A Real-Time Traffic Simulation System

Anthony Theodore Chronopoulos, *Senior Member, IEEE,* and Charles Michael Johnston

***Abstract*—This article studies the usefulness of parallel processing in real-time traffic-flow simulation based on continuum modeling of traffic dynamics. Computational fluid dynamics (CFD's) methods for solving simple macroscopic traffic-flow continuum models have been studied and efficiently implemented in traffic simulation codes (on serial computers) in the past. We designed a traffic-flow simulation code and mapped it onto a parallel computer architecture. This *traffic simulation system* is capable of simulating freeway traffic flow in *real time*. Tests with *real traffic data* collected from the freeway network in the metropolitan area of Minneapolis, MN, were used to validate the accuracy and computational rate of the parallel simulation system. The execution time for a 2-h traffic-flow simulation of about 200 619 vehicles in an 18-mi freeway, which takes 2.35 min of computer time (on a single-processor computer simulator), took only 5.25 s on the parallel traffic simulation system. This parallel system has a lot of potential for real-time traffic engineering applications.**

***Index Terms*— Freeway network, parallel, real time, traffic simulation.**

I. Introduction

RESEARCH in intelligent vehicle/highway systems (IVHS's) traffic has generated considerable thrusts in the last two decades. The most recent advances in vehicle controllers and highway management technology seem to indicate that it is possible to start implementing such systems in everyday traffic. The main potential advantages are:

1) significant increase in highway capacity and thus traffic volume served;
2) upgrading the highway safety;
3) decreasing the environmental harm due to vehicle pollution;
4) economic impact (savings in fuel, driving time, new technology generation, etc.).

IVHS consists of both intelligent vehicles and intelligent or automated highways. Intelligent vehicles would allow an increase in highway capacity of up to 300%. This would be the result of elimination of traffic congestion by increasing the speed and decreasing the intervehicle distance. The safety would result from (partial or entire) elimination of the human factor in vehicle control. It is assumed that the highway traffic management will be highly intelligent (entry/exit ramps, traffic signing, and incident detection).

A very important component of an intelligent highways' management system is **a traffic simulation system**. Such a system, consists of a traffic-flow simulation code, which is able to simulate traffic on a freeway and arterials network and a computer system. This computer system consists of hardware and software. Input/output devices provide data of real-time traffic measurements from a network of traffic detectors (loops or cameras) and data on the road geometries or other traffic characteristics. The system uses a mathematical traffic flow model to perform traffic-flow simulation and predict the traffic conditions in real time. These predictions can be used for real-time traffic control and drivers' guidance.

Accurate mathematical and computer modeling of the main characteristics of standard traffic-flow dynamics has been used in a better understanding of the collective behavior of the traffic, designing efficient control and management strategies, assessing the effects of roadway geometries, and in the design of new highway lanes. Traffic models can be characterized as either **microscopic** or **macroscopic**.

The microscopic models which have been extensive studied are the so-called **vehicle-follower** models, where the behavior of each vehicle is specified in terms of the vehicle immediately ahead. Such models have been used in simulating mostly single-lane traffic and automatic longitudinal/lateral vehicle control of individual vehicles (see [1], [6], [10], [12], [24], and [25]). These models (in their continuum formulation) involve **ordinary differential equations** describing the movement of each individual vehicle following the vehicle ahead.

Macroscopic or **continuum** traffic-flow models based on **traffic density, volume, and speed** have been proposed and analyzed in the past. See, for example, [14], [16]–[19], and [21]–[23]. These models involve **partial differential equations** (PDE's) defined on appropriate domains with suitable boundary conditions, which describe various traffic phenomena and road geometries.

The enhancement of computational efficiency in the continuum traffic models has been the focal point in the development of traffic simulation programs. One of the main goals of the traffic-flow simulation programs is to be used as a **computational component** in a **real-time traffic system**. Such a system would be fed with real-time traffic input data, and it would predict traffic conditions in real time. These predicted traffic conditions would be used for traffic control and drivers' guidance. In past research, traffic simulation systems have been designed with their computational component being a single-processor computer.

Junchaya and Chang (see [11]) demonstrated that real-time traffic simulation is feasible on parallel computers. They considered a traffic simulation model based on the macroparticle traffic simulation model (MTSM) (see [2]). This model uses macroscopic traffic relations to approximate the speed of a

Manuscript received June 23, 1995; revised October 6, 1996. This work was supported in part by the NSF under Grant CCR-9496327.

A. T. Chronopoulos is with the Department of Computer Science, Wayne State University, Detroit, MI 48202 USA (e-mail: chronos@cs.wayne.edu).

C. M. Johnston is with Concurrent Computer Corporation, Southfield, MI 48034 USA.

Publisher Item Identifier S 0018-9545(98)00098-X.

cluster of vehicles in a given link. Then, the vehicles are simulated individually. This means that this type of simulation allows car following and lane changing, as in microscopic simulations. Thus, this model combines macroscopic and microscopic model characteristics. This model was implemented on a parallel computer system. The simulation tests space domain was a computer-generated (hypothetical) grid network. No real traffic data were used.

The main objective of this paper is to demonstrate that the computational component in a real-time system is feasible for the **macroscopic models**. Some preliminary results on the issue of parallelizing computational fluid dynamic (CFD) methods for transportation problems were presented in [5]. Such a real-time simulation system can be designed using a parallel computer as its computational component. We design such a computational component by parallelizing a CFD method to solve the momentum conservation (macroscopic) model (see [9], [18], and [23]) and implementing it on the nCUBE2 parallel computer.

Tests with real data from the I-494 freeway in Minneapolis were conducted. Each processor of the nCUBE2 is as powerful as a SUN 3/50 workstation. We ran tests on the nCUBE2 at the Sandia National Laboratory. The execution time for a 2-h traffic-flow simulation of an 18-m freeway, which takes 2.35 min of computer time (on a single-processor computer simulator), took only 5.25 s on the parallel traffic simulation system.

The main differences between this work and that of Junchaya and Chang (see [11]) are: 1) our model is a pure macroscopic model based on the momentum equation, which implies that we solve a set of PDE's to obtain the traffic density, flow, and speed at every discrete time-space point; 2) our space domain is an actual freeway with multiple-entry/exit ramps; 3) our simulation tests use real traffic data as input data.

The structure of the article is as follows:

1) In Section II, a traffic-flow simulation model is described.
2) In Section III, the parallel implementation of the traffic model is discussed.
3) In Section IV, the test results are shown.
4) In Section V, conclusions are discussed.

II. A Traffic-Flow Model

In this section, we outline the traffic-flow model that we adopt for our research. This model is based on a continuum traffic-flow model, a discrete computer model and a freeway model (space domain).

A. A High-Order Continuum Model

Lighthill and Whitham (1995) [17] first proposed the following **simple continuum conservation equation model** (**LWM**) for the traffic-flow problem

$$\frac{\partial k}{\partial t}+\frac{\partial q}{\partial x}=g(x,t) \tag{1}$$

where $k(x,t)$ and $q(x,t)$ are the traffic density and flow, respectively, at the space-time point (x,t). The generation term $g(x,t)$ represents the number of cars entering or leaving the traffic flow in a freeway with entries/exits. The traffic-flow, density, and speed are related by

$$q=ku \tag{2}$$

where the equilibrium speed $u_e(x,t)=u(k)$ must be provided by a theoretical or empirical u-k model. The theoretical u-k model can take the general form

$$u_e=u_f[1-(k/k_{\text{jam}})^{\alpha}]^{\beta} \tag{3}$$

where u_f is the **free-flow speed** and k_{jam} the **jam density** model parameters. More information on this and other forms of the u-k relationships can be found elsewhere (see [4] and the references therein).

Since the simple continuum model does not consider acceleration and inertia effects, it does not describe accurately nonequilibrium traffic-flow dynamics. **High-order continuum traffic models** that include the momentum equation have also been developed. Such a model is the **semiviscous model** [23] (**SVM**). This mathematical model for the traffic flow is adopted for our research. This choice is not a restriction. The same parallel method can be applied to other continuum models (e.g., [17] and [22]).

SVM takes into account acceleration and inertia effects by replacing (3) with a momentum equation. For example, the equation in [23] has the following form:

$$\frac{du}{dt}=\frac{1}{T}[u_f(x)-u]-\nu k^{\beta}\frac{\partial k}{\partial x} \tag{4}$$

where $\frac{du}{dt}$ is the acceleration of an observer moving with the traffic stream and is related to the acceleration $\frac{\partial u}{\partial t}$ of the traffic stream as seen by an observer at a fixed point of the road, i.e.,

$$\frac{du}{dt}=\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}. \tag{5}$$

The *first term* on the right-hand side of (4) $\frac{1}{T}[u_f(x)-u]$ represents the relaxation term, the tendency of traffic flow to adjust speeds due to changes in free-flow speed $u_f(x)$ along the roadway, where relaxation time T is assumed to vary with density k according to

$$T=t_0\left(1+\frac{rk}{k_{\text{jam}}-rk}\right) \tag{6}$$

where $t_0>0$ and $0<r<1$ are model parameters. The *second term* on the right-hand side of (4) $\nu k^{\beta}\frac{\partial k}{\partial x}$ represents the anticipation term which is the effect of drivers reacting to downstream traffic conditions. In this term, ν is the anticipation parameter. As implied in this example, if downstream density is higher due to congestion, speed has to be decreased accordingly. Conversely, if downstream density is lower, speed can be increased. From (4)–(6), one derives a momentum model for the traffic flow described by the following system of PDE's:

$$\frac{\partial \vec{U}}{\partial t}+\frac{\partial \vec{E}}{\partial x}=\vec{Z} \tag{7}$$

where $\vec{U}$, $\vec{E}$, and $\vec{Z}$ are the following vectors:

$$\vec{U} = \begin{pmatrix} k \\ q \end{pmatrix}$$
$$\vec{E} = \begin{pmatrix} ku \\ u^2 k + \frac{\nu}{\beta+2} k^{\beta+2} \end{pmatrix}$$
$$\vec{Z} = \begin{pmatrix} g \\ \frac{k}{T}[u_f(x) - u] + gu \end{pmatrix}.$$

A typical choice of parameters is $u_f = 60$, $k_{\text{jam}} = 180$, $\beta = -1$, $\nu = 180$, $t_0 = 50$, and $r = 0.80$. These parameters depend on the geometry of the freeway, but also on the time of the day and even on the weather conditions.

We note that the momentum conservation model does not require a $q-k$ curve as in the case of the simple continuum model. However, speed data are required for the boundary conditions. If speed data are not available from the real traffic data measurements, then a $q-k$ curve based on **occupancy** data ([23]) is used to generate the speed data. This is discussed in the next section.

B. Volume-Density Relation

A $q-k$ model curve is an indispensable part of the LWM. This relation can be used to express the volume as a function of the flow density, i.e., $q = q(k)$ (see [4]). The equations that define the $q-k$ curve are used in the programs to convert density to volume and to convert volume to density. SVM **does not require** a $q-k$ curve. However, if the **speed data** are not available from the **measured traffic data**, then a $q-k$ curve is used to compute the traffic speed.

The method that we adopt for estimating traffic density and speed is based on **occupancy** data [7].

Lane occupancy (ϕ) is defined as *the time that loop detector is ''turned on''* divided by *the measured time* and multiplied by 100. The value of lane occupancy is available from the loop detector installed on the freeway pavement. If we assume that the speed of the vehicle is constant during measurement time and each vehicle's length is the same, we can derive the relationship between density and occupancy as follows:

$$k = 52.8 \times \frac{\phi}{L_e}$$

where N_v = number of vehicles that crossed the loop, the effective length $L_e = \frac{\sum_{l=1}^{N_v}(L_l + L_d)}{N_v}$, L_l = vehicle length, and L_d = intervehicle distance. (See Fig. 1.)

C. A Discrete Model

We now apply the computer method (Lax) to discretize SVM. This discrete model will be called **Lax Momentum** model. For each traffic model, the road section (the space dimension) is discretized using a uniform mesh. Let Δt and Δx be the time and space mesh sizes. We use the following notation:

- k_j^i density (vehicles/mi/lane) at space node $j\Delta x$ and at time $i\Delta t$;
- q_j^i flow (vehicles/h/lane) at space node $j\Delta x$ and at time $i\Delta t$;

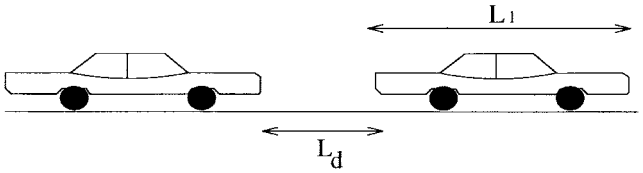


Fig. 1. **Density/occupancy** calculation from vehicle relative positions.

- u_j^i speed (mi/h) at space node $j\Delta x$ and at time $i\Delta t$.

At time $(i+1)\Delta t$, the density value k_j^{i+1} and volume value q_j^{i+1} are computed directly from the density and volume at the preceding time step i

$$\vec{U}_j^{i+1} = \frac{\vec{U}_{j+1}^i + \vec{U}_{j-1}^i}{2} - \frac{\Delta t}{\Delta x}\frac{\vec{E}_{j+1}^i - \vec{E}_{j-1}^i}{2} + \frac{\Delta t}{2}\left(\vec{Z}_{j+1}^i + \vec{Z}_{j-1}^i\right). \quad (8)$$

The method is of first-order accuracy with respect to Δt, i.e., the error is $O(\Delta t)$. To maintain numerical stability time and space step sizes must satisfy the Courant–Friedrichs–Lewy (CFL) condition $\frac{\Delta x}{\Delta t} > u_f$, where u_f is the free-flow speed in order to maintain numerical stability in the computation (see [9]). Typically, the space and time meshes $\Delta x = 200$ ft, and $\Delta t = 1$ s are recommended for **numerical stability** [23].

D. A Freeway Model

We have used two schemes to add/subtract entry/exit (ramp) traffic volumes to the main-lane traffic volume in SVM.

1) Point entry/exit scheme: ramp volumes are assumed to merge into (diverge from or exit from) the freeway main lane at a single space node. This treatment is necessary to simplify the modeling and reduce computation time at such main-lane nodes.
2) Weaving entry/exit scheme: this is used when the ramp is directly connected to another freeway, and it is explained in more detail below.

The weaving scheme is outlined as follows. In the following discussion, let us consider the traffic-flow volume in a freeway section shown in Fig. 2 at a fixed discrete time. In Fig. 2, volume v_1 represents the through-traffic volume flow from link A to link B, volume v_2 represents the diverging volume from link A to link F, and $q_A = v_1 + v_2$. v_3 is the merging volume from link E to link B, volume v_4 is the through-volume from link E to link F, and $q_E = v_3 + v_4$. It is obvious that $q_F = v_2 + v_4$ and $q_B = v_1 + v_3$. Because there are interchanges of v_2 and v_3, traffic friction at link B and link E, in this case, is greater than the case of a single entrance ramp or exit ramp. Thus, this must be taken into account by calibrating (locally) the u_f parameter in the mathematical model for these space nodes. Also, only merging dynamics at an entrance ramp must be employed, if $v_2 = 0$. Similarly, only diverging dynamics must be employed, if $q_E = 0$.

When the distance between links E and F is less than 600 ft, merging and diverging movements must be completed within a short distance. However, since both q_E and q_F require lane changing in the same limited length of roadway at the same time, the sum of q_E and q_F must be included in the generation

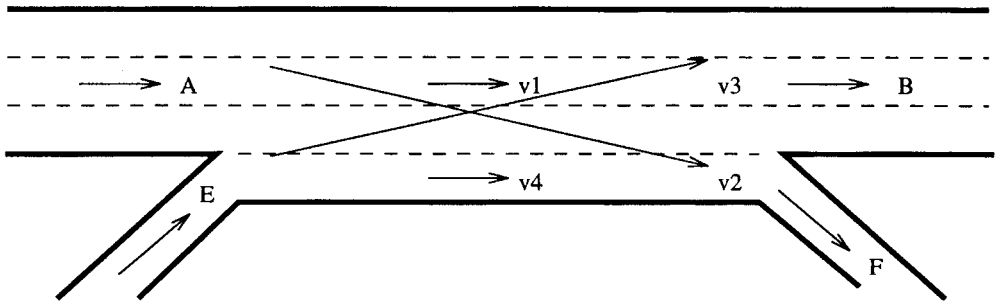


Fig. 2. Weaving flows in a freeway.

term of the model. If the generation term $g > 0$, the short weaving section is treated as a single on ramp. If the generation term $g < 0$, it is treated as a single off ramp. The generation term then becomes

$$g = (q_E - q_F)/\Delta x.$$

III. A Real-Time Traffic Simulation System

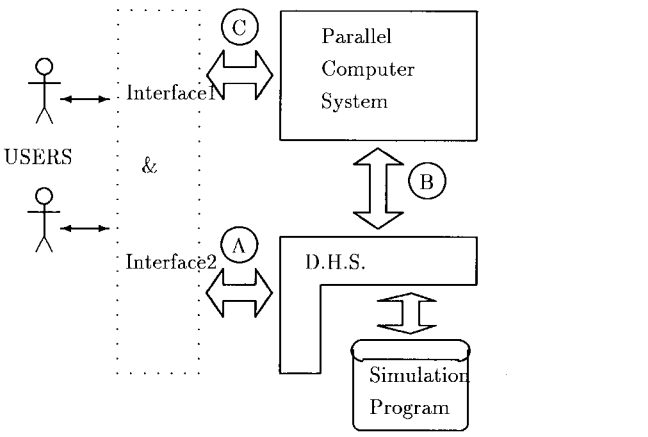
In this section, we outline the basic features of the real-time traffic simulation system. We show that such a system is feasible as far as its computational component is concerned.

A. The Real-Time Simulation System Architecture

Fig. 3 shows the architecture diagram of a real-time simulation system. Such a system would consist of a parallel computer system, data-handling system (DHS), simulation program and two interface devices. The DHS is the software which handles the data structures for storing the measured traffic data and the road characteristics data. DHS could be built around a database system. The simulation program is the software which implements the traffic-flow model on the parallel computer. The interfaces devices consist of networks which channel data to/from the parallel simulation system. Traffic measured data are channeled to the parallel simulation system from the data collection stations. The simulation output data are channeled to the traffic control and drivers' guidance devices.

We call a **parallel traffic simulation system** a traffic-flow model, and the parallel computer architecture on which the traffic-flow model is implemented. Our contribution in this paper is the design of the parallel traffic simulation system. The parallel computer system adopted here is nCUBE2. The following terminology is introduced in order to evaluate the parallel simulation system.

1) **Number of processors** (p) is the number of processors in the parallel computer that are assigned to solve simultaneously the traffic-flow problem.
2) **Serial execution time** (T_1) is the time required to solve the problem on a single-processor computer system.
3) **Parallel execution time** (T_p) is the time required to solve the problem on p processors. For a fixed parallel computer system, T_p is a function of T_1 and of the number of processors available.
4) **Parallel speedup** (S_p) is the ratio $\frac{T_1}{T_p}$.
5) **Efficiency** (E_p) is the ratio $\frac{S_p}{p}$.



Legend:

(A) Interface2: Interface of D.H.S. (Data Handling System) with Traffic Data Collection Devices Network.

(B) Transmittal of Traffic Measurements Data and Road Geometries Data.

(C) Interface1: Interface of Parallel Computer System with Real-Time Guidance and Control System.

Fig. 3. Real-time traffic simulation system architecture.

B. The nCUBE2 Parallel Computer

The nCUBE2 is a multiple-instruction multiple-data (MIMD) **hypercube** parallel computer system. A hypercube model is an example of a distributed-memory message passing parallel computer. Let $ndim$ be a positive integer. In a hypercube of dimension $ndim$, there are $p = 2^{ndim}$ processors. These processors are labeled by $0, 1, \cdots, p-1$. Two processors P_{j_1} and P_{j_2} are directly connected if the binary representation of j_1 and j_2 differ in exactly 1 b. Each edge of the hypercube graph represents a direct connection between two processors. In a hypercube of dimension $ndim$, each processor is connected to $ndim$ other processors. Thus, any two processors in a hypercube graph are connected by a maximum distance of $ndim$ edges. Fig. 4 shows a hypercube graph of dimension $ndim = 4$. The number of processors to be active is chosen by the user, but must be a power of two.

In Table I, we show a summary of interprocessor communication times for neighbor processors and the basic floating

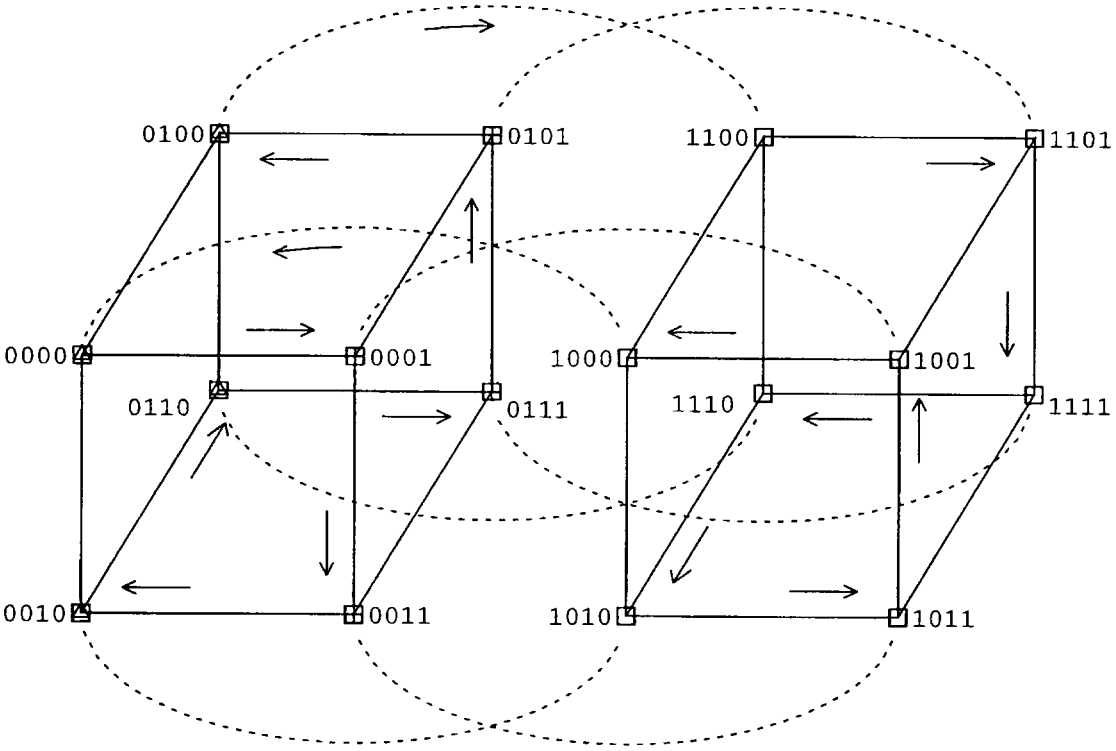


Fig. 4. Hypercube (of $ndim = 4$) with Gray code mapping of linear arrays in its subcubes (of $ndim = 3$).

TABLE I
COMPUTATION AND COMMUNICATION TIMES ON THE mCUBE2

Operation	Time	Comm/Comp
8 Byte transfer	111 μ sec	-
8 Byte Add	1.23 μsec	90
8 Byte Multiply	1.28 μsec	86

point operation times for nCUBE2 [13]. We see that communication even between neighbor processors is several times slower than floating point operations.

In a hypercube with a high communication latency, the algorithm designer must structure the algorithm so that large amounts of computation are performed between the communication steps.

The two important factors which influence the delivered performance of this machine are load-balancing and reduction of the communication overhead. A program is load balanced if all the processors are kept busy. An efficient algorithm is one for which both computations and data are distributed among the processors so that the computations run in parallel, balancing the computational loads of the processors as well as possible.

C. Parallelization of Lax Momentum

Let p be the number of processors available in the system. The parallelization of the discrete model is obtained by partitioning the space domain (freeway model) into equal **segments** $\mathbf{Seg}_0, \cdots, \mathbf{Seg}_{p-1}$ and assigning each segment to the processors $P_{j_0}, \cdots, P_{j_{p-1}}$. The choice of indexes $j_0, \cdots, j_{p-1}$ defines a mapping of the segments to the processors.

The computations associated with each segment have as their goal to compute the density, volume, and speed over that segment. The computation in the time dimension is not parallelized. At a fixed discrete time, this essentially means that the quantities k_j^i, q_j^i, and u_j^i are computed by processor P_{j_k} if the space node $j\Delta x$ belongs to the segment Seg_{j_k}. This segment-processor mapping must be such that the communication delays for data exchanges, required in the computation, are minimized. Such a mapping of a linear array (of sets) onto a hypercube is achieved by the **Gray code mapping** (see [15]). To explain this mapping, we consider $p = 16$ and a freeway section with only 16 space nodes. We map one space node to each processor. We know that the space nodes form a **linear array** in terms of their location. The discrete model equation (at a fixed discrete time i) requires that the density/speed/volume of nodes $j-1$ and $j+1$ are sent to the processor handling space node j in order to compute density/speed/volume (at discrete time $i+1$). However, the processors are not interconnected in a linear array topology. Thus, we must map a linear array of nodes onto the hypercube. The same would be true if instead of 16 single space node we considered a freeway section divided into 16 segments forming a linear array. Then, the density/speed/volume at the boundaries of these segments must be sent between processors handling adjacent segments.

The Gray code mapping is easier to explain by an example. Here, let the ordering of the nodes, onto which the indexes $0, 1, 2, 3, \cdots$ are mapped, be from left to right, and let the node numbers be in binary. For a hypercube of dimension $ndim$, the nodes are $ndim$-bit numbers. For the 1-b case 0, 1 are mapped to 0, 1. To get the 2-b case, we reflect the 1-b case numbers

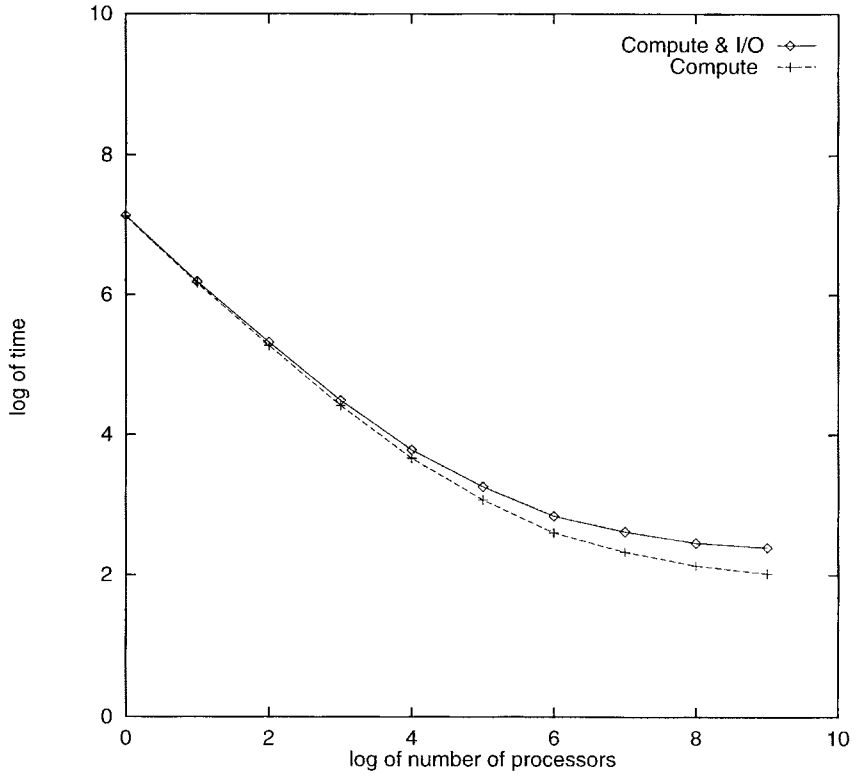


Fig. 5. Parallel execution time (s) on nCUBE2. $dt = 1$ s and $dx = 200$ ft.

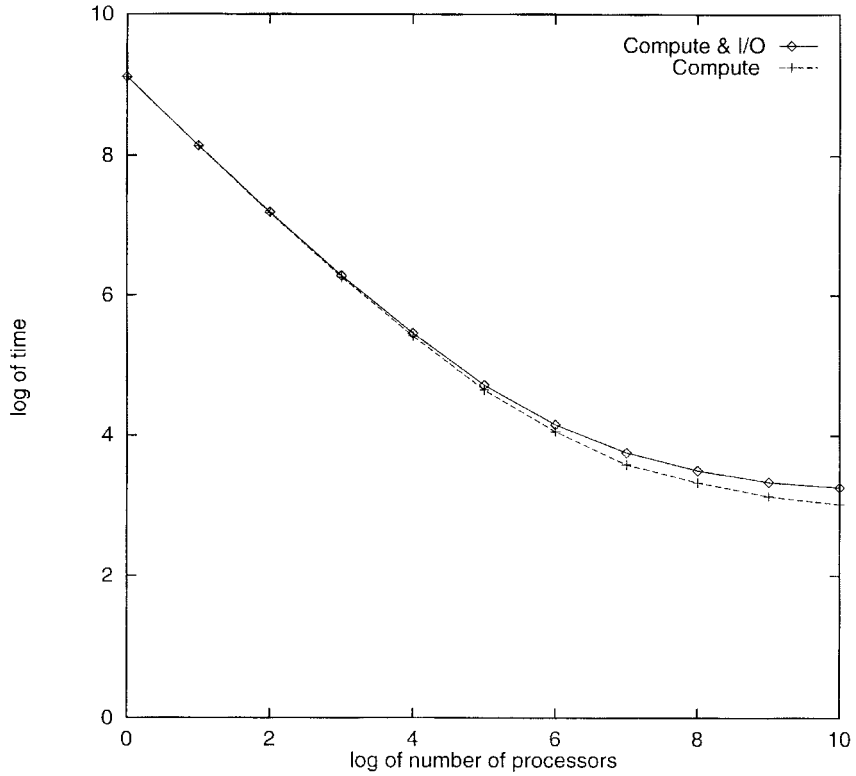


Fig. 6. Parallel execution time (s) on nCUBE2. $dt = 0.5$ s and $dx = 100$ ft.

around the right-most number (1) to get: 0, 1 | 1, 0, where | is used to separate the reflected numbers from the preexisting ones. We add a prefix of 0 (1) to the numbers on the left (right) (of |) and remove | to get 00, 01, 11, 10. Inductively, we obtain the same way the $(ndim+1)$-bit Gray code numbers from the $ndim$-bit code numbers. Fig. 4, illustrates the Gray

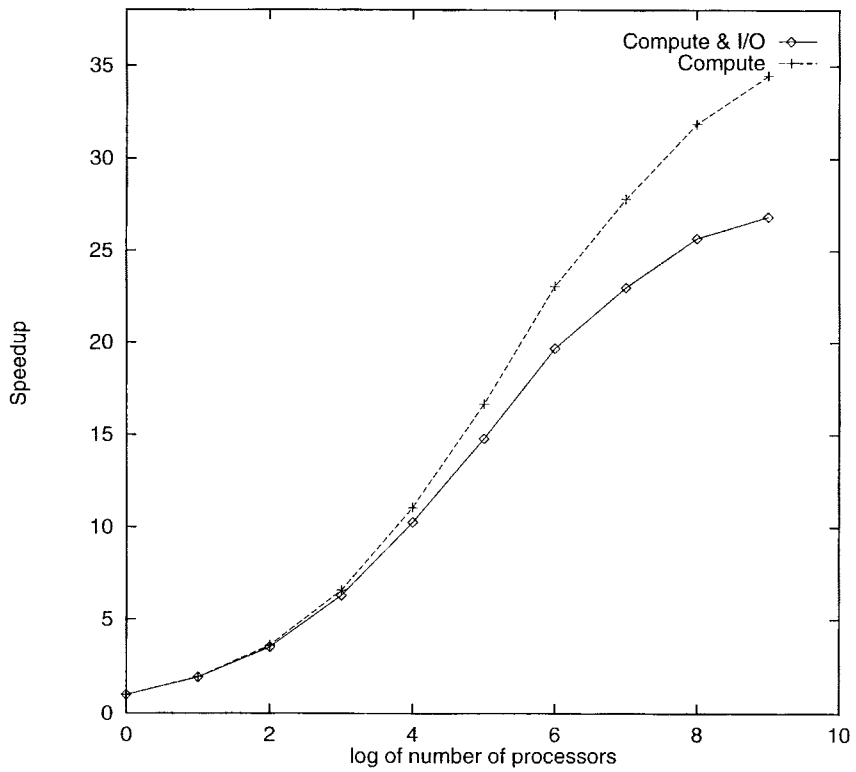


Fig. 7. Speedup on nCUBE2. $dt = 1$ s and $dx = 200$ ft.

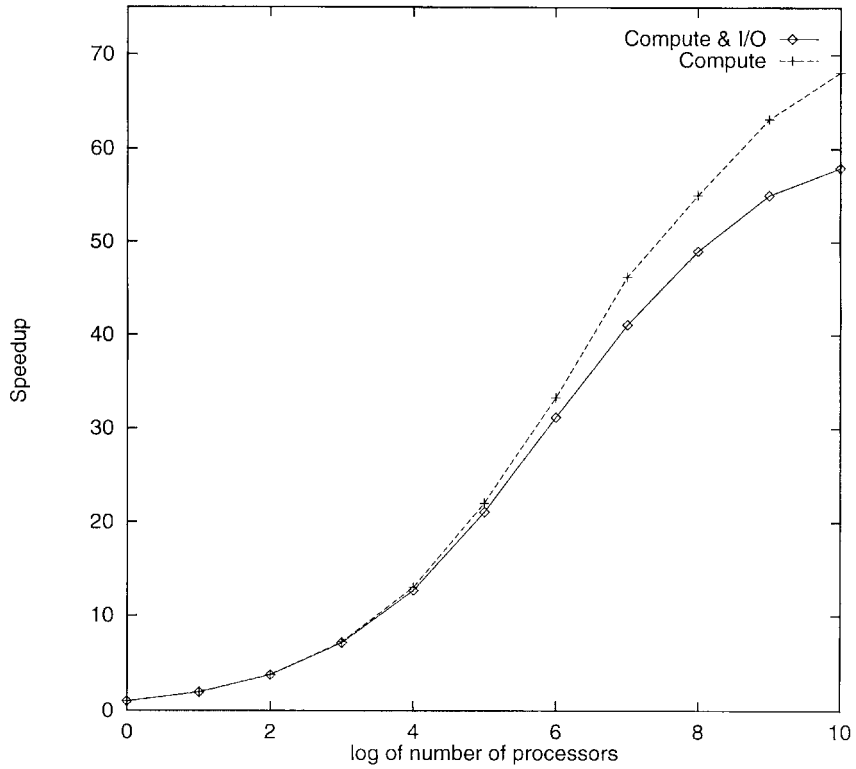


Fig. 8. Speedup on nCUBE2. $dt = 0.5$ s and $dx = 100$ ft.

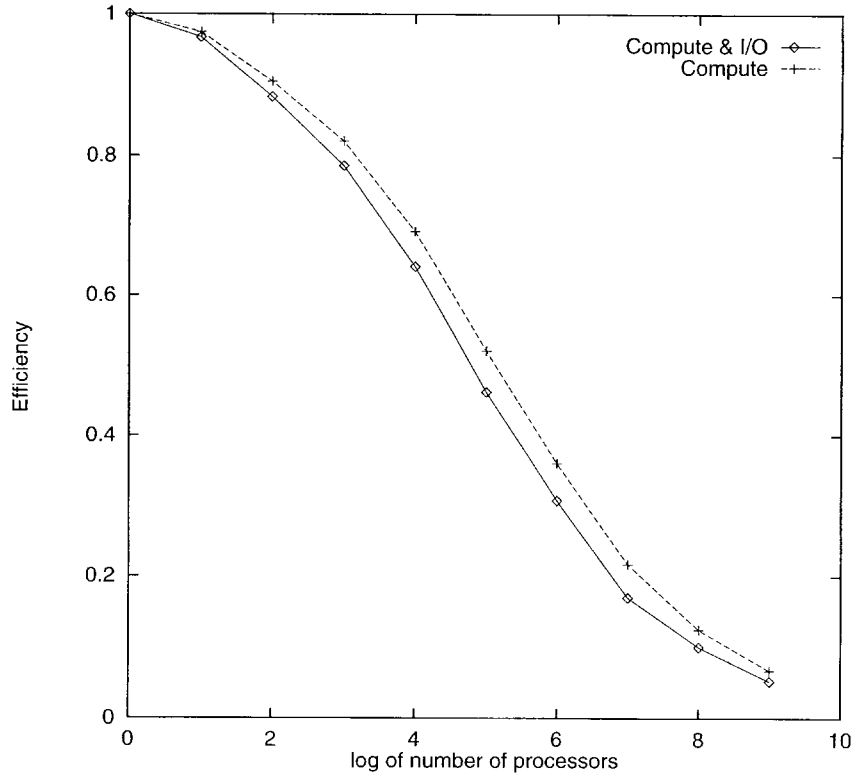


Fig. 9. Efficiency on nCUBE2. $dt = 1$ s and $dx = 200$ ft.

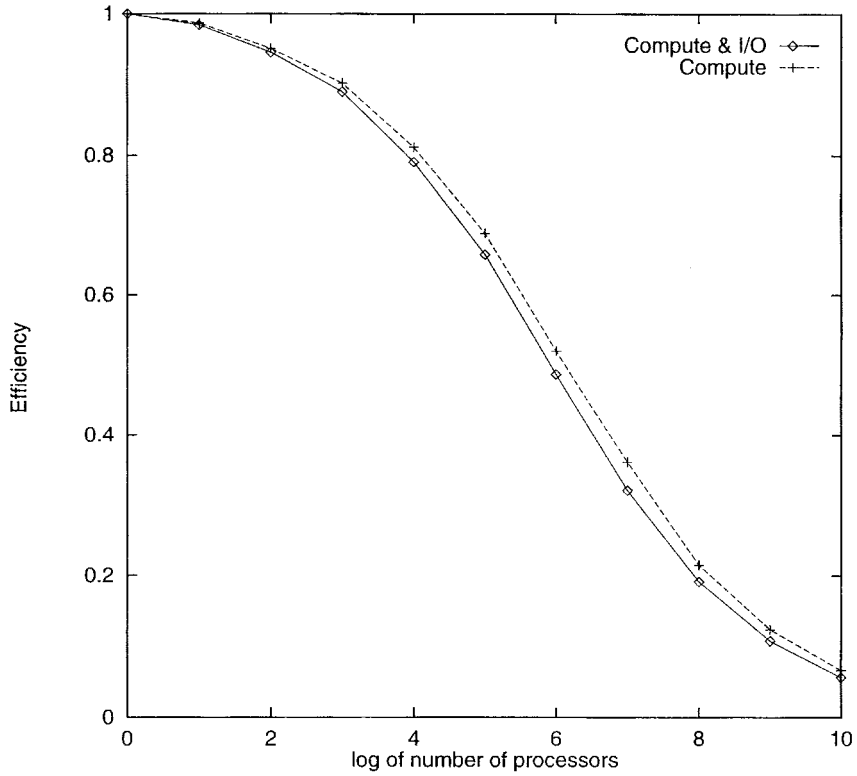


Fig. 10. Efficiency on nCUBE2. $dt = 0.5$ s and $dx = 100$ ft.

code mapping of a linear array of 16 nodes onto the hypercube of $ndim = 4$. The nodes' sequence is marked by arrows. In the scheduling part of the parallelization, this sequence is generated (for each $ndim$ dimension hypercube) only once, and it can be stored and reused.

The parallelization of Lax Momentum with a space domain a **network of freeways** can be achieved in a similar way, by dealing with each member freeway one at time. Each member freeway is mapped onto a subcube of the hypercube. The issue of mapping automatically arbitrary networks onto a

TABLE II
ERROR STATISTICS FOR TRAFFIC FLOW VOLUME (I-494)

Point Entry/Exit scheme.
Lax $dt = 1sec$, Volume error $(veh/5min)$

Sites	*Maximum*	*Max. Rel.*	*2 - Norm*	*Average*	*Average Rel.*	Std. Dev.
1	56.4	0.24	0.12	29.3	.12	31.3
2	60.4	0.19	0.11	31.7	.10	34.7
3	73.4	0.18	0.10	27.4	.09	32.6
4	40.8	0.12	0.07	15.5	.05	20.1
5	92.6	0.23	0.13	34.6	.11	42.1

TABLE III
ERROR STATISTICS FOR TRAFFIC FLOW SPEED (I-494)

Point Entry/Exit scheme.
Lax $dt = 1sec$, Speed error $(mile/hour)$

Sites	*Maximum*	*Max. Rel.*	*2 - Norm*	*Average*	*Average Rel.*	Std. Dev.
1	2.9	0.07	0.04	1.7	.03	1.8
2	3.9	0.09	0.04	1.9	.04	2.1
3	4.3	0.10	0.04	1.6	.04	2.0
4	3.9	0.09	0.03	1.2	.03	1.6
5	8.2	0.22	0.06	2.2	.05	3.0

parallel computer system (like an nCUBE2) is an active area of research.

IV. SYSTEM TESTING

We have implemented our simulation code on the (1024-processor) nCUBE2 parallel computer located at the Sandia National Laboratory in Alburquerque, NM. As our test site, we considered a multiple-entry/exit freeway section in the Minneapolis freeway network. This is a section of Eastbound I-494. The Eastbound I-494 section extends from the Carlson Parkway to Portland Avenue. It is 18 mi long, and it has 21 entry and 18 exit ramps. To test the program, the time and space mesh sizes were $\Delta t = 1$ s and $\Delta x = 200$ ft. The discrete model contains 425 space nodes. In order to be able to test the simulation code on the full configuration, we also run a test with $\Delta t = 0.5$ s and $\Delta x = 100$ ft. Then, the number of (space nodes)/(discrete times) doubles. From these results one could extrapolate the performance of the parallel system for a (36 mi) freeway traffic simulated for 4 h.

Our tests are distinguished into two units: comparisons with real data and performance analysis.

A. Comparisons with Real Data

Traffic data are collected at the **upstream/downstream boundaries** of the freeway section and at **check-station sites** inside the freeway section. Let N be the number of discrete time points at which real traffic-flow data are collected. We compare our simulation computed traffic-flow volume and speed data with the check-station sites' data. We use the following error moduli to measure the effectiveness of the simulation in comparison with actual data

$$\text{Maximum Absolute Error} = \text{Max}_{1\le j\le N}|\text{Observed}_j - \text{Simulated}_j| \tag{9}$$

$$\text{Maximum Relative Absolute Error} = \text{Max}_{1\le i\le N}\frac{|\text{Observed}_j - \text{Simulated}_j|}{\text{Observed}_j} \tag{10}$$

$$\text{Mean Absolute Error} = \frac{1}{N}\sum_{j=1}^{N}|\text{Observed}_j - \text{Simulated}_j| \tag{11}$$

$$\text{Mean Relative Error} = \frac{1}{N}\sum_{j=1}^{N}\frac{|\text{Observed}_j - \text{Simulated}_j|}{\text{Observed}_j} \tag{12}$$

$$\text{Relative Error with } 2-\text{Norm} = \left[\frac{\sum_{j=1}^{N}(\text{Observed}_j - \text{Simulated}_j)^2}{\sum_{j=1}^{N}\text{Observed}_j^2}\right]^{1/2} \tag{13}$$

$$\text{Standard Deviation} = \left[\frac{1}{(N-1)}\sum_{j=1}^{N}(\text{Observed}_j - \text{Simulated}_j)^2\right]^{1/2}. \tag{14}$$

The error statistics are summarized in Tables II and III. The relative errors are at a level about 10% for the volume, but are lower for the speed measurements. These error sizes are consistent with past simulations carried out by simulation systems based on a single-processor computer (see [3]).

B. Performance Analysis

We measure **the execution time**, which consists of the **time for input/output data** and **the computation time** as follows.

1) Time for input data: time for reading from the computer disk of the initial data by processor 0 and then broadcasting them to all the processors. Initial data are: 1)

TABLE IV
PARALLEL EXECUTION TIMES (IN SECONDS)

Procs= No. of Processors
etime/itime/ctime/otime = execution/(input data)/compute/(output data) time
$\Delta t = 1$ sec, $\Delta x = 200$ feet

ndim	*Used Procs*	*Max. Procs*	*etime*	*itime*	*ctime*	*otime*
9	425	512	5.250	0.80	4.06	0.44
8	213	256	5.493	0.73	4.39	0.40
7	107	128	6.133	0.73	5.03	0.39
6	61	64	7.166	0.72	6.07	0.38
5	31	32	9.539	0.71	8.40	0.44
4	16	16	13.745	0.71	12.65	0.40
3	8	8	22.438	0.70	21.32	0.41
2	4	4	39.900	0.72	38.66	0.52
1	2	2	72.906	0.70	71.81	0.40
0	1	1	141.012	0.67	139.94	0.40

TABLE V
PARALLEL EXECUTION TIMES (IN SECONDS)

Procs= No. of Processors
etime/itime/ctime/otime = execution/(input data)/compute/(output data) time
$\Delta t = 0.5$ sec, $\Delta x = 100$ feet

ndim	*Used Procs*	*Max. Procs*	*etime*	*itime*	*ctime*	*otime*
10	851	1024	9.563	0.90	8.12	0.63
9	426	512	10.070	0.80	8.76	0.56
8	213	256	11.312	0.79	10.05	0.50
7	122	128	13.497	0.80	11.97	0.74
6	61	64	17.790	0.76	16.63	0.41
5	32	32	26.319	0.76	25.11	0.46
4	16	16	43.829	0.75	42.61	0.47
3	8	8	77.829	0.76	76.66	0.42
2	4	4	146.508	0.74	145.37	0.40
1	2	2	281.571	0.81	280.33	0.43
0	1	1	554.444	0.71	553.33	0.40

volume/speed at entry/exit, upstream/downstream, and check-station sites and 2) an array of flags.

2) Time for computation: time for discrete model computations.
3) Time for output data: time for gathering output data from all processors to processor 0 and then processor 0 writing to the disk. Output data are: 1) volume/speed at every sixth space node (i.e., about every 0.25 mi) at every minute and 2) an array of errors.

In a real-time simulation system, the output data are channeled to the traffic control and drivers' guidance devices.

The parallel execution times are tabulated in Tables IV and V. We note that for the larger configurations about 16% of the processors are idle because our task scheduling strategy divides the freeway into **equal** segments and maps onto the processors. We are currently working to improve this task scheduling strategy.

The parallel execution time is plotted in Figs. 5 and 6. This execution time is sufficiently fast to justify the usefulness of the proposed parallel system as part of a real-time traffic system. The parallel speedup and efficiency curves are given in Figs. 7–10. It can be seen that a parallel speedup of 27 (57) is achieved for $\Delta t = 1$ s (0.5 s) and $\Delta x = 200$ ft (100 ft), on the largest system configurations. The efficiency of the parallel processing drops as the number of processors increase. This is unavoidable because the size of the problem is small compared to the number of processors used. If we increase the size of the problem, then the efficiency would increase.

V. CONCLUSION

A very important component of an intelligent highways' management system is a traffic simulation system. The design of a real-time traffic simulation system is a challenging problem. We demonstrate the design of a parallel (macroscopic) traffic simulation system. This system can be used as a component of a real-time simulation system. The parallel simulation system consists of a parallel computer and a traffic simulation program. We implemented this parallel system on the nCUBE2 parallel computer. A single processor of nCUBE2 is as powerful as a workstation processor. We ran tests with real traffic data to validate the accuracy and computational rate of the system. The computer time for a

2-h traffic-flow simulation of an 18-mi freeway, with real input data, takes 5.05 s versus 2.35 min on a single-processor system time. This demonstrates that there is a great potential of using parallel processors in the design and implementation of real-time traffic systems.

ACKNOWLEDGMENT

The comments of the anonymous reviewers, which helped improve the presentation of some parts of the paper, are gratefully acknowledged.

REFERENCES

[1] J. G. Bender, "An overview of systems studies of automated highway systems," *IEEE Trans. Veh. Technol.*, vol. 40, no. 1, pp. 82–99, 1991.
[2] G. L. Chang, H. S. Mahmassani, and R. Herman, "A macropartite traffic simulation model to investigate peak-period commuter decision dynamics," *Transport. Res. Rec.*, vol. 1005, 1985.
[3] A. T. Chronopoulos *et al.*, "Traffic flow simulation through high order traffic modeling," *Math. Comput. Modeling*, vol. 17, no. 8, pp. 11–22, 1993.
[4] A. T. Chronopoulos *et al.*, "Efficient traffic flow simulation computations," *Math. Comput. Modeling*, vol. 16, no. 5, pp. 107–120, 1992.
[5] A. Chronopoulos and G. Wang, "Traffic flow simulation through parallel processing," *Parallel Comput.*, vol. 22, pp. 1965–1983, 1997.
[6] R. E. Fenton and R. J. Mayan, "Automated highway studies at the Ohio State University—An overview," *IEEE Trans. Veh. Technol.*, vol. 40, no. 1, pp. 100–113, 1991.
[7] D. L. Gerlough and M. J. Huber, "Traffic flow theory," Transport. Res. Bd. Special Rep. 165, National Res. Council, Washington, DC, 1975.
[8] J. Gustafson, G. Montry, and R. Benner, "Development of parallel methods for a 1024-processor hypercube," *SIAM J. Sci. Stat. Comput.*, vol. 9, pp. 609–638, 1988.
[9] C. Hirsch, *Numerical Computation of Internal and External Flows*. New York: Wiley, 1988, vol. 2.
[10] P. Ioannou, C. C. Chien, and J. Hauser, "Autonomous intelligent cruise control," in *IVHS America 1992 Annu. Meet.*, pp. 97–112.
[11] T. Junchaya and G. Chang, "Exploring real-time traffic simulation with massively parallel computing architecture," *Transport. Res. C*, vol. 1, no. 1, pp. 57–76, 1993.
[12] J. L. Kim *et al.*, "The areawide real-time traffic control (ARTC) system: A new traffic control concept," *IEEE Trans. Veh. Technol.*, vol. 42, no. 2, pp. 212–223, 1993.
[13] S. K. Kim and A. T. Chronopoulos, "A class of Lanczos-like algorithms implemented on parallel computers," *Parallel Comput.*, vol. 17, pp. 763–778, 1991.
[14] R. D. Kühne, "Microscopic distance strategies and macroscopic traffic flow model," in *Proc. Int. Conf. Control, Computers and Communication in Transportation*, Paris, France, Sept. 19–21, 1989.
[15] V. Kumar *et al.*, *Introduction to Parallel Computing Design and Analysis of Algorithms*. Redwood City, CA: Benjamin/Cummings, 1994.
[16] C. J. Leo and R. L. Pretty, "Numerical Simulation of macroscopic continuum traffic models," *Transport. Res.*, vol. 26B, no. 3, pp. 207–220, 1990.
[17] M. H. Lighthill and G. B. Witham, "On kinematic waves: II a theory of traffic flow on long crowded roads," *Proc. R. Soc. Ser.*, vol. A 229, no. 1178, pp. 317–345, 1955.
[18] A. S. Lyrintzis *et al.*, "Continuum Modeling of traffic dynamics," in *Proc. 2nd Int. Conf. Appl. Advanced Tech. Transportation Eng.*, Minneapolis, MN, Aug. 18–21, 1991, pp. 36–40.
[19] L. Mikhailov and R. Hanus, "Hierarchical control of congested urban traffic-mathematical modeling and simulation," *(IMACS) Math. Comput. Simulation*, vol. 37, pp. 183–188, 1994.
[20] *nCUBE2 Programmers Guide 1992*, nCUBE, Foster City, CA.
[21] M. Papageorgiou and J. C. M. Banos, "Optimal control of multidestination traffic networks," in *Proc. 29th IEEE CDC*, Honolulu, HI, Dec. 1990, pp. 1355–1361.
[22] H. J. Payne, "FREEFLO: A macroscopic simulation model of freeway traffic," *Tranport. Res. Rec.*, vol. 772, pp. 68–75, 1979.
[23] P. Yi *et al.*, "Development of an improved high order continuum traffic flow model," *Transport. Res. Rec.*, vol. 1365, pp. 125–132, 1993.
[24] S. Sheikholeslam and C. A. Desoer, "A system level study of the longitudinal control of a platoon of vehicles," *Trans. AMSE, J. Dynamic Syst., Meas. Contr.*, vol. 114, pp. 286–292, 1992.
[25] S. J. Sklar, J. P. Bevans, and Stein, "Safe-approach vehicle-following control," *IEEE Trans. Veh. Technol.*, vol. VT-28, no. 1, pp. 56–62, 1979.

Anthony Theodore Chronopoulos (SM'98) was born on October 18, 1956. He received the B.Sci. degree in mathematics from the University of Athens, Athens, Greece, in 1979 and the Ph.D. degree in computer science at the University of Illinois, Urbana-Champaign, in 1987.

He is an Associate Professor at the Department of Computer Science, Wayne State University, Detroit, MI. He has published over 25 refereed journal publications in the areas of computational science, parallel and distributed computing and applications in traffic-flow simulation, and helicopter aerodynamics simulation.

Charles Michael Johnston received the B.S. degree in mathematics from the University of Michigan, Ann Arbor, in 1980 and the M.S. degree in computer science from Wayne State University, Detroit, MI, in 1997.

Currently, he is a Regional Specialist/Analyst for Concurrent Computer Corporation, Southfield, MI.