NETWORK TRAFFIC SIMULATION AND ASSIGNMENT: SUPERCOMPUTER APPLICATIONS

Hani S. Mahmassani, R. Jayakrishnan,
Kyriacos C. Mouskos, and Robert Herman

RESEARCH REPORT CRAY-SIM-1988-F

May 1988

CENTER FOR TRANSPORTATION RESEARCH

BUREAU OF ENGINEERING RESEARCH
THE UNIVERSITY OF TEXAS AT AUSTIN

ACKNOWLEDGEMENTS

Principal funding for this study came from a grant from Cray Research Inc. Additional funding and computer time was provided by the Department of Civil Engineering, Bureau of Engineering Research at the University of Texas at Austin. Computing resources for this work were provided by The University of Texas System Center for High Performance Computing (CHPC).

We are grateful for the help of consultants at the CHPC in the process of implementing and vectorizing the codes used in this study. In particular, the help of Jeanette Garcia is gratefully acknowledged. In addition, Spiros Vellas' contribution to the vectorization of the network equilibrium assignment routines has been invaluable.

The single-class network equilibrium assignment code used in this study is a modified version of a code initially provided by Dr. Fred Mannering, presently at the University of Washington, who modified a program originally supplied by Dr. Stella Dafermos at Brown University.

We are grateful to Dr. James C. Williams, presently at the University of Texas at Arlington, for the advice on getting the modified CDC version of NETSIM to run and on interpreting on some of his earlier data files and modifications to the program that he had implemented while a graduate student at the University of Texas at Austin.

The preparation and production of this report has been coordinated by Mrs. Carla F. Cripps.

PREFACE

This report presents the results of a study conducted to assess the potential offered by supercomputer architectures in solving large-scale network problems that arise in transportation systems analysis and planning. Two principal problems are addressed: 1) the microscopic simulation of vehicular traffic in urban street networks, and 2) the computation of equilibria in congested transportation networks.

The first problem arises in traffic science research and traffic engineering practice when it is desired to simulate traffic network conditions by keeping track of the movement and maneuvers of individual vehicles. Microscopic simulation codes have been available for many years; however, their applicability has been limited to very small portions of an area's network. Furthermore, earlier use of microscopic simulation for fundamental research into the characterization of network traffic behavior has been limited to contrived potentially unrealistic small networks. The objectives of the work presented herein are 1) to demonstrate the ability to consider large realistic urban traffic networks using supercomputer capabilities, 2) to provide some computational experience with such applications, and 3) to examine several research questions pertaining to network traffic theory using this enhanced capability to solve large congested networks.

The second problem addressed in this study arises in transportation analysis and planning when it is desired to determine the vehicular and/or passenger flows using each link of a particular urban network. It is known as the traffic assignment problem, and the solution sought by the algorithm considered herein satisfies "User Equilibrium" conditions. In addition to its inherent importance in transportation planning, the network equilibrium assignment algorithm is an essential routine in codes for the more general network design problem. In this study, we report on the results of limited local modifications of the code aimed at removing obstacles to its vectorization in order to take greater advantage of the CRAY X-MP/24 supercomputer's vectorizing capabilities.

This report is organized in two chapters, with the first corresponding to the microscopic traffic simulation problem and the second addressing the vectorization of the network assignment codes. Each chapter is self-contained and independent of the other, with the exception of a shared list of references.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF TABLES, Continued

CHAPTER 1. EXPERIMENTS WITH MICROSCOPIC SIMULATION OF TRAFFIC IN NETWORKS

1.1. INTRODUCTION

1.1.1. Motivation

Microscopic vehicular traffic simulation models are computer programs which simulate the behavior and flow of traffic by representing and keeping track of the maneuvers of the individual vehicles. By contrast, macroscopic models utilize relationships between aggregate descriptors such as concentration, average speed, etc., and in some cases keep track of platoons of vehicles. A large body of work on the mathematical relationships underlying both kinds of models was developed during the 1950's and 60's; the development of computer simulation codes of both types began in the late 60's.

Microscopic models, due to their attention to the minute details of the traffic, are computationally more demanding, in terms of time and memory, than macroscopic models. Of course, the former kind of models have the potential to: 1) represent more complicated situations, 2) yield more detailed information, and 3) give more precise and "correct" simulation results than the latter kind. Examples of microscopic models are NETSIM for traffic network simulation and TEXAS for individual intersections. PASSER and TRANSYT, for arterial simulation are two examples of macroscopic models.

The advent of the supercomputers, which offer at least an order of magnitude improvement over the conventional mainframes in speed and memory capabilities, provides the possibility of using the microscopic traffic simulation models for much more elaborate and realistic simulations than were done in the past. In this regard, one of the primary microscopic models to consider for implementation on a supercomputer for exploratory applications is NETSIM, arguably among the most detailed of existing traffic simulation models. This chapter concentrates mainly on the NETSIM model for network-level traffic simulation studies using the CRAY X-MP/24 supercomputer.

The present interest in microscopic traffic simulation using supercomputers is motivated by the past experience of limitations in conventional mainframe computing. From a research perspective, microscopic simulation is an important tool to study the behavior of traffic in networks. For instance, there has been considerable interest over the past few years in its use to support the development of a network-level traffic theory and the characterization of the performance of urban traffic networks (Mahmassani, Williams, and Herman, 1984; 1987). However, work to date has been limited to very small and possibly unrealistic networks. From an engineering practice perspective, microscopic simulation has rarely been performed for large urban networks. Most applications have been limited to small-scale local subsystems. Finally, an important motivator from both research and practice standpoints is the use of microscopic simulation as a predictive tool to study the effectiveness and support the design of in-vehicle information systems for electronic route guidance and traffic control in large networks.

1.1.2. The NETSIM Model

The NETSIM model was developed for the Federal Highway Administration (FHWA) as a part of the Urban Traffic Control System (UTCS) program, and the initial version was released in 1971 as UTCS-1 (Bruggeman, Leiberman and Worrall, 1971). This was subsequently updated in 1973 and 1978. All the modifications and tests performed as part of the present research were done on the latest available mainframe version of NETSIM.

The NETSIM computer program consists of 60 separate routines. The urban system must be represented as a set of nodes and directed one-way links. Vehicles enter the network via designated entry links and leave via exit links. Vehicles can also enter and leave at the sources or sinks at the mid-link locations. A range of controls, from no control to fully actuated control, can be specified for the intersections. Synchronization of the signals also can be achieved using user-specified offsets. Short term events, such as vehicle stoppages, and long-term events, such as lane-closures and accidents, can be modelled. Detailed modelling of bus traffic is also possible. Calibrated values are used for the characteristics of different specified driver types and vehicles types, and for the parameters of the microscopic behavioral relations, with provisions for the model user to substitute other values, if needed.

NETSIM is a discrete-time simulator which processes the vehicles by individually examining their location, speed, acceleration, etc., during each time step (generally one second) and updating the values according to the embedded mathematical models. The values are stored in the vehicle-specific array with 10 elements for each vehicle. Similarly, the information on the link-level is stored in the link-specific array with 33 elements for each link. It is this level of detail that results in the time and memory intensiveness of NETSIM.

The effectiveness of NETSIM has been severely affected by the limits of conventional mainframe computers. Even the most powerful of these mainframes allow only up to about 150 links and about 100 nodes in the network. However, even for a medium-sized city like Austin, Texas, the simulation of traffic in the central business district (CBD) requires about 300 nodes and 900 links. As for time, the simulation of traffic in even a 5x5 square grid network with 800 vehicles for two simulation hours resulted in about an hour of execution time on a dual CPU CYBER (CDC), (Mahmassani, Williams and Herman, 1984). In fact, there is no documented evidence of the use of NETSIM for a complete CBD simulation of any city, though there have been some disappointing attempts to use NETSIM for disaster-evacuation simulation in a large network (Seabrook, New Hampshire).

1.1.3. Supercomputing and the CRAY

The hardware performance of computers and supercomputers have shown dramatic improvement over the years. From bulky transistors to compact silicon chips, the progress in the last 35 years has been quite rapid (Buzbee and Sharp, 1985). However, the performance of silicon chips is fast approaching its quantum-mechanical limits. Also the fact that the signal has to travel across the dimensions of the computer is imposing a limit on the gate speeds to a few nanoseconds. Thus today's supercomputers utilize radical changes in architecture to achieve improvements within these limits (Zenios and Mulvey, 1986). The CRAY X-MP series has gained the widest acceptance among the different supercomputers that have emerged.

There are two dimensions to parallel processing using supercomputers (Chen, 1983). One is that of multi-programming, multi-tasking or micro-tasking which correspond to multiple processors working on different jobs, executing different tasks within a job and executing different portions of the same task, respectively. The other dimension is the parallelism achieved by using vector or matrix operations of an algorithm (vectorization). Compilers are generally available for the CRAY supercomputer to "vectorize" a particular code, by identifying those independent portions that can be executed in parallel, and sequencing the processing and task allocation accordingly. However, there are many inherently parallel activities that may have been programmed in ways intended for conventional scalar processing, but which actually inhibit the vectorization capabilities of the compiler. It is therefore generally possible to take fuller advantage of the capabilities of the supercomputer's architecture by modifying, or vectorizing the code. Three levels of vectorization can be distinguished. Local vectorization is the first level, where the program is re-examined in its subparts and subroutines, and redesigned only locally, without program-wide repercussions. Global vectorization is the next level where we examine the whole implementation of the algorithm and redesign the data structures and application choices to take advantage of the machine. At the third level, the algorithm itself is conceived to take advantage of the machine architecture.

1.2. MODIFICATIONS OF THE NETSIM PROGRAM

1.2.1. The CDC CYBER Version

This study used the 1978 version of NETSIM installed on the CYBER 170/750 (CDC) mainframe at the University of Texas at Austin. This version of NETSIM is implemented in FORTRAN IV, and uses overlays to minimize the memory requirements by utilizing the independence between different blocks of subroutines in NETSIM. Overlays essentially define a tree storage structure. At the head of the tree, there are four routines: The main program which calls the preprocessor block of routines, the simulation block of routines, the post processor routine and the fuel routine. At the next level in the tree, there are four independent blocks of routines corresponding to preprocessing, simulation, post processing and fuel data analysis. Of these, the preprocessor overlay has three sub-level overlays and the simulation overlay has two sub-level overlays. The overlay structure is shown in Fig. 1.1.

The version installed on the CRAY does not have any overlays, because the CFT compiler on CRAY does not allow overlays. This is not expected to create any significant limitations for the kind of large scale simulations that the CRAY version is intended to perform. In these cases, the vehicle and link information arrays will constitute the largest portion of the memory requirements, and neither of these arrays are cleared by the use of overlays.

1.2.2. Array Expansions in NETSIM

As said earlier, the existing versions of NETSIM allow only up to about 150 links and about 100 nodes. The CYBER version at the University of Texas allowed only 70 nodes and 115 links, and had to be significantly increased for this study to accommodate the large Austin CBD network, which is comprised of about 550 nodes and 1600 links.

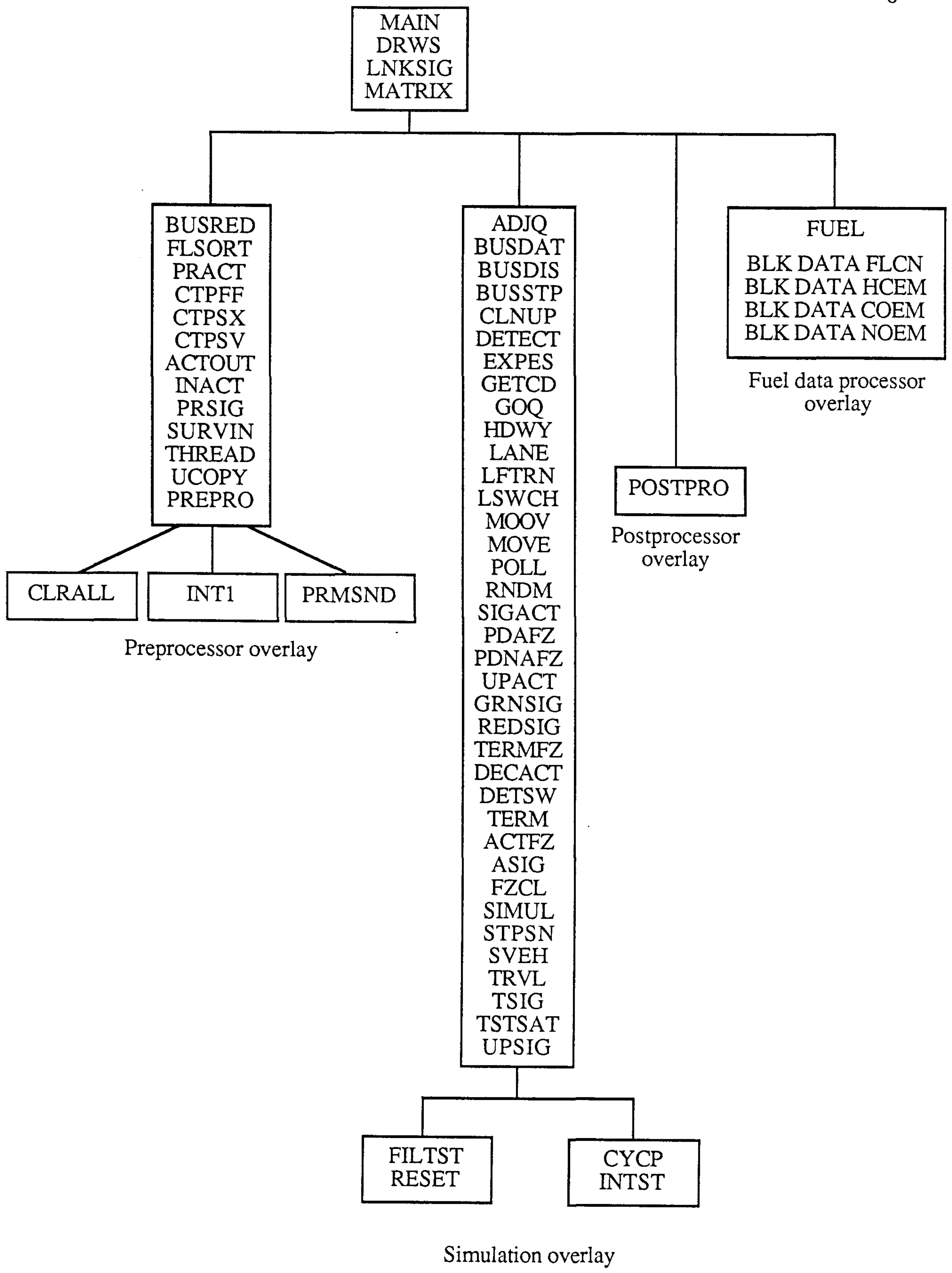


Figure 1.1. Overlay Structure of NETSIM on the CYBER 170/750

The modifications were made according to the guidelines given in the NETSIM user's manual (FHWA, 1980).

There are three main variables which control the dimensions of three sets of arrays in NETSIM. These are NMAX, the number of links, NCAR, the number of actuated nodes, and MXND, the number of nodes. In addition to the arrays affected by these variables, there is the common array V which is to be modified according to the value of MAXV, the maximum number of vehicles, and the common array DTCTR which is to be modified according to NDET, the maximum number of detectors. The values of all these key control variables are defined in the subroutine, CLRALL.

When NMAX is changed, the arrays which need modification are LINK, PCD, BLNK, NSPLBK, MNUVR, PSDLNK, ENDNDS, SUMO, SUMS, LTRNV, RTRNV, THRUV, VPROC and BUSLNK in common storage, LNAM and IS of subroutine PREPRO, and SLNK and X of subroutine CYCP. When MCAR is changed, the common arrays which need modification are NAFZ, NAAFZ, NATMR, NADET, NACNT, NALNK, NACDS, NACYC, NADEC and NASTAT. When MXND is changed, the common arrays, SIG, SIGT, SIGI, SNODE, VB, NDCNT, NPHS, VEMIT, NACT and IVENT need modifications.

Another array modification, which is not mentioned in the NETSIM user's manual, is in the common arrays IBFSRT and ITHRED, according to the variable ISSIZE which is defined in subroutine PREPRO. This has to be increased to the maximum expected number of lines in the NETSIM data deck.

For the purpose of the current study, NMAX was increased from 115 to 1650, MXND was increased from 1,000 to 3,000, 6,000, 9,000, etc., according to the different simulation cases. The NCAR value was not modified as there were no plans to study any simulation cases with actuated signals at present. The memory requirements with the above modifications are well below (less than 40%) the memory limit on the CRAY X-MP, thus subsequent modifications to accommodate more detectors should not cause any difficulties. ISSIZE, mentioned above, was increased from 300 to 6,000.

1.2.3. Other Modifications

One of the most important modifications made on the NETSIM version distributed by the FHWA was in the way the model calculates the stopped time of a vehicle in a queue at an intersection. In fact, the necessity for this change was felt earlier during the research on network level traffic variables at the University of Texas. The definition of stopped time used by NETSIM was found to be inappropriate for the fundamental research in network traffic theory which is being continued in this project. The modifications made earlier are described in detail by Williams (1986) and have been adopted in this research as well.

More capabilities were added to NETSIM, mainly to obtain additional information and specific details on traffic characteristics in the network. For instance, some of the research questions of interest require the ability to sample some individual vehicles in the network (Williams, 1986). For this purpose, the capability to track, accumulate and save results (final and intermediate) on individual vehicles also was incorporated into the model. Most of these modifications were made in the CLNUP subroutine of NETSIM. Another important new addition is the ability to obtain the speed distribution of the vehicles in the network. A list of additional output capabilities is given as Appendix 1.a.

1.3. SIMULATION EXPERIMENTS

1.3.1. Objectives of Experiments

As mentioned in the introduction, there were two primary objectives for this study. The first is to investigate traffic behavior in large, real-sized networks, so that better ways of characterizing urban network flow can be formulated. This objective is important from a fundamental traffic science perspective as well as for practical traffic engineering purposes. The second objective is to gain computational experience in simulating large-scale urban traffic using supercomputing capabilities, so as to assess the potential of the latter to expand the scope and scale of applications of microscopic traffic simulation. In fact, such improvements on the computational side are essential in achieving the former objective.

In this chapter, the simulation experiments conducted to address the above objectives are described. Simulations were performed with test networks of controlled characteristics as well as with a large network, that of the Austin central core. In addition to the substantive traffic theoretic results, the simulations also provided information on the computational aspects, such as the program execution time and memory requirements, and the time intensiveness of different parts of the program, which is useful for decisions on future modifications in the program. Compiler vectorization characteristics of the NETSIM program were also studied to some extent.

1.3.2. Description of Experiments

1.3.2.1. Experimental Design - NETSIM Code

Even though supercomputers can be expected to provide an order of magnitude improvement in execution speed compared to the conventional mainframes, using existing codes directly on vector computers will result in massive underutilization of supercomputing capabilities. Vectorization of the computer codes can increase the execution speed and efficiency considerably. In this study, vectorization was achieved only by using the CFT compiler of the CRAY X-MP/24, which provides the facility of vectorizing blocks of code of a given maximum length in the program. The responsiveness of NETSIM to different values of this maximum block length limit was studied. The results discussed in Section 1.4, suggest that using block length limits larger than the default used by the CFT does not result in any significant improvements.

It should be noted that considerably more execution efficiency could be achieved by rewriting the code incorporating vectorizable constructs. Some results on this aspect are given in the second part of this report in connection with network equilibrium assignment algorithms. In the case of the NETSIM code, this was not attempted in the present study because of the tremendous size of the code (about 15,000 lines). However, a time requirement analysis was performed using the FLOWTRACE capability of the CFT compiler (see Appendix 1.b). The results provide good indications as to the time intensive routines in the program which should be concentrated on if local vectorization by manual recoding is attempted. These results are discussed in more detail in a later section.

1.3.2.2. Experimental Design - Test Networks

As noted earlier, traffic simulations were performed both on small test networks as well as a large actual traffic network. The small test networks provide the opportunity to investigate traffic behavior under better controlled conditions than the large-scale real networks. These experiments were also expected to provide insight into the relative performance of NETSIM on the CRAY and the conventional mainframes for different characteristics of the networks. The main control variables with respect to which the studies were conducted are the size of the networks and the traffic concentration in the networks. Four different sizes and three different traffic concentrations were considered.

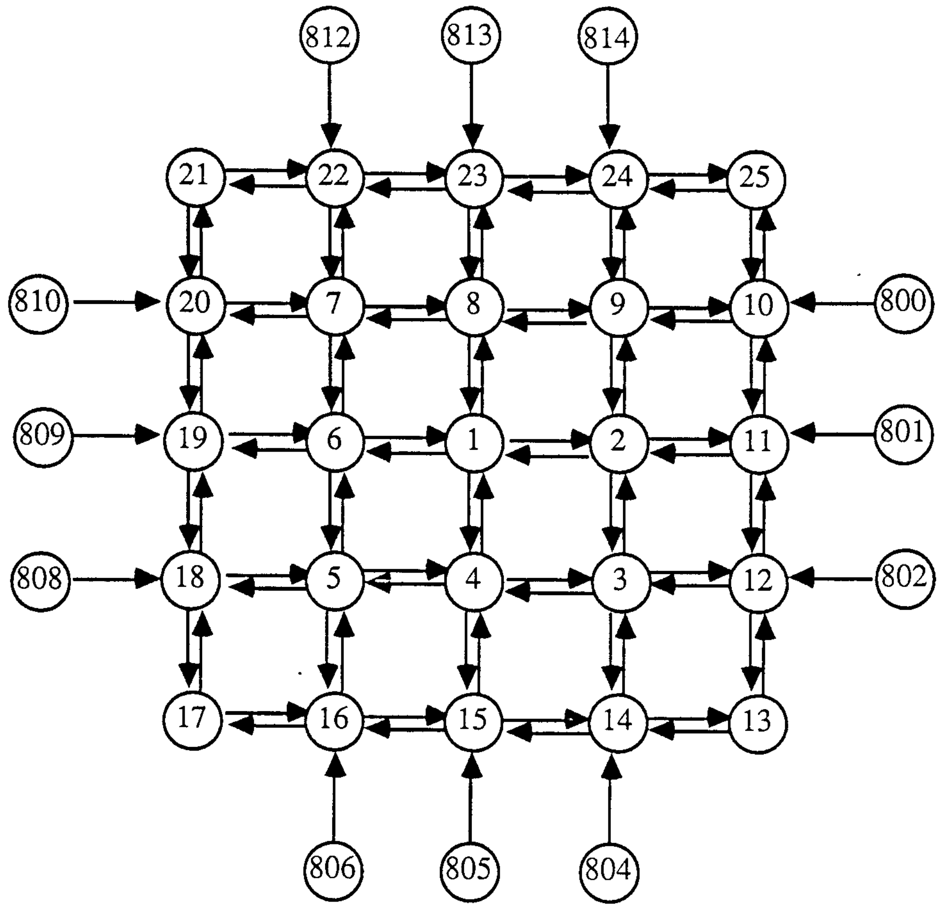
Due to the obvious geometric regularity that they offer, square grid networks were used in this study, as per the earlier work of Mahmassani et al. (1984, 1987) and Williams (1986). The sizes varied from 5 nodes by 5 nodes to 8 nodes by 8 nodes, the nodes being the intersections, with link lengths of 500 ft. Larger square grids could not be tested for the relative computational performance comparison, due to the limitations of the CYBER mainframe. In fact, to accommodate the 224 links associated with an 8x8 square grid of two-way streets, several array modifications and expansions were required on the NETSIM version on the CYBER mainframe, taking advantage of the fact that there were no detectors, actuated signals or bus routes in the test network. Networks larger than an 8x8 grid could not be attempted successfully. The test networks are shown in Figures

1.2-1.5. The spiral numbering of the nodes was adopted because of its advantages when modifying an 8x8 network to a 7x7 or smaller network.

The entry links of the test networks were at all the peripheral nodes of the grid, with the exception of the corner nodes. All the links in the networks were of 2 lanes each, with all feasible movements allowed at all the intersections. The turning percentages were 10% left, 75% through, 15% right at all the intersections where all three movements are allowed (i.e., the interior intersections), 50% -50% at intersections with two movements (i.e., the peripheral intersections) and, of course, 100% at intersections with one movement (i.e., the corner nodes). The performance of a 5x5 network with such turning patterns and geometric characteristics have been studied extensively in previous work (Williams, 1986), and the networks have been found to behave satisfactorily with respect to the uniform dissipation of traffic, which is important in preventing uneven or pathological development of congestion.

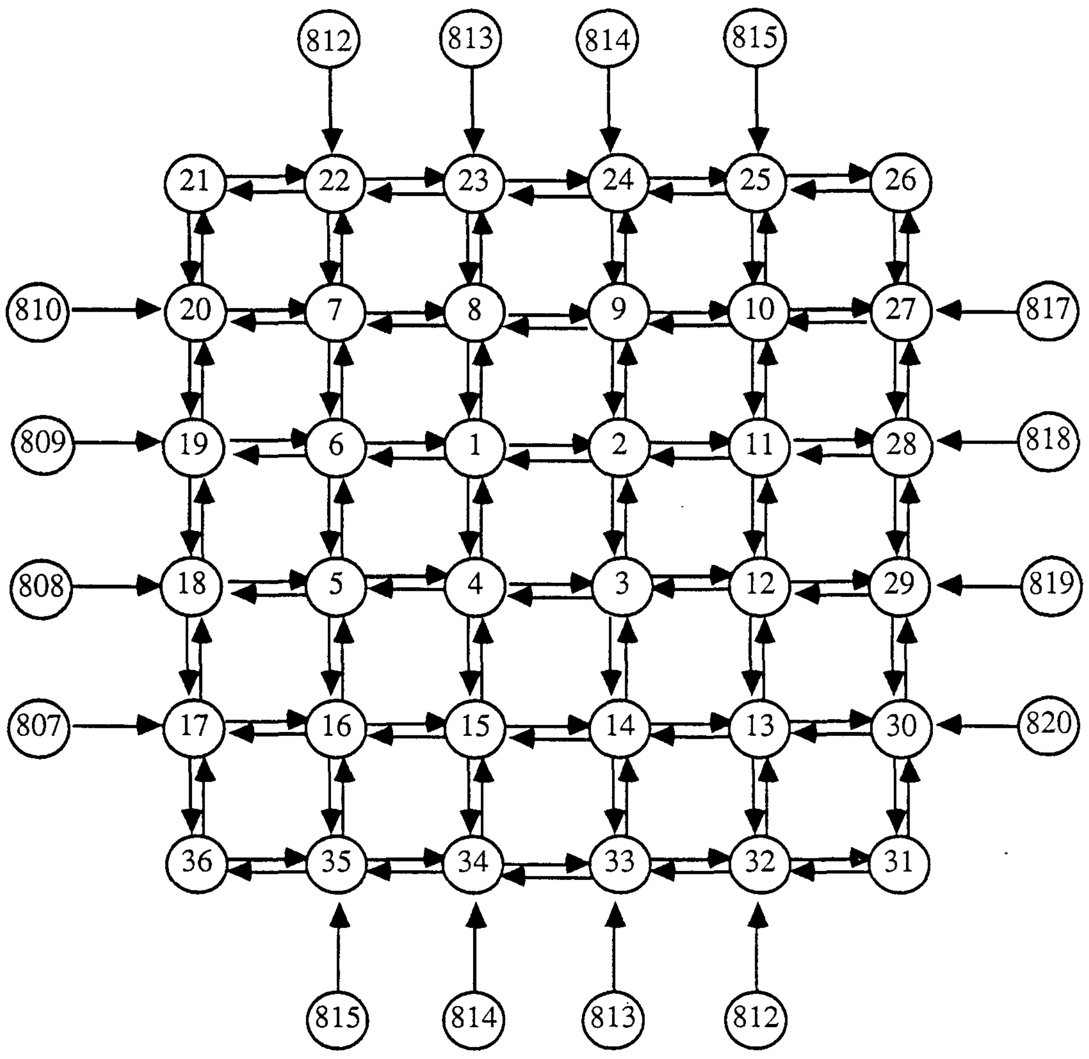
Six different target traffic concentration levels were studied: 10 veh/lane-mile, 20 veh/lane-mile, 30 veh/lane-mile, 40 veh/lane-mile, 50 veh/lane-mile, 60 veh/lane-mile. These concentrations are kept steady during a period of 10 minutes by not allowing vehicle entry or exit, following an initial loading period of 6 minutes† . Of course, as vehicle generation is stochastic, these concentration values were achieved only approximately. (See Table 1.1 for actual concentrations.) These simulations were performed on the CRAY with the simulations for concentrations up to 30 veh/mile being duplicated on the CYBER mainframe. The higher concentration levels could not be tried on the CYBER because of the maximum vehicle number limit on the CYBER version of NETSIM. In fact, the limit is reached with 20 veh/mile concentration in the case of the 8x8 network.

† Because the loading time was kept constant for all the simulations, the entry rate at the entry links were set in each case to achieve the desired concentration level.



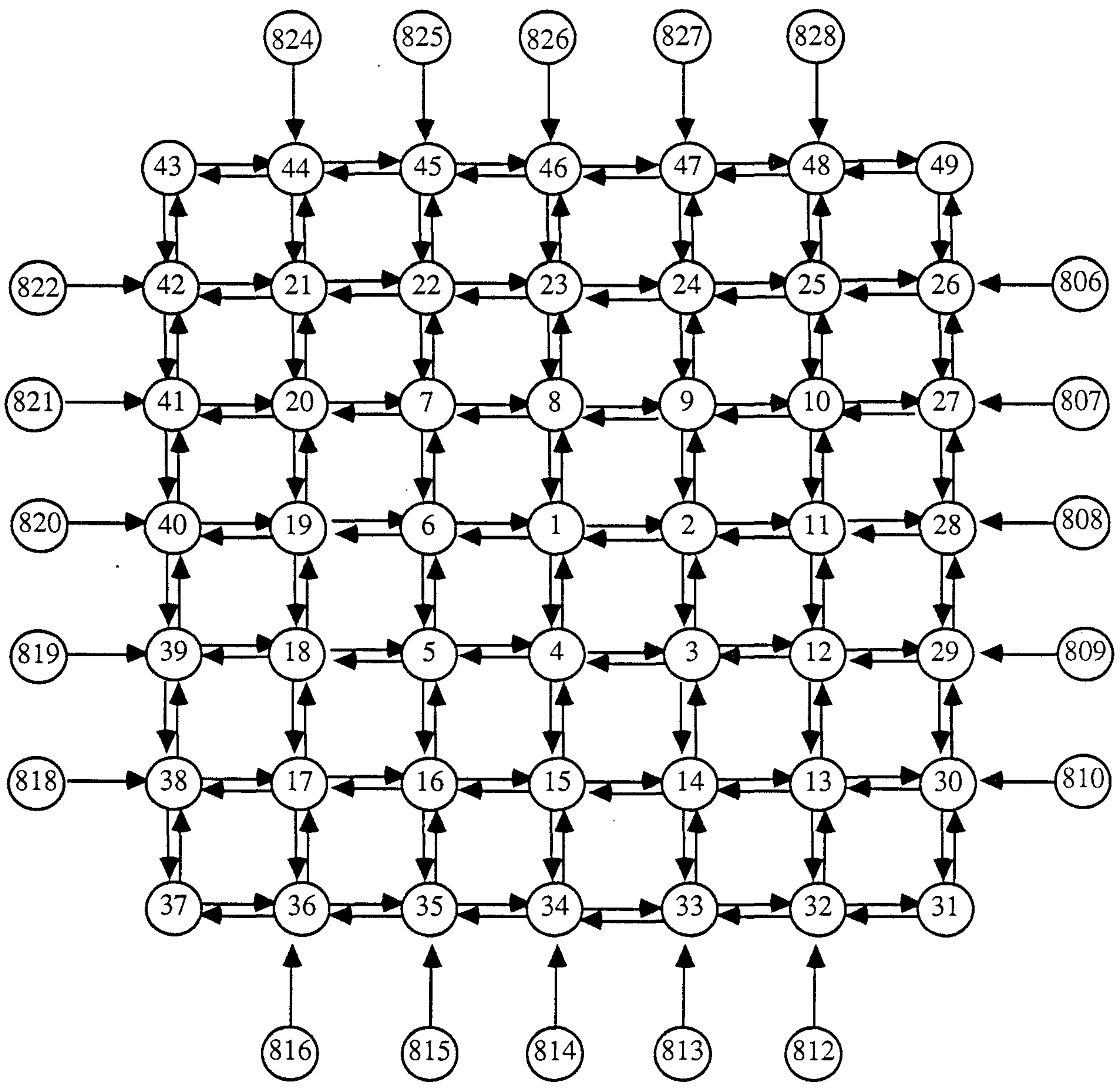
Number of Nodes	= 25
Number of Links	= 80
Number of Entry Points	= 12
Lane Miles	= 15.2

Figure 1.2. 5x5 Network



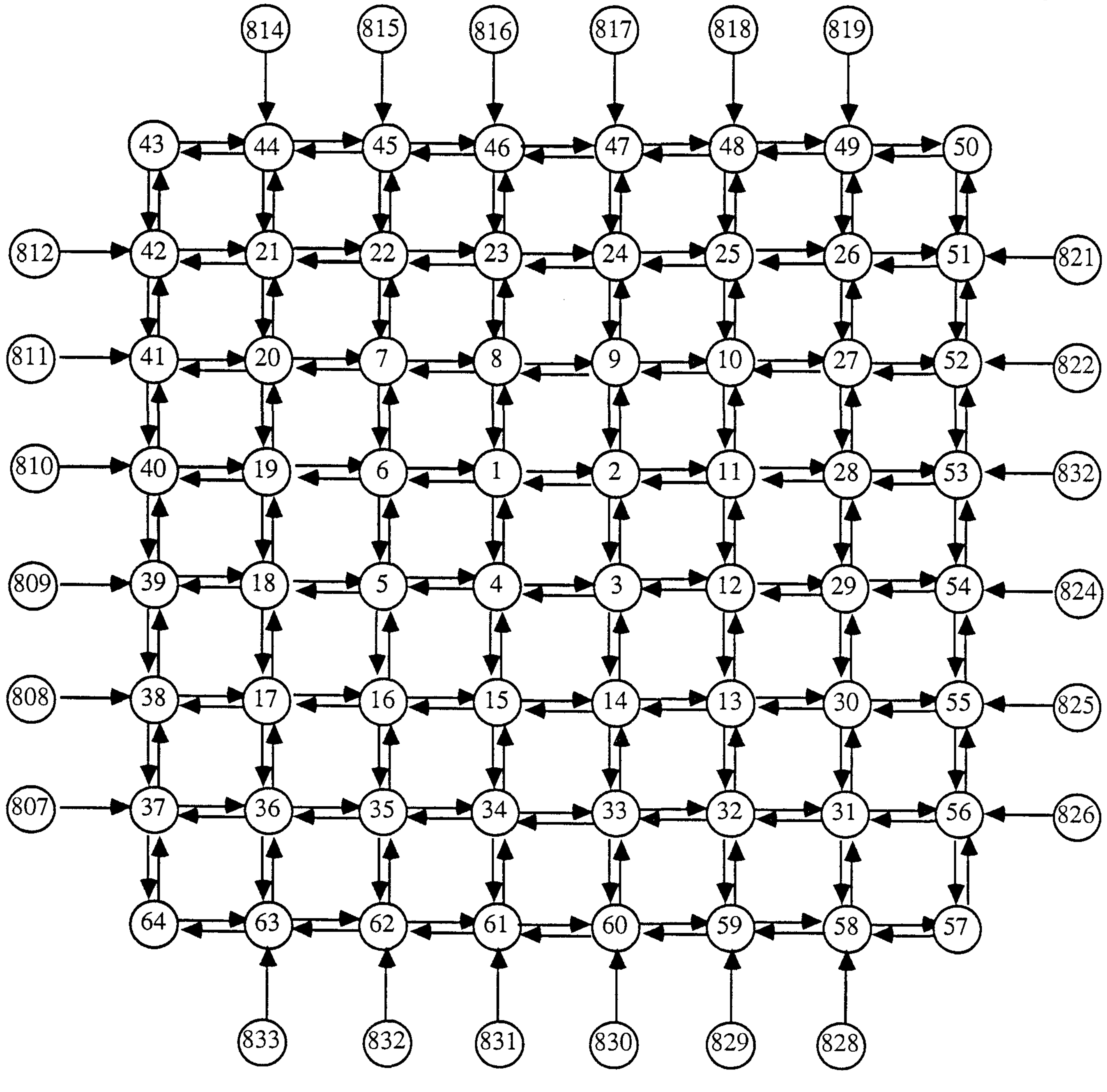
Number of Nodes	= 36
Number of Links	= 120
Number of Entry Points	= 16
Lane Miles	= 22.9

Figure 1.3. 6x6 Network



Number of Nodes	=	49
Number of Links	=	168
Number of Entry Points	=	20
Lane Miles	=	32

Figure 1.4. 7x7 Network



Number of Nodes	=	64
Number of Links	=	224
Number of Entry Points	=	24
Lane Miles	=	42.7

Figure 1.5. 8x8 Network

All the traffic signals in the networks were timed so as to have 40 seconds cycles, with both directions in the grid having single alternate signal synchronization. The signal cycles were found to be adequate for the range of concentrations considered in these simulations.

Additional simulations were performed for the case of an unsignalized 5x5 grid network, with all-way STOP signs at all the intersections. The reason for performing these simulations was to compare the results of the small test networks with the results from the large Austin network, which, as described in the next section, was simulated for the scenario of STOP sign control at all the intersections. The unsignalized network was also simulated for the six target concentration levels of 10, 20, 30, 40, 50, and 60 vehicles/lane-mile, respectively.

Thus, four different test networks were simulated for six different traffic concentrations, on both the mainframe CYBER and the CRAY for most cases, under both signalized and STOP sign control. Additional runs were performed on the CRAY for the 5x5 signalized test network at the 30 veh/lane-mile concentration level in order to examine the effectiveness of compiler vectorization for different maximum lengths of code blocks (i.e., for different values of the MAXBLOCK parameter in the CFT command). However, these additional runs did not generate any new traffic performance data as they were intended to study the computational aspects only. The results of these simulations are discussed in a later section, following the description of the large Austin network.

1.3.2.3. Experimental Design - Large Network (Austin)

Because one of the principal objectives of this study is to demonstrate the ability to perform a microscopic simulation of traffic in a real-sized large city network, a NETSIM dataset was developed for the core area of Austin, Texas. As stated earlier, tasks of this magnitude have not generally been performed previously, but can now be contemplated due to the availability of supercomputer facilities. The delineation of the study area followed relatively evident boundaries, which included the Central Business District as well as the University of Texas campus area. The area is bound on the east by the Interstate Highway I-35, on the west by the MoPac Freeway, on the south by the Colorado River, and on the north by 26th Street, which is just to the north of the University.

Another consideration in selecting the size of the study area was to avoid having more than 800 nodes in the network, because NETSIM reserves node numbers above 800 for entry nodes. Extensive code modification would have been necessary in order to increase this number, which is beyond the scope of the present study, and not essential to its objectives. The delineated study area in Austin was found to have less than 600 nodes, which is acceptable from the above standpoint.

Because of the exploratory illustrative nature of the study objectives, some simplifying assumptions were made in preparing the dataset representing the Austin network. Firstly, the topographic features of the area (hills, grades, etc.) were neglected. Secondly, all the nodes of the network were assumed to be unsignalized intersections with four-way STOP control. The principal reason for this assumption was the difficulty of obtaining detailed and reliable signal timing information for all the intersections in the network in the limited time available for the study, especially since such information is not essential to our objectives. From a traffic theory standpoint, the question of how a network such as the Austin CBD would perform under unsignalized STOP-sign control is of interest in its own right, especially since this performance is to be examined at different network concentration levels. More importantly, from a computational standpoint, this assumption does not in any way limit the validity of the conclusions on the size network that can be readily simulated on the supercomputer nor on the associated computational performance. The assumption is therefore not a serious shortcoming, because NETSIM does not specifically keep track of whether a node is signalized or not. Instead, all the node-related arrays are dimensioned and primed, regardless of whether the nodes are signalized or not. This means that specifying nodes as unsignalized does not in any way reduce the memory

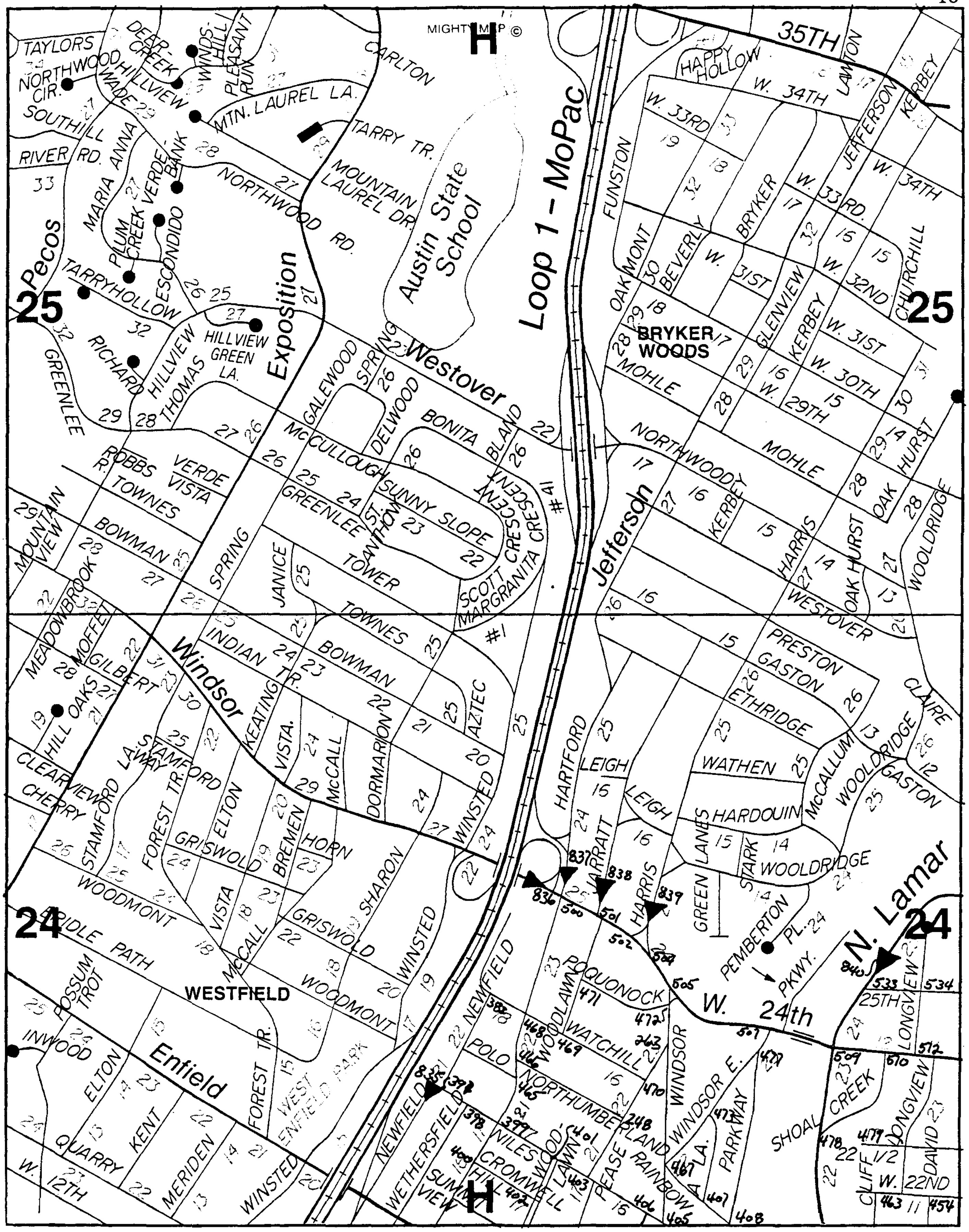

MIGHTY MAP ©
H
35TH
Loop 1 - MoPac
Austin State School
Exposition
Westover
BRYKER WOODS
Jefferson
Windsor
WESTFIELD
Enfield
W. 24th
N. Lamar
Pecos
25
24

Figure 1.6.a

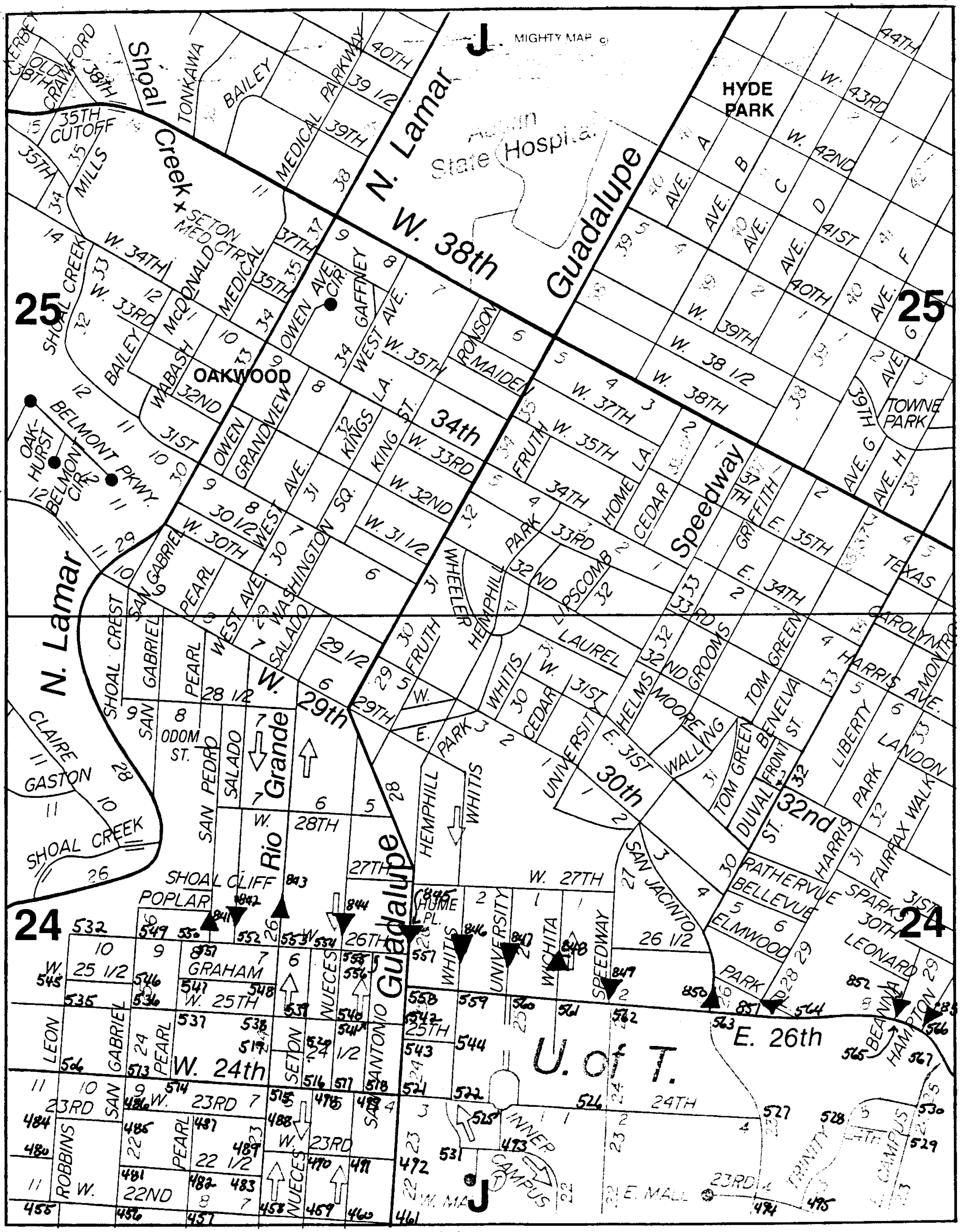

MIGHTY MAP
HYDE PARK
OAKWOOD
N. Lamar
W. 38th
Guadalupe
Speedway
Shoal Creek
SETON MED. CTR.
Austin State Hospital
34th
W. 29th
30th
32nd
Rio Grande
W. 24th
E. 26th
U. of T.
25
24
J

Figure 1.6.b

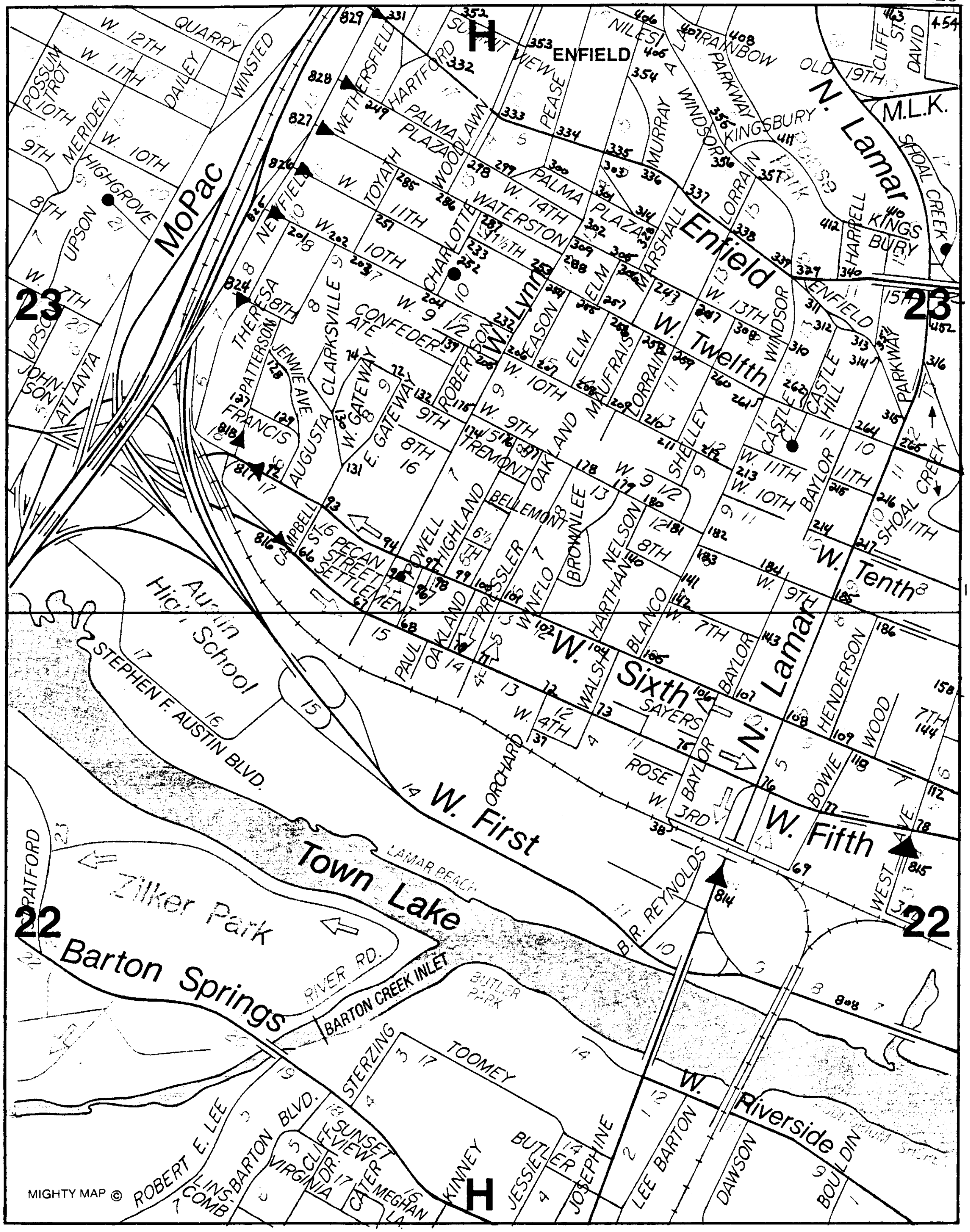

H
ENFIELD
MoPac
N. Lamar
Enfield
W. Twelfth
W. Lynn
W. Tenth
W. Sixth
N. Lamar
W. Fifth
W. First
Town Lake
Zilker Park
Barton Springs
W. Riverside
Austin High School
STEPHEN F. AUSTIN BLVD.
BARTON CREEK INLET
M.L.K.
23
22
MIGHTY MAP ©

Figure 1.6.c

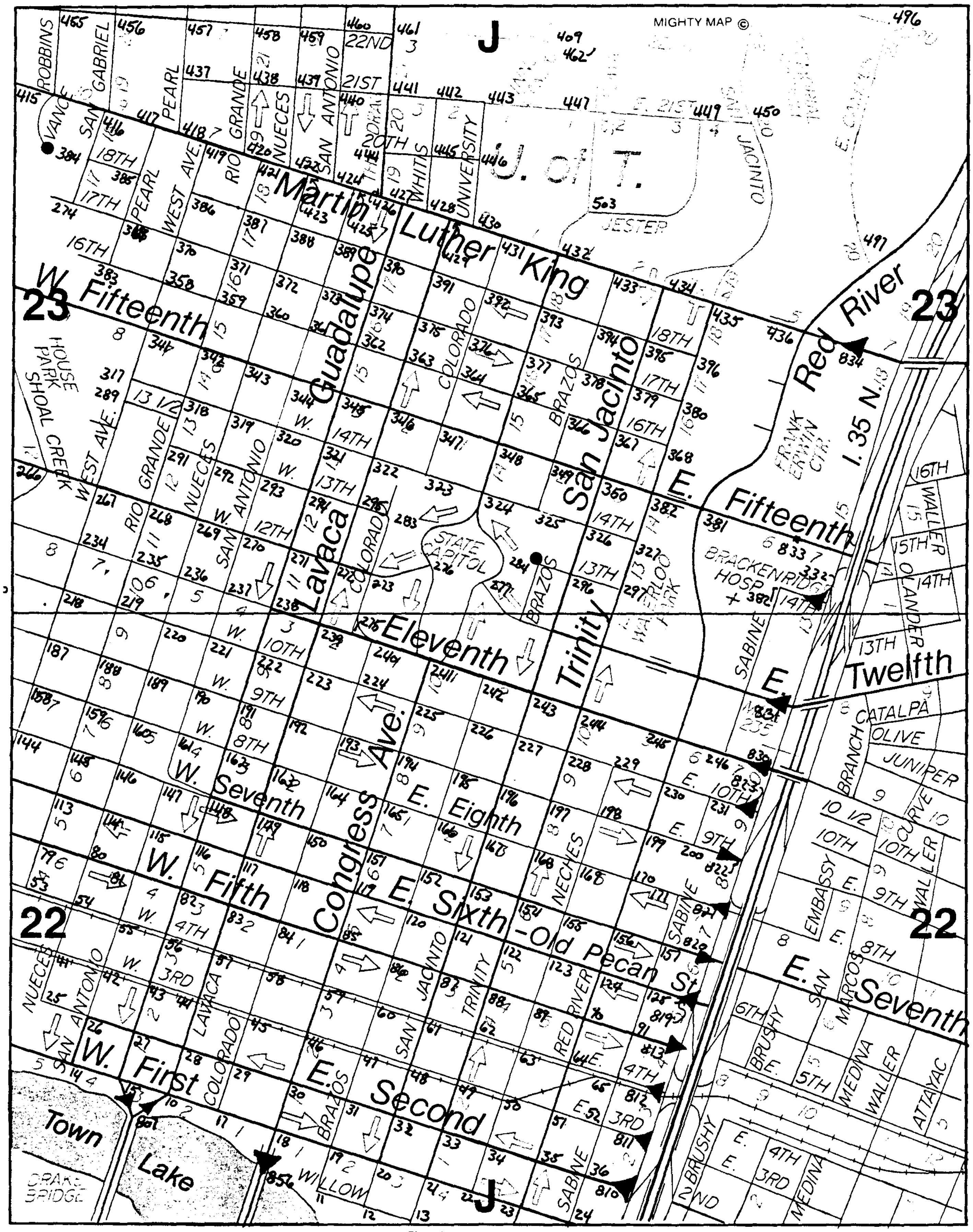
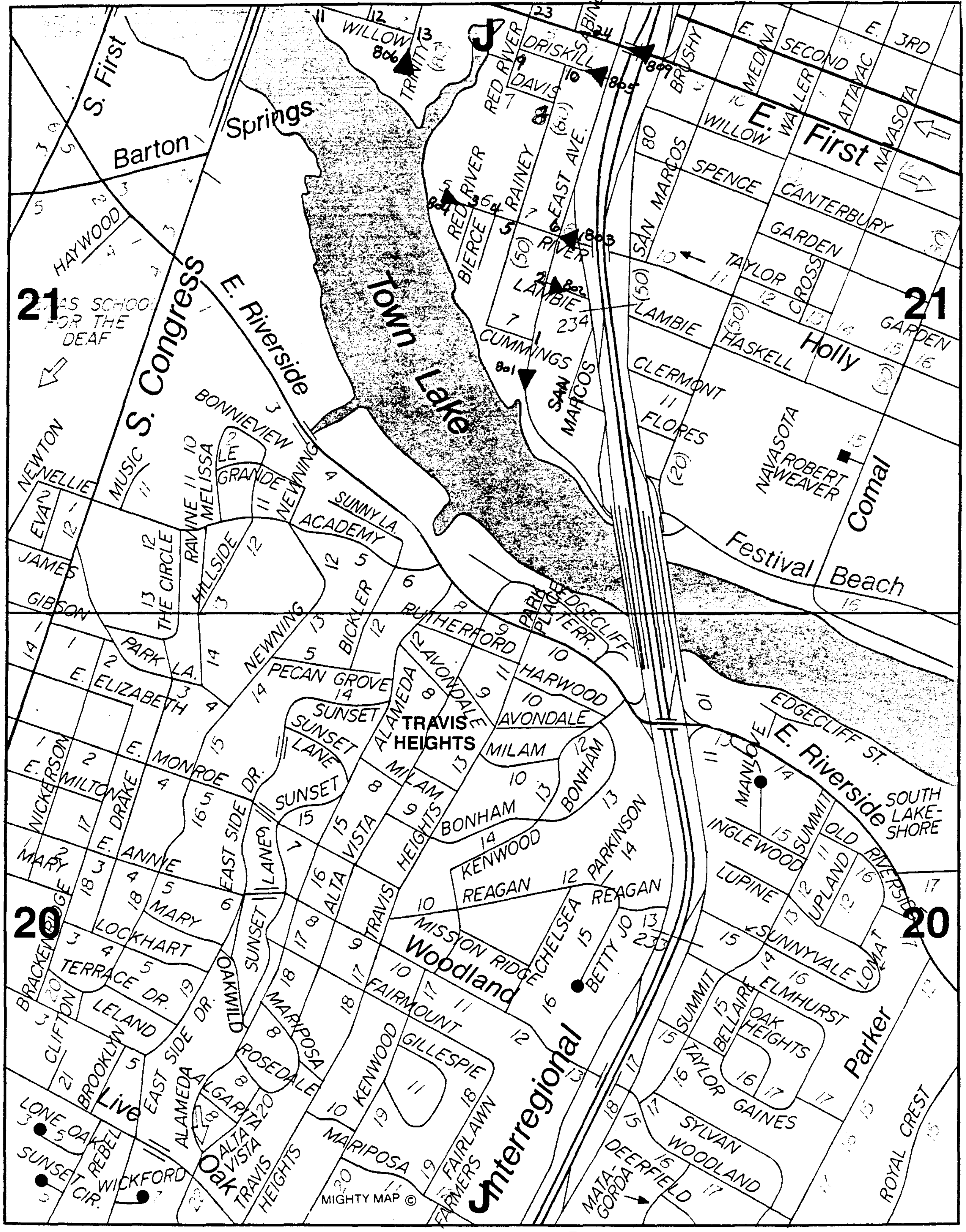

MIGHTY MAP ©
J
23
22
U. of T.
Martin Luther King
W. Fifteenth
E. Fifteenth
Guadalupe
Lavaca
Congress Ave.
San Jacinto
Trinity
Red River
I. 35 N.
Eleventh
E. Twelfth
W. Seventh
E. Eighth
W. Fifth
E. Sixth-Old Pecan St.
E. Seventh
W. First
E. Second
Town Lake
STATE CAPITOL
BRACKENRIDGE HOSP.
FRANK ERWIN CTR.
HOUSE PARK
SHOAL CREEK
JESTER
WALLER
OLANDER
CATALPA
OLIVE
JUNIPER
BRANCH
EMBASSY
BRUSHY
SAN MARCOS
MEDINA
ATTAYAC
WILLOW
SABINE
NECHES
BRAZOS
COLORADO
NUECES
SAN ANTONIO
RIO GRANDE
WEST AVE.
PEARL
SAN GABRIEL
ROBBINS
VANCE
UNIVERSITY
WHITIS
WATERLOO PARK
DRAKE BRIDGE

Figure 1.6.d



Town Lake
S. Congress
E. Riverside
Barton Springs
S. First
E. First
Festival Beach
Travis Heights
Woodland
Interregional
Live Oak
Holly
Comal
Parker
Mighty Map ©
21
20
J

Figure 1.6.e

requirements of NETSIM. In fact, a STOP sign is equivalent to any kind of signal at an intersection, and is specified as signal control code 5 on the signalization data cards. Preliminary investigation of the code did not indicate that signalization was more demanding in terms of execution time than stop control. This was subsequently verified experimentally in connection with the small test networks. A map of the study area and the NETSIM network nodes are given in Figs. 1.6 a-e.

The above network was simulated under five different target average vehicle concentration levels: 5, 10, 15, 20, and 30 veh/lane-mile, corresponding to about 1250, 2500, 3750, 5000, and 7500 vehicles, respectively, in the network (see Table 1.1 for actual concentrations). Higher concentration levels were not attempted because of concern over potentially high execution times. The memory capabilities of the supercomputer would have allowed much higher concentrations, but the execution time resources available for this study did not allow such high-concentration simulations. It should be noted that measured concentrations during peak period operations in the Austin CBD network are within the range of concentrations considered in the simulations (Ardekani and Herman, 1987).

Due to the rather small number of entry links (only 43 for the entire 250 lane miles of road network), special care was taken to prevent undue congestion at the entry links, especially for the higher traffic concentration simulations. Multiple simulation subintervals with and without vehicle entry were used for the higher concentration cases to facilitate the dissipation of entry point congestion. The traffic loading patterns are shown in Table 1.2 for the five concentration levels.

To compare the results of compiler vectorization, different code block length limits (for vectorization) were tested, by varying the value of the MAXBLOCK parameter of the CRAY FORTRAN (CFT) compiler. MAXBLOCKS of 1, 2310 and 4,620 were used for the 10 veh/lane-mile concentration level, 1 and 2310 for the of 20 veh/lane-mile concentration level, and a single value of 2310 for the 30 veh/lane-mile concentration level.

Table 1.1. Average Network Speeds for Different Vehicle Traffic Concentrations

Network	Concentration (Vehicles/lane-mile)		Speed (mph)
	Target	Actual	
5x5 SIG	10	11.81	18.61
5x5 SIG	20	22.18	16.31
5x5 SIG	30	33.26	13.08
5x5 SIG	40	44.28	10.93
5x5 SIG	50	54.58	9.04
5x5 SIG	60	66.06	7.25
6x6 SIG	10	11.92	18.02
6x6 SIG	20	22.53	15.71
6x6 SIG	30	33.79	12.46
6x6 SIG	40	42.24	10.59
6x6 SIG	50	52.80	8.84
6x6 SIG	60	62.39	6.73
7x7 SIG	10	11.31	18.16
7x7 SIG	20	22.60	15.48
7x7 SIG	30	33.32	12.76
7x7 SIG	40	40.23	10.90
7x7 SIG	50	50.88	8.75
7x7 SIG	60	60.34	6.47
8x8 SIG	10	11.31	18.29
8x8 SIG	20	22.60	15.77
8x8 SIG	30	33.31	13.61
8x8 SIG	40	40.23	11.37
8x8 SIG	50	50.88	8.34
8x8 SIG	60	60.34	6.55
5x5 ST	10	11.81	7.95
5x5 ST	20	22.11	4.83
5x5 ST	30	33.26	3.37
5x5 ST	40	40.26	2.85
5x5 ST	50	48.90	2.22
5x5 ST	60	55.31	1.85
Austin ST	5	5.73	11.40
Austin ST	10	11.23	7.98
Austin ST	15	16.41	5.58
Austin ST	20	21.47	4.35
Austin ST	30	29.43	2.69

SIG - refers to signalized network
ST - refers to STOP-sign controlled network

Table 1.2. Vehicle Entry Rates for Different Simulation Subintervals - Austin Network
[The relaxation periods were used to dissipate congestion at the entry point]

Vehicle Concentration (Target)†	Vehicle Entry Rate (per link/hour)	Length of the Period (sec)	Simulation Sub-Interval	No. of Entry Links	Expected Total No. of Vehicles Input
5 veh/lane-miles	180 0	600 900	loading simulation	43	1290
10 veh/lane-miles	360 0	600 900	loading simulation	43	2580
15 veh/lane-miles	300 0 300 0 300 0	360 120 360 120 360 900	loading relax loading relax loading simulation	43	3870
20 veh/lane-miles	400 0 400 0 400 0	360 120 360 120 360 900	loading relax loading relax loading simulation	43	5160
30 veh/lane-miles	400 0 400 0 400 0	540 120 540 120 540 900	loading relax loading relax loading simulation	43	7640

†See Table 1.1 for actual concentrations

Note that a MAXBLOCK of 1 will lead to a compilation with no vectorization. The results of the different simulations are discussed the next section.

1.4. EXPERIMENTAL RESULTS

1.4.1. Experimental Results - Traffic Aspects

As noted in the introduction, microscopic simulation can be an important tool to study the behavior of traffic networks and characterize their performance through the formulation of relations among network-level descriptors of traffic quality and the dependence of these relations on the physical and operational features of the network. The simulation results are examined from the standpoint of two such relations: a speed-concentration relation and the two-fluid model, discussed hereafter.

One important indicator of the performance of urban traffic networks is the variation of the average speed in the network with the vehicular concentration in the network. In previous research, the result of microscopic simulation indicated that the relation between network-level averages of speed and concentration followed the familiar shape long observed for individual highway facilities (Williams et al., 1987; Mahmassani et al., 1987). However, in that research, the sensitivity of this relation to various network features was examined only for simulated networks of a given size, due in part to the previously explained computational limitations preventing the microscopic simulation of traffic in larger networks. In the present research, it is possible to examine the effect of network size on the speed-concentration relationship. Figure 1.7 shows the speed-concentration curves for the grid test networks of different sizes, as well as for the larger Austin core area network. Table 1.1 lists the corresponding average speed and concentration values. The most notable aspect from these curves is the large drop in speed for a given concentration when the intersections in the network are operated in the unsignalized, STOP-sign controlled mode. This seems to occur irrespective of the size of the network. There does not appear to be discernable systematic differences due to size among the speed-

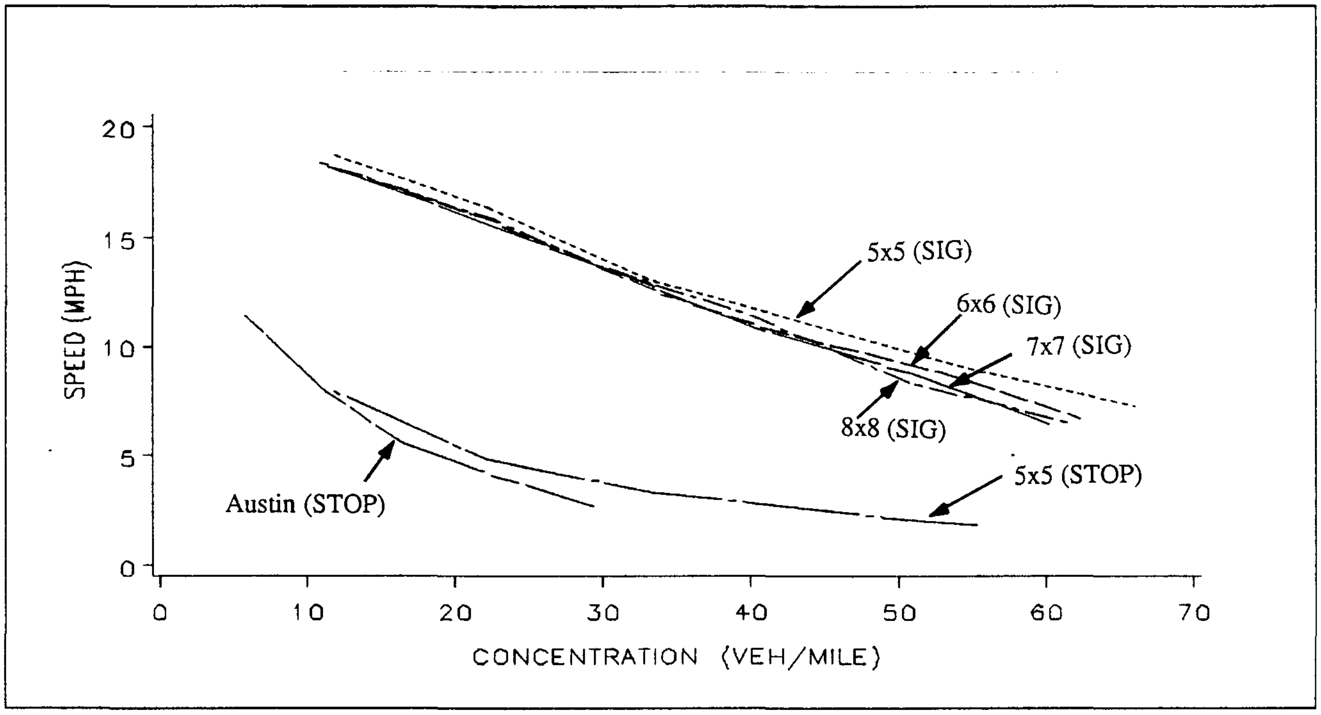


Fig. 1.7. Speed-Concentration Curves for the Different Networks.

concentration curves for the four signalized test networks, whose sizes varied from 15.2 miles (for the 5x5 grid) to 42.7 miles (for the 8x8 grid). However, when the signal control is replaced by all-way stop control at all intersections, the average speed values decrease by about 10 mph, suggesting that changes in control from unsignalized to signalized operation at intersections has much more of an impact on the speed-concentration curves than the size of the network.

It should be noted that the concentration values mentioned above are intended values which could only be approximated through the vehicle loading process. Actual concentration values could be somewhat different, especially for the large Austin network, as congestion at the entry points may have prevented the realization of the intended concentration levels.

The next aspect of the traffic quality studied were the parameters of the two-fluid model, which is based on a theory of town traffic proposed by Herman and Prigogine (1979), and validated in several field studies as well as simulation experiments. The theory proposes that traffic in street networks can be viewed as two fluids, one consisting of moving vehicles and the other of vehicles stopped due to reasons other than parking. The basic postulate of the theory leads to a characterization of the quality of traffic services with the two parameters n and T_m in the following relation:

$$T_r = T_m^{\frac{1}{n+1}} \; T^{\frac{n}{n+1}}$$

where

T_r is the average running (moving) time per unit distance and

T is the average total trip time per unit distance (both T_r and T are network-wide averages).

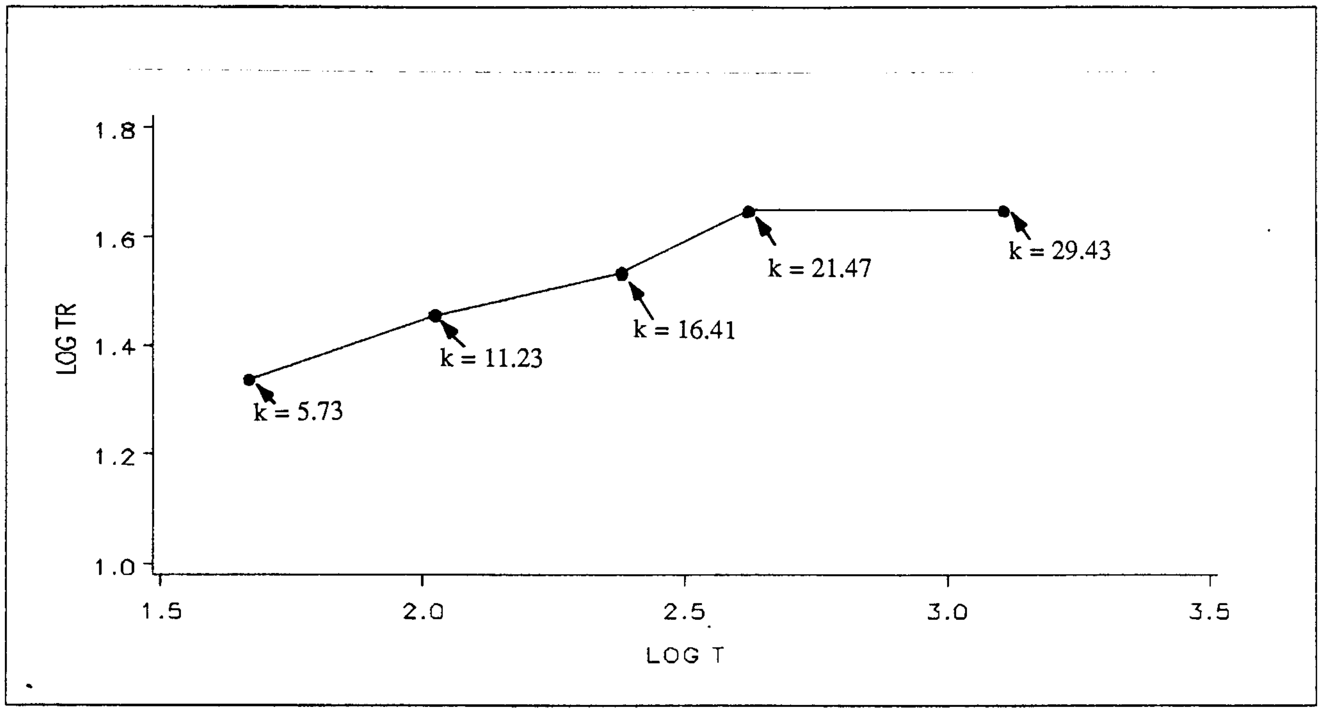
The parameter T_m can be interpreted as the average maximum moving time per unit-distance, i.e., in the absence of interference due to the presence of other vehicles. The parameter n captures the degree to which the average running time increases with congestion in the network.

Taking the natural log of both sides of the above equation yields

$$\ln T_r = \frac{1}{n+1} \ln T_m + \frac{n}{n+1} \ln T$$

In order to estimate the two parameters, n and T_m, the average running time and total trip time values, both per unit distance, are obtained for the different concentration levels considered in the simulations. The $\ln T_r$ vs. lnT relation is shown in Fig. 1.8 for the large Austin network. The depicted relation is essentially linear over the range of concentrations considered, with the exception of the highest values for which a levelling off can be noted. Similar behavior has been observed in earlier research with small signalized networks (Mahmassani et al., 1987; Williams, 1986) for concentrations in excess of 60 or more vehicles per lane-mile. However, a concentration of 30 vehicles/lane-mile can be considered to be extremely high for a network controlled only by stop signs at intersections. This again suggests that the two-fluid model assumptions may remain correct over practically-encountered range of concentrations, but may not necessarily hold for extreme cases (which would be difficult to observe anyway). Figure 1.9 shows the lnT vs $\ln T_r$ curve for the 5x5-node test network with STOP-only control, revealing an essentially linear pattern.

To estimate the two-fluid parameters, n and T_m, a simple least-square regression was performed with the $\ln T_r$ and lnT values. For the large Austin network, this was done with and without the data point corresponding to the 30 veh/mile concentration level (in light of the above-mentioned nonlinearity). The estimation steps and results are shown in Table 1.3 for the Austin network. The estimated parameter values for n and T_m are 0.3 and 3.57, respectively, for the case with all the data points and 0.45 and 3.28 for the case with the one data point removed.



Note: k stands for concentration in vehicles/lane-mile

Fig. 1.8. $\ln T_r$ vs. $\ln T$ for the Austin Network.

Table 1.3. Two-Fluid Parameter Estimation for the Austin Network

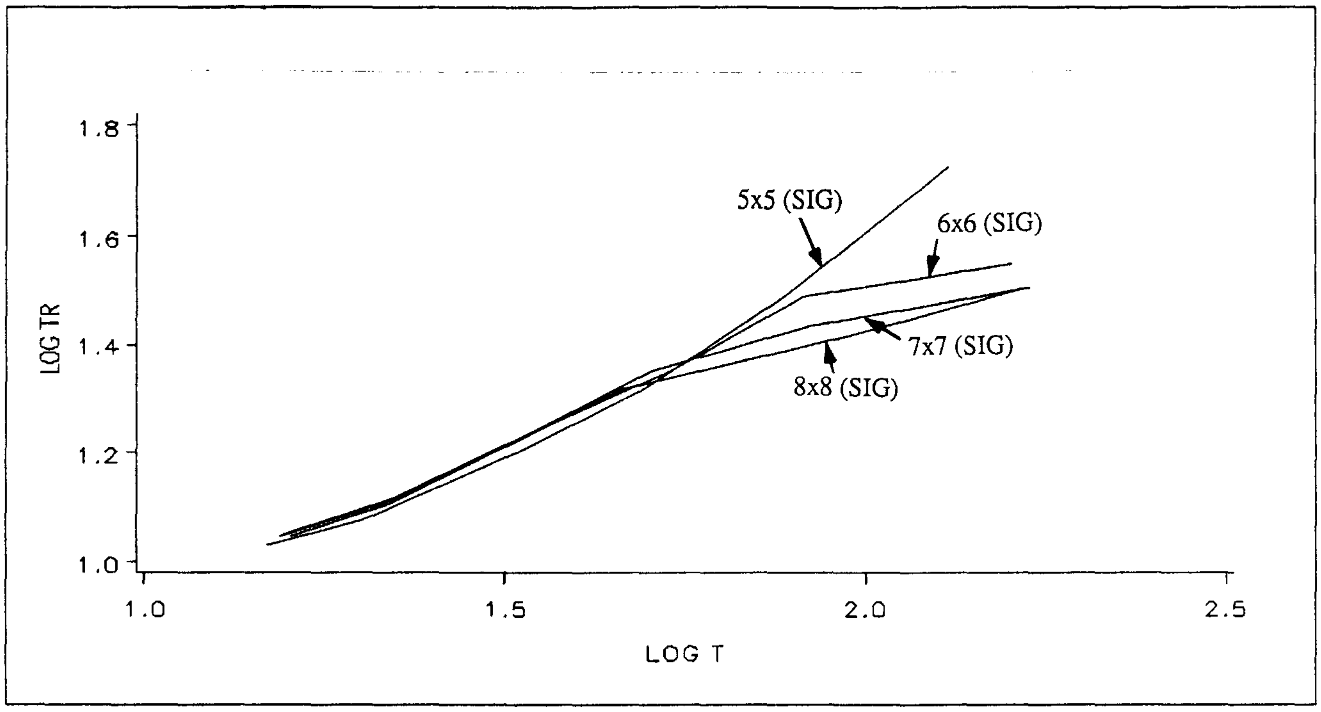
Case	Parameter A		Parameter B		n	T_m
	Estimate	T-stat	Estimate	T-stat		
Austin (concentration levels of 5.73, 11.23, 16.41, and 21.47)	0.817	12.58	0.311	10.52	0.45	3.27
Austin (concentration levels of 5.73, 11.23, 16.41, 21.47, and 29.43)	0.980	9.40	0.231	5.32	0.30	3.57

Model: $\ln T_r = A + B \ln T$

$$= \left(\frac{\ln T_m}{n+1}\right) + \left(\frac{n}{n+1}\right)\ln T$$

The two-fluid parameters were also estimated for the four test networks of different sizes. The $\ln T_r$ vs. $\ln T$ relations for the four networks are shown in Fig. 1.9, revealing a pattern that is essentially linear over most of the range of observations considered with the exception of the higher concentrations. The same type of nonlinearity observed earlier is also present here, with the curves levelling off at higher concentrations. In general, this happens above about 40 veh/lane-miles concentration. Regression lines were calibrated separately for the simulation results for all the concentration levels and also for those for up to the 40 veh/lane-miles concentration level. The linear portions of the $\ln T_r$ vs. $\ln T$ curves almost coincide, and so the latter regressions resulted in comparable estimates of n and T_m. This was further confirmed by an F-test of the hypothesis that the parameters of the $\ln T_r$ vs. $\ln T$ relation are not significantly different across the four networks for concentrations up to about 40 veh/lane-miles. On the other hand, a similar F-test conducted on the estimates obtained using all six concentration levels showed that the model parameters are significantly different (see Table 1.4 for the regression results, two-fluid parameter estimates and the F-tests). The reason for this is the nonlinearity of the relationship at higher concentrations, and the fact that different networks exhibit nonlinearity above different "critical" concentrations. In fact, Fig. 1.9 shows that this "critical" concentration becomes lower as the network size increases. Recall that for the Austin network, this critical level was closer to 20 veh/lane-miles, as seen in Fig. 1.8. The parameter estimates of n and T_m appear to be quite robust, at about 1.4 and 2.3, respectively (Table 1.4), for all the four networks at the lower concentrations. This further confirms the applicability of the two-fluid theory over the range of practically meaningful concentrations, and points to the need for further investigation of the behavior of traffic networks at very high concentrations.

The results obtained also indicate that the traffic control in the network does have a significant effect on the $\ln T_r$ vs. $\ln T$ relation, as can be seen in Fig. 1.10, which shows a plot of these relations for the signalized and unsignalized (STOP-controlled) 5x5



Note: Concentrations at successive points are:

11.81, 22.18, 33.26, 44.28, 54.58 and 66.06 for 5x5 network,
11.92, 22.53, 33.79, 42.24, 52.80 and 62.39 for 5x5 network,
11.31, 22.60, 33.31, 40.23, 50.88 and 60.34 for 5x5 network, and
10.89, 22.63, 29.98, 40.14, 50.89 and 61.59 for 5x5 network,

Fig. 1.9. $\ln T_r$ vs. $\ln T$ the Four Test Networks.

Table. 1.4. Two-Fluid Parameter Estimation for Signalized Networks of Different Sizes

Network	No. of Data Points	Parameter A		Parameter B		Sum of Squared Errors (SSE)	n	T_m
		Estimate	T-stat	Estimate	T-stat			
Case 1: All six concentration levels included in estimation								
5x5 SIG	6	0.129	1.34	0.730	12.50	8.66×10^{-3}	2.70	1.61
6x6 SIG	6	0.412	5.63	0.536	12.42	5.08×10^{-3}	1.15	2.42
7x7 SIG	6	0.514	6.88	0.465	10.53	5.61×10^{-3}	0.87	2.61
8x8 SIG	6	0.532	10.59	0.447	14.95	2.72×10^{-3}	0.81	2.61
Pooled	24	0.410	7.44	0.535	16.29	66.63×10^{-3}	1.15	2.41
Case 2: Two highest concentration levels deleted from estimation								
5x5 SIG	4	0.367	6.70	0.556	14.57	4.64×10^{-4}	1.25	2.28
6x6 SIG	4	0.335	9.01	0.586	23.29	2.13×10^{-4}	1.42	2.25
7x7 SIG	4	0.328	6.46	0.595	17.10	3.64×10^{-4}	1.47	2.25
8x8 SIG	4	0.361	6.86	0.571	15.51	3.36×10^{-4}	1.33	2.32
Pooled	16	0.347	14.64	0.577	35.27	22.64×10^{-4}	1.37	2.38

<u>F-test of difference between the models (case 1)</u>

$$Q^{unrestricted} = \sum SSE = 22.07 \times 10^{-3}$$

$$Q^{restricted} = SSE_{pooled}\, 66.63 \times 10^{-3}$$

$$F^* = \left(\frac{Q^r - Q^u}{r}\right) \Big/ \left(\frac{Q^u}{N - K}\right) = \frac{66.63 - 22.07}{6} \Big/ \frac{22.07}{24 - 8}$$

$$= 5.38 > F_{6,\,16,\,0.025} = 3.3$$

So the models are significantly different.

<u>F-test of difference between the models (case 2)</u>

$$Q^u = 13.77 \times 10^{-4}$$

$$Q^r = 22.64 \times 10^{-4}$$

$$F^* = \frac{22.64 - 13.77}{6} \Big/ \frac{13.77}{16 - 8}$$

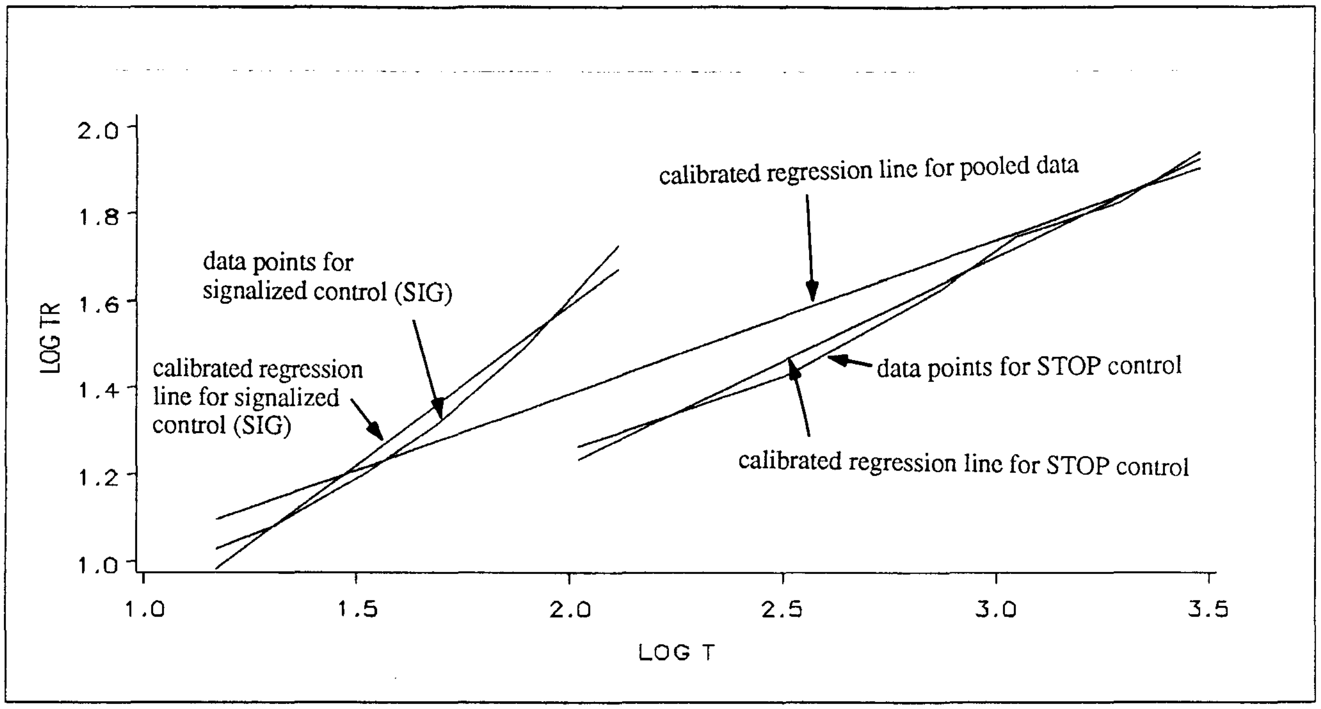
$$= 0.86 < F_{6,8,0.025} = 4.65$$

So the models are not significantly different.

networks, respectively. The figure also depicts the individual regression lines for the network under the two control strategies, as well as the line for the pooled data. The two-fluid parameter estimates are shown in Table 1.5, along with the F-test indicating that the parameters of the $\ln T_r$ vs. $\ln T$ relations are significantly different for the two cases. The estimates of n and T_m are 0.89 and 1.71, respectively, for the STOP-controlled network, and 2.70 and 1.61 for the signalized network.

The principal substantive conclusions of the above simulations in so far as the behavior of traffic is concerned are that 1) the size of the grid network does not appear to significantly influence its performance characteristics, and 2) the effect of drastic changes in control policies at intersections appears to be much more significant than the effect of size, at least for the type of grid test networks considered. Furthermore, by considering the results obtained here with earlier findings (Mahmassani et al., 1987), the effect of traffic control strategies is such that drastic changes, such as going from unsignalized to signalized control has a markedly greater impact on network performance than do improvements in the timing and/or coordination of traffic signals. For instance, going from a non-coordinated timing plan to a coordinated one that provided smoother progression along the major directions in the network resulted in a much lesser impact (in earlier studies) than the "jump" observed in going from STOP control to signal control in this study.

The finding herein that the size of the grid network does not particularly influence its performance characteristics alleviates some of the concerns expressed in connection with earlier work regarding the potential existence of a "boundary effect". The latter would be due to vehicles being deflected along or inwards as they hit the boundary of the grid. If such boundary effects were significant, then systematic variation of the network's performance characteristics with the size of the grid would have been observed, because the larger networks have less boundary lane-miles relative to the total lane-miles of the



Note: The Concentration Levels for the Successive Data Points are:
11.81, 22.11, 33.26, 40.26, 48.90 and 55.31 for 5x5 (STOP) and
11.81, 22.18, 33.26, 44.28, 54.58 and 66.06 for 5x5 (SIG)

Fig. 1.10. $\ln T_r$ vs. $\ln T$ and Calibrated Regression Lines for the Signalized and Unsignalized 5x5 Networks.

Table. 1.5. Two-Fluid Parameter Estimation for Signalized Networks of Different Sizes

Network	No. of Data Points	Parameter A		Parameter B		Sum of Squared Errors (SSE)	n	T_m
		Estimate	T-stat	Estimate	T-stat			
5x5 SIG	6	0.129	1.34	0.730	12.50	8.66×10^{-3}	2.70	1.61
5x5 ST	6	0.282	3.87	0.472	18.83	3.58×10^{-3}	0.89	1.71
Pooled	12	0.687	5.97	0.350	7.19	161.4×10^{-3}	0.54	2.88

F-test of difference between the models

$$Q^{\text{unrestricted}} = \sum SSE = 12.24 \times 10^{-3}$$

$$Q^{\text{restricted}} = SSE_{\text{pooled}} = 161.4 \times 10^{-3}$$

$$F^* = \left(\frac{Q^r - Q^u}{r}\right) \Big/ \left(\frac{Q^u}{N - K}\right)$$

$$= \left(\frac{161.4 - 12.24}{2}\right) \Big/ \left(\frac{12.4}{12 - 4}\right)$$

$$= 52.7 >> F_{2,2,0.025} = 5.7$$

So, the lnTr - lnT models for the signalized and unsignalized 5x5 networks are significantly different.

network. The absence of such systematic variation suggests that the validity of earlier results is not compromised by the potential of such a boundary effect.

1.4.2. Experimental Results - Computational Aspects

Since the ability to perform microscopic simulations of realistically large networks is one of the principal motivations for this research, several aspects of the computational requirements deserve serious consideration. The results are expected to yield valuable information for further studies and applications in large networks. Detailed knowledge of the computational aspects of the programs is important in order to take full advantage of the special architectures of supercomputers. In the following analysis, we assess the reduction in execution times on the CRAY relative to mainframe processing both with and without the compiler vectorization capabilities. In addition, information is presented on the specific elements of the program which are particularly computationally demanding, and which would therefore be the obvious candidates for subsequent code modification to remove non-essential inhibitors to vectorization.

The first aspect studied is a comparison of the execution times for the different simulation cases obtained both on the conventional mainframe and the CRAY X-MP/24. These results are shown in Table 1.6. The simulation cases which could be tried on both the CYBER mainframe and the CRAY show that the CPU execution times on the CRAY are only about 10 to 15% of those on the CYBER. Two other trends are also apparent from these results. Firstly, the execution times on the CRAY appear to increase at a somewhat higher rate than on the CYBER as the network size increases. Similarly, for a given network size the execution times on the CRAY increase at a faster rate as the vehicular concentration increases than on the CYBER (about 50% increase in execution time from the 10 veh/mile concentration to the 30 veh/mile concentration for the CYBER, compared to about a 100% corresponding increase on the CRAY). The CRAY version of the NETSIM program used for these simulations was vectorized using the CFT compiler.

Table 1.6. Execution Times

Simulation Network	Target Concentration Level (veh/lane-mile)	Vehicle Loading Time (sec)	Simulation Time (sec)	Execution Times (CPU seconds)	
				CYBER 170 Mainframe	CRAY X-MP/24 Supercomputer
5x5 (SIG)	10	330	600	308.52	25.69
5x5 (SIG)	20	330	600	371.11	38.73
5x5 (SIG)	30	330	600	427.68	50.43
5x5 (SIG)	40	330	600	———	6.105
5x5 (SIG)	50	330	600	———	70.22
5x5 (SIG)	60	330	600	———	77.44
6x6 (SIG)	10	330	600	295.29	35.13
6x6 (SIG)	20	330	600	382.21	55.06
6x6 (SIG)	30	330	600	469.50	73.16
6x6 (SIG)	40	330	600	———	85.40
6x6 (SIG)	50	330	600	———	99.36
6x6 (SIG)	60	330	600	———	108.87
7x7 (SIG)	10	330	600	351.95	45.18
7x7 (SIG)	20	330	600	470.98	74.17
7x7 (SIG)	30	330	600	584.28	99.12
7x7 (SIG)	40	330	600	———	112.12
7x7 (SIG)	50	330	600	———	130.69
7x7 (SIG)	60	330	600	———	141.91
8x8 (SIG)	10	330	600	412.12	24.00
8x8 (SIG)	20	330	600	574.74	34.23
8x8 (SIG)	30	330	600	———	41.70
8x8 (SIG)	40	330	600	———	46.76
8x8 (SIG)	50	330	600	———	53.03
8x8 (SIG)	60	330	600	———	59.17
5x5 (SIG)	10	330	600	———	57.00
5x5 (SIG)	20	330	600	———	62.39
5x5 (SIG)	30	330	600	———	119.10
5x5 (SIG)	40	330	600	———	146.80
5x5 (SIG)	50	330	600	———	167.09
5x5 (SIG)	60	330	600	———	188.06
AUSTIN (ST)	5	600	900	———	291.78
AUSTIN (ST)	10	600	900	———	443.96
AUSTIN (ST)	15	1320	900	———	747.58
AUSTIN (ST)	20	1320	900	———	889.60
AUSTIN (ST)	30	1860	900	———	1297.95

SIG refers to signalized networks
ST refers to unsignalized STOPs-only networks

It is probable that the above trends are due to the presence of coding constructs that inhibit vectorization in the subroutines that perform the computations, and that the effect of these inhibitors increases with the number of vehicles being simulated.

The next aspect considered is the effect of different levels of compiler vectorization on the computational performance of the simulation. The CFT compiler provides the capability to vectorize the program code in blocks of maximum specified limit (in terms of words of internal intermediate text) using the MAXBLOCK control parameter. The execution times for the simulations of both the large Austin network and the 5x5 node test network for different levels of vectorization (values of MAXBLOCK) are shown in Table 1.7. The five cases reported for the Austin network reveal a drop of about 35 to 40% in execution time that can be achieved by using the default vectorization capabilities of the compiler. However, there does not appear to be any significant improvement when a higher block length limit is used. Several additional simulations were performed with the 5x5 test network, and the results are rather interesting. For instance, there is a sharp drop in execution time when a maximum block length of 47 is used as compared to the case with 46. Whereas only about a 6% drop in execution time is obtained from the case of no vectorization to that of a MAXBLOCK of 46, a 25% drop is observed as the value of MAXBLOCK is changed from 46 to 47. Beyond that level, no more than an additional 5% drop is achieved. This aspect of vectorization is closely related to the architecture of the CRAY. It can be concluded that the default value of MAXBLOCK for vectorization of the NETSIM code results in more or less the maximum level of efficiency that can be achieved by compiler vectorization.

Another important aspect of the computational performance of NETSIM that was studied was the relative time-intensiveness of different routines in the program. The CFT compiler provides the capability to keep track of the time spent in the various subroutines of the program. Appendix 1.b shows an example case of the time requirement analysis results

Table 1.7. Execution Times
(for different levels of vectorization)

Simulation Network	Target Concentration (veh/lane-mile)	Vectorization level MAXBLOCK*	CPU Execution Time (seconds)
AUSTIN (ST)	10	1 (no vectorization)	670.57
AUSTIN (ST)	10	2310 (default)	443.95
AUSTIN (ST)	10	4620	441.94
AUSTIN (ST)	20	1 (no vectorization)	1443.66
AUSTIN (ST)	20	2310 (default)	889.61
5x5 (SIG)	30	1 (no vectorization)	74.89
5x5 (SIG)	30	25	72.73
5x5 (SIG)	30	46	70.48
5x5 (SIG)	30	47	53.26
5x5 (SIG)	30	1155	50.66
5x5 (SIG)	30	2310 (default)	50.61
5x5 (SIG)	30	4620	50.57
5x5 (SIG)	30	9240	50.57

SIG refers to signalized networks
ST refers to unsignalized STOPs-only networks

*Maximum length of block of code, in words of internal intermediate text, to be vectorized produced by the CFT compiler. Table 1.8 lists the percentages of time used up by the three most time intensive subroutines in the program (ADJQ, CLNUP, and TRVL) for different levels of compiler vectorization. The most striking aspect of the time requirement analysis is the extreme time-intensiveness of the subroutine CLNUP, which generally accounts for half or more of the execution time. This subroutine performs most of the bookkeeping at the end of each simulation time step, by looping over all the vehicles and all the links in the network. Its time-intensiveness is therefore understandable. Table 1.8 reveals that the percentage of time spent in this routine falls by about 20 to 25% under compiler vectorization. This, coupled with the fact that there is about 35 to 40% drop in overall program execution time under vectorization, means that the subroutine CLNUP vectoizes better than the program overall, causing an overall drop of more than 50% in execution time. There are no significant patterns in the changes in the percentages for other major subroutines. This points to the possibility of concentrating on subroutine CLNUP for further modifications by rewriting the code using more vectorizable structures.

1.5. CONCLUSIONS AND FUTURE DIRECTIONS

1.5.1. Further Possible Modifications of NETSIM

This study has demonstrated that supercomputer capabilities provide the ability to perform microscopic simulation of realistically large urban traffic networks that could not be previously performed on mainframe computers. An encouraging indication from the research is that this was accomplished without exhausting the capabilities of the CRAY. For instance, the simulation of the Austin Central Area Network of 1600 links with 256 lane-miles required only about 30% of the memory capabilities of the CRAY. The CPU time needed for simulating realistic traffic concentrations of 20 to 30 veh/lane-mile was of the order of one half of the real time, even when no recoding of the NETSIM program was attempted to improve vectorizability. This suggests considerable scope for further

Table 1.8. Execution Times
(in the three most time-intensive routines of NETSIM)*

Simulation Network	Target Concentration (veh/lane-mile)	Vectorization level MAXBLOCK**	Percentage of Execution Times †		
			in ADJQ	in CLNP	in TRVL
AUSTIN (ST)	10	1	8.22	59.08	12.19
AUSTIN (ST)	10	2310	10.49	43.85	17.25
AUSTIN (ST)	10	4620	10.51	43.89	17.12
AUSTIN (ST)	20	1	12.60	63.20	8.85
AUSTIN (ST)	20	2310	16.52	47.59	13.28
5x5 (SIG)	30	1	7.23	61.46	18.07
5x5 (SIG)	30	25	7.76	62.75	17.41
5x5 (SIG)	30	46	6.81	62.93	17.55
5x5 (SIG)	30	47	9.63	47.70	24.60
5x5 (SIG)	30	1155	9.11	48.26	25.80
5x5 (SIG)	30	2310	9.05	48.12	25.07
5x5 (SIG)	30	4620	9.06	48.31	24.87
5x5 (SIG)	30	9240	9.06	48.25	24.80

*See Appendix 1.b for an example of the complete break-up of the execution times

** Maximum length of block of code, in words of internal intermediate text, to be vectorized

†See Table 1.7 for the total execution times for these cases.

meaningful improvements, which would make the microscopic simulation of traffic in any large urban network a viable proposition and a tool that can contribute to research in traffic theory as well as to traffic engineering practice.

One of the most important areas for improvement is that of execution speed. A real time to simulation time ratio of 2:1 for a reasonably sized downtown area, albeit acceptable for microscopic simulation, is nevertheless still costly on supercomputers. However, it is clear that this time can be greatly reduced by local recoding of portions of the NETSIM program to achieve greater parallel processing efficiency. In fact, the computational experience with such local vectorization of other network analysis programs during this research (see Chapter 2), indicates that execution times can be reduced to much lower values (even to 20 or 30%) than those obtained with automatic compiler vectorization. Such results can be achieved with relatively minor changes in the program code.

With respect to execution time improvement, this research study was able to produce some pointers for future modifications. The time requirement analysis (see Appendix 1.b and Table 1.8) shows that almost half of the execution time is spent in one subroutine CLNUP, and almost 75% to 80% in just three out of the 60 subroutines (CLNUP, ADJQ, TRVL). Clearly, these are the subroutines that should be concentrated on in further code modifications. The CLNUP subroutine does the bookkeeping after every simulation time step, and in this process loops over all the vehicles and all the links. Some rearrangement of the DO loops in the subroutine could be instrumental in reducing the execution times (See Chapter 2 for examples of vectorizable DO loop structures.)

The memory capabilities of the CRAY of about 4 million words seemed to be more than adequate, and no immediate modifications are warranted in the program for more efficient memory management. It appears that urban areas of up to even 5 or 6 times the size of the Austin network tested in this study can be managed within the 4 megaword memory limit of CRAY. One possibility along these lines is to have a preprocessor for the NETSIM code itself, which does intelligent array dimensioning depending on the problem at hand and recompiles the code. For example, if the urban area does not have many actuated signals, the memory dimensions needed for actuated signals can be reduced. Similar modifications are possible, based on whether any bus-routes analysis is needed, whether fuel consumption analysis is needed, etc. Again, this would be needed only for some extremely large networks.

1.5.2. Further Research Possibilities

Having demonstrated the feasibility of performing microscopic simulations of large networks a wide spectrum of possibilities become available for simulation-based studies of real urban traffic networks. This research initiated a first step in this direction in the area of the characterization of the quality of urban traffic based on network-level traffic parameters. Several other meaningful directions for future traffic research should be considered.

In the area of the two-fluid theory of urban traffic, further investigation of the behavior of the traffic system under extremely congested conditions is worth pursuing. As the Austin network used for this study was assumed to have only STOP control at the intersections, definitive conclusions could not be reached on this aspect. Simulating networks with different types of signalized control at different traffic concentrations would be useful, as would the study of the traffic system under short and long-term events (closing of the full length of a street for construction, for example). Actually, such simulations have been conducted by Williams et al. (1985), though only for a 5x5 grid network. conducting similar tests on larger realistic networks on the CRAY is now possible. Another aspect of interest is the effect of the topology of large networks, and of configurations of the main streets on network performances and traffic quality.

Another important application area that could be addressed in future research is that of strategies for electronic route guidance in urban traffic networks. The basic telecommunications and microprocessor hardware technologies for in-vehicle route guidance are largely available; however, the integration of these technologies for the purpose of purveying real-time information to vehicles remains in its infancy. Furthermore, important system design questions must be addressed, pertaining to the type and frequency of supplied information, as well as the extent of its availability to network users. Microscopic simulation provides an obvious tool for evaluating the effectiveness of various strategies and supporting the design of such systems. However, to conduct such studies using NETSIM, some modifications are needed in the code, especially with regard to the routing of vehicles in the network. An additional subroutine to determine users' route choice in response to information could be incorporated into the program for this purpose.

Appendix 1.a. Additional Output Generation

FORTRAN output unit number	Output written
41-58	Individual vehicle information for every specified vehicle in the network. The vehicles are specified in the subroutine CLNUP. Information on the vehicle position, link occupied, headway in front, etc., are available. More variables can easily be added.
35	Vehicle speed distribution at the end of each cumulative output. The frequency of vehicles in 100 speed classes, (0 ft/sec to 100 ft/sec). A 100-element array called IVEL is the output. IVEL(J) = The number of of times that a vehicle with speed between (J-1) ft/s and J ft/s is encountered, at specified sampling intervals.
32	Information on particular links in the network
36	Total number of time steps the vehicles are either moving or stopped

Appendix 1.b. An Example of the Results of the Time-Requirement Analysis. (Austin Network, Vectorization of MAXBLOCK = 2310, Concentration = 20 veh/lane-miles)

```
F L O W T R A C E  --  Alphabetized summary

   Routine      Time executing   Called   Avg T
29 ADJQ      115.442    ( 16.52%)1422131      > Called by     GOQ      MOVE
53 BUSDIS      0.002    (  0.00%)   2280    >>> Called by   CLNUP
17 BUSOT2      0.006    (  0.00%)      6      > Called by  PREPRO
49 CLNUP     332.613    ( 47.59%)   2280  0.146 Called by   SIMUL
 8 CLRALL      0.024    (  0.00%)      2  0.012 Called by  PREPRO
52 CYCP        5.129    (  0.73%)      7  0.733 Called by   OUTPT
 2 DRWS        8.930    (  1.28%)  28526      > Called by  FLSORT    PREPRO
                                                   PRMSND   PRSIG    UTCSUP
43 EXPES       0.138    (  0.02%) 107160      > Called by    SVEH
16 FLOOUT      0.046    (  0.01%)      6  0.008 Called by   FLOWS
13 FLOWS           >    (  0.00%)     18      > Called by  PREPRO
 5 FLSORT      0.034    (  0.00%)      1  0.034 Called by  PREPRO
26 GETCD       2.904    (  0.42%) 181829      > Called by    LANE
23 GOQ         3.211    (  0.46%) 257644      > Called by   LFTRN      MOOV
                                                     MOOV
30 HDWY        0.864    (  0.12%)  63206      > Called by    ADJQ      MOVE
 9 INT1        2.576    (  0.37%)     12  0.215 Called by  PREPRO
51 INTST       3.340    (  0.48%)      4  0.835 Called by   OUTPT
25 LANE        5.140    (  0.74%) 181829      > Called by     GOQ      SVEH
40 LFTRN           >    (  0.00%)     72      > Called by    MOOV
12 LNKSIG      0.020    (  0.00%)  16564      > Called by   PRSIG     UPSIG
34 LSWCH       0.716    (  0.10%)  45017      > Called by   CLNUP      TRVL
37 MOOV       37.241    (  5.33%)   2280  0.016 Called by   SIMUL
20 MOVE       35.219    (  5.04%)   2280  0.015 Called by   SIMUL
50 OUTPT           >    (  0.00%)     11      > Called by   CLNUP     SIMUL
 3 PREPRO      6.610    (  0.95%)      7  0.944 Called by  UTCSUP
46 PRKD        2.146    (  0.31%)   2280      > Called by   SIMUL
14 PRMSND      0.008    (  0.00%)     12      > Called by   FLOWS
10 PRSIG       0.024    (  0.00%)      2  0.012 Called by  PREPRO
57 RESET       0.013    (  0.00%)      1  0.013 Called by  STABLE
22 RNDM        3.060    (  0.44%) 499302      > Called by   GETCD       GOQ
                                                     HDWY   STPSN      SVEH
18 SIGOUT      0.378    (  0.05%)      2  0.189 Called by  PREPRO
19 SIMUL       0.158    (  0.02%)      6  0.026 Called by  UTCSUP
56 STABLE          >    (  0.00%)      1      > Called by   SIMUL
21 STPSN      14.275    (  2.04%) 978086      > Called by    MOVE
42 SVEH        0.336    (  0.05%)   2280      > Called by   SIMUL
 7 THREAD      0.073    (  0.01%)      6  0.012 Called by  FLSORT
32 TRVL       92.834    ( 13.28%)2842142      > Called by     GOQ      MOOV
33 TSIG        8.095    (  1.16%)1061465      > Called by    TRVL
28 TSTSAT      3.656    (  0.52%) 257644      > Called by     GOQ
47 UPSIG      13.652    (  1.95%)   2280  0.006 Called by   SIMUL
 1 UTCSUP          >    (  0.00%)      1      >

* * * TOTAL      698.915    7958682 Total calls
```

CHAPTER 2. VECTORIZATION OF NETWORK EQUILIBRIUM ASSIGNMENT ALGORITHMS FOR ONE AND TWO CLASSES OF USERS

2.1. INTRODUCTION

The main objective of this chapter is to examine the computational improvements that can be achieved by running two network traffic assignment codes, for the single class and the two-class user equilibrium problems, on the CRAY X-MP supercomputer. The network traffic assignment problem arises in connection with many transportation planning activities, including the analysis of the cost-effectiveness of capital improvement projects and the evaluation of operational planning strategies in traffic networks. Both codes are also used in the more general transportation network design problem, which is a non-polynomial hard problem that cannot generally be solved optimally using current computational techniques. In view of the CRAY's high speed and ability to handle vectorization, the two codes were tested to determine if the reduction in execution time is substantial, especially after modification aimed at taking advantage of the capabilities of the CRAY's architecture. The results have important implications for practice, in terms of the size and complexity of the problems that can be addressed, and more importantly, for the future development of solution approaches to the network design problem.

2.2. GENERAL GUIDELINES

The introduction of supercomputers, faster and capable of handling vector and parallel processing, enables the solution of problems which had previously been considered unsolvable on slower machines. However, they have increased the programmers' responsibilities, requiring a better understanding of computer architectures and characteristics. This point is emphasized by many researchers (Zenios and Mulvey, 1986) as well as CRAY consultants (CHPC User Services, 1987a, b). The main feature that is emphasized from the programming point of view is the vectorization of the DO LOOPS, which generally are the main source of computer time expense. Emphasis is given to the identification and modification of those programming steps which inhibit vectorization, given that the CRAY X-MP compiler automatically tries to vectorize the loops where applicable. Some recommended steps for transforming a code initially intended for a scalar process to one that can take advantage of vector processing capabilities are summarized below.

The first step is to perform a time requirements analysis of the computer code to determine the parts of the code which consume most of the execution time. The user should then employ a combination of compiler directives and program optimization by identifying those parts of the code which inhibit vectorization. Then, having achieved a certain level of improvement, the user should try to attempt code restructuring, eventually considering different algorithms which may be implemented in a way that can better address the vector and parallel processing capabilities of the computer. This process is iterative, and can be continued until the programmer is satisfied that no significant changes can be made. These general steps are recommended by the UT CRAY group (CHPC User Services, 1987a, b), as well as most other experienced users (see, for example, Zenios and Mulvey, 1986).

The vectorization of the DO LOOPS is emphasized as a first step towards improvement of the code. When trying to determine whether or not to vectorize a particular DO LOOP, the CFT compiler checks for the existence of any dependencies within the loop. The CFT compiler produces a code which contains vector instructions to drive the high-speed vector and floating point functional units and the eight vector registers in their specified operation. Vectors are processed in a pipeline fashion; after an initial startup period the first result appears, followed by the other results, one every cycle. The compiler, to be on the safe side, does not attempt vectorization when it recognizes certain dependencies within the loop. Below are some statements which should be avoided within the DO LOOPS, according to the UT CHPC User Services Group (1987a):

- CALL statements;
- I/O statements;
- branches to statements not in the loop;
- inner DO LOOPS that are not unrolled;
- backward branches within the loop;
- statement numbers with references from outside the loop;
- references to character variables, arrays, or functional IF statements which may not execute due to the effects of previous IF statements;
- ELSEIF statements;
- external function references not declared in a CDIR $ VFUNCTION directive;
- bounds checking on any array referenced in the loop;
- specifying the DEBUG option;
- loop size exceeds the optimized MAXBLOCK size; and
- loop has been unrolled or replaced by a $SCILIB routine.

Furthermore, as a guideline to vectorize the two codes, we followed some techniques suggested in publications from both the UT CHPC User Services (1987) and the San Diego Supercomputer Center (June 1987), such as:

1) Eliminate data dependencies; a loop will not vectorize if, for example, an array is referencing values dependent on computations in lower positions of the array in an incrementing loop. The computations cannot be pipelined.

2) Eliminate subscript ambiguities; try to eliminate the dependency of a subscript on a previous calculation, i.e., include the operation in the array.

3) Assign as the innermost loop the one with the largest range; the code is more effective when the inner loop is the largest one since it is the only one that is vectorized.

4) Eliminate conditionals; IF THEN ELSE statements can be replaced by conditional vector merge procedures. Simple IF statements are vectorizable, but depending on their reference they might inhibit vectorization.

5) Unroll the loops to a certain depth; it eliminates checking for termination conditions and enforces chaining and functional unit overlap.

6) Separate vectorizable from unvectorizable loop; if possible, separate loops which contain CALL statements or I/O statements or any of the statements mentioned previously which are independent of the other computations within the loop.

Before describing the application of these rules to the network assignment codes considered in the study, we present, in the next section, the basic steps of the algorithms to solve for the Single Class User Equilibrium and the Multiclass User Equilibrium with asymmetric costs.

2.3. THE NETWORK EQULIBRIUM ALGORITHMS

The single user equilibrium algorithm is an iterative procedure that solves for the flows onto the links of a transportation network by assigning given trips between origin and destination points in a way that achieves certain equilibrium conditions in the network. Users are assumed to choose the path with the minimum travel time between their origin and their destination. However, because link travel time (costs) depend on the prevailing link flow, it is necessary to solve jointly for link flows and travel times. This solution is based on the principle that no driver can improve his travel time by unilaterally switching routes when the equilibrium state is reached (Wardrop, 1952). The single class user equilibrium problem is based on Beckman's equivalent mathematical programming formulation (1956), and can therefore be solved by any of several nonlinear optimization techniques. The most widely used algorithm for its solution is based on the Frank-Wolfe (1956) or convex combinations method.

The two-class user equilibrium problem arises when two classes of users, e.g., cars and trucks, share the use of the physical right-of-way on the highway facilities. The travel times (costs) experienced by one class of users depend not only on the flow of elements belonging to that class, but also on the flow of the other class. When the respective effects of the flow of one class on the travel time of the other are not symmetric (e.g., the effect of one additional truck on the cars' average travel time is greater than the effect of an additional car on the trucks' travel time), the resulting user equilibrium problem does not have an equivalent mathematical programming formulation. The algorithm used for its solution is a direct algorithm, called the diagonalization algorithm. Both algorithms have similar computational characteristics, mainly a shortest path routine, a line search routine (to find the optimal move size along a particular direction), and an all-or-nothing assignment routine. The principal cost is due to the frequent use of the shortest path routine followed by the move size calculation. A considerable cost is also incurred in the computation of the relatively complicated travel cost functions. The principal steps of the two algorithms are presented hereafter. Note that the functions $t_a(.)$ denote the link performance (cost) functions, which capture the dependence of link travel time (cost) on link flows.

Steps of the single class user equilibrium algorithm:

STEP 0: Initialization. Perform all-or-nothing assignments based on the free flow travel times $t_a = t_a(0)$, $\forall a$;

This yields the set of link flows $\{X_a^1\}$. Set counter n = 1.

STEP 1: Update. Set $t_a^n = t_a(X_a^n)$, $\forall a$.

STEP 2: Direction finding. Perform all-or-nothing assignment based on $\{t_a^n\}$. This yields a set of (auxiliary) link flows $\{y_a^n\}$.

STEP 3. Line search. Find α_n that solves $\min \sum_a \int_0^{x_a^n + \alpha (y_a^n - x_a^n)} t_a(w)dw$

subject to $0 \leq \alpha_n \geq 1$.

STEP 4: Set $X_a^{n+1} = X_a^n + \alpha_n (y_a^n - X_a^n), \forall a$.

STEP 5: Convergence test. If a convergence criterion is met, STOP (the current solution, $\left\{X_a^{n+1}\right\}$ is the set of equilibrium link flows); otherwise, set n = n+1 and GO TO STEP 1.

In the solution of the two-class user equilibrium problem, a separate copy of the physical network is created for each class of users. The interactions between classes sharing the same physical links are then represented through the performance (cost) functions associated with each link in the individual network copies. In the general case, these functions would specify the dependence of a link 's travel cost on flows on any other link. In the two-class case, the specification of the cost functions reflects the desired dependence between user classes as interactions among links.

Steps of the diagonalization algorithm:

STEP 0: Initialization. Find a feasible link flow vector X^n. Set n = 0.

STEP 1: Diagonalization. Solve the following subproblem:

$$\min \tilde{Z}^n(X) = \sum_\varepsilon \int_0^{X_\varepsilon} t_\varepsilon (X_1^n, ..., X_{\varepsilon-1}^n, w, X_{\varepsilon+1}^n, ..., X_E^n)dw$$

subject to

$$\sum_k f_k^{rs} = q_{rs}, \quad \forall k, r, s$$

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

where

the subscript ε denotes a link of the combined network (which includes as many copies of the physical network as there are different classes of users),

X_ε is the flow on link ε

f_k^{rs} denotes path k from origin r to destination s; and

q_{rs} denotes the total flow from origin r to destination s.

STEP 2: Convergence test. If a convergence criterion is met, STOP (the current solution $\left\{X_a^{n+1}\right\}$ is the set of equilibrium link flows); otherwise, set $n = n+1$ and GO TO STEP 1.

STEP 1 is solved using the Frank-Wolfe algorithm, where at the n-th iteration all cross-link effects are fixed and the flow on one link depends only on its own flow. It is more computationally demanding than the single class algorithm, because there are as many origin-destination trip matrices as there are classes of users, which means greater use of the shortest path and the all-or-nothing assignment procedures. Furthermore, the travel cost functions are more complicated, increasing the computational burden for the move size finding.

Nevertheless, the two computer codes do not differ significantly from each other. They are composed of the same subroutines, with some modifications in the two-class UE to take into account the division of the traffic into trucks and passenger cars. Following is a descriptive summary of how the algorithmic steps are implemented in the two computer codes.

The input of the characteristics of the network, the Origin-Destination (O-D) matrices, link characteristics and convergence measures, are included in TRAFASN and UETRDIA for the Single Class UE and the Two-Class UE, respectively. The initialization STEP 0 takes place in subroutines UE and UED, respectively, where all the main steps of the algorithms are controlled. Following the initialization of all the paths to zero flows, subroutine AON (AONUED for the two-class diagonalization code), is called to initialize the flows on the links to zero. Then the travel times on the links are computed, initially with zero flows. Given these travel times, subroutine SHPATH (SHPUED) is called, as many times as the number of O-D pairs, which identifies the shortest path for each O-D pair. Then the flow for each O-D pair is allocated on the links which comprise each shortest path. The calculation of the travel times and the allocation of the flows to the links (all-or-nothing assignment), correspond to STEP 1 and STEP 2 of the Single Class UE algorithm, respectively. These steps are all included in STEP 1 of the diagonalization algorithm, which, as noted earlier, mainly involves the solution of many Single Class UE problems. STEP 3 of the Single Class UE is controlled by subroutine BISECT (BISUED), where the step size is determined by a line search using the bisection method. Then the flows are updated in STEP 4 using the step size calculated in STEP 3 and finally, STEP 5 (STEP 2), the convergence test, is calculated in subroutine UE (UED). For the diagonalization algorithm, we have the additional burden of the internal convergence test for each UE subproblem. The output of the program is controlled by subroutine DUMP (DUMPUED), mainly reporting the flows on the links, the convergence measure, the number of iterations and the volume-to-capacity ratio for each link. A more detailed description of the two computer codes can be found in Mouskos et al. (1986).

The main computational effort is required by the shortest path routine, its cost depending on the size of the network and the number of O-D pairs. The computational effort required by the step size determination routine is also dependent on the size of the network, due to the repetitive calculation of the link travel times and the iterative nature of the line search. The computation of link travel times is also the major burden in the all-or-nothing and the output subroutines. The convergence measure determination is the main source of time expense in the UE (UED) routine. In the following sections, the modifications to the two codes to enhance their vectorizability are described, along with the computational results obtained in connection with the numerical experiments performed to test these modifications.

2.4. COMPUTATIONAL RESULTS FOR SINGLE CLASS UE CODE

The algorithm was tested on a network with 364 O-D pairs, 128 nodes, and 336 links. This is considered a medium-size transportation network, and has been used in earlier tests of a total of 500 iterations were allowed before the code was terminated for any test run.

The first step before doing any changes to the code was to execute the program without the vectorizing capabilities of the CFT compiler and perform a time analysis. The results are shown in Table 2.1. To total time to execute was 15.679 seconds, using the CFT compiler (with vectorization blocked using the statement MAXBLOCK = 1).

Next, the program was executed by removing the prohibition for vectorization, using both the CFT and the CFT77 compilers. The results are shown in Tables 2.2 and 2.3, respectively. From the results, it can be seen that the execution times for some of the routines were reduced considerably. A total reduction of 4.404 secs (or 28.1%) has been achieved by the vectorized compilation (See Tables 2.1 and 2.2) (without any program modification). However, the functions COSTFN and FINT have not been reduced much (by 0.234 and 0.022 sec, or 3.9% and 4.3%, respectively), thereby pointing our efforts towards seeking an opportunity to improve those functions.

One strategy to achieve such improvement is to include the travel cost functions within the BISECT routine, in which the COSTFN function is called repeatedly. As noted earlier, calling functions or subroutines in a loop sometimes prohibit vectorization. The execution time summary following this modification, is given in Table 2.4. As can be seen, there is a reduction of 1.274 sec (or 11.3%). However, it was suspected that a further data dependency existed in the loop for calculating the travel times which inhibited vectorization. More specifically, the separate calculations of the parameters A1 and B1 of

SUBROUTINE	TIME EXECUTING (Seconds)	(PERCENTAGE TIME)
AON	2.195	(14.00)
BISECT	3.065	(19.55)
COSTFN	5.928	(37.81)
DUMP	0.222	(1.41)
FINT	0.507	(3.24)
SHPATH	3.330	(21.24)
TRAFASN	0.051	(0.32)
UE	0.381	(2.43)
	TOTAL 15.679	

Table 2.1. Execution Times for Each Subroutine When Vectorization is Blocked (MAXBLOCK = 1 in CFT Compiler)

SUBROUTINE	TIME EXECUTING (Seconds)	(PERCENTAGE TIME)
AON	1.430	(12.68)
BISECT	1.813	(16.08)
COSTFN	5.694	(50.50)
DUMP	0.215	(1.91)
FINT	0.485	(4.30)
SHPATH	1.391	(12.33)
TRAFASN	0.050	(0.44)
UE	0.198	(1.75)
TOTAL	11.275	

Table 2.2. Execution Times for Each Subroutine with Vectorization Using the CFT compiler

SUBROUTINE	TIME EXECUTING (Seconds)	(PERCENTAGE TIME)
AON	0.917	(8.99)
BISECT	1.517	(14.87)
COSTFN	5.785	(56.72)
DUMP	0.201	(1.97)
FINT	0.487	(4.78)
SHPATH	1.055	(10.34)
TRAFASN	0.048	(0.47)
UE	0.191	(1.87)
	TOTAL 10.199	

Table 2.3. Execution Times for Each Subroutine with Vectorization Using the CFT77 Compiler

SUBROUTINE	TIME EXECUTING (Seconds)	(PERCENTAGE TIME)
AON	1.424	(14.24)
BISECT	5.746	(57.46)
COSTFN	0.481	(4.80)
DUMP	0.213	(2.13)
FINT	0.489	(4.89)
SHPATH	1.402	(14.02)
TRAFASN	0.050	(0.50)
UE	0.196	(1.96)
	TOTAL 10.001	

Table 2.4. Execution Tims Summary Following the Modification of BISECT to Calculate the Travel Time Within BISECT Instead of Calling Function COSTFN (with Vectorization Using CFT Compiler).

the link performance functions were thought to inhibit vectorization because they could not be recognized by the CFT compiler. The expressions for these parameters were therefore included directly in the travel time equation. The above changes in the code are exhibited below:

original loop in bisect:

```
       DO 30 N=1, NARC
       X = FL(N) + AMD*(NFL(N)-FL(N))
       A1 = ALP (TYP(N))
       B1 = BET(TYP(N))
       CST = COSTFN (L(N), C(N), V(N), X, A1, B1)
30     D = D + CST*(NFL(N) - FL(N))
```

1st Change: Removing the call function COSTFN

```
       DO 30 N=1, NARC
       X = FL(N) + AMD* (NFL(N) -FL(N))
       A1 = ALP (TYP(N))
       B1 = BET(TYP(N))
       CST = L(N)/V(N)
       IF(C(N). NE.0) CST = CST*(1 + A1*(X/C(N))**B1)
30     D = D + CST*(NFL(N) - FL(N))
```

2nd Change: Incorporating expressions for A1 and B1 directly in the cost (CST) calculation

```
       DO 30 N=1, NARC
       X = FL(LN) + AMD* (NFL(N) - FL(N))
       CST = L(N)/V(N)* (1+ ALP(TYP(N))*(X/C(N))**BET(TYP(N)))
30     D = D + CST*(NFL(N) - FL(N))
```

With the last change shown above, the program was executed by both compilers. As it can be seen from the results in Tables 2.5 and 2.6, there was a dramatic drop in

SUBROUTINE	TIME EXECUTING (Seconds)	(PERCENTAGE TIME)
AON	1.433	(25.73)
BISECT	1.328	(23.85)
COSTFN	0.477	(8.56)
DUMP	0.211	(3.79)
FINT	0.475	(8.53)
SHPATH	1.411	(25.34)
TRAFASN	0.050	(0.90)
UE	0.184	(3.30)
	TOTAL 5.568	

Table 2.5. Execution Time Summary Following Modification of BISECT by Including the Parameters A and BET in the Travel Cost Equation (with Vectorization Using CFT Compiler).

SUBROUTINE	TIME EXECUTING (Seconds)	(PERCENTAGE TIME)
AON	0.931	(22.93)
BISECT	0.654	(16.09)
COSTFN	0.494	(12.15)
DUMP	0.203	(4.99)
FINT	0.488	(12.02)
SHPATH	1.061	(26.12)
TRAFASN	0.047	(1.17)
UE	0.184	(4.52)
	TOTAL 4.061	

Table 2.6. Execution Time Summary for Same Code as Table 2.5, but with Vectorization Using CFT77 compiler

execution time from 10.001 to 5.568 secs (or a 44.3% improvement), primarily due to a drop in BISECT from 5.746 to 1.328 (or a 76.9% improvement), confirming the prior existence of a dependency which had inhibited the vectorization of the loop.

Given the above results, similar changes were made wherever the functions COSTFN and FINT were called. The time analysis after those changes were implemented is shown in Table 2.7. The total execution time was reduced from 4.061 secs to 3.325 secs (using the CFT77 compiler), or an 18.1% decrease.

In the same subroutine, a further step was to specify the $\frac{1}{C(N)}$ as a variable $C^1(N)$ so that X/C(N) was transformed to $X*C^1(N)$, which eliminates the repetitive division. Furthermore, the equation X = FL(N) + AMD*(NFL(N) - FL(N)) was taken out of the loop as follows:

```
      DO  29  N=1, NARC
      DNFL(N) = NFL(N) - FL(N)
   29 CONTINUE
      DO  31  N = 1, NARC
      X(N) = FL(N) + AMD*DNFL(N)
   31 CONTINUE
      DO  30  N=1, NARC
      CST = L(N)/V(N)*(1. + ALP(TYP(N))*(**C1(N))** BET(TYP(N))
   30 D = D + CST*DNFL(N).
```

Also, the following loop was similarly changed:

original form

```
      DO  300 N=1, NARC
      NFL(N) = FL(N) + AMD*(NFL(N) - FL(N))
  300 CONTINUE
```

modified form

```
     DO  300 N=1, NARC
     NFL(N) = FL(N) + AMD*DNFL(N)
 300 CONTINUE
```

The results of this change are given in Table 2.8 indicating a decrease from 3.325 to 3.301 (0.024 sec, less than a 1% reduction). Thus, the improvements from these further changes were not as significant as the earlier ones.

2.5. COMPUTATIONAL RESULTS FOR THE TWO-CLASS DIAGONALIZATION CODE

The diagonalization algorithm was applied to a relatively large network, with two classes of vehicles operating on it. The network consists of 364 O-D pairs, 1400 nodes, and 3912 links. A total of 25 iterations were allowed before the code was terminated for all test runs.

A similar procedure was followed as with the single class user equilibrium code. First, a time analysis was performed using the CFT compiler options ON=F and MAXBLOCK=1 to block vectorization. The results are given in Table 2.9. Table 2.10 shows the time analysis with MAXBLOCK=1 removed, thus allowing compiler vectorization to take place. Comparing the results, we see an improvement from 23 seconds to 13 seconds (i.e., a 41.5% reduction). The routines which showed improvement are the AONUED (the all-or-nothing procedure), BISUED (the direction finding procedure), DUMPUED (the output routine), SHPUED (the shortest path routine), and UETRDIA (the input routine). No improvement occurred for the travel cost function TRCOST and the UED (the routine calling the other routines and performing the general diagonalization process). These results parallel those obtained with the single class UE algorithm. Therefore, similar changes in the code were implemented, as shown below:

SUBROUTINE	TIME EXECUTING (Seconds)	(PERCENTAGE TIME)
AON	0.850	(25.56)
BISECT	0.656	(19.74)
DUMP	0.208	(6.25)
SHPATH	1.058	(31.83)
TRAFASN	0.048	(1.45)
UE	0.504	(15.17)
	TOTAL 3.325	

Table 2.7. Execution Time Summary Following Further Modification by Removing Calls to Separate Functions COSTFN and FINT (with Vectorication Using CFT77 compiler).

SUBROUTINE	TIME EXECUTING (Seconds)	(PERCENTAGE TIME)
AON	0.847	(25.66)
BISECT	0.641	(19.43)
DUMP	0.206	(6.23)
SHPATH	1.056	(31.99)
TRAFASN	0.048	(1.47)
UE	0.505	(15.23)
	TOTAL 3.301	

Table 2.8. Execution Time Summary Following Further Modifications in the BISECT Routine, Mainly Decomposing a Loop to Smaller Loops (with Vectorication Using CFT77 compiler).

SUBROUTINE	TIME EXECUTING (Seconds)	(PERCENTAGE TIME)
AONUED	1.192	(5.17)
BISUED	5.726	(24.84)
DUMPUED	0.434	(1.88)
SHPUED	10.033	(43.52)
TRCOST	4.608	(19.99)
UED	0.245	(1.06)
UETRDIA	0.815	(3.54)
TOTAL	23.053	

Table 2.9. Execution Times Summary for the Diagonalization Code, with Vectorization Blocked (MAXBLOCK = 1, on CFT compiler).

SUBROUTINE	TIME EXECUTING (Seconds)	(PERCENTAGE TIME)
AONUED	0.661	(4.90)
BISUED	2.856	(21.17)
DUMPUED	0.201	(1.49)
SHPUED	4.389	(32.54)
TRCOST	4.440	(32.91)
UED	0.125	(0.93)
UETRDIA	0.816	(6.05)
TOTAL	13.489	

Table 2.10. Execution Time Summary for the Diagonalization Code, with Vectorization Using the CFT Compiler.

BISUED routine - Original code for LOOP 30:

```
DO  30  N=1, NARC/2
    X1 = NFLI(N) + (NFL(N) - NFLI(N))*AMD
    A1 = ALP(TYP(N))
    B1 = BET(TYP(N))
    E1 = E(TYP(N))
    K = N+NARC/2
    C1 = C(N)
    C2 = C(K)
    FLA = FL(N)
    FLT = FL(K)
    T1 + T(N)
    T2 = T(K)
    CST1 = TRCOST(L(N), C1, C2, VEL(N), X1, FLT, A1, B1, E1, T1, T2,
               RL(N))
    X2 = NFLI(K) + AMD*(NFL(LK) - NFLI(K))
    A2 = ALP(TYP(K))
    B2 = BET(TYP(K))
    E2 = E(TYP(K))
    CST2 = TRCOST (L(K), C1, C2, VEL(K), FLA, X2, A2, B2, E2, T2,
               T1, RL(K))
30  D = D + CST1*(NFL(N) - NFLI(N)) + CST2*(NFL(K) -NFLI(K)
```

BISUED routine - modified code for LOOP 30

```
      DO 30 N=1, NARC/2
      X1 = NFLI(N) + (NFL(N) - NFLI(N))*AMD
      K = N + NARC/2
      X2 = NFLI(K) + AMD*(NFL(K) - NFLI(K))
      CST1 = L(N)*RL(N)/VEL(N)*(1. + ALP(TYP(N))* (X1*T(N)/C(N) +
             E(TYP(N))*FL(K)*T(K)/C(K))**BET(TYP(N)))
      CST2 = L(K)*RL(K)/VEL(K)*(1. + ALP(TYP(K))* (FL(N)*T(K)/C(N) +
             E(TYP(K)*X2*T(N)/C(K))**BET(TYP(K)))
      D = D + CST1*(NFL(N) - NFLI(N)) + CST2*(NFL(K) - NFLI(K))
   30 CONTINUE
```

Again, as in the BISECT routine of the single class UE code, the dependencies were removed and entered into the equation of the travel cost function which is no longer calculated as a separate function call statement. This allows the compiler to vectorize this loop. Table 2.11 indicates that these modifications led to a reduction in execution time for this subroutine from 5.726 secs for the non-vectorized form to 0.861 secs for the vectorized modified one, i.e., a remarkable 85% reduction.

Additional changes in subroutine AONUED were implemented as follows.

AONUED routine - initial code LOOP 10

```
DO 10 N=1, NARC
A1 = ALP(TYP(N))
B1 = BET(TYP(N))
E1 = E(TYP(N))
IF (N.LE.NARC/2) THEN
          K = NARC/2 + N
          NFL(N) = 0
          NFL(K) = 0
          C1 = C(N)
          C2 = C(K)
          FLA = NFLI(N)
          FLT = FL(K)
          T1 = T(N)
          T2 = T(K)
ELSE
          K = N - NARC/2
          NFL(K) = 0
          NFL(N) = 0
          C1 = C(K)
          C2 = C(N)
          FLA = FL(K)
          FLT = NFLI(N)
          T2 = T(K)
          T1 = T(N)
ENDIF
10 COST(N) = TRCOST (L(N), C1, C2, VEL(N), FLA, FLT, A1, B1, E1,
              T1, T2, RL(N))
```

AONUED routine - Changed code LOOP 10

```
DO  10  N = 1, NARC/2
        NFL(N) = 0
        K = NARC/2 + N
        NFL(K) = 0
COST(N) = L(N)*RL(N)/VEL(N)*(1. + ALP(TYP(N))*(NFLI(N)
            *T(N)/C(N) + E(TYP(N))*FL(K)*T(K)/C(K))
            **BET(TYP(N)))
COST(K) = L(K)*RL(K)/VEL(K)*(1. + ALP(TYP(K)*(FL(N)
            *T(K)/C(N) + E(TYP(K))*NFLI(K)*T(N)/C(K))
            **BET(TYP(K)))
10 CONTINUE
```

The basic changes made above are: 1) the removal of the IF THEN ELSE statement, 2) the removal of the call function TRCOST which was substituted by the expression of the function within the loop, and 3) removal of the dependencies by including the variables into the equation. Although there is a dependency due to the calculation of K, the loop is still vectorized by the compiler. The overall execution time of this subroutine changed from 1.192 to 0.423 seconds (a 64.5% reduction), with the modifications in the above loop accounting for the change in execution time from 0.661 to 0.423 seconds (or a 36% reduction).

The above are the major changes that accounted for the principal improvements obtained in this effort. Other minor changes involved the minimization of the division operations, as in the single-class case, by defining $C^1(N) = \frac{1}{C(N)}$ and unrolling of some DO LOOPS. However, these changes did not lead to further significant reductions. Although all possibilities may not have been exhausted, a remarkable 70% reduction, from 23 seconds to approximately 7 seconds, has been achieved, corresponding to a non-

SUBROUTINE	TIME EXECUTING (Seconds)	(PERCENTAGE TIME)
AONUED	0.423	(6.26)
BISUED	0.861	(12.74)
DUMPUED	0.217	(3.21)
SHPUED	4.326	(63.99)
UED	0.125	(1.85)
UETRDIA	0.808	(11.95)
	TOTAL 6.759	

Table 2.11. Execution Times Summary Following Modifications in BISUED and AONUED Subroutines (Vectorized Using CFT Compiler).

vectorized to vectorized improvement ratio in excess of 300%. It should also be noted here that the changes made also enhance the scalar performance of the computer code's execution, as can be seen in Table 2.12 where a time analysis was performed with blocked vectorization after the changes were made. The total execution time was 17.427 sec compared to the 23.053 sec of the original code (in Table 2.9). The corresponding codes required 182 seconds to execute on the CYBER CDC 170/750.

2.6. CONCLUSION

2.6.1 Summary of Results

The main objective of this study was to examine the possible reductions in execution time that can be achieved, using the CRAY X-MP/24 supercomputer on two computer codes used for traffic assignment -- the single class user equilibrium and the two-class user equilibrium. In both cases, a considerable reduction in execution time has been achieved: 80% and 70%, respectively, over the unvectorized execution. Our experience confirms the effectiveness of the recommendations followed to optimize these two FORTRAN codes, mainly trying to avoid dependencies, IF and CALL statements within the DO LOOPS. Inserting in line the travel cost functions proved very helpful in both cases.

In both algorithms, the shortest path routine contributes most to the computation time, 32% for the single class UE and 64% for the two-class UE, even after reduction by about 70% in the first case and 55% in the second case. A possible change in the codes might be to try other shortest path routines as well as a different method to calculate the move size.

SUBROUTINE	TIME EXECUTING (Seconds)	(PERCENTAGE TIME)
AONUED	0.927	(5.31)
BISUED	5.171	(29.60)
DUMPUED	0.370	(2.12)
SHPUED	9.953	(56.98)
UED	0.244	(1.40)
UETRDIA	0.802	(4.59)
	TOTAL 17.467	

Table 2.12. Vectorization Blocked (MAXBLOCK=1, CFT compiler).

2.6.2. Directions for Future Research

The encouraging results obtained in this study allow some optimism towards running the transportation network design code on the CRAY and attempting to vectorize it. In our previous attempts on the Cyber CDC computer, the above code required about 300 seconds to execute when tested over a trivial 3 link network! Given that the traffic assignment routine for the two-class user equilibrium problem required about 200 seconds to run on the Cyber for a large realistic network, it became evident that it would not be feasible to run the network design code, of which the traffic assignment routine is a part.

The network design problem by itself is computationally demanding as it falls into the category of np-hard problems. Given a network, with known O-D matrices for each of the categories of users of the network, and a number of links n, the problem is to propose various improvements to the links so as to improve operating conditions and service levels offered by the network. Assuming a problem with k improvement options for each link, its combinatorial complexity rises to k^n. In transportation networks, the calculation of the travel costs associated with a particular combination of improvements requires the application of a traffic assignment procedure, which may attempt to achieve either a User Equilibrium solution or a System Optimum solution. This is why the traffic assignment routine is important to the network design problem.

Our future work involves the development of a network generator for the network design problem and the traffic assignment routine. This network generator will enhance our efforts to provide more comprehensive results on the required execution times of the traffic assignment routine and the network design problem. Parallel to that, further attempts to improve the efficiency of the codes will be made using some of the previously mentioned measures, as well as moving towards more global restructuring of the codes to take advantage of the vector characteristics of CRAY.

References

Ardekani, S. A. & Herman, R. (1987), "Urban Network-Wide Variables and Their Relations," Transportation Science, Vol. 21, No. 1.

Beckman, M.J., McGuire, C. B. , and Winston, C. B. (1956). Studies in the Economics of Transportation. Yale University Press, New Haven, Connecticut.

Bruggeman, J.M., Leiberman, E., and Worrall, R.D. (1971), "Network Flow Simulation for Urban Traffic Control System", Federal Highway Administration, U.S. Department of Transportation.

Buzbee, B.L., and Sharp, D.H. (1985), "Perspectives on Supercomputing", Science, Vol. 227, pp. 591-597.

Center for High Performance Computing (UT CHPC) User Services (1987a), "CRAY FORTRAN Optimization and Performance Analysis", The University of Texas at Austin.

Center for High Performance Computing (UT CHPC) User Services (1987b), "Program Performance Analysis", The University of Texas at Austin.

Chen, S.S. (1983), "Large-Scale and High-Speed Multiprocessor System for Scientific Applications", High Speed Computation, NATO ASI Series F, Vol. 7, Springer-Verlag, Berlin.

Federal Highway Administration (January 1980), "Traffic Network Analysis with NETSIM - User's Guide," implementation package, FHWA-IP-80-3.

Frank, M., and P. Wolfe (1956), "An Algorithm for Quadratic Programming." ? Research Logistics Quarterly 3 (1-2), pp. 95-110.

Herman, R., and Prigogine, I. (1979), "A Two-Fluid Approach to Twon Traffic", Science, Vol. 204, pp. 148-151.

Mahmassani, H.S., Wiliams, J.C., and Herman, R. (1984), "Investigation of Network-Level Traffic Flow Relationships: Some Simulation Results", Transportation Research Record 971, Transportation Research Board, Washington, D.C, pp. 121-130.

Mahmassani, H.S. and Mouskos, K.C. (1988), "Some Numerical Results on the Diagonalization Network Assignment Algorithm with Assymmentric Interactions between Cars and Trucks", forthcoming in Transportation Research B, Vol. 22B.

Mahmassani, H.S., Wiliams, J.C., and Herman, R. (1987), "Performance of Urban Traffic Networks", in Gartner, N. and Wilson, N.H.M. (eds.), Transportation and Traffic Theory, Proceedings of the 10th International Symposium on Transportation and Traffic Theory, Elsevier Science Publishing, New York, N.Y.

Mouskos, K.C., H.S. Mahmassani, and C.M. Walton (1986), "Network Assignment Methods for the Analysis of Truck-Related Highway Improvements." Research Report 356-2F, Center for Transportation Research, The University of Texas at Austin, Texas.

San Diego Supercomputer Center. (June 1987). User Guide. Chapter 12: Optimizing Your FORTRAN Code.

Sheffi, Y. (1984). Urban Transportation Networks. Prentice-Hall, Englewood Cliffs, New Jersey, pp. 204-230.

U.S. Bureau of Public Roads (1964). Traffic Assignment Manual. U.S. Department of Commerce, Washington, D.C.

Wardrop, J.G. (1952). "Some Theoretical Aspects of Road Traffic Research." Proceedings, Institution of Civil Engineers II (1), pp. 325-378.

Williams, J.C. (1986), "Urban Traffic Network Performance - Flow Theory and Simulation Experiments," Ph.D. Dissertation, Department of Civil Engineering, University of Texas at Austin, Texas.

Williams, J.C., Mahmassani, H.S., and Herman, R. (1985), "Analysis of Traffic Network Flow Relations and Two-Fluid Model Parameter Sensitivity", Transportation Research Record 1005, pp. 95-106.

Williams, J.C., Mahmassani, H.S., and Herman, R. (1987), "Urban Traffic Network Flow Models", Transportation Research Record 1112, pp. 78-88.

Zenios, S.A., and Mulvey, J.M. (1986), "Nonlinear Network Programming on Vector Computers: A Study on the CRAY X-MP", Operations Research, Vol. 34, No. 5, (Sept.-Oct.), pp. 667-682.