Proceedings of the 1993 Winter Simulation Conference
G.W. Evans, M. Mollaghasemi, E.C. Russell, W.E. Biles (eds.)

DISTRIBUTED/PARALLEL TRAFFIC SIMULATION FOR IVHS APPLICATIONS

Paul T. R. Wang
William P. Niedringhaus

The MITRE Corporation
McLean, VA

ABSTRACT

This paper presents two algorithms developed for a distributed, discrete-event, and object-oriented traffic simulation, such as the Traffic and Highway Objects for REsearch, Analysis, and Understanding (THOREAU) (McGurrin and Wang, 1991) and (Hsin and Wang, 1992). THOREAU was designed for the study and analysis of Intelligent Vehicle Highway Systems (IVHS) [1] applications. The purpose of using distributed processing for traffic simulation is to extend the scope which can be modeled at an individual vehicle behavior level, by significantly increasing execution speed. The first algorithm was derived to decompose a large traffic model into submodels distributed over a network of workstations, with a minimum amount of inter-processor interactions, and to achieve the highest degree of parallelism. The second algorithm is an improvement of the Floyd algorithm for finding shortest paths using submodel decomposition and node to arc incidency to achieve a $10m^3-$ fold speed improvement using m distributed processors. Both algorithms are being implemented for IVHS-related applications in a new version of THOREAU.

1 INTRODUCTION

IVHS is a multimillion dollar program, jointly undertaken by government and private sector, to improve roadway mobility, driving quality, and safety using advanced computer, communication, and control technologies. Two major parts of IVHS are Advanced Traffic Management Systems (ATMS) and Advance traveler Information Systems (ATIS) (Federal Highway Administration, 1991). Object-oriented traffic simulations serve as very powerful tools in determining cost/benefit tradeoffs and in analysis of implementation alternatives under various IVHS technologies. Object-oriented discrete-event simulation is desirable because of its modularity, portability, and flexibility. Microscopic (individual vehicle behavior) trip-specific traffic simulations are preferable because of the dynamics of interactions between vehicles, roadway, and control elements such as signal controllers, detectors, and a traffic management center. Large scale trip simulation is essential for IVHS cost/benefits analysis. Only by large scale traffic simulation, can the cumulative impacts of ATIS and ATMS technologies over a great number of trips covering an entire metropolitan area or networks of surface streets and freeways be determined. To date, more abstract macroscopic and mesoscopic flow models must supplement microscopic models due to the long run-time associated with large area microscopic models.

It is quite obvious that simulation time for object-oriented microscopic event simulation tends to increase almost exponentially with respect to model size since the number of objects and potential interactions among all objects increases rapidly as model size increases. Therefore, to achieve a reasonable speed in simulation, decomposition of a large model into submodels of equal size to be distributed evenly among a number of processing elements will be necessary. Moreover, certain key algorithms that require access to objects in each submodel may have to be processed in parallel among submodels with results combined later to obtain the final result for global use. One such example is the computation of shortest paths for every node (or link) pair to be used by ATIS for route guidance simulations. Thus, it is quite important to derive optimal or near optimal model decomposition techniques and parallel algorithms for object-oriented IVHS traffic simulation.

Intuitively, an m-fold speed improvement for object-oriented simulation over m distributed processing elements can be expected. However, because of inter model interactions and potential causality violations, the expected gain in simulation speed may be noticeably less than the full $m-$fold improvement. Since the prime goals for distributed object-oriented traffic simulation are

scope and speed, distributed submodels should be balanced in both size and speed. Balance in the pace of simulation clocks among submodels is the most significant factor in achieving the desired speed of the overall simulation. One natural assumption is that the simulation clock of an object-oriented traffic simulator runs inversely proportional to some power of the number of active vehicles in the model. Strategies to balance the number of objects among submodels while minimizing inter-model interactions (messages or objects transferred) are examined and a near optimal submodel decomposition strategy is developed.

In addition to submodel decomposition for distributed simulation, certain algorithms that deal with objects across submodel boundary may be distributed and the results from submodels combined to provide solutions for use by objects among all the submodels. One such example is the shortest path between any origin/destination node-pair of a traffic simulation model. Shortest paths within a submodel may be computed locally; the results must be combined to yield shortest paths across submodels. In this paper, it is shown that the Floyd algorithm to compute shortest paths for all node pairs may be revised and distributed to achieve a $10m^3$–fold improvement in computation time using m distributed processing elements. Alternatively, the revised Floyd algorithm may be applied sequentially to all submodels and the results from submodels combined to obtain shortest paths for the entire model with a $10m^2$–fold speed improvement without the use of distributed or parallel processors. Consequently, a revised and distributed Floyd algorithm offers a real possibility for use in real-time IVHS applications without the need of buying an expensive massive parallel processing computer.

2 SUBMODEL DECOMPOSITION

The scope and speed of an object-oriented simulation may be improved noticeably by distributed processing. The first step to achieve this goal is to balance the number of objects among submodels, subject to additional constraints such as adjacency of objects and minimum boundary interactions. The objectives of balanced submodels, maintaining adjacency, and minimizing inter-model interactions may not always be feasible because of the dynamic nature of all the contributing factors. Therefore, optimal submodel decomposition appears to be NP-hard, and a polynomial time algorithm to solve for optimal submodel decomposition is unlikely to exist. Consequently, one will have to settle for near optimal by simplification or heuristic rules. Some submodel constraints in traffic simulation, such as the need to minimize submodel interactions and the desirability of including closely connected nodes in the same submodel (because of traffic signal synchronization) may turn out to be incompatible to each other. As a result, perfectly balanced submodel decomposition may not always be achievable. For practical applications, any submodel decomposition that is well-balanced and near optimal should be quite acceptable. The following polynomial time algorithm is an approach designed to maintain adjacency, minimize boundary interactions, and balance model size for object-oriented traffic simulation.

In our approach, a traffic network (model) consists of intersections (nodes) and surface streets or freeway segments (arcs or links). Submodels are defined by subsets of adjacent nodes and related arcs. Submodels are sized by the sum of all node traffic, R_x, for all the nodes in a submodel. To decompose a large traffic network consisting of thousands of nodes and arcs into m balanced submodels, the following steps are recommended.

I Node and link sizing: Each node, $i \in N$, (the set of intersections or branching points) is sized by total expected rate of vehicles, R_i, (volume in unit of time) passing at the node, i. Similarly, each link, $(x, y) \in L$ where x and y are nodes in N and L is the set of all links in a traffic network, is sized by $T_{(x,y)}$, the total expected rate of vehicles passing thru that link. More precisely, we define

$$R_i = \sum_{i \in P_j} \lambda_j \quad ,$$

$$T_{(x,y)} = \sum_{(x,y) \in Q_i} \lambda_i$$

where P_j : consists of all nodes traversed by the j-th path

Q_i : the set of all links traversed by the i-th path

λ_j or λ_i : the arrival rate of trips using path j or i

II Expected submodel size: Compute expected submodel volume: $E_m = (\sum_i R_i)/m$.

III Recursion: Let $r = 1$ and $U = N$, where U is the set of unallocated nodes available for submodel allocation. The recursion consists of five substeps:

i *Seed identification*: A seed, $x \in U$, for a new submodel S_r is determined by the highest expected node traffic volume in U. Equivalently, a new submodel is created as:

$$S_r = \{x \mid R_x = \max_{i \in U}\{R_i\}\}$$

ii *Boundary links computation*: Compute boundary links for S_r as $B_r = \{(x,y) \mid$ with $(x,y) \in L$, and $x \in S_r$, and $y \in U$, or vice versa $\}$ If $B_r = \emptyset$ and $\sum_{x \in S_r} R_x < (1/2)E_m$, redistribute S_r to its neighboring submodels by Step IV

iii *Submodel expansion*: Increase submodel size of S_r by selecting the node $z \in U$ with $y \in S_r$ for the highest link sum $L_{(z,y)} + L_{(y,z)}$, and all the nodes, $w \in U$ such that either link (w,y) or (y,w) in B_r has a very small link length $l_{(w,y)}$ or $l_{(y,w)}$. Equivalently, let the nodes selected be ΔS_r with

$$\begin{aligned}\Delta S_r = &\{z \mid T_{(z,y)} + T_{(y,z)} = \\ &\max_{(x,y) or (y,x) \in B_r} \{T_{(x,y)} + T_{(y,x)}\}\} \\ &\cup \{w \mid l_{(y,w)} < l_{min} \text{ or} \\ &l_{(w,y)} < l_{min};\ \forall\, y \in S_r\}\end{aligned}$$

where $l_{(x,y)}$ denotes length of line (x,y) and l_{min} is the minimum link length acceptable for a boundary link between submodels. Node z from U is unique while node(s) w may be different from node z or none. The inclusion of w in ΔS_r is optional and the value l_{min} should be determined by the overall model topology. Optimal value of l_{min} such that the ratio of total node traffic volume of the largest submodel to that of the smallest submodel is minimum may differ from model to model. Heuristic and iterative procedures can be used to determine the range and sensitivity of l_{min} for a given network.

iv *Recursion*: Update $S_r = S_r \cup \Delta S_r$ and $U = U - S_r$. Repeat ii and iii until $\sum_{x \in S_r} R_x > E_m$ or $B_r = \emptyset$ with $\sum_{x \in S_r} R_x > (1/2)E_m$.

v *Termination*: If $\sum_{z \in U} R_z > (1/2)E_m$, increase r and repeat i, ii, and iii, otherwise, distribute nodes in U to neighboring submodels by Step IV.

IV Distribution of nodes to neighboring submodels:

We now reach a regional submodel with size less than $(1/2)E_m$ or the remaining nodes in U are not adjacent to each other and the size of U is less than $(1/2)E_m$. Since minimizing inter-submodel traffic is equivalent to maximizing intra-submodel traffic because of their complementary nature, the best strategy in this case is to redistribute these nodes to its immediate neighboring submodel with the highest link traffic volume. To achieve this, for each node, x, in S_r or U from Step III, determine its neighboring links $M_x = \{(x,y) \in L$ or $(y,x) \in L \mid y \notin U(\text{or } S_r)\}$. If M_x is nonempty, determine y^* such that $T_{(x,y^*)} + T_{(y^*,x)} = \max_{\forall (x,y) \in M_x}\{T_{(x,y)} + T_{(y,x)}\}$. Add node x to the submodel where y^* belongs if the submodel does not exceed $2E_m$. Otherwise, select the submodel adjacent to x with the smallest submodel size. Repeat Step IV until either S_r or U is empty.

The proposed submodel decomposition algorithm is local greedy in that it always seeks for a neighboring node with the most traffic volume while maintaining the size of a submodel as close to E_m as possible. Since traffic may be distributed to a given network in any conceivable way, no single decomposition algorithm can balance all submodels perfectly while maintaining adjacency and the node clustering (ignoring very short links) requirements, the goal of submodel decomposition should be limiting the maximum submodel size. For randomly distributed objects and events, this local greedy approach will generate submodels with the smallest submodel size greater then $(1/2)E_m$ (Steps III-iv & III-v) and the maximum submodel size smaller than $2E_m$ (Step III-ii and Step III-v may contribute up to $(1/2)E_m$ each).

3 DISTRIBUTED PATH UPDATES

Traffic simulation for IVHS requires the simulation of ATIS. One important aspect of ATIS is the capability of providing route selection and route guidance information to individual drivers. Consequently, there is a need to determine shortest paths for every driver from his current position to his destination. Effective algorithms to compute shortest paths between node (or link) pairs include Moore (1957), Dijkstra, and Floyd algorithms (Chandy and Misra, 1982), (Mateti and Deo, 1982), (Harary, 1968), (Chrsutofides, 1975), and (Berge, 1973). The Moore algorithm was shown to have a worst case of exponential search in computational complexity; the Dijkstra algorithm computes shortest paths from a single node to all other nodes; while Floyd algorithm computes shortest paths for all node pairs. Both the Dijkstra and Floyd algorithms are of $O[n^3]$ in computational complexity. Recently, a modified Floyd

algorithm was shown to achieve a computational complexity of $O[n \log n]$ with $\sqrt{n}$ massive parallel processors (Romeijr and Smith, 1992). Without the use of $\sqrt{n}$ parallel processors, its Floyd algorithm is of $O[n^3 \log n]$ which is slower than the original Floyd algorithm of $O[n^3]$. Moreover, aggregation techniques are needed to achieve the desired degree of parallelism. The modified Floyd algorithm can only provide approximations to shortest paths across submodels as macronodes (Romeijr and Smith, 1992). In this paper, we take a different approach to improve the Floyd algorithm for distributed traffic simulations.

Two factors contribute to the significant improvement in the speed of the Floyd algorithm. First, the triple operation described below need only be computed for adjacent arcs (i,l) and (l,j) while l is reachable from i and j is reachable from l. $d_{ij}^{(k)}$ is the shortest path length (delay, or cost) from node i to node j computed during the kth iteration.

$$d_{ij}^{(k)} = \min_{\forall h}\left\{d_{ij}^{(k-1)}, d_{ih}^{(k-1)} + d_{hj}^{(k-1)}\right\},$$
$$\forall \quad 1 \le i,j,k \le n$$

For traffic networks, not every node or link are reachable from all the other nodes (or links). For example, source nodes do not reach each other, sink nodes do not reach each other, and certain links and nodes have only one-way traffic. Besides, most of the nodes or links are not adjacent to each other. Therefore, the Floyd algorithm may be modified to take this sparsity in network connectivity into consideration by using the adjacency relation of nodes and links and performing only the required triple operations. Figure 1 compares the original Floyd algorithm with a revised Floyd algorithm. Both algorithms store shortest paths between every node pair as a successor matrix. The total number of updates of the successor matrix needed for the revised Floyd algorithm is considerably less than the original Floyd algorithm in a sparsely connected network, e.g, a traffic network consisting of arterial streets, intersections, and freeway segments. Experience on traffic networks has shown that the expected improvement in speed of the revised Floyd algorithm is 10 times faster over the unmodified algorithm. The gain of 10-fold improvement in speed can be achieved without using either distributed or massively parallel computing resources.

Second, the revised Floyd algorithm can be further enhanced to provide exact solutions of shortest paths for all node (or link) pairs using distributed computing resources. The following distributed Floyd algorithm will achieve an additional $[m^3]-$ fold in speed.

Let S be an n node traffic network to be decomposed evenly into m subnetworks with approximately n/m nodes in each subnetwork. Assume that nodes in a subnetwork are adjacent to each other such that no hole or island nodes are allowed in each subnetwork. Let I_k and O_k be the sets of boundary nodes of the $k-$th subnetwork for which there is an incoming and outgoing arc respectively. For a typical traffic network, the size of I_k and O_k are of $O[\sqrt{n_k}]$; where n_k is the size (number of nodes) of the subnetwork (or submodel) S_k.

Shortest paths between every node pair in a submodel can be computed with the revised Floyd algorithm in Figure 1 in $O[n_k^3]$. Referring to Figure 2, let P_{xy} be a path from a node x in the $i-$th submodel to a node y (exit nodes in O_j only) in the $j-$th submodel. The shortest path between x and y is simply:

$$\min_{j \in O_i}\left(P(x,j) + P(j,y)\right)$$

If the shortest paths $P(x,j)$ and $P(j,y)$ are known, exact shortest path $P(x,y)$ can be computed accordingly. The shortest path $P(x,j)$ is available from the submodel S_i. For $P_{j,y}$, we can construct an aggregated network (a supermodel) consisting of only and all the nodes in I_k and O_k for $k = 1,2,\ldots,m$. Given the successor matrices for all shortest paths within each submodel, and applying the revised Floyd algorithm to the aggregated supermodel, we obtain shortest paths for each node pair in the original model, S, in $O[h^3]$, where $h = \sum_{k=1}^{m} c(\sqrt{n_k}) = c(\sqrt{nm})$. The parameter c is the ratio of the expected number of boundary nodes in a submodel to $\sqrt{n_k}$ where n_k is the expected submodel size. For traffic networks, we have approximately $4 \le c \le 5$. If all submodels are perfectly balanced in nodes, i.e., $n_k = n/m$, The computational complexity of the revised Floyd algorithm is $O[(n/m)^3]$; while that of the supermodel is $O[(\sqrt{nm})^3]$. Without using distributed or parallel computing elements, we can use a "divide and conquer" approach to achieve an $10m^2-$fold speed improvement of the Floyd algorithm by computing each submodel sequentially. If $n < m^2$, the Floyd's algorithm for the super-

model can be computed faster than the submodel and a m^3- fold improvement in speed over the undistributed model, S, can be achieved. Consequently, we conclude that with the distributed and revised Floyd algorithm, a $10m^3$ improvement in shortest path computation for real time ATIS or ATMS is theoretically feasible.

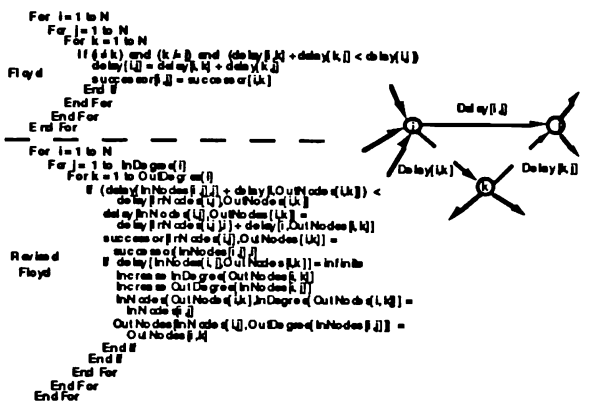

Figure 1 A Revised Floyd Algorithm

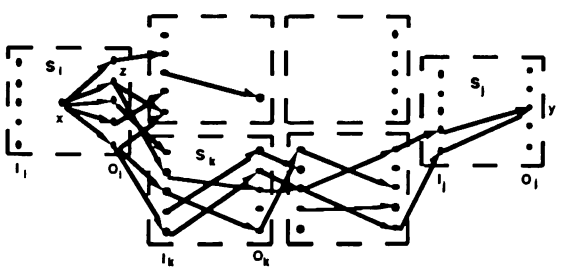

Figure 2 Distributed Shortest Paths Across Submodels

4 CONCLUSIONS

We have concluded that distributed, object-oriented, traffic simulation is a very promising technique for IVHS applications; it offers potential for use in real time ATIS or ATMS applications. A local greedy algorithm to decompose a large traffic network into balanced submodels for distributed simulation is recommended. The Floyd algorithm is enhanced by skipping redundant triple operations to achieve a 10–fold and additional m^3–fold improvements using m distributed processors. A technique to obtain exact shortest paths for the aggregated model from the distributed submodel is also provided. Future research in distributed/parallel object-oriented traffic simulations should include enhancement and analysis of various ATIS and ATMS control algorithms, examination of various object-oriented simulation models for their speed factors (i.e., number of objects vs. simulation time) so that effective distributed/parallel techniques may be derived.

ACKNOWLEDGMENTS

The authors thank Michael McGurrin for his leadership and guidance in developing a new traffic simulation model based upon state-of-the-art object-oriented and distributed simulation concepts. He has served as the Project Leader for a number of MITRE sponsored research projects that leads to the development and enhancement of the parallel THOREAU model for IVHS studies. The authors also thank Bob Nystrom, Karl Wunderlich, and Victor Hsin for their reviews and helpful comments.

REFERENCES

Berge, C. 1973. *Graphs and Hypergraphs,* University of Paris, North-Holland publishing Company, Amsterdam.

Chandy, K. M. and Misra, J. 1982. "Distributed Computation on Graphs: Shortest Paths Algorithms," Communications of the ACM, Vol. 25, Number 11, November, 1982.

Christofides, N. 1975. *Graph Theory,* Management Science Imperial College, Academic Press, London.

Edward F. Moore. 1957. "The Shortest Path Through A Maze", International Symposium on Theory of Switching," Harvard University, pp. 285-292.

Federal Highway Administration. 1991. "An Overview of the IVHS Program Through FY 1991," October 1991, DOT, Washington, D.C.

Harary, F. 1968. *Graph Theory,* Addison-Wesley Publishing Company, Reeding, Massachusetts.

Hsin, V. J. and Wang, P. T. R. 1992. "Modeling Concepts for Intelligent Vehicle Highway Systems(IVHS) Applications", *WSC'92 Conference Proceedings,* Arlington, Virginia, December 1992, pp. 1201-1209.

Mateti, P. and Deo, N. 1982. "Parallel Algorithms for Single Source Shortest Path Problem," *Computing*, Vol. 29, pp. 31-49.

McGurrin, M. F. and Wang, P. T. R. 1991. "An Object-Oriented Traffic Simulation with IVHS Applications," *VNIS'91 Conference Proceedings*, Detroit, Michigan, September, 1991, pp. 551-561

Romeijr, H. E. and Smith, R. L. 1992. "Parallel Algorithms and Aggregation for Solving Shortest-Path Problems," IVHS Technical Report: #92-66, The University of Michigan, Ann Arbor, Michigan, February.

Wang, P. T. R., Niedringhaus, W. P., Codelli, D. K. and McGurrin, M. F. 1992. "A Comparison of Travel Time Reduction Using Current vs. Partially Predictive Travel Time," *VNIS'92 Conference Proceedings*, Oslo, Norway, September, 1992, pp. 476-479.

AUTHOR BIOGRAPHIES

PAUL T. R. WANG, graduated from National Cheng Kung University, Taiwan, China in 1966. He received the M.S. and Ph.D. degrees in Computer and Information Sciences from the Ohio State University in 1973. From 1973 to 1983 he was with the General Electric Company, serving as a consulting engineer at the Information Services Division in Rockville, Maryland. Since 1983, he has been with the MITRE Corporation. Currently, he is a Lead Engineer at MITRE's IVHS site located at Washington D.C. His interests include optimization, simulation, modeling, and queuing theory. He is a member of MAA.

WILLIAM P. NIEDRINGHAUS, born in Pittsburgh, PA in 1949, received the B.S. degree in Mathematics from Yale University (New Haven, CT) in 1971, and the Ph.D. degree in Electrical Engineering and Computer Science from Princeton University (Princeton, NJ) in 1980. He is a senior member of AIAA. He has worked at The MITRE Corporation in McLean, VA, since November 1979, on algorithms for automated airborne collision avoidance and automated ground based air traffic control, as a Group Leader (1983-85) and as a Lead Engineer (since 1986). In 1990 he joined MITRE's Modeling and Simulation Technical Center. He has had extensive experience in simulation design, object-oriented design, and parallelization of simulations.