A Flow-Based Approach to the Dynamic Traffic Assignment Problem: Formulations, Algorithms and Computer Implementations

by

Yiyi He

Submitted to the Department of Civil and Environmental Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Transportation

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1997

Author .... [illegible] ....
Department of Civil and Environmental Engineering
May 27, 1997

Certified by ....[illegible]..........................
Ismail Chabini
Assistant Professor of Civil and Environmental Engineering,
Massachusetts Institute of Technology
Thesis Supervisor

Accepted by .... [illegible] ....
Joseph M. Sussman
Chairman, Departmental Committee on Graduate Studies

A Flow-Based Approach to the Dynamic Traffic Assignment Problem: Formulations, Algorithms and Computer Implementations

by

Yiyi He

Submitted to the Department of Civil and Environmental Engineering
on May 27, 1997, in partial fulfillment of the
requirements for the degree of
Master of Science in Transportation

Abstract

Intelligent Transportation Systems (ITS) are being developed to improve the efficiency of present transportation systems. Two of the building blocks of ITS are Advanced Traffic Management Systems (ATMS) and Advanced Traveler Information Systems (ATIS). Dynamic Traffic Assignment (DTA) models can provide support to the evaluation and operation of ATMS/ATIS. The objectives of this thesis are (1) to propose a general modeling framework for the DTA problem, (2) to formulate an analytical DTA model along with the development and implementation of solution algorithms based on the proposed framework. The model is expected to overcome some limitations of known models and algorithms.

The proposed modeling framework contains four major components: users' behavior model, dynamic network loading model, link performance model and paths generation module. These components are interrelated to find a solution to the DTA problem. A DTA model is formulated based on this framework. Three types of route choice behaviors are considered: (1) following fixed route, (2) choosing the route with minimum perceived travel time, and (3) choosing the route with minimum actual travel time. These route choice behaviors are formulated as an equivalent variational inequality. The dynamic network loading model component is formulated as a system of equations expressing link dynamics, flow conservations, flow propagations and boundary constraints. Two link performance models are proposed for uninterrupted and interrupted traffic, respectively.

Heuristic solution algorithms are developed and implemented for the users' behavior model, the dynamic network loading model and the DTA model. Computational efficiency is achieved by using efficient data structures and special methods to reduce memory usage and running time. Two computational examples demonstrate that the software system can not only give meaningful results, but also can find a solution much faster than real time.

Thesis Supervisor: Ismail Chabini
Title: Assistant Professor of Civil and Environmental Engineering, Massachusetts Institute of Technology

Acknowledgments

I would like to thank my thesis supervisor Professor Ismail Chabini for his guidance, advice and encouragement that he provided at all stages of this research.

Dr. Qi Yang provided the data for the Amsterdam example and information every time it was needed. Hong Jin helped to get the results in the computational examples. Xiujuan Wang and Winston Guo offered valuable assistance in plotting some figures in this thesis. Todd Pendleton and Jennifer Glover proofread a draft of this thesis. I appreciate their help.

Thanks to my fellow students and friends for making my experience at MIT enjoyable.

Thanks to the Center for Transportation Studies for providing me financial support.

I would also like to thank my family for their love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The introduction of the automobile has changed our world. It has stimulated the economy and increased our independence and mobility. The enormous growth of road traffic, however, has resulted in heavy congestion in and around urban areas, causing serious problems for the economy because of increased delays and for the environment due to increased emissions.

There could be several solutions to resolve congestion problems. One solution, on the supply side, would be to expand the roadway system. However, such an approach is expensive and is not always feasible because of spatial and environmental limitations, and the very high cost of building new roads, especially in urban areas.

Another solution, on the demand side, would be to reduce car use by changing land use patterns and encouraging people to travel by foot, bicycle, public transit or high occupancy vehicles. However, this approach may not be easy to implement and could achieve results in the long term only.

A new way to alleviate congestion problems is to use the existing transportation infrastructure more efficiently through providing information to travelers and better traffic management. For example, travelers can make travel decisions to improve their traveling benefits by using real-time information, and congestion can be reduced by using dynamic ramp metering and optimizing traffic signals.

The above approach can be implemented by using a group of technologies known as Intelligent Transportation Systems (ITS). ITS can help to use the existing transportation infrastructure more efficiently and improve travel conditions. ITS are composed of a number of technologies, including information processing, communications, control and electronics. The expected benefits from ITS are:

- to improve travel conditions,
- to increase safety, and
- to reduce energy use and environmental impacts.

Two building blocks of ITS are the Advanced Traffic Management Systems (ATMS) and the Advanced Traveler Information Systems (ATIS). ATMS is expected to integrate the management of various roadway functions. Real-time data will be collected and disseminated. Dynamic traffic control systems will respond in real-time to changing network conditions.

ATIS involves providing information to travelers in their vehicle, in their home, or at their place of work. Examples of information are: location of incidents, weather problems, road conditions and optimal routing. Users can make their travel decisions based on the information provided by ATIS.

In order to support the evaluation and operation of ATMS/ATIS, decision support systems, in form of computer software tools, are needed. These are computer implementations of solution algorithms aimed at solving complex dynamic models of transportation systems. Models, solution algorithms and computer implementations constitute the intelligent core of ATMS/ATIS.

Two key requirements for these decision support systems are accuracy and fast response to changing network conditions. In order to meet these requirements, the dynamic traffic models in these systems must be accurate in representing relevant aspects of the transportation system, and the solution algorithms and computer implementations must be able to find a solution in a running time that is faster than real time.

1.1 Problem Definition and Research Approach

As has been previously mentioned, ATMS/ATIS are the two building blocks of ITS. One of the integral parts of ATMS/ATIS is its capability of routing vehicles in response to changing traffic conditions. An effective route guidance should be generated by considering future traffic conditions. Therefore, dynamic traffic models are needed to provide ATMS/ATIS the capability of traffic prediction.

An example of dynamic traffic models is the Dynamic Traffic Assignment (DTA) model. A DTA model is aimed to determine the network conditions for given time-dependent Origin-Destination (O-D) demands, driver's behaviors and a road network.

Thst two types of approaches to dynamic traffic assignment modeling. One is based on simulation models. The other is based on analytical models. Simulation models explicitly consider movement of vehicles. MITSIM[47] and MesoTS[48] are two examples of such models. Simulation models are then more suitable for detailed design and evaluation purposes. However, they do not possess analytical properties.

Analytical approaches use flow dynamics equations to represent traffic movements so they possess some analytical properties. However, as we will see in the remaider of this thesis, existing DTA models have a number of limitations in modeling users' behaviors and network performance. Solution algorithms for these models are not efficient and large problems cannot be solved in real time.

This thesis focuses on a flow-based approach to the DTA problem. Model formulations, algorithms development and computer implementations constitute the core of this thesis work. The objectives of this thesis are:

- to propose a modeling framework for dynamic traffic modeling problems,
- to formulate a DTA model based on the proposed framework to be used within the context of ATMS/ATIS,
- to develop algorithms to obtain solutions to the proposed models,
- to develop computer implementations that can solve realistic DTA problems in real time, and

- to evaluate the proposed models, algorithms and computer implementations using a real application.

1.2 Thesis Outline

The thesis are organized as follows:

Chapter 2 reviews the relevant literature on analytical DTA models. A common structure is abstracted from the proposed framework introduced in Chapter 3 and used as a guide to analyze some representative models. It can be seen from this chapter that existing models have several limitations and that research towards a new or improved DTA model is needed.

In Chapter 3, we propose a flow-based modeling framework for the DTA problem and formulate a DTA model based on the proposed framework to be used within the context of ATMS/ATIS. The framework has a modularized structure. It has helped us to understand existing DTA models in the literature and provides better flexibility in model formulation, algorithm development and computer implementation. The DTA model based on this framework captures different users' route choice behaviors. Link performance is modeled more accurately by using different link travel time models for uninterrupted and interrupted traffic flows. models for uninterrupted and interrupted traffic flows.

Chapter 4 develop algorithms to obtain solutions to various models formulated in Chapter 3. The solution algorithms presented in this chapter includes: (1) two dynamic network loading algorithms, (2) a route choice algorithm, and (3) a DTA algorithm. The dynamic network loading algorithms are used to solve the dynamic network loading model. The route choice algorithm is used to assign O-D flows among the set of paths based on the users' route choice behaviors and the current network conditions. The DTA algorithm is a process to solve the DTA model based on dynamic network loading and route choice algorithms.

Chapter 5 describe a software system that implements the solution algorithms presented in Chapter 4. This chapter presents the methods that were developed in order to achieve fast execution time.

Chapter 6 uses two examples to (1) to show that the proposed DTA model is capable of giving reasonable results, (2) understand the behaviors of the solution algorithms, and (3) to evaluate the computational performance of the computer implementations.

The final chapter summarize the thesis work and suggests directions for future research.

Chapter 2

Literature Review

The Dynamic Traffic Assignment (DTA) problem has been studied by a number of researchers, who recognized dynamic aspects of network flows especially during peak hours. Various models have been proposed. This review aims to (1) extract a common structure of the existing models, (2) discuss the modeling assumptions and the methodology used in those models, (3) understand the advantages and limitations of the existing models and solution algorithms.

Section 2.1 presents a common structure of the existing DTA models. A number of representatives models are analyzed in Sections 2.2, 2.3, 2.4, 2.5 and 2.6. A summary is given in Section 2.7.

2.1 A Common Structure of DTA Models

Most existing DTA models share a common structure. In the literature, this common structure is often not explicitly stated in model formulation, but it can be extracted from those models. As will be seen in Chapter 3, this common structure can be viewed as a high-level abstraction of our proposed modeling framework.

This common structure consists of the following components:

- a demand model
- a supply model

- a supply/demand interaction mechanism

This structure is depicted in Figure 2-1. The demand model component repre-

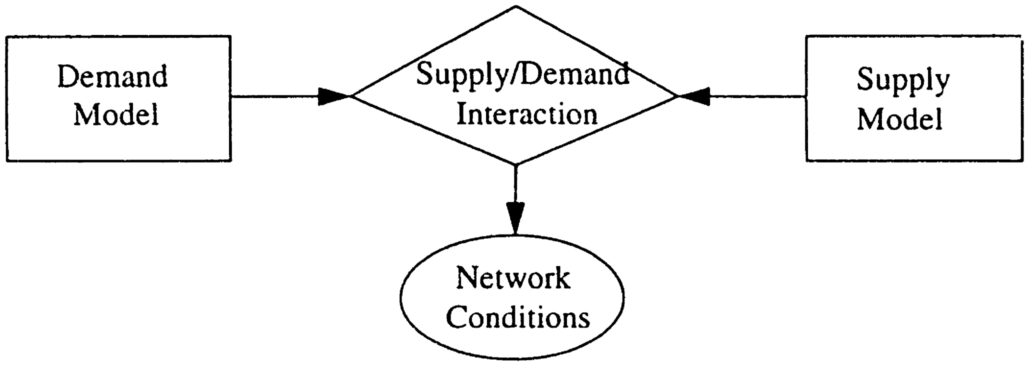

Figure 2-1: A Common Structure of Existing DTA Models

sents the demand for the transportation system. The demand is usually given by a set of time-dependent O-D flows and path flows. The set of O-D flows and path flows generated by the demand model often satisfy certain conditions such as system optimal and user optimal conditions. It should be noted that these two optimal conditions do not generally coincide. To achieve a system optimum, users must behave according to the system optimal conditions instead of following their own behaviors such as departure time choice, mode choice and route choice. On the other hand, if the demand model represents users' behaviors, a user optimum is attained.

The supply model represents the network and the flow progression in the network. A network is a directed and connected graph consisting of links and nodes. A travel cost is associated with each link. The supply model generates the network performance in response to a given demand.

The supply/demand interaction mechanism represents how the supply model and demand model interact. The interaction produces certain network conditions such as link or path flows and travel times. The network conditions must satisfy both the demand model and the supply model.

In the subsequent sections, this common structure is used to review some representative DTA models presented in the literature. The criterion for selecting the representative models is the methodology used to represent the demand, supply and supply/demand interaction.

2.2 Merchant and Nemhauser's Model

The work by Merchant and Nemhauser[31] in 1978 is the first to address the theory of DTA modeling. They considered a directed network consisting of a set of nodes and a set of links. Traffic flows move towards a single destination in the network. A time horizon $[0, T]$ is divided into K time subintervals of equal length. The O-D flows are discrete. They are given for each time period and for every origin node.

Merchant and Nemhauser's (M-N) model is the following constrained optimization problem:

$$\min \sum_k \sum_a X_{a,k}(x_{a,k})$$

subject to

$$\begin{aligned} x_{a,k} - x_{a,k-1} &= u_{a,k} - w_a(x_{a,k-1}) \\ \sum_{a \in A(i)} u_{a,k} - \sum_{a \in B(i)} w_a(x_{a,k-1}) &= p_{i,k} \qquad \forall a \in N - \{q\} \\ \sum_{a \in A(q)} u_{a,k} - \sum_{a \in B(q)} w_a(x_{a,k-1}) + s_k &= 0 \\ u_{a,k} &\geq 0 \\ x_{a,k} &\geq 0 \end{aligned}$$

where

N	=	a set of nodes in the graph
$A(i)$	=	the set of incoming links of node i
$B(i)$	=	the set of outgoing links of node i
q	=	the single destination node
$x_{a,k}$	=	number of vehicles in link a at interval k
$u_{a,k}$	=	number of incoming vehicles at link a at interval k
$w_a(x_a)$	=	exit link function of link a

$p_{i,k}$	=	number of vehicles entering the network at node i at interval k
s_k	=	number of vehicles leaving the network from destination q at interval k
$X_{a,k}(x_{a,k})$	=	cost function for link a at interval k

We analyze the model according to the common structure described before. The users' behaviors are not considered. Since the total system cost is minimized, the resulting network condition is system optimal.

The supply model is formulated as a set of constraints to the minimization problem. The constraints are link-based. The first group of constraints are state equations expressing the conservation of vehicles on link a. The second group of constraints are balance equations of flows at nondestination nodes. The last equation is the balance of flows at the destination node q.

In order to obtain link costs and model flow progression, Merchant and Nemhauser associated a cost function and an exit function with each link. Both are functions of traffic volume on a link at the beginning of a time period. The link cost, obtained through a function for every time interval, is nonnegative, nondecreasing, continuous and convex. It represents the disutility of the link. The link exit function represents the amount of traffic flow that exits from the link. In order to adequately model congestion, this function is assumed to be nondecreasing, continuous and concave.

The supply and demand models interact through the link volumes. However, the interaction mechanism is not explicitly represented.

We now analyze the properties of the solution algorithm for this model. Since the model is nonlinear and nonconvex because of the concavity of the link exit function, the problem may have multiple local optima. Merchant and Nemhauser proposed a solution algorithm based on piecewise linearization and linear programming. The piecewise linear version can be solved for a global optimum by using a one-pass simplex algorithm. Moreover, it has a staircase structure and can be solved by decomposition techniques or compactification methods for sparse matrices. However, this approach requires that a certain ordered set property be satisfied. Because of the nonconvexity of the problem, this requires a potentially expensive computational scheme. The authors were able to show that the simplex algorithm will find the optimal objective function value for their original linear program. However finding the optimal solution will almost always require an additional computational scheme.

The Merchant and Nemhauser's model has a number of limitations. First, in the demand model, the users' behaviors are not modeled. Second, the supply model considers only one destination and treat congestion indirectly. Third, the supply/demand interaction mechanism is not explicitly represented. It is implicitly represented in the solution algorithm. In other words, it relies on a particular algorithm. Therefore, it does not allow for the flexibility in choosing different supply and demand models, as well as solution algorithms.

2.3 Carey's Modified Model

Carey[14] proposed an improved Merchant-Nemhauser's model to obtain a well-behaved convex nonlinear program. He also developed extensions of the model which can handle multiple destinations and multiple commodities, although not all of these extensions yield convex programs. This is done by introducing artificial arcs linking the given destinations to a single artificial destination.

In order to overcome the nonconvexity problem, Carey includes an exit flow variable $v_{a,k}$ for each link a in the formulation as a decision variable. The exit flows on links are nonnegative and are bounded from above by the exit link functions $w_a(x_a)$. This modification leads to a convex nonlinear programming model with substantial advantages. For instance, good solution algorithms are available, since a piecewise linearized version is a standard linear program. This program automatically satisfies the ordered set property. Moreover, there exist both necessary and sufficient conditions to characterize the optimal solution.

However, Carey's model still shares the limitations mentioned for Merchant-Nemhauser's model with respect to the system optimal solution, the simplified traffic propagation modeling, and implicit supply/demand interaction mechanism. Although this reformulation solves the problems due to the nonconvexity of the previous model, it brings the undesirable property known as "FIFO discipline violation". In [15], Carey formulates four classes of constraints that ensure the FIFO discipline of multicommodity flows. Each of the four classes of constraints results in a nonconvex constraint set and nonlinear integer program. Carey suggested an empirical solution as follows: first, solve dynamic assignment problems without constraints to ensure FIFO discipline, then analyze the degree of overtaking or FIFO violations and introduce FIFO constraints if necessary.

2.4 Friesz's Models

Friesz *et al.*[24] reformulated Merchant-Nemhauser's (M-N) model as a continuous time optimal control problem, who considered two demand models, one corresponding to system optimization and the other to a version of user optimization. The system optimal model does not capture users' behaviors. This is a limitation of this model. The user optimal model is reviewed next.

The demand model is formulated as minimizing the summation over all link a, of the integral of link cost function:

$$\min z(x) = \sum_a \int_0^T f(x_a(t))dt$$

The variable t denotes a time instant. The cost function is denoted by $f(x_a(t))$ and represents the instantaneous cost on link a when it contains x users at time t. The function is considered to be a continuous, increasing and convex function.

The significant advantage of Friesz's model over the M-N model is that users' route choice behavior is modeled. This demand model implies that the users' behavior observes a generalized Wardrop's First Principle which is stated as:

If, at each instant of time, for each origin-destination pair, the unit costs of flow on utilized paths are identical and equal to the minimum instantaneous unit path cost, the corresponding flow pattern is said to be user *optimal.*

The supply model is essentially a continuous time version of that in the M-N model. Hence, the analysis is not repeated here.

Again, the supply/demand interaction is through the total link flows. The interaction mechanism is implicitly implied in a minimization process.

There still exist a number of limitations in Friesz' model. The demand model has several drawbacks in considering users' behavior. First, it implies that routing decisions are made on the basis of current, not future, information, although such decisions may be (and are) instantly and continuously altered over time as network conditions change. Consequently, the users with a common destination and departing from the same node at the same time can experience different travel cost to their destination by following different paths. Second, the instantaneous unit path cost is defined as consisting of a static and a dynamic term. The dynamic term is defined as the ratio between the time variation (rate of change) of the Lagrange multiplier, and the derivative of the exit flow function with respect to the link flow state variable. Both the time variation of the Lagrange multiplier and the derivative of the exit flow may change from problem to problem, making it more difficult to find a physical interpretation for them. Third, their analysis is restricted to only one type of users' behavior, that is, the users are homogeneous.

With respect to the supply model, the model shares all the weaknesses of the M-N model since it is just a continuous version of the M-N model.

In a later work, Wie *et al.*[44] extended Friesz's model to multiple destination networks, overcoming the single destination limitation.

2.5 Ran's Model

Ran *et al.*[36] propose a new class of instantaneous user-optimal models. The users' behavior model is refined to correspond to the following user-optimal condition:

The dynamic traffic flow over the network is in a dynamic user optimal state if for each O-D pair at each decision node at each instant of time, *the travel times for all routes that are being used equal the minimum instantaneous route travel time.*

This definition is different from the one given in Friesz *et al.*[24] in that the user can have route choice not only at the first node but also at the intermediate nodes and the optimal condition holds for the intermediate nodes as well.

With respect to the supply modeling, Ran *et al.* add a new constraint called flow propagation constraint for each link. The flow propagation constraint states that vehicles on a link using a given route at any time must result either in added vehicles on a downstream link or the links on a subroute following the arc at a posterior instant of time, or in added exiting vehicles at the destination. Thus, flows on the links are forced to remain on the arc for an amount of time consistent with the link's travel time. This is an improvement in flow propagation modeling with respect to the previous models where an exit flow function is used.

The supply/demand interaction mechanism is implicitly represented in the proposed solution algorithm. The solution algorithm is summarized below.

First the continuous model is reformulated as a discrete-time nonlinear program (NLP). Then, the diagonalization technique and the Frank-Wolfe algorithm are employed to solve the NLP. In the diagonalization procedure, the estimated link travel time is updated iteratively. To apply the Frank-Wolfe algorithm, an expended time-space network is constructed so that each linear program subproblem can be decomposed according to O-D pairs and can be viewed as a set of shortest-path problems. The flow propagation constraints representing the relationship of link flows and travel times are automatically satisfied in modified minimal-cost route searches so that only flow conservation constraints for links and nodes remain.

The weaknesses in the demand model are (1) only one type of users' behavior is studied; (2) the instantaneous travel times are based on current, not future conditions. Hence, route flows with the same departure time and the same origin-destination may actually experience somewhat different route travel times. This is because the route travel time may subsequently change due to changing network traffic conditions over time, even though at each decision node the flows select a route that has a minimum current travel time.

The limitations in their supply model are:

- Link travel time depends on link inflow rate, outflow rate and link volumes. This may be incorrect in congestion situations where flow rate is not a function of travel time only.

- Link exit flows are used as control variables. Queues may not be properly modeled.

Although Ran *et al.* claim that the solution algorithm is efficient to solve the problem for large networks, the expanded time-space network involved is actually much larger than the original network. For instance, if there are k time intervals and the original network contains a links, n nodes and s destination nodes, the expanded time-space network will have $(3a + s)k$ links and $(n + a)k + 1$ nodes. Moreover, the link costs on the expanded network need updating at each iteration. Therefore, the algorithm has the disadvantages in memory usage and running time.

2.6 Smith's Model

Smith[41] proposed a DTA model by using different methods to represent the demand and supply.

The demand model is formulated as an equivalent variational inequality, an equivalent fixed point problem or an equivalent minimization problem. With the fixed point formulation and by showing that the route costs are continuous functions of the route inflows, Smith proved that a dynamic route equilibrium exists.

Smith represent the supply with a macroscopic traffic model. A significant feature of the model is that it uses a traffic model similar to the CONTRAM[34] traffic model to determine the time-varying experienced costs incurred in traversing the various routes when time-varying route inflows are specified. The model is summarized as follows:

Vehicles are grouped together as a packet. A packet is regarded as a set of vehicles that have certain common characteristics. For example, they all follow the same route. They are at the same node at a given time. They must also have experienced the same delays at previous nodes (and so must have all begun their trip at the same previous epoch). But the packet volume will not usually be an integer. The model is capacity constrained, meaning that a node restricts the output from that node due to the exit capacity of the node. In order to exactly fulfill the node exit capacities, it is also often necessary to split a packet, so part of it proceeds while the remaining part stays at the node. Splitted packets follow the same route. The splitting of packets means that different parts of the same original packet will often reach the destination at different times. Priorities are introduced so as to place packets of traffic queuing at the same node at the same time in an appropriate order. As time proceeds, packet priorities are adjusted so as to ensure that, as traffic traverses the network, there is no overtaking. These priority rules require that the complete history of each packet be retained. Time is discretized and its unit is defined as the free-flow travel time taken to traverse each unit link. The packets are moved through the network time-epoch by time-epoch and the average route costs are computed for each route and departure time.

This supply model has a number of advantages over other DTA models:

- The link exit capacity and FIFO condition are respected.
- The path travel time is the experienced travel time by a user from origin to destination, instead of the instantaneous travel time based on current network conditions.

Again, the supply/demand interaction is through experienced path travel time. The interaction mechanism is implicitly represented in a solution algorithm which is based on an optimization formulation of the problem.

Smith's model also has a number of drawbacks. For the demand model, only one type of users' behavior is modeled. For the supply model, the weaknesses are:

- It is storage demanding to record the history of each packet in order to determine priorities.

- Packet splitting will make packet size smaller and the number of packets in the network larger, which results in increasing memory and computing time requirements.

- The only delays considered by the model are queuing delays. Link travel times do not depend on the amount of vehicles traversing the link.

- The uncongested travel times along all links in the model are assumed to be identical. In order to deal with real networks, long traffic lanes should be regarded as a contiguous sequence of traffic lanes of a short standard unit length. This will create a bunch of intermediate artificial nodes which are assigned a very large exit capacity. This representation increases the number of nodes and arcs in the network. Hence, the computational effort increases accordingly.

2.7 Summary

Several existing analytical DTA models have been reviewed in this chapter. A common structure exists in those models, that is, they all have a demand model simulating users' behaviors, a supply model simulating network performance, and a supply/demand interaction. Various models use different methods to model the supply, demand and supply/demand interaction.

Most models, called user-optimal models, consider only one type of users' behavior, that is, users choose only minimum-cost paths. The definition of path cost varies in whether it is instantaneous (based on current information) or experienced (based on predictive information).

Most supply models are formulated as a system of equations that express link dynamics, flow conservation and flow propagation constraints. However, these models do not respect queues and do not observe link capacities directly. Although Smith's model aims to overcome these limitations, it incurs the problem of computational inefficiency.

Various solution algorithms have been developed. However, implementations of the algorithms are not discussed and most numerical examples were done on very small networks only.

Chapter 3

Formulation of a Flow-Based Dynamic Traffic Assignment Model

As reviewed in Chapter 2, there exist different approaches to the Dynamic Traffic Assignment (DTA) problem. The objectives of this chapter are (1) to propose a flow-based modeling framework for the DTA problem, and (2) to formulate a DTA model based on the proposed framework to be used within the context of ATMS/ATIS.

The modeling framework contains four components: (1) users' behavior model, (2) dynamic network loading model (3) link performance model, and (4) path generation module. Each of these components can be formulated independent of each other. The interaction between these models is explicitly represented.

The framework has a modularized structure. It has helped us to understand existing DTA models in the literature (see Chapter 2). In this chapter and in the following chapters, we will see that this framework also provides better flexibility in model formulations, algorithms development and computer implementations.

A DTA model is formulated based on this framework. The model aims to overcome some limitations present in existing models within the context of ATMS/ATIS. The model captures different users' route choice behaviors. Link performance is modeled more accurately by using different link travel time functions for freeway links and arterial street links.

This chapter is organized as follows: Section 3.1 presents the overall modeling framework for the DTA problem. Section 3.2 defines basic notations used in the formulation. In section 3.3, the formulation of users' behavior model is presented. Section 3.4 discusses link travel time models. A dynamic network loading model is presented in Section 3.5. The existence of a solution to the model is shown in Subsection 3.5.2. The computation of path travel times from link travel times is described in Section 3.6. The DTA model is given in Section 3.7.

3.1 A Framework for the Dynamic Traffic Assignment Problem

A modeling framework for the dynamic traffic assignment problem is shown in Figure 3-1. The framework contains the following components:

- a users' behavior model component,
- a dynamic network loading model component, and
- a link performance model component.

The function of each model and the interaction between these models are described next.

The users' behavior model component takes as input the dynamic O-D trips and a subset of paths between each O-D pair. The dynamic O-D trips are the time-dependent traffic demand for each O-D pair. In the continuous time horizon, the dynamic O-D trips are given as departure flow rates at each origin and each time instant. In discrete time representation, they are given as number of trips during a time interval. These dynamic O-D trips can be predicted and are treated in the DTA model as input.

The subset of paths between each O-D pair is assumed to be the set of routes from which the users choose when they depart from their origins. These subsets of paths

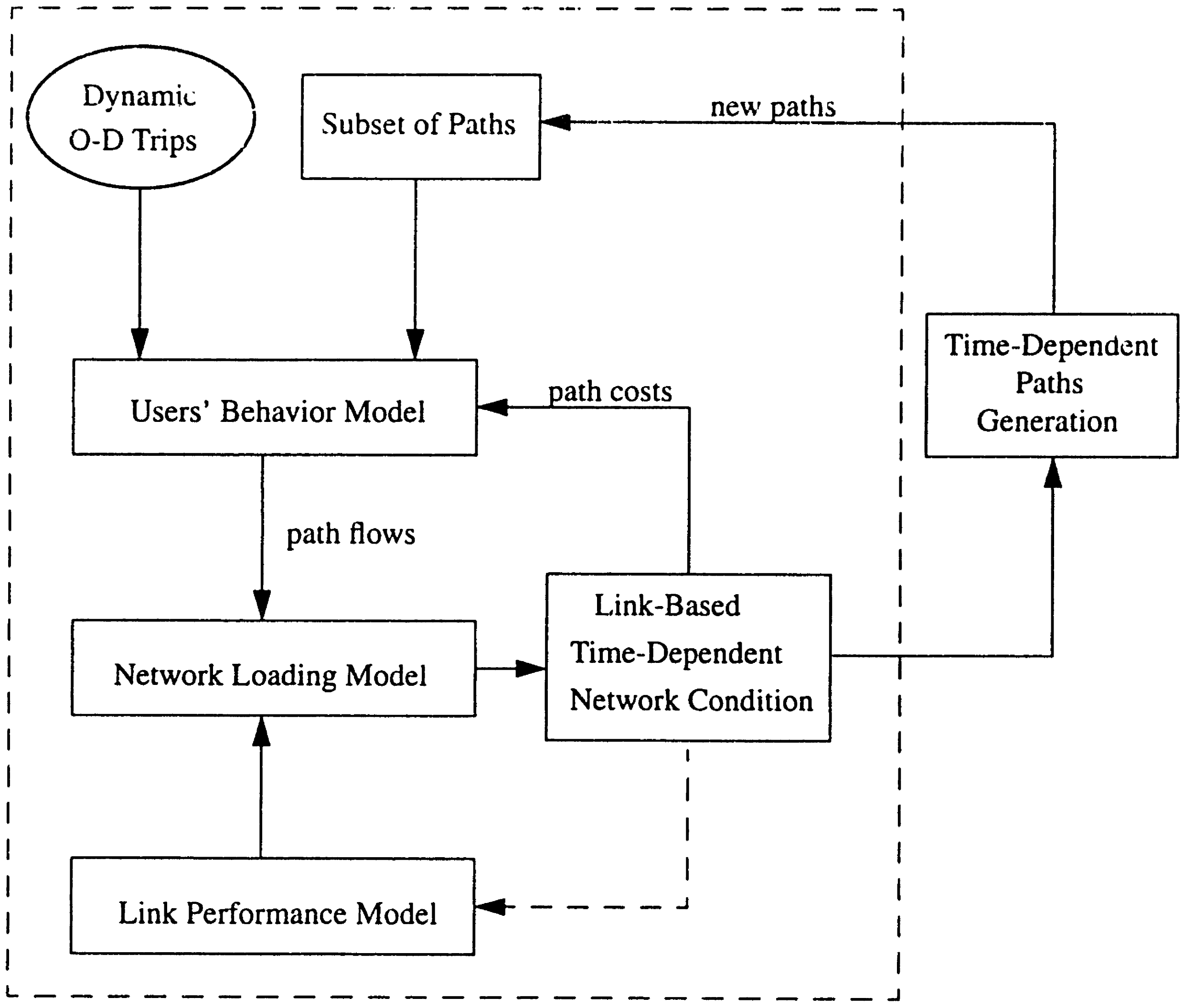

Figure 3-1: A Framework for Dynamic Traffic Assignment Models

can be dynamically augmented by using a path generation module based on certain criteria. The path generation module monitors the network conditions and adds new paths to the subset of paths according to certain criteria. The users' behavior model component then assigns the dynamic O-D trips among the subset of paths according to the users' route choice behaviors. This results in a set of time-dependent path flows.

The network loading model takes the path flows from the users' behavior model as input and uses link performance models to generate the resulting link-based network conditions such as time-dependent link volumes and link travel times. The link-based network conditions serve two purposes. First, they are used to compute path travel times. The path travel times are then used by the users' behavior model to assign O-D trips. Second, the network conditions are input to the path generation module to come up with a subset of new paths for each O-D pair.

Clearly, the framework has a modularized structure. The components interrelate through specified inputs and outputs. The framework provides flexibility in both model formulations and computer implementations because one model can be changed without affecting others.

The common structure in existing DTA models presented in Chapter 2 can be viewed as a high-level abstraction of this framework. The users' behavior model corresponds to the demand model in the common structure, the dynamic network loading model and link performance model together correspond to the supply model. The interaction between the three model components in the framework represents a supply/demand interaction mechanism.

In the subsequent sections, a DTA model is formulated based on this framework. The path generation module is not considered in the current development; thus the subset of paths between each O-D pair are assumed to be fixed.

3.2 Notations

The physical traffic network is represented by a conceptual directed network $G = (N, A)$, where N is the set of nodes and A is the set of directed links. In the following, the index m denotes a user type, the index r denotes an origin node, the index s denotes a destination node and the index p denotes a path between O-D pair (r, s). The subset of paths between O-D pair (r, s) is denoted by K_{rs}. All other notations are grouped into path variables, link variables and link-path flow variables for each pair (a, p) and time t.

Path variables:

$f_{mp}^{rs}(t)$	:	type m departure flow rate on path p from origin r toward destination s at time t
$f_p^{rs}(t)$	=	$\sum_m f_{mp}^{rs}(t)$, the departure flow rate of type m for O-D pair (r, s) at time t
$f_m^{rs}(t)$	=	$\sum_{p \in K_{rs}} f_{mp}^{rs}(t)$, the departure flow rate of type m for O-D pair (r, s) at time t
$c_p^{rs}(t)$	:	the experienced travel time over path p from origin r toward destination s by the flow departing at time t
$\pi_p^{rs}(t)$	=	$\min_{p \in K_{rs}}\{c_p^{rs}(t)\}$, minimum experienced travel time over path p from origin r toward destination s by the flow departing at time t
$P_p^{rs}(t)$	:	the proportion by which type 2 O-D departure flow is assigned to path p of O-D pair (r, s)

Link variables:

$U_a(t)$	:	total cumulative entrance flow
$V_a(t)$	:	total cumulative exit flow
$X_a(t)$	:	load of link a at time t
$\tau_a(t)$	:	travel time over link a for flows entering link a at time t
$s_a(t)$	:	exit time from link a for flows entering link a at time t

Link-path flow variables:

$u_{ap}^{rs}(t)$	:	entrance flow rate at time t
$v_{ap}^{rs}(t)$	:	exit flow rate at time t
$U_{ap}^{rs}(t)$	:	cumulative entrance flow at time t
$V_{ap}^{rs}(t)$	:	cumulative exit flow at time t
$X_{ap}^{rs}(t)$	:	cumulative link flow induced by path p flow at time t

Time variables:

t	:	index for continuous time
$[0, T_d]$	:	O-D traffic demand period
$[0, T]$	:	DTA analysis period (the period from the time when flows enter the network to the time all flows exit the network)
Δ	:	minimum free flow link travel time over all links
δ	=	$\frac{\Delta}{M}$, M is a positive integer
k	:	index for time interval $[(k-1)\delta, k\delta]$

3.3 The Users' Behavior Model

As seen in Chapter 2, existing DTA models either do not model users' behaviors or if they do, they assume that all users have homogeneous route choice behavior. For example, they assume that all users choose the route with minimum cost. This assumption is not generally realistic since (1) not all users have perfect information, and (2) even if they do have perfect information, users may not fully comply with this information. In either case, users may have different route choice behaviors. Therefore, it is necessary to have a model that takes into account differences in users' behaviors in order to study the impact of information on network conditions.

One of the objectives of our DTA model is to overcome the limitation of existing models in modeling users' behaviors. To achieve this objective, we study three representative types of users' route choice behaviors in the context of ATMS/ATIS. They are described in the following subsections.

It should be noted that users' behaviors are not restricted to route choice. Departure time choice and mode choice are two other examples of users' behaviors. Modeling these two behaviors is not considered in this thesis.

3.3.1 Classification of Users

The users are classified into the following three representative types based on their route choice behaviors:

type 1: users who follow fixed routes

This type of users can be described as those who either do not have real-time traffic information (unguided) and use their habitual routes, or those who disregard the information and continue to use their habitual routes. Therefore, the departure flow of each path is known.

type 2: users who follow routes with minimum perceived travel time

This type of users can be used to describe users who receive or have partial traffic information about the network conditions and determine their routes based on their "perceived" rather than actual travel times. Thus each users' perceived travel time is a random variable with certain distribution.

type 3: users who choose routes with minimum actual travel time

This type of users are those who have access to real-time traffic information, and fully comply with the route guidance. Therefore, the routes used by this type of users have minimum actual travel time.

Type 3 users can be seen as a special case of type 2 users. If the travel time perception error by a type 2 user approaches zero, type 2 users are identical to type 3 users in route choice behavior. Nevertheless, we differentiate them for two purposes: (1) to cover the general situation where both types of users are present, (2) to improve computational efficiency, since all type 3 O-D flows are simply assigned to the shortest paths.

3.3.2 Dynamic Route Choice Conditions

In this subsection, the three types of users' route choice behaviors are expressed in equivalent mathematical forms. These are called dynamic route choice conditions.

Regardless of the network condition, the route choice of type 1 users is fixed and their path flows are also known. Therefore, their route choice behavior can be expressed as follows:

$$f_{1p}^{rs^*}(t) - f_{1p}^{rs}(t) = 0, \qquad \forall (r,s), \forall p \in K_{rs}, \forall t \in [0, T_d]. \tag{3.1}$$

Superscript * denotes the optimum value.

For class 2 users, their route choice behavior is equivalent to the following route choice condition:

> *For each O-D pair (r, s) at any time t, the perceived experienced travel time of a path that is chosen equals the minimum perceived experienced travel time.*

The definition of *experienced* path travel time is given in Section 3.6.

A type 2 user's perceived travel time is a random variable with a certain distribution. The probability that the path is chosen by a type 2 user among the set of available paths is equal to the probability that path p is perceived as minimum. If we assume that all type 2 users are homogeneous (i.e., their perceived path travel times are independent, identically distributed random variables), then for each O-D pair (r, s), the flow on path p is equal to product of the total O-D flow and the probability that path p is chosen by a type 2 users, that is,

$$f_{2p}^{rs}(t) \quad = \quad f_2^{rs}(t) \times \text{probability that path } p \text{ is chosen.}$$

Therefore, the probability that path p is chosen is equal to the proportion $P_p^{rs}(t)$ by which the O-D flow is assigned to path p. This route choice condition can then be expressed as the following equation:

$$f_{2p}^{rs^*}(t) - f_2^{rs}(t) P_p^{rs}(t) = 0, \qquad \forall (r,s), \forall p \in K_{rs}, \forall t \in [0, T_d]. \tag{3.2}$$

For type 3 users, they all choose the routes that have minimum travel time. Therefore, their route choice behavior is equivalent to the following condition:

For each O-D pair at each instant of time, the experienced travel time of the used paths by the users departing at the same time are equal and minimal.

This is the dynamic generalization of the conventional static user-optimal condition with path travel time defined as experienced (actual) travel time. The condition can be written in mathematical forms as follows:

$$c_p^{rs^*}(t) - \pi^{rs^*}(t) \geq 0, \tag{3.3}$$

$$f_{3p}^{rs^*}(t)[c_p^{rs^*}(t) - \pi^{rs^*}(t)] = 0, \tag{3.4}$$

$$f_{3p}^{rs^*}(t) \geq 0, \tag{3.5}$$

$$\forall (r,s), \forall p \in K_{rs}, \forall t \in [0, T_d].$$

3.3.3 Formulation of the Users' Route Choice Behavior Model

The users' route choice behavior model is formulated as an equivalent Variational Inequality (VI) problem, which is stated in the following theorem:

Theorem 3.1 *The three types of users' route choice behaviors can be modeled by the following equivalent VI problem:*

$$\int_0^{T_d} \sum_{rs} \sum_{p \in K_{rs}} \{F_{2p}^{rs^*}(t)[f_{2p}^{rs}(t) - f_{2p}^{rs^*}(t)] + c_p^{rs^*}(t)[f_{3p}^{rs}(t) - f_{3p}^{rs^*}(t)]\}dt \geq 0, \tag{3.6}$$

where

$$F_p^{rs^*}(t) = [f_{2p}^{rs^*}(t) - f_2^{rs}(t) P_p^{rs}(t)] \frac{\partial c_p^{rs}(t)}{\partial f_{2p}^{rs}(t)}.$$

Proof:

We will prove the theorem by showing the necessity and sufficiency of VI (3.6).

Proof of necessity:

The idea to prove necessity is to establish VI (3.6) from the dynamic route choice conditions (3.1)–(3.5). From (3.2), it follows that

$$F_p^{rs^*}(t) = 0, \qquad \forall (r,s), \forall p \in K_{rs}, \forall t \in [0, T_d],$$

and

$$\sum_{rs} \sum_{p \in K_{rs}} \{F_{2p}^{rs^*}(t)[f_{2p}^{rs}(t) - f_{2p}^{rs^*}(t)]\} = 0, \qquad \forall t \in [0, T_d].$$

Integrating the above inequality over $t \in [0, T_d]$ gives

$$\int_0^{T_d} \sum_{rs} \sum_{p \in K_{rs}} \{F_{2p}^{rs^*}(t)[f_{2p}^{rs}(t) - f_{2p}^{rs^*}(t)]\} dt = 0. \tag{3.7}$$

The route choice conditions (3.3)–(3.5) for type 3 users imply that

$$[c_p^{rs^*}(t) - \pi^{rs^*}(t)][f_{3p}^{rs}(t) - f_{3p}^{rs^*}(t)] \geq 0, \qquad \forall (r,s), \forall p \in K_{rs}, \forall t \in [0, T_d]. \tag{3.8}$$

Since if $f_{3p}^{rs^*}(t) = 0$, $c_p^{rs^*}(t) - \pi^{rs^*}(t) \geq 0$ and $f_{3p}^{rs}(t) \geq 0$, (3.8) holds. On the other hand, if $f_{3p}^{rs^*}(t) > 0$, $c_p^{rs^*}(t) - \pi^{rs^*}(t) = 0$ and (3.8) still holds.

Inequality (3.8) can be expanded as follows:

$$c_p^{rs^*}(t)[f_{3p}^{rs}(t) - f_{3p}^{rs^*}(t)] - \pi^{rs^*}(t)[f_{3p}^{rs}(t) - f_{3p}^{rs^*}(t)] \geq 0.$$

Summing both sides of the above inequality over (r,s) and $p \in K_{rs}$ gives

$$\sum_{rs} \sum_{p \in K_{rs}} \{c_p^{rs^*}(t)[f_{3p}^{rs}(t) - f_{3p}^{rs^*}(t)]\} - \sum_{rs} \sum_{p \in K_{rs}} \{\pi^{rs^*}(t)[f_{3p}^{rs}(t) - f_{3p}^{rs^*}(t)]\} \geq 0, \quad \forall t.$$

The second term on the left-hand side vanishes because:

$$\begin{aligned} \sum_{rs} \sum_{p \in K_{rs}} \{\pi^{rs^*}(t)[f_{3p}^{rs}(t) - f_{3p}^{rs^*}(t)]\} &= \sum_{rs} \pi^{rs^*}(t) \sum_{p \in K_{rs}} [f_{3p}^{rs}(t) - f_{3p}^{rs^*}(t)] \\ &= \sum_{rs} \pi^{rs^*}(t)[f_3^{rs}(t) - f_3^{rs}(t)] = 0. \end{aligned}$$

Integrating the above inequality over $t \in [0, T_d]$ gives

$$\int_0^{T_d} \sum_{rs} \sum_{p \in K_{rs}} \{c_p^{rs^*}(t)[f_{3p}^{rs}(t) - f_{3p}^{rs^*}(t)]\} dt \geq 0. \tag{3.9}$$

Combining (3.7) and (3.9) yields VI (3.6). Therefore, the dynamic route choice conditions imply VI (3.6). The necessity is proved.

Proof of sufficiency:

The sufficiency requires that any solution to VI (3.6) satisfies the dynamic route choice conditions.

Suppose that $\{f_{2p}^{rs^*}(t)\}$ and $\{f_{3p}^{rs^*}(t)\}$ are solutions of VI(3.6). We will show by contradiction that $\{f_{2p}^{rs^*}(t)\}$ and $\{f_{3p}^{rs^*}(t)\}$ must satisfy the route choice conditions for type 2 and type 3 users, respectively.

First, assume that the route choice condition for type 2 users do not hold at time t_1. Because of the continuity of the variables, if the route choice conditions do not hold at time instant t_1, it will not hold within a vicinity of t_1. Therefore, the assumption implies that there exists a set of routes $S_{mn}(t)$ of O-D pair (m, n) during time interval $[t_1 - \delta, t_1 + \delta]$ such that

$$f_{2l}^{mn^*}(t) = f_2^{rs}(t) P_l^{rs}(t) + \epsilon_l(t) \quad \text{and} \quad \epsilon_l(t) \neq 0, \quad \forall l \in S_{mn}(t), \forall t \in [t_1 - \delta, t_1 + \delta].$$

Note that for any path $p \notin S_{mn}(t)$, we can choose $f_{2p}^{rs}(t) = f_{2p}^{rs^*}(t)$ so that the left-hand side of VI (3.6) can be reduced as

$$\begin{aligned} &\int_0^{T_d} \sum_{rs} \sum_{p \in K_{rs}} \{F_{2p}^{rs^*}(t)[f_{2p}^{rs}(t) - f_{2p}^{rs^*}(t)] + c_p^{rs^*}(t)[f_{3p}^{rs}(t) - f_{3p}^{rs^*}(t)]\} dt \\ = &\int_{t_1-\delta}^{t_1+\delta} \sum_{l \in S_{mn}(t)} \{\epsilon_l(t) \frac{\partial c_l^{mn}(t)}{\partial f_{2l}^{mn}(t)} [f_{2l}^{mn}(t) - f_2^{rs}(t) P_l^{rs}(t) - \epsilon_l(t)]\} dt. \end{aligned}$$

Since the above inequality holds for all feasible $f_{2l}^{mn}(t)$, we can choose $f_{2l}^{mn}(t)$ as follows:

$$f_{2l}^{mn}(t) = f_2^{mn}(t) P_l^{mn}(t), \qquad \forall t \in [t_1 - \delta, t_1 + \delta].$$

Note that these chosen $f_{2l}^{mn}(t)$ are feasible since $\sum_{l\in S_{mn}(t)} f_{2l}^{mn}(t) + \sum_{l\notin S_{mn}(t)} f_{2l}^{mn}(t) = f_2^{mn}(t)$. Consequently,

$$\begin{aligned}
&\int_{t_1-\delta}^{t_1+\delta} \sum_{l\in S_{mn}(t)} \{\epsilon(t)\frac{\partial c_l^{mn}(t)}{\partial f_{2l}^{mn}(t)}[f_{2l}^{mn}(t) - f_2^{mn}(t)P_l^{mn}(t) - \epsilon(t)]\}dt \\
= &\int_{t_1-\delta}^{t_1+\delta} \sum_{l\in S_{mn}(t)} (-\epsilon^2(t))\frac{\partial c_l^{mn}(t)}{\partial f_{2l}^{mn}(t)}dt.
\end{aligned}$$

It is assumed that the mean experienced route travel time $c_p^{rs}(t)$ is increasing with path departure flow $f_{2p}^{rs}(t)$ of type 2 users. Hence,

$$\frac{\partial c_p^{rs}(t)}{\partial f_{2p}^{rs}(t)} > 0, \qquad \forall(r,s), \forall p \in K_{rs}, \forall t \in [0, T_d].$$

It follows that

$$\int_{t_1-\delta}^{t_1+\delta} \sum_{l\in S_{mn}(t)} (-\epsilon^2(t))\frac{\partial c_l^{mn}(t)}{\partial f_{2l}^{mn}(t)}dt < 0.$$

This contradicts VI (3.6). The contradiction implies that the solution of VI (3.6) for type 2 users must satisfy that route choice conditions for type 2 users.

Next, we prove by contradiction that solution $f_{3p}^{rs^*}(t)$ to the VI satisfies the route choice condition for type 3 users. Suppose that the route choice condition for type 3 users does not hold. Then there exists a route l of O-D pair (m,n) during time interval $[t_1-\delta, t_1+\delta]$ such that

$$f_{3l}^{mn^*}(t) > 0 \quad \text{and} \quad c_l^{mn^*}(t) - \pi^{mn^*}(t) > 0, \quad \forall t \in [t_1-\delta, t_1+\delta].$$

Since l is not the shortest path, l is not the only path in K_{mn}. Thus, there must be a path $k(t) \neq l$ such that

$$f_{3k(t)}^{mn^*}(t) \geq 0 \quad \text{and} \quad c_{k(t)}^{mn^*}(t) - \pi^{mn^*}(t) = 0, \quad \forall t \in [t_1-\delta, t_1+\delta].$$

Then the left-hand side of VI (3.6) can be written as

$$\int_0^{T_d} \sum_{rs} \sum_{p \in K_{rs}} \{F_{2p}^{rs^*}(t)[f_{2p}^{rs}(t) - f_{2p}^{rs^*}(t)] + c_p^{rs^*}(t)[f_{3p}^{rs}(t) - f_{3p}^{rs^*}(t)]\}dt$$
$$= \int_{t_1-\delta}^{t_1+\delta} \{c_l^{rs}(t)[f_{3l}^{mn}(t) - f_{3l}^{mn^*}(t)] + \pi^{mn^*}(t)[f_{3k(t)}^{mn}(t) - f_{3k(t)}^{mn^*}(t)]\}dt.$$

Since $f_{3l}^{mn^*}(t) > 0$ by assumption, we can shift a small amount of flow from path l to path $k(t)$, that is, let

$$\begin{aligned} f_{3l}^{mn}(t) &= f_{3l}^{mn^*}(t) - \epsilon(t), & \forall t \in [t_1 - \delta, t_1 + \delta]. \\ f_{3k(t)}^{mn}(t) &= f_{3k(t)}^{mn^*}(t) + \epsilon(t), & \forall t \in [t_1 - \delta, t_1 + \delta], \end{aligned}$$

where $\epsilon(t) > 0$.

Consequently, we have

$$\int_{t_1-\delta}^{t_1+\delta} \{c_l^{mn^*}(t)[f_{3l}^{mn}(t) - f_{3l}^{mn^*}(t)] + \pi^{mn^*}(t)[f_{3k(t)}^{mn}(t) - f_{3k(t)}^{mn^*}(t)]\}dt$$
$$= \int_{t_1-\delta}^{t_1+\delta} \{c_l^{mn^*}(t)[f_{3l}^{mn^*}(t) - \epsilon(t) - f_{3l}^{mn^*}(t)] + \pi^{mn^*}(t)[f_{3k(t)}^{mn^*}(t) + \epsilon(t) - f_{3k(t)}^{mn^*}(t)]\}dt$$
$$= \int_{t_1-\delta}^{t_1+\delta} (-\epsilon(t))[c_l^{mn^*}(t) - \pi^{mn^*}(t)]dt.$$

Since both $\epsilon(t) > 0$ and $c_l^{mn^*}(t) - \pi^{mn^*}(t) > 0$ for all $t \in [t_1 - \delta, t_1 + \delta]$, we have

$$\int_{t_1-\delta}^{t_1+\delta} (-\epsilon(t))[c_l^{mn^*}(t) - \pi^{mn^*}(t)]dt < 0.$$

This contradicts VI (3.6). Therefore, any solution $\{f_{3p}^{rs^*}(t)\}$ of the VI must satisfy the route choice condition for type 3 users.

The necessity and sufficiency proofs conclude that VI (3.6) is equivalent to the dynamic route choice conditions.

3.4 The Link Performance Models

In DTA models, link performance models are needed in order for the dynamic network loading model to generate network conditions. Measures of link performance can be link travel time, out-of-pocket cost, etc. In this thesis, we use link travel time as the measure of link performance because travel time is an important factor when users choose a route. In this context, we can call link performance models as link travel time models. Travel time models are often expressed in certain functional forms called link travel time functions.

The objective of this section is to present appropriate link travel time models to be used by the dynamic network loading model. The volume-delay functions used in static traffic assignment express link travel time as a function of flow rate. This is not correct in dynamic situation. In this section, We first show the invalidity of the volume-delay functions in dynamic traffic assignment. With this conclusion, we will use density instead of flow rate to determine link travel times if the traffic flow is uninterrupted. In the case that traffic flow is interrupted, we present a queuing model to determine the link travel times. Finally, we discuss the Fist-In-First-Out condition and its implication on link travel time functions.

3.4.1 Invalidity of Volume-Delay Functions in Dynamic Traffic Assignment

Figure 3-2 depicts the fundamental relationship between travel speed and flow rate on a freeway link observed from traffic engineering experience. The figure shows that there are two domains for the speed-flow rate relationship. The first domain, indicated by the unshaded area in the figure, describes an uncongested situation. In this domain, travel speed decreases with increasing flow rate. This relationship continues until the flow rate reaches a maximum value.

The second domain, indicated by the shaded area, describes a congested situation. In this domain, the flow rate decreases while speed continues to decrease.

With regard to link travel time, Figure 3-3 shows that the curve of true relationship

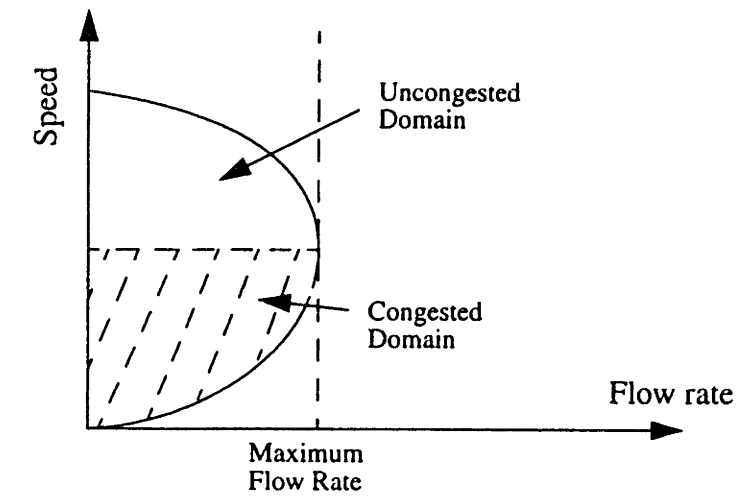

Figure 3-2: Relationship between Speed and Flow Rate

between link travel time and flow rate resembles the curve in Figure 3-2. However,

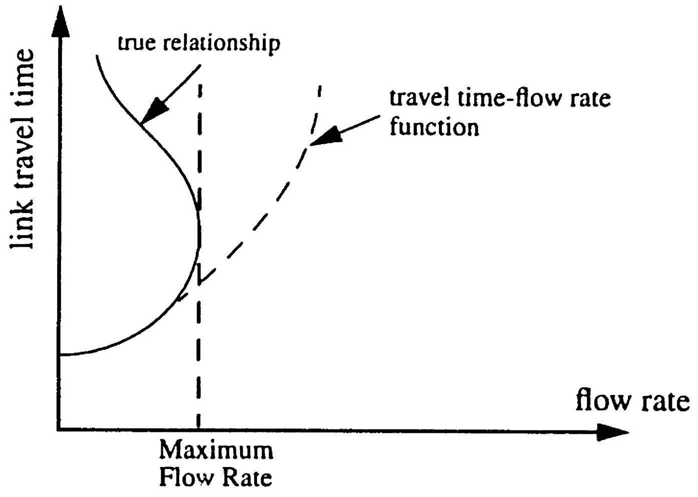

Figure 3-3: True Travel Time-Flow Rate Relationship and Travel Time-Flow Rate Function

one flow rate value could correspond to more than one link travel time. This means travel time is not a function of flow rate. Therefore, volume-delay functions which use flow rate as explanatory variable are unsuitable for describing both uncongested and congested traffic conditions.

In the following subsection, we will present a link travel time function which use density or link load as explanatory variable.

3.4.2 A Link Travel Time Model for Uninterrupted Traffic

Unlike flow rate, density can be used as an explanatory variable in a function to determine link travel time in both uncongested and congested situations. Figure 3-4 depicts the relationship between speed and density as well as the relationship between travel time and density when the traffic is uninterrupted. The figure shows

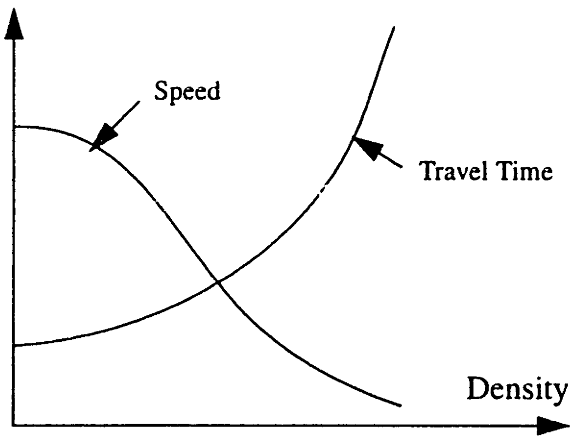

Figure 3-4: Speed and Travel Time as a Function of Density

that speed decreases as density increases. Consequently, travel time increases as density increases. Furthermore, one value of density corresponds to only one value of speed or link travel time. Therefore, speed or travel time can be expressed as a function of density.

We use a modified Greenshields' speed-density relationship to determine the travel time for uninterrupted traffic. An example of uninterrupted flow is the flow on freeway links with no incident. The speed-density relationship is expressed as follows:

$$w_a(t) = w_a^{min} + (w_a^{max} - w_a^{min})[1 - (\frac{k_a(t)}{k_{aj}})^{\alpha}]^{\beta},$$

where

$$
\begin{aligned}
w_a(t) &= \text{speed on link } a \text{ at time } t, \\
k_a(t) &= \text{density on link } a \text{ at time } t, \\
w_a^{min}, w_a^{max} &= \text{minimum and free-flow travel speed,} \\
k_{aj} &= \text{jam density, and} \\
\alpha, \beta &= \text{model parameters.}
\end{aligned}
$$

A minimum travel speed w_a^{min} is imposed in order to prevent flows from stopping on the link.

After travel speed is determined, the link travel time is given by

$$\tau_a(t) = \frac{L_a}{w_a(t)} = \frac{L_a}{w_a^{min} + (w_a^{max} - w_a^{min})[1 - (\frac{k_a(t)}{k_{aj}})^\alpha]^\beta}, \tag{3.10}$$

where L_a is the length of the link.

Since density $k_a(t)$ can be calculated from link load (number of vehicles on the link), that is,

$$k_a(t) = \frac{X_a(t)}{L_a},$$

the link travel time can also be expressed as a function of link load $X_a(t)$:

$$\tau_a(t) = \frac{L_a}{w_a^{min} + (w_a^{max} - w_a^{min})[1 - (\frac{X_a(t)}{L_a k_{aj}})^\alpha]^\beta}. \tag{3.11}$$

In this expression, $L_a k_{aj}$ is the maximum link load (link capacity). The model parameters α and β need to be calibrated for each link.

Figure 3-5 depicts an example of the link travel time function for $\alpha = 1.4$ and $\beta = 3.2$. These values are also used for the two case studies presented in Chapter 6. The figure shows that link travel time increases with link load. The travel time increases more rapidly after the link load reaches about 40% of link capacity. It approaches a maximum value when the link load is close to the capacity.

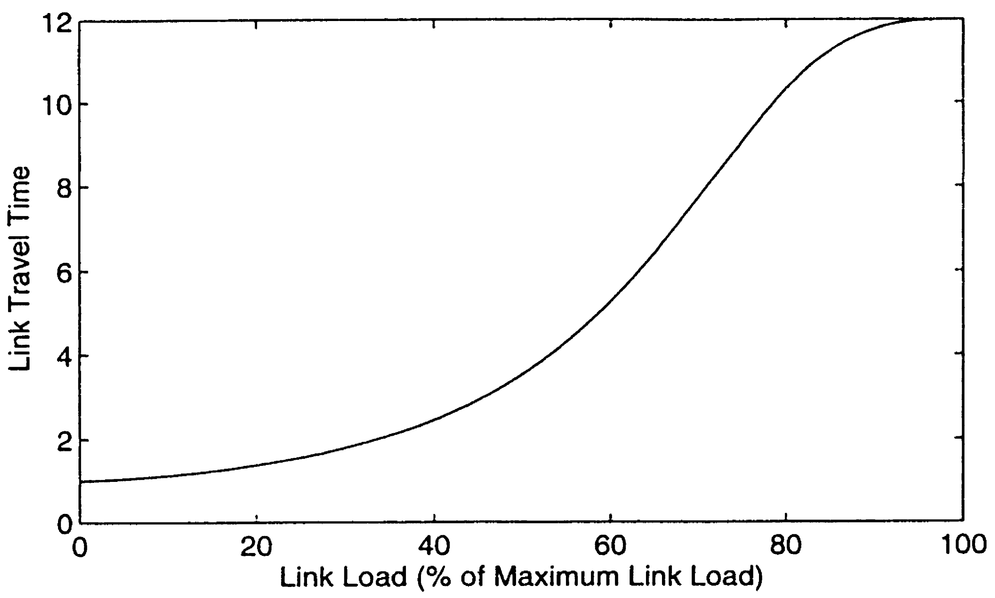

Figure 3-5: An Example of the Link Travel Time Function for Freeway Links

3.4.3 A Link Travel Time Model for Interrupted Traffic

Interrupted traffic due to signal settings or incidents often results in queues on a link. The queuing delay cannot be determined by using the speed-density relationships only. Therefore, we present a link travel time model which captures the queuing delay on a link for interrupted flow.

The travel time for interrupted flow can be decomposed into two components: (1) moving time, and (2) queuing delay. That is,

$$\tau_a(t) = \tau_a^m(t) + \tau_a^q(t)$$

where $\tau_a^m(t)$ denotes the moving time and $\tau_a^q(t)$ denotes the queuing delay. To compute these two travel time components, the link is divided into two parts as show in Figure 3-6.

To derive the link travel time, we define the following additional notations:

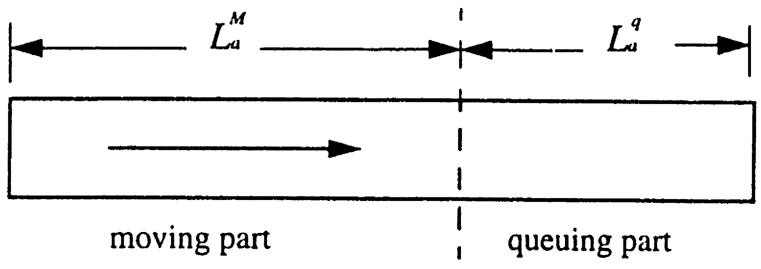

Figure 3-6: Conceptual Moving Part and Queuing Part on an Arterial Link

$X_a^q(t)$	:	the number of vehicles in the queuing part when a user enters the link at time t,
$L_a^q(t)$	:	the queue length in miles when a user enters the link at time t,
$k_a^m(t)$	:	density on the moving part when a user enters the link at time t, and
Q_a	:	average exit flow rate out of link a.

For the queuing part of the link, assume that vehicles are closely lined up in a bumper-to-bumper condition where the density is assumed to be equal to the jam density. Thus, the queuing length is given by:

$$L_a^q(t) = \frac{X_a^q(t)}{k_{aj}}.$$

Then the density on the moving part is

$$k_a^m(t) = \frac{X_a(t) - X_a^q(t)}{L_a - L_a^q(t)}.$$

The density on the moving part is used to determine the moving speed $w_a(t)$ as follows:

$$w_a(t) = w_a^{min} + (w_a^{max} - w_a^{min})[1 - (\frac{k_a^m(t)}{k_{aj}})^{\alpha}]^{\beta}. \tag{3.12}$$

After knowing this moving speed, the moving time and the queuing delay can be determined. The queuing delay is given by

$$\tau_a^q(t) = \max\{0, [X_a(t) - \tau_a^m(t)Q_a]/Q_a\}.$$

If there is no queuing delay, the moving part is the entire link, the travel time is:

$$\tau_a(t) = \tau_a^m(t) = \frac{L_a}{w_a(t)}.$$

If there is a queuing delay, the moving part is not the entire. Then the travel time is:

$$\begin{aligned}\tau_a(t) &= \tau_a^m(t) + \tau_a^q(t) \\ &= \tau_a^m(t) + [X_a(t) - \tau_a^m(t)Q_a]/Q_a \\ &= X_a(t)/Q_a.\end{aligned}$$

In summary, the link travel time $\tau_a(t)$ is determined as follows:

$$\tau_a(t) = \begin{cases} \frac{L_a}{w_a(t)} & \text{, if } X_a(t) - \frac{L_a}{w_a(t)}Q_a \leq 0 \\ X_a(t)/Q_a & \text{, otherwise} \end{cases} \tag{3.13}$$

where $w_a(t)$ is computed by using (3.12).

After we compute $\tau_a(t)$, we can also determine the number of vehicles in queue when the user entering the link at time t reaches the tail of queue. This value, denoted by $X_a^q(t + \tau_a^m(t))$, will be used to compute the travel time for the user who enters the link at time $t + \tau_a^m(t)$.

$X_a^q(t + \tau_a^m(t))$ can be determined by solving the following two equations:

$$X_a^q(t + \tau_a^m(t)) = X_a^q(t) - \tau_a^m(t)Q_a \tag{3.14}$$

$$\tau_a^m(t) = \frac{L_a - X_a^q(t + \tau_a^m(t))/k_{aj}}{w_a(t)} \tag{3.15}$$

Therefore, we obtain:

$$X_a^q(t + \tau_a^m(t)) = \frac{(X_a(t)w_a(t) - Q_aL_a)k_{aj}}{w_a(t)k_{aj} - Q_a}. \tag{3.16}$$

3.4.4 First-In-First-Out Condition

On a road network, it is necessary to require that the First-In-First-Out (FIFO) discipline be observed for all links. The link FIFO condition states that if a vehicle enters a link at time t, the other vehicles which enter the link later than that vehicle will also leave the link later than that vehicle. This subsection discusses the FIFO condition and its implication on link travel time function $\tau_a(t)$.

Mathematically, the link FIFO condition can be expressed as

$$t_1 + \tau_a(t_1) < t_2 + \tau_a(t_2) \quad \forall t_1 < t_2, \forall a \tag{3.17}$$

According to (3.17), the exit time function $s_a(t) = t + \tau_a(t)$ must be strictly increasing with time t. For any $\Delta t > 0$, it is required that:

$$s_a(t) < s_a(t + \Delta t).$$

or

$$t + \tau_a(t) < t + \Delta t + \tau_a(t + \Delta t).$$

Dividing the above inequality by Δt gives

$$1 + \frac{\tau_a(t + \Delta t) - \tau_a(t)}{\Delta t} > 0. \tag{3.18}$$

If $\tau_a(t)$ is differentiable for any t, taking the limit of the above equation ($\Delta t \to 0$) gives

$$1 + \frac{d\tau_a(t)}{dt} > 0$$

or

$$\frac{d\tau_a(t)}{dt} > -1. \tag{3.19}$$

The above condition must be met to avoid FIFO violation. If the decreasing rate of link travel time on any link exceeds 1, overtaking may occur.

If FIFO condition is satisfied, $s_a(t)$ is strictly increasing. Hence, its inverse $s_a^{-1}(t)$

exists and:

$$s_a(t) = s_a^{-1}(t) + \tau_a(s_a^{-1}(t)).$$

The property that $s_a(t)$ is invertible will help in the development of algorithms for solving the dynamic network loading model presented in the following section.

3.5 The Dynamic Network Loading Model

As stated in the modeling framework, the dynamic network loading model is used to generate link-based network conditions, such as link volumes and link travel times, for given path flows and link performance models. The link-based network conditions are then used to compute path-based network conditions such as path travel times. The mapping from dynamic path flows to link flows is referred to as the dynamic network loading problem. Link performance models are required for the mapping.

Two types of approaches have been used to solve the dynamic network loading problem. One is based on simulation models. CONTRAM[34] and MITSIM[49] are two examples of the simulation approach. Interested readers are referred to the thesis by Yang[47] for a survey of simulation models. One advantage of the simulation approach is its detailed description of the traffic movements over the network. However, it is computationally intensive, especially when the simulation is at a microscopic level and used within an interactive environment.

The other approach is based on analytical models. A number of analytical models have been proposed by Smith[40], Friesz *et al.*[23] and Wu *et al.*[45]. Smith's model is packet-based. It does not have a mathematical formulation. His solution algorithm ensures the FIFO condition but is not efficient. Friesz's model is formulated as a system of equations expressing the link dynamics, flow conservation, flow propagation and boundary constraints. His model assumes that the FIFO condition is satisfied. No solution algorithm is given. Wu's model is essentially the same as Friesz's model but he gave a sufficient condition under which the FIFO condition is satisfied. A solution algorithm was proposed and tested on a network of 9 nodes and 12 links.

The objective of this section is to present the formulation of an analytical dynamic network loading model. Similar to those by Friesz and Wu, our model is formulated as a system of equations expressing link dynamics, flow conservation, flow propagation and boundary constraints. However, our model does not require the assumption of the FIFO condition. In the next chapter, we will also give different solution algorithms for the proposed model.

This section is arranged as follows. First, the formulation of the dynamic network loading model is presented. Then, the existence of a solution to the model is shown and some properties of the solution are discussed.

3.5.1 Formulation of the Dynamic Network Loading Model

The dynamic network loading problem can be formulated as a system of equations that express link dynamics, flow conservation constraints and the flow propagation constraints.

Link dynamics equations

The link dynamics equations express the relationship between the flow variables of a link (see Figure 3-7).

$$\frac{dX_{ap}^{rs}(t)}{dt} = u_{ap}^{rs}(t) - v_{ap}^{rs}(t), \qquad \forall (r,s), \forall p \in K_{rs}, \forall a, \tag{3.20}$$

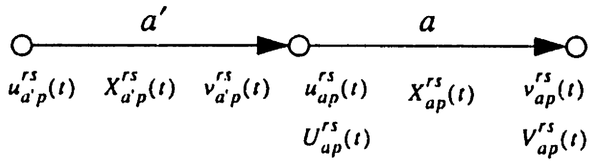

Figure 3-7: Flow Variables

Flow conservation equations

The flow conservation equations for the origin nodes are:

$$u_{ap}^{rs}(t) = f_p^{rs}(t), \qquad \forall (r,s), \forall p \in K_{rs}, \tag{3.21}$$

where a is the first link of path p.

The flow conservation equations between two consecutive links on a path $p \in K_{rs}$ can be expressed as

$$u_{ap}^{rs}(t) = v_{a'p}^{rs}(t), \tag{3.22}$$

where a is after a' (see Figure 3-7).

Flow propagation equations

Flow propagation equations are used to describe the flow progression over time. Note that a flow entering link a at time t will exit the link at time $s_a(t)$. Therefore, by time t, the cumulative exit flow from link a should be equal to the integration of all inflows which entered link a at some earlier time ω and exit link a by time t. This relationship is expressed by the following equation:

$$V_{ap}^{rs}(t) = \int_{\omega \in W} u_{ap}^{rs}(\omega) d\omega, \qquad \forall a, \forall (r,s), \forall p \in K_{rs}, \tag{3.23}$$

where $W = \{\omega : s_a(\omega) \leq t\}$.

Since the network is empty at $t = 0$, the following boundary conditions are required:

$$U_{ap}^{rs}(0) = 0, \quad V_{ap}^{rs}(0) = 0, \quad X_{ap}^{rs}(0) = 0, \qquad \forall (r,s), \forall p \in K_{rs}. \tag{3.24}$$

In summary, the continuous dynamic network loading problem is formulated as the following system of equations:

Link dynamics equations

$$\frac{dX_{ap}^{rs}(t)}{dt} = u_{ap}^{rs}(t) - v_{ap}^{rs}(t), \qquad \forall (r,s), \forall p \in K_{rs}, \forall a, \tag{3.25}$$

Flow conservation equations

$$u_{ap}^{rs}(t) = \begin{cases} f_p^{rs}(t) & , a \text{ is the first link on path } p \in K_{rs} \\ v_{a'p}^{rs}(t) & , a \text{ is after } a' \end{cases} \tag{3.26}$$

Flow propagation equations

$$V_{ap}^{rs}(t) = \int_{\omega \in W} u_{ap}^{rs}(\omega) d\omega, \qquad \forall a, \forall (r,s), \forall p \in K_{rs}, \tag{3.27}$$

where $W = \{\omega : s_a(\omega) \leq t\}$.

Boundary Conditions

$$U_{ap}^{rs}(0) = 0, \quad V_{ap}^{rs}(0) = 0, \quad X_{ap}^{rs}(0) = 0, \qquad \forall (r,s), \forall p \in K_{rs} \tag{3.28}$$

In this system of equations, the known variables are departure flow rates $f_p^{rs}(t)$. Link travel times $\tau_a(t)$ are obtained from link performance models. The unknown variables are $u_{ap}^{rs}(t)$, $v_{ap}^{rs}(t)$, $U_{ap}^{rs}(t)$, $V_{ap}^{rs}(t)$ and $X_{ap}^{rs}(t)$. After these unknown variables are determined, link loads $X_a(t)$ can be computed by

$$X_a(t) = \sum_{rs} \sum_{p \in K_{rs}} X_{ap}^{rs}(t).$$

This model has two advantages over Wu's model[45]: (1) the FIFO condition is not assumed and (2) the model allows for choosing flexible link performance models.

If the FIFO condition is satisfied, the flow propagation equations can be rewritten as

$$V_{ap}^{rs}(t) = \int_0^{s_a^{-1}(t)} u_{ap}^{rs}(\omega) d\omega, \qquad \forall a, \forall (r,s), \forall p \in K_{rs}. \tag{3.29}$$

Wu's model is then a particular case of our model.

3.5.2 Existence of a Solution to the Dynamic Network Loading Model

In this subsection, we will show that the dynamic network loading model formulated in the previous subsection has a solution. In order to prove the existence of a solution to the model, we make the following two assumptions:

1. link travel times are bounded from below by a positive number, and

2. the travel time of a link depends only on the current and/or past traffic conditions on the link.

These two assumptions are realistic because (1) a link has a minimum length and the travel speed is finite and (2) the travel time for a user entering the link usually depends only on the number of vehicles that entered the link earlier.

We have denoted Δ as the minimum link travel time of all links. The entire analysis period $[0, T]$ can be divided into intervals of length Δ. We number these intervals from $0, 1, 2, \cdots$. The i^{th} interval corresponds to $[i\Delta, (i+1)\Delta)$. We will prove the existence of a solution to the model by induction on these intervals in a chronological order.

For $i = 0$ and $t \in [0, \Delta]$, since $\tau_a(t) \geq \Delta$, there is no exit flow. Therefore, $\forall a \in A$, $V_{ap}^{rs}(t) = v_{ap}^{rs}(t) = 0$.

The entrance flow rate $u_{ap}^{rs}(t)$ can be determined using flow conservation equations (3.26). $U_{ap}^{rs}(t)$ can be computed as follows:

$$U_{ap}^{rs}(t) = \int_0^t u_{ap}^{rs}(t)dt.$$

After $U_{ap}^{rs}(t)$ and $V_{ap}^{rs}(t)$ are determined, $X_{ap}^{rs}(t)$ can be computed as follows:

$$X_{ap}^{rs}(t) = U_{ap}^{rs}(t) - V_{ap}^{rs}(t).$$

Therefore, we have shown that for $i = 0$, a solution of all unknown variables can be found.

For $i > 0$, we assume that a solution of all unknown variables exists for intervals $0, 1, \cdots, i$. We will show that a solution of all unknown variables can be found for interval $i + 1$.

For interval $i + 1$ and $t \in [(i+1)\Delta, (i+2)\Delta)$, the exit flows for all links can be computed using the flow propagation equations as follows:

$$V_{ap}^{rs}(t) = \int_{\omega \in W} u_{ap}^{rs}(\omega) d\omega.$$

Since $\tau_a(t) \geq \Delta$, ω must be less than $(i+1)\Delta$. By the induction assumption, $u_{ap}^{rs}(\omega)$ is known for all $\omega < (i+1)\Delta$. Therefore, $V_{ap}^{rs}(t)$ and $v_{ap}^{rs}(t)$ can be determined for all $a \in A$.

By using the flow conservation equations and link dynamics equations, we can determine $u_{ap}^{rs}(t)$, $U_{ap}^{rs}(t)$ and $X_{ap}^{rs}(t)$. Therefore, a solution of all unknown variables can be found for interval $i + 1$.

By induction, we have shown that a solution of all unknown variables exists for every interval i. This proves the existence of a solution to the dynamic network loading model.

3.6 Computation of Path Travel Times

Solving the dynamic network loading problem gives link-based network conditions. However, path travel times are considered for the users to make route choice decisions. The computation of path travel times is described as follows.

For an n-link path $L_p = \{a_1, a_2, \cdots, a_n\}$ from r to s. When a user departs at time t and travels along the path, the arrival time at the tail of link a_i (denoted as t_{a_i}) is given by a sequence of equations:

$$\begin{aligned} t_{a_1}(t) &= t \\ t_{a_2}(t) &= t_{a_1}(t) + \tau_{a_1}(t_{a_1}(t)) \\ \vdots \quad & \quad \vdots \end{aligned}$$

$$t_{a_n}(t) = t_{a_{n-1}}(t) + \tau_{a_{n-1}}(t_{a_{n-1}}(t))$$

Then, the path travel time is given by

$$c_p^{rs}(t) = \sum_{i=1}^{n} \tau_{a_i}(t_{a_i}(t))$$

It is worth noting that the path travel time is the travel time experienced by the users, which depends not only on the network conditions at the time when the user departs, but also on the future network conditions while users is traveling.

3.7 A Flow-Based Dynamic Traffic Assignment Model

According to the modeling framework, the DTA model consists of a users' behavior model, dynamic network loading model and link performance model, along with the interaction between these models. A flow-based DTA model can be formulated as solving the following VI problem:

$$\int_0^{T_d} \sum_{rs} \sum_{p \in K_{rs}} \{F_{2p}^{rs^*}(t)[f_{2p}^{rs}(t) - f_{2p}^{rs^*}(t)] + c_p^{rs^*}(t)[f_{3p}^{rs}(t) - f_{3p}^{rs^*}(t)]\}dt \geq 0 \qquad (3.30)$$

subject to the dynamic network loading equations (3.20)-(3.24).

As seen before, VI (3.30) represents the users' route choice behaviors. The dynamic network loading model is expressed by (3.20)-(3.24). The link performance model is not explicitly included in the DTA model but it must be specified in order to determine the link performance in the dynamic network loading equations. The interaction between these models is captured by the DTA model. The interaction is through path flows $\{f_{mp}^{rs}(t)\}$ and travel times $\{c_p^{rs}(t)\}$. The solution to the DTA model must satisfy the VI and the dynamic network loading equations simultaneously. The development of solution algorithms is the topic of the next chapter.

Chapter 4

Solution Algorithms for the Dynamic Traffic Assignment Model

In chapter 3, we have proposed a framework for the Dynamic Traffic Assignment (DTA) problem. We have also formulated a number of models based on this framework. These models can be used for various purposes. For example, they can be used to evaluate the effectiveness of information and to support the operation of ATMS/ATIS. Therefore, solutions to these models are needed.

Solutions to these models presented in the previous chapter are not available in closed forms. The objective of this chapter is to develop solution algorithms for these models. The solution algorithms presented in this chapter includes

- two dynamic network loading algorithms,
- a route choice algorithm, and
- a DTA algorithm.

The dynamic network loading algorithms are used to solve the dynamic network loading model. The route choice algorithm is used to assign O-D flows among the set of paths based on the users' route choice behaviors and the current network conditions.

The DTA algorithm is a process to solve the DTA model based on dynamic network loading and route choice algorithms.

This chapter is organized as follows. Section 4.1 discusses the discretization of time. Two dynamic network loading algorithms are presented in Section 4.2. A flow update algorithm is given in Section 4.3. Section 4.4 describes the algorithm for the DTA model. Measures of deviation from the users' route choice conditions are presented in Section 4.5.

4.1 Discretization of Continuous Time

The development of solution algorithms is based on discrete versions of the continuous models proposed in Chapter 3. The continuous models are first converted into corresponding discrete models by dividing continuous time into small intervals. The discrete models are then used as a basis for the development of solution algorithms and for computer implementations.

An important consideration in discretization of time is determination of interval length. In general, smaller interval length gives more accurate results since a discrete model will approach a continuous model as interval length goes to zero. However, for the same analysis period, the number of intervals will increase when the interval length decreases. Since the computational effort increases with the number of intervals, more intervals result in loss of computational efficiency. Therefore, in determining the interval length, a trade-off must be made between accuracy and efficiency.

We use the following method to discretize the continuous time into small intervals. From the constructive proof of existence of a solution to the dynamic network loading model in Chapter 3, we can see that network conditions evolve over time from Δ to Δ. Therefore, we choose the interval length δ as follows:

$$\delta = \frac{\Delta}{M}, \quad M \text{ is a positive integer.}$$

Note that the interval length never exceeds the minimum link travel time Δ. This can give reasonably accurate results for shorter links. One can increase M to obtain more accurate results, but computational effort will increase accordingly.

After the continuous time period is divided into small intervals of length δ, each interval is indexed by an integer k, numbered from $k = 0, 1, 2, \cdots$. The k^{th} interval represents $[k\delta, (k+1)\delta)$.

4.2 Solution Algorithms for the Dynamic Network Loading Model

This section presents two algorithms for solving the dynamic network loading model. The algorithms are developed based on the discrete dynamic network loading model.

We will first present the discrete network loading model and then outline two solution algorithms.

4.2.1 Discrete Dynamic Network Loading Model

Based on the continuous model given by (3.25)–(3.28), the discrete dynamic network loading model is derived. For each interval k, we regard continuous variables as constant within that interval. For example, $u_{ap}^{rs}(t) = u_{ap}^{rs}(k), \forall t \in [k\delta, (k+1)\delta)$. Therefore, the discrete dynamic network loading model is as follows:

Link dynamics equations:

$$X_{ap}^{rs}(k) = U_{ap}^{rs}(k) - V_{ap}^{rs}(k) \qquad \forall (r,s), \forall p \in K_{rs}, \forall a \in A. \tag{4.1}$$

Flow conservation equations:

$$u_{ap}^{rs}(k) = \begin{cases} f_p^{rs}(k), & a \text{ is the first link on path } p, \forall (r,s), \forall p \in K_{rs}, \\ v_{a'p}^{rs}(k), & a \text{ is after } a'. \end{cases} \tag{4.2}$$

Flow propagation equations:

$$V^{rs}_{ap}(k) = \sum_{j \in \{j: 0 \le j\delta + \tau_a(j) \le k\delta\}} u^{rs}_{ap}(j)\delta, \qquad \forall (r,s), \forall p \in K_{rs}. \tag{4.3}$$

Boundary conditions:

$$U^{rs}_{ap}(0) = 0, \quad V^{rs}_{ap}(0) = 0, \quad X^{rs}_{ap}(0) = 0, \qquad \forall (r,s), \forall p \in K_{rs}, \forall a \in A. \tag{4.4}$$

Note that $u^{rs}_{ap}(k)$ is needed in flow propagation equations. This variable is determined from flow conservation equations. If a link is not the first link on a path, $u^{rs}_{ap}(k)$ is determined by the exit flow rate $v^{rs}_{a'p}(k)$ of the preceding link. We use the following method to approximate $v^{rs}_{ap}(k)$:

$$v^{rs}_{ap}(k) = \frac{V^{rs}_{ap}(k) - V^{rs}_{ap}(k-1)}{\delta}. \tag{4.5}$$

The cumulative entrance flow $U^{rs}_{ap}(k)$ is needed to compute $X^{rs}_{ap}(k)$ in link dynamics equations. We compute $U^{rs}_{ap}(k)$ as follows:

$$U^{rs}_{ap}(k) = \sum_{j=0}^{k} u^{rs}_{ap}(j)\delta. \tag{4.6}$$

Based on the above discrete model, two dynamic network loading algorithms are developed. They are presented in the following two subsections.

4.2.2 An Iterative Dynamic Network Loading Algorithm

We will first show the idea behind the development of this algorithm before presenting it. Note that if link travel times $\tau_a(k)$ are known and do not change with network conditions, a solution to the discrete model can be found by propagating flows along the paths from origins to destinations. However, if $\tau_a(k)$ are function of network conditions, this method may not be valid. The reason is that, using a set of fixed link travel time $\{\tau_a(k)\}$ to propagate path flows will result in a new network conditions, and hence a set of new link travel times, say $\{\tau_a^{new}(k)\}$. The method is not valid if $\{\tau_a(k)\}$ is not equal to $\{\tau_a^{new}(k)\}$.

Nevertheless, an iterative process on $\tau_a(k)$ may lead to a solution. At each iteration, $\tau_a(k)$ are updated. If, after a number of iterations, the set of new link travel times $\{\tau_a^{new}(k)\}$ converge to a set of link travel times $\{\tau_a^*(k)\}$, a solution is found.

The Method of Successive Averages (MSA)[39] has been used in static traffic assignment. We adopt MSA in this algorithm as a heuristic method since no proof of convergence is available at this moment for the dynamic network loading problem.

Based on the above idea, we develop an iterative algorithm for solving the dynamic network loading model. We call it "I-Load" algorithm. The algorithm is outlined as follows:

I-Load Algorithm

```
Step 0: (Initialization)
        $N$ = maximum number of iterations;
        $\{\tau_a^{(0)}(k)\}$ = free flow travel time;
        $n = 0$.
Step 1: (Main loop)
        1.1: Move departure flows along the paths according to $\{\tau_a^{(n)}(k)\}$;
        1.2: Compute the resulting total link volumes;
        1.3: Compute the new link travel times $\{\tau_a^{new}(k)\}$;
        1.4: Update link travel times:
             $\tau_a^{(n+1)}(k) = \tau_a^{(n)}(k) + \alpha^n(\tau_a^{new}(k) - \tau_a^{(n)}(k))$;
             $\alpha^n = \frac{1}{n+1}$.
Step 2: (Stopping criterion)
        If $n = N$, then stop;
        Otherwise, $n = n + 1$ and go to Step 1.
```

4.2.3 A Chronological Dynamic Network Loading Algorithm

The construction proof of existence of a solution to the continuous dynamic network loading model given in Chapter 3 suggests another different algorithm for the dynamic network loading model. Instead of solving the problem iteratively, the algorithm find the solution within each $(i\Delta, (i+1)\Delta]$ in a chronological order. We call it "C-Load" algorithm. The algorithm is outlined as follows:

C-Load Algorithm

```
Step 0: (Initialization)
        Determine $\Delta$ by $\Delta = \min_a[\tau_a(0)]$;
        $M$: number of $\delta$ intervals within a $\Delta$ interval;
        $i = 0$.
Step 1: (Solve the system of equations within $i^{th}$ $\Delta$ interval)
        1.1: for ($a \in A$ and $p$ passing through $a$) do
                 for $k = iM$ to $(i+1)M - 1$ do:
                     Compute $V_{ap}^{rs}(k)$ using (4.3);
                     Compute $v_{ap}^{rs}(k)$ using (4.5).
        1.2: for ($a \in A$ and $p$ passing through $a$) do:
                 for $k = iM$ to $[(i+1)M - 1]$ do
                     Compute $u_{ap}^{rs}(k)$ using (4.2);
                     Compute $U_{ap}^{rs}(k)$ using (4.6);
                     Compute $X_{ap}^{rs}(k)$ using (4.1);
                     Compute $\tau_a(k)$ using given link performance model.
Step 2: (Stopping criterion)
        If the network is empty, then stop;
        Otherwise, $i = i + 1, T = i\Delta$ and go to Step 1.
```

When this algorithm terminates, link-based network conditions are obtained. The time at which the algorithm stops is also the duration of the analysis period T.

We make some comments on the C-Load algorithm. First, this algorithm does not require an iterative process on link travel times. Thus, there is no convergence problem. Second, it allows for the flexibility to choose various link performance models such as queuing models. Third, we feel that incidents can be conveniently studied with this algorithm because of its similarity to a simulation model. In the next chapter, we will present efficient computer implementations for the C-Load algorithm.

4.3 A Route Choice Algorithm

A route choice algorithm is used to find a solution to the users' behavior model. The inputs to the users' behavior model are dynamic O-D demands and path-based network conditions. The output is a set of path flows. A route choice algorithm is to assign the given O-D demands to a subset of paths.

In this thesis, we assume that users choose routes based on path travel times. Recall that the dynamic network loading model provides link travel times. Path travel times can then be computed from link travel times by using the method presented in Chapter 3.

We now describe the method for assigning O-D demands to each path. Denote by $g_{mp}^{rs}(k)$ the path flow of type m users for path p of O-D pair (r, s) at interval k. Based on the users' route choice behaviors, $g_{mp}^{rs}(k)$ are determined as follows:

- For type 1 users, the assignment is trivial and no computation is needed since they always choose fixed routes.

- For type 2 users, we first compute the probability that a route is chosen by a type 2 user using a specific discrete choice model. Appendix A describes one of the choice models. Then $g_{2p}^{rs}(k)$ is given by:

$$g_{2p}^{rs}(k) = f_2^{rs}(k) \times \text{Probability that route } p \text{ is chosen by type 2 users.} \qquad (4.7)$$

- For type 3 users, we currently assign all type 3 users to a minimum travel time routes at each interval. Therefore, $g_{3p}^{rs}(k)$ is given by:

$$g_{3p}^{rs}(k) = \begin{cases} f_3^{rs}(k), & \text{if path } p \text{ is a minimum travel time path,} \\ 0, & \text{otherwise.} \end{cases} \tag{4.8}$$

In case that more than one paths are minimum, we can assign type 3 users equally to these minimum travel time paths. Denote by $N_{min}(k)$ the number of minimum travel time paths at interval k. $g_{3p}^{rs}(k)$ is given by:

$$g_{3p}^{rs}(k) = \begin{cases} \frac{f_3^{rs}(k)}{N_{min}(k)}, & \text{if path } p \text{ is a minimum travel time path,} \\ 0, & \text{otherwise.} \end{cases} \tag{4.9}$$

In summary, the route choice algorithm is outlined as below.

Route Choice Algorithm

```
Step 1: for all O-D pair $(r,s)$ and $p \in K_{rs}$;
        Compute path travel times from link travel times.
Step 2: for all O-D pair $(r,s)$ and $p \in K_{rs}$,
        Compute $g_{2p}^{rs}(k)$ for type 2 users using (4.7);
        Compute $g_{3p}^{rs}(k)$ for type 3 users using (4.8).
```

4.4 The Solution Algorithm for the DTA Model

An iterative process is needed to solve the DTA model. This is because network conditions may change after performing network loading. This results in a set of new path travel times and thus a set of new path flows. The set of new path flows are not necessarily equal to the set of path flows used in the previous network loading procedure.

The idea of the solution algorithm is to find a solution to the DTA model by an iterative process on path flows. At each iteration, the path flows are updated by combining the results from the current iteration with the previous iteration. We use MSA[39] to update path flows. Since no proof of convergence is available at this moment for dynamic traffic assignment, the method is heuristic. The DTA algorithm is outlined below.

DTA Algorithm

```
Step 0: (Initialization)
        $N$ =maximum number of iterations;
        Compute initial path flows $\{f_{mp}^{rs^{(0)}}(k)\}$ from free-flow path travel times;
        $n = 0$.
Step 1: (Main loop)
        1.1: Perform dynamic network loading procedure using I-Load or
             C-Load Algorithm;
        1.2: Compute $g_{mp}^{rs}(k)$ by the route choice algorithm;
        1.3: Update path flows:
             $f_{mp}^{rs^{(n+1)}}(k) = f_{mp}^{rs^{(n)}}(k) + \alpha^{(n)}[g_{mp}^{rs}(k) - f_{mp}^{rs^{(n)}}(k)]$,
             $\alpha^{(n)} = \frac{1}{n+1}$, $m = 2, 3$.
Step 2: (Stopping criterion)
        If $n = N$, then stop;
        Otherwise, $n = n + 1$ and go to Step 1.
```

4.5 Measures of Deviation from Users' Route Choice Conditions

As we mentioned before, the DTA algorithm is a heuristic since no proof of convergence is available at this moment. Therefore, there is no guarantee that the solution generated by the algorithm will satisfy the users' route choice conditions. In order to evaluate the deviation of the solution from the users' route choice conditions, we will present the measures of deviation for type 2 and type 3 users.

We first recall an elementary theorem that is relevant to the measures we will present next. This theorem states that a vector ($\{x_i\} = 0$) is equivalent to $\sum_i \mid x_i \mid = 0$.

The measures of deviation are based on the above theorem and the route choice conditions stated in Chapter 3. For type 2 users, we define

$$\begin{aligned} D_{2p}^{rs}(k) &= \mid f_{2p}^{rs*}(k) - f_2^{rs}(k)P_p^{rs}(k) \mid, \qquad \forall (r,s), \forall p \in K_{rs}, \forall k, && (4.10) \\ D_2 &= \sum_k \sum_{rs} \sum_{p \in K_{rs}} D_{2p}^{rs}(k) && (4.11) \end{aligned}$$

Here, $f_{2p}^{rs*}(k)$ denotes the final result when the DTA algorithm terminates. $P_p^{rs}(k)$ is the proportion of O-D flow assigned to path p at time interval k computed based on the path travel times $c_p^{rs}(k)$ in the route choice algorithm.

We then use $D_{2p}^{rs}(k)$ as a disaggregate measure of deviation from type 2 users' route choice condition for path p of an O-D pair (r, s) at interval k. We note $D_{2p}^{rs}(k) = 0$ is equivalent to $f_{2p}^{rs*}(k) - f_2^{rs}(k)P_p^{rs}(k) = 0$. We use D_2 as an aggregate measure of this deviation over all paths and intervals. $D_2 = 0$ is equivalent to $D_{2p}^{rs}(k) = 0, \forall (r,s), p \in K_{rs}, \forall k$.

For type 3 users, we define

$$\begin{aligned} D_{3p}^{rs}(k) &= f_{3p}^{rs*}(k)(c_p^{rs}(k) - \pi^{rs}(k)), \qquad \forall (r,s), \forall p \in K_{rs}, \forall k, && (4.12) \\ D_3 &= \sum_k \sum_{rs} \sum_{p \in K_{rs}} D_{3p}^{rs}(k) && (4.13) \end{aligned}$$

Here, $f_{3p}^{rs*}(k)$ denotes the final result when the algorithm terminates. $c_p^{rs}(k)$ is the travel time of path p at time interval k computed in the route choice algorithm. $\pi^{rs}(k)$ is the minimum of $c_p^{rs}(k)$ for O-D pair rs.

We then use $D_{3p}^{rs}(k)$ as a disaggregate measure of deviation from type 3 users' route choice condition for path p of an O-D pair (r, s) at interval k. We note $D_{3p}^{rs}(k) = 0$ is equivalent to type 3 users' route choice condition since, $D_{3p}^{rs}(k) = 0$ is exactly Equation (3.4) (see Chapter 3) and Equations (3.3), while (3.5) are automatically satisfied by the solution obtained from the DTA algorithm. We use D_3 as an aggregate measure of this deviation over all paths and intervals. $D_3 = 0$ is equivalent to $D_{3p}^{rs}(k) = 0, \forall (r,s), p \in K_{rs}, \forall k$.

It is desirable that these measures decrease as the number of iterations increases. However, there is no reason that these measures will behave link this. In Chapter 6, we will use these measures to evaluate the results from the DTA algorithm using two examples, one of which is a real-life application.

Chapter 5

Computer Implementations

In Chapter 4, we have presented solution algorithms for the Dynamic Traffic Assignment (DTA) model. In this research work, we developed a software system that implements these solution algorithms. This software system can solve realistic DTA problems in real time and can be integrated within larger decision support software system for ITS. In this chapter, we describe this software system and present the methods that were developed in order to achieve fast execution time.

Unlike static traffic assignment problems, the DTA problem involves a path dimension and a time dimension. Both dimensions substantially increase the problem size. The problem, with increased size, requires more time to solve and more memory for storing static and dynamic data. This may prevent us from solving realistic DTA problems in real time.

As we saw in Chapter 2, different models and algorithms have been proposed in the literature. We note that only a limited number of papers dealt with implementing those solution algorithms. Tests of these implementations have been carried out on very small networks (in the order of 10 links). Even for these small networks, running times were in the order of minutes.

Real-world transportation networks can contain hundreds of links and thousands of O-D pairs and paths. For example, the Amsterdam A10 beltway (see Chapter 6) contains more than 300 links and over 1000 paths. The morning peak period lasts for two hours. This problem would not be solved faster than real time by known analytical computer implementations.

In developing our computer implementations, we set two objectives. The first objective is to be able to solve a DTA problem for realistic networks in a running time faster than real time. The second objective is to develop a software system that (1) can be integrated with other ITS decision support modules, and (2) is easy to maintain, expand and upgrade.

To achieve the first objective, solution algorithms were efficiently implemented by using the following approaches:

- design of efficient data structures, and
- efficient implementation of bottleneck operations.

To achieve the second objective, we use an object-oriented design and development paradigm[38]. The concept of object-orientation means that a program is organized as a collection of objects. Each such object is composed of both static data and methods for manipulating these data. The benefits from using the object-oriented design and development paradigm include:

- better organization of data and methods,
- better communication between objects, and
- easier debugging, better readability, maintainability and reusability of the code.

This chapter is organized as follows. In Section 5.1, we present the major challenges that we overcome in order to achieve fast running times. Section 5.2 describes how the network data are represented. Section 5.3 describes the implementations of the algorithms. Section 5.4 presents two novel methods to achieve computational efficiency.

5.1 Challenges to Achieve Efficient Implementations

We saw that one of the primary objectives of computer implementation is to achieve running times faster than real time. We observe that dynamic network loading is a bottleneck procedure. Therefore, in order to improve overall computational efficiency, one needs to implement efficiently the dynamic network loading solution algorithm.

Below we list three major challenges to be overcome in order to achieve fast running time:

- **Representation of paths:** As mentioned before, the DTA problem involves a path dimension. Each path consists of a sequence of links. A link can be shared by many paths. To represent paths efficiently, we use a special data structure call subpath table. The details of this representation is described in Section 5.2.4.

- **Reduction of the number of link-path flow variables:** Every link-path variable requires memory to store and takes time to process. Reducing the number of such variables can substantially improve computational efficiency and lower memory requirement. In Section 5.4.1 , we will present a method to consolidate link-path flow variables that have the same destination and follow the same subpath to that destination.

- **Computation of exit flows:** In the C-Load algorithm, computation of exit flows $V_{ap}^{rs}(k)$ could cause bottleneck if not implemented efficiently. In Section 5.4.2, we will present an efficient method to compute these exit flows.

5.2 Network Data Representations

The network data include links, O-D pairs and paths. In general, a transportation network contains two basic elements: links and nodes. O-D pairs and paths are built on the top of these two elements. An O-D pair contains a set of paths. A path is a sequence of links. The time dimension present in DTA problems is represented as discrete time intervals.

As mentioned before, we adopt an object-oriented design method for designing the software system. To facilitate the object-oriented implementation of this system, we chose C++ as programming language. C++ is an object-oriented extension of the widely used C programming language[42]. It has built-in capabilities of supporting the implementation of a solution designed using an object-oriented paradigm.

In C++, classes are used to represent a group of objects that have a set of common attributes, variables and methods. Attributes are static, while variables can change during the execution of the program. Methods are functions that operate on the attributes and variables. As we will see later, links and O-D pairs each have a set of common attributes, variables and methods. Therefore, it is natural to declare two classes, link class and O-D pair class, to describe link and O-D pair data respectively. Links and O-D pairs are respectively instances (objects) of the link class and O-D pair class. In the following two subsections, we describe the contents of link class and O-D pair class.

5.2.1 Representation of O-D Pairs

Each O-D pair is represented as an object of the O-D pair class. The O-D pair class defines a set of attributes, variables and methods associated with an O-D pair. The attributes of an O-D pair include:

- origin and destination nodes,
- number of paths connecting the origin to the destination,
- the subpath number for the first path of the O-D pair (see the next subsection),
- dynamic O-D trips represented as the number of trips during a given time period, and
- commonality factors for the paths (see Appendix A).

The variables in the O-D pair class are path flow proportions for each type of user and path at each time interval. These variables are arranged in a three-dimensional array. To save memory, path travel times and path flows are not stored in the object. However, we can compute path travel times from link travel times contained in the link objects. Path flows can be computed from O-D demand and the path flow proportions.

The following methods are associated with the O-D pair class:

- *MoveFlow()*: move path departure flows along the path (see the I-Load algorithm in Chapter 4),
- *ComputePathTravelTime()*: compute path travel times (see Section 3.6 in Chapter 3),
- *ComputePathFlow()*: compute a set of new path flow proportions and update path flows, and
- *OutputPathFlowAndTravelTime()*: return path flows and path travel times.

Note that the arguments of these methods are not indicated.

All the O-D pair objects are organized in an array. An O-D pair object is then referenced by a unique integer number (internal O-D pair identifier). The order in which O-D pair objects are numbered is arbitrary.

5.2.2 Representation of Links

Links are directed edges with two end nodes. The direction is from the tail node to the head node. Each link is represented as an object of link class. The link class contains a set of common attributes, variables and methods for all link objects. The following attributes are currently included in the link class:

- link identification number,
- link type,

- tail node and head node,
- link length (miles),
- free-flow speed (miles/hour),
- acceptance flow rate (vehicles/hour),
- exit flow rate (vehicles/hour), and
- jam density (vehicles/mile).

The link attributes are used mainly in the link performance model to determine the link travel times. In the current implementation, they are assumed to be static.

The link class also includes the following variables for storing the time-dependent information:

- total cumulative entrance flow $U_a(k)$ at each time interval,
- total cumulative exit flow $V_a(k)$ at each time interval,
- link travel time at each time interval, and
- pointer to the objects of link-path flow variables (see Section 5.4.1).

In order to reduce memory requirement, we do not include in an object those variables that can be computed from other variables that are in the object. For example, it is not necessary to store link volume in the link objects, because it can be computed as a difference between entrance flow and exit flow.

For the C-Load algorithm, link-path flow variables are needed. The running time taken to solve dynamic loading model by the C-Load algorithm depends on the number of such variables. In Section 5.4.1, we will show that we can reduce the number of link-path flow variables by a consolidation technique.

A link class contains the following methods to perform the common functions associated with a link:

- *AllocateMemory()*: dynamically allocate memory for the time-dependent link variables,
- *ComputeExitFlow()*: compute exit flows within each Δ interval (see Equation (4.3)),
- *ComputeLinkTravelTime()*: compute link travel time using a given link performance model,
- *UpdateLinkTravelTime()*: update link travel times (see the I-Load algorithm in Chapter 4), and
- *OutputLinkCondition()*: return link conditions, such as time-dependent link volumes and link travel times.

All the link objects are organized in an array. A link object is then referenced by a unique integer number (internal identifier or link number). The order in which links are numbered is arbitrary.

5.2.3 Implementation of Link Performance Models

As we saw in the modeling framework presented in Chapter 3, the link performance model component provides the dynamic network loading model component with the link performance (measured here as link travel times). We note that to obtain link travel times for different types of link, different link performance models are needed. For example, speed-density relationship is suitable for freeway links while a queuing model is suitable for arterial street links.

At the time when this thesis is written, we write a C++ function for each type of links to compute link travel time. This function has a number of arguments that are needed to compute link travel times. Recall that each link object contains a method *ComputeLinkTravelTime()*. This method calls a suitable link travel time function depending on the link type.

In the future, we can implement link performance models in a better way. The link performance model component in the framework can be represented as a generic

Link Performance Model (LPM) class. From this LPM class, specific link performance model classes can be derived. For example, speed-density relationship and a queuing model are two derived link performance classes. Then, each link object contains a link performance model object belonging to certain derived link performance model class. The advantage of this representation is that all link objects will have a common interface with link performance models and it is easier to define new link performance models.

5.2.4 Representation of Paths

A path is a sequence of links. One approach to represent a path is to store its sequence of links in an array. In such representation, paths are independent of each other. Although this approach is simple, it is not efficient, because a link may be used by a large number of paths. Therefore, there may be many repetitions of links, which result in a waste of memory. For example, the network in Figure 5-1 has two O-D pairs, (r_1, s) and (r_2, s), each of which is connected by one path. An independent

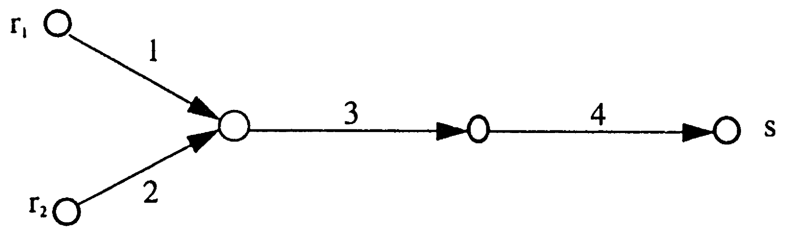

Figure 5-1: Two Paths with Two Common Links

array representation would be [1,3,4] and [2,3,4]. Note that links 3 and 4 are shared by the two paths and repeated in the two arrays.

To avoid some of the repetitions of links, we design a new data structure named *subpath table* to represent paths. Before presenting the general idea of subpath table, we illustrate it using the above example. Table 5.1 shows how the links of the two paths for the above example in Figure 5-1 are stored in a subpath table. Each of the original paths is assigned a subpath number. For example, path (1,3,4) is assigned subpath number 0. A sequence of links for a path can be easily obtained by traversing

Table 5.1: An Example of Subpath Representation of Paths

subpath number	next-link	next-subpath
0	1	2
1	2	2
2	3	3
3	4	-1

the subpath table. For example, to find the sequence of links for path (1,3,4), we first go to subpath 0 and get link 1, then go to subpath 2 and get link 3. Finally, we go to subpath 3 and get link 4. From the subpath table, it can be seen that links 3 and 4 are stored only once in the table. In general, the subpath structure can avoid link repetitions when paths with a common destination follow the same subpath to that destination.

We next describe the method of representing paths with a subpath table. A subpath is defined as a portion of a path that contains the destination of that path. Figure 5-2 depicts the general structure of a subpath.

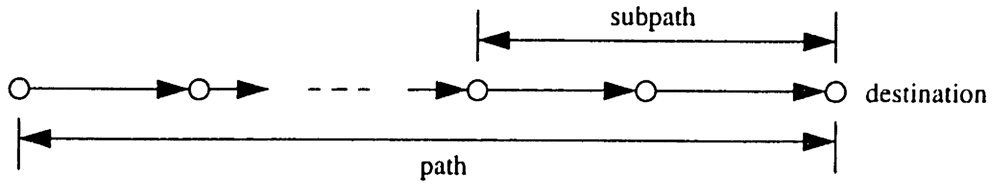

Figure 5-2: A Subpath of a Path

Each of the subpaths from a link to a destination is identified by an identification number. All paths that belong to an O-D pair can be regarded as subpaths and they are numbered consecutively. A subpath $(L_1, L_a, \cdots, L_n)$ can be viewed as consisting of two parts: $L_1 and$ subpath $(L_2, \cdots, L_n)$. That is, after L_1 is traversed, $(L_2, \cdots, L_n)$ itself is a subpath from L_2 to the same destination. If some paths have a common destination and follow the same subpath to that destination, this recursive property of subpaths allows for those paths to share that subpath.

The subpath table is a two-dimensional array with two columns. The first column (denoted as *next-link*) stores the link number which a subpath will enter next. The second column (denoted as *next-subpath*) stores the subpath number which a subpath must follow after exiting from the current link. If this link is the last link of the subpath, the next-subpath is marked as -1, indicating the end of both subpath and path.

The subpath table is stored in the network data file and read in memory at the initialization step.

5.2.5 Relationship between the Network Data

As we mentioned before, O-D pairs and paths are built on the top of nodes and links. The hierarchical relationship of these data is as follows: an O-D pairs contains a set of paths. A path consists of a sequence of links. Therefore, the three set of network data described in the previous subsections are not independent of each other. Instead, they are related in such a way that lower-level objects can be referenced by higher-level objects via a reference number or index.

Figure 5-3 illustrates the relationship and the referencing scheme for the network data. To look up a path connecting the O-D pair, an O-D pair object uses a subpath

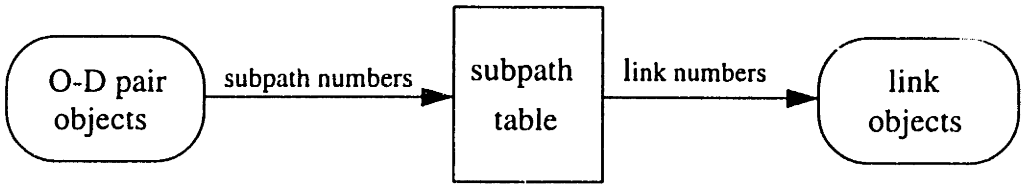

Figure 5-3: Relationship between the Network Data

number as index to the subpath table. As explained in the previous subsection, a sequence of links on the path can be obtained by traversing the subpath table. The relationship between the network data makes it easy and efficient to compute path travel times and length.

5.3 Implementations of the Solution Algorithms

In the previous section, we have presented the representation of the network data. In this section, we will describe the implementations of the dynamic network loading algorithms, the route choice algorithm and the DTA algorithm. As will be seen next, the implementations of these algorithms are constructed by calling the methods in the link and the O-D pair classes.

5.3.1 Implementation of the Dynamic Network Loading Algorithms

We have implemented two dynamic network loading algorithms: the I-Loand algorithm and the C-Load algorithm. Each algorithm is implemented as a function that carries out a series of operations outlined in the algorithm.

Now we briefly describe the implementation of the I-Load algorithm. At the initialization step, the initial link travel times are computed by calling *ComputeLinkTravelTime()* method in the link objects with zero link load. In the main loop, at each iteration, we call *MoveFlow()* method in the O-D pair objects to move departure flows along all paths. After calling this method for all O-D pair objects, total link loads are obtained. We then compute new link travel times by calling *ComputeLinkTravelTime()* method in the link class with the new link loads. We update the link travel times by calling *UpdateLinkTravelTime()* method in link objects. The loop stops when the maximum number of iterations is reached.

For the implementation of the C-Load algorithm, at each time interval, each step is carried out by calling a method in link objects. No methods in O-D class are needed. At the initialization step, we determine Δ by calling *ComputeLinkTravelTime()* in link objects to compute free flow travel times for all links. At each time interval, we first compute $V_{ap}^{rs}(k)$ and $v_{ap}^{rs}(k)$ by calling *ComputeExitFlow()* method in the link object for all links. It is not necessary to compute the entrance flows, since we can obtain these from the upstream link objects by using the upstream pointers in the LPV objects. After entrance and exit flows are obtained, link travel times are computed by calling *ComputeLinkTravelTime()* in link class.

For the C-Loading algorithm, we need to call a function called *IsEmpty()* to check whether the network is empty, meaning that no users are in the network and no users will enter the network. Note that this function is called only after the demand period T_d because the network cannot be empty within this period. The algorithm stops when this function return a TRUE value, and the current time is return as the total analysis period T.

5.3.2 Implementation of the Route Choice Algorithm

The method *ComputePathFlow()* in the O-D pair class is implemented as the route choice algorithm. In this method, the travel time of each path is first computed by calling another method *ComputePathTravelTime()* in the O-D pair class. Then, a set of new path flows are computed for type 2 and type 3 users based on these path travel times.

5.3.3 Implementation of the DTA algorithm

The DTA algorithm is implemented as a series of operations corresponding to each step outlined in the algorithm. At the initialization step, initial path flows are computed from free-flow path travel times. A function *Consolidation()* is also called to consolidate link-path flow variables. In the main loop, at each iteration, the network loading function is first called. Then, for each O-D pair object, the method *ComputePathFlow()* is called to compute a set of new path flow proportions. Since we do not want to store intermediate path flows $g^{rs}_{mp}(k)$, path flows are also updated using the MSA inside this method. The function returns when a maximum number of iterations is reached.

5.4 Special Methods for Achieving Computational Efficiency

In Section 5.1, we listed three challenges to to overcome to achieve computational efficiency. We have presented a method for representing paths in the previous section. This section presents two special methods that meet the other two challenges, namely the reduction of the number of link-path flow variables and the computation of exit flows.

5.4.1 A Consolidation Method of Link-Path Flow Variables

In the C-Load algorithm, all link-path flow variables belonging to a link need to be processed at each interval. Hence, computational efficiency depends on the number of such variables. Since the number of link-path flow variables of a link is equal to the number of paths to which the link belongs, this number can be very large if many paths share a link.

The basic idea of reducing the number of link-path flow variables is rooted in the following observation: it is not necessary to distinguish those path flows which have a common destination and follow the same subpath from a link to that destination. Such path flows will experience the same link travel times, and the total flows of the links on the subpath are the sum of the link-path flow variables. For example, in Figure 5-1, paths $(1, 3, 4)$ and $(2, 3, 4)$ have a common destination s and follow the same subpath $(3, 4)$ to the same destination after first traversing links 1 and 2, respectively. In this situation, the link-path flow variables of the two paths on links 3 and 4 can be consolidated into one variable for each of the last two links. Therefore, the number of link-path flow variables is reduced by consolidating link-path variables. This will improve computational efficiency.

Before presenting the consolidation technique, we describe a special data structure to represent link-path flow variables. This data structure is named LPV class (LPV stands for Link-Path flow Variable). Each link-path flow variable is an object of the

LPV class.

Figure 5-4 shows the LPV objects contained in a link object. The information contained in the LPV class are:

- O-D pair identification number,
- path identification number,
- a pointer to a list of upstream LPV objects (this pointer is NULL if the link is the first link on a path),
- a pointer to the downstream LPV object (this pointer is NULL if the link is the last link on a path), and
- a pointer to the next LPV object of the same link.

Note that all LPV objects of a link are organized in a singly-linked list. An LPV object in the list may point to more than one upstream LPV object (which means some link-path flow variables are consolidated), but it points to only one downstream LPV object.

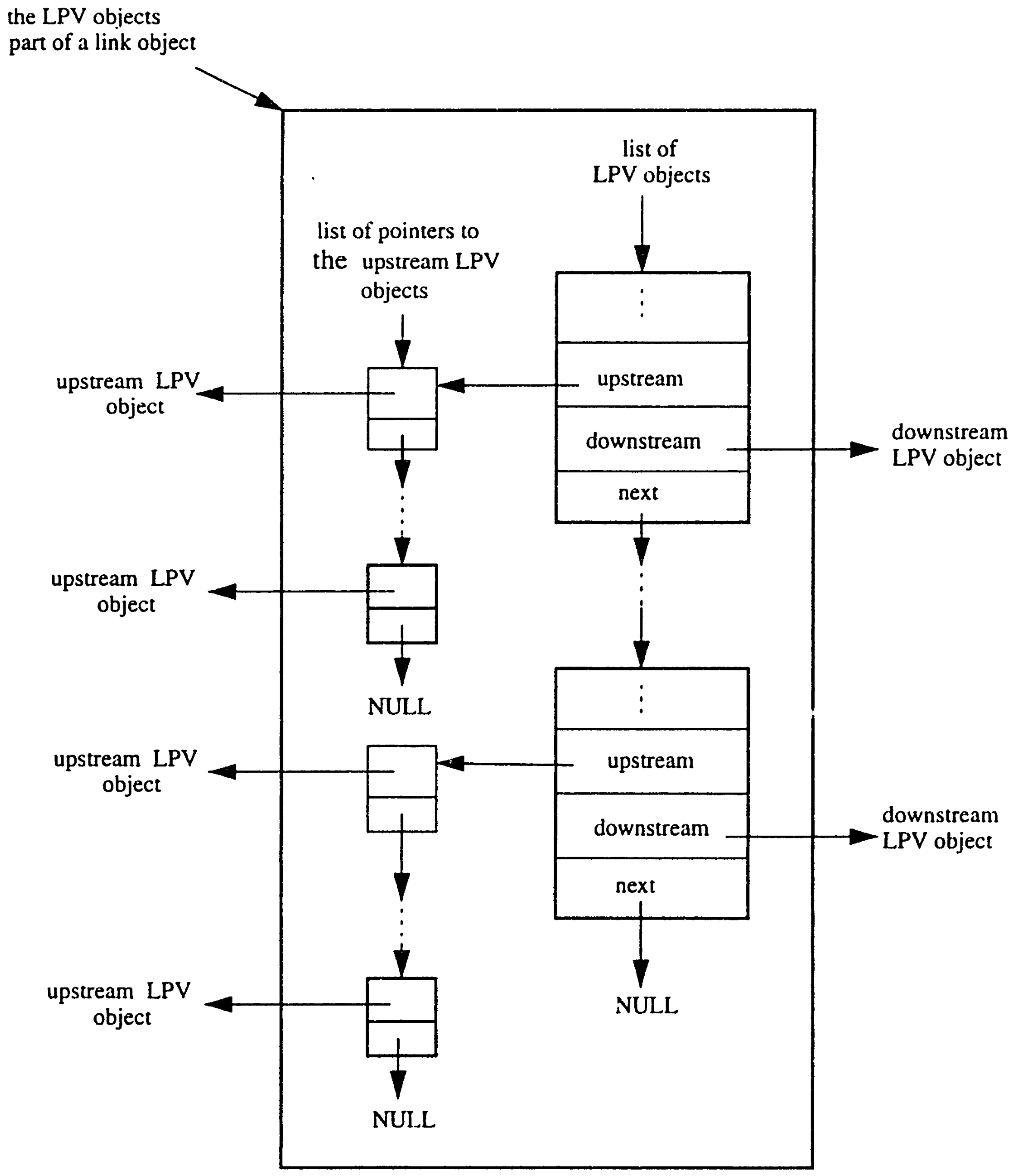

Figure 5-4: The LPV Objects Part of a Link Object

The consolidation procedure is outlined below:

Consolidation Procedure

Step 1: (Initialization)

For each path, create an LPV object for each link on a path by traversing forward that path.

Step 2: (Consolidation)

For each path,

2.1: Find the sequence, in forward format, of links on this path. Denote the sequence by $(1, 2, \cdots, n)$.

2.2: Process each link on the path in a backward manner

- For the last link n, consolidate those LPV objects whose downstream pointers are NULL by using any one of the objects as representative, say R. The upstream pointers in those deleted LPV objects are added to the list of upstream pointers in the representative LPV object R. All the other objects are then deleted.
- For link $n-1$, consolidate those LPV objects which are pointed to by upstream pointers in R. Again use one of those objects as representative and delete all the other ones. Add the upstream pointers in the deleted objects to the representative object.
- Reset R to be the representative LPV object of link $n-1$. For links $n-2, \cdots, 1$, repeat the the above procedure as for link $n-1$ to complete the consolidation for all links on the path.

After running the consolidation procedure, the redundant link-path flow variables are deleted. The C-Load algorithm is performed on the reduced set of link-path flow variables.

5.4.2 Method for Computing Exit Flows in the C-Load Algorithm

In C-Load network loading algorithm presented in Chapter 4, at each time interval k, we need to compute the exit flows. These are obtained by using the following link flow propagation equations:

$$V_{ap}^{rs}(k) = \sum_{j \in \{j : 0 \leq j\delta + \tau_a(j) \leq k\delta\}} u_{ap}^{rs}(j)\delta, \quad \forall (r,s), p \in K_{rs}, \quad \forall a \in p.$$

The computation of $V_{ap}^{rs}(k)$ looks straightforward. However, it can be very time-consuming if the summation starts from $j = 0$, for all k. Instead of computing $V_{ap}^{rs}(k)$ in this naive way, an efficient method is developed below.

First, the original flow propagation equations are rewritten as:

$$V_{ap}^{rs}(k) = V_{ap}^{rs}(k-1) + \sum_{j \in \{j : (k-1)\delta < j\delta + \tau_a(j) \leq k\delta\}} u_{ap}^{rs}(j)\delta, \quad \forall a \in A, \forall (r,s), p \in K_{rs}. \quad (5.1)$$

Note that Equation (5.1) does not require the FIFO condition to be satisfied. It can be seen from the transformed equations that to compute $V_{ap}^{rs}(k)$ at interval k, only the second term on the right hand side needs to be computed. Furthermore, each link needs to save only the values of $V_{ap}^{rs}(k-1)$ and a limited number of the past entrance flow rate variables $u_{ap}^{rs}(j)$. Below we first show which portion of the past entrance flow rates that should be saved. Next we present an efficient method to store this relevant portion of past values of entrance flows.

To determine which portion of the past entrance flow rates that should be saved, we note that the flow that exists link a at time interval k may have entered link a at different past time intervals. Denote by $j_a^{min}(k)$ the minimum of these entrance time intervals. Mathematically, $j_a^{min}(k)$ can be expressed as:

$$j_a^{min}(k) = \min_j[(k-1)\delta < j\delta + \tau_a(j) \leq k\delta], \quad j \text{ is positive integer.} \quad (5.2)$$

We need to store only the entrance flow rate value for all intervals $j \in [j_a^{min}(k), k-1]$

in order to compute the exit flow for interval k.

The method to store this relevant portion of past values of entrance flows exploits a particular property of $j_a^{min}(k)$ stated in the following theorem.

Theorem 5.1 *If the link travel time function $\tau_a(t)$ is continuous and nonnegative, then $j_a^{min}(k)$ defined by 5.2 is a nondecreasing function of k.*

Proof:

We will prove this theorem by contradiction. Suppose that this theorem is not true. Then, there exist $k_1 < k_2$ such that $j_a^{min}(k_1) > j_a^{min}(k_2)$. Since $s_a(k) \geq s_a(0), \forall k$ and $s_a(j_a^{min}(k_2) > s_a(j_a^{min}(k_1)$, by the theorem of continuity, there exists a number $t^* \in [0, j_a^{min}(k_2))$ such that $s_a(t^*) \in [k_1 - 1, k_1)$. Since $t^* < j_a^{min}(k_1)$. This contradicts the definition of $j_a^{min}(k)$. The contradiction proves that $j_a^{min}(k)$ is a nondecreasing function of k.

We note that the only condition for this theorem is the continuity and nonnegativity of the link travel time. The condition is generally satisfied in reality. Thus, it is not restrictive.

By this theorem, we see that the interval $[j_a^{min}(k), k-1]$ has a dynamic nature. The length of the interval could grow and shrink. However, a past value of entrance flow rate for some $j \in [j_a^{min}(k), k-1]$ can be discarded when $j_a^{min}(k^*) > j$ for some $k^* > k$.

To compute $V_{ap}^{rs}(k)$, we sum up flows $u_{ap}^{rs}(j)$ for all $(k-1)\delta < j\delta + \tau_a(j) \leq k\delta$ such that, $j \in [j_a^{min}(k), k-1]$. If the FIFO condition is satisfied, this summation can be reduced to interval $[j_a^{min}(k), j_a^{max}(k)]$, where $j_a^{max}(k) = \max_{j \geq j_a^{min}(k)}[j\delta + \tau_a(j) \leq k\delta]$. Therefore, if the FIFO condition is respected, this operation can speed up.

While computing $V_{ap}^{rs}(k)$, we evaluate and update $j_a^{min}(k)$. This is done as follows. Starting from $j = j_a^{min}(k)$, increment j until $j\delta + \tau_a(j) > k\delta$. This j is then $j_a^{min}(k+1)$.

A circular queue list depicted in Figure 5-5 is suitable to store the portion of historic values of entrance flow rate. Each element of the list stores one past value, $u_{ap}^{rs}(j)$, and contains a pointer to the next element in the list. Two pointers *tail* and *head*, point to the tail and the head of the circular queue list, respectively.

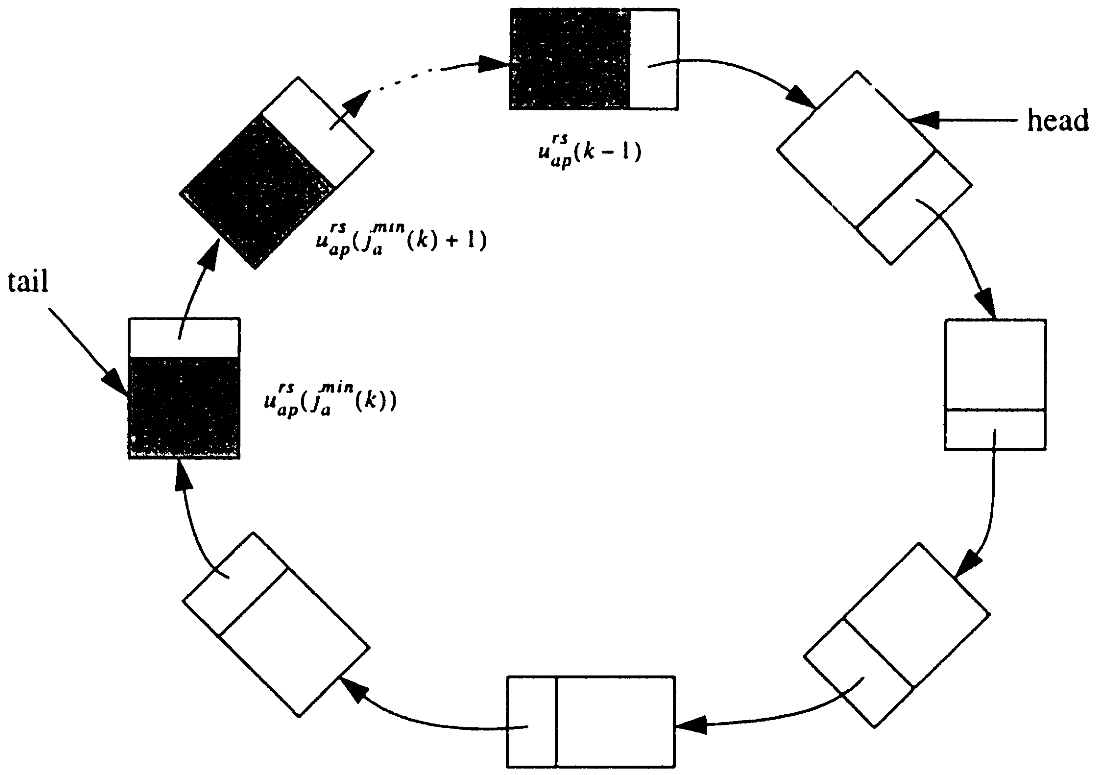

Figure 5-5: A Circular Queue List

In Figure 5-5, the partly shaded elements indicate a current portion of historic entrance flow rates stored in RAM. The unshaded elements are available for storing future values of link entrance flow rates. At each interval k, a value of $u_{ap}^{rs}(k)$ is added to the head of the list and the head pointer moves to the next element. If the head pointer catches up with the tail pointer, the list is said to be full and is then expanded.

When $j_a^{min}(k)$ increases, the element pointed by the tail pointer becomes available (it changes to "unshaded" state) and the tail pointer advances to the next element.

The circular queue list data structure makes the operation of expansion easier and faster than using an array. The only minor disadvantage of using the circular queue list is that extra memory is needed to store pointers.

Chapter 6

Computational Examples

In Chapter 3, we presented the formulation of a DTA model. In Chapter 4, we developed two network loading algorithms and a DTA algorithm. In Chapter 5, we presented computer implementations of these solution algorithms. The objectives of this this chapter are:

1. to demonstrate that the proposed DTA model indeed gives the results that satisfy the route choice conditions,

2. to better understand the performance of the two network loading algorithms and the convergence behavior of the DTA algorithm,

3. to show that the developed software system can solve a real-world problem in a running time faster than real time, and

4. to evaluate the performance of the computer implementations and quantify impacts of the implementation techniques presented in the previous chapter.

To achieve the above objectives, we study two examples. The first example uses small hypothetical network. This small example serves dual purposes. On one hand, we use it to analyze the algorithms and show that the DTA model can give a meaningful solution. On the other hand, we use it to illustrate the usage of the software system. The second example is a real-world application representing the Amsterdam

A10 beltway network. This example is used to evaluate the computational performance since the network has a large size.

This chapter is organized as follows. Section 6.1 presents a study using a small example. Section 6.2 presents the study of the Amsterdam A10 beltway network example. Section 6.3 evaluates the computational performance of the software system and analyze impacts of the implementation techniques developed in this thesis.

6.1 A Small Example

In this section, we use a small example to show how to use the software system and to do computational analysis of various solution algorithms. We choose this small example for the following reasons. First, it is convenient to show the inputs and outputs of the software system so that one can learn how to use it. Second, it is easier to analyze the results and understand the behaviors of the solution algorithms.

We first show how to use the software system. Then, we compare the two dynamic network loading algorithms, the I-Load and the C-Load algorithms, from a computational point of view. We use the C-Load algorithm to analyze the convergence behavior of the DTA algorithm and demonstrate that our DTA software system is able to give meaningful solutions.

6.1.1 Example Description

In this subsection, we describe the test network, link performance functions, O-D demands and paths for this example.

The network is a grid composed of 9 nodes and 12 links (see Figure 6-1). The attributes of the links are given in Table 6.1 on Page 88. Link travel times are determined using a speed-density relationship given by Equation (3.11) in Chapter 3. The parameters used in this function are the following: minimum speed $w_a^{min} = 5$ miles/hour, $\alpha = 1.4$ and $\beta = 3.2$.

A total of seven O-D pairs and fourteen paths are considered (see Table 6.2 on Page 89). Some O-D pairs have only one path, but they should be included since the

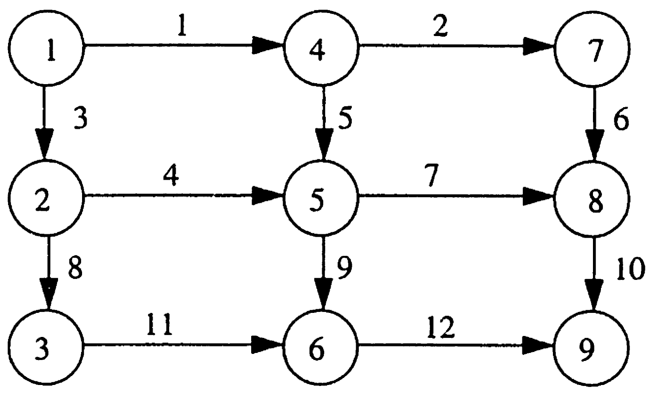

Figure 6-1: The Network for Example 1

Table 6.1: Attributes of The Links

Link	Length (miles)	Free-Flow Speed (miles/hour)	Jam Density (vehicles/mile)
1	2.0	60	210
2	2.0	60	210
3	1.5	60	210
4	1.8	60	210
5	1.5	60	210
6	2.1	60	210
7	1.7	60	210
8	1.8	60	210
9	2.0	60	210
10	2.3	60	210
11	2.2	60	210
12	2.5	60	210

Table 6.2: O-D Pairs and Paths

O-D Pair	Path
(1,9)	1,2,6,10
	1,5,7,10
	1,5,9,12
	3,4,7,10
	3,4,9,12
	3,8,11,12
(1,5)	1,5
	3,4
(5,9)	7,10
	9,12
(1,3)	3,8
(3,9)	11,12
(1,7)	1,2
(7,9)	6,10

flows from those O-D pairs may affect the traffic conditions on other paths.

The above network data are stored in a file. This file is described in Appendix B.

The departure time period is $[0, 300]$ seconds. The time-dependent O-D demands are given by

$$f^{rs}(t) = \eta^{rs}[\frac{t}{75} - (\frac{t}{150})^2], \qquad t \in [0, 300].$$

Note that all O-D demands have similar profile. The coefficient η^{rs} is used to scale these O-D demands to different levels. The values of η^{rs} for each O-D pair (r, s) are given in Table 6.3. The O-D demands are stored in a data file. This file is described

Table 6.3: Value of η^{rs} for Each O-D Pair

O-D Pair	1	2	3	4	5	6	7
η^{rs}	1.7	0.6	0.6	0.25	0.2	0.2	0.7

in Appendix B.

The above O-D demands can be distributed among three types of users in different ways, which we call scenarios. The network conditions may vary with different combinations of user types. In order to study the effect of a scenario on network conditions, we select four hypothetical scenarios. Table 6.4 shows the four scenarios. The first three scenarios represent the situations where all users fall into one type. Scenario D considers a mix of three types of users. We also assume that type 1 users are equally assigned among all paths connecting an O-D pair. A scenario is specified in a scenario file. This file is described in Appendix B.

Table 6.4: The Four Scenarios

Scenario	% of type 1 users	% of type 2 users	% of type 3 users
A	100	0	0
B	0	100	0
C	0	0	100
D	20	30	50

The route choice probability for type 2 users is determined from a C-Logit model described in Appendix A. We use the following values for the model parameters: $\theta = 0.1, \beta_0 = 1.0$ and $\gamma = 2.0$. These parameters are stored in a model parameter file. This file is described in Appendix B.

6.1.2 Usage of the Software System

Before analyzing the results, we briefly show how to use the software system to get the results. Before running the software, one needs to create the necessary input files. We have mentioned these files when we described the example in the previous subsection. Below, we give a summary of the contents of these input files:

- network data file, which contains O-D pairs, links and paths data,
- O-D demand file, which contains time-dependent O-D trips,
- scenario file, which contains information about users' behavior types and demand proportions for each type of users,
- model parameter files, which contains parameters for the C-Logit model, and

- control file, which specifies the file names of the above files, maximum number (N) of iterations, the number (M) of δ intervals with a Δ interval, and the demand period.

The formats of these files are described extensively in Appendix B.

The outputs from the software include:

- memory usage,
- running time,
- time-dependent link loads and travel times, and
- time-dependent path flow and travel times.

These outputs are sent to the files specified in the control file.

The software is run with the control file name as the only argument. After the program terminates, the results are saved in the specified output files. These results can be further analyzed by using some analysis software, such as MATLAB.

6.1.3 Comparison of the I-Load and C-Load Algorithms

The DTA algorithm needs to call a dynamic network loading algorithm at each iteration. Thus, the performance of the DTA algorithm depends on the dynamic network loading algorithm used. In Chapter 4, we developed two dynamic network loading algorithms, namely the I-Load algorithm and the C-Load algorithm. Before using the DTA algorithm to generate the results, we need to choose a good network loading algorithm.

In Chapter 4, we have compared the I-Load algorithm and the C-Load algorithm from a theoretical point of view. Here, we compare these two algorithms from a computational point of view. Specifically, we compare the path travel times generated by these two algorithms.

The I-Load algorithm involves an iterative process. The resulting path travel times may depend on the number of iterations. Before comparing these two algorithms, we first study the effect of the number of iterations on path travel times generated by the I-Load algorithm.

Figure 6-2 shows the travel times of path 1 for 1, 5, and 10 iterations. From this figure, we see that there is a large difference in path travel time corresponding to 1 iteration and 5 iterations. However, the change in path travel time is not significant for 5 iterations and 10 iterations. Therefore, we will perform 5 iterations for the I-Load algorithm during the comparison of these two algorithms.

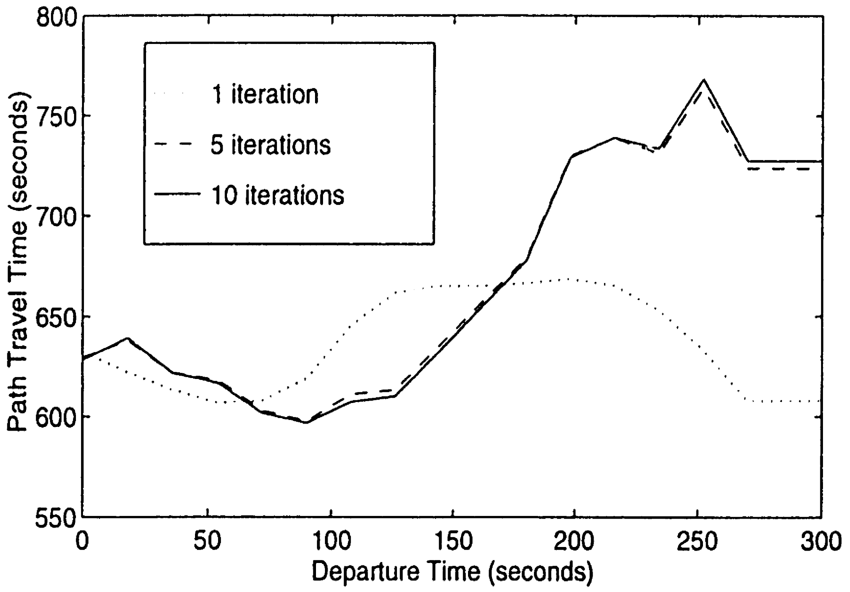

Figure 6-2: Travel Time of Path 1 for 1, 5 and 10 Iterations

The minimum link travel time Δ for this network is 90 seconds. The time interval length may have some effect on the precision of the results. Therefore, we compare the two network loading algorithms with two interval lengths: $\delta = \frac{\Delta}{5} = 18$ seconds and $\delta = \frac{\Delta}{10} = 9$ seconds. Figure 6-3 and Figure 6-4 shows the travel times of path 1 for O-D pair (1,9) as a function of departure time.

The results show that (1) the path travel times are smoother when the interval length decreases; (2) both algorithms generate a similar path travel time pattern, but the path travel time from the I-Load algorithm is lower than that from the C-Load algorithm.

Since the I-Loag algorithm is a heuristic, it may not give a unique solution, or it may not even converge. The solution property of the I-Load algorithm is among the subjects of ongoing research projects. In the following analysis of the DTA algorithm, we will use the C-Load algorithm for the DTA algorithm to perform network loading tasks.

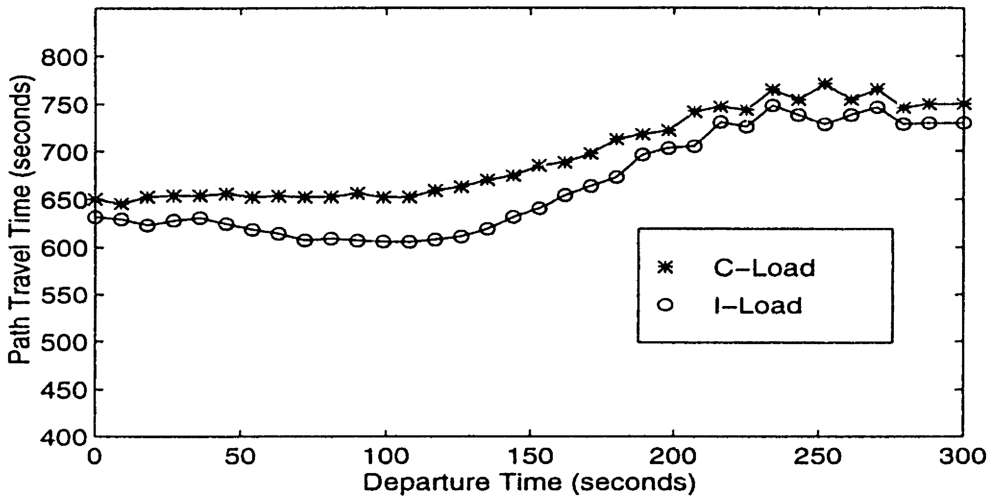

Figure 6-3: Path Travel Times by I-Load and C-Load for δ = 18 seconds (Δ = 90 seconds)

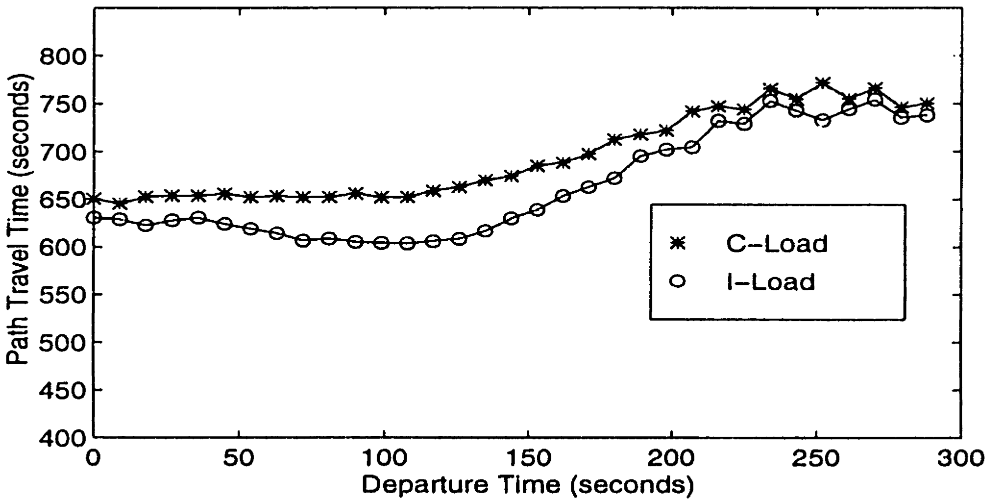

Figure 6-4: Path Travel Times by I-Load and C-Load for δ = 9 seconds (Δ = 90 seconds

6.1.4 Convergence of the DTA Algorithm

We analyze the DTA algorithm with the measures of deviation given in Chapter 4. We will study the convergence behavior of the DTA algorithm under Scenario D, since both the aggregate measures D_2 and D_3 can be computed. The results are presented in Figure 6-5 and 6-6 on Page 94.

Figure 6-5 and 6-6 show that D_2 and D_3 tend to decrease as more iterations are performed. This implies that the path flows approach a solution to the DTA model as the number of iterations increases. However, the values of these measures can go up or down from iteration to iteration. There also exists a tail effect for both D_2 and D_3. That is, after about 10 iterations, the marginal change in both D_2 and D_3 decreases as more iterations are performed.

We think that the convergence behavior shown above is common to all DTA algorithms based on an MSA method for updating path flows. Therefore, we will not repeat this analysis for the Amsterdam example.

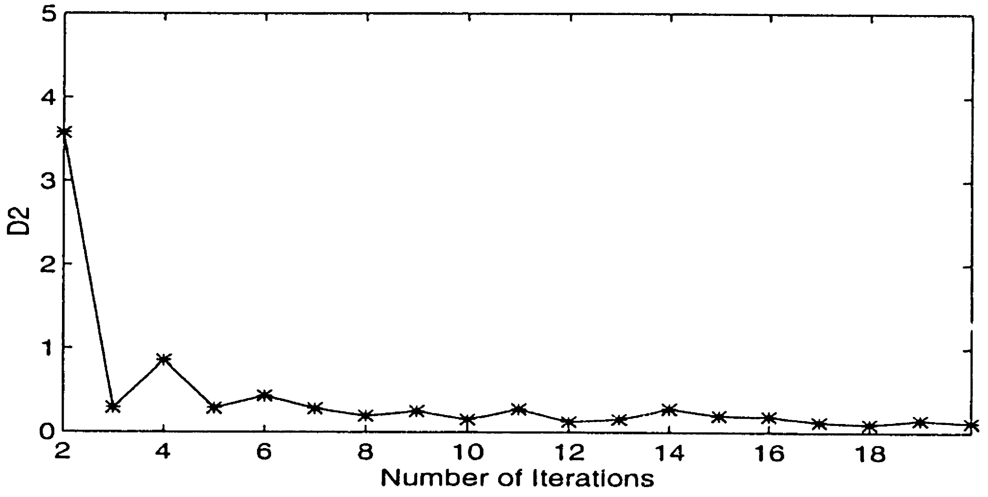

Figure 6-5: Measure of Deviation for Type 2 Users

6.1.5 Verification of the DTA model

With this small example, we show that the developed DTA implementation can give meaningful results, that is, the results are compatible with the demand scenarios. The path flows and travel times are obtained by performing 10 iterations for each of Scenarios B, C and D. Since Scenario A is equivalent to a network loading problem, only one iteration is necessary.

Since it is cumbersome to report the results for all paths, we arbitrarily choose two paths, say path 1 and 2, of O-D pair 1. The path travel times and flows for these two paths are presented in Figures 6-7 through 6-10 on Page 96 and 97 for each scenario.

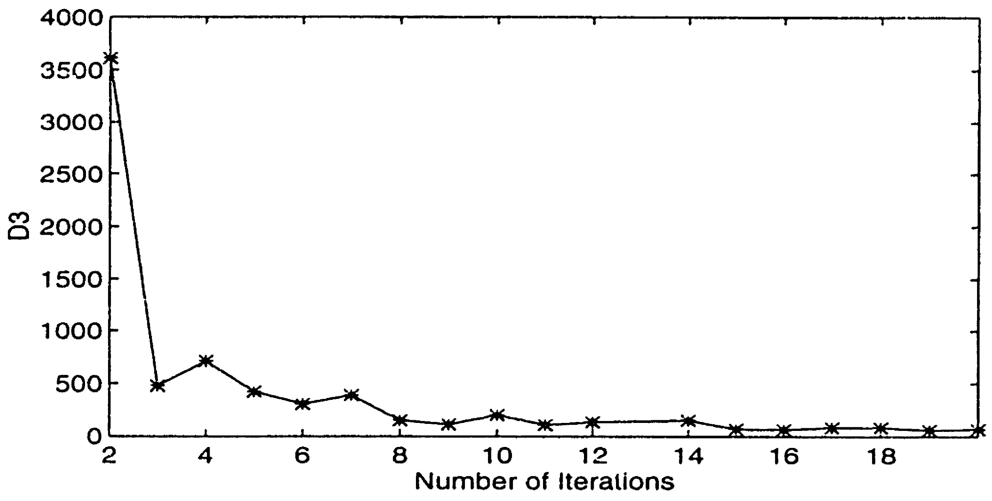

Figure 6-6: Measure of Deviation for Type 3 Users

From Figure 6-7, we see that the curve of path flow and the curve of travel time are very different. This shows that the transformation from path flow to path travel time by the dynamic network loading model is nonlinear. We also observe that the travel time peak is shifted to the later time period. This indicates that there is a backward propagation of congestion.

A larger difference in path travel time is seen for Scenario A and B, while the path flows show small difference (no difference in Scenario A by the hypothesis). For Scenario C and D, the path travel times tend to be equalized. This is due to the route choice behavior of type 3 users who choose only minimum routes.

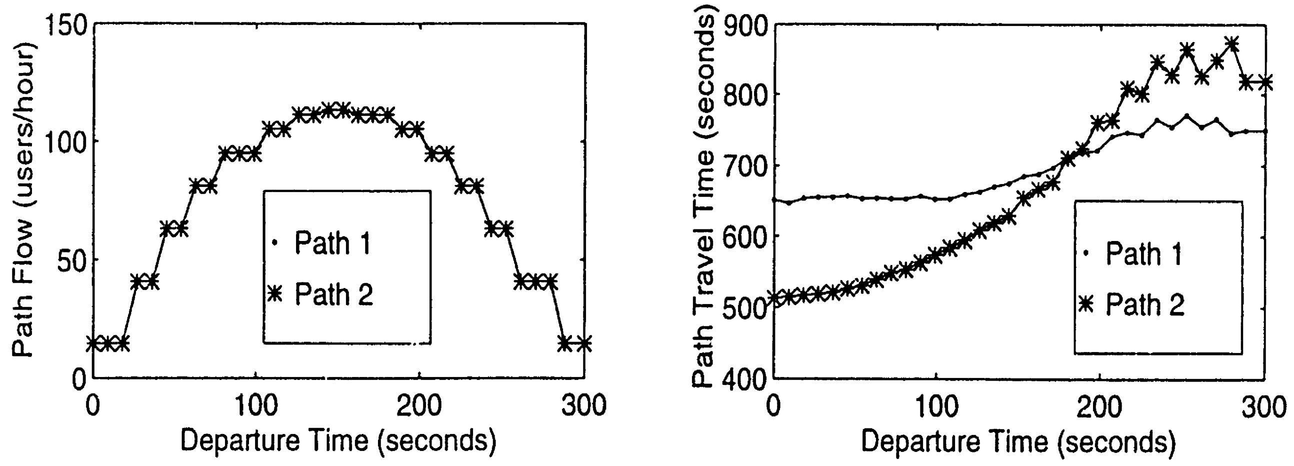

Figure 6-7: Path travel Times and Flows of O-D Pair 1 under Scenario A

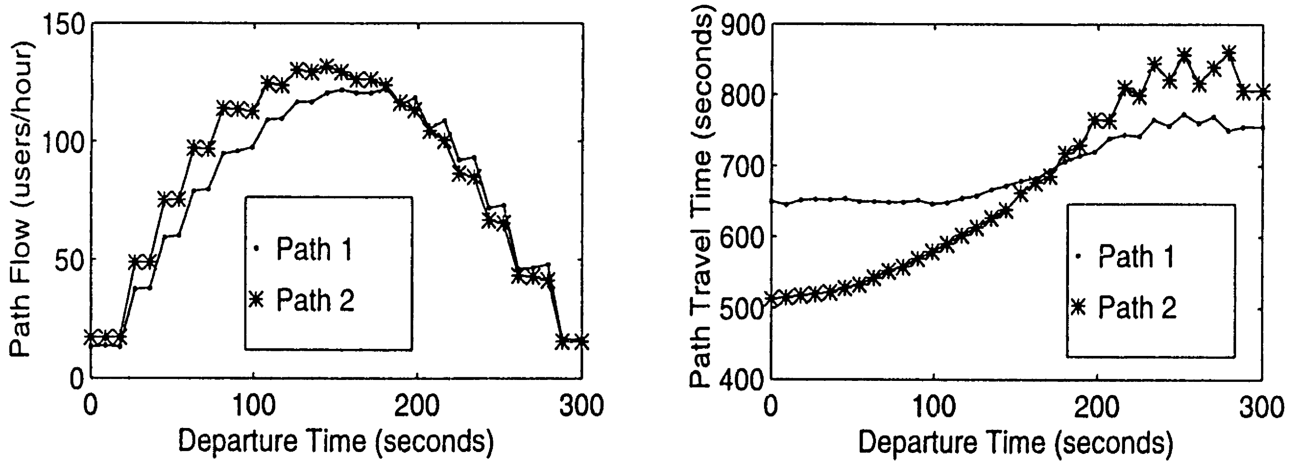

Figure 6-8: Path travel Times and Flows of O-D Pair 1 under Scenario B

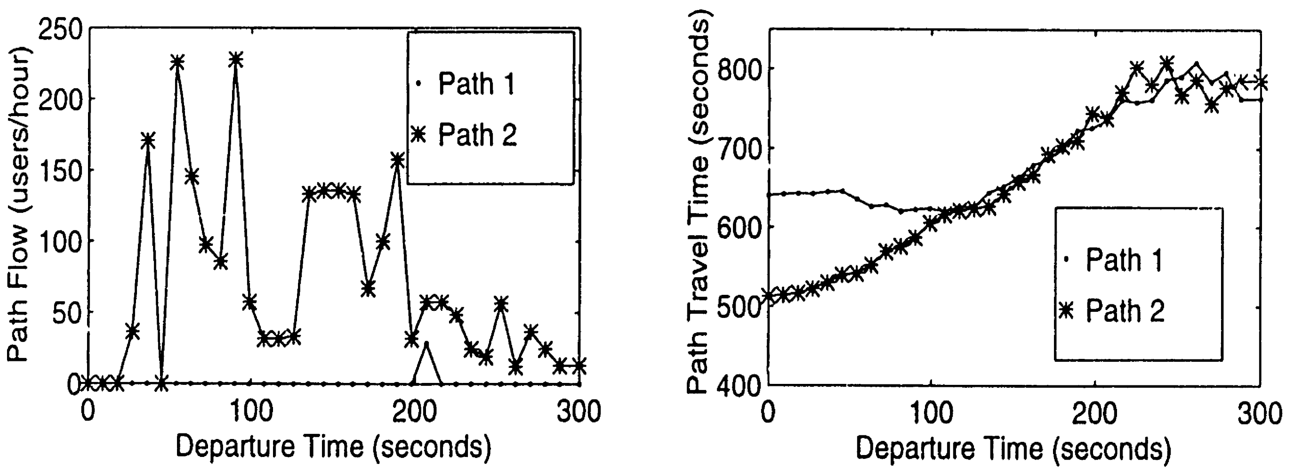

Figure 6-9: Path travel Times and Flows of O-D Pair 1 under Scenario C

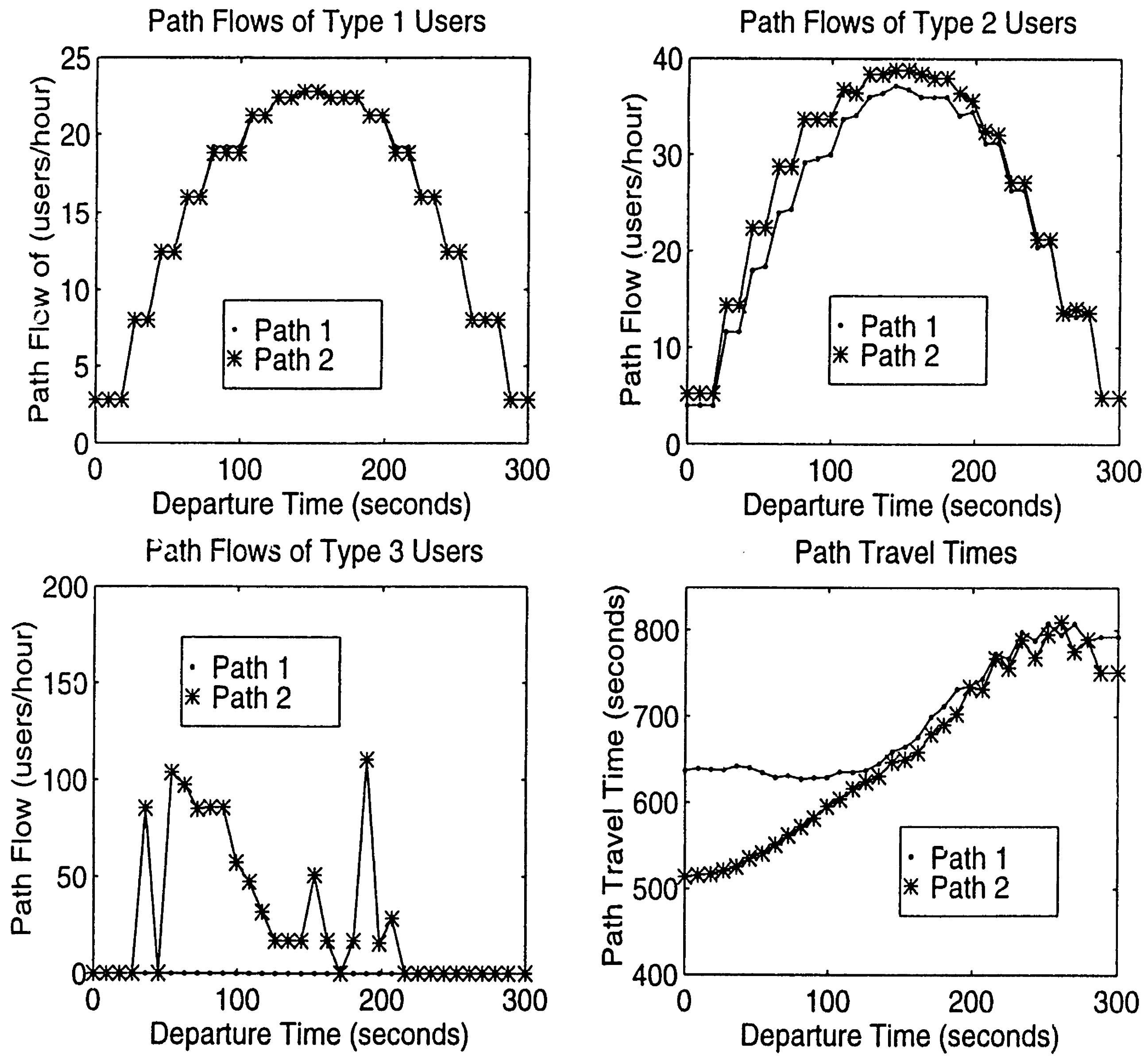

Figure 6-10: Path travel Times and Flows of O-D Pair 1 under Scenario D

6.2 The Amsterdam A10 Beltway Example

In this section, we use the Amsterdam A10 beltway as another computational example. The objectives of doing this example are (1) to demonstrate that our software system can solve a realistic DTA problem in a running time that is faster than the real time, (2) to show that the results generated by the software system are meaningful, and (3) to evaluate the computational performance of the software system and the impacts of the implementation techniques presented in the previous chapter.

6.2.1 Example Description

The application example was recently studied by Yang[48] using simulation-based models. All the data are borrowed from Yang[48]. Yang gives an extensive description of this application. Below we present a brief description of the example.

The Amsterdam A10 beltway (see Figure 6-11) consists of two 32-km freeway loops which intersect with five major freeways and have 20 interchanges of various sizes (75 ramp intersections). The network serves local and regional traffic and acts as a hub for traffic entering and exiting north Holland. The network is subject to considerable recurrent congestion, especially on the northern and southern parts of A10. Because route choices can affect traffic conditions, dynamic traffic assignment management could potentially improve traffic conditions during peak periods. There are 196 nodes and 310 links in the network. The number of O-D pairs is about 1000. The link travel time function is the same as the one used in the small example.

The time-dependent O-D trips were originally estimated by Ashok[3] between 20 centroids based on speeds and counts measured at 65 sensor stations. Figure 6-12 shows the profile of the total O-D demands. The time period for the O-D trips is from 7:15AM to 9:15AM (morning peak period). The centroid-based O-D trips are distributed to the corresponding "entrance" and "exit" nodes of the network, in order to derive departure rates for approximately 1000 node-based O-D pairs.

For every O-D pair in the network, we use the same subset of paths generated by MITSIM, a microscopic traffic simulator developed by the Intelligent Transportation

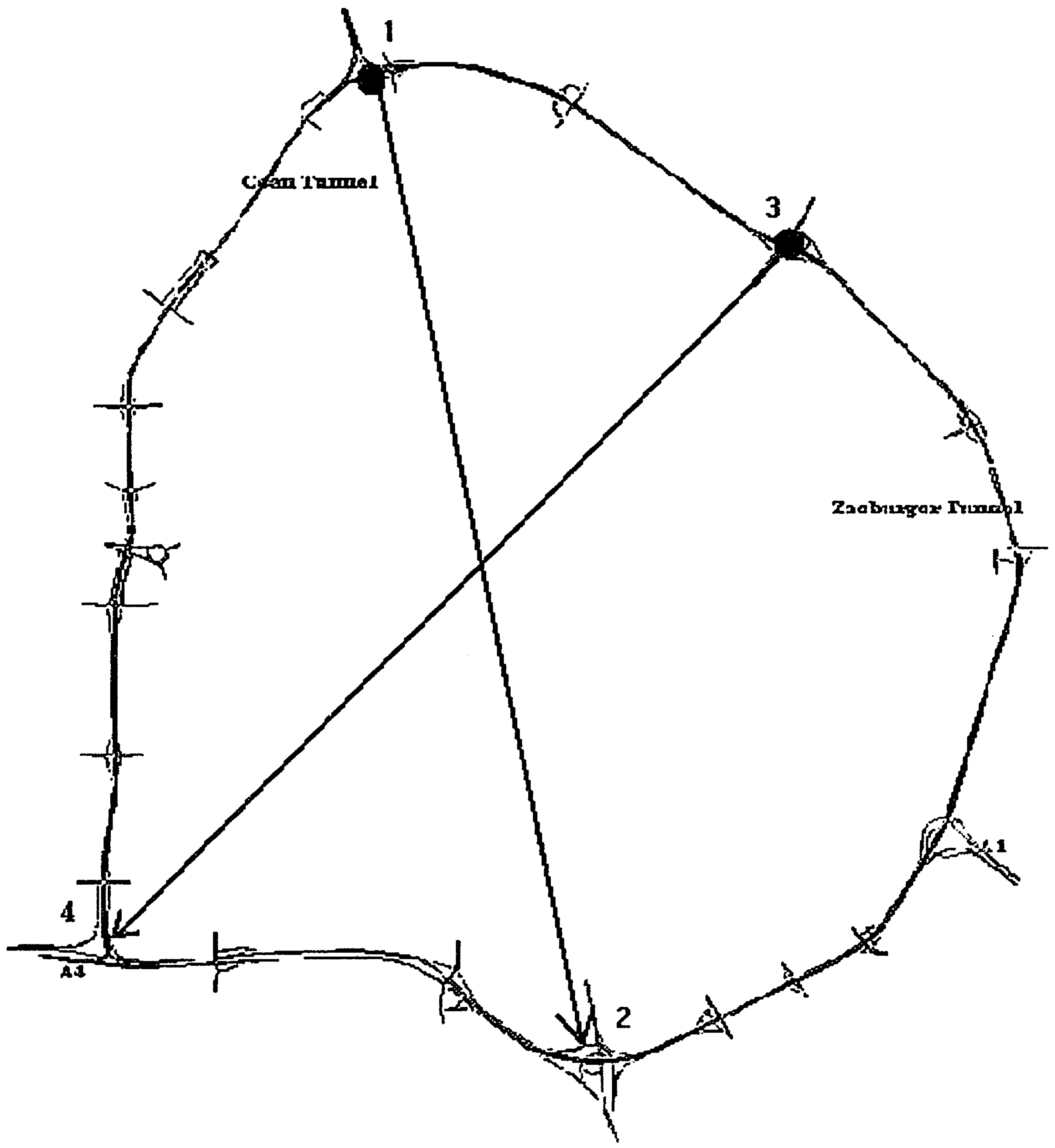

Figure 6-11: The Amsterdam A10 Beltway Network and Two O-D Pairs (1,2) and (3,4)

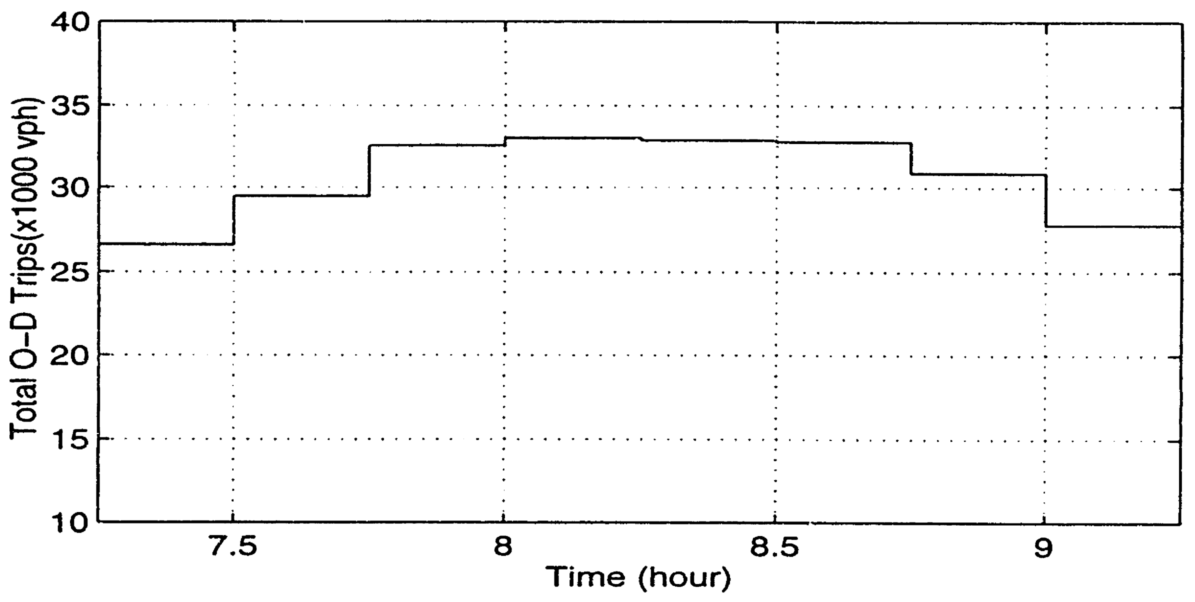

Figure 6-12: The O-D Trip Profile

Systems Program at Massachusetts Institute of Technology[48]. Most O-D pairs in the A10 network have two routes; therefore, the total number of paths is about 1500.

6.2.2 Results

For this example, we show the average path travel time under each scenario and path travel times for two representative O-D pairs (1,2) and (3,4) (see Figure 6-11). The purpose of this section is to demonstrate that our software system is able to yield meaningful results for this real-life application. The evaluation of computational performance will be presented in the next section.

We first define two average travel times: the overall average travel time and the time-dependent average travel time. The overall average travel time is computed by averaging the path travel times over all users and departure times. The time-dependent average travel time is computed by averaging the path travel times over all users for a given departure time. Thus, the time-dependent average travel time can be seen as a function of departure time.

The overall average travel times for the four scenarios are given in Table 6.5. Under Scenario A, the average travel time is 7.19. Under all other scenarios, the

Table 6.5: Comparison of Average Travel Times

	Scenario A	Scenario B	Scenario C	Scenario D
Average Travel Time	7.19	6.38	6.31	6.36

average travel time is about 6.3 minutes. The reduction in overall average travel time can be explained by type 2 and 3 users in Scenario B, C and D. Also note that the overall average travel time decreases as the proportion of type 3 users increases.

We now analyze the time-dependent average travel time for the four scenarios. Figure 6-13 shows the average travel time differences among the four scenarios at different departure times. Under Scenario A, a severe congestion starts at about

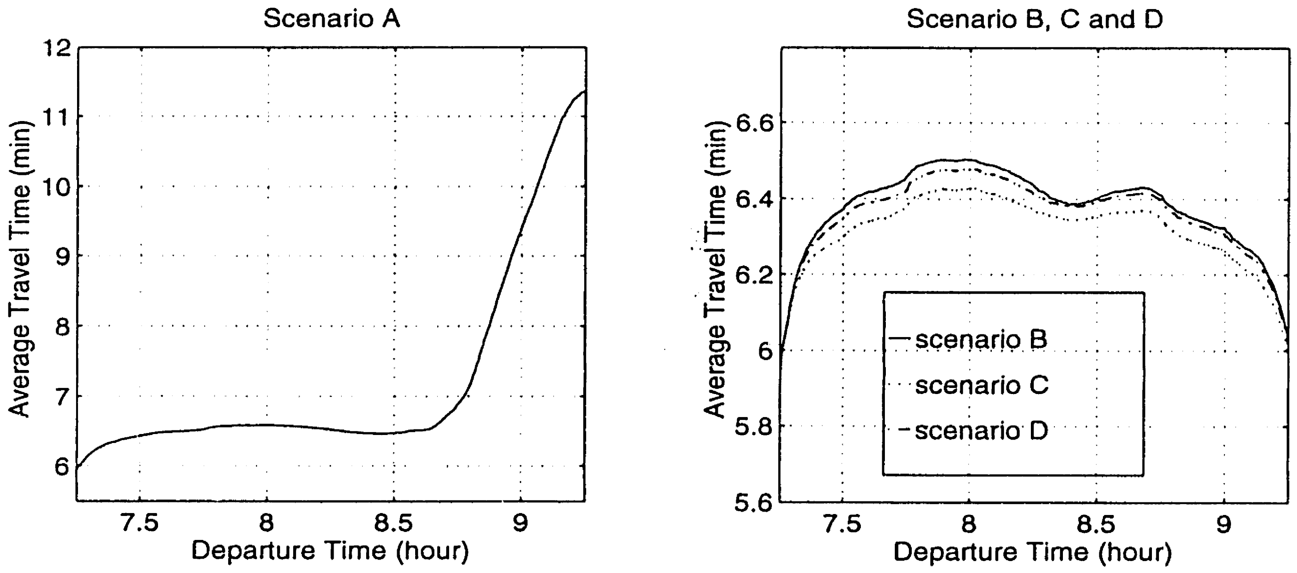

Figure 6-13: Time-Dependent Average Travel Times

8:45AM. The average travel time increases quickly. The other three scenarios do not show this congestion. We also see that the average travel time is lower at every departure time when the proportion of type 3 users is higher.

Figure 6-14 and Figure 6-15 shows the travel times of the two paths of the two representative O-D pairs (1,2) and (3,4) under each scenario. The figures show that the difference in travel time between two paths of the O-D pairs tends to be smaller as the percentage of type 3 users increases.

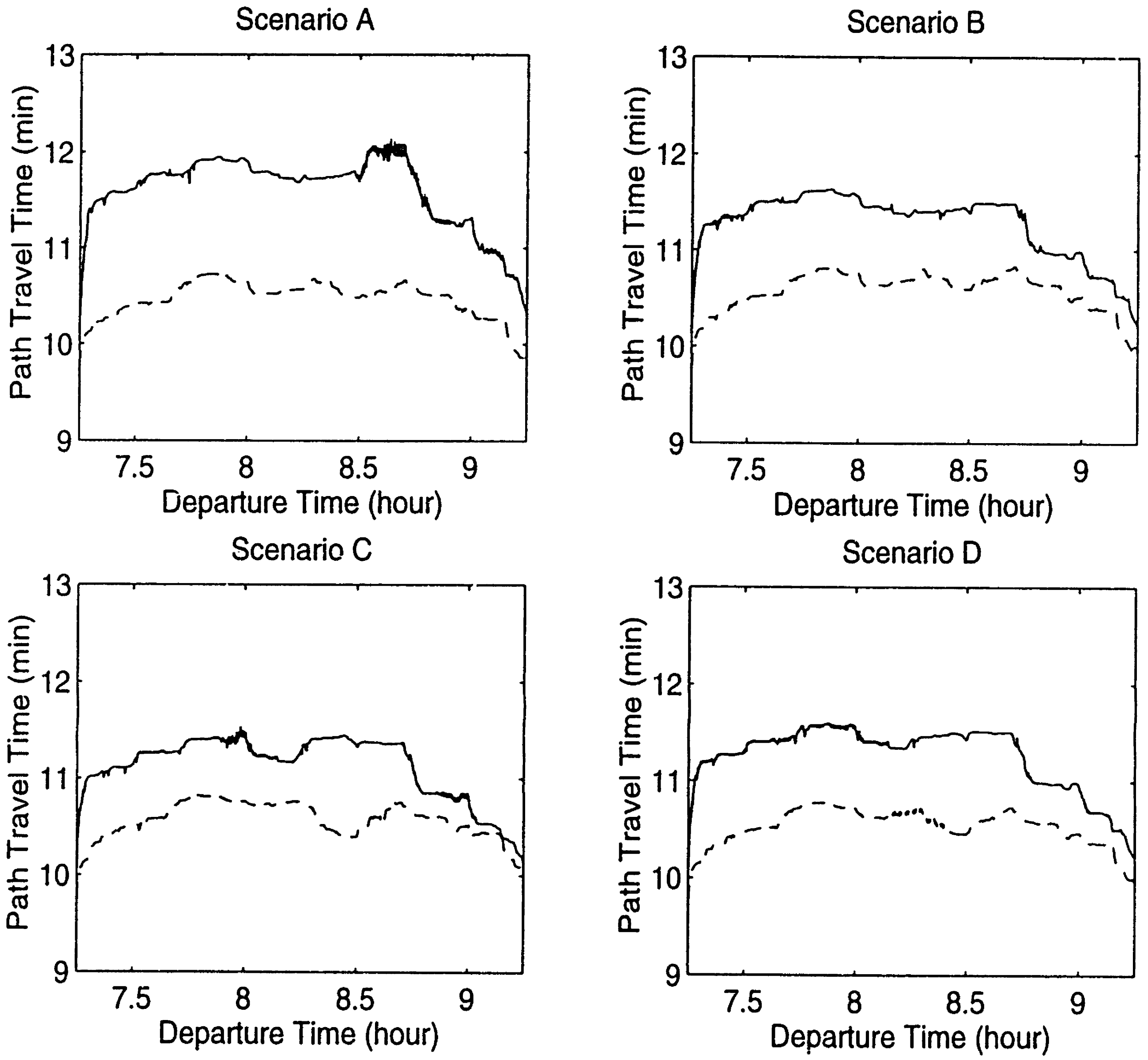

Figure 6-14: Path Travel Times of O-D Pair (1,2) under the Four Scenarios

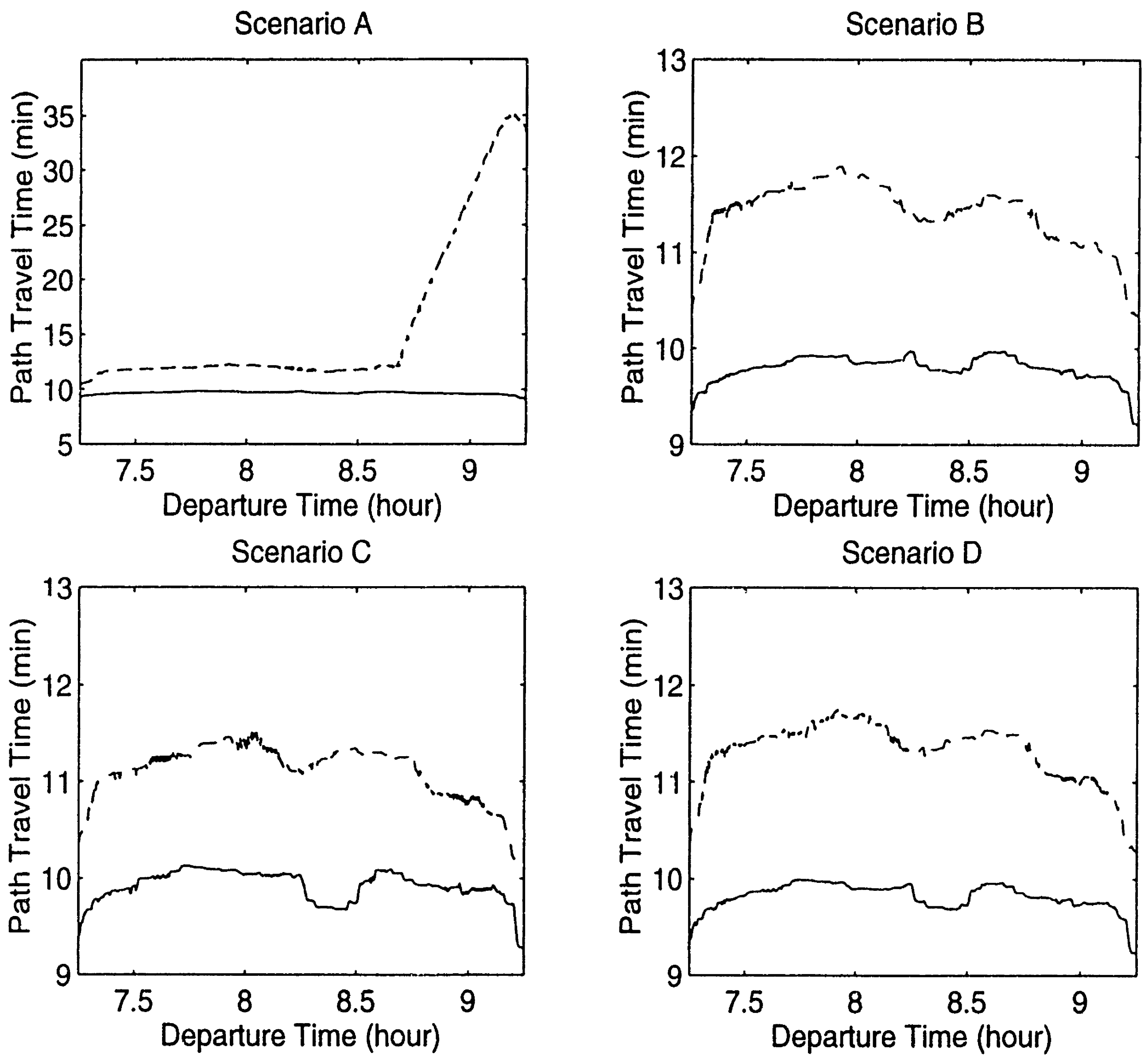

Figure 6-15: Path Travel Times of O-D Pair (3,4) under the Four Scenarios

6.3 Evaluation of Computational Performance

In Chapter 5, we stated that a primary objective of this research work is to develop a software system to solve realistic DTA problems in a running time that is faster than real time. In this section, we evaluate the computational performance of our computer implementations by using the Amsterdam network introduced in the previous section. In particular, we are interested in quantitatively evaluating the impacts of the new implementation methods presented in Chapter 5.

The Amsterdam example was solved using an SGI Indy Workstation with 150MHz CPU and 64MB memory. The demand period is 2 hours, the interval length is 3.5 seconds. The total analysis period returned by the system is 2 hour and 20 minutes (or 2357 intervals). The memory used by each category of the variables in this application example are listed below:

- Link variables: 25MB,
- Path variables: 34MB,
- Link-path flow variables: 0.5MB, and
- Entrance flow history list: 0.3MB

At each DTA iteration, the network loading step, using the C-Load algorithm, takes about two minutes out of three minutes. These results show that the dynamic network loading is the most computationally expensive step in the DTA algorithm.

To the best of our knowledge, no DTA problem of size similar the size of the Amsterdam example has been solved in the literature using analytical approaches. As we will show below, the achieved performance of the software system developed in this thesis is attributable to the the following implementation techniques:

- consolidation of link-path flow variables, and
- efficient computation of exit flows.

We now evaluate the impacts of each technique on the computational performance. In the C-Load algorithm, the number of link-path flow variables on a link is equal to the number of paths that pass through that link. Initially, the total number of these variables is 27,100. After running the consolidation procedure, this number was reduced to 5,770. Therefore, the saving ratio is about 4.5 in both memory and running time.

The method for computing exit flows has a significant impact on both memory and running time. In the Amsterdam example, the number of intervals for the analysis period is 2357. A usual method would store the value of entrance flow for each of these intervals. By using the new method for computing exit flows, an average of only 15 intervals of past entrance flow values were necessary to store in memory per LPV object. Therefore, the memory and running time saving ratio is in the order of 157.

Table 6.6 summarizes the impacts of these two methods on memory and running time. These results demonstrate that the two techniques have a substantial impact

Table 6.6: Memory Requirement and Running Time for a Dynamic Network Loading Step with and without the Two Techniques Introduced in Chapter 5

Category	with techniques	without techniques (estimated)
Link Variables	25 MB	25 MB
Path Variables	34 MB	34 MB
Link-Path Flow Variables	0.5 MB	2.25 MB
Entrance Flow History List	0.3 MB	210 MB
Total Memory	60 MB	276 MB
Running Time	2 minutes	23 hours

on computational performances. Without these techniques, contemporary computers may not be able to store all variables in the Random Access Memory (RAM). Even if a computer with large enough RAM is used, it would take almost a day to perform one network loading task.

Contrary to what is believed in the folklore in this research area, the developed analytical software system is even competitive with state-of-art simulation-based approaches. To perform one network loading task for the Amsterdam example, two simulation-based software tools and our analytical software system yields the following running time:

- MITSIM: 1.56 times slower than real time[48],
- MesoTS: 16 times faster than real time[48], and
- our software system: 70 times faster than real time.

Five iterations of the DTA algorithm returned a satisfactory solution. This requires about 20 minutes of CPU time. The DTA algorithm is then about 7 times faster than the real time for the Amsterdam example.

Chapter 7

Summary and Future Research

This final chapter summarizes the thesis work and provides directions for future research.

7.1 Summary

The study presented in this thesis is mainly motivated by the applications of the dynamic traffic assignment (DTA) model to Intelligent Transportation Systems (ITS). The main contributions of the thesis include: (1) proposing a framework for the dynamic traffic modeling, (2) formulating a flow-based DTA model based on the proposed modeling framework, (3) developing solution algorithms for the DTA model, and (4) developing efficient computer implementations of the solution algorithms.

The modeling framework contains four components: (1) users' behavior model, (2) dynamic network loading model (3) link performance model, and (4) path generation module. Each of these components can be formulated independent of each other. The interaction between these models is explicitly represented. The framework has a modularized structure which can provide better flexibility in model formulations, algorithms development and computer implementations.

A DTA model is formulated based on this framework. The model overcomes some limitations present in existing models within the context of ATMS/ATIS. The model captures different users' route choice behaviors. Link performance is modeled more accurately by using different link travel time models for uninterrupted and interrupted traffic flows.

The formulation of the DTA model is based on a proposed framework for the DTA problem. The users' behavior model considers three types of users' route choice behaviors. Variational Inequality (VI) is used to model these route choice behaviors. The dynamic network loading model is formulated as a system of equations describing link dynamics, flow conservation, flow propagation and boundary constraints. We show that there exists a solution to this system of equations. The network conditions determined by the dynamic network loading model are used to generate path travel times, which are input to the users' behavior model to update the path flows. A solution to the DTA model can be found by performing a number of network loading and path flow update iterations.

The solution algorithms we have developed include: (1) two different dynamic network loading algorithms, (2) a route choice algorithm, and (3) a DTA algorithm. One of the dynamic network loading algorithms, called I-Load, is based on an iterative process. The other, called C-Load, performs network loading in chronological order and no iterations are needed. The Method of Successive Average (MSA) is used for the path flow update in the DTA algorithm. At this moment, this solution algorithm is a heuristic.

We develop efficient computer implementations of the solution algorithms. The product of these computer implementations is a software system that is able to solve a DTA problem for realistic networks in a running time faster than real time. We design efficient data structures and methods to improve the running time and reduce memory usage.

We do two computational examples to demonstrate that the software system can not only give meaningful results, but also can find a solution much faster than the real time. For instance, a network loading task for the Amsterdam A10 beltway network was carried out in a running time 70 faster than real time on an SGI Indy 150 MHz Workstation. The fast running time is attributable to our new implementation techniques.

7.2 Future Research

Future research are needed to improve model formulations, algorithms and computer implementations. A number of directions are listed below.

Improving the Dynamic network loading model

A dynamic network loading model is a key component of the DTA model. In this thesis, the incidents are not considered in the dynamic network loading model. This is a limitation of the thesis and should be overcome in the future.

Incidents are not unusual in traffic network and can have a significant impact on traffic conditions. An important function of ATMS/ATIS is incident management. Therefore, a proper incident model should be added to the dynamic network loading model. To model an incident in analytical models, it is necessary to consider (1) queues on links, (2) link capacity constraints, and (3) spillback of flows. To address these issues, the proposed dynamic network loading model and solution algorithm may need to be restudied and generalized.

Improving flow updating method

In the current solution algorithm, MSA is used to update path flows at each iteration. The method is a heuristic and convergence to an optimum solution is not always guaranteed. The development of convergent and efficient flow updating methods is an open research area.

Updating subset of paths

To obtain more accurate results, the subset of paths between each O-D pair should be updated according to the changing network conditions. Paths may need to be added to the subset of paths. This should be possible by integrating a path generation module based on time-dependent shortest path algorithms[17, 18, 19].

Implementation refinements

The current implementation can be refined in several aspects. First, it is necessary to add a better user interface so that the program is easier to use and control. A Graphic User Interface (GUI) would be useful, which would allow users to input, modify and create input data from a menu. Users can also visualize the results on the screen. Second, the computer implementation can benefit from using parallel computing platforms, especially when the path generation module is incorporated.

Appendix A

A modified Logit Model

This appendix describes a discrete choice model for computing the route choice probabilities for type 2 users.

In order to compute the path flows for type 2 users, it is necessary to compute the probability that a path with a given travel time is chosen by a type 2 user. Random utility choice models are usually used to determine the probability. Among the random utility choice models, the logit choice model is chosen for the current implementation because it has a closed form and can be efficiently calibrated from disaggragated data.

However, the conventional logit choice model can give unrealistic path choice probabilities because of the overlapping of the paths. To overcome path overlapping problems, a modified logit route choice model (C-Logit) proposed by Cascetta *et al.*[16] is adopted.

According to the C-Logit model, the probability that path p is chosen is given by:

$$P_p^{rs}(k) = \frac{exp[V_p^{rs}(k) - CF_p]}{\sum_{h \in K_{rs}} exp[V_h^{rs}(k) - CF_h]} \tag{A.1}$$

where the term $V_p^{rs}(k)$ denotes the systematic utility of path p and is given by

$$V_p^{rs}(k) = -\theta c_p^{rs}(k)$$

The CF_p, denoted as "commonality factor" of path p, is directly proportional to the degree of similarity (or overlapping) of path p with other paths belonging to K_{rs}. The role played by CF_p is clear: heavily overlapping paths have larger commonality factors and thus a smaller systematic utility with respect to similar, but independent paths.

One way to specify the commonality factor is as follows:

$$CF_p = \beta_0 \ln \sum_{h \in K_{rs}} \left(\frac{L_{hp}}{\sqrt{L_h L_p}}\right)^\gamma$$

where

L_{hp} = the length of links common to paths h and p

L_h = length of path h

L_p = length of path p

The model parameters θ, β_0 and γ need to be calibrated.

Appendix B

Formats of Input Data Files

This appendix describes the formats of the input data files. The input data files include:

- network data file,
- O-D trips data file,
- scenario file,
- model parameter file,
- control file.

The data files are stored in ASCII mode so that they can be viewed conveniently. Comments are allowed in the files. Comments should begin with "#" and occupy an entire line. The delimiter between the data values should be either spaces or tab characters.

B.1 Format of the Network Data File

The network data file contains the link, O-D pair and path information. The following is an example of the network used in Example 1 presented in Chapter 6.

```
# This is the network data file for Example 1
7       # number of O-D pairs
# [ID] [origin] [destination] [number of paths] [strat index]
1 1 9 6 0
2 1 5 2 6
3 5 9 2 8
4 1 3 1 10
5 3 9 1 11
6 1 7 1 12
7 7 9 1 13
12      # number links
# [Link ID] [Link type] [tail node] [head node] [lenght] [free speed] [max inflow] [max outflow] [density]
0 0 1 4 2.0 60 4000 4000 210
1 0 4 7 2.0 60 4000 4000 210
2 0 1 2 1.5 60 4000 4000 210
3 0 2 5 1.8 60 4000 4000 210
4 0 4 5 1.5 60 4000 4000 210
5 0 7 8 2.1 60 4000 4000 210
6 0 5 8 1.7 60 4000 4000 210
7 0 2 3 1.8 60 4000 4000 210
8 0 5 6 2.0 60 4000 4000 210
9 0 8 9 2.3 60 4000 4000 210
10 0 3 6 2.2 60 4000 4000 210
11 0 6 9 2.5 60 4000 4000 210
26      # number of subpaths in the subpath table
# [next link] [next path] [subpath number]
0 20  0
0 21  1
0 22  2
2 23  3
2 24  4
2 25  5
0 18  6
2 19  7
6 14  8
```

```
8 16  9
2 17  10
10 16 11
0 15  12
5 14  13
9 -1  14
1 -1   15
11 -1  16
7 -1   17
4 -1   18
3 -1   19
1 13   20
4 8    21
4 9    22
3 8    23
3 9    24
7 11   25
# end of the file
```

B.2 Format of the O-D Trip Data File

The O-D trip data file contains the number of trips for each O-D pair within each time interval. The number of intervals and the interval duration are indicated at the beginning of the file. The unit of interval duration is seconds. The O-D trips are given in a matrix format. Each row corresponds to an O-D pair and each column corresponds to a time interval. The order of the O-D pair should be the same as that in network data file. The following is an O-D trip data file used in Example 1 presented in Chapter 6.

```
# This is the O-D trip data file used in Example 1
# [number of intervals] [interval duration]
15 20.00
# [trip at interval 1] [trip at interval 2] ...
4.331852  12.189629 ... ...
```

```
1.528889 4.302222 ... ...
1.528889 4.302222 ... ...
0.637037 1.792593 ... ...
0.509630 1.434074 ... ...
0.509630 1.434074 ... ...
1.783704 5.019259 ... ...
```

B.3 Format of the Scenario Data File

The scenario file defines some settings of the program. A loading factor is specified to change the O-D trips given in the O-D trips file by the factor.

The following is an example of the scenario data file used in Example 1 presented in Chapter 6.

```
# This is the scenario data file in Example 1
3                          # number of types of users
1.0                        # loading factor
1.0 0.0 0.0                # demand proportion for type 1, 2 and 3
# the flow proportion for each path of each O-D pair (for type 1 user only)
0.1667 0.1666 0.1666 0.1666 0.1666 0.1666  # for OD pair 1
0.5 0.5                    # for OD pair 2
0.5 0.5                    # for OD pair 3
1.0                        # for OD pair 4
1.0                        # for OD pair 5
1.0                        # for OD pair 6
1.0                        # for OD pair 7
```

B.4 Format of the Model Parameter File

The model parameter file is used to specify model parameters. Currently, it contains the parameters of the C-logit model.

The following is an example of the model parameter file used in case study 1 presented in Chapter 6.

```
# logit model parameters
0.1             # scale parameter (theta) for type 2 users
1.0 2.0         # [beta0] [gama]
```

B.5 Format of the Control File

The control file specifies the file names for the network data file, O-D trip file, scenario file, model parameter file and the output files. The control file is provided to the program as the command line argument.

The following is an example of the control file used in Example 1 presented in Chapter 6.

```
# control file for Example 1
9n12l.net         # network data file name
9n12l-1.sce       # scenario file name
model.par         # model parameter file name
9n12l.dmd         # OD demand data file name
misc.out          # running time, memory output file name
pt.out            # path travel times output file name
pf.out            # path flows output file name
linkstat.out      # link statistics output file name
```

Appendix C

Sample Output Files

The DTA software system outputs the following four sets of results:

- running time and memory usage,
- link statistics, such as link loads and link travel times.
- path travel times, and
- path flows.

Each set of results is saved in one file in ASCII code, so that they can be viewed and edited directly with a text editor. The name of each file are specified in the control file (see Appendix B).

Below we show a portion of each output file from Example 1 in Chapter 6.

C.1 Running Time and Memory Usage Results

```
CPU time used:  0.19 seconds
memory used by link variables:  92976 bytes
memory used by path variables:  6300 bytes
memory used by link-path flow variables:  3016 bytes
memory used by entrance flow history list:  2408 bytes
```

C.2 Link Statistics

For each link, we output total entrance flow, total exit flow, link load, travel time and exit time for each interval. The unit of time is second. The results are arranged by the order of link number.

```
   -------------- link 1 ------------
 t(sec.)  Inflow  Outflow  Load    tau     Exit Time
 0.00     3.10    0.00     3.10    120.36  120.36
 18.00    3.10    0.00     6.19    120.97  138.97
 36.00    8.71    0.00     14.91   123.34  159.34
 54.00    13.47   0.00     28.37   128.44  182.44
 72.00    17.35   0.00     45.73   137.17  209.17
 .        .       .        .       .       .
 .        .       .        .       .       .
 .        .       .        .       .       .
   -------------- link 2 ------------
 t(sec.)  Inflow  Outflow  Volume  tau     Exit Time
 0.00     0.00    0.00     0.00    120.00  120.00
 18.00    0.00    0.00     0.00    120.00  138.00
 36.00    0.00    0.00     0.00    120.00  156.00
 54.00    0.00    0.00     0.00    120.00  174.00
 72.00    0.00    0.00     0.00    120.00  192.00
 90.00    0.00    0.00     0.00    120.00  210.00
 108.00   0.00    0.00     0.00    120.00  228.00
 126.00   1.11    0.00     1.11    120.09  246.09
 .        .       .        .       .       .
 .        .       .        .       .       .
 .        .       .        .       .       .
```

C.3 Path Travel Times

Travel times of all paths for an O-D pair are arranged in columns. The unit of travel time is second.

```
   === From origin 1 to destination 9 ===
---- Path travel times (seconds) --
 t(sec.)  Path-1  Path-2  Path-3  Path-4  Path-5  Path-6
 0.00     628.37  528.18  521.08  510.96  507.32  514.25
 18.00    639.26  536.19  524.84  519.95  511.59  517.90
 36.00    621.74  550.01  529.71  536.39  517.02  521.99
 54.00    616.61  550.86  541.81  540.89  530.16  534.35
 72.00    601.60  563.45  549.74  550.90  542.69  544.82
 90.00    596.88  572.90  569.47  567.70  561.60  560.22
 108.00   607.18  568.88  580.18  568.66  579.81  577.84
 .        .       .       .       .       .       .
 .        .       .       .       .       .       .
 .        .       .       .       .       .       .
```

C.4 Path Flows

Path flows of each type of user for an O-D pair are arranged in columns. The unit of the flows is number of vehicles/second.

```
   === From origin 1 to destination 9 ===
-- type 1 path flows ---
 t(sec.)   Path-1   Path-2   Path-3   Path-4   Path-5   Path-6
 0.00      0.007    0.007    0.007    0.007    0.007    0.007
 18.0      0.007    0.007    0.007    0.007    0.007    0.007
 36.00     0.007    0.007    0.007    0.007    0.007    0.007
 54.00     0.020    0.020    0.020    0.020    0.020    0.020
 72.00     0.020    0.020    0.020    0.020    0.020    0.020
 90.00     0.031    0.031    0.031    0.031    0.031    0.031
 108.00    0.031    0.031    0.031    0.031    0.031    0.031
 .         .        .        .        .        .        .
 .         .        .        .        .        .        .
 .         .        .        .        .        .        .
```

Bibliography

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms and Applications.* Prentice-Hall, Englewood, NJ, 1992.

[2] IVHS AMERICA. A strategic plan for intelligent vehicle-highway systems in the united states. Technical report, IVHS AMERICA, 1992.

[3] K. Ashok. *Estimation and Prediction of Time-Dependent Origin-Destination Flows.* PhD thesis, Massachusetts Institute of Technology, 1996.

[4] J. Barcelo, J. L. Ferrer, D. Garcia, R. Grau, M. Florian, I. Chabini, and E. Le Saux. Microscopic traffic simulation for its analysis. Technical report, Centre de Recherche sur les Transports, University de Montreal, 1996.

[5] J. Barcelo, J. L. Ferrer, R. Grau, I. Chabini, M. Florian, and E. Le Saux. A route based variant of the AIMSUN2 microsimulation model. In *Proceedings of the Second World Congress on Intelligent Transportation Systems*, Yokohama, Japan, 1995.

[6] M. Ben-Akiva. Dynamic network equilibrium research. *Transportation Research*, 29A(5):429–431, 1985.

[7] M. Ben-Akiva, M. Cyna, and A. de Palma. Dynamic model of peak period congestion. *Transportation Research*, 18B(4):339–355, 1984.

[8] M. Ben-Akiva, A. de Palma, and I. Kaysi. Dynamic network models and driver information systems. *Transportation Research*, 25A(5):251–266, 1991.

[9] M. Ben-Akiva and S. Lerman. *Discrete Choice Analysis: Theory and Applications to Travel Demand.* The MIT Press, Cambridge, MA, 1985.

[10] Transportation Research Board. *Highway Capacity Manual.* Transportation Research Board, National Research Council, Washington D. C., 1985.

[11] Piet H. L. Bovy. *Transportation Modeling for Tomorrow.* Delft University Press, The Netherlands, 1996.

[12] David Branston. Link capacity functions: A review. *Transportation Research*, 10:223–236, 1975.

[13] M. Carey. A constraint qualification for a dynamic traffic assignment model. *Transportation Science*, 20:55–58, 1986.

[14] M. Carey. Optimal time-varying flows on congested networks. *Operations Research*, 35(1):58–69, January-February 1987.

[15] M. Carey. Nonconvexity of the dynamic traffic assignment problem. *Transportation Research*, 26B(2):127–133, 1992.

[16] E. Cascetta, A. Nuzzolo, F. Russo, and A. Vitetta. A modified logit route choice model overcoming path overlapping problems: Specification and some calibration results for interurban networks. In *Proceedings of 13th International Symposium on Transportation and Traffic Theory*, Lyon, France, 1996.

[17] I. Chabini. A new algorithm for shortest paths in discrete dynamic network. In *Proceedings of the IFAC Symposium on Transportation Systems*, Chania, Greece, June, 1997.

[18] I. Chabini and M. Florian. High performance computation of temporal shortest routes for intelligent transportation system applications. In *Proceedings of the Second World Congress on Intelligent Transportation Systems*, Yokohama, Japan, 1995.

[19] I. Chabini, M. Florian, and E. Le Saux. Parallel and distributed computation of shortest routes and network equilibrium models. In *Proceedings of the IFAC Symposium on Transportation Systems*, Chania, Greece, June, 1997.

[20] E. Codina and J. Barcelo. Dynamic traffic assignment: Considerations on some deterministic modeling approaches. *Annals of Operations Research*, 60:1–58, 1995.

[21] S. C. Dafermos and S. C. Sparrow. The traffic assignment problem for a general network. *Journal of Research, National Bureau of Standards*, 73B:91–118, 1969.

[22] S. C. Dafermos and S. C. Sparrow. Traffic equilibrium and variational inequalities. *Transportation Science*, 14:42–54, 1980.

[23] T. L. Friesz, D. Bernstein, T. E. Smith, R. L. Tobin, and B. W. Wie. A variational inequality formulation of the dynamic network user equilibrium problem. *Operations Research*, 41(1):179–191, 1993.

[24] T. L. Friesz, J. Luque, R. L. Tobin, and B. Wie. Dynamic network traffic assignment considered as a continuous time optimal control problem. *Operations Research*, 37(6):893–901, November-December 1989.

[25] B. G. Haydecker and J. D. Addison. Analysis of traffic models for dynamic equilibrium traffic assignment. In *Proceedings of 13th International Symposium on Transportation and Traffic Theory*, Lyon, France, 1996.

[26] B. N. Janson. A convergent algorithm for dynamic traffic assignment. In *Proceedings of the 70th Annual Meeting of the Transportation Research Board*, Washington D. C., 1991.

[27] B. N. Janson. Dynamic traffic assignment for urban road networks. *Transportation Research*, 25B(2):143–161, 1991.

[28] R. Jayakrishnan, Anthony Chen, and Wei K. Tsai. Freeway and arterial traffic flow simulation analytically embedded in dynamic assignment. In *Proceedings of*

the 76th Annual Meeting of the Transportation Research Board, Washington D. C., 1997.

[29] Barbara J. Kanninen. Intelligent transportation systems: An economic and environmental policy assessment. *Transportation Research-A*, 30(1):1–10, 1996.

[30] I. A. Kaysi. *Framework and Models for the Provision of Real Time Driver Information.* PhD thesis, Massachusetts Institute of Technology, 1992.

[31] D. K. Merchant and G. L. Nemhauser. A model and an algorithm for the dynamic traffic assignment problem. *Transportation Science*, 12(3):183–199, August 1978.

[32] D. K. Merchant and G. L. Nemhauser. Optimality conditions for a dynamic traffic assignment model. *Transportation Science*, 12(3):200–207, August 1978.

[33] A. Nagurney. *Network Economics: A Variational Inequality Approach.* Kluwer Academic Publisher, Norwell, MA, 1993.

[34] Taylor N.B. CONTRAM5: An enhanced traffic assignment model, transportation laboratory, crowthorne. 1990.

[35] B. Ran and D. E. Boyce. *Dynamic Urban Transportation Network Models: Theory and Implications for Intelligent Vehicle-Highway Systems.* Springer-Verlag, 1994.

[36] B. Ran, D. E. Boyce, and L. J. LeBlanc. A new class of instantaneous dynamic user-optimal traffic assignment models. *Operations Research*, 41(1):192–202, 1993.

[37] B. Ran, Irene Y. Li, and Widodo B. D. Soetopo. An analytical path-based multi-class dynamic traffic assignment model. In *Proceedings of the 76th Annual Meeting of the Transportation Research Board*, Washington D. C., 1997.

[38] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design.* Prentice-Hall, Englewood, NJ, 1991.

[39] Y. Sheffi. *Urban Transportation Networks.* Prentice-Hall, Englewood, NJ, 1985.

[40] M. J. Smith. The existence, uniqueness and stability of traffic equilibria. *Transportation Research*, 13B:295–304, 1979.

[41] M. J. Smith. A new dynamic traffic model and the existence and calculation of dynamic user equilibria on congested capacity-constraint road networks. *Transportation Research*, 27B(1):49–63, 1993.

[42] B. Stroustrup. *The C++ Programming Language.* Addison Wesley, 1991.

[43] P. C. Vythoulkas. A dynamic stochastic assignment model for the analysis of general network. *Transportation Research*, 24B:453–469, 1990.

[44] B. Wie, T. L. Friesz, and R. L. Tobin. Dynamic user optimal traffic assignment on congested multidestination networks. *Transportation Research*, 24B(6):431–442, 1990.

[45] J. H. Wu, Y. Chen, and M. Florian. The continuous dynamic network loading problem: A mathematical formulation and solution method. Technical report, CRT-95-25, Centre de Recherche sur les Transports, University de Montreal, 1995.

[46] Y. Xu, J. H. Wu, M. Florian, P. Marcotte, , and D. L. Zhu. New advances in the continuous dynamic network loading problem. Technical report, CRT-96-26, Centre de Recherche sur les Transports, University de Montreal, 1996.

[47] Q. Yang. A microscopic traffic simulation model for ivhs applications. Master's thesis, Massachusetts Institute of Technology, 1993.

[48] Q. Yang. *A Simulation Laboratory for Evaluation of Dynamic Traffic Management Systems.* PhD thesis, Massachusetts Institute of Technology, 1997.

[49] Q. Yang and H. N. Koutsopoulos. A microscopic traffic simulator for evaluation of dynamic traffic management systems. *Transportation Research*, 4(3):113–129, 1996.