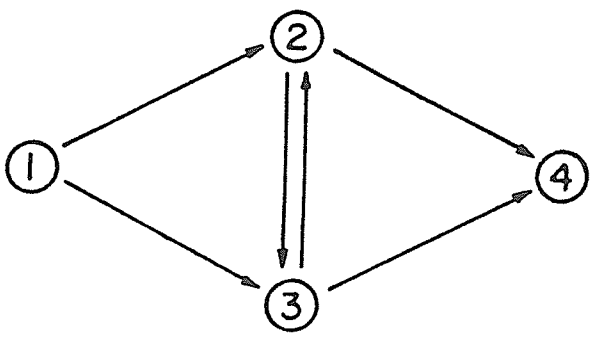

Figure 3.6: Network g_0 (Example 3.4)

Table 3.9: Data for Network g_0

(i,j)	$a_{i,j}$	$b_{i,j}$	(i,j)	$a_{i,j}$	$b_{i,j}$	(i,j)	$a_{i,j}$	$b_{i,j}$
1,2	21.0	0.008	1,3	19.0	0.008	2,3	1.0	0.008
2,4	20.0	0.008	3,2	2.0	0.008	3,4	20.0	0.008

Table 3.10: Link Volume Comparison of DUE and SUE Methods

Assig. Method	C-C	MSA	MSA	MSA	MSA	MSA	MSA	MSA
Link		$\theta = 10.0$	$\theta = 1.0$	$\theta = 0.5$	$\theta = 0.1$	$\theta = 0.01$	$\theta = 0.001$	$\theta = 0.0001$
1,2	4.89	4.87	4.75	4.75	4.75	4.75	4.75	4.75
1,3	5.11	5.13	5.25	5.25	5.25	5.25	5.25	5.25
2,3	0.00	0.00	0.25	0.78	1.35	1.42	1.42	1.41
2,4	4.89	4.87	5.00	5.01	5.07	5.20	5.25	5.25
3,2	0.00	0.00	0.50	1.04	1.67	1.88	1.91	1.92
3,4	5.11	5.13	4.99	4.99	4.93	4.80	4.76	4.75

C-C: Convex Combinations Method
MSA: Method of Successive Averages

Examples 3.5 and 3.6 are applications of the branch and bound network design algorithm.

Example 3.5

Example 3.5 illustrates the branch and bound network design algorithm. Assume that there are four candidate links to be built into G_0. The network G_0 is shown in Figure 3.1. The parameters of G_0 are the same as shown in Table 3.1. Positions of the four candidate links are represented by the broken lines shown in Figure 3.7. The constants a_{ij}, b_{ij} and construction cost C_{ij} of the four candidate links are shown in Table 3.11. It is assumed that there are 2 origin-destination pairs. One is from node 1 to node 12; the other is from node 4 to node 9. Traffic demands from the above origins to destinations are assumed to be 50 units and 30 units, respectively. Results generated by the heuristic algorithm are shown in Table 3.12 for various values of λ_u. For example, it shows that the optimal solution is to add all candidate links into the original network if $\lambda_u = 1.0$. The generated solutions seem reasonable because the number of links is reduced with the increase of the value of λ_u. The algorithm was compared to an enumeration method in which all the 16 candidate projects for the 10 different values of λ_u were evaluated. No better solution was found, which means that Braess' paradox did not arise in this example. As expected, CPU time for the total enumeration procedure over the branch and bound algorithm was longer by about 6 to 10 seconds for this small size problem. This example shows the algorithm solving problems where new links are added. The algorithm can also be used, without major modifications, to solve problems where links are improved.

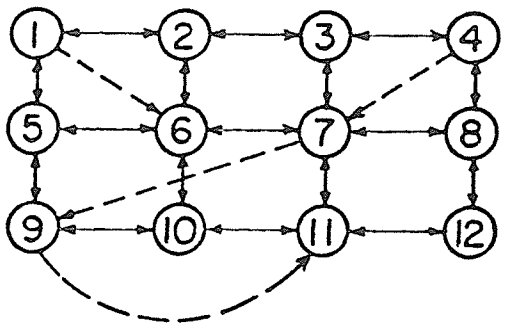

Figure 3.7: Candidate Links in Example 3.5

Table 3.11: Data for Candidate Links of Example 3.5

Link Number	(i,j)	$a_{i,j}$	$b_{i,j}$	$C_{i,j}$
1	1,6	19.0	0.008	7000.0
2	9,11	25.0	0.008	10000.0
3	4,7	19.0	0.008	6000.0
4	7,9	30.0	0.008	8000.0

Table 3.12: Computational Results of Example 3.5

λ_u	Num. of Assig. Performed	Solution Generated	Total Cost	CPU Time (sec.)
1.0	6	(1,1,1,1)	287815.4	5.64
2.0	6	(1,0,1,1)	316544.0	5.62
3.0	7	(1,0,1,1)	337544.0	6.55
4.0	8	(1,0,1,1)	358544.0	7.49
5.0	10	(1,0,1,1)	379544.0	9.25
6.0	10	(1,0,1,1)	400544.0	9.29
7.0	10	(1,0,1,0)	415341.0	9.39
8.0	10	(1,0,1,0)	428341.0	9.26
9.0	10	(1,0,0,0)	436029.2	9.23
10.0	10	(1,0,0,0)	443029.2	9.29

Example 3.6.

The branch and bound algorithm was applied to a realistic network based on the urban transportation system of the city of Winnipeg. Several parallel links were combined and some zones were aggregated. These modifications were performed to limit the size of the network. The adapted network G_w and the 5 candidate links are shown in Figure 3.8. The data for the performance function of each link is shown in Table 3.13 and the demand matrix of the 23 origin-destination pairs is presented in Table 3.14. Two experiments (A and B) were performed in which the construction costs for 4 of the 5 links were changed as shown in Table 3.15. Note that each link ij in Tables 3.11 and 3.13 denotes a "two way road". Node #21 denotes Winnipeg downtown area. The time units given in Tables 3.13 and 3.15 are in minutes and the time units given in Table 3.14 are in hours. For $\lambda_u = 0.6$ and $\lambda_u = 1.0$, the results were (0,0,0,0,1) and (1,0,0,0,0) for experiment A, and (0,0,0,0,1) and (0,1,0,0,0) for experiment B. Computational time ranged from 22 to 45 minutes. The number

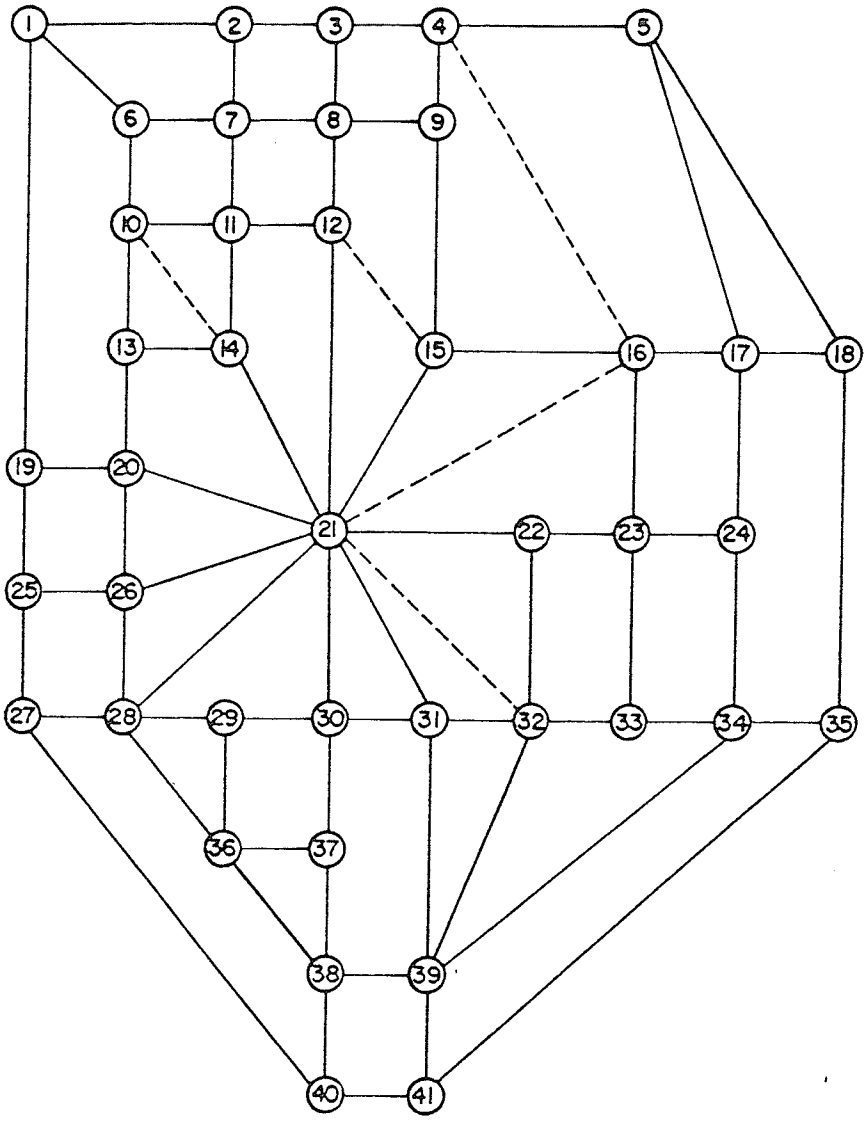

Figure 3.8: Network G_w of Example 3.6

Table 3.13: Data for Network G_w

(i,j)	$a_{i,j}$	$b_{i,j}$	(i,j)	$a_{i,j}$	$b_{i,j}$	(i,j)	$a_{i,j}$	$b_{i,j}$	(i,j)	$a_{i,j}$	$b_{i,j}$
1, 2	5.0	0.162	1, 6	5.0	0.081	1,19	12.0	0.162	2, 3	4.0	0.162
2, 7	8.0	0.065	3, 4	2.0	0.162	3, 8	6.0	0.065	4, 5	3.0	0.162
4, 9	7.0	0.081	5,17	8.0	0.108	5,18	8.0	0.162	6, 7	4.0	0.065
6,10	4.0	0.065	7, 8	5.0	0.054	7,11	4.0	0.054	8, 9	1.0	0.162
8,12	4.0	0.046	9,15	3.0	0.081	10,11	4.0	0.081	10,13	2.0	0.065
11,12	5.0	0.081	11,14	4.0	0.065	12,21	4.0	0.046	13,14	4.0	0.065
13,20	3.0	0.054	14,21	4.0	0.032	15,26	3.0	0.081	15,21	2.0	0.108
16,17	5.0	0.065	16,23	4.0	0.162	17,18	6.0	0.065	17,24	2.0	0.108
18,35	5.0	0.162	19,20	13.0	0.108	19,25	2.0	0.162	20,21	5.0	0.046
20,26	5.0	0.054	21,22	3.0	0.108	21,26	6.0	0.065	21,28	9.0	0.108
21,30	5.0	0.041	21,31	3.0	0.108	22,23	3.0	0.162	22,32	3.0	0.162
23,24	2.0	0.162	23,33	3.0	0.162	24,34	4.0	0.108	25,26	13.0	0.065
25,27	2.0	0.162	26,28	8.0	0.065	27,28	8.0	0.065	27,40	13.0	0.162
28,29	6.0	0.065	28,36	7.0	0.108	29,30	6.0	0.054	29,36	4.0	0.108
30,31	2.0	0.108	30,37	5.0	0.081	31,32	3.0	0.162	31,39	6.0	0.108
32,33	3.0	0.108	32,39	4.0	0.081	33,34	4.0	0.162	34,35	5.0	0.162
34,39	8.0	0.162	35,41	10.0	0.162	36,37	3.0	0.065	36,38	7.0	0.108
37,38	7.0	0.065	38,39	4.0	0.081	38,40	5.0	0.081	39,41	5.0	0.081
40,41	3.0	0.162									

Table 3.14: Demands of 23 O-D Pairs of Network G_w

From \ To	4	7	8	9	10	12	16	17	19	20	21	22	23	25	27	28	29	30	31	32	33	37	38
4	0	4	6	8	3	5	3	9	4	3	19	2	3	4	3	5	6	5	3	4	5	1	7
7	46	0	96	130	58	81	21	160	64	54	339	35	51	64	45	71	74	81	55	62	77	26	58
8	21	32	0	107	58	61	17	67	27	58	151	15	22	27	19	30	32	34	23	28	34	11	43
9	18	59	36	0	22	32	19	90	33	20	129	14	21	33	17	29	29	27	21	26	32	7	44
10	38	118	86	115	0	74	23	112	58	49	315	33	47	58	41	65	69	73	50	56	70	23	105
12	14	24	54	42	19	0	17	53	21	53	148	12	18	21	15	23	24	26	18	20	26	9	38
16	18	34	67	58	27	40	0	75	30	25	159	17	24	30	21	33	35	37	7	31	42	12	54
17	88	74	97	160	60	86	52	0	67	55	353	42	58	75	46	66	71	48	21	70	86	27	84
19	6	10	13	18	8	11	7	26	0	7	34	5	7	18	6	10	10	11	8	9	11	4	16
20	39	66	89	118	53	76	47	114	60	0	368	34	48	60	42	66	70	75	51	62	76	24	107
21	421	716	973	1402	584	814	511	1652	644	429	0	368	529	644	450	580	620	810	549	666	829	260	1164
22	12	21	28	37	18	25	15	46	19	16	111	0	26	19	14	21	630	23	16	19	24	7	33
23	15	24	14	46	20	9	19	58	23	20	125	23	0	23	16	26	27	26	16	25	32	9	42
25	6	10	13	18	8	11	7	26	18	7	34	5	7	0	6	10	10	11	8	9	11	4	16
27	2	4	6	8	4	5	1	8	4	4	23	2	3	4	0	7	5	5	4	4	5	1	7
28	7	11	18	21	10	14	4	21	11	9	65	6	8	11	5	0	21	14	9	8	14	4	19
29	9	12	21	27	12	18	12	35	14	12	130	14	18	14	10	18	0	24	18	22	25	6	26
30	28	46	55	81	37	46	32	104	41	35	203	22	32	41	29	46	40	0	51	43	53	26	74
31	12	20	27	37	16	23	7	35	18	15	97	14	14	18	13	21	22	39	0	19	23	7	32
32	6	10	14	18	8	12	4	18	9	8	44	5	7	9	6	10	11	11	8	0	18	4	16
33	9	16	21	28	13	18	7	26	14	12	106	8	13	19	13	15	16	17	12	22	0	11	25
37	15	26	35	46	21	30	18	58	23	20	124	13	19	23	16	26	27	38	20	24	30	0	42
38	95	163	218	290	131	186	115	281	149	123	779	83	119	146	103	162	172	183	124	151	187	117	0

Table 3.15: Data for Candidate Links in Network G_w

Link Number	(i,j)	$a_{i,j}$	$b_{i,j}$	A	B
				$C_{i,j}$	$C_{i,j}$
1	12,15	3.0	0.008	200×10^3	200×10^3
2	21,32	4.0	0.010	750×10^3	350×10^3
3	10,14	2.0	0.010	450×10^3	250×10^3
4	4,16	5.0	0.006	700×10^3	300×10^3
5	16,21	4.0	0.010	850×10^3	450×10^3

I in Step 1.4 in the assignment part of Algorithm 3.8 was set to 7. The values of θ and ε in the assignment part of the network design algorithm were set to 1.0 and 0.1, respectively, for both Examples 3.5 and 3.6.

Chapter 4

Network Design Problems with Dynamic Traffic Assignment

4.1 Introduction

In Chapter Three, transportation assignment and network design (ND) problems with static traffic demand were discussed. In this chapter, we will discuss transportation assignment and network design problems with dynamic traffic demands. Dynamic demands means that in a transportation network, the number of vehicles for each O-D pair is time varying during the considered time period. A static problem can also be viewed as a dynamic problem having uniformly distributed demands with time variable t. Static problems can be solved by dynamic method. Thus the static traffic assignment and ND problems are the special cases of the more general dynamic problems. The dynamic problem is also more realistic since traffic demands are normally time-varying, especially in urban areas during peak hours. Figure 4.1 compares the differences in static and dynamic traffic demands for a same O-D pair in a network. Most of the available models and algorithms for traffic assignment including those presented in Chapter Three are only for static conditions. Although these static models have very strong mathematical support

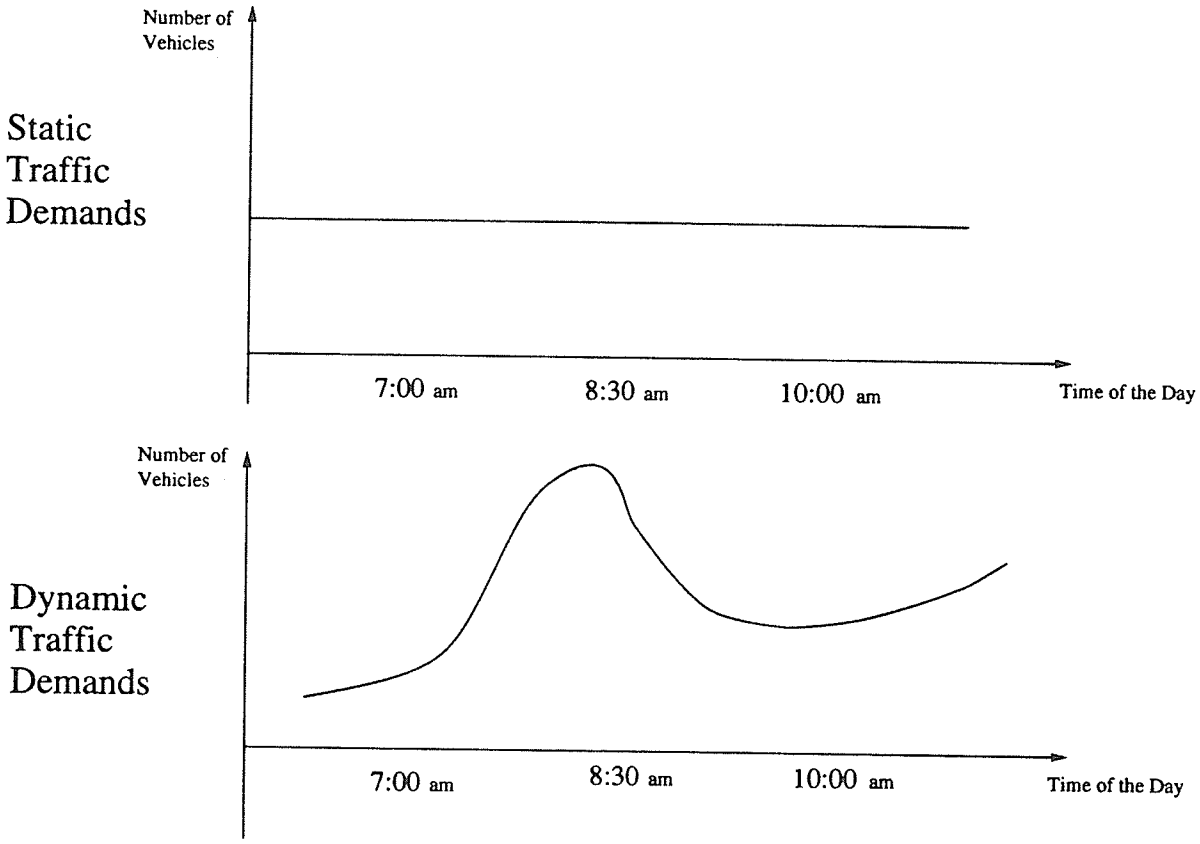

Figure 4.1: Static vs Dynamic Traffic Demands

for their results, as pointed out by Ben-Akiva (1985), they are based on a simplified assumption and some of them failed to capture the essential features of traffic congestion. The lack of realistic assignment algorithms for dynamic traffic demands is mainly due to the difficulties associated with:

- a lack of an appropriate approach to analyze the complicated interfering traffic flows in the dynamic networks,
- developing a suitable mathematical model.

In fact, the complicated interfering traffic flows in a dynamic network result in the difficulty in developing a suitable mathematical model.

In most major urban areas, transportation system planning is often carried out for the peak period during which demands are time-varying. In order to obtain the realistic traffic assignment which is appropriate for peak traffic conditions and is the basis for exploration of dynamic ND problems, the time-varying aspect of travel demand should be taken into account. In this chapter, we will discuss a number of dynamic traffic assignment models and two algorithms for user equilibrium dynamic assignment in a general network with multiple O-D pairs. A branch and bound algorithm for a discrete dynamic ND problem will also be discussed.

4.2 Dynamic Traffic Assignment Problems

Similar to that discussed in Chapter Three, consider a network $G = \{V, L\}$ where V and L are the sets of nodes and links in G, respectively. Let W be the set of origin or destination nodes of the network with $W \subseteq V$. In the static problems discussed in Chapter Three, the performance function of each link in G is assumed to be the BPR curve. In the dynamic problems, the types of "performance function" to be used vary with different models and they will be discussed along with these models.

4.2.1 Dynamic Traffic Assignment Models

There are a number of models dealing with dynamic traffic assignment problems. Some of these models are developed for the situations of varying trip departure times in a single link or two link networks. A literature review of such dynamic traffic assignment problems can be found in Alfa (1986). In this subsection, we consider a traffic assignment problem in general networks with time varying demands, but assume that the temporal distribution of the travellers remains the same. Four models or algorithms for solving dynamic assignment problems in a general transportation

network will be discussed.

System Optimal Model with Discrete Time (Model 4.1)

Merchant and Nemhauser (1978) developed a system optimal model for solving the dynamic traffic assignment problems with discrete time variable. The model aimes at an assignment problem with multiple origins and a single destination in a general network. It is assumed that the total time span is N which is partitioned into equally spaced discrete time epochs denoted by n, $n = 1, ..., N$. The model is as follows:

$$\min \sum_{n=1}^{N} \sum_{ij} t_{ij}(n)[x_{ij}(n)] \tag{4.1}$$

subject to:

$$x_{ij}(n+1) = x_{ij}(n) - s_{ij}[x_{ij}(n)] + y_{ij}(n), n = 0, 1, ..., N-1, \forall ij \in L \tag{4.2}$$

$$\sum_{j \in O_r} y_{ij}(n) = d_r(n) + \sum_{r} s_{ir}[x_{ir}(n)], n = 1, ..., N-1, \forall r \tag{4.3}$$

$$x_{ij}(0) = R_{ij} \geq 0, \forall ij \tag{4.4}$$

$$y_{ij}(n) \geq 0, n = 0, ..., N-1, \forall ij, \tag{4.5}$$

$$x_{ij}(n) \geq 0, n = 1, ..., N, \forall ij. \tag{4.6}$$

In the above model, $x_{ij}(n)$ is traffic flow on link ij at time epoch n; travel time $t_{ij}(n)$ (the performance function) on link ij at time epoch n is a non-negative and non-decreasing function of flow $x_{ij}(n)$; s_{ij} is the number of vehicles that can exit from link ij during a time epoch; $d_r(n)$ is the traffic demand at time epoch n from origin r to the single destination; O_r is the set of nodes connected with and directed from origin r; and R_{ij} are the variables for initialization. Decision variables of this model are $y_{ij}(n)$ in Eq(4.2).

Merchant and Nemhauser developed a piecewise linear programming method to solve this model. The model can be extended to solve a multiple origin and multiple destination problem by creating a super single destination and by connecting the multiple destinations with the created single destination. Since this model is based on the system optimal principle, it will not be appropriate to use this model to solve a dynamic assignment problem in an urban transportation network. User equilibrium models will be discussed next.

User Equilibrium Model with Continuous Time (Model 4.2)

Friesz et al. (1989) proposed a user equilibrium model with continuous time variable to solve dynamic traffic assignment problems. It is assumeed that the total time span is $\mathcal{T}$ and τ is a continuous time variable with $0 \leq \tau \leq \mathcal{T}$. The model can be presented as follows:

$$\min \sum_{ij} \int_0^{\mathcal{T}} \int_0^{x_{ij}(\tau)} t_{ij}(\omega) g'_{ij}(\omega) d\omega d\tau \tag{4.7}$$

subject to:

$$\frac{dx_{ij}(\tau)}{d\tau} = d_{ij}(\tau) - g_{ij}[x_{ij}(\tau)] \tag{4.8}$$

$$S_k(\tau) = \sum_k d_{ik}(\tau) - \sum_k g_{kj}[x_{kj}(\tau)] \tag{4.9}$$

$$x_{ij}(0) \geq 0, \forall ij, \tag{4.10}$$

$$d_{ij}(\tau) \geq 0, x_{ij}(\tau) \geq 0, \forall ij. \tag{4.11}$$

In Eqs(4.7) to (4.11), t_{ij} is cost function on link ij, d_{ij} and g_{ij} are variables of incoming and outgoing flows, respectively, on link ij. $S_k(\tau)$ is the traffic flow generated at node k which is a nonnegative and continuous function of time τ, $0 \leq \tau \leq \mathcal{T}$. The above formulation of the dynamic user equilibrium (UE) model is a reasonable

generalization of the static UE model of Eqs(3.7) and (3.8) (Chapter Three). Theoretical analysis has also been provided for this model by Friesz et al.(1989). A major feature of this model is that it is based on the assumption that each traveller in the network will re-calculate his/her minimum travel time path and shift to a new shortest path for the rest of his/her trip at each node as he/she arrives at the nodes. This assumption requires that any algorithm developed to solve this model would calculate the shortest paths at every time point whenever a vehicle arrives at an intersection in the network. This model will be very useful for real time analysis, but it could be very difficult to develop an algorithm based on this model. At present time, there is no algorithm available that can solve this formulation.

TRRL/CONTRAM (Model 4.3)

CONTRAM is a well-developed computer software. It is able to solve dynamic traffic assignment problems without developing an optimization model. CONTRAM is developed by Transport and Road Research Laboratory (TRRL) of Britain and has been widely used, mainly in European countries, for a number of years (Leonard et al., 1989). CONTRAM is a powerful tool for detailed traffic management. An iterative algorithm is included in CONTRAM for dynamic traffic assignment. Some of the main features of CONTRAM are:

- CONTRAM performs the assignment by the portions (packets) of vehicles. The size of a packet is fixed in each of its iterations.
- As discussed in Alfa (1987), CONTRAM assumes, for the second and subsequent iterations, that the congestion delay to a packet is a result of other vehicles and is independent of that packet. CONTRAM therefore subtracts, the volume of this packet, from the volume estimates used for estimating travel

times experienced by a packet on a link. It then calculates the minimum travel time path for that packet and allocates that packet accordingly.

- Time epochs used in CONTRAM are not necessarily equal. Time epochs become smaller when traffic demands vary sharply. This feature allows CONTRAM to be more accurate in solving dynamic traffic assignment problems.
- CONTRAM has many well-developed functions for detailed traffic management. It is a very useful computer package for solving realistic traffic assignment problems.
- The assignment algorithm used by CONTRAM may need further improvement.

Other models have been proposed for solving dynamic traffic assignment problems in general networks, but the critical feature of the problem, that is, the interfering of traffic flows has not properly been addressed.

User Equilibrium Model with Discrete Time (Model 4.4)

Alfa (1987) proposed a discrete time model for solving dynamic UE assignment problems in a general network with multiple O-D pairs. The problem of interfering traffic flows has been taken into account by this model. A major development in this thesis research for dynamic assignment is based on this model. Assume that in a network G, associated with each link ij is the capacity $c_{ij} > 0$ and the zero-flow travel time $a_{ij} > 0, ij \in L$. Similar to that in Model 4.1, it is assumed that the total time span is N which is partitioned into equally spaced discrete time epochs denoted by n, $n = 1, ..., N$. In this model, n starts ($n = 1$) from the epoch that the first vehicle enters the network and ends ($n = N$) at the epoch that the last

vehicle leaves the network. The number of vehicles departing at time epoch n and travelling between O-D pair rs is expressed by $d_{rs}(n'), n' = 1, ..., N_d, 1 \leq N_d < N$. For this assignment model we assume that:

- the route selection approach is deterministic, that is, travellers at each time epoch n have perfect knowledge of travel time in the system and will choose the minimum travel time path (shortest path) among all the possible paths from their origins r to destinations s; $r, s \in W$,
- there is no due time for any travellers to arrive at their destinations, and
- the temporal distribution of demand remains the same, that is, no traveller will change his/her departing time epoch even it can reduce his/her travel time.

The deterministic route selection mode is considered unrealistic as discussed in Chapter Three. Because the problem will become too complicated if the stochastic route selection mode is used in dynamic assignment algorithms, we only consider the deterministic mode in traffic assignment and ND models in this chapter.

The no due time requirement and constant temporal distribution of demands are the same features that Models 4.1, 4.2 and 4.3 have. Some of the dynamic assignment models based on single or two link networks address the problem with due time requirement and changing temporal distribution of traffic demands. Including these features makes the model more realistic but those models are normally not applicable to the problems in a general network with multiple O-D pairs. Because of the excessive complexity of the models and algorithms, the due time requirement and the changing distribution of demands are not considered in solving dynamic assignment and ND problems in this chapter.

Figure 4.2: Bottleneck at the Exiting End of a Link

For this dynamic UE model with discrete time variable, we assume that the bottleneck causing the traffic delay is located at the exiting end of each link ij, $\forall ij \in L$. This is illustrated in Figure 4.2.

For each link ij and time epoch n, define $x_{ij}(n)$ as the flow on the link, $s_{ij}(n)$ as the number of vehicles exiting from the bottleneck, and $t_{ij}(n)$ as the travel time on the link at time epoch $n, n = 1, ..., N$. Also define that O_i is the set of links connected with and directed to node i. Let $[a_{ij}]$ represent the integer greater than or equal to a_{ij}, that is, $a_{ij} \leq [a_{ij}] < a_{ij} + 1$. Then $x_{ij}(n)$ can be evaluated by the following difference equations:

$$\begin{aligned} x_{ij}(n) = \; & x_{ij}(n-1) + \textstyle\sum_{rs} \delta_{ij}^{rs}(n', n) d_{rs}(n) \\ & + [\textstyle\sum_{hi \in O_i} \sum_{rs} \delta_{ij}^{rs}(n', n) s_{hi}(n)] - s_{ij}(n); \qquad (4.12) \\ & n' = 1, ..., N_d, n = 2, ..., N; \end{aligned}$$

where:

$$\begin{cases} x_{ij}(1) = & \sum_{rs} \delta_{ij}^{rs}(1,1) d_{rs}(1); \\ s_{ij}(n) = & 0, \quad n = 1, .., [a_{ij}]; \\ s_{ij}(n) = & \min\{\sum_{rs} \delta_{ij}^{rs}(1,1) d_{rs}(1), c_{ij}\}, \quad n = [a_{ij}] + 1; \\ s_{ij}(n) = & \min\{\sum_{m=1}^{n-[a_{ij}]} \sum_{rs} \delta_{ij}^{rs}(n',m) d_{rs}(m) + \\ & \sum_{m=2}^{n-[a_{ij}]} \sum_{hi \in O_i} \sum_{rs} \delta_{ij}^{rs}(n',m) s_{hi}(m), c_{ij}\}, \\ & n' = 1, ..., N_d, n = [a_{ij}] + 2, ..., N, \end{cases} \tag{4.13}$$

and $\delta_{ij}^{rs}(n', n)$ is defined as:

$$\delta_{ij}^{rs}(n', n) = \begin{cases} & \text{if any of the vehicles between } rs, \text{ departing} \\ 1, & \text{at } n', \text{ can arrive at node } i \text{ and} \\ & \text{will pass link } ij \text{ at time epoch } n, \\ 0 & \text{otherwise.} \end{cases}$$

Travel time $t_{ij}(n)$ on link ij at time epoch n is evaluated by:

$$t_{ij}(n) = x_{ij}(n)/c_{ij} + a_{ij}, \quad ij \in L, n = 1, ..., N, \tag{4.14}$$

It is obvious that for a same set of demands d_{rs}, values of the traffic volumes $x_{ij}(n)$ depend upon the routes selected by the travellers from their origins r to destinations s, $\forall r, s \in \mathbf{V}$. Different route selections will result in different values of $\delta_{ij}^{rs}(n', n)$ in Eqs(4.12) and (4.13). The shortest path with time-varying travel times is obtained by the method given in Cooke and Halsey (1966). The details of the shortest path algorithm is as follows:

Let $t_{rs}^{*}(n)$ be the travel time along the minimum time path from node r to node s if the trip starts at node r at time n, then:

$$t_{rs}^{*}(n') = \min_{\forall z \neq r}\{t_{rz}(n') + t_{zs}^{*}(n' + t_{rz}(n'))\}, \quad rs \in V, \tag{4.15}$$

where:

$$t^{*}_{rr}(n') = 0, \forall r, n' = 1, ..., N_d. \tag{4.16}$$

Eqs(4.15) and (4.16) follow those in Alfa (1987).

The traffic assignment mechanism in this dynamic model is similar to that in the static DUE assignment method discussed in Section 3.2.3. In this model, we assume that travellers will select the shortest path obtained by the dynamic shortest path algorithm presented in Eqs(4.15) and (4.16). When the roads become congested, some of the travellers will shift to a new shortest path. After large number of iterations of traffic loading and reloading, user equilibrium travel pattern (if it exists) is expected to be achieved and no traveller can reduce his travel time by unilaterally shifting his/her route. It should be noted that there is no specific mathematical formulation for this dynamic traffic assignment model. It can be expressed as above stated, or by conceptual symbols as presented in Alfa (1987).

4.2.2 Dynamic Traffic Assignment Algorithms

In this subsection two algorithms, Algorithms 4.1 and 4.2, will be presented that can be used to solve Model 4.4. Theoretical analysis based on a simple network for Algorithm 4.2 will be given after the algorithms are presented.

Alfa (1987) developed a heuristic algorithm to solve Model 4.4. The algorithm is rewritten and presented as Algorithm 4.1. A major result of this research presented in this chapter is the development of Algorithm 4.2. Both of Algorithms 4.1 and 4.2 are iterative. Algorithms 4.2 incorporates the method of successive averages (MSA) into Algorithm 4.1. The traffic loading and reloading mechanism of Algorithm 4.1 is improved by Algorithm 4.2. Although technical improvement has been made, Algorithm 4.2 is still a heuristic algorithm when it is used for solving dynamic

traffic assignment problems in a general network. Convergence to the dynamic user equilibrium (if it exists) by any of the two algorithms has not been mathematically proven.

In Algorithm 4.1 presented below, traffic demand of each O-D pair at each time epoch will be repeatedly loaded on the corresponding shortest paths obtained by Eqs(4.15) and (4.16). Travel time calculated in the current iteration is used to obtain the shortest paths for traffic loading in the next iteration. The step by step procedure of this algorithm is presented below:

Algorithm 4.1

Step 0.0: Let $k = 0$.

Step 0.1: Assume no flows in the network, i.e. $x_{ij}^{(0)}(n) = 0$, $n = 1, ..., N, \forall ij$.

Step 0.2: Find the shortest paths $\rho_{rs}^{(0)}(n')$ using travel times

$t_{ij}(n) = a_{ij}, n' = 1, ..., N_d, n = 1, ..., N; \forall rs$.

Step 0.3: Assign $d_{rs}(n')$ to path $\rho_{rs}^{(0)}(n')$, $n' = 1, ..., N_d; \forall rs$.

Step 0.4: Calculate $x_{ij}^{k}(n)$ using Eqs(4.12) and (4.13), $n = 1, ..., N, \forall ij$.

Step 0.5: Using $x_{ij}^{(0)}(n)$ as the current flows in the network, calculate travel times $t_{ij}^{(0)}(n)$ by Eq(4.14), $n = 1, ..., N; \forall ij$.

Step 0.6: Goto Step 1.0.

Step 1.0: Let $k = 1$.

Step 1.1: Assume no flows in the network, i.e. $x_{ij}^{(k)}(n) = 0$, $n = 1, ..., N; \forall ij$.

Step 1.2: Find the shortest paths $\rho_{rs}^{(k)}(n')$ using the latest travel times

$t_{ij}^{(k-1)}(n), n' = 1, ..., N_d, n = 1, ..., N; \forall rs$.

Step 1.3: Assign $d_{rs}(n')$ to path $\rho_{rs}^{(k)}(n')$, $n' = 1, ...N_d; \forall rs$.

Step 1.4: Calculate $x_{ij}^{k}(n)$ using Eqs(4.12) and (4.13), $n = 1, ..., N, \forall ij$.

Step 1.5: Using $x_{ij}^{(k)}(n)$ as the current flows in the network, calculate travel times $t_{ij}^{(0)}(n)$ by Eq(4.14), $n = 1, ..., N; \forall ij$.

Step 1.6: If $|x_{ij}^{(k)}(n) - x_{ij}^{(k-1)}(n)| \leq \epsilon; n = 1, ..., N, \forall ij$, or $k \geq K$, stop; otherwise, let $k = k + 1$, goto Step 1.1.

In Step 1.6, ϵ is a given small real number and K is a given large integer for the stopping criterion, that is, the algorithm will terminate if the difference between the traffic volumes of any two consecutive iterations, for each link and each time epoch n, is smaller than a given small real number, or the iteration number is greater than a given large integer.

Algorithm 4.1 has a simple traffic loading and reloading mechanism. As reported in Alfa (1987), traffic flows generated by this algorithm fluctuate in some case. In Alfa (1987), an incremental algorithm was also presented. Since the incremental algorithm needs much more computational effort, we did not make the effort to improve that algorithm in this research.

To improve the traffic loading and reloading mechanism of Algorithm 4.1, we incorporated MSA into this algorithm and developed Algorithm 4.2 to solve the dynamic assignment problem. The MSA approach is the same as in the static assignment algorithm (Algorithm 3.4). One technical difference is that, for the dynamic problem, MSA uses the dynamic shortest path algorithm (Eqs(4.15) and (4.16)) and generates the shortest path for each O-D pair at each time epoch in an iteration. Another difference is that the traffic loading and reloading is performed based on the traffic routes in the dynamic model, instead of on links as in the static case. The stopping criterion of this algorithm is same as in Algorithm 4.1.

Dynamic Method of Successive Averages (MSA)

Solutions of dynamic traffic assignment problems are based on the assumption that there exists a "dynamic user equilibrium" traffic pattern such that no traveller, departing at n, $n = 1, ..., N_d$, can reduce his/her travel time by unilaterally changing his/her route. Travel time by the travellers might be reduced if some of them change their departing times unilaterally. In the models in this chapter, however, we assume that the temporal distribution of traffic demands is fixed. To the knowledge of the author, there are no appropriate mathematical formulations available for solving this assignment problem. Such mathematical formulations will not be developed and in this thessi research either. The major difficulty is in finding the objective function of the model if it can be formulated into an optimization model. To sum the objective functions of the static user equilibrium model by all the time epochs cannot yield a proper formulation for solving the above dynamic user equilibrium assignment problem.

The main reason that the dynamic MSA is used for solving this problem is that the evaluation of the objective function is not needed.

The other reasons why the MSA method is used are:

- The convergence to the user equilibrium of MSA has been proven (Powell and Sheffi (1982)) for static assignment problems in general networks under normal conditions. Since the application of MSA in the dynamic assignment is a generalization of the static one, the convergence behavior in the static case is a good basis that encouraged the author to use the same approach.

- The convergence of MSA in solving dynamic assignment problems can be mathematically verified in a simplified case, as presented in Section 4.2.3.

- The procedure of MSA is consistent with normally assumed behavior of the travellers. Because it is an iterative procedure, the congestion occured in a link at a time epoch in the previous iteration can be avoided by the traffic loadings of the later iterations. All the relevant information obtained in the earlier iterations can be taken into account by the later iterations of the traffic loadings.

- MSA is also a simple method. It requires one execution of the shortest path algorithm (Eqs(4.15) and (4.16)) in each iteration of the assignment procedure. Hence the computational time is usually at an acceptable level.

In using MSA for dynamic assignment, one needs to store the information identifying all the shortest paths obtained during all the iterations. The storage requires extra computer memory more than that required by the static assignment algorithms or the dynamic Algorithm 4.1. For those algorithms, such information does not need to be stored. The step by step procedure of Algorithm 4.2 is presented below.

Algorithm 4.2

Step 0.0: Let $k = 0$.

Step 0.1: Assume no flows in the network, i.e. $x_{ij}^{(0)}(n) = 0$, $n = 1, ..., N, \forall ij$.

Step 0.2: Find the shortest paths $\rho_{rs}^{(0)}(n')$ using travel times

$$t_{ij}(n) = a_{ij}, n' = 1, ..., N_d,\ n = 1, ..., N; \forall rs.$$

Step 0.3: Assign $d_{rs}(n')$ to path $\rho_{rs}^{(0)}(n')$, $n' = 1, ..., N_d; \forall rs$.

Step 0.4: Calculate $x_{ij}^{k}(n)$ using Eqs(4.12) and (4.13), $n = 1, ..., N, \forall ij$.

Step 0.5: Using $x_{ij}^{(k)}(n)$ as the current flows in the network, calculate

times $t_{ij}^{(0)}(n)$ by Eq(4.14), $n = 1, ..., N; \forall ij$.

Step 0.6: Goto Step 1.0.

Step 1.0: Let $k = 1$.

Step 1.1: Assume no flows in the network, i.e. $x_{ij}^{(k)}(n) = 0$, $n = 1, ..., N; \forall ij$.

Step 1.2: Find the shortest paths $\rho_{rs}^{(k)}(n')$ using the latest travel times $t_{ij}^{(k-1)}(n), n' = 1, ..., N_d$, $n = 1, ..., N; \forall rs$.

Step 1.3: Let $\bar{d}_{rs}(n') = d_{rs}(n')/k, n' = 1, ..., N_d; \forall rs$.

Assign $\bar{d}_{rs}(n')$ to path $\rho_{rs}^{(k')}(n')$, $n' = 1, ...N_d; \forall rs; k' = 1, ..., k$.

Step 1.4: Replace $d_{rs}(n')$ by $\bar{d}_{rs}^{k'}(n')$ and replace $\delta_{ij}^{rs}(n', n)$ by $\delta_{ij}^{rs(k')}(n', n)$ according to $\rho_{rs}^{k'}(n')$, in Eqs(4.12) and (4.13), $k' = 1, ..., k, n' = 1, ..., N_d, \forall rs$.

Calculate $\bar{x}_{ij}^{k'}(n)$ using Eqs.(4.12) and (4.13), $n' = 1, ..., N_d, n = 1, ..., N$, $\forall ij$. Let $x_{ij}^{(k)}(n) = \sum_{k'=1}^{k} \bar{x}_{ij}^{k'}(n)$.

Step 1.5: Using $x_{ij}^{(k)}(n)$ as the current flows in the network, calculate travel times $t_{ij}^{(0)}(n)$ by Eq(4.14), $n = 1, ..., N; \forall ij$.

Step 1.6: If $|x_{ij}^{(k)}(n) - x_{ij}^{(k-1)}(n)| \leq \epsilon; n = 1, ..., N, \forall ij$, or $k \geq K$, stop; otherwise, let $k = k + 1$, goto Step 1.1.

This algorithm can be used for solving dynamic assignment problems in general networks with multiple O-D pairs. A theoretical analysis for the algorithm based on a simple two link network case with one O-D pair is presented next.

4.2.3 Theoretical Analysis of Dynamic Method of Successive Averages (MSA) in a Two Link Network

Horowitz (1984) developed an approach for analyzing the convergence behavior of a number of static assignment methods using a simple two link network. Here we use the same analysis approach for the dynamic assignment Algorithm 4.2. Consider a

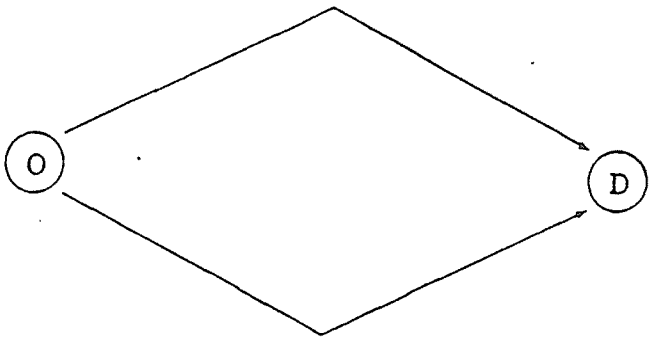

Figure 4.3: A Two Link Network

two link network with one origin destination pair O-D as shown in Figure 4.3. Since the two links in Figure 4.3 have the same pair of identifying nodes, we temporarily eliminate the double subscripts ij of all the variables. Instead, we use a single subscript $i, i = 1, 2$, to identify the two links. For example $x_i(n)$ refers to the flow on link i at time n. Since we have only two links in this section, then by recursively solving the simplified Eq(4.12), it can be obtained that:

$$x_i(n) = \sum_{m=1}^{n} d_i(m) - \sum_{m=2}^{n} s_i(m), \; i = 1, 2; \; n = 1, ..., N_d. \tag{4.17}$$

Let $J_i = 1/c_i$ and then from Eqs(4.14), and (4.17) we have:

$$t_i(n) = a_i + J_i \sum_{m=1}^{n} d_i(m) - J_i \sum_{m=2}^{n} s_i(m) = A_i(n) + J_i \sum_{m=1}^{n} d_i(m), \tag{4.18}$$

or expressed as:

$$t_i(n) = f_i^n(d_i(1), d_i(2), ..., d_i(n)), \tag{4.19}$$

where $f_i^n(d_i(1), d_i(2), ..., d_i(n))$ are non-decreasing for all of their arguments, $i = 1, 2; \; n = 1, ..., N_d$.

Following Horowitz (1984), let $P_1(t_1(n), t_2(n))$ denote the probability that a traveller from origin O to destination D at time epoch n chooses link 1 if the travel time on link 1 is less than the travel time on link 2. For a two link situation, we have:

$$P_2(t_1(n), t_2(n)) = 1 - P_1(t_1(n), t_2(n)).$$

And the following equation holds:

$$d_i(n) = P_i(t_1(n), t_2(n)),\ i = 1, 2;\ n = 1, ..., N_d. \tag{4.20}$$

We assume, for the analysis, without loss of generality, that $d_1(n) + d_2(n) = 1, n = 1, ..., N_d$. We can also write the probability $P_1(t_1(n), t_2(n))$ in the function form as (c.f. Horowitz):

$$P_1(t_1(n), t_2(n)) = F(t_2(n) - t_1(n)) = F(\Delta t(n)) \tag{4.21}$$

and

$$P_2(t_1(n), t_2(n)) = 1 - F(\Delta t(n)),\ n = 1, ..., N_d. \tag{4.22}$$

For the deterministic link selection described in the MSA algorithm, $F(\Delta t(n))$ is:

$$F(\Delta t(n)) = \begin{cases} 1, & \text{if } \Delta t(n) > 0, \\ 0.5, & \text{if } \Delta t(n) = 0, \\ 0 & \text{if } \Delta t(n) < 0. \end{cases} \tag{4.23}$$

Note that $F(\Delta t(n))$ is discontinuous at $\Delta t(n) = 0$, $n = 1, ..., N_d$.

The assignment procedure of the MSA for each iteration k, $k \to \infty$, can be expressed by:

$$\bar{d}_i^{(k)}(m) = \frac{1}{k} d_i^{(k)}(m) + (1 - \frac{1}{k}) \bar{d}_i^{(k-1)}(m),\ i = 1, 2;\ m = 1, ..., N_d, \tag{4.24}$$

and the travel time is calculated by:

$$\bar{t}_i^{(k)}(n) = A_i(n) + J_i \sum_{m=1}^{n} \bar{d}_i^{(k)}(m),\ i = 1, 2;\ n = 1, ..., N_d. \tag{4.25}$$

The purpose of the following analysis is to show that:

$$\lim_{k\to\infty} \Delta\bar{t}^{(k)}(n) = \lim_{k\to\infty} (\bar{t}_2^{(k)}(n) - \bar{t}_1^{(k)}(n)) = 0,\ n = 1, ..., N_d.$$

From Eqs(4.24) and (4.25), we have:

$$\bar{t}_i^{(k)}(n) = A_i(n) + J_i \sum_{m=1}^{n} [\frac{1}{k} d_i^{(k)}(m) + (1 - \frac{1}{k})\bar{d}_i^{(k-1)}(m)],\ i = 1, 2; \tag{4.26}$$

and from Eq(4.26), we have:

$$\Delta\bar{t}^{(k)}(n) = \Delta\bar{t}^{(k-1)}(n) + \frac{1}{k}\{t_2^{(k)}(n) - t_1^{(k)}(n) - \Delta\bar{t}^{(k-1)}(n)\},\ n = 1, ..., N_d;\ k = 1, 2, ..., \tag{4.27}$$

since $\Delta\bar{t}^{(k)}(n) = \bar{t}_2^{(k)}(n) - \bar{t}_1^{(k)}(n)$. Note that Eq(4.27) holds because of the linearity between the travel time and the travel volume as shown in Eq(4.14).

Since the arrival rate on link 1 at time epoch n for iteration k is determined by the travel time difference in the two links of iteration $k-1$, then from Eqs(4.20) and (4.21), we have:

$$d_1^{(k)}(n) = P_1(t_1^{(k-1)}(n), t_2^{(k-1)}(n)) = F(\Delta t^{(k-1)}(n)). \tag{4.28}$$

From Eqs(4.19), (4.27) and (4.28) we have:

$$\begin{aligned} \Delta\bar{t}^{(k)}(n) \quad = \quad & \Delta\bar{t}^{(k-1)}(n) \\ & + \tfrac{1}{k}\{f_2^n[d_2^{(k)}(1), ..., d_2^{(k)}(n-1), (1 - F(\Delta\bar{t}^{(k-1)}(n)))] \\ & - f_1^n[d_1^{(k)}(1), ..., d_1^{(k)}(n-1), F(\Delta\bar{t}^{(k-1)}(n))] \\ & - \Delta\bar{t}^{(k-1)}(n)\}. \end{aligned} \tag{4.29}$$

Let:

$$H_n(\mathbf{Y}, x) = f_2^n[\mathbf{Y}, (1 - F(x))] - f_1^n[\mathbf{Y}, F(x)], \tag{4.30}$$

where $\mathbf{Y}$ is a vector of $n-1$ dimensions, $n = 1, ..., N_d$, and H_n is a non-increasing function. For each n and $\Delta t(n) \neq 0$, there is always an $M_n > 0$ such that:

$$|H_n(\mathbf{Y}, \Delta t(n))| \leq M_n |\Delta t(n)|, \tag{4.31}$$

From Eqs(4.29) and (4.30) we have:

$$\Delta\bar{t}^{(k)}(n) = \Delta\bar{t}^{(k-1)}(n) + \frac{1}{k}\{[H_n(\mathbf{Y}, \Delta\bar{t}^{(k-1)}(n))] - \Delta\bar{t}^{(k-1)}(n)\}. \tag{4.32}$$

For each $k > 2$ and each $n, n = 1, ..., N_d$, define $z_k(n)$ by

$$z_k(n) = \begin{cases} -H_n(\mathbf{Y}, \Delta\bar{t}^{(k-1)}(n))/\Delta\bar{t}^{(k-1)}(n), & \text{if } \Delta\bar{t}^{(k-1)}(n) \neq 0; \\ 0, & \text{if } \Delta\bar{t}^{(k-1)}(n) = 0. \end{cases} \tag{4.33}$$

From Eqs(4.32) and (4.33), we have:

$$\Delta\bar{t}^{(k)}(n) = [1 - \frac{1}{k}(z_k(n) + 1)]\Delta t^{(k-1)}(n). \tag{4.34}$$

By recursively solving Eq(4.34), it can be obtained that:

$$\Delta\bar{t}^{(k)}(n) = \prod_{l=2}^{k}[1 - \frac{1}{l}(z_l(n) + 1)](\Delta t^{(1)}(n)). \tag{4.35}$$

Since $k \rightarrow \infty$, then for l with $l > \frac{M+1}{2}$, we have $\frac{1}{l} < \frac{2}{M+1}$. From Eqs(4.31) and (4.33), it can be seen that $0 \leq z_k(n) \leq M_n$ for all $k \geq 2$. So there is:

$$|1 - \frac{1}{l}(z_l(n) + 1)| \begin{cases} < 1 & \text{if } l \neq \infty, \\ = 1 & \text{if } l = \infty. \end{cases} \tag{4.36}$$

Because

$$\lim_{k\rightarrow\infty} \frac{1}{k} = 0, \tag{4.37}$$

then for any $\epsilon > 0$, there is always an l^* such that $\forall l, l > l^*$, it has $\frac{1}{l} < \epsilon$ and $0 < 1 - \frac{1}{l}(z_l(n) + 1) \leq 1$. Since

$$\lim_{l\rightarrow\infty}(1 - \frac{1}{l}(z_l(n) + 1)) = 1,$$

then, without loss of generality, let l^* be the one that for $l > l^*, l \neq \infty$, it also has:

$$\log|1 - \frac{1}{l}(z_l(n) + 1)| \leq -\epsilon < -\frac{1}{l}. \tag{4.38}$$

Then from Eq(4.35):

$$\log|\Delta\bar{t}^{(k)}(n)| = \sum_{l=2}^{l^*}\log|1-\frac{1}{l}(z_l(n)+1)| + \sum_{l=l^*+1}^{k}\log|1-\frac{1}{l}(z_l(n)+1)| + \log|\Delta t^{(1)}(n)|. \tag{4.39}$$

From Eqs(4.38) and (4.39), we have:

$$\log|\Delta\bar{t}^{(k)}(n)| \leq \sum_{l=2}^{l^*}\log|1-\frac{1}{l}(z_l(n)+1)| - \sum_{l=l^*+1}^{k}\frac{1}{l} + \log|\Delta t^{(1)}(n)|. \tag{4.40}$$

Since:

$$\sum_{k=1}^{\infty}\frac{1}{k} = \infty, \tag{4.41}$$

then

$$\lim_{k\to\infty}\log|\Delta\bar{t}^{(k)}(n)| = -\infty. \tag{4.42}$$

It means that

$$\lim_{k\to\infty}\Delta\bar{t}^{(k)}(n) = 0. \tag{4.43}$$

Remarks.

i) The analysis presented above is a dynamic extension of the static assignment presented by Horowitz. The only requirement for the performance functions in Horowitz's approach is that they should be non-decreasing in terms of traffic volumes on the links. This requirement is satisfied by this dynamic case as shown in Eq(4.14).

ii) It should be pointed out that the counter example given in Horowitz (1984) does not apply to the MSA algorithm. This is because in our algorithm,

travel times $t_{ij}^k(n)$ at each iteration k are generated by the successive averages of the demands, not by the travel times of the previous iterations.

iii) Algorithm 4.2 can be developed based on any other types of performance functions. The linearity of the performance function in Eq(4.14) is not necessary for MSA to generate dynamic traffic assignment. For example, congestion delay can be calculated by queueing models. In that case, however, the analysis approach presented above will not be valid, because the linear performance function is the key factor in obtaining Eq(4.27) from Eq(4.26).

iv) It would be very difficult to extend the same analysis approach to the multiple O-D pair assignment problem in a general network, although it is a more realistic situation. Even for a simple network as shown in Figure 4.4, with two O-D pairs (from Node 1 to Node 4, and from Node 3 to Node 4), the above analysis approach will not be applicable. One of the reasons is that, for example, Eq(4.17) is not valid for link 2,4.

A major benefit of Algorithm 4.2 is in applying it to the dynamic network design (ND) problems. Most ND models are for static travel demands whereas dynamic models are more appropriate in urban areas. Traffic assignment needs to be performed repeatedly in almost all network design models based on the user equilibrium principle. Since Algorithm 4.2 does not require much computational time, it will be a good candidate algorithm to be incorporated into a dynamic ND algorithm. Such algorithm will be discussed in the next section.

A very important and still unanswered question of the dynamic assignment problem is whether the existence of a dynamic user equilibrium for a general network can be proved. This can only be answered if the dynamic assignment problem can be

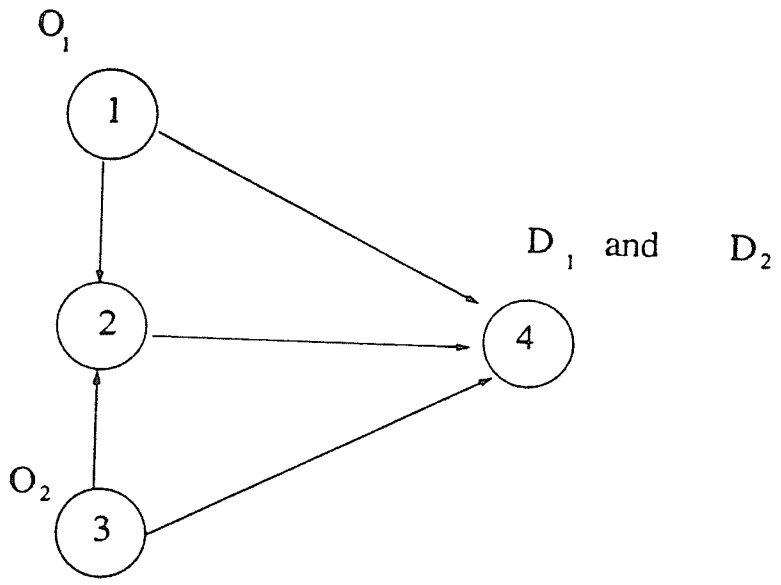

Figure 4.4: A Network with Multiple O-D Pairs

formally modeled mathematically. Even if the dynamic equilibrium can be proved to exist, it is still difficult to show that Algorithm 4.2 does converge in a general network.

Next we will present an example to illustrate Algorithm 4.2. The convergence behavior of this algorithm can be seen from this dynamic assignment example for a small size general network with multiple O-D pairs.

4.2.4 A Dynamic Assignment Example

In this section we will present a dynamic assignment example to illustrate Algorithm 4.2

Example 4.1

The Network and Input Data

Consider a traffic network G as shown in Figure 4.5. Each link ij in G represents a one-way road from i to j with zero-flow travel time a_{ij} and link capacity c_{ij}, as

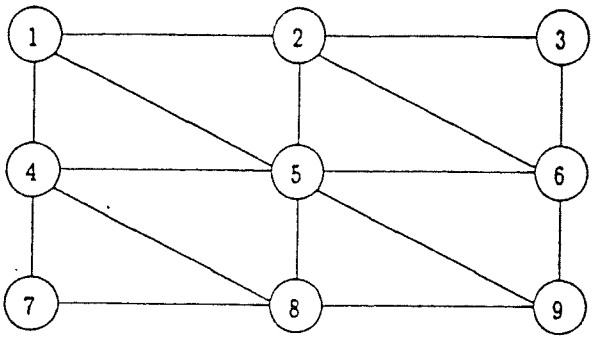

Figure 4.5: Network G of Example 4.1

Table 4.1: Link Parameters of Network G

(i,j)	a_{ij}	c_{ij}	(i,j)	a_{ij}	c_{ij}
(1, 2)	2.0	3.0	(1, 4)	1.0	3.0
(1, 5)	3.0	3.0	(2, 3)	3.0	2.0
(2, 5)	4.0	3.0	(2, 6)	2.0	2.0
(3, 6)	1.0	3.0	(4, 5)	2.0	4.0
(4, 7)	3.0	2.0	(4, 8)	1.0	2.0
(5, 6)	1.0	3.0	(5, 8)	3.0	4.0
(5, 9)	1.0	4.0	(6, 9)	4.0	3.0
(7, 8)	1.0	2.0	(8, 9)	2.0	3.0

shown in Table 4.1. Traffic demands of the three O-D pairs for seven time epochs are given in Table 4.2. The total iteration number K (Step 1.6 in Algorithm 4.2) was set to 16 for this example.

The Computational Results

The total travel cost (time), $T^{(k)}$, that the travellers spent in the network for each iteration k is shown in Figure 4.6. It is evaluated as:

$$T^{(k)} = \sum_{ij \in L} \sum_{n=1}^{N} t_{ij}^{(k)}(n), \quad k = 1, ..., 16. \tag{4.44}$$

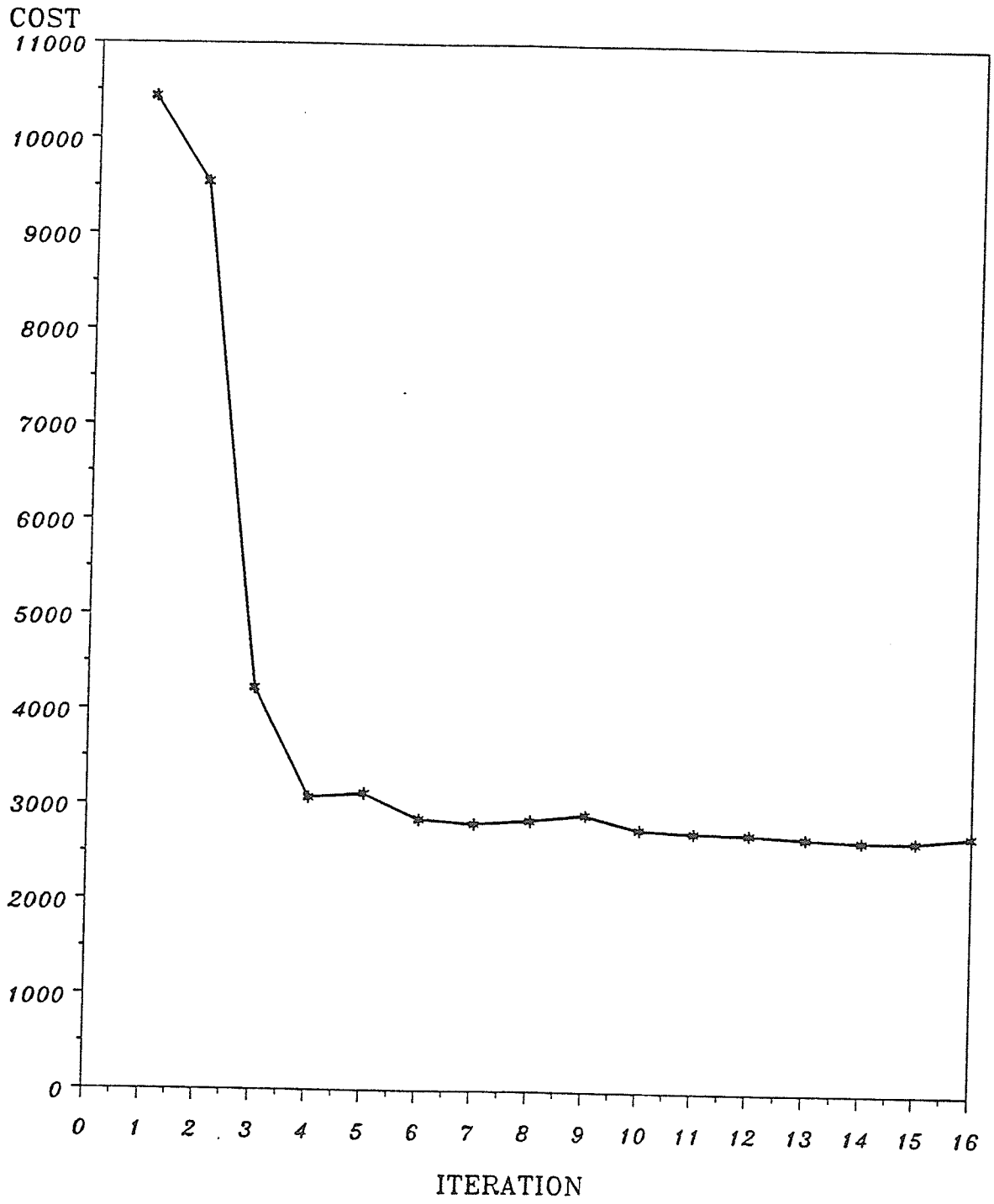

Figure 4.6: Total Traffic Cost of Each Iteration

Table 4.2: Traffic Demands for 7 Time Epochs

O-D	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$	$n=7$
(1,6)	4.0	6.0	8.0	8.0	6.0	5.0	4.0
(2,9)	2.0	8.0	10.0	9.0	5.0	4.0	5.0
(4,9)	4.0	7.0	8.0	7.0	6.0	5.0	5.0

Convergence phenomena of the traffic volumes generated by the algorithm are visible on all the links. For brevity, the volumes and travel times on links (4,5), (4,8) and (1,2) will be presented for this example. The varying traffic volumes of the 7 time epochs with demands for 16 iterations on links (4,5), (4,8) and (1,2) are shown in Figures 4.7, 4.8 and 4.9, respectively.

In Figures 4.7, 4.8 and 4.9, each curve represents traffic volumes on the link at a same time epoch for different iterations. Traffic volumes on links (1,4), (1,5), (2,3), (2,5) and (2,6) have the similar convergence shape as shown in Figures 4.7, 4.8 and 4.9. Traffic volumes on the rest of the links have stable value (a positive constant or zero).

Observations

From the computational results of Example 4.1, we observe the followings:

- total travel time (travel cost) decreases and slightly oscillates around a convergence point as the number of iterations, k, increases (Figure 4.6).
- Traffic volumes, for the same time epoch, on certain links do not change as the number of iterations, k, increases; or when they do change, the changes are very small.
- For each time epoch, traffic volumes tend to converge as the number of itera-

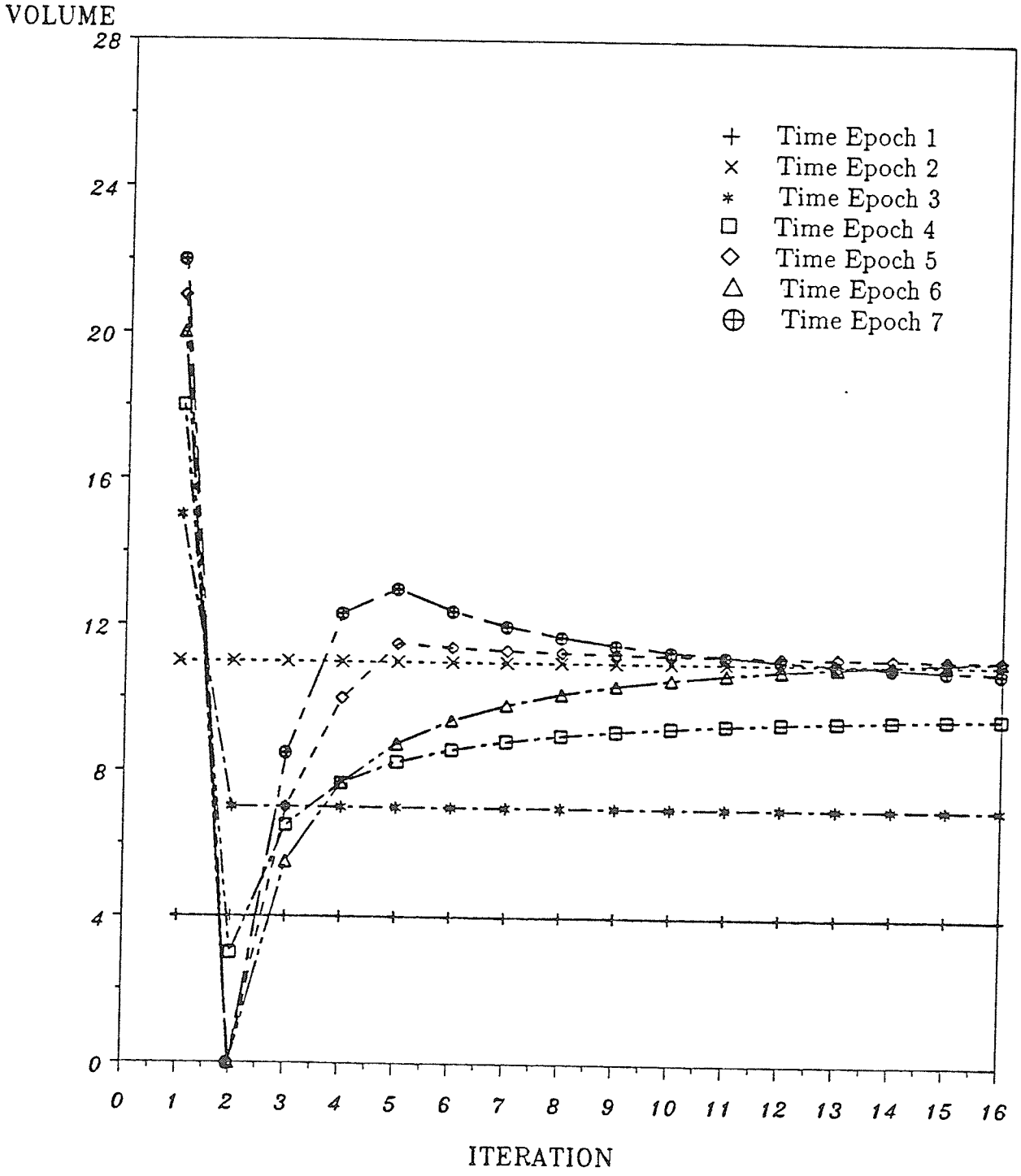

Figure 4.7: Varying Volumes on Link (4,5) at Each Iteration

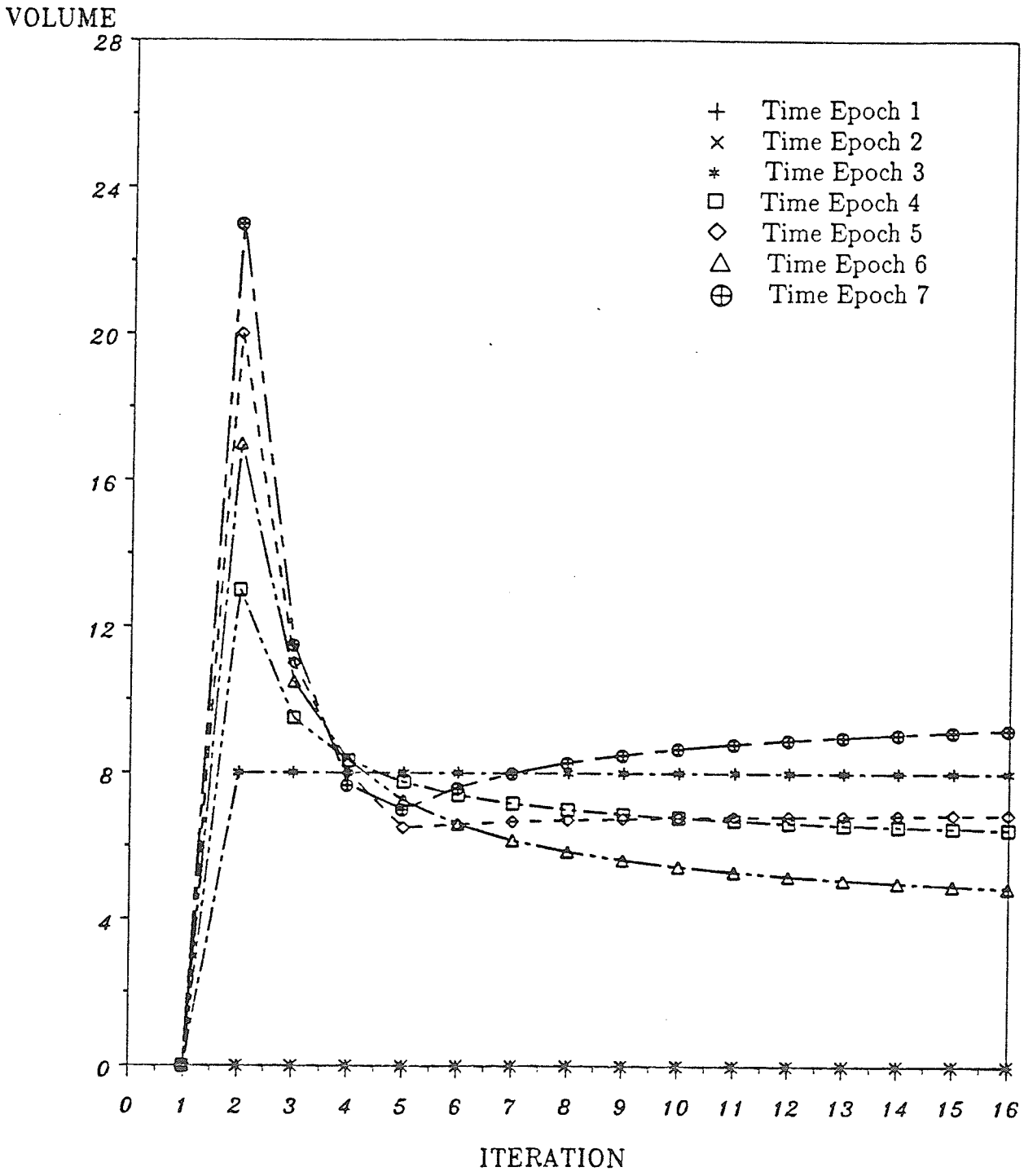

Figure 4.8: Varying Volumes on Link (4,8) at Each Iteration

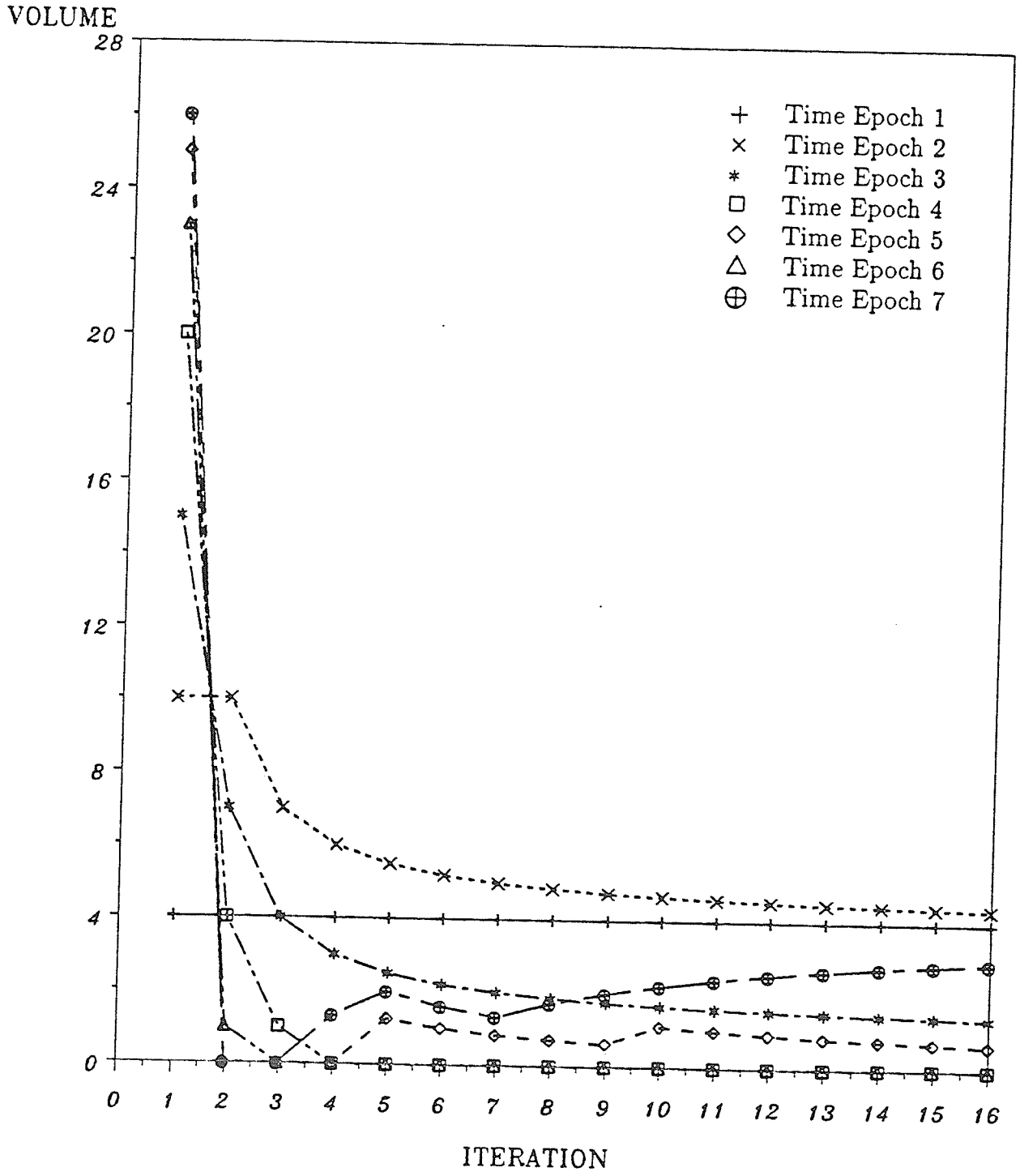

Figure 4.9: Varying Volumes on Link (1,2) at Each Iteration

tions, k, increases.

- MSA is a good approach for further exploration in pursuing an algorithm that can generate convergent solutions for dynamic traffic assignment problems.

CPU time for this example is 9.20 seconds on the mainframe computer. We have assigned various larger numbers of iterations for a number of similar examples, and the convergence phenomena were also observed. Computational complexity depends upon the number of time epochs N_d that have demands, the total number of iterations K and, of course, the network size and configuration. The algorithm needs to calculate, N_d times, the shortest paths in every iteration k. The number of times that the shortest paths are calculated is additive as k increases. However, as k increases, the evaluation of the traffic volumes $x_{ij}^k(n)$ on each link ij (Step 1.4 of Algorithm 4.2) becomes more complicated and the computational time increases exponentially with iteration number k.

The main disadvantage of Algorithm 4.2 is that it requires larger computer memory size, especially when a large number of iterations is needed to achieve convergence. This is because the information required to identify the shortest paths in all previous iterations has to be stored.

The results of this section (Section 4.2) have been submitted by Chen and Alfa (1991c).

4.3 Dynamic Network Design Problems

In Section 3.3, Chapter Three, static network design (ND) problems were discussed. As mentioned earlier at the beginning of this chapter, transportation network planning and design in major urban areas are often carried out for the peak period

during which traffic demands are not constant. Hence it may be difficult to directly apply static ND models and algorithms to solve realistic ND problems in an urban traffic network. To the best knowledge of the author, there are no models reported in the literature that can be used to solve dynamic network design problems. In this section, we will present a dynamic ND model. The model is similar to the ones discussed in Section 3.3, that is, a discrete model with integer decision variables and the construction cost being part of the objective function. A branch and bound algorithm is developed for solving this model.

4.3.1 The Model and the Algorithm

In this model, the sum of total traffic and construction cost is to be minimized; subject to the dynamic UE assignment. The dynamic ND model is:

(DYUEND)

$$\min_{U_h} \sum_{ij \in L_h} T^h + \lambda_d \sum_{l=1}^{M} C_l v_l \tag{4.45}$$

where: $T^h = \sum_{ij} T_{ij}$ with

$$T_{ij} = \sum_{n=1}^{N} x_{ij}^{*}(n) t_{ij}(n)[x_{ij}^{*}(n)], ij \in L_h. \tag{4.46}$$

In Eqs(4.45) and (4.46), L_h is the set of links in the network $G_h = \{V, L_h\}$. $U_h, h = 1, ..., 2^M - 1$, are the decision variables indicating the candidate links in a project. v_l is the cell of a vector U_h. U_h, v_l and C_l are the same as defined in Section 3.3. λ_d expresses the conversion between construction cost and travel time; and $x_{ij}^{*}(n)$ is traffic flow on link ij at time epoch n which satisfies Model 4.4, the dynamic UE assignment model with discrete time.

The above discrete dynamic ND model can be solved by a branch and bound algorithm. The algorithm presented next is similar to Phase 2 of Algorithm 3.8, the

static ND algorithm. The step by step procedure of the algorithm is presented below.

Algorithm 4.3

Step 1. Perform dynamic traffic assignment on network $G_0 = (N, L_0)$;
using Algorithm 4.2 to generate the total traffic cost T_0.
Set the current lower bound $LB = T_0$.
Set current optimal solution $U^* = U_0 = (0, 0, ..., 0, 0)$.
Set $h = D$, where $D = 2^M - 1$.

Step 2. For $G_h = \{V, L_h\}$, perform the dynamic UE assignment
using Algorithm 4.2. Let $TC_h = T_h + \lambda_d C_h^P$.

Step 3. If $TC_h < LB$, set $LB = TC_h$ and $U^* = U_h$. For all $k, k < h$ and $U_h \succ U_k$,
if: $\sum_{ij \in L_h} T_h + \lambda_d C_k^P \geq LB$, (similar to that in Algorithm 3.8, "$U_h \succ U_k$"
means that all the candidate links in project k are included in
project h), then eliminate project k from further consideration,
go to Step 2.3.

Step 4. If $h = 1$, output U^* as the optimal solution, stop;
otherwise, set $h = h - 1$, go to Step 2.

In Algorithm 4.2, C_k^P is the construction cost for project k. It is the same as defined in Section 3.3.

4.3.2 Discussion

In this section, a network design model incorporated with one of the dynamic traffic assignment models discussed in Section 4.2 is presented. The reasons why a model with discrete decision variables is selected are the same as stated in Section 3.3.

Algorithm 4.3 is a branch and bound approach to solve dynamic ND problems.

Since the MSA method, instead of the incremental method as in Algorithm 3.8, is used to generate the traffic assignment, the number of iterations to generate traffic assignment cannot be reduced. On the other hand, the branch and bound procedure in Algorithm 4.3 is less complicated than that in Algorithm 3.8. Algorithm 4.3 is still a heuristic method because:

- the dynamic MSA assignment method embedded in the algorithm is a heuristic dynamic traffic assignment algorithm;
- the phenomenon similar to Braess' Paradox in the static ND problem needs to be explored for the dynamic ND problem if existance of a dynamic user equilibrium can be proved.

Next a numerical example is presented to illustrate the branch and bound algorithm for the dynamic ND problem.

4.3.3 A Dynamic Network Design Example

Example 4.2

The Problem and Computational Result

Consider network G_0 as shown in Figure 4.10. Each link ij in G_0 represents a one-way road from i to j with zero-flow travel time a_{ij} and link capacity c_{ij}, as shown in Table 4.1. Traffic demands of three O-D pairs for seven time epochs are presented in Table 4.2. Four candidate links considered for adding into network G_0 are expressed by the dashed lines as shown in Figure 4.11. The parameters a_{ij} and c_{ij} of network G_0 and the construction costs of the candidate links are presented in Table 4.3. Total iteration number K for dynamic traffic assignment in Steps 1 and 2 of Algorithm 4.3 was set to 15. Computational result of this example are presented

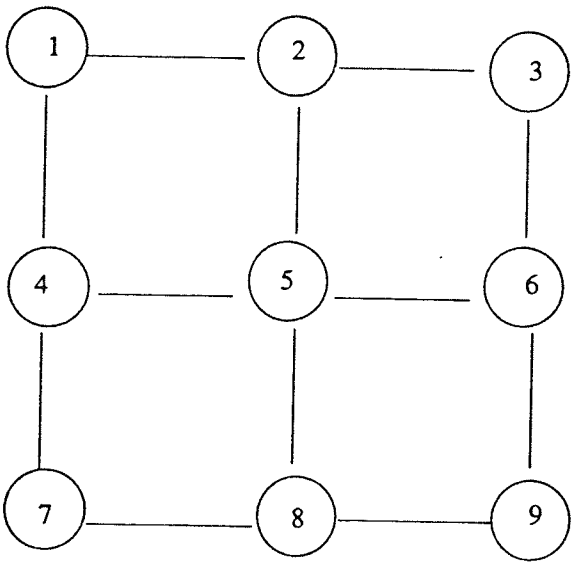

Figure 4.10: Network G_0

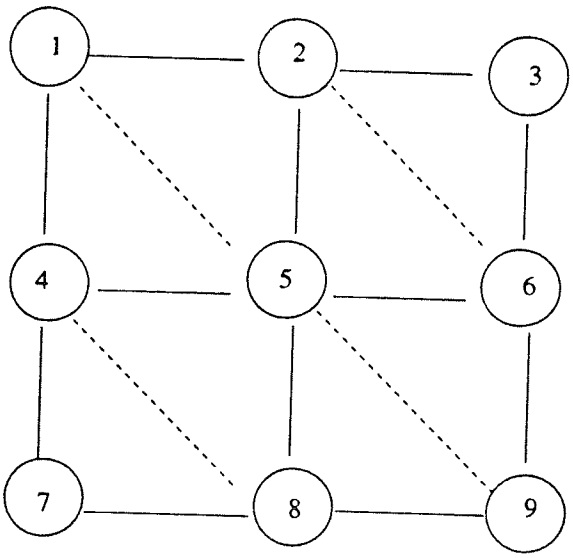

Figure 4.11: Candidate Links in G_0

Table 4.3: Data for Candidate Links of Example 4.2

Link Number	(i,j)	$a_{i,j}$	$c_{i,j}$	$C_{i,j}$
1	1,5	3.0	3.0	1200.0
2	2,6	2.0	2.0	800.0
3	4,8	1.0	2.0	900.0
4	5,9	1.0	4.0	1000.0

Table 4.4: Computational Results of Example 4.2

λ_d	Num. of Assig. Performed	Solution Generated	Total Cost	CPU Time
1.0	14	(0,1,1,1)	5055.2	1M22.4SEC
1.5	12	(0,1,0,0)	5669.9	1M10.6SEC
2.0	10	(0,1,0,0)	6069.9	58.3SEC
2.5	7	(0,1,0,0)	6469.9	41.7SEC
3.0	6	(0,0,0,0)	6715.6	35.9SEC

in Table 4.4 for different values of the conversion parameter λ_d.

Observations

From the computational results presented in Table 4.4, it can be seen that:

- results generated by Algorithm 4.3 are reasonable since the number of the new links to be constructed is reduced with the increase in the value of the converting parameter λ_d;
- CPU time is much longer than the computation of static ND problems due to the higher complexity of the dynamic assignment procedure.

The model and algorithm may be further explored by converting the travel time to be measurable with the construction cost, opposite to the way that the converting parameter is currently used. In that case, a group of converting parameters, that is, a vector instead of a scalar, could be used. The elements of the vector will correspond to the relative importance of the travel time at different time epochs.

4.4 A Comparison Study

In Chapter Three, the traffic assignment and network design (ND) problems with static traffic demand are discussed. In this chapter, the traffic assignment and ND problems with time-varying demands are discussed. Traffic assignment models for solving ND problems with static and dynamic demands are based on different types of performance functions. A direct comparison of the two different models is difficult because the results by the two performance functions on a same link will be very different. The difference will be significant when traffic demand is low compared to the capacities of the links in the network. In this section, we will incorporate an efficient method developed by Alfa (1990) for evaluating expected queue length

into the dynamic model. The impact from the different types of traffic demands and different performance functions on the ND problem solutions will be presented through a number of numerical examples.

4.4.1 Performance functions of Static and Dynamic Assignment Models

It can be seen that in Chapter Three, the performance function of each link ij in the network is assumed to be of BPR (Bureau of Public Roads) type curve, that is

$$t_{ij}(x_{ij}) = a_{ij} + b_{ij}x_{ij}^4,$$

where x_{ij} is the traffic flow on link ij, and a_{ij} and b_{ij} are parameters independent of the traffic flow on link ij. This function is widely used in static traffic analysis.

For the dynamic assignment model discussed in this chapter, the performance function given by the BPR function is inappropriate. The main reason is that this function cannot capture the dynamic traffic flow relationship between the connected links in the network. In the dynamic traffic analysis, the entering flow of link ij at time epoch $n = n_1$ is composed by the exiting flows of those links connected with and directed to link ij at time epoch $n = n_1 - 1$. This relationship cannot be expressed by the performance function given by the BPR curve. Based on this consideration, a different type of "performance function" is used in this chapter for the dynamic traffic analysis as shown in Eqs(4.12) to (4.14).

The performance function expressed by Eqs(4.12) to (4.14) is an acceptable function for dynamic traffic analysis. However, there is a major difference of the behaviors between this function and the static one, the BPR curve.

- For the BPR function, travel time t_{ij} will be greater than the zero-flow travel time for any value of traffic flow $x_{ij} > 0$ on link ij.

- For the function by Eqs(4.12) to (4.14), travel time $t_{ij}(n)$ for time epoch n will be equal to zero-flow travel time for any value of traffic flow $x_{ij}(n)$, $0 \leq x_{ij}(n) \leq c_{ij}$.

Theoretically, traffic assignment with static demand is a special case of traffic assignment with dynamic demands. The comparison of the two assignment models should easily be done by using the dynamic model to solve static problems. However, the differences in the performance functions prevent such a direct comparison. It can be seen that the performance function for dynamic problems is only a simplified form. In a realistic transportation system, travel time on a link ij will normally be greater than zero flow travel time when traffic flow $x_{ij} > 0$, even if $x_{ij}(n) \leq c_{ij}$. Traffic assignment could be different due to different performance functions used. The performance function in the dynamic model needs to be modified before static and dynamic assignment models can be compared. To conduct the comparison study properly, we include an expected queue length calculation procedure, developed by Alfa (1990), into the dynamic assignment algorithm (Algorithm 4.2). The queueing procedure utilizes the maximum entropy principle and is an efficient method for evaluating the expected queue length in a $M(t)/D/1$ queueing system. That procedure will be applied in the network to calculate the queue length and then the vehicle delay time for each link. Details of the procedure for the $M(t)/D/1$ queueing system can be found in Alfa (1990).

4.4.2 Numerical Examples

In this section, four numerical examples will be used for comparison. All the example problems are ND problems. The same branch and bound method given by Algorithm 4.3 is used to solve the ND problems based on the same network and

Table 4.5: Link Parameters of Network G

(i,j)	a_{ij}	c_{ij}	(i,j)	a_{ij}	c_{ij}
(1, 2)	2.0	3.0	(1, 4)	1.0	3.0
(2, 3)	3.0	2.0	(2, 5)	4.0	3.0
(3, 6)	1.0	3.0	(4, 5)	2.0	4.0
(4, 7)	3.0	2.0	(5, 6)	1.0	3.0
(5, 8)	3.0	4.0	(6, 9)	4.0	3.0
(7, 8)	1.0	2.0	(8, 9)	2.0	3.0

Table 4.6: Data for Candidate Links of Network G

Link Number	(i,j)	$a_{i,j}$	$c_{i,j}$	$C_{i,j}$
1	1,5	3.0	3.0	1200.0
2	2,6	2.0	2.0	800.0
3	4,8	1.0	2.0	900.0
4	5,9	1.0	4.0	1000.0

same set of candidate links. The network and the candidate links are presented in Figure 4.12. Data of the links of the network and candidate links are shown in Tables 4.5 and 4.6.

Four Numerical Examples

The four examples are designed to compare the models with simple performance function to the performance function from the $M(t)/D/1$ queue. These examples are also used to compare situations where the traffic demands are constant vs. time-varying.

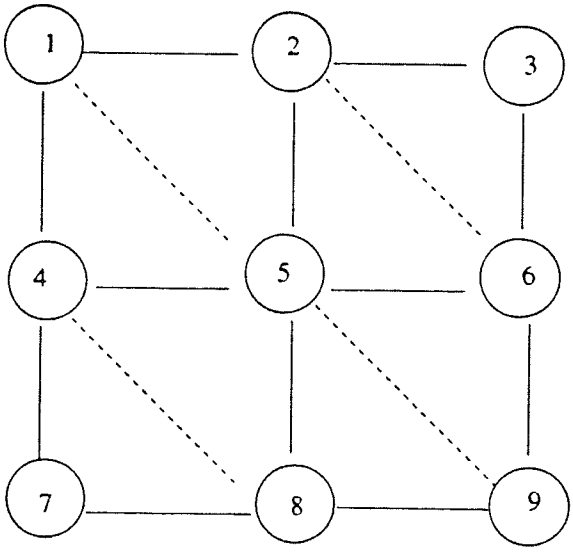

Figure 4.12: A Network for Comparison

Table 4.7: Dynamic Traffic Demands for 7 Time Epochs

O-D	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$	$n=7$
(1,6)	2.0	3.0	4.0	4.0	3.0	2.5	3.0
(2,9)	2.0	4.0	5.0	4.5	2.5	2.0	2.5
(4,9)	2.0	3.5	4.0	3.5	3.0	2.5	2.5

In Example 4.3, an ND problem with time-varying demands as shown in Table 4.7 is solved by Algorithm 4.3. The "regular" performance function presented in Eqs(4.12), (4.13) and (4.14) is assumed for each link in the network. The results of Example 4.3 are presented in Table 4.8. Different values of parameter λ_d are used to convert the construction cost to be measureable with the total travel times.

In Example 4.4, the problem is the same as in Example 4.3, except that the travel time calculation of each link includes the results by the $M(t)/D/1$ queueing model. The entropy approximate method developed in Alfa (1990) is used to generate the expected queue length at the exiting end of each link in the network. The results of Example 4.4 are presented in Table 4.9. The same set of parameter λ_d as in

Table 4.8: Computational Results of Example 4.3

λ_d	Num. of Assig. Performed	Solution Generated	Total Cost	CPU Time
1.0	3	(0,0,0,0)	1247.4	17.7SEC
0.8	5	(0,0,0,0)	1247.4	29.0SEC
0.6	6	(0,0,0,0)	1247.4	35.3SEC
0.4	8	(0,1,0,0)	1081.3	45.2SEC
0.2	14	(0,1,0,0)	901.3	1M21.7SEC

Table 4.9: Computational Results of Example 4.4

λ_d	Num. of Assig. Performed	Solution Generated	Total Cost	CPU Time
1.0	8	(0,1,0,0)	1694.5	1M31.6SEC
0.8	9	(0,1,0,0)	1534.5	1M45.4SEC
0.6	14	(0,1,0,0)	1374.5	2M42.3SEC
0.4	13	(0,1,0,1)	1124.1	2M32.8SEC
0.2	10	(0,1,0,1)	764.1	1M59.0SEC

Table 4.10: Static Traffic Demands for 7 Time Epochs

O-D	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$	$n=7$
(1,6)	2.9	2.9	2.9	2.9	2.9	2.9	2.9
(2,9)	3.2	3.2	3.2	3.2	3.2	3.2	3.2
(4,9)	3.0	3.0	3.0	3.0	3.0	3.0	3.0

Table 4.11: Computational Results of Example 4.5

λ_d	Num. of Assig. Performed	Solution Generated	Total Cost	CPU Time
1.0	3	(0,0,0,0)	1256.3	18.1SEC
0.8	5	(0,0,0,0)	1256.3	30.2SEC
0.6	6	(0,0,0,0)	1256.3	36.2SEC
0.4	10	(0,0,0,1)	1094.6	59.0SEC
0.2	16	(0,0,0,1)	894.6	1M33.7SEC

Example 4.3 is used for this example.

Example 4.5 is similar to Example 4.3. The difference is that the static traffic demands are assumed in this example. The constant traffic demands are shown in Table 4.10. Results of this example are presented in Table 4.11. Example 4.6 is similar to Example 4.4. The difference is that the static traffic demands are assumed. The traffic demands, shown in Table 4.10, are the same as in Example 4.5. The results of Example 4.6 are shown in Table 4.12.

Table 4.12: Computational Results of Example 4.6

λ_d	Num. of Assig. Performed	Solution Generated	Total Cost	CPU Time
1.0	6	(0,1,0,0)	1454.2	1M10.2SEC
0.8	7	(0,1,0,0)	1294.2	1M22.5SEC
0.6	10	(0,1,0,0)	1134.2	1M58.2SEC
0.4	12	(0,1,0,0)	974.2	2M22.5SEC
0.2	11	(0,1,1,1)	812.5	2M09.6SEC

4.4.3 Observations

From the results of the above examples, it can be seen that the decisions of network design problems vary when different performance functions are used in the same model with same type of traffic demands. This can be observed by comparing the results of Examples 4.3 and 4.4, and by comparing the results of Examples 4.5 and 4.6. It also can be seen that the network design decisions will be different for a same model if different types of (static or dynamic) traffic demands are assumed. Two of the conclusions from the computational results of the four examples are:

- Dynamic traffic assignment and ND models should be developed and used to solve dynamic problems. If a dynamic problem is treated as a static problem, wrong solutions could be generated.

- More accurate performance functions should be used whenever it is necessary to obtain accurate results for dynamic ND problems.

4.5 Summary

In this chapter, the traffic assignment problem and network design problem with time-varying demands were discussed. We concentrated on the models and algorithms that can be used to solve the problems in a general network.

The major results presented in this chapter are as follows:

- A reasonable and simple algorithm (Algorithm 4.2) for dynamic traffic assignment is developed.
- Mathematical support for the new algorithm in a simplified (two link) network is presented.
- A dynamic traffic assignment model is, for the first time, incorporated into a ND model.
- Static and dynamic ND approaches are compared through a number of randomly selected numerical examples.

In Chapter Three we discussed static traffic assignment and network design problems. In this chapter we discussed the dynamic problems. It is obvious that the dynamic problems are much more complicated and require much more effort to solve than static ones. However, it is necessary to use dynamic models to solve dynamic problems. This conclusion can be drawn from the comparison study of the static and dynamic assignment and ND problems conducted in this chapter.

Chapter 5

An Extension of the Transportation Models

5.1 Introduction

In the previous chapters, we discussed the static and dynamic transportation assignment and network design (ND) problems. A number of mathematical models and algorithms are presented in those chapters. In this chapter, we will present an application of dynamic traffic assignment and ND models in a job routing problem and a system design problem in a flexible manufacturing system.

Traffic assignment and ND models not only can be used to solve transportation problems but also can be applied to solve problems in the other fields of Operations Research and Industrial Engineering. A job scheduling and routing problem in the flexible manufacturing systems (FMS) with time-varying demands and alternative machines can be solved by a modified dynamic traffic assignment model and an algorithm presented in Chapter Four.

5.2 A Brief Literature Review

Routing flexibility in a multi-machine production system is one of the most important aspects in achieving a high level flexibility in FMS. Techniques used to solve parts routing problems include:

- Optimization models, based on integer programming or branch and bound approach. This approach may not be efficient for large-scale problems.
- Heuristic job dispatching rules are widely applied in solving FMS scheduling problems. The effectiveness and efficiency of heuristic rules may vary for problems with different features. Extensive simulation study is needed to test these rules for different types of manufacturing systems and different routing problems.

Parts routing and scheduling problems in FMS has attracted much attention from both researchers and practitioners. In the literature, we found that Nsar and Elsayed (1990) discussed a scheduling problem in a job shop with alternative machines. Avonts and Van Wassenhove (1988) developed an approach by combining linear programming and queueing network models for solving a routing and scheduling problem. Maimon and Choong (1987) studied a flow shop system with time dependent product demands. Raman et al. (1989) extended the static scheduling rules for a static single machine system to a dynamic single machine system. Shanthikumar and Buzacott (1984) used a queueing network model to study the time spent by a job in a dynamic job shop. Maimon and Gershwin (1988) developed a dynamic programming model for solving a dynamic FMS scheduling and routing problem.

The parts routing problem considered in this chapter is a more complicated one which can be solved using the method developed in this transportation study.

5.3 Parts Routing in a Flexible Manufacturing System

In this section, we use a heuristic algorithm to solve a part routing and scheduling problem in a flexible manufacturing system with time-varying job (part) demands. Time-varying aspect in this context means that at different times there may be different types and different numbers of parts coming to the system and waiting to be scheduled and processed. Scheduling and routing problems involving time-varying demands are more realistic in a manufacturing system, because most of today's manufacturing systems are operated on a continuing basis. Types and numbers of the parts to be processed will vary from time to time. Solving this scheduling and routing problem requires the assignment of operations of parts to machines in response to the time-varying demands in quantities and varieties of parts production.

The goal here is as follows: given a job's arrival (release) time, to schedule its processing via the appropriate set of machines, taking full advantage of the flexibility of the machines and of the job's manufacturing alternatives. By the manufacturing alternatives we mean that an operation of a part can be carried out by one of several different machines in the manufacturing system.

The heuristic algorithm is adopted from the dynamic traffic assignment method (Algorithm 4.2, Chapter Four). It is not based on an integer programming approach or heuristic job dispatching rules. The algorithm is based on the method of successive averages (MSA) and it can be used for solving dynamic parts routing problems in a FMS where the objective is to minimize the completion time of an individual batch

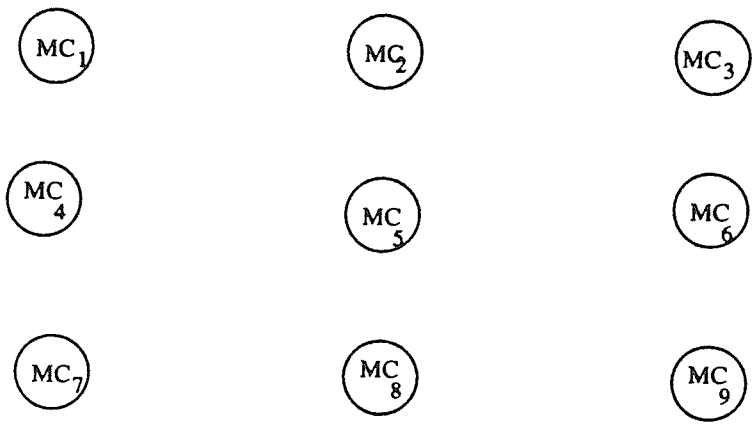

Figure 5.1: A Manufacturing System

of parts.

5.3.1 Problem Statement

Consider an FMS consisting of a number of machines or machine centers, $\mathrm{MC}_m, m = 1, 2, ..., M$, as shown in Figure 5.1. Observe the FMS at equally spaced time epochs sequentially numbered by $n, n = 1, 2, ..., N$. Assume that at the beginning of some of the time epochs, there are a number of different part types waiting to be processed by the FMS. Let N_d $(1 \leq N_d < N)$ be the last time epoch at which there are parts to enter the system. Define $N_p(n)$ as the number of part types entering the FMS at time epoch n. Then by the definition of N_d, we have: i) $N_p(N_d) > 0$; ii) $N_p(n) \geq 0$, for n, $1 \leq n < N_d$; and iii) $N_p(n) = 0$, for n, $N_d < n \leq N$. For the parts to be

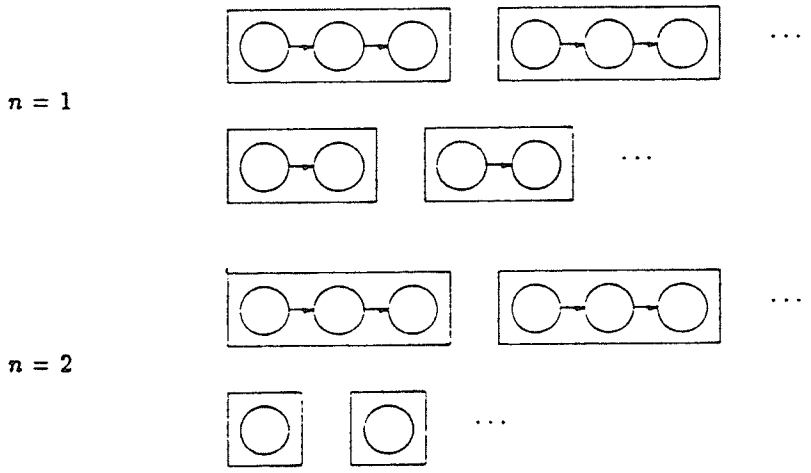

Figure 5.2: Parts to Be Processed

processed, we assume that:

i) all the part types are in batches with each batch having at least one part (Figure 5.2);

ii) each part type has a number of operations to be carried out by different machines in the systems;

iii) an operation of a part might be carried out by one of a number of the machines in the system;

iv) a part cannot be processed by more than one machine at the same time;

v) any operation being carried out will continue to completion without interruption; and

vi) the precedence relationships of operations for each part type are predetermined and enforced (Figure 5.2);

For machines in the FMS, we assume that:

i) different machines can carry out the same operation (but not necessarily with the same processing time), so that the system is "flexible";

ii) there are no machine breakdowns during the time horizon of the part processing;

iii) in process storage is allowed with an unlimited buffer capacity for the parts queued, if there is a queue, at each machine;

iv) each machine applies "first come, first serve" rule for processing the parts waiting in the queue at the machine;

v) no machine has idle time as long as jobs are waiting to be processed; and

vi) transportation times for the parts from one machine to another are negligible.

With the above assumptions, we develop an algorithm to solve the parts routing problem in FMS with time-varying demands. The objective of the problem is to complete, as early as possible, the processing of each part by the manufacturing system. Achieving this objective may not necessarily result in the total makespan of all the parts being minimized. In other words, the result is "individual optimal" rather than "system optimal". Therefore, the method and the algorithm are more suitable for scheduling a small service system which processes contracted parts for outside companies.

5.3.2 Using MSA for Parts Routing

First we present a small example to illustrate the basic idea of the MSA algorithm for solving this parts routing problem.

Example 5.1

Consider the machines in the FMS as nodes in a network and all the possible routings of each part type as connections (links) between the nodes. Let us assume that there is only one part type with three operations to be carried out by the manufacturing system shown in Figure 5.1. Table 5.1 shows the three manufacturing alternatives ($j = 1, 2, 3$) for the operations of this part type for this illustrative example. Table 5.1 also shows the machine numbers (columns 2,4 and 6) and the corresponding processing times for the operations (columns 3,5 and 7). Assume that the batch size of this part type is four. The machines shown in Figure 5.1 can be connected by the links according to the manufacturing alternatives given in Table 5.1. The constructed network can be found in Figure 5.3(a). To simplify the analysis, an artificial node (Node 0) was added to the network in Figure 5.3(a). At the first

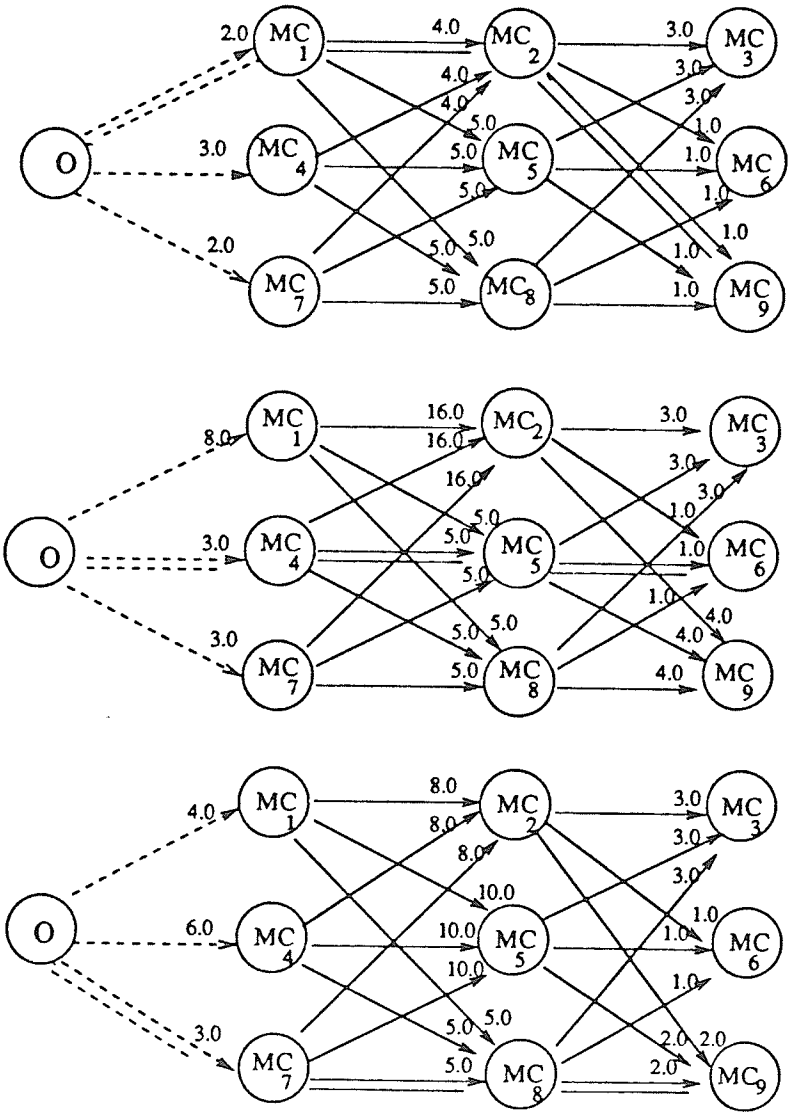

Figure 5.3: Constructed Networks

Table 5.1: Manufacturing Alternatives for Example 6.1

j	$m_j^1(1)$	$t_j^1(1)$	$m_j^1(2)$	$t_j^1(2)$	$m_j^1(3)$	$t_j^1(3)$
1	1	2.0	2	4.0	3	3.0
2	4	3.0	5	5.0	6	1.0
3	7	3.0	8	5.0	9	1.0

iteration of the dynamic routing algorithm, the distances of links in the network are assigned by processing times as shown in Figure 5.3(a). A shortest route algorithm, is then applied to this network to find the minimum completion time routing (MCTR) for the part type, which is the routing (0-)1-2-9 in this network. At the second iteration, the distances of the links in the network will be updated based on the assumption that the whole batch of the (four) parts have been loaded on the generated routing (0-)1-2-9. Assume that the updated distances of the network are as shown in Figure 5.3(b). A new MCTR, (0-)4-5-6 at this iteration, is generated based on the updated distances. At the third iteration, the distances in the network will be updated again based on the assumption that $4 \times 1/2(= 2)$ parts have been loaded on the routing (0-)1-2-9 and the other two parts have been loaded on routing (0-)4-5-6 (Figure 5.3(c)). Assume that a new MCTR, routing (0-)7-8-9, is found at the third iteration. Then the distances will be updated based on the assumption that each of the routings (0-)1-2-9, (0-)4-5-6 and (0-)7-8-9 has been loaded with $4 \times 1/3$ parts. Let us assume that the loading and reloading procedure will stop at the fourth iteration and the MCTRs found at each of the four iterations are (0-)1-2-9, (0-)4-5-6, (0-)7-8-9 and (0-)1-2-9 again. Then the final part routings for this part type will be: two parts to be processed by machines 1,2,and 9; one part to be

processed by machines 4,5, and 6; and one part to be processed by machines 7,8, and 9.

The above illustration of the MSA routing procedure is for a single part type and a single time epoch case. For the general situations, the idea is the same as illustrated in Example 5.1. The dynamic shortest route algorithm, similar to the one in Eqs(4.15) and (4.16), Chapter Four, is used to take into account the time-varying aspect of the network.

The MSA approach has been successfully used in solving static and dynamic transportation assignment problems as discussed in Chapters 3 and 4. We have not been able to prove the optimality of the algorithm in this manufacturing application. MSA can only be considered as a heuristic algorithm in solving this parts routing problem. In order to formally present the algorithm, we define:

$N_p(n)$ number of part types entering the FMS at time epoch $n, n = 1, 2, ..., N_d$;

$N_b^i(n)$ number of parts in a batch of part type i entering the FMS at time epoch n, $i = 1, 2, ..., N_p(n), n = 1, 2, ..., N_d$;

N_o^i number of operations of part type i, $i = 1, 2, ..., N_p(n), n = 1, 2, ..., N_d$;

N_{pp}^i number of manufacturing alternatives for the operations of part type i, $i = 1, ..., N_p(n), n = 1, ..., N_d$;

pp_j^i manufacturing alternatives for the operations of part type i, $j = 1, ..., N_{pp}^i$, $i = 1, ..., N_p(n), n = 1, ..., N_d$;

A "manufacturing alternative" for an operation of a part type is composed by the machine number and the processing time for that operation. For brevity, a manufacturing alternative is given for a part type instead of individual operations. Thus, the j-th manufacturing alternative of part type i, pp_j^i, is expressed by:

$$pp_j^i = [(m_j^i(1), t_j^i(1)), (m_j^i(2), t_j^i(2)), ..., (m_j^i(N_o^i), t_j^i(N_o^i))],$$

$$j = 1, ..., N^i_{pp}, i = 1, ..., N_p(n), n = 1, ..., N_d;$$

where $m^i_j(h)$ and $t^i_j(h)$ are:

$m^i_j(h)$ the machine to carry out operation h of part type i, according to manufacturing alternative j,

$t^i_j(h)$ the processing time for operation h of part type i, according to manufacturing alternative j,

$$h = 1, ..., N^i_o, j = 1, ..., N^i_{pp}, i = 1, ..., N_p(n), n = 1, ..., N_d;$$

We assume that $N^i_{pp} \geq 1, \forall i$. This means that an operation on a part type can normally be carried out by one of many machines in the system. As defined earlier, machine numbers in the manufacturing alternatives indicate which machine can be used to carry out an operation of a part type. They do not specify a particular part routing to produce a part of that part type. For example, if part type 1 has 3 operations and there are 2 manufacturing alternatives known as:

$pp^1_1 = [(1, 5.5), (2, 6.0), (3, 3.5)]$ and

$pp^1_2 = [(4, 5.0), (5, 7.0), (6, 4.5)]$,

then it means that part type 1 can be produced by any of the following $8(2^3)$ part routings: 1-2-3, 1-2-6, 1-5-3, 1-5-6, 4-2-3, 4-2-6, 4-5-3 and 4-5-6.

For each machine or machine center MC_m, $m = 1, 2, ..., M$, define $T_m(n)$ as the time to complete the processing of the last part waiting at MC_m at time epoch n. $T_m(n)$ is evaluated by the following recursive formula:

$$\left\{ T_m(n) = T_m(n-1) + \delta^i_m(n)\tilde{N}^i_{m,b}(n)t^i_j(h)_n - s_m(n),\ n = 2, 3, ..., N, \right. \qquad (5.1)$$

where: $T_m(1) = 0$,

and

i) $\delta_m^i(n) = \begin{cases} 1, & \text{if a part of part type } i \text{ entering the FMS at time epoch } n \\ & \text{is loaded on machine } m; \\ 0 & \text{otherwise,} \end{cases}$

ii) $\tilde{N}_{m,b}^i(n)$ is the number of parts of part type i entering the FMS at time epoch n and being loaded on machine MC_m,

iii) $t_j^i(h)_n$ is the processing time for operation h according to pp_j^i for the part type i, entering the FMS at time epoch n, $1 \leq j \leq N_{pp}^i$, and

iv) $s_m(n)$ is a counting parameter given as follows:

$$\begin{cases} s_m(1) = 0 \\ s_m(n) = \begin{cases} s_m(n-1)+1, & \text{if } T_m(n-1) \neq 0; \\ 0, & \text{otherwise;} \end{cases}, \quad n = 2, 3, ..., N. \end{cases} \tag{5.2}$$

With the above definitions and expressions, the MSA algorithm for the dynamic parts routing is given below.

Algorithm 5.1

Step 0. Set $K(K \geq 1)$ to an integer and let $k = 0$.

For each time epoch n', $n' = 1, ..., N_d$, use the manufacturing alternatives to generate a routing in the network for each part type.

Assume no parts are being processed by the system.

For each machine MC_m, $m = 1, ..., M$, generate the waiting/completion time $T_m^{(0)}(1)$, by $(m_j^i(h), t_j^i(h))$, $\forall i, j, h$.

Let $T_m^{(0)}(2) = \cdots = T_m^{(0)}(N) = T_m^{(0)}(1)$.

Step 1. Let $k = k + 1$.

Based on the latest waiting/completion time $T_m^{(k-1)}(n), n = 1, ..., N$, $m = 1, ..., M$, use the dynamic shortest route algorithm to obtain

k-th MCTR for each part type i, for $n', n' = 1, ..., N_d$.

Step 2. Let $\bar{N}_b^i(n') = N_b^i(n')/k$, $n' = 1, ..., N_d; \forall i$.

Load the number equal to $\bar{N}_b^i(n')$ of parts of part type i on each MCTR k', $k' = 1, ..., k$, for each time epoch $n', n' = 1, ...N_d, \forall i$;

Step 3. Calculate the waiting/completion time $T_m^{(k)}(n)$ by using Eqs(5.1) and (5.2); for $n = 1, ..., N, m = 1, ..., M$.

Step 4. If $k \geq K$, stop. The result is to load $N_b^i(n')/K$ number of parts of part type i entering the system at time epoch $n', n' = 1, ..., N_d$, on the k-th MCTR, goto Step 1.

A numerical example is presented next to illustrate the algorithm.

5.3.3 A Parts Routing Example

Example 5.2

The Input Data of the Example

This example is based on the nine machine system as shown in Figure 5.1. Parts are released to the systems in three time epochs ($N_d = 3$). The number of part types, the batch size and the number of operations of each part type are given in Table 5.2. At time epoch 1, for instance, there are 24, 12 and 12 parts in a batch for part types 1, 2 and 3, respectively. The number of operations and the number of manufacturing alternatives of each part type for this example are also shown in Table 5.2. The manufacturing alternatives for the part types are given in Table 5.3.

Computational Results

Computational results of this example with 12 total iterations are presented in Tables 5.4, 5.5 and 5.6. They show the different parts routings and the number of parts assigned to these routings for each part type. The latest finishing time $\tilde{T}$ of

Table 5.2: Data of Parts for Example 5.2

n	$N_p(n)$	i	$N_k^i(n)$	N_o^i	N_{pp}^i
1	3	1	24	3	3
		2	12	3	3
		3	12	2	3
2	3	1	24	3	2
		2	12	2	2
		3	24	3	2
3	2	1	12	2	2
		2	12	3	3

Table 5.3: Manufacturing Alternatives of Example 5.2

n	i	j	$m_j^i(1)$	$t_j^i(1)$	$m_j^i(2)$	$t_j^i(2)$	$m_j^i(3)$	$t_j^i(3)$
1	1	1	1	2.0	2	4.0	3	3.0
		2	4	3.0	5	5.0	6	1.0
		3	7	3.0	8	5.0	9	1.0
	2	1	1	4.0	4	3.0	7	4.0
		2	2	2.0	5	2.0	8	4.0
		3	3	6.0	6	4.0	9	2.0
	3	1	1	2.0	2	3.0	–	–
		2	4	6.0	6	2.0	–	–
		3	7	4.0	9	4.0	–	–
2	1	1	1	1.0	4	4.0	8	2.0
		2	2	3.0	5	3.0	9	4.0
	2	1	7	4.0	1	2.0	–	–
		2	6	3.0	3	4.0	–	–
	3	1	3	2.0	2	4.0	4	5.0
		2	6	3.0	5	2.0	7	7.0
3	1	1	1	1.0	7	4.0	–	–
		2	3	3.0	9	2.0	–	–
	2	1	7	1.0	4	3.0	1	4.0
		2	8	2.0	5	2.0	2	3.0
		3	9	1.0	6	3.0	3	4.0

Table 5.4: Generated Parts Routings for Example 5.2 ($n = 1$)

$i = 1$		$i = 2$		$i = 3$	
Routing	$NP_1(1)$	Routing	$NP_2(1)$	Routing	$NP_3(1)$
1-2-6	4	3-4-7	1	4-6	1
4-5-3	2	3-5-9	1	1-2	1
7-5-6	2	2-5-7	1	7-6	1
1-8-9	6	2-5-8	1	1-6	1
4-2-9	2	2-5-9	3	4-9	1
4-8-9	2	2-4-7	1	7-9	4
1-5-6	2	2-6-9	1	1-9	3
7-8-6	2	3-6-9	1	–	-
4-8-6	2	2-4-9	1	–	-
–	-	3-4-9	1	–	-

where $NP_i(n)$ is the number of parts of part type i loaded on the routing, $n = 1, 2, 3$.

Table 5.5: Generated Parts Routings for Example 5.2 ($n = 2$)

$i = 1$		$i = 2$		$i = 3$	
Routing	$NP_1(2)$	Routing	$NP_2(2)$	Routing	$NP_3(2)$
2-4-8	2	6-1	6	6-2-4	4
1-5-8	12	6-3	2	3-2-7	2
2-5-8	6	7-1	3	3-5-4	6
1-4-8	2	7-3	1	6-5-7	2
2-4-9	2	–	-	3-5-7	2
–	-	–	-	3-2-4	4
–	-	–	-	6-2-7	4

Table 5.6: Generated Parts Routings for Example 5.2 ($n = 3$)

$i = 1$		$i = 2$	
Routing	$NP_1(3)$	Routing	$NP_2(3)$
3-7	1	8-4-2	1
1-9	6	7-5-2	1
1-7	2	9-5-2	3
3-9	3	9-4-1	1
-	-	8-4-3	1
-	-	8-5-1	2
-	-	9-5-1	2
-	-	9-6-3	1

all the machines is used as a measurement of the results. $\tilde{T}$ is evaluated as:

$$\tilde{T} = \max_m \{\tilde{T}_m\}, \tag{5.3}$$

where $\tilde{T}_m$ is the time epoch that the processing of all the parts assigned to machine MC_m has been finished. The values of $\tilde{T}$ at different iterations are shown in the chart of Figure 5.4. CPU times for this example is 31.01 seconds. Results of this section (Section 5.2) has been sumbitte by Chen and Alfa (1991d).

The parts routing algorithm can be embedded in a manufacturing system design model which will be discussed next.

5.4 A Manufacturing System Design Problem

The FMS parts routing model can be further extended to a manufacturing system design model. A branch and bound algorithm similar to Algorithm 4.3, Chapter Four, is used to solve the manufacturing system design problem.

5.4.1 The Model and the Algorithm

A manufacturing system design problem needs to be addressed if:

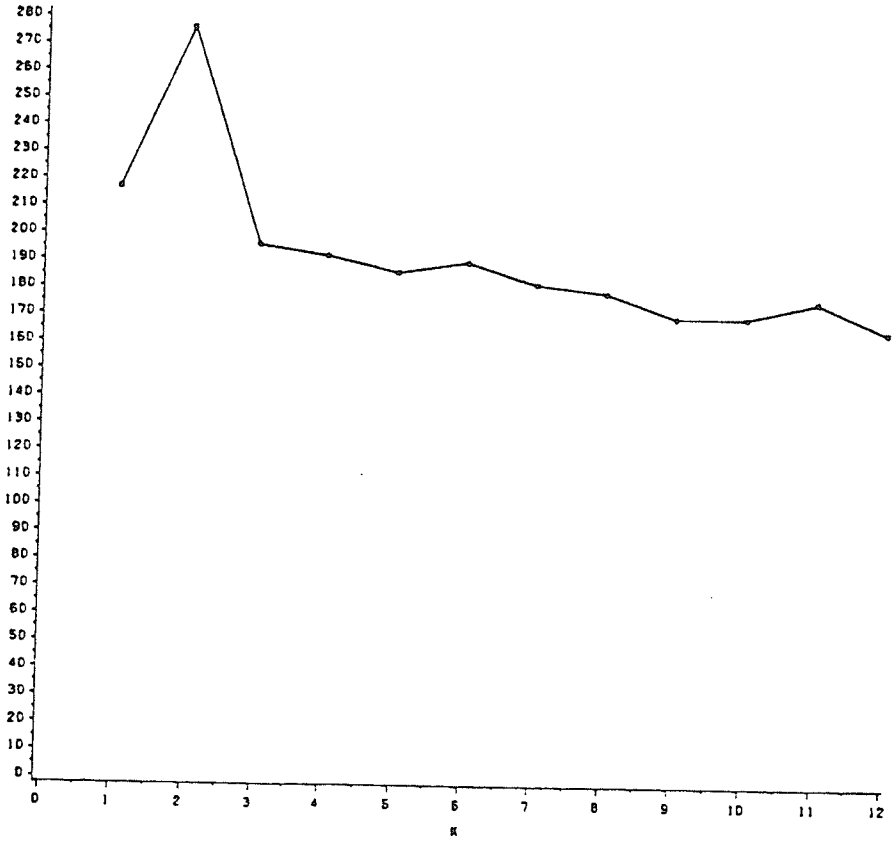

Figure 5.4: Total Completion Time of Each Iteration

- the time-varying demands of the jobs to be processed by the FMS is stable, that is, the pattern of the varying demands of the jobs will repeat upon a larger time horizon; and
- the operating cost of a certain number of the machines is very high and some of the machines in the system should be non-active for a particular set of jobs.

For example, consider the FMS parts routing problem in Example 5.2. Assume that there are 3 more machines available and the cost of including these machines is known. Similar to the transportation ND problem, making decisions to include some or all of the new machines into the current system can be considered as a system design problem. The objective function of this model is the sum of the total time to complete all the jobs and the fixed and/or operating cost added (saved) by including (removing) the new (existing) machines. This objective is to be minimized subject to the results generated by Algorithm 5.1. The system design model can be presented as:

(MSD)

$$\min_{U_h} \sum_{m \in \mathrm{MS}_h} \tilde{T}_h + \lambda_m \sum_{l=1}^{\mathrm{MC}} C_l v_l \tag{5.4}$$

where:

$$\tilde{T}_h = \max_{m \in \mathrm{MS}_h} \{\bar{T}_m\} \tag{5.5}$$

with

$$\bar{T}_m = \{n^* | x_m^*(n^*) > 0 \text{ and } x_m^*(n^*+1) = 0\}, m \in \mathrm{MS}_h \tag{5.6}$$

In Eqs(5.4), (5.5) and (5.6), MS_h is the set of machines in the manufacturing system corresponding to the h-th set of "candidate machines"; $U_h, h = 1, ..., 2^M - 1$, are the decision variables indicating the new (existing) machines considered to be added (removed); v_l is the cell of vector U_h and is corresponding to each individual

machine considered; C_l is the sum of the fixed and operating cost of including a machine l (note that C_l dose not include the cost of removal); λ_m expresses the conversion between the machine cost and the job completion time; and $x_m^{\star}(n)$ is the number of parts on machine m at time epoch n generated by Algorithm 5.1. These variables are similar to the corresponding variables defined for transportation ND problems in Chapters Three and Four.

To solve this model, the branch and bound algorithm similar to Algorithm 4.3 in Chapter Four is used. In Algorithm 5.2, C_h^P is the total cost to include the machines indicated by vector U_h, $h = 1, ...2^M - 1$. The step by step procedure of the algorithm is presented below:

Algorithm 5.2

Step 1. Use Algorithm 5.1 to generate part routings based on the existing manufacturing system, MS_0; calculate the latest completion time $\tilde{T}_0$.
Set the current lower bound $LB = \tilde{T}_0$.
Set current optimal solution $U^{\star} = U_0 = (0, 0, ..., 0, 0)$.
Set $h = D$, where $D = 2^M - 1$.

Step 2. For the system MS_h generate part routings using Algorithm 5.1.
Let $TC_h = \tilde{T}_h + \lambda_m C_h^P$.

Step 3. If $TC_h < LB$, set $LB = TC_h$ and $U^{\star} = U_h$. For all $k, k < h$ and $U_h \succ U_k$, if: $\sum_{ij \in L_h} T_h + \lambda_d C_k^P \geq LB$ (similar to that in Algorithm 4.3, "$U_h \succ U_k$" means that all the machines in the machine set k are included in the machine set h), then eliminate machine set k from further consideration, go to Step 4.

Step 4. If $h = 1$, output $U^{\star}$ as the optimal solution, stop; otherwise, set $h = h - 1$,

go to Step 2.

5.4.2 A Manufacturing System Design Example

Example 5.3

Example 5.3 is to illustrate Algorithm 5.2. This example is based on the parts routing problem of Example 5.2. Assume that the same set of jobs in Example 5.2 will be carried out by the manufacturing system. Also assume that in the manufacturing system shown in Figure 5.1, there are 3 new machines, Machines 10, 11 and 12 are considered to be included in the system. And assume that two existing machines, Machines 4 and 8 are considered to be made non-active. We call these five machines the "candidate machines". The manufacturing system with these "candidate machines" are shown in Figure 5.5. The savings of the operating cost from stopping Machines 4 and 8 in the existing system are 150 and 150 units, respectively; and the cost of adding Machines 10, 11 and 12 to the manufacturing system are 100, 100 and 65 units, respectively. A new set of manufacturing alternatives for the same set of parts are shown in Table 5.7. Note that there is a larger number of manufacturing alternatives for each operation of the parts shown in Table 5.7 than those shown in Table 5.3 for Example 5.2, since the new machines are also capable of carrying out some of these operations. The computational result of this example with $\lambda_m = 0.1$ is (0, 0, 1, 1, 0). That is, using the two existing machines, Machines 4 and 8 in the system, should be stopped; and two new machines, Machines 10 and 11, should be included in the system. In solving this manufacturing system design problem, 32 times of the parts routing algorithm (Algorithm 5.1) have been performed. CPU time is about 15 minutes for this system design example in the mainframe computer.

Table 5.7: Manufacturing Alternatives (Example 5.3)

n	i	j	$m_j^i(1)$	$t_j^i(1)$	$m_j^i(2)$	$t_j^i(2)$	$m_j^i(3)$	$t_j^i(3)$
1	1	1	1	2.0	2	4.0	3	3.0
		2	4	3.0	5	5.0	6	1.0
		3	7	3.0	8	5.0	9	1.0
		4	10	1.0	11	2.0	12	1.0
	2	1	1	4.0	4	3.0	7	4.0
		2	2	2.0	5	2.0	8	4.0
		3	3	6.0	6	4.0	9	2.0
		4	10	2.0	11	1.0	12	1.0
	3	1	1	2.0	2	3.0	–	–
		2	4	6.0	6	2.0	–	–
		3	7	4.0	9	4.0	–	–
		4	11	2.0	12	1.0	–	–
2	1	1	1	1.0	4	4.0	8	2.0
		2	2	3.0	5	3.0	9	4.0
		3	3	6.0	6	6.0	11	1.0
	2	1	7	4.0	1	2.0	–	–
		2	6	3.0	3	4.0	–	–
		3	10	2.0	5	8.0	–	–
	3	1	3	2.0	2	4.0	4	5.0
		2	6	3.0	5	2.0	7	7.0
		3	9	7.0	11	2.0	10	2.0
3	1	1	1	1.0	7	4.0	–	–
		2	3	3.0	9	2.0	–	–
		3	2	6.0	12	1.0	–	–
	2	1	7	1.0	4	3.0	1	4.0
		2	8	2.0	5	2.0	2	3.0
		3	9	1.0	6	3.0	3	4.0
		4	12	1.0	11	2.0	10	2.0

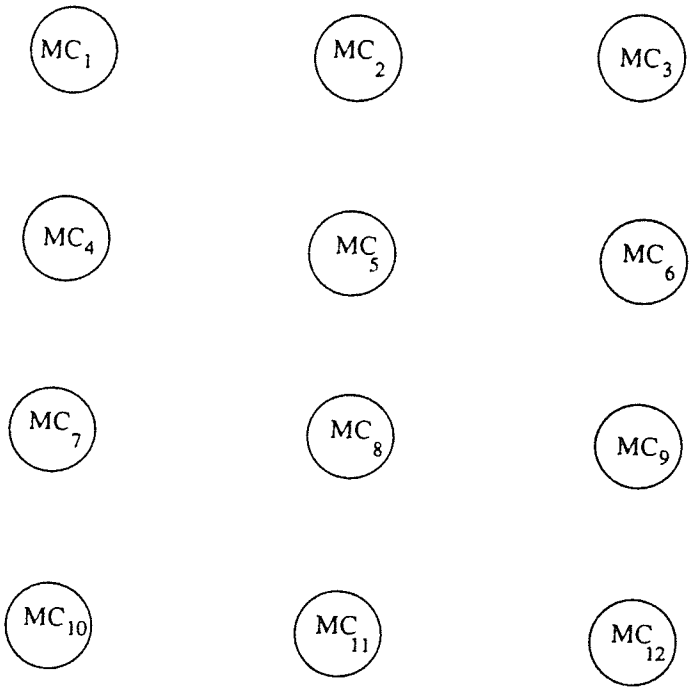

Figure 5.5: A Manufacturing System for Example 5.3

5.5 Discussion

A parts routing and scheduling problem in an FMS is a combinatorial problem and is usually difficult to solve. Adding the aspect of time-varying demands makes the problem more complicated, even though it becomes more realistic. The method used to solve the parts routing and system design problems presented in this chapter are based on dynamic traffic assignment and network design algorithms. The advantages of using Algorithms 5.1 and 5.2 is that they are simple and easy to implement. A number of limitations in using Algorithm 5.1 for parts routing and scheduling are:

- The objective of Algorithm 5.1 is to complete the operations of each individual batch of the parts as soon as possible, not to minimize the total makespan of all the parts. As mentioned earlier, this method is for "individual optimal" rather than for "system optimal". Algorithm 5.1 may not be applicable in solving a problem in which system optimal is more important.

- The results generated by Algorithm 5.1 are not necessarily integers. If the method is used for generating long term plans and schedules, this would not be a major problem; otherwise, adjusted schedules from round-off solutions have to be considered. One approach for handling this non-integer problem is to select different iteration number K in the algorithm for different problems to obtain integer solutions.

The branch and bound algorithm, Algorithm 5.2, for solving the manufacturing system design problem can be used to select machines for a particular set of parts. The objective function we used is the maximum completion time of the parts on all the machines with converted fixed and operating cost of the machines in the system.

5.6 Summary

In this chapter, we discussed an extension of the transportation assignment and network design problems. The application of the dynamic transportation assignment models for the parts routing problem in a flexible manufacturing system with time-varying demands is discussed. A modified transportation MSA algorithm is developed for solving this manufacturing problem. The parts routing problem is further extended to a manufacturing system design problem and a branch and bound algorithm is developed. Numerical examples are provided to illustrate the approaches and algorithms. The manufacturing problems and the methods discussed in this chapter show that certain transportation problems and manufacturing problems have similarities and they can be solved by similar techniques. The branch and bound algorithm developed to solve the manufacturing system design problem is a direct transformation from the dynamic network design model. Solving realistic manufacturing system design problems can be expected with further modifications and development of the models and the algorithms presented in this chapter.

Chapter 6

Discussion and Conclusions

6.1 Discussion and Conclusions

In this thesis the research work focuses on ND problems in road transportation networks. Traffic assignment problems need to be solved before ND problems can be properly studied. Static system optimal or deterministic user equilibrium traffic assignment problems have been studied extensively by many researchers. Well-supported models and efficient algorithms are available for solving these problems.

To study ND problems in urban road networks, time-varying traffic demands should be considered. In an urban network, the objective of solving ND problems is to reduce the traffic congestions of the peak hours during which traffic demands are time-varying. Dynamic traffic assignment and ND problems are much more complicated than static ones and much more work needs to be done in order to properly address these problems.

A few concluding points can be drawn from this research as listed below.

- Well-developed techniques and algorithms are available for solving static traffic assignment problems. Normally, there will be no major difficulty in applying the available models and algorithms to solve large scale and realistic traffic

assignment problems with the help of a mainframe or even a micro computer. The static models and algorithms are also supported by very strong mathematical analysis and most of them can be solved optimally. Although there may be little space left for future research of the static deterministic user equilibrium (DUE) assignment problems, there are still unsolved problems in the static stochastic user equilibrium (SUE) assignment. Stochastic traffic assignment is a very important issue. As discussed in Chapter Three, it eliminates the unrealitic assumption of the deterministic traffic loading approach. It also can eventually replace the DUE model since DUE can be considered as a special case of SUE. Also as shown in Chapter Three, network design decisions depend on the traffic assignment models used. The decision could be different if a SUE model is used instead of a DUE model. A major reason that the SUE model was not sufficiently addressed is that it normally requires more computation. The development of the new algorithms presented in Chapter Three for solving a static SUE assignment problem is an attempt to find efficient methods to solve the problem.

- Solving network design problems, even without considering the time-varying aspect, is a more difficult and complex research topic. Completely different models must be formulated for ND problems with different assumptions. Approaches for solving these various models range from traditional linear programming method to expert systems techniques. In this thesis research, we constrain our interest in ND problems in a congested network with a user equilibrium traffic assignment. The branch and bound ND algorithm developed in this research can solve static stochastic user equilibrium ND problems in a comparably large general network. This is the first time that a stochastic

traffic assignment model is embedded in a ND algorithm. The weakness of the algorithm is that, like any other method based on branch and bound approach, it will not be very efficient when an ND solution is to be generated from large numbers of candidate links.

- Adding the time-varying aspect to traffic assignment models and ND models results in a much higher complexity of problems. There are very few models and algorithms available for solving a user equilibrium traffic assignment problem with time-varying demands in a general network. Mathematical models for this problem are difficult to formulate and theoretical analysis is difficult to develop. One of the main difficulties is to mathematically formulate the phenomenon that travellers departing at different origins and at different time epochs may encounter to each other at a same node in the network. The dynamic traffic assignment algorithm developed in this thesis (Chapter Four) is an attempt to address this problem properly. Although it is a heuristic method, the theoretical basis for this algorithm in a simple network is established in this research. Randomly selected numerical examples presented in this thesis also shows encouraging results. However, much more research is needed to develop appropriate mathematical models and efficient algorithms for solving dynamic traffic assignment problems. With proper models and efficient algorithms for dynamic traffic assignment, it will not be very difficult to formulate and solve dynamic ND problems by using available techniques. The comparison study presented in Chapter Four shows that solutions of ND problems are different when different types of traffic demands are assumed. Wrong decisions might be reached if the time-varying demands of an ND problem are replaced by constant traffic demand. Hence using a static approach based on

a simplified assumption to solve a dynamic ND problem may generate wrong solutions.

- It is interesting to note that there are similarities between traffic assignment problems in transportation and parts routing problems in manufacturing. A same approach, after minor modifications, can be applied to solve similar problems in the two different areas. A model and an algorithm developed for solving dynamic traffic assignment and ND problems are successfully applied to solve a parts routing problem and a system design problem in a manufacturing system as presented in Chapter Five.

6.2 Possible Future Research

The research of transportation ND problems presented in this thesis is a complete research in itself. Possible future research topics in this area may include:

- For the user equilibrium network design problem with static traffic demands, continuous models and algorithms need further development. In this research, we used discrete ND models and algorithms which can generate realistic results. Like any other discrete method, these algorithms suffer inefficiency when they are used to solve large scale problems. The results from continuous models may not be directly implementable and they normally need further round-offs. Algorithms for solving continuous models are normally more efficient. More study of continuous models and algorithms, therefore, should be conducted.

- For the traffic assignment problem with time-varying demands, there are many difficult and open problems to be solved. To conduct a thorough theoretical

analysis of the problem, one needs to develop suitable mathematical optimization models. Since it could be very difficult to formulate proper mathematical models for this problem, more heuristic algorithms based on simple and reasonable common logic should be developed. These algorithms should be efficient to solve realistic large scale problems. More models and algorithms for solving dynamic ND problems also need to be developed. Also, more computational experiments need to be done to compare different dynamic assignment and ND models and algorithms.

- Applications of expert system techniques in solving ND problems need to be explored. In solving realistic ND problems, normally there are many qualitative factors involved. Expert system techniques are good at handling such unstructured problems. There are not many published papers reporting applications of this powerful technique for solving transportation ND problems.

- Applications of transportation models and algorithms to solve problems in other related areas such as manufacturing system analysis need further exploration because some problems in those different areas may be similar in nature.

6.3 Final Remarks

Transportation network design problems are very important issues facing transportation system planners in their daily work. The study of transportation network design problems is also a very interesting research topic. A large amount of research work has been done in this area and many useful results have been achieved. Models, algorithms and computer software developed for solving practical traffic as-

signment and ND problems have been applied in practical planning work. Benefits of applications of the research work are visible.

In this area, there still are many problems remaining open. More realistic models and algorithms need to be developed in order that this very crucial problem can be addressed more properly and solved more accurately and efficiently.

Bibliography

[1] Abdulaal, M. and LeBlanc, L.J. (1979) "Continuous equilibrium network design models", *Transportation Research (B),* Vol.13B, pp.19-32.

[2] Alfa, A.S. (1986) "A review of models for the temporal distribution of peak traffic demand", *Transportation Research (B),* Vol. 20B, pp.491-499.

[3] Alfa, A.S. (1987) "Deterministic model for dynamic assignment of time-varying traffic demand", *Civil Engineering Systems,* Vol.4, pp.191-198.

[4] Alfa, A.S. (1990) "Approximating queue lengths in $M(t)/D/1$ queues", *European Journal of Operational Research,* Vol.44, pp.60-66.

[5] Avonts, L.H. and Van Wassenhove, L.N. (1988) "The part mix and routing mix problem in FMS: A coupling between an LP model and a closed queueing network", *International Journal of Production Research,* Vol.26, pp.1891-1902.

[6] Ben-Akiva, M. (1985) "Dynamic network equilibrium research", *Transportation Research,(A),* Vol.19A, No.5/6, pp.429-431.

[7] Blum, J.R. (1954) "Multidimensional stochastic approximation methods", *Annals of Mathematical Statistics,* Vol.25, pp.737-744.

[8] Boffey, T.B. and Hinxman, A.I. (1979) "Solving for optimal network problems", *European Journal of Operational Research,* Vol.3, pp.386-393.

[9] Carey, M. (1987) "Optimal time varying flows on congested networks", *Operations Research*, Vol.35, No.1 pp.56-69.

[10] Chen, M. and Alfa, A.S. (1991a) "Algorithms for solving Fisk's stochastic traffic assignment model", *Transportation Research*, forthcoming, 1991.

[11] Chen, M. and Alfa, A.S. (1991b) "A network design algorithm using stochastic incremental traffic assignment approach", *Transportation Science*, forthcoming, 1991.

[12] Chen, M. and Alfa, A.S. (1991c) "The method of successive averages for solving dynamic traffic assignment problems", submitted to *Operations Research.*

[13] Chen, M. and Alfa, A.S. (1991d) "Parts routing in a flexible manufacturing system with time-varying demands", submitted to *European Journal of Operational Research.*

[14] Cooke, K.L. and Halsey, E. (1966) "The shortest route through a network with time-dependent transit times", *Journal of Math. Anal. & Appl.*, Vol.14, pp.493-498.

[15] Current, J.R., ReVelle, C.S. and Cohon, J.L. (1985) "The maximum covering/shortest path problem: a multiobjective network design and routing formulation", *European Journal of Operational Research,* Vol.21, pp.189-199.

[16] Current, J.R., ReVelle, C.S. and Cohon, J.L. (1987) "The median shortest path problem: a multiobjective approach to analyze cost vs. accessibility in the design of transportation networks", *Transportation Science,* Vol.21, No.3, pp.188-197.

[17] Daganzo, C.F. (1979) *Multinomial Probit: The Theory and Its Application to Demand Forecasting,* Academic Press, New York.

[18] Daganzo, C.F. and Sheffi, Y. (1977) "On stochastic models of traffic assignment", *Transportation Science*, Vol.11, No.3, pp.253-274.

[19] Dantzig, G.B. and Maier, S.F. (1979) "Formulating and solving the network design problem by decomposition", *Transportation Research (B),* Vol. 13B, pp.5-17.

[20] Dial, B.R. (1971) "A probabilistic multipath traffic assignment algorithm which obviates path enumeration", *Transportation Research*, Vol.5, No.2, pp.83-111.

[21] Drissi-Kaitouni, O. (1990) "A model for the dynamic traffic assignment problem", presented at the ORSA/TIMS 30th Joint National Meeting, Philadelphia, Pennsylvania, October 29-31, 1990.

[22] Ferland, J.A., Florian, M. and Achim, C. (1975) "On incremental methods for traffic assignment," *Transportation Research*, Vol. 9, 237-239.

[23] Fisk, C. (1980) "Some developments in equilibrium traffic assignment", *Transportation Research(B)*, Vol.14B, No.3, pp.243-255.

[24] Florian, M. and Nguyen, S. (1976) "An application and validation of equilibrium trip assignment methods", *Transportation Science*, Vol.10, pp.374-390.

[25] Friesz, T.L., Luque, J., Tobin, R.L. and Wie, B.-W. (1989) "Dynamic network traffic assignment considered as a continuous time optimal control problem", *Operations Research*, Vol.37, No.6, pp.893-901.

[26] Geoffrion, A.M. (1972) "Generalized Benders decomposition", *Journal of Optimization Theory and Applications,* Vol.10, No.4, pp.237-260.

[27] Hoang, H.H. (1982) "Topological optimization of network: a nonlinear mixed integer model employing generalized Benders decomposition", *IEEE Transactions on Automatic Control,* Vol. AC-27, No.1, pp.164-169.

[28] Horowitz, J.L. (1984) "The stability of stochastic equilibrium in a two-link transportation network", *Transportation Research(B),* Vol.18B, No.1, pp.13-28.

[29] Janson, B.N. (1990) "A convergent algorithm for dynamic traffic assignment", presented at the ORSA/TIMS 30th Joint National Meeting, Philadelphia, Pennsylvania, October 29-31, 1990.

[30] Janson, B.N. and Husaini, A. (1987) "Heuristic ranking and selection procedures for network design problems", *Journal of Advanced Transportation,* Vol.21, Spring, pp.17-46.

[31] LeBlanc, L.J. (1975) "An algorithm for the discrete network design problem", *Transportation Science,* Vol.9, pp.183-199.

[32] LeBlanc, L.J. (1976) "Global solutions for a nonconvex, nonconcave rail network model", *Management Science,* Vol.23, No.2, pp.131-139.

[33] LeBlanc, L.J. and Abdulaal, M. (1979) "An efficient dual approach to the urban road network design problem", *Computer and Mathematics with Applications,* Vol.5, pp.11-19.

[34] LeBlanc, L.J. and Abdulaal, M. (1984) "A comparison of the user-optimum versus system-optimum traffic assignment in transportation network design", *Transportation Research B,* Vol.18B, No.2, pp.115-121.

[35] Leonard, D.R., Gower, P. and Taylor, N.B. (1989) "CONTRAM: Structure of the model", Research Report 178, Transport and Road Research Laboratory, Crowthorne, Berkshire, U.K.

[36] Magnanti, T.L. and Wong, R.T. (1984) "Network design and transportation planning: models and algorithms", *Transportation Science,* Vol.18, No.1, pp.1-55.

[37] Mahmassani, H. and Herman, R. (1984) "Dynamic user equilibrium departure time and route choice on idealized traffic arterials", *Transportation Science,* Vol.18, No.4, pp.362-384.

[38] Maimon, O.Z. and Choong, Y.F. (1987) "Dynamic routing in reentrant flexible manufacturing systems", *Robotics and Computer-Integrated manufacturing,* Vol.3, pp.295-300.

[39] Maimon, O.Z. and Gershwin, S.B. (1988) "Dynamic scheduling and routing for flexible manufacturing systems that have unreliable machines", *Operations Research,* Vol.36, pp.279-292.

[40] Marcotte, P. (1983), "Network optimization with continuous control parameters", *Transportation Science,* Vol.17, No.2, pp.181-197.

[41] Merchant, D.K. and Nemhauser, G.L. (1978) "A model and an algorithm for the dynamic traffic assignment problems", *Transportation Science,* Vol.12, No.3, pp.183-199.

[42] Nsar, N. and Elsayed, E.A. (1990) "Job shop scheduling with alternative machines", *International Journal of Production Research*, Vol.28, pp.1595-1690.

[43] Poorzahedy, H. and Turnquist, M.A. (1982) "Approximate algorithms for the discrete network design problem", *Transportation Research (B)*, Vol.16B, No.1, pp.45-55.

[44] Powell, W.B. and Sheffi, Y. (1982) "The convergence of equilibrium algorithms with predetermined step sizes", *Transportation Science*, Vol.16, No.1, pp.45-55.

[45] Raman, N., Rachamadugu, R.V. and Talbot, F.B. (1989) "Real-time scheduling of an automated manufacturing center", *European Journal of Operational Research*, Vol.40, pp.222-242.

[46] Ravindran, A., Phillips, D.T. and Solberg, J.J. (1987) *Operations Research: Principles and Practice*, John Wiley and Sons, NY.

[47] Robbins, H. and Monro, S. (1951) "A stochastic approximation method", *Annals of Mathematical Statistics*, Vol.22, pp.400-407.

[48] Shanthikumar, J.G. and Buzacott, J.A. (1984) "The time spent in a dynamic job shop", *European Journal of Operational Research*, Vol.17, pp.215-226.

[49] Sheffi, Y. (1985), *Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods*, Prentice-Hall, Englewood Cliffs, NJ.

[50] Sheffi, Y. and Powell, W.B. (1981) "A comparison of stochastic and deterministic traffic assignment over congested networks", *Transportation Research(B)*, Vol.15B, No.1, pp.53-64.

[51] Steenbrink, P.A. (1974), *Optimization of Transportation Networks,* John Wiley and Sons, London.

[52] Strang, G. (1980) *Linear Algebra and Its Applications (Second Edition)*, Academic Press Inc., New York, NY.

[53] Suwansirikul C., Friesz, T.L. and Tobin, R.L. (1987) "Equilibrium decomposed optimization: a heuristic for the continuous equilibrium network design problem", *Transportation Science,* Vol.21, No.4, pp.254-263.

[54] Taha, H.A. (1975), *Integer Programming: Theory, Applications, and Computations,* Academic Press, New York, N.Y.

[55] Wilde, D.J. (1964), *Optimum Seeking Methods*, Prentice-Hall, Inc., Englewood Cliffs, N.J.