1251

TRANSPORTATION RESEARCH RECORD

Transport Supply Analysis

TRANSPORTATION RESEARCH BOARD
NATIONAL RESEARCH COUNCIL
WASHINGTON, D.C. 1989

Transportation Research Record 1251
Price: $12.00

modes
1 highway transportation
2 public transit

subject areas
12 planning
13 forecasting

Printed in the United States of America

Library of Congress Cataloging-in-Publication Data
National Research Council. Transportation Research Board.

Transport supply analysis.
p. cm.—(Transportation research record, ISSN 0361-1981 ; 1251)
"Contains a series of papers sponsored by the Committee on Transportation Supply Analysis"—Foreword.
ISBN 0-309-05002-2
1. Transportation—Mathematical models. I. National Research Council (U.S.). Committee on Transportation Supply Analysis.
II. Series.
TE7.H5 no. 1251
[HE147.7]
388 s—dc20
[388'.01'5118] 90-35960
CIP

Transportation Research Board publications are available by ordering directly from TRB. They may also be obtained on a regular basis through organizational or individual affiliation with TRB; affiliates or library subscribers are eligible for substantial discounts. For further information, write to the Transportation Research Board, National Research Council, 2101 Constitution Avenue, N.W., Washington, D.C. 20418.

Transportation Research Record 1251

Contents

Foreword

This Record contains a series of papers sponsored by the Committee on Transportation Supply Analysis.

Daganzo proposes two approaches to determine the effect of crane operations on ship service at port terminals, first, a simple approximate approach to calculate the maximum berth throughput during periods of congestion, and second, the effect of two extreme crane-operating strategies when the traffic level does not exceed the maximum throughput.

Mouskos and Mahmassani describe the modification of the CRAY-X-MP series to enhance its vector-processing performance. Specifically, codes for the solution of two network equilibrium assignment problem formations are vectorized and tested on a CRAY X-MP24 supercomputer. The test results and the significance of the results for research and practice are also discussed.

Safwat and Hasan expand and improve on the application of STEM (Simultaneous Transportation Equilibrium Model). They report the results of their work to investigate the relative computational efficiency of the algorithm as a function of demand, performance, and network parameters for two small example and one large-scale real-world network. The results are encouraging according to the authors, and application of the STEM approach to large-scale urban transportation studies is encouraged.

Chan et al. applied the tree-search method to three spatially abstracted networks, coming up with a hierarchical search algorithm for reducing the network-design problem. The branch-and-bound and branch-and-backtrack techniques were used in the first two formulations of the problem, assuming the objective function of least budget and least travel cost, respectively. These techniques result in a greatly reduced search space as well as functional grouping of the detailed links into access/egress, line-haul, and by-pass categories.

Tsay and Lin describe methods for selecting the optimal facility location. The authors focus on the one-center problem. Various methods in current use and their applications are described and assessed.

Smith and Sheffi discuss the problems faced by railroads in allocating power to trains. The authors formulate a multicommodity flow problem with convex objective function on a time-space network. The convex objective allows a minimization of expected cost under uncertainty by penalizing trip areas likely to have too little power. The author solution heuristic sends locomotives down shortest paths in the time-space network and attempts to improve interchanges of locomotives around cycles. The test results are reported.

Chang et al. discuss a time-space network formulation for the system-optimal assignment of commuters to departure times and routes subject to specified constraints on acceptable arrivals. Time is discretized, and congestion is represented using simplified deterministic queueing stations. A numerical application is presented to illustrate the methodology, indicating a network generator developed for this purpose.

Wie explores the application of optimal control theory to the problem of dynamic traffic assignment corresponding to use optimization. Two continuous time formulations are considered, one with fixed demand and the other with elastic demand. As stated by the author, the paper is concerned with dynamic extensions of the steady-state network equilibrium model, in particular Beckmann's equivalent optimization problem, which is a mathematical programming function.

Crane Productivity and Ship Delay in Ports

CARLOS F. DAGANZO

This paper studies the effect of crane operations on ship service at port terminals. It first proposes a simple, approximate approach to calculate the maximum berth throughput during periods of congestion. The key assumption is that the workload distribution (over time) for the ships at berth is the same as the workload distribution for the ship population as a whole. The validity of this assumption is tested with simple, exact models for a variety of scenarios involving different kinds of ships and crane operating strategies. The paper then examines the effect that two extreme crane operating strategies have on ship delay, when the traffic level does not exceed the maximum throughput. This is done for an idealized situation designed to highlight the impact of crane operations while admitting closed-form solutions. The average ship delay can vary considerably with the crane operating strategy.

A port's efficiency is often measured in terms of its throughput and typical ship turnaround time (i.e., a ship's time at berth plus any delay caused by the port). High turnaround times are not acceptable in the shipping industry because of the very large opportunity cost typically associated with ship delay. However, port construction, maintenance, and equipment are also very expensive. Thus it is important for ports to set an appropriate expenditure level, and to allocate their resources efficiently among their different functions. For example, they should decide carefully how berth length should be divided among the various traffic types, and how much cargo handling equipment should be allocated to each terminal. Although such decisions often depend on factors that cannot be quantified, rational solutions should be found with an understanding of how the ships' turnaround time and the port throughput depend on different resource allocation levels.

The port elements that influence ship turnaround most directly are berth space and crane availability. Although other elements have the potential for delaying operations (tugboat unavailability and land-side congestion, for example) they are not considered in this paper.

Even though queuing theory has been applied to ports [see, for example, work by Plumlee, Mettam, Jones and Blunden, Nicolau, Miller, Koenisberg and Meyers, Daskin and Walton, and Sabria (*1–8*), and other references in Sabria's dissertation], and to the berth system in particular, no models seem to recognize explicitly the interaction between berth availability and crane operating strategies. This may be because the requirements for onshore (un)loading equipment can vary considerably from ship to ship, and may also be subject to peculiar restrictions, which complicates matters. Work by Atkins (*9*) contains one of the best descriptions of the ship loading process for modern container ports.

Department of Civil Engineering, University of California, Berkeley, 94720.

The goal of this paper is to develop an understanding of the impact that different crane scheduling strategies have in the long run on maximum throughput and ship delay. To achieve this goal, we will work with a representation of the world that, although highly idealized, preserves the phenomena of interest. The paper builds on previous work (*10,11*) that used the same idealized model to develop crane scheduling strategies.

The model in these references assumed that ships were divided into holds; that each hold had a certain amount of work that needed to be done (measured in time units of crane time); that certain holds could be handled without the need for a shore crane; and that shore cranes could be moved rapidly. The objective was to assign cranes to holds to reduce ship delays. Sometimes this meant that a large ship with little need for cranes would seize the cranes working on another ship that required more work.

For the most part all the ships were assumed to be already at berth, but a case in which ships had to queue for berth space was also discussed. For this purpose it was assumed that a ship departure always freed enough space for another ship and that ships were chosen from the queue in order of arrival. A justification for all of these modeling simplifications (which are also adopted here) can be found elsewhere (*10*).

This paper attempts to take these results one step further. It studies the system's steady-state performance as a function of the ship arrival pattern when the aforementioned crane operating rules are used. It presents simple expressions for maximum expected throughput as a function of the number of cranes and total berth length. It also provides ship delay formulas when ships have to queue for cranes but the berth space is never in short supply.

The next section gives approximate expressions for the average number of busy and idle cranes during periods of congestion. These expressions lead to berth throughput and crane productivity formulas. The approximation, which is proposed for reasonably efficient crane operations, is tested with exact expressions for a special case in which all the holds requiring a crane take the same amount of time to be handled. (This assumption, which still preserves the main phenomena we want to model, is also used in later sections.)

The following section applies the results from the previous one; it compares efficient and inefficient crane scheduling strategies and examines the trade-off between cranage cost and maximum productivity.

Next is a study of ship delay for a multipurpose terminal in which ships are either self-sufficient or require, at most, two cranes. It is assumed that berth space is never in short supply [this is reasonable from a port economics standpoint

(*8*)], but ships may not always get all the cranes they need immediately on arrival. The final section summarizes the results and suggests further work.

CRANE PRODUCTIVITY

Averaged over time, the number of busy port cranes is related to cargo throughput by the relationship:

(cargo throughput)

$$= \text{(busy cranes)} \times \text{(crane capacity)} \qquad (1)$$

where the crane capacity is the maximum number of cargo units that a fully used crane can handle per unit time.

It is thus important to be able to predict the number of busy cranes during periods of congestion. The result can indicate the maximum possible berth throughput.

A Simple Model

We assume that there is an infinite ship queue and that the berth can hold exactly S ships. The ith ship to enter the berth is assumed to have H_i holds requiring attention. The H_i are mutually independent, identically distributed random variables with cumulative distribution function, F_H.

At any given time, the number of busy port cranes equals the minimum of two values: the number of available cranes, C, and the number of active holds, A (holds still requiring attention at the time).

If the number of active holds present at a berth at a random time has the same cumulative distribution function (cdf) as the number of holds requiring attention for S ships randomly sampled from the queue, then berth throughput can be calculated simply. The accuracy of this assumption is tested in the next section. The resulting simple throughput expressions are derived next.

Because A is distributed like the sum of S independent, identically distributed random variables with cdf, $F_{H(h)}$ A is likely to be well approximated by a normal random variable and the expected number of busy cranes by the mean of the truncated normal variable, min $\{A, C\}$:

$$E\,(\text{busy cranes}) = C - \sigma\sqrt{S}\,\psi\,([C - Sm]/\sigma\sqrt{S}) \qquad (2a)$$

and similarly

$$E\,(\text{idle cranes}) = \sigma\sqrt{S}\,\psi\,([C - Sm]/\sigma\sqrt{S}) \qquad (2b)$$

In these equations, m and σ^2 are the mean and variance of H_i, and ψ $(*)$ represents the integral of the standard normal cumulative distribution function. This function is given by $\varphi(*) + (*)\Phi(*)$, where $\Phi(*)$ and $\varphi(*)$ are the standard normal cdf and probability density function, respectively [see Clark (*12*) for a derivation]. The function $\psi(*)$ is positive, increasing, and convex; it approaches 0 as its argument approaches $-\infty$, and for large positive arguments (greater than 3) its value barely exceeds the argument. See Figure 1.

Equation 2b shows that the number of idle cranes depends on only two parameters: the "average crane surplus," $C - Sm$, and the "holds at berth variability," $\sigma\sqrt{S}$. Although the expected number of idle cranes always exceeds the average crane surplus, these two are close when there is little variability.

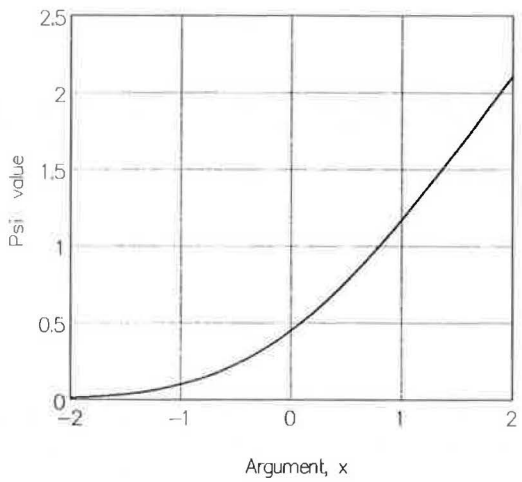

FIGURE 1 Graph of ψ $(*)$.

Equation 2a can be used in conjunction with Equation 1 to calculate berth throughput.

An Assessment of Its Accuracy

Equations 2a and 2b are based on the assumption that the distribution of active holds per ship is the same at berth and in the queue.

Two factors that work in opposite directions (with an intensity that depends on the specific crane scheduling strategy) tend to disrupt this equality:

1. A ship's hold with little work may become inactive before the ship departs. If this happens often, it will tend to decrease the number of active holds at berth; and
2. Because, with an efficient strategy, ships with low workloads are given priority, the ships with most active holds will tend to be overrepresented at the berth. This tends to increase the number of active holds at berth.

The first factor should be most significant when the distribution of (active) hold workloads within a ship is very uneven. The second factor should be most significant when the workload changes drastically across ships.

The scheduling strategy discussed elsewhere (*10*) tends to reduce the impact of the first factor and increase the impact of the second. As mentioned in that reference, the strategy "tends to hoard at berth the holds that require work." In the remainder of this section we derive, for comparison purposes, exact expressions for two simple cases in which the first factor does not play a role and the crane allocation strategy proposed in earlier work (*10*) is used.

It is assumed that all the active holds take exactly the same amount of time (without loss of generality we take this time to be one unit) and that only one crane can work on a hold at a time. We start our observation with an empty system; thus, at time $t = 0$, the first S ships in the queue join the berth.

The preceding assumption, which is also used for the ship delay analysis, still preserves the main phenomena that we try to model; that is, because not all ships require the same number of cranes, the number of cranes needed by the ships at berth fluctuates. If at times there is a need for more cranes than the available number and at other times some cranes are idle, crane productivity is wasted. Our crane allocation rules are designed to restrain these fluctuations.

Before starting the analysis, strategy G needs to be described. For the simple cases studied in this paper (in which holds can be handled in exactly one time unit, etc.), strategy G reduces to the following:

Strategy G: Each time a new ship joins the berth reallocate all the cranes again; assign as many cranes as possible to the ship with fewest active holds; if some cranes are left, allocate as many as possible to the ship with second fewest holds; repeat this process until either no more cranes or no more ships are left.

The results of the analysis about to be presented indicate that the approximate and exact formulae are pretty close. Although the expressions should be tested further (with simulations geared to verify the importance of the first factor), the results suggest that Equations 2a and 2b may be good first-order approximations, useful for planning purposes.

Multipurpose Terminals

Two types of ships are considered in this subsection: type-0 ships that do not require the port's equipment ($H_i = 0$) and type-1 ships that require exactly one crane ($H_i = 1$). This situation could represent a multipurpose terminal. It is studied first because, with this traffic pattern, one does not require an involved crane scheduling algorithm. Allocating cranes to ships on a first berthed, first served, basis (which happens to be the result of strategy G) results in maximum productivity.

Type-0 ships spend exactly one time unit at berth, but type-1 ships may spend a little more time; they may have to wait for a crane if the berth has more type-1 ships than there are cranes. Thus Factor 2 applies. There will tend to be more active holds at berth than would be predicted from the queue, and Equations 2a and 2b should underpredict throughput.

Because all the holds take exactly one time unit to be handled, and because the system starts empty, ships and cranes move only at integer times ($t = 0, 1, 2, \ldots$). The number of active holds at berth can change only at these times. In fact, the whole system can be modeled exactly as a Markov chain embedded at integer times. The state is the number of (type-1) ships remaining at berth at the end of one period, but immediately before the next batch of ships joins the berth. It is thus possible to derive exact numerical results to compare them with the approximation.

Let p denote the fraction of type-1 ships. Then, the (i,j) element of the one-step transition probability matrix, M, m_{ij}, is:

$$m_{ij} = \Pr\{(C + j - i) \text{ type-1 ships join the berth}\}; \text{ if } j = 1, 2, \ldots, S - C$$
$$= \Pr\{(C - i) \text{ or less type-1 ships join the berth}\} \text{ if } j = 0.$$

For $j > 0$ the m_{ij} are the binomial probabilities:

$$m_{ij} = \binom{S-i}{C+j-i} p^{(C+j-i)}(1-p)^{(S-C-j)}$$

For $j = 0$, $m_{i0} = 1 - (m_{i1} + m_{i2} + \ldots + m_{i(S-C)})$

The steady-state probability (row) vector, $\mathbf{m}$, can be obtained by solving: $\mathbf{m} = \mathbf{mM}$, and ensuring that its elements, m_i, add up to 1.

The expected number of cranes in use, K, is

$$K = \sum_{i=0}^{S-C} m_i \left[\sum_{j=0}^{S-C-i} \min[j + i, C] \Pr\{j \text{ type-1 ships join the berth}\} \right]$$
$$= \sum_{i=0}^{S-C} m_i \left[\sum_{j=0}^{S-C-i} \min[j + i, C] \binom{S-i}{j} p^j (1-p)^{(S-i-j)} \right]$$

The expected number of cranes in use also gives the throughput of type-1 ships. Because the fraction of these ships is p and because ships join the berth on a first come, first served, basis, the total ship throughput, P, must be $P = K/p$.

Example: Assume that $S = 3$ and $C = 2$. In this case the calculations required by the preceding expressions are simple. We obtain the exact result:

$$P = 3 - [p^3 (1 - p^2 + p^3)]$$

If the distribution of ships at berth is the same as in the queue, the number of busy cranes is the minimum of C and a binomial random variable with S trials and probability of success, p. For our case, the expectation of such a variable is $(3p - p^3)$, and

$$P = 3 - p^2$$

As expected, this expression underpredicts the exact one, but the maximum difference is only about 0.1 when $p \approx 0.6$. The error is much smaller when p is close to 0 or 1; it never exceeds 4 percent. See Figure 2.

The result derived from Equation 2a, which includes a normal approximation to the binomial, is quite close to this last expression (for $p = \frac{2}{3}$, one obtains $P = 2.56$ with the last expression and $P = 2.51$ with Equation 2a). The normal approximation would be even better in a case with more berths and cranes, just when the binomial calculations become cumbersome.

In general, either approximation should be quite good if $p << C/S$, because then cranes are almost never in short supply and both ship types spend the same time at berth. The approximations should also be quite good when $p >> C/S$, as then the exact and approximate formulas predict $K \approx C$. These observations are consistent with the example; the worst underprediction occurs when $p \approx C/S$.

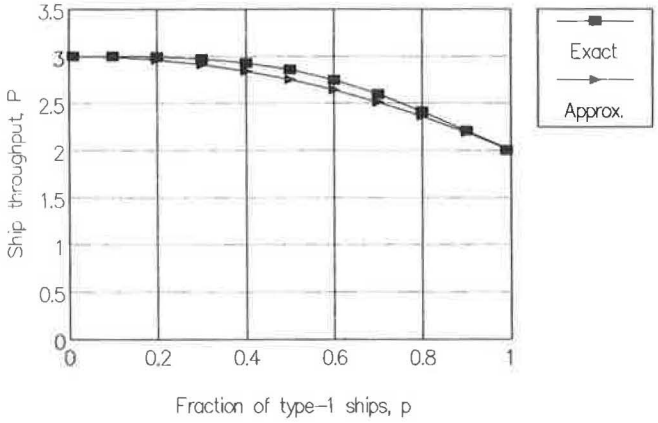

FIGURE 2 Comparison of exact and approximate throughput expressions for $S = 3$ and $C = 2$ at a multipurpose terminal.

Single-Purpose Terminals

This subsection explores the accuracy of our simple model in a more complex situation. It is assumed that all the ships require at least one crane, but that ships can have a varying number of active holds. Now the crane scheduling strategy can make a difference, and strategy G is used. (The implications of changing the strategy are examined in the Crane Usage Evaluation section.)

As in the preceding section, the system can be modeled as a Markov chain. Here the state is a vector composed of the numbers of ships with 1, 2, 3, . . . active holds that are still at berth immediately before the next batch of ships joins the berth. Although the state space is multidimensional, it is finite; numerical analysis is possible.

We present numerical results for a terminal with 4 cranes. In the first instance (Case A) we assume that the berth can hold 3 ships and that the ships request either 1 or 2 cranes each. For the second case (B) the berth can hold only 2 ships, but the ships can request either 1, 2, or 3 cranes.

Case A is characterized by a single parameter: the fraction, p, of ships that have 2 active holds. Only three possible states are possible because, at most, 1 ship can be left at berth, and this ship can only have either 1 or 2 active holds. This makes the search for the steady-state probability vector (and the associated measures of performance we seek) rather simple; the analysis is equally simple for an arbitrary number of ships and cranes. The process is similar to that outlined in the preceding section. Thus, only the results are given here.

The productivity in ships handled by the berth per unit time, P, is

$$P = 3 - p^2 [(3 + p)/(1 + p + 2p^2)] \quad (3)$$

Clearly, P cannot exceed 3.

In holds per unit time, the productivity is equal to the average number of busy cranes, K. Because the average number of holds per ship is $(1 + p)$, $K = P(1 + p)$. This reduces to

$$K = 4 - [(1 + p^2)(1 - p)^2/(1 + p + 2p^2)] \quad (4)$$

which cannot exceed $C = 4$. Note that, as expected, if all ships have 1 hold ($p = 0$), then $P = 3$ and $K = 3$; also as expected, if all ships have 2 holds ($p = 1$), $P = 2$ and $K = 4$. Figure 3 shows graphically how P and K vary with p.

We now test the accuracy of the assumption that states that the distribution of active holds at berth is equal to the distribution of active holds for S ships in the queue. We calculate K assuming that the 3 berthed ships have been randomly taken from the queue. Then, 4 cranes will be at work unless the 3 ships have exactly 1 hold each. This happens with probability $(1 - p)^3$, and thus K is approximately given by

$$K \approx 4 - (1 - p)^3 \quad (5)$$

The maximum difference between Expressions 4 and 5 occurs when $p = 0.53$, which results in $K = 3.87$ and 3.90, respectively. The discrepancy is less than 1 percent. As expected, Equation 5 yields larger values than does Equation 4. If one uses Equation 2a, which also includes a normal approximation, the result is not very different (3.87 instead of 3.90); just by chance, it nearly matches the exact value, which is also 3.87. In any case, it appears that our assumption (about the distribution of active holds at berth) does not lead to large inaccuracies for this example.

For Case B, the ship workload changes more from ship to ship, thus one expects the approximation to be less accurate. Two parameters now define the problem: p, the probability that a ship has 2 active holds, and q, the probability that the ship has 3 active holds. Of course, the fraction of ships with a single active hold is $(1 - p - q)$. The Markov analysis can still be used. In this case, too, only three possible states can arise: the berth either is empty or has 1 ship that can have either 1 or 2 active holds; no other possibilities exist.

The berth productivity (ships per unit time) is found to be:

$$P = 1 + 1/(1 + 2pq + q^2 + q^3) \quad (6)$$

This value remains between ⅔ and 2; if there are no ships with 3 holds ($q = 0$), then, as expected, $P = 2$. The crane usage, which coincides with the number of holds served per unit time is $K = P(1 + p + 2q)$, where the quantity in parentheses is the expected number of active holds per ship:

$$K = (1 + p + 2q)(1 + 1/(1 + 2pq + q^2 + q^3)) \quad (7)$$

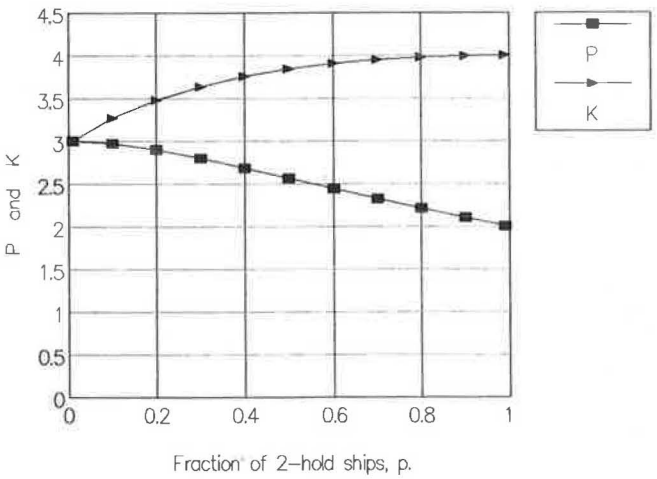

FIGURE 3 Exact expressions for berth throughput and crane usage at a single-purpose terminal with $S = 3$ and $C = 4$.

The approximation for K using the distribution of holds in the queue is:

$$K \approx 4 - 2(1 - q)(1 - p - q) \tag{8}$$

Note that if $(p + q) = 1$, Equation 8 yields $K \approx 4$, but consideration shows that it should be a little smaller: whenever 5 active holds are at berth, 1 ship with 1 active hold must remain for the next period. If the next ship requests only 2 cranes, 1 crane will have to be idle. Thus K cannot be 4 except when either $q = 1$ or $p = 1$. Then all the ships are identical, and 5 active holds can never be at berth; in that case the approximation is exact.

Equation 7 is consistent with these observations. The maximum difference between the exact and approximate expressions over all possible values of p and q occurs when $p = .72$ and $q = .28$. Then the exact value is 3.80, and the approximation is 4.00 (a 5.26 percent error). In most other instances the overprediction is less severe. The average (root mean square) error over all possible values of p and q is slightly less than 3 percent.

This error is not very large (given the rather large workload variability exhibited by this example), suggesting that Equations 2a and 2b may be reasonable predictors in actual situations. Still to be tested, however, is the extent to which Factor 1 counterbalances (and perhaps overcorrects) this error.

CRANE USAGE EVALUATION

The results from the section on crane productivity are now demonstrated. The section immediately following investigates the importance of an efficient crane allocation scheme and the subsequent section, the trade-off between cranage cost and maximum productivity.

Effect of a Bad Crane Allocation Method

In this subsection we explore the changes to productivity for the single-purpose terminal scenario of the preceding subsection, when a "bad" crane allocation method (strategy B) is used. This strategy, which is also described in an earlier work (*10*), is almost the exact opposite of strategy G. For the idealized scenarios in this paper, the strategy is easy to describe:

At every integer time ($t = 0, 1, 2 \ldots$), reallocate all the cranes (one at a time) to the ship with most active unattended holds.

As before, the system can be studied as a Markov chain, and the results are as follows:

$$P = 3 - p^2(3 + 2p + 2p^2)/(1 + p + 3p^2 + p^3 + p^4) \tag{9}$$

for case A, and

$$P = 1 + (1 - q^2)/(1 + 2pq + q^2) \tag{10}$$

for Case B.

Equations 3 and 9 are rather close. They differ the most in the range from $p = 0.4$ to 0.8, when the difference is on the order of 0.02 to 0.025. Thus for Case A, the specific crane allocation strategy used does not seem to matter much. The situations where the wrong crane choice influences productivity do not arise often enough. When $p = 0.5$, the cranes are idle 3.9 percent of the time with the good scheme but only 5.1 percent with the bad one.

For Case B we perform the same comparison when the average number of active holds per ship is 2. At one extreme, all the ships have 2 active holds ($p = 1$), and at the other extreme, half the ships have 1 hold and half, 3 holds ($p = 0$ and $q = 0.5$). When $p = 1$, both strategies are equal (clearly), and as one moves toward the other extreme the bad strategy deteriorates: 3.5 cranes are busy on average with the efficient strategy, but only 3.2 with the bad strategy. This is a 10 percent difference in productivity.

These comparisons illustrate the productivity increases that can be obtained with efficient operation. The increases are not enormous, but when ships are very different from one another, they can be significant. Even in these cases, however, the percentage changes in productivity are only a few percentage points larger than the errors in Equations 2a and 2b. This suggests that these expressions should be quite robust and applicable even if the scheduling strategy only vaguely resembles strategy G.

Although it may seem like a contradiction, increases in productivity comparable with the errors in Equation 2 should not be dismissed. A 5 percent increase in productivity would be highly desirable at a port, but a 5 percent error in our ability to predict it does not invalidate a preliminary planning tool (in fact, in the planning stages a 5 percent prediction error may be quite satisfactory).

The next subsection explores the trade-off between crane cost and productivity.

Optimum Number of Cranes

Clearly, there are some benefits associated with a high maximum productivity. If maximum productivity is increased, say, by purchasing more cranes, the terminal can attract more business and generate more revenue. Maximum productivity also increases with S (see Equation 2a). Thus, it is possible to use Equation 2a to determine the most cost-effective combination of berth capacity and number of cranes to achieve a certain productivity goal.

Equations 2a and 2b can also be used to determine the equipment needs for a given berth capacity. Let α denote the yearly marginal profit associated with one unit of productivity, and let us measure the productivity by the average number of busy cranes as given by Equation 2a. Let β denote the yearly cost associated with owning one crane. This cost does not include any operating costs, which should have been factored into α. Thus, the total yearly profit associated with owning C cranes is:

$$\text{Profit} = \alpha\{C - \sigma\sqrt{S}\,\psi([C - Sm]/\sigma\sqrt{S})\} - \beta C$$

This is a concave function of C, which will have a unique maximum at the point where the derivative vanishes: the root of the equation,

$$(1 - \beta/\alpha) = \Phi\{(C - Sm)/(\sigma\sqrt{S})\} \tag{11}$$

where $\Phi(*)$ stands for the standard normal cdf. Equation 11 has a solution if $\beta < \alpha$. The best number of cranes to have is the nearest positive integer to this solution. In reality one should never have $\beta > \alpha$ because this would mean that the profits obtained by continuous operation of a crane are not enough to offset its fixed cost: the terminal should not operate.

The calculations suggested in this subsection assume that the marginal profit associated with an extra productivity unit is constant. This is a coarse approximation that may be valid for long-term planning (when it is planned to use the terminal capacity nearly to its fullest), but not always. If, as is more common, to provide a good level of service to its users, the terminal is not used to its fullest, then the most significant benefit derived from the availability of more cranes is a reduction in ship delay; and ship delay is not linear with the number of cranes.

The next section derives ship delay expressions that can be used to address these questions.

SHIP DELAY

This section explores the relationship between ship delay and crane operations. As before, this is done by means of idealized models that can be solved analytically. It is assumed that ship arrivals to the terminal are stationary and random, and that while the terminal may not have enough cranes from time to time to serve all the ships at berth, the berth is long enough so that ship queuing is extremely rare. This should be the case at well-run ports and will help to separate the effects of crane operations on delay from those of berth availability.

The Model

Ships fall into two categories: type-0 ships that need no cranes and type-1 ships that need cranes. The service times of type-0 ships are arbitrary. Type-1 ships can be one- or two-hatched; that is, they may have either one or two active holds, which require exactly one time unit of a crane's attention.

Because the berth is almost never congested, it will be assumed that it never is; for all practical purposes, its length is infinity. This implies that the type-0 ships never interact with the type-1 ships and that the operations of both can be studied separately. Of course, to make sure that the infinite berth length assumption is reasonable, one will have to check a posteriori that the total number of type-0 and type-1 ships at berth is very unlikely to exceed the maximum possible number.

The two crane scheduling strategies already presented will be compared. Strategy G (good) gives priority to the ships with one active hold and strategy B (bad), to the ships with two holds.

For both ship types we seek the expectation and the variance of the number of ships at berth. The expectations give an indication of the cost of delay; and together with the variance they yield insight into the maximum number of ships that are likely to be present simultaneously at the berth.

Results

Let us focus our attention on the type-1 ships and imagine that all ships have exactly two holds; that is, one-hatched ships have another (empty) hold. Let Q denote the total number of holds at berth that are still active. This does not include any holds that have already been handled, even if the ship still is at berth. We define $Q = Q_1 + Q_2$, where Q_1 denotes the number of holds belonging to one-hatched ships and Q_2, to two-hatched ships.

The total number of type-1 ships, N_1, in the system can be obtained as a function of Q_1 and Q_2. This is because with strategy G all the active holds on two-hatched ships are spread across as few ships as possible. Thus,

$$N_1 = Q_1 + (Q_2/2)^+ \tag{12}$$

where the last term in this equation is rounded to the nearest integer, if Q_2 is odd. With strategy G, ships with one hold have priority. Thus, Q_1 can be visualized as the number of customers (holds) in a queuing system with C servers with deterministic (unit) service times. A simple model for Q_2, however, is not readily available (it would seem to require priority queues). To avoid this complication, we express N_1 as a function of Q_1 and Q. Because $Q = Q_1 + Q_2$, we can write:

$$N_1 = [(Q + Q_1)/2]^+ \tag{13a}$$

or approximately,

$$N_1 \approx (Q + Q_1)/2 \tag{13b}$$

This expression is more useful because the total number of holds can be modeled as a queuing system with C servers where the customers are the holds on all ships; some arrive in batches of two.

Queuing systems with many servers and a variety of arrival and service processes have been extensively studied. Here we use Newell's approximate formulas (*13*) because of their simplicity and generality. They apply to arrival processes that can be approximated by a diffusion process (e.g., with independent increments, compound Poisson).

A similar type of argument can be made for strategy B. Because now two-hatched ships have priority, Q_2 (and not Q_1) is easily predicted. Thus it is now advantageous to express Equation 12 as a function of Q and Q_2 as follows:

$$N_1 = [Q - (Q_2/2)]^+ \tag{14}$$

Newell's Queuing Expressions

For our deterministic service time queuing system (assuming that the customer arrival process follows a stationary process that can be approximated by a diffusion process), Newell's (*13*) approximate expressions simplify. Let λ denote the average customer arrival rate and σ^2, the variance of the number of arrivals in one time unit (this value equals λ for a Poisson process). These two parameters characterize the arrival process. Then the expected number of customers in the system

(being and waiting to be served) is a function of λ, σ^2, and the following dimensionless constant, μ:

$$\mu = (C - \lambda)/\sigma \tag{15}$$

This constant represents how far the system is from being saturated. If it is negative, the system is oversaturated; a steady-state solution does not exist and the queue would grow steadily with time. If μ is close to zero but positive, the system has a steady state in which there usually is a queue; and if μ is greater than 2, queues arise only rarely. The probability that all the servers are busy is:

$$\Pr\{\text{busy}\} \approx \varphi(\mu)/\psi(\mu) \tag{16}$$

where $\psi(*)$ is the function appearing in Equations 2a and 2b. The expected number of customers in the system is

$$E\{\text{no. customers}\} \approx \lambda + \sigma\{\mu\,\Phi(-\mu) + \varphi(\mu)/(2\,\mu\,\psi(\mu))\} \tag{17}$$

which for uncongested systems ($\mu > 2$) can be approximated by

$$E\{\text{no. customers}\} \approx \lambda + \sigma\,\varphi(\mu)$$

Note that as μ approaches infinity, the expected number of customers approaches λ. This is the result that is obtained for the infinite channel queue, and it is a lower bound to the actual number. Figure 4 displays the quantity in braces in Equation 17 and the probability that all servers are busy; both plotted against μ.

Expected Number of Ships and Expected Delay

To calculate $E(Q)$ and $E(Q_1)$ (or $E(Q_2)$) for strategy B, one needs to determine the mean and variance of the pertinent hold arrival process. Let a_1 and a_2 represent the arrival rates for one- and two-hatched ships, respectively, and σ_1 and σ_2 the corresponding variances per unit time. If ships are tramps (they do not follow a schedule), one would expect these variances to be close to the arrival rates. The arrival rates for holds (total, and on one- and two-hatched ships) are $(a_1 + 2\,a_2)$, a_1, and $2a_2$.

One can then use Equations 15 and 17 with these arrival rates and the corresponding variances. These are either $(\sigma_1 + 4\sigma_2)$, σ_1, or $4\sigma_2$. The coefficient 4 appears in these expressions because some holds arrive in batches of two.

Equations 15 and 16 can be used to calculate the probability that the system is busy, p_0, and the probability that the system is busy with all the cranes attending priority holds: p_1 for strategy G (where priority ships have only one hold) and p_2 for strategy B. Clearly, $p_0 > p_1, p_2$.

The expected number of ships at berth is given by the expectation of Equations 13a or 14. These are not linear functions of the Qs, but the equations need only to be rounded up when the system has an odd number of active holds belonging to two-hatched ships. For strategy G this can happen only when there is a queue, and then only about half the time. Thus the expectations of Equations 13a and 13b differ only by $p_{0/2}$; but Equation 13b is linear. Thus:

$$E(N_1) = (E(Q) + E(Q_1) + p_0)/2 \tag{18}$$

For strategy B, an odd number of active holds belonging to two hatched ships can arise only if the system has an odd number of cranes, and then only for about half the time when the system is saturated with these types of ships. Thus:

$$E(N_1) = (2E(Q) - E(Q_2) + p_2)/2 \text{ if } C \text{ is odd} \tag{19a}$$

$$= (2E(Q) - E(Q_2))/2 \text{ if } C \text{ is even} \tag{19b}$$

The average ship time in port is obtained by dividing these expressions by the average ship arrival rate:

$$E\{\text{time in port}\} = E(N_1)/(a_1 + a_2) \tag{20}$$

Example

To illustrate these expressions, assume that $C = 4$ and that ship arrivals are Poisson with $a_1 = 1$ and $a_2 = 0.5$. Then the total hold arrival rate is $(1 + 2(0.5)) = 2$, and the combined σ^2 is $(1 + 4(0.5)) = 3$. Thus, $\mu = 2/\sqrt{3}$, $p_0 = 0.17$, $E(Q) = 2.39$, $E(Q_1) = 1.0$, and $E(Q_2) = 1.07$. The average number of ships with the good strategy is about 1.78 and with the bad strategy, 1.86. The average ship time in port is 1.19 time units for strategy G and 1.24 for strategy B. If cranes were never in short supply, these numbers would be 1. Thus, one can think of the excess (0.19 and 0.24 time units) as the delay caused by crane shortages; switching strategies can reduce this delay by about 25 percent (0.06 time units). If the delays are longer, choosing the best crane allocation strategy should be more important. With 3 cranes, for example, the average number of ships in the system is 2.25 with strategy G and 2.52 with strategy B. The corresponding times in port are 1.5 and 1.68 time units; the difference between the strategies still amounts to about 25 percent of the ship delay, but the difference is now larger in absolute value.

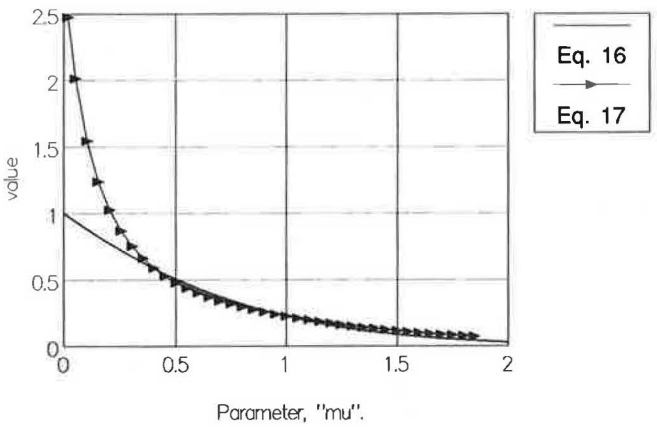

FIGURE 4 Graphs for quick evaluation of Equations 16 and 17.

Discussion

The results in the preceding section assumed that the berth is so long that ships never have to queue for berthing space and that the ship arrival process has independent increments. To check that ships do not have to queue for berthing space, one can calculate the mean and variance of the total number of ships at berth and verify that both are small enough. For type-0 ships, the mean and variance, $E(N_0)$ and var (N_0), can be obtained with the formulas for an infinite server system [see Newell (*13*)]. Earlier, formulas were given for the mean number of type-1 ships, $E(N_1)$, but not for its variance. If Q_1 and Q were independent (they should be positively correlated), Equation 13b would indicate that:

$$\text{var}(N_1) = (\text{var}(Q) + \text{var}(Q_1))/4$$

where var(Q) and var(Q_1) are given by a formula, which is similar to Equation 17 but is not given here. If Q and Q_1 were perfectly correlated, the variance would instead be:

$$\text{var}(N_1) = [(\text{var}(Q))^{1/2} + (\text{var}(Q_1))^{1/2}]^2/4$$

The actual value should be between these limits, which should then be added to var(N_0) to obtain the variance for the total number of ships. Although an exact value is not given here, the calculations may indicate whether the available berth space is likely to suffice; great accuracy is not always needed for this purpose. If some of the ships are liners, the assumption of an arrival process with independent increments does not hold. Some graphical simulations can be done. For example, if all the ships are liners, two cumulative plots of the number of cranes demanded by one- and two- hatched ships versus time (as per their schedules) can be constructed. These graphs will help determine when each hold gets served with algorithm G and the departure time of each ship. This yields the desired information. If only some of the ships are liners (and liners have priority), one can use the preceding process to determine how many free cranes there are on average after serving the liners. If this number does not fluctuate with time very much (the liner schedules could be fairly regular), one could use this average (instead of C) with the expressions in the earlier section to obtain a first estimate of tramp delay. Clearly, there are many situations where the queuing formulas presented in this section do not apply. Nonetheless, the results give an indication of the kind of delay savings that can be attained by efficient crane scheduling.

CONCLUSION

This paper represents an initial attempt at understanding crane operations at ports by means of simple analytical formulas. It provides some approximate expressions for the average number of busy cranes during congested periods (a measure directly related to the maximum terminal throughput) and for ship delay.

The maximum terminal throughput depends on several factors: the berth capacity (in ships), the number of cranes, the amount of work per hold and its variability within and across ships, and the crane operating strategy. The crane operating strategy influences throughput considerably less than the other factors. In all the cases examined, throughput does not change by more than about 10 percent when one switches from an inefficient to an efficient strategy. This indicates that detailed models of crane operations are not needed to obtain rough productivity estimates.

A simple formula, which is proposed for efficient crane operations, was tested against exact expressions for some special cases. The errors were on the order of just a few percent. Although further testing is needed, this suggests that such a formula may be useful for quick response economic and planning purposes, in instances where detailed simulations are not possible.

The paper also illustrates how the maximum productivity expressions can be used for evaluating the effectiveness of various terminal configurations. As an example, it calculates the optimum number of cranes when the berth capacity is fixed and the cost of additional cranes is counterbalanced by corresponding productivity increases.

The paper also studies the impact of crane scheduling on ship delay for a berth that has a finite number of cranes but is ample enough to hold all ships; ships arrive at random so some of them may have to wait for a crane if too many are already at berth. The paper examines idealized situations that can be modeled analytically, and yet are rich enough to be sensitive to the crane allocation strategy. For a given strategy, the expected delay depends on only three parameters: the number of cranes and the average and standard deviation of the number of arrivals in the time that it takes to serve one hold. For the examples studied, representing lightly congested conditions, the expected delay was reduced by about 25 percent when switching from an inefficient to an efficient crane scheduling strategy.

The results in this paper represent only an initial effort toward providing crane usage analytic models. It definitely would be desirable to validate the approximate productivity equations under a wider set of conditions, and to extend the queuing models to situations where berth space is not quite so plentiful.

ACKNOWLEDGMENT

This research was supported in part by a grant from American President Lines to the Institute of Transportation Studies, University of California, Berkeley.

REFERENCES

1. C. H. Plumlee. Optimum Size Seaport. *ASCE Journal of the Waterways and Harbors Division*, Vol. 92, 1966, pp. 1–24.
2. J. D. Mettam. Forecasting Delays to Ships in Port. *Dock and Harbour Authority*, Vol. 47, 1967, pp. 380–382.
3. J. H. Jones and W. R. Blunden. Ship Turn-Around at the Port of Bangkok. *ASCE Journal of the Waterways and Harbors Division*, Vol. 94, 1968, pp. 135–148.
4. S. N. Nicolau. *Berth Planning by Evaluation of Congestion and Cost. ASCE Journal of the Waterways and Harbors Division*, Vol. 95 (WW3), 1969, pp. 419–425.

5. A. J. Miller. Queueing at a Single-Berth Shipping Terminal. *ASCE Journal of the Waterways, Harbors and Coastal Division,* Vol. 97 (WW1), 1971, pp. 43–55.
6. E. Koenisberg and D. A. Meyers. An Interacting Cyclic Queue Model of Fleet Operations. *Logistics and Transportation Review,* Vol. 16, No. 1, 1980, pp. 59–71.
7. M. S. Daskin and C. H. Walton. An Approximate Analytical Model of Supertanker Lightering Operations. *Transportation Research,* Vol. 17B, 1983, pp. 201–219.
8. F. Sabria. *Analysis of Potential Improvements in Port Operations.* Ph.D. thesis. Department of Civil Engineering, University of California, Berkeley, 1986.
9. W. H. Atkins. *Modern Marine Terminal Operations and Management.* Port of Oakland, Oakland, Calif., 1983.
10. C. F. Daganzo. The Crane Scheduling Problem. *Transportation Research,* Vol. 23B, No. 3, June 1989, pp. 159–176.
11. R. Peterkofsky and C. F. Daganzo. A Branch and Bound Method for the Crane Scheduling Problem. *Transportation Research*, in press.
12. C. E. Clark. The Greatest of a Finite Set of Random Variables. *Operations Research,* Vol. 9, 1961, pp. 145–162.
13. G. F. Newell. *Approximate Stochastic Behavior of* n-*Server Service Systems with Large* n. Lecture Notes in Economics and Mathematical Systems 87. Springer-Verlag, New York, 1973.

Guidelines and Computational Results for Vector Processing of Network Assignment Codes on Supercomputers

KYRIACOS C. MOUSKOS AND HANI S. MAHMASSANI

Supercomputers derive their computational performance from faster processors as well as innovations in their architecture. To take advantage of the vector processing capabilities of supercomputers, such as the CRAY X-MP series, it is necessary to modify the code to enhance its vector processing performance. These modifications can range from simple localized recoding of existing mainframe codes to devising new algorithms with the hardware's architecture in mind. In this paper, codes for the solution of two network equilibrium assignment problem formulations (Frank-Wolfe algorithm for the single-class user equilibrium problem and the diagonalization algorithm for multiple user classes with asymmetric interactions) are vectorized and tested on a CRAY X-MP/24 supercomputer. Only local vectorization by limited recoding of existing programs is performed. Guidelines are given for this purpose, and their application to the assignment codes is illustrated. The computational tests performed indicate an improvement in execution time of about 70 to 80 percent of the modified code relative to its unvectorized performance on the CRAY supercomputer. Execution of the vectorized code on the CRAY is about 22 times faster than the execution of the unmodified code on a mainframe computer. The significance of the results for research and practice is also discussed.

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. Two decades of research have resulted in efficient and widely available algorithms for this problem, particularly for the case of a single class of users and no interactions across links. Such programs are routinely executed on microcomputers, though only for moderately sized networks. A review and textbook presentation can be found elsewhere (*1*). For more complicated and realistic cases, especially those involving multiple user classes and asymmetric link interactions (1–5), existing algorithms are much more demanding computationally, especially for large-scale systems. Network assignment procedures are also critical for solving the network design problem, which is an *np*-hard problem that cannot generally be solved optimally using current computational techniques.

Supercomputers offer at least an order of magnitude improvement over conventional mainframes in terms of speed and memory capabilities, and they greatly enhance our ability to solve large problems under more realistic assumptions. Supercomputers derive their high performance not only from inherently faster silicon chips, whose performance is fast approaching its quantum-mechanical limits, but also from their radically different architectures that reflect different degrees of parallelism (*6,7*). The CRAY X-MP series of supercomputers, which is used in the present study, appears to have gained the widest acceptance and accessibility in the American academic community. Its architecture provides a dimension of parallelism by using vector or matrix operations of an algorithm (vectorization). More detailed description of the hardware aspects of the CRAY X-MP that are relevant to applications programmers can be found in papers by Zenios and Mulvey (*7*) and Chen (*8*).

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 that 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 (*7*):

1. Local software vectorization, where the program is reexamined in its parts and subroutines, and redesigned only locally, without program-wide repercussions;
2. Global software vectorization, affecting the whole implementation of the algorithm and the design of the data structures; and
3. Overall algorithm vectorization, where the solution algorithm itself is conceived to take advantage of the machine architecture.

Recently, Zenios and Mulvey (*7*) provided an example of the kinds of local modifications needed to vectorize codes for the solution of nonlinear network programs and reported related computational experience on the CRAY X-MP/24. In addition to illustrating the potential of supercomputers for solving large-scale network optimization problems, their results highlighted the need to modify the codes to achieve better vec-

Department of Civil Engineering, University of Texas at Austin, Austin 78712.

torization. The present paper presents similar information for codes to solve the traffic network equilibrium assignment problem. The principal objective is to assess the computational improvements that can be achieved by local vectorization of network traffic assignment codes, for the single-class and the two-class (with asymmetric interactions) user equilibrium problems. The computational experiments are performed on the CRAY X-MP/24 supercomputer. The results have important implications for practice in terms of the size and complexity of the problems that can be addressed and, more important, for the future development of solution approaches to the network design problem.

The next section presents general guidelines for the local vectorization of FORTRAN codes. Following a brief description of the algorithms, the application of these principles to the single-class user equilibrium assignment codes, and the corresponding computational improvements are described. Results for the two-class problem are presented next, followed by concluding comments.

CODE VECTORIZATION GUIDELINES

To develop vectorizable programs and properly exploit the supercomputer capabilities, some appreciation of the machine's architecture and characteristics is helpful (*7,8*). The CRAY X-MP consists of separate dedicated functional units for vector floating point operations, vector integer operations, and scalar integer operations, respectively. It contains eight vector and eight scalar registers where vectors and scalars, respectively, are held before and after being processed on their way from and back to the memory. 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 Cray FORTRAN (CFT) compiler produces a code that contains vector instructions to drive the high-speed vector and floating point functional units and the eight vector registers in their specified operation. The compiler, to be on the safe side, does not attempt vectorization when it suspects certain dependencies within DO loops, even if the corresponding operations are inherently vectorizable. Another important feature of the CRAY X-MP is the abundance of memory and availability of a very high speed, large solid-state device. As such, many of the techniques typically used to reduce and carefully manage storage in programs developed for mainframe computers may actually inhibit vectorization and degrade performance on the supercomputer.

The first step in the local vectorization of a program initially developed for scalar processing is to perform a time requirements analysis to determine the time-intensive parts of the code. These should then become the primary targets of the recoding effort. A combination of code modifications and compiler directives can then be employed. This process is iterative and can be continued until the programmer is satisfied that no further meaningful improvement can be achieved. Beyond this level, additional improvements would have to be sought by higher-level vectorization, as described earlier.

The primary programming constructs that should be targeted in vectorization efforts are DO loops, where the majority of computer time expense is incurred. As already noted, the CRAY X-MP compiler automatically tries to vectorize the loops where applicable. 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. Statements that should be avoided within the DO loops, according to the UT CHPC User Services Group (*9*), include CALL statements; I/O statements; branches to statements not in the loop; statement numbers with references from outside the loop; references to character variables, arrays, or functional IF statements that may not execute because of the effects of previous IF statements; ELSE IF statements.

The guidelines presented next were followed in vectorizing the network assignment codes, based on suggestions in the publications of the UT CHPC User Services (*9*) and the San Diego Supercomputer Center (*10*):

1. Data dependencies should be eliminated; a loop will not vectorize if, for example, an array is referencing values dependent on computations in lower portions of the array in an incrementing loop. The computations cannot be pipelined.
2. Subscript ambiguities should be eliminated; try to eliminate the dependency of a subscript on a previous calculation by including the operation in the array.
3. In the case of nested loops, the one with the largest range should be assigned as the innermost loop; this would contribute the most to the overall effectiveness of the code because the inner loop is the only one that is vectorized.
4, Conditionals should be eliminated; IF THEN ELSE statements can be replaced by conditional vector merge procedures. Simple IF statements are vectorizable but might inhibit vectorization if their references lead to some of the aforementioned dependencies.
5. The loops should be unrolled to a certain depth, thereby eliminating checking for termination conditions and enforcing chaining and functional unit overlap.
6. Vectorizable loops should be separated from unvectorizable loops—in particular, separate loops that contain CALL statements or I/O statements or any of the statements mentioned previously that are independent of the other computations within the loop.

Before describing the application of these rules to the network assigment codes considered in the study, the basic steps of the algorithms for the single-class user equilibrium and the multiclass user equilibrium with asymmetric costs problems are presented.

REVIEW OF THE NETWORK EQUILIBRIUM ALGORITHMS

Given a known matrix of origin-destination flows, a network of directed links connecting nodes, and link performance functions that describe the dependence of link costs on the corresponding link flows, the single-class user equilibrium algorithm solves for the flows onto the individual links of the network so as to achieve certain equilibrium conditions whereby no driver can improve her travel time by unilaterally switching routes. Exact solution algorithms for the single-class user equilibrium problem are based on Beckman's equivalent mathematical programming formulation (*11*), which can be solved by any of several nonlinear optimization techniques. The most widely used algorithm for its solution is based on the Frank-Wolfe or convex combinations method. This algo-

rithm is well documented, and a detailed presentation can be found in a paper by Sheffi (*1*). A brief overview is presented here.

The algorithm consists of an iterative procedure that, at each iteration, first finds a search direction by solving a linearized approximation, then solves for the optimal move size along that direction. The efficiency of the algorithm derives from the fact that the direction-finding step is equivalent to performing an all-or-nothing assignment. The latter requires the repeated application of a shortest path routine, which is the principal computationally demanding element of the code. An additional source of computational cost is the line search to find the optimal move size along a particular direction and the computation of the relatively complicated nonlinear travel cost (link performance) functions. Letting $t_a(.)$ denote the link performance function for link a, the principal steps of the algorithm can be summarized as follows:

STEP 0: Initialization. Perform all-or-nothing assignment based on the free flow travel times $t_a = t_a(0)$, $\forall a$; This yeilds 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 optimal move size 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 $0 \leq \alpha_n \leq 1$.

STEP 4: Move. 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 is the set of equilibrium link flows); otherwise, set $n = n + 1$ and GO TO STEP 1.

The preceding algorithmic steps are implemented in the computer code as follows. The input of the characteristics of the network, the O-D matrix, link characteristics, and convergence measures, are included in TRAFASN. The initialization STEP 0 takes place in subroutine UE, where all the main steps of the algorithm are controlled. Following the initialization of all the paths to zero flows, subroutine AON is called to initialize the flows on the links to zero. Then the travel times on the links are computed, initially with zero flows. All travel time computations are performed by calling a separate function called COSTFN. Given these travel times, subroutine SHPATH is called, as many times as the number of origins, to identify the shortest path for each O-D pair. Then the flow for each O-D pair is allocated on the links that make up 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 algorithm, respectively. STEP 3 is controlled by subroutine BISECT, where the move size is determined by a line search using the bisection method. This move size is used in updating the flows (STEP 4), followed by the convergence test (STEP 5), calculated in subroutine UE. The output of the program is controlled by subroutine DUMP.

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. One of the most commonly used algorithms for its solution is a direct algorithm called the diagonalization algorithm. A discussion of other approaches is given in the review paper by Friesz (*12*).

In the solution of the two-class user equilibrium problem, a separate copy of the physical network is created for each class of users, as described in Mahmassani et al. (*4*). 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.

At each iteration, the diagonalization algorithm requires the solution of a single-class user equilibrium problem as a subproblem. The latter arises because at the nth iteration, all cross-link effects are fixed at their levels from the $(n - 1)$th iteration, and the cost on any given link is allowed to respond only to its own corresponding flow. This subproblem is solved using the Frank-Wolfe algorithm. Because each iteration of the diagonalization algorithm requires several iterations of the Frank-Wolfe algorithm to solve the diagonalized subproblem, it is more computationally demanding than the single-class algorithm. In addition, because there are as many origin-destination trip matrices as there are classes of users, greater use must be made 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 computer code for the diagonalization algorithm, especially for its streamlined versions (*1*,*5*), does not differ significantly from the single-class code. It is composed of the same subroutines, with some modifications to take into account the division of the traffic into trucks and passenger cars. The previously listed subroutines and functions are renamed in this case, in the respective order in which they were previously mentioned, as UETRDIA, UED, AONUED, TRCOST, SHPUED, BISUED, and DUMPUED. For this reason, the modifications performed to vectorize the single-class code are directly beneficial to the diagonlization code. In the next section, these modifications are described for the single-class code, along with computational results with the vectorized code on two networks used in previous numerical experiments (*4*,*5*).

COMPUTATIONAL RESULTS FOR SINGLE-CLASS UE CODE

The purpose of this section is to illustrate the process followed to vectorize the network assignment code and to document

the improvements achievable by different types of modifications. Most of the testing accompanying the various individual changes was performed on a medium-sized network with 182 O-D pairs, 128 nodes, and 336 links. A similar network was used extensively in earlier experiments with streamlined versions of the diagonalization algorithm (*4*,*5*). A maximum of 500 iterations of the algorithm were allowed before the code was terminated for any test run with this network. All runs were performed on the CRAY X-MP/24 using two available Fortran compilers: the CFT 1.15 and the CFT77 v2.0. The CFT 1.15 is written in CRAY assembly language, the CFT77 in Pascal. The CFT77 has superior scalar performance and implements array syntax (arrays handled as entities) and automatic arrays (storage allocated at run time). In many cases, it has closer FORTRAN syntax error handling and vectorizes some loops that the CFT 1.15 would not. The CFT 1.15 generates scalar and conditional vector loops and chooses between the two at run time, whereas the CFT77 generates only vector code and computes the vector length at run time.

Following the steps described earlier, the performance of the code was first assessed without the vectorizing capabilities of the CFT compilers, and a time analysis was performed to determine the most computationally intensive elements of the program. The results are shown in Table 1. The total time to execute was 15.679 sec, using the CFT 1.15 compiler and 14.99 sec using the other compiler (with vectorization blocked in both cases). This compares with 79 sec on a CYBER CDC 170/750 mainframe or about five times more than the supercomputer without any vectorization.

Next, the program was executed by removing the prohibition of vectorization. The results, shown in Table 2 for both compilers, indicate that the execution times for some of the routines were reduced considerably, though not uniformly. A total reduction of 28 percent was achieved by the vectorized compilation using the CFT 1.15 compiler, and of 32 percent using the other compiler, without any program modification. The shortest path routine vectorized quite well, exhibiting a reduction of about 60 percent. The reductions for functions COSTFN and FINT were much more modest, however, less than 5 percent, thereby pointing our efforts toward seeking to improve them.

TABLE 1 EXECUTION TIMES AND PERCENTAGE OF TOTAL EFFORT FOR EACH SUBROUTINE WHEN VECTORIZATION IS BLOCKED (MAXBLOCK = 1) IN COMPILER FOR THE SINGLE CLASS UE CODE ON NETWORK 1

	CFT 1.15		CFT77 v 2.0	
SUBROUTINE	EXECUTION TIME (Seconds)	(%)	EXECUTION TIME (Seconds)	(%)
AON	2.195	(14.00)	2.559	(17.07)
BISECT	3.065	(19.55)	2.412	(16.09)
COSTFN	5.928	(37.81)	6.115	(40.79)
DUMP	0.222	(1.41)	0.202	(1.35)
FINT	0.507	(3.24)	0.521	(3.47)
SHPATH	3.330	(21.24)	2.827	(18.86)
TRAFASN	0.051	(0.32)	0.050	(0.33)
UE	0.381	(2.43)	0.304	(2.03)
Total Execution Time	15.679	(100)	14.990	(100)

TABLE 2 EXECUTION TIMES (IN SECONDS) AND PERCENTAGE OF TOTAL EFFORT WITH VECTORIZATION USING BOTH CFT COMPILERS FOR THE SINGLE CLASS UE CODE ON NETWORK 1

	CFT 1.15		CFT77 v 2.0	
SUBROUTINE	EXECUTION TIME (Seconds)	(%)	EXECUTION TIME	(%)
AON	1.430	(12.68)	0.917	(8.99)
BISECT	1.813	(16.08)	1.517	(14.87)
COSTFN	5.694	(50.50)	5.785	(56.72)
DUMP	0.215	(1.91)	0.201	(1.97)
FINT	0.485	(4.30)	0.487	(4.78)
SHPATH	1.391	(12.33)	1.055	(10.34)
TRAFASN	0.050	(0.44)	0.048	(0.47)
UE	0.198	(1.75)	0.191	(1.87)
Total Execution Time	11.275	(100)	10.199	(100)

To achieve such improvements, one needs to eliminate data dependencies that inhibit vectorization, as discussed earlier. One strategy in this case is to include the travel cost functions within the BISECT routine instead of repeatedly calling a separate function (COSTFN). Calling functions or subroutines in a loop may inhibit vectorization. This change led to a reduction of 1.274 sec (or 11.3 percent) using the CFT 1.15 compiler. However, it was suspected that a further data dependency existed in the loop for computing the link performance functions that inhibited vectorization. These functions have the following general form:

$$t_a(X_a) = t_{oa}(1 + \beta[X_a/C_a]^{\gamma}),$$

where t_{oa} is the travel time on link a under free flow conditions, C_a is a parameter generally interpreted as the capacity of link a, and β and γ are link-specific parameters. The data dependency in the manner in which the computation of these functions was originally coded arises from the separate calculations of the parameters $A1$ and $B1$, as shown in Figure 1. The expressions for these parameters were therefore included directly in the travel time equation. The foregoing changes are shown in Figure 1 as an example of the kind of local code modifications that can dramatically improve the vector performance of FORTRAN codes. The execution time summary following these changes is reported in Table 3 for both compilers. There was a dramatic drop in execution time to 5.568 sec (or a 51 percent improvement over the unmodified code) for the CFT 1.15 compiler, and to 4.061 (60 percent reduction) for the other, primarily because of a drop in BISECT, confirming the prior existence of a dependency that had inhibited the vectorization of the loop.

Given the preceding results, similar changes were made wherever the functions COSTFN and FINT were called. A further step was to specify the $1/C(N)$ in the travel cost equations a variable $C1(N)$, calculated early in the program, so that $X/C(N)$ was transformed to $X * C1(N)$, which eliminates the repetitive division. A division is computationally more demanding than a multiplication on the CRAY. The execution time summary after these and other minor changes is shown in Table 4 for both compilers. The total execution times dropped by about 57 percent and 68 percent relative to the unmodified but compiler vectorized code for the CFT 1.15 and CFT v2.0 compilers, respectively, and by about 69 percent and 78 percent relative to the unmodified and noncompiler

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(N) + 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))
```

FIGURE 1 Changes to subroutine BISECT to eliminate data dependencies.

TABLE 3 EXECUTION TIME SUMMARY FOLLOWING MODIFICATION OF BISECT AS SHOWN IN FIGURE 1, USING BOTH COMPILERS FOR THE SINGLE CLASS UE CODE ON NETWORK 1

	CFT 1.15		CFT77 v 2.0	
SUBROUTINE	EXECUTION TIME (Seconds)	(%)	EXECUTION TIME	(%)
AON	1.433	(25.73)	0.931	(22.93)
BISECT	1.328	(23.85)	0.654	(16.09)
COSTFN	0.477	(8.56)	0.494	(12.15)
DUMP	0.211	(3.79)	0.203	(4.99)
FINT	0.475	(8.53)	0.488	(12.02)
SHPATH	1.411	(25.34)	1.061	(26.12)
TRAFASN	0.050	(0.90)	0.047	(1.17)
UE	0.184	(3.30)	0.184	(4.52)
Total Execution Time	5.568	(100)	4.061	(100)

TABLE 4 EXECUTION TIME SUMMARY FOLLOWING ALL MODIFICATIONS TO THE SINGLE CLASS UE CODE, USING BOTH COMPILERS, FOR NETWORK 1

	CFT 1.15		CFT77 v 2.0	
SUBROUTINE	EXECUTION TIME (Seconds)	(%)	EXECUTION TIME	(%)
AON	1.357	(27.99)	0.847	(25.66)
BISECT	1.313	(27.07)	0.641	(19.43)
DUMP	0.204	(4.21)	0.206	(6.23)
SHPATH	1.392	(28.71)	1.056	(31.99)
TRAFASN	0.050	(1.04)	0.048	(1.47)
UE	0.533	(10.99)	0.505	(15.23)
Total Execution Time	4.849	(100)	3.301	(100)

vectorized case. The ratio of CDC mainframe to vectorized performance thus becomes of the order of 25 times, compared with about 5 times without any vectorization. This highlights the need for and potential of relatively simple local code modifications to take better advantage of supercomputing capabilities. It is of course possible to improve further on the code's performance; however, the point was reached where the marginal improvements due to additional changes did not justify further effort.

Additional tests of the final vectorized code were performed on a large network of 700 nodes and 1,956 links, confirming the magnitude of the improvement achieved by local vectorization relative to the execution of the unmodified code on the supercomputer and to the CDC mainframe.

COMPUTATIONAL RESULTS FOR DIAGONALIZATION CODE

As explained earlier, the diagonalization program for multiple user classes with asymmetric interactions is very similar to the single-class code. Thus the modifications implemented for the former closely parallel those described in the previous section for the latter. These changes primarily affected the computation of the link performance functions, which are more complicated in the case of multiple user classes, and the BISUED subroutine (the equivalent of the BISECT subroutine for the single-class code). Additional details can be found in the report by Mahmassani et al. (*13*).

The performance of the vectorized diagonalization code was tested on a relatively large network, with two classes of vehicles operating on it. The interactions between vehicle classes are represented in the link performance functions, as described by Mahmassani and Mouskos (*4,5*). The network consists of 364 O-D pairs, 1,400 nodes, and 3,912 links. A total of 25 iterations were allowed before the code was terminated for all test runs. For this network, time analyses were performed for (a) original code with no compiler vectorization, (b) original code with compiler vectorization, and (c) modified code with compiler vectorization. The corresponding execution time analyses are summarized in Tables 5, 6, and 7, respectively, for both CFT compilers.

Comparing the results of Tables 5 and 6, compiler vectorization without code modification leads to an improvement from 23 sec to about 13.5 sec (i.e., a 41.5 percent reduction)

TABLE 5 EXECUTION TIME SUMMARY FOR THE UNMODIFIED DIAGONALIZATION CODE WITH VECTORIZATION BLOCKED

	CFT 1.15		CFT77 v 2.0	
SUBROUTINE	EXECUTION TIME (Seconds)	(%)	EXECUTION TIME (Seconds)	(%)
AONED	1.192	(5.17)	0.949	(4.89)
BISUED	5.726	(24.84)	2.795	(14.40)
DUMPUED	0.434	(1.88)	0.257	(1.32)
SHPUED	10.033	(43.52)	8.429	(43.41)
TRCOST	4.608	(19.99)	5.945	(30.41)
UED	0.245	(1.06)	0.239	(1.23)
UETRDIA	0.815	(3.54)	0.802	(4.13)
Total Execution Time	23.053	(100)	19.414	(100)

TABLE 6 EXECUTION TIME SUMMARY FOR THE UNMODIFIED DIAGONALIZATION CODE WITH VECTORIZATION USING BOTH CFT COMPILERS

	CFT 1.15		CFT77 v 2.0	
SUBROUTINE	EXECUTION TIME (Seconds)	(%)	EXECUTION TIME (Seconds)	(%)
AONED	0.661	(4.90)	0.448	(3.45)
BISUED	2.856	(21.17)	1.983	(15.25)
DUMPUED	0.201	(1.49)	0.136	(1.04)
SHPUED	4.389	(32.54)	3.712	(28.56)
TRCOST	4.440	(32.91)	5.865	(45.13)
UED	0.125	(0.93)	0.084	(0.64)
UETRDIA	0.816	(6.05)	0.770	(5.92)
Total Execution Time	13.489	(100)	12.998	(100)

TABLE 7 EXECUTION TIME SUMMARY FOR THE MODIFIED DIAGONALIZATION CODE WITH COMPILER VECTORIZATION

	CFT 1.15		CFT77 v 2.0	
SUBROUTINE	EXECUTION TIME (Seconds)	(%)	EXECUTION TIME (Seconds)	(%)
AONED	0.423	(6.26)	0.299	(5.18)
BISUED	0.861	(12.74)	0.795	(13.76)
DUMPUED	0.217	(3.21)	0.187	(3.24)
SHPUED	4.326	(63.99)	3.647	(63.11)
UED	0.125	(1.85)	0.084	(1.45)
UETRDIA	0.808	(11.95)	0.766	(13.26)
Total Execution Time	6.759	(100)	5.779	(100)

for the CFT 1.15 compiler and a 33 percent reduction for the other compiler. This time is cut by about half after the code is modified, as shown by Table 7, for a total reduction of about 70 percent, corresponding to a nonvectorized to vectorized improvement ratio in excess of 300 percent, for both compilers. As a reference, the code executed in 126 sec on the CDC mainframe, so the vectorized code on the CRAY performed 22 times better than the unmodified code on the mainframe.

CONCLUDING COMMENTS

The results presented in this paper provide an indication of the magnitude of the reductions in execution time of network assignment codes on the CRAY X-MP/24 supercomputer that can be achieved by the vectorization of the codes, and relative to mainframe computers. For both the single-class user equilibrium and the two-class user equilibrium problem with asymmetric interactions, considerable improvement was achieved following local vectorization by limited modifications to the codes: about 80 percent and 70 percent, 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 within the DO LOOPS. Inserting in line the travel cost functions proved very helpful in both cases. The unmodified codes ran about 5 times faster on the CRAY X-MP without compiler vectorization, and between about 7 and 10 times faster with compiler vectorization, than on the CDC mainframe. However, after the modifications, execution on the CRAY was about 22 times faster than on the mainframe. Of course, generalization of these conclusions requires additional experiments on networks with different configurations and sizes. It is expected that the relative improvement due to the modifications would depend on the extent to which the shortest path routine is called in a particular problem.

It is therefore important to realize that off-the-shelf codes for network analysis originally developed to maximize efficiency on mainframes are not likely to run very efficiently on supercomputers with vector processing capabilities. The results given here demonstrate that relatively simple local modifications can have significant impacts on the vector performance of such codes. The generally applicable guidelines followed in our vectorization of these codes are easy to implement and have been shown to be quite effective.

In this study, no attempt was made to go beyond the local level of code vectorization. It is quite possible that additional improvements can be achieved by using more efficient data structures, or different algorithms, for the overall problem or any of its parts, specifically conceived or selected for their potential for efficient vector performance. Interesting challenges lie ahead along those lines as solution procedures are revised and devised to take advantage of increasingly available innovative hardware. For instance, local modifications in the shortest path routine did not yield significant improvements, suggesting that additional reduction may require more global attempts.

Having established the foregoing results, it is important to ask what their implications might be for research and practice. Should researchers and practitioners attempt to perform all assignment runs on supercomputers? The answer is of course that most everyday applications of traffic assignment models, especially of the fixed-demand single-class variety, will and should continue their migration to microcomputers. The capabilities offered by supercomputers mean that one can address very large-scale problems, and afford greater detail in network representation and, more important, greater realism in the underlying assumptions. For instance, problems with multiple user classes and asymmetric interactions are notoriously demanding computationally; supercomputers offer an attractive computing environment in which to solve such problems and not be discouraged from performing sensitivity analyses. In addition, supercomputer capabilities may lead to breakthroughs in two subjects of current interest to researchers and of great potential practical significance: dynamic assignment problems and the network design problem. Both problems give rise to serious computational hurdles that have considerably slowed progress on their substantive aspects and on their solution in practical applications.

The network design problem belongs to the category of *np*-hard problems. A particular variant of practical interest arises in connection with the selection of truck-related improvements, described by Mahmassani et al. (*4*,*14*), that can be stated as follows: Given a network with known O-D matrices for each category of network users 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. If k improvement options are available for each link, the problem's combinatorial complexity rises to k^n. Because the calculation of the travel costs associated with a particular combination of improvements requires the application of a traffic assignment procedure (to find either a user equilibrium solution or a system optimum solution), improve-

ments in the execution of traffic assignment codes have important implications for the size of practical network design problems that can be solved. The encouraging results obtained in this study allow some optimism toward vectorizing transportation network design codes, of which the network equilibrium assignment is a component, and attempting their execution on the CRAY. Furthermore, it would be useful to go beyond local code vectorization to consider global restructuring of the code to achieve greater levels of computational efficiency.

ACKNOWLEDGMENTS

Principal funding for the study on which this paper is based came from a grant from Cray Research Inc. Computing resources for this work were provided by the University of Texas System Center for High Performance Computing (CHPC). The assistance and cooperation of CHPC staff in the course of this study are appreciated. In particular, the contribution of Spiros Vellas to the vectorization of the codes is gratefully acknowledged. The single-class network equilibrium assignment code used in this study is a modified version of a code initially provided by Fred Mannering, presently at the University of Washington, who modified the program originally supplied by Stella Dafermos at Brown University. The authors of course are solely responsible for the content of this paper.

REFERENCES

1. Y. Sheffi. *Urban Transportation Networks*. Prentice-Hall, Englewood Cliffs, N.J., 1985.
2. S. C. Dafermos. Relaxation Algorithms for the General Asymmetric Traffic Equilibrium Problem. *Transportation Science*, Vol. 16, No. 2, 1982, pp. 231–240.
3. A. B. Nagurney. Computational Comparison of Algorithms for General Traffic Equilibrium Problems with Fixed and Elastic Demands. *Transportation Research*, Vol. 20B, No. 1, 1986, pp. 78–84.
4. H. S. Mahmassani, K. C. Mouskos, and C. M. Walton. Application and Testing of the Diagonalization Algorithm for the Evaluation of Truck-Related Highway Improvements. In *Transportation Research Record 1120*, TRB, National Research Council, Washington, D.C., 1987, pp. 24–32.
5. H. S. Mahmassani and K. C. Mouskos. Some Numerical Results on the Diagonalization Network Assignment Algorithm with Asymmetric Interactions Between Cars and Trucks. *Transportation Research*, Vol. 22B, 1988, pp. 275–290.
6. B. L. Buzbee and D. H. Sharp. Perspectives on Supercomputing. *Science*, Vol. 227, 1985, pp. 591–597.
7. S. A. Zenios and J. M. Mulvey. Nonlinear Network Programming on Vector Computers: A Study on the CRAY X-MP. *Operations Research*, Vol. 34, No. 5, 1986, pp. 667–682.
8. S. S. Chen. Large-Scale and High-Speed Multiprocessor System for Scientific Applications. *High Speed Computation*. NATO ASI Series F, Vol. 7. Springer-Verlag, Berlin, West Germany, 1983.
9. *CRAY FORTRAN Optimazation and Performance Analysis*. Center for High Performance Computing (UT CHPC) User Services, University of Texas at Austin, 1987.
10. *User Guide*. Chapter 12: Optimizing Your FORTRAN Code. San Diego Supercomputer Center, San Diego, Calif., June 1987.
11. M. J. Beckman, C. B. McGuire, and C. B. Winston. *Studies in the Economics of Transportation*. Yale University Press, New Haven, Conn., 1956.
12. T. L. Friesz. Transportation Network Equilibrium, Designed Aggregation: Key Developments and Research Opportunities. *Transportation Research*, Vol. 19A, No. 5/6, 1985, pp. 413–427.
13. H. S. Mahmassani, R. Jayakrishnan, K. C. Mouskos, and R. Herman. *Network Traffic Simulation and Assignment: Supercomputer Applications*. Research Report CRAY-SIM-1988-F. Center for Transportation Research, University of Texas at Austin, 1988.
14. H. S. Mahmassani, C. M. Walton, K. Mouskos, J. J. Massimi, and I. Levinton. *A Methodology for the Assessment of Truck Lane Needs in the Texas Highway Network*. Research Report 356-3F. Center for Transportation Research, University of Texas at Austin, 1985.

Computational Experience with a Simultaneous Transportation Equilibrium Model Under Varying Parameters

K. NABIL A. SAFWAT AND MOHAMAD K. HASAN

Safwat and Magnanti have developed a combined trip generation, trip distribution, modal split, and traffic assignment model that can predict demand and performance levels on large-scale transportation networks simultaneously—that is, a simultaneous transportation equilibrium model (STEM). Safwat and Brademeyer have developed a globally convergent algorithm for predicting equilibrium on the STEM. The objective of this paper is to investigate the relative computational efficiency of the algorithm as a function of demand, performance, and network parameters for two small, sample networks and one large-scale, real-world network. The algorithm was found indeed to be sensitive to the values of several variables and constants of the model. Many of the results were as expected and could be generalized. As the values of demand parameters increase, the algorithm tends to take more iterations, on the average, to arrive at a given accuracy level. Beyond maximum "practically feasible" values, however, the algorithm may require a considerable computational effort to satisfy a given tight level of accuracy. Network configuration may have a considerably greater influence on convergence rate than network size. These results should further encourage application of the STEM approach to large-scale urban transportation studies.

Safwat and Magnanti (*1*) have developed a combined trip generation, trip distribution, modal split, and traffic assignment model that can predict demand and performance levels on large-scale transportation networks simultaneously—that is, a simultaneous transportation equilibrium model (STEM). The model achieves a practical compromise between behavioral and computational aspects of modeling the equilibrium problem. It is formulated as an equivalent convex optimization problem, yet it is behaviorally richer than other models that can be cast as equivalent convex programs. Although the model is not as behaviorally rich as the most general equilibrium models, it has computational advantages. It can be solved with a globally convergent algorithms [see Safwat and Brademeyer (*2*) for proof of convergence of the logit distribution of trips (LDT), algorithm under milder assumptions compared with the strict "norm" conditions required for convergence of existing algorithms for general asymmetric models], that is also computationally efficient for large-scale networks [see Safwat and Walton (*3*) for computational experience with an application of the STEM model to the urban transportation network of Austin, Texas]. It is not clear, however, how the computational efficiency of the LDT algorithm would be influenced by variations in demand, performance, and network characteristics of the STEM model for different applications.

The objective of this paper is to investigate the relative computational efficiency of the LDT algorithm as a function of demand, performance, and network parameters for selected example networks as well as the large-scale Austin network. This sensitivity analysis should provide useful guidelines for future applications of the approach.

In the following section a brief summary of the STEM model and the LDT algorithm is presented. The next section includes the sensitivity analysis procedures, results, and interpretations. The final section contains the summary and major conclusions.

Department of Urban & Regional Planning, Texas A&M University, College Station 77843.

A STEM METHODOLOGY

Following is a brief description of a STEM model and the LDT algorithm that predicts equilibrium on the STEM model by solving an equivalent convex program (ECP). For a detailed description of the methodology, the reader may refer to work of Safwat and Magnanti (*1*). Proof of convergence of the LDT algorithm may be found in work by Safwat and Brademeyer (*2*).

A STEM Model

In this subsection, a STEM model that describes users' travel behavior in response to system's performance on a transportation network is presented as follows:

$$G_i = \alpha S_i + E_i \qquad \text{for all } i \in I \tag{1}$$

$$S_i = \max \{0, \ln \sum_{j \in D_i} \exp (-\theta U_{ij} + A_j)\} \qquad \text{for all } i \in I \tag{2}$$

$$T_{ij} = G_i \exp (-\theta U_{ij} + A_j) / \sum_{k \in D_i} \exp (-\theta U_{ik} + A_k) \qquad \text{for all } ij \in R \tag{3}$$

$$C_p \begin{cases} = U_{ij} \text{ if } H_p > 0 \\ \geq U_{ij} \text{ if } H_p = 0 \end{cases} \qquad \text{for all } p \in P_{ij}, \text{ all } ij \in R \tag{4}$$

$$C_p = \sum_{a \in A} \delta_{ap} C_a(F_a) \qquad \text{for all } p \in P_{ij}, \text{ all } ij \in R \tag{5}$$

In this model, the demand variables are

G_i = the number of trips generated from origin i,
T_{ij} = the number of trips distributed from origin i to destination j,
H_p = the number of trips traveling via path p from any given origin i to any given destination j, and
F_a = the number of trips using link a.

The performance variables are

S_i = an accessibility variable that measures the expected maximum utility of travel on the transport system as perceived from origin i;
U_{ij} = the average minimum "perceived" cost of travel from i to j;
C_p = the average cost of travel via path p from any given i to any given j; and
C_a = the average cost of travel on link a expressed as a function of the number of trips (F_a) on that link.

The rest of the quantities are

E_i = a composite measure of the effect that the socioeconomic variables, which are exogenous to the transport system, have on trip generation from origin i;
A_j = a composite measure of the effect that the socioeconomic variables, which are exogenous to the transportation system, have on trip attraction at destination j;
α = a parameter that measures the additional number of trips that would be generated from any given origin i if the expected maximum utility of travel, as perceived by travelers at i, increased by unity;
θ = a parameter that measures the sensitivity of the utility of travel between any given origin-destination pair ij as a result of changes in the system's performance between that given O-D pair;

$$\delta_{ap} = \begin{cases} 1 & \text{if link } a \text{ belongs to path } p \\ 0 & \text{otherwise;} \end{cases}$$

and the defined sets are

I = set of origins,
R = set of destinations,
P_{ij} = set of simple paths from i to j, and
D_i = set of destinations accessible from origin i.

The basic assumptions of this STEM model may be summarized as follows:

1. Trip generation (G_i) is given by any general function as long as it is linearly dependent on the system's performance through an accessibility measure (S_i) based on the random utility theory of travel behavior (i.e., the expected maximum utility of travel).
2. Trip distribution (T_{ij}) is given by a logit model where each measured utility function includes the average minimum perceived travel cost (U_{ij}) as a linear variable.
3. Modal split and trip assignment are simultaneously user optimized. Notice that the STEM framework allows for the modal split to be given by a logit model or (together with trip assignment) to be system optimized [see Safwat (*4*)].

LDT Algorithm

The LDT algorithm belongs essentially to the class of feasible direction methods. At any given iteration r, the method involves two main steps. The first step determines a direction for improvement (d^r). The second step determines an optimum step size (λ^*) along that direction. The current solution x^r is then updated, that is, $x^{r+1} = x^r + \lambda^* d^r$, and the process is repeated until a convergence criterion is met. Feasible direction algorithms differ mainly in the way feasible directions are determined and may not always converge to the optimum solution.

The feasible direction d^r, in the LDT algorithm, is determined as follows:

Step 1. Update link cost by calculating $C_a^r = C_a(F_a^r)$ for all $a \in A$. Set $i = 1$ in an ordered set of origin I.

Step 2. Find the shortest path tree from i to all $j \in D_i$. Let U_{ij}^r be the cost of the shortest path from i to j.

Step 3. Find $d^r = Y^r - X^r$ where the vector $X^r = (S^r, T^r, F^r)$ and the vector $Y^r = (L^r, Q^r, V^r)$ are given by

$$L_i^r = \max\{0, \ln \sum_{j \in D_i} \exp(-\theta_i U_{ij}^r + A_j)\} \quad \text{for all } i \in I$$

$$Q_{ij}^r = (\alpha_i L_i^r + E_i) \exp(-\theta_i U_{ij} + A_j) / \sum_{k \in D_i} (-\theta_i U_{ik}^r + A_k) \quad ij \in R$$

$$B_p^r = \begin{cases} Q_{ij}^r & \text{if } p = p^* \in P_{ij} \\ 0 & \text{otherwise,} \end{cases} \quad \text{for all } p \in P_{ij},\ ij \in R$$

$$V_a^r = \sum_{ij \in R} \sum_{p \in P_{ij}} \delta_{ap} B_p^r \quad \text{for all } a \in A$$

Then the feasible direction at iteration r is the vector d^r with the following components:

$$d_i^r = L_i^r - S_i^r \quad \text{for all } i \in I$$

$$d_{ij}^r = Q_{ij}^r - T_{ij}^r \quad \text{for all } ij \in R$$

$$d_a^r = V_a^r - F_a^r \quad \text{for all } a \in A$$

Safwat and Brademeyer (*2*) proved that the LDT algorithm is globally convergent under the same mild assumptions as with the STEM model.

SENSITIVITY ANALYSIS PROCEDURES AND RESULTS

Several major factors may influence the convergence rate of the LDT algorithm:

1. Trip generation parameter (α),
2. Minimum trip generation (E_i),
3. Trip distribution parameter (θ),
4. Attractiveness measure (A_{ij}),
5. Link performance function (C_a),
6. Network configuration,
7. Network size,
8. Convergence criterion, and
9. Accuracy level.

It is very clear that the combinations of values for these factors are enormous; hence, we have to be selective and more focused, particularly when initial experimental results revealed that the LDT algorithm is indeed sensitive to the selected values. This required a systematic approach and additional care in the "selection process."

Two small example networks and one large, real-world network were used in the analysis. The first small example network (Network 1) was obtained from work by Nguyen and Dupuis (*5*) and the second (Network 2), from the work of Nagurney (*6*); both were proposed for testing algorithms for the asymmetric traffic assignment problem. Network 1 consists of 19 links, 13 nodes, 4 origin-destination pairs, and 2 origins (see Figure 1); and Network 2 consists of 36 links, 22 nodes, 12 origin-destination pairs, and 4 origins (see Figure 2). Tables 1 and 2 include the "observed" interzonal demand volumes on Networks 1 and 2, respectively. Note that these networks are, however, different from the "original" ones in terms of their demand and link performance functions.

The trip generation parameter E_i was selected as the "observed" trip generation. Two values of the attractiveness measure A_{ij} were tested.

1. A_{ij} equals the natural logarithm of the observed trip distribution from i to j (this is a reasonable estimate that is based on theoretical grounds), and
2. A_{ij} equals five times the value in item 1.

Two link performance functions were considered: linear and the usual BPR (i.e., Bureau of Public Roads) 4th power function. These are

Cost** 1: $C_a = t_{oa} [1 + b (F_a/\text{CAP}_a)]$ and

Cost** 4: $C_a = t_{oa} [1 + b (F_a/\text{CAP}_a)^4]$

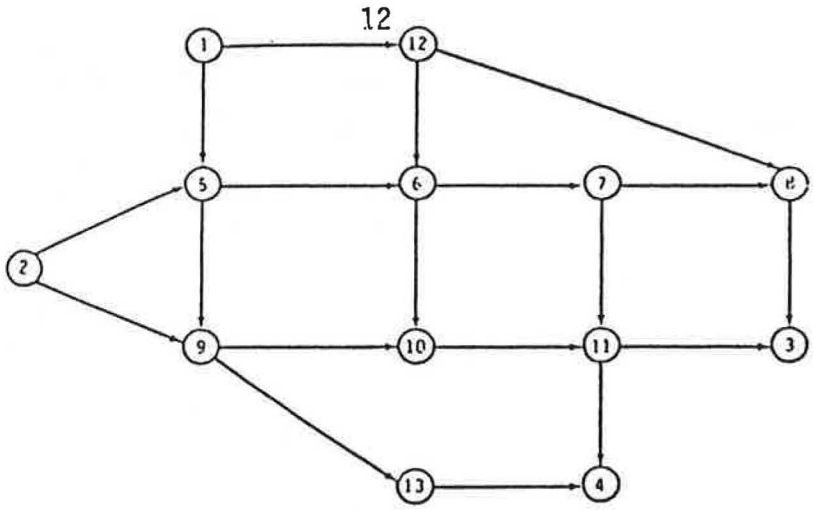

FIGURE 1 Network 1.

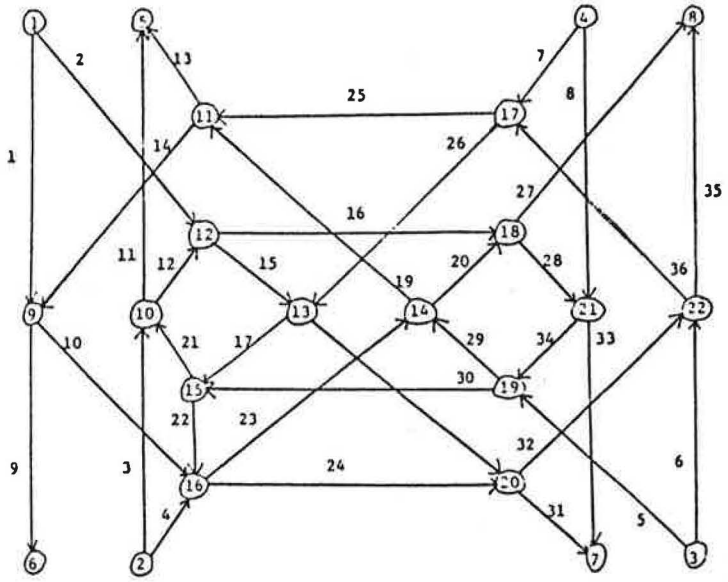

FIGURE 2 Network 2.

TABLE 1 TRIP DISTRIBUTION MATRIX FOR NETWORK 1

	3	4
1	400	800
2	600	200

TABLE 2 TRIP DISTRIBUTION MATRIX FOR NETWORK 2

	5	6	7	8
1	—	235	230	220
2	240	—	235	225
3	230	220	—	235
4	235	225	240	—

where t_{oa} is the free-flow travel time on link a, b is the link congestion parameter, and CAP_a is the practical capacity of link a. These "parameters" were selected at "reasonable" values for all links of a given network such that the average volume-to-capacity ratio on the network at equilibrium is approximately 0.6 (i.e., $t_{oa} = 1$ and $b = 1.15$ for both networks, and $CAP_a = 700$ for Network 1 and 400 for Network 2).

The third network (i.e., the large-scale urban transportation network of Austin, Texas) consists of 7,096 links, 2,137 nodes, 19,213 origin-destination pairs, and 520 origins. The network was used earlier by Safwat and Walton (*3*), and no changes were made in its demand or performance functions. The average volume-to-capacity ratio on the Austin network was approximately 0.2; this is quite conceivable because the network includes existing, committed, and proposed improvements for the year 2000.

The analysis focused on the two major travel demand parameters α and θ. For the two example networks, possible values of these two parameters were considered at two different values of the other one. That is, the values of α varied between 0.001 and 50 while values for θ were set at 0.05 and 0.12, and the values of θ varied between 0.01 and 0.9 while values of α were set at 0.001 and 10. For the Austin network, the values of α varied between 1 and 50 while the value of θ was set at 0.05, and the values of θ ranged between .05 and 0.14 while the value of α was set at 1. The ranges of values were selected to capture "significant" variability in the computational efficiency of the algorithm, as reflected by the number of iterations required to arrive at a prescribed accuracy level based on a given convergence criterion. In some cases, however, there were "practically maximum" values of the parameters beyond which the algorithm could not arrive at the prescribed accuracy level (which was selected to be tight) in thousands of iterations.

Two convergence criteria and two accuracy levels were included in the analysis. Notice that at any iteration in the LDT algorithm the following equation holds true for all $ij \in R$ [see Safwat and Magnanti (*1*)]:

$$T_{ij}^r = \frac{G_i^r \exp(-\theta U_{ij}^r + A_j) \exp(\theta C_{ij}^r)}{\sum_{k \in D_i} (-\theta U_{ik}^r + A_k)}$$

where

$$C_{ij}^r = 1/\theta \left[S_i^r - \ln(\alpha S_i^r + E_i) + \ln T_{ij}^r - A_j\right] + U_{ij}^r \qquad \text{for all } ij \in R$$

It is obvious that at equilibrium $\theta C_{ij}^r = 0$ for all $ij \in R$; hence, two convergence criteria may be specified as follows:

1. Stop whenever $-\varepsilon_1 < \theta C_{ij}^r < +\varepsilon_1$ for all $ij \in R$ or
2. Stop whenever TERMS $= \sqrt{\sum (\theta C_{ij}^r)^2} < \varepsilon_2$

where $\varepsilon_1, \varepsilon_2 > 0$ are small accuracy levels (selected at 0.05 and 0.1 in our analysis) and TERMS is the Total Equilibrium Root Mean Squares error.

The convergence rate of the LDT algorithm was measured in terms of the number of iterations required to achieve a given level of accuracy. This is a proxy measure for the CPU time as it was more or less constant for each iteration. In particular, for the example networks, the CPU time for input and initial solution was 0.09 sec and, per iteration, 0.01 sec on a VAX 8650 minicomputer that was used for analysis. For the Austin network the CPU times were about 190 and 170, respectively.

Because the emphasis in analysis is on the demand parameters α and θ, values of other factors were selected on the basis of their respective influence on the effect of changes in these two parameters on the convergence rate of the algorithm. For instance, to select the appropriate value for the attractiveness measure A_{ij}, Figure 3 shows the effect of θ on the number of iterations to arrive at a prespecified accuracy level (which was selected at $\varepsilon_1 = 0.05$ as determined by the sensitivity analysis procedure itself, as is explained later) for the two different values of the attractiveness measure A_{ij} already

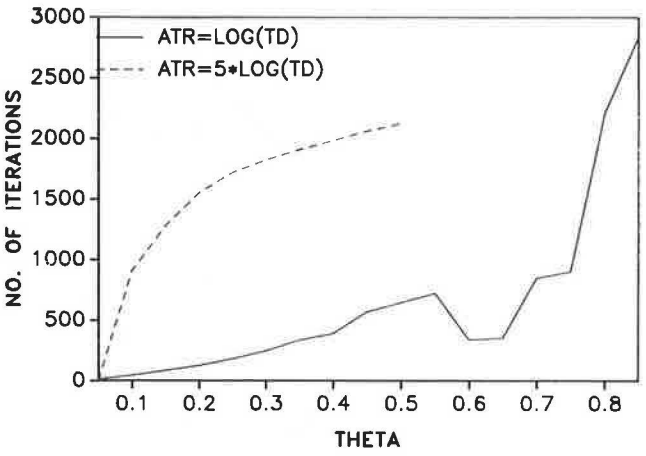

FIGURE 3 Effect of theta on convergence rate (Network 1, Cost 4, alpha = 0.001, epsilon = 0.05).**

indicated. In Figure 3, the parameter α was set at a small value of 0.001 to reduce its influence on results to a minimum; the usual BPR 4th power link cost function (Cost** 4) was selected because it is more realistic than the linear cost function (Cost** 1); and Network 1 was used because its configuration was found to have more influence on the results than Network 2 (see Figure 4).

The graphs in Figure 3 show very clearly that using five times the value of a "reasonable estimate" for A_{ij} caused the number of iterations to increase considerably for all values of θ; the increase becomes more significant as θ increases. On the basis of these results, the attractiveness measure for the remainder of the analysis was set at its more "reasonable" value—that is

A_{ij} = ln ("observed" trips from i to j)

To select a convergence criterion, Figure 5 shows the sensitivity of results with respect to the two proposed criteria. As expected, the second criterion (i.e., TERMS) was always met before the first, "stricter" one, and the patterns of convergence are similar. This is so because $\varepsilon_2 = 0.1$ implies achieving an average value of $\varepsilon_1 = 0.05$, whereas the first criterion allows a maximum value of 0.05 on each individual link. The first criterion was used in the remainder of the analysis to achieve more accurate results. As for the accuracy level, Figure 6 shows the results for two different values of ε_1 (i.e., 0.05 and 0.1). Again the results were as expected in terms of the magnitudes and shapes of the two curves in the figure. The value of 0.05 was used throughout the analysis to obtain more accurate results.

The effect of network configuration and size is shown in Figure 4 for the two example networks. Surprisingly, the "larger" Network 2 always converged considerably more quickly regardless of the change in the parameter θ, whereas the "smaller" Network 1 revealed relatively slower convergence rates, particularly at higher values of θ. It seems that Network 2 has a significantly "simpler" configuration than Network 1 in terms of layout, traffic circulation, and travel demand data (see Figures 1 and 2). These results indicate that network configuration may be a significant factor that could override the effect of network size.

The convergence rates of the algorithm with respect to changes in θ are shown in Figures 7, 8, 9, and 10. For the example networks, Figures 7 and 8 show that regardless of the value of α and network configuration, the number of iterations would on the average increase as θ increases, as would be expected, because larger values of θ imply higher

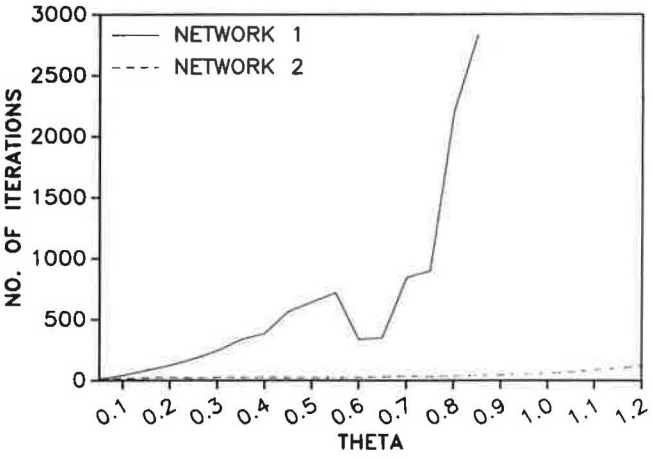

FIGURE 4 Effect of theta on convergence rate (Cost 4, alpha = 0.001, epsilon = 0.05).**

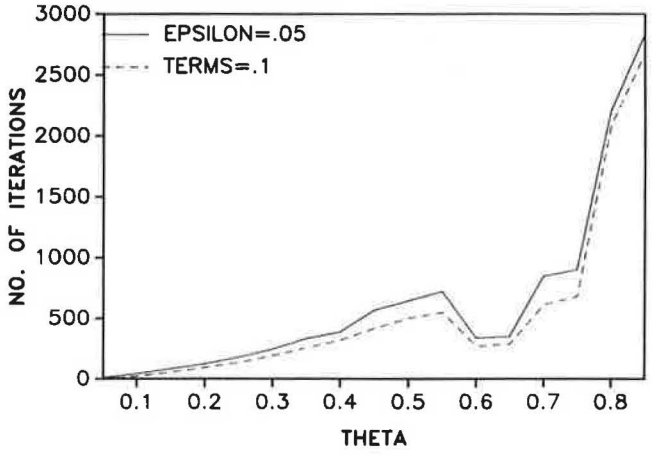

FIGURE 5 Effect of theta on convergence rate for epsilon = 0.05 and TERMS = 0.1 (Network 1, Cost 4, alpha = 0.001).**

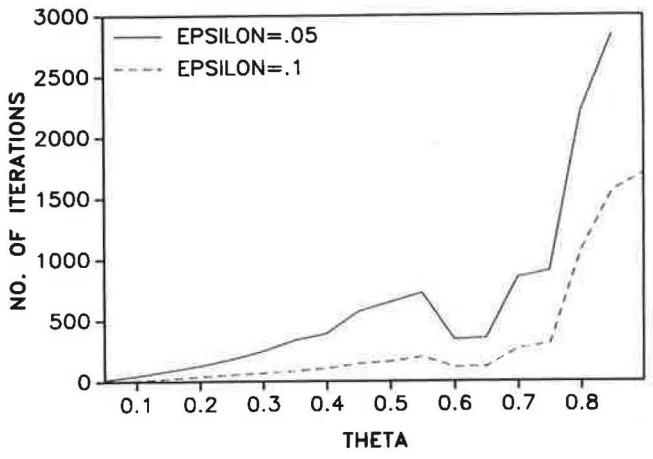

FIGURE 6 Effect of theta on convergence rate for epsilon = 0.05 and epsilon = 0.1 (Network 1, Cost 4, alpha = 0.001).**

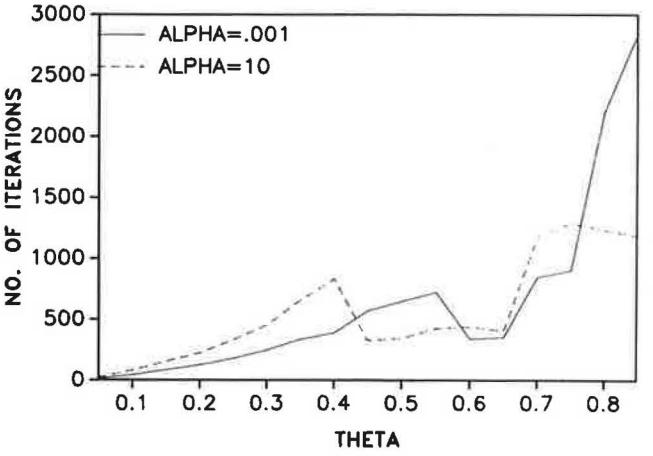

FIGURE 7 Effect of theta on convergence rate (Network 1, Cost 4, epsilon = 0.05).**

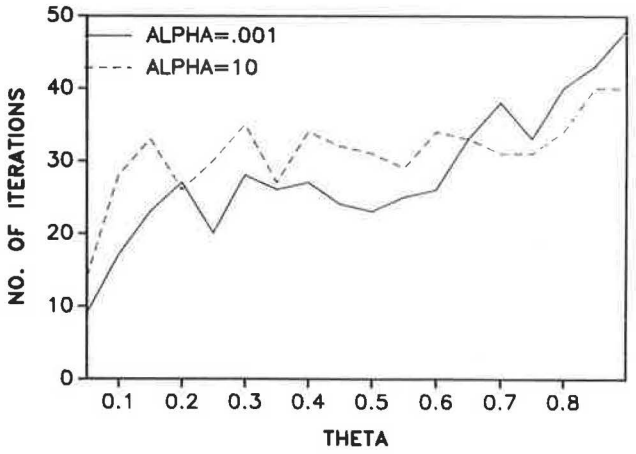

FIGURE 8 Effect of theta on convergence rate (Network 2, Cost 4, epsilon = 0.05).**

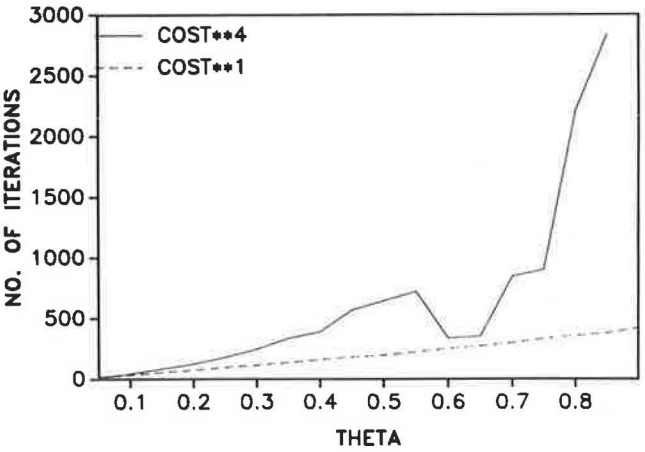

FIGURE 9 Effect of theta on convergence rate (Network 1, alpha = 0.001, epsilon = 0.05).

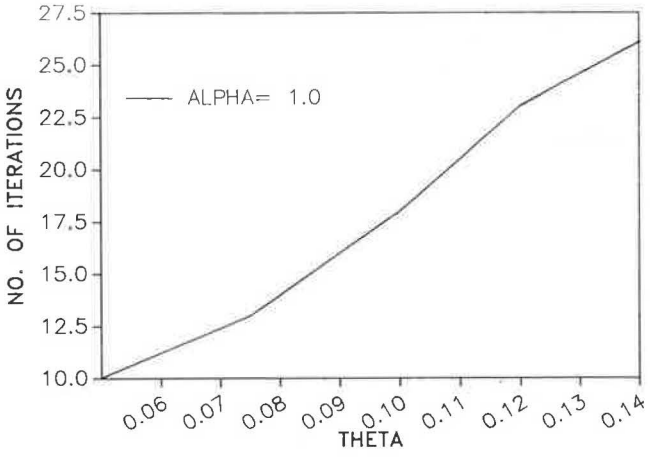

FIGURE 10 Effect of theta on convergence rate (Austin network, epsilon = 0.05).

sensitivity of travel demands to changes in the system's performance. The rate of increase, however, may depend on network configuration and, more important, the shape of the link performance function; as the cost function becomes steeper, Figure 9 shows that, again as expected, the rate of convergence becomes nonlinearly slower.

Figure 10 shows that for the Austin network the results are monotonic and confirm the same trend. The relatively faster convergence on the Austin network may be due to the fact that it is far less congested than the two example networks. Also, network configuration may have been a significant factor that superseded the effect of network size, which does not seem to be a significant factor.

The results for the effect of the demand parameter α on convergence rate are shown in Figures 11 through 14. Figure 11 shows the effect of α for two different values of θ (0.05 and 0.12). It is very clear that the decrease in the value for θ has dampened the effect of α on convergence rate. This behavior is in conformity with our intuition. A similar trend was observed for different cost functions (see Figure 12) and network configuration (see Figures 13 and 14). In Figure 12, then BPR 4th power function adversely influenced the rate of convergence nonlinearly, whereas the linear cost function

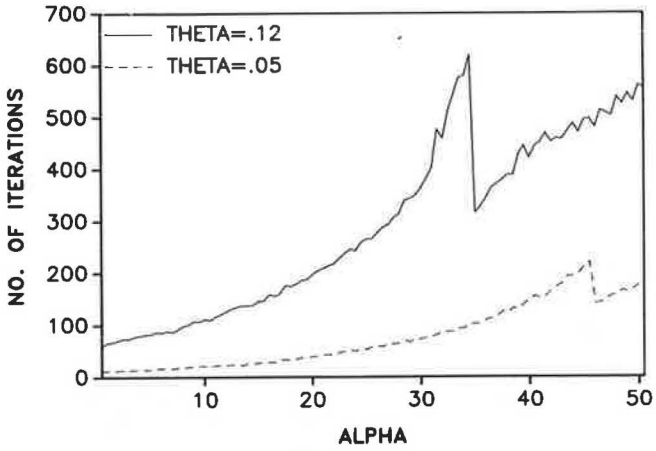

FIGURE 11 Effect of alpha on convergence rate (Network 1, Cost 4, epsilon = 0.05).**

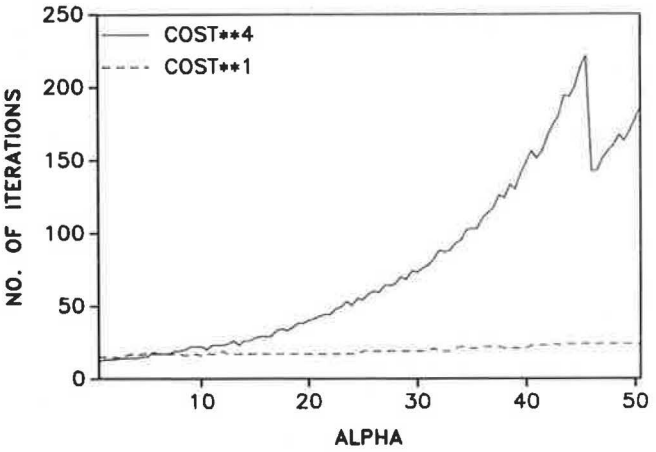

FIGURE 12 Effect of alpha on convergence rate (Network 1, theta = 0.05, epsilon = 0.05).

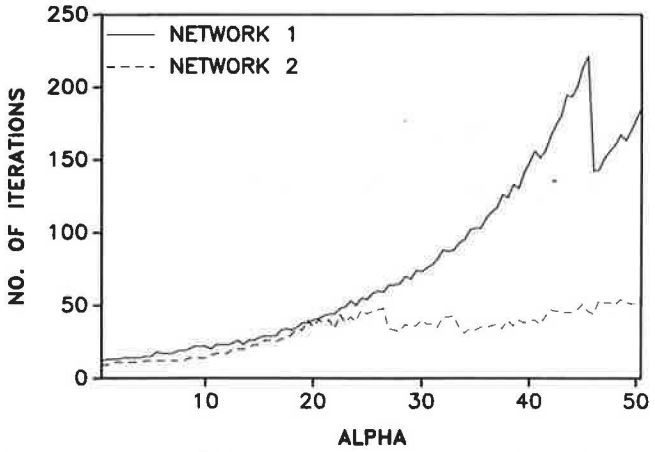

FIGURE 13 Effect of alpha on convergence rate (Cost 4, theta = 0.05, epsilon = 0.05).**

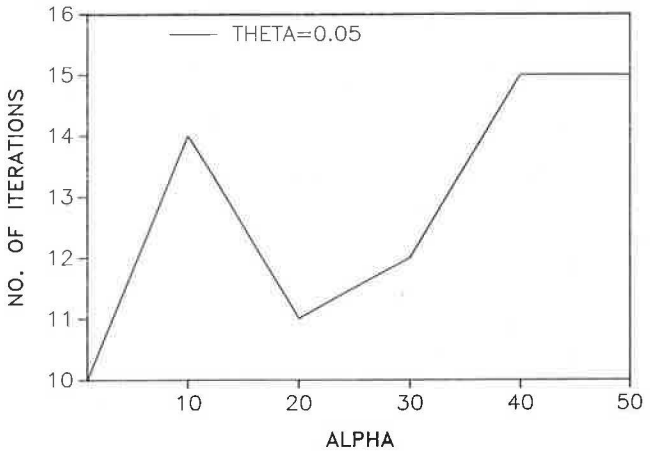

FIGURE 14 Effect of alpha on convergence rate (Austin network, epsilon = 0.05).

had virtually no effect. Figure 13, consistent with Figure 4, shows that configuration of Network 2 appears to be "simpler" than that of Network 1. The results of the Austin network shown in Figure 14 are also consistent with those of the example networks.

SUMMARY AND CONCLUSIONS

The objective of this paper was to investigate the computational efficiency of the LDT algorithm for predicting equilibrium on a simultaneous transportation equilibrium model (STEM) as influenced by several demand and performance parameters of the STEM model as well as network characteristics. The sensitivity analysis considered several major factors, including demand parameters (α, θ, E_i and A_{ij}), performance functions (linear and 4th power), convergence criterion, accuracy level, and network configuration and size. The focus, however, was on the two major demand parameters α, θ.

The main conclusions of this paper may be summarized as follows:

1. The effect of each of the two major parameters α, θ on convergence rate was found to be, as expected, sensitive to the values of the other one in addition to the values of other major variables and constants of the STEM model and the network configuration and size.
2. In general, as the value of the parameter increases, the number of iterations to arrive at a prespecified accuracy level will tend to increase as expected. The effect of θ seems to be more significant than that of α. The combined effect of both parameters is considerably greater than that of the individual parameters separately.
3. There are maximum "practically feasible" values of α, θ beyond which the algorithm may take a considerable computational effort to satisfy a given tight level of accuracy. These maximum values may differ from one application to another. The possible reason for the existence of such practically "upper bounds" on the values of parameters may be related to the flatness of the objective function of the equivalent convex program that is being solved by the LDT algorithm, particularly when the network is less congested.
4. Network configuration may have considerable effects on the convergence rate whereas network size may not.

These results, especially those of the Austin network, further encourage the application of the STEM approach to real-world urban transportation studies. Actual calibration of demand and performance parameters will certainly provide additional insights into the practicality of the proposed method.

ACKNOWLEDGMENTS

The authors would like to express their deep appreciation to Hani Mahmassani, Chairman of TRB Committee on Transportation Supply Analysis, and three anonymous referees for their invaluable comments on an earlier version of this paper.

This work is supported by a research grant from the National Science Foundation.

REFERENCES

1. K. N. A. Safwat and T. L. Magnanti. A Combined Trip Generation, Trip Distribution, Modal Split and Traffic Assignment Model. *Transportation Science*, Vol. 22, No. 1, Feb. 1988, pp. 14–30.
2. K. N. A. Safwat and B. Brademeyer. Proof of Global Convergence of an Efficient Algorithm for Predicting Trip Generation, Trip Distribution, Modal Split and Traffic Assignment Simultaneously on Large-scale Networks. *International Journal of Computer and Mathematics with Applications*, Vol. 16, No. 4, 1988, pp. 269–277.
3. K. N. A. Safwat and C. M. Walton. Computational Experience with an Application of a Simultaneous Transportation Equilibrium Model to Urban Travel in Austin, Texas. *Transportation Research B*, Vol. 22B, No. 6, Dec. 1988, pp. 457–467.
4. K. N. A. Safwat. A Simultaneous Transportation Equilibrium Model: A Unified Consistent Methodology for Transportation Planning. Ph.D. dissertation. Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge, 1982.
5. S. Nguyen and C. Dupuis. An Efficient Method for Computing Traffic Equilibria in Networks with Asymmetric Transportation Costs. *Transportation Science*, Vol. 18, No. 2, 1984, pp. 185–202.
6. A. Nagurney. Comparative Tests of Multimodal Traffic Equilibrium Methods. *Transportation Research B*, Vol. 18B, 1984, pp. 469–485.

Transportation-Network Design Problem: Application of a Hierarchical Search Algorithm

YUPO CHAN, T. STEVEN SHEN, AND NIZAR M. MAHABA

Two variants of a network design problem are solved by application of the tree search method. The first formulation aims to reduce a specified vehicle-minutes of traffic congestion at the least possible budget expenditure, and the second minimizes traffic congestion for a given budget. Both involve system-optimizing traffic assignment models with multipath flows. The solution method consists of network abstraction, tree search, and network disaggregation—collectively referred to as the "hierarchical search algorithm." It is shown that such an algorithm reduces the search space by reducing the number of nodes and links and providing a tighter bound during the tree search. It also groups detailed links according to the function they perform—whether it be access/egress, line-haul, bypass, or internal circulation. However, the algorithm yields only a suboptimal solution, the quality of which is measured by an error function. The metropolitan network of Taipei, Taiwan, Republic of China, is used as a case study to verify some of the algorithmic properties, confirming its role in real-world applications. Finally, the performance of the algorithm, which is based on network abstraction, is favorably compared with a network-extraction network-design model.

Theoretical advances in the last two decades have significantly improved our understanding of network traffic flow. Numerous equilibration models have been put forth under both system-optimizing and user-optimizing assumptions. In spite of advances in computational hardware and software, however, the network-design problem is an NP-hard problem that defies efficient solution techniques (*1*). This is a particularly acute problem in practice, where the size of networks can easily go into hundreds of nodes and links (as in our case study later). No practical solution algorithms exist today to tackle such problems satisfactorily. Difficulties still arise, for example, in solving the network design problem exactly, because it is computationally demanding to solve a user-optimized flow pattern at each step of a system-optimizing search process in the presence of Braess's paradox.

There have been several attempts to address this NP-hard problem by reducing the size of the network, which tends to cut the computational requirement exponentially. For example, network extraction techniques have been practiced for a long, long time to cut down the size of a network design problem. The approach calls for removing "insignificant" nodes and links from a network, leaving only the "important" topological features (*2,3*). However, in spite of carefully designed controlled experiments conducted during the past 20 years (*2,4*), such procedures are still heuristic in nature, often resulting in unpredictable accuracies.

In lieu of extraction, network abstraction has been examined as an alternative to cut down on dimensionality (*5,6*). In this approach, nodes and links are aggregated together to reduce network size. Partial success has been reported in placing error bounds on a limited class of transportation problems, typically variants of the classical Hitchcock transportation/assignment model (*7,8*).

Continuous equilibrium network design formulations have also been proposed. Instead of discrete node-arc representation, improvement variables are continuous (*9*). Some computational gains have been reported, even for user-optimizing traffic assignments.

The preceding aggregation efforts, although improving our ability to solve larger problems, are not quite enough—as pointed out already (*10*). Recent attempts have been made to obtain approximate solutions to both discrete and continuous network design problems. Wong (*1*), for example, revisited Scott's seminal work on discrete network design heuristics and placed worst-case analyses on the computational procedure. Suwansirikul et al. (*11*), on the other hand, suggested a heuristic for finding an approximate solution to the continuous user-optimizing network design model.

To summarize, much work remains to be done in the classic problem of network design. An obvious void is in the abstraction of a realistic transportation network (instead of the Hitchcock assignment problem) and in the placement of error bounds on the corresponding network design problem as we disaggregate back to the original problem.

Here a network design problem is formulated as a hierarchical mathematical program. The original network is abstracted into an aggregate network, thus reducing the number of nodes and links in the process (*5,12*). A tree search is then performed in the aggregate network, which serves as a proxy for the detailed network (*3,13*). The resulting network investment strategy is then disaggregated back to the original, detailed network for implementation (*14*), with a statement on the quality of the approximate solution.

PROBLEM FORMULATION

We state here the first of two network design problems, in which a lowest budget expenditure objective function is formulated as follows:

$$\text{Minimize } B = \sum_i b_i y_i = \bar{b}^T \bar{y} \quad (1)$$

Y. Chan, Department of Operational Sciences, School of Engineering (ENS), Air Force Institute of Technology, Wright-Patterson, Ohio 45433-6583. T. S. Shen, Barton-Aschman Associates, 1133 15th Street, N.W., Washington, D.C. 20005. N. M. Mahaba, 25 Hamadan Street, Giza, Egypt.

where b_t is the budget expenditure for the tth project (performed on link i,j) and

$$y_t = \begin{cases} 1 \text{ if project } t \text{ is implemented} \\ 0 \text{ otherwise} \end{cases} \quad (2)$$

Congestion cannot be allowed to exceed a certain level E_o:

$$E(\bar{y}) \leq E_o \quad (3)$$

where $E(\bar{y})$, in vehicle-minutes, is the result of a system-optimizing traffic assignment. Thus the model is more suited for system-control applications than for evaluation purposes, and y_t's are best interpreted as traffic control strategies to effect an overall improvement of areawide traffic congestion.

By way of definition

$$E(\bar{y}) = \min \sum_{kl} \sum_{ij \in R^{kl}} [f_{ij}(x_{ij}) - \Delta f_{ij}(x_{ij}) y_{ij}] x_{ij}^{kl} \quad (4)$$

is the total congestion after link improvements with the node-arc incidence matrix:

$$\sum_i x_{ip}^{kl} - \sum_j x_{pj}^{kl} = \begin{cases} -v^{kl} & \text{if } p = k \\ v^{kl} & \text{if } p = l \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

and the link-flow "bundling" equation

$$x_{ij} = \sum_{kl} x_{ij}^{kl} \quad (6)$$

where

R^{kl} = the set of links en route from origin k to destination l;
f_{ij} = a convex travel cost function of nonnegative link flow x_{ij}, constrained by a limiting capacity c_{ij};
Δf_{ij} = the improvement in link ij consisting of either travel cost reduction or capacity expansion or both;
x_{ij}^{kl} = nonnegative integer, standing for the flow from origin k to destination l in link (i,j);
y_{ij} = the same as y_t where t is specified for link ij; and
v^{kl} = the origin-destination demand.

Although an alternate formulation will be offered in sequel, recapitulated below are the basic assumptions in our network design models throughout this paper:

1. Travel demand is fixed for each origin-destination (O-D) pair and
2. System-optimizing equilibration procedure is employed.

Through network abstraction (*5,15–17*), we wish to simplify the preceding optimization by reducing the number of links and nodes. One collapses v^{kl} into V^{KL} through zonal aggregation where

$$\sum_{\substack{k \in K \\ l \in L}} v^{kl} = V^{KL} \quad (7)$$

In other words, adjacent zones k are grouped into aggregate zone K and likewise l into L. Finally, the links (i,j)s are aggregated into composite links (I,J)'s with the corresponding travel times, $\hat{F}_{IJ}$, at flow volumes x_{ij} and X_{IJ}, respectively:

$$f_{ij}(x_{ij}) \rightarrow \hat{F}_{IJ}(X_{IJ}) \qquad \forall (i,j) \quad (8)$$

We can now rewrite the preceding network design formulation (Expressions 1 through 8) by replacing every symbol with a capitalized, underlined letter, or Greek letter, converting it from the detailed space to the aggregate space:

Objective function: $B \rightarrow \underline{B}$ with $b_m \rightarrow \beta_m$ (9)

System travel cost function: $E(\bar{y}) \rightarrow \underline{E}(\bar{Y})$ (10)

Traffic flow: $x \rightarrow X$ (11)

The tree search is now carried out in the abstracted network instead of the detailed one, resulting in an optimal solution consisting of link improvements $\{\Delta \hat{F}_{IJ}\}$, rather than $\{\Delta f_{ij}\}$. Finally, a disaggregation method has to be employed to convert each of these link improvements back to the detailed network:

$$\Delta \hat{F}_{IJ} \rightarrow \{\Delta f_{ij}\} \qquad \forall (I,J) \quad (12)$$

The state of the art in network aggregation, particularly in the context of network design, is still quite rudimentary, as alluded to earlier. In the words of Zipkin (*7*):

> [E]ven with computational experience and good software, we do not envision universally appropriate procedures for aggregation. Rather, modellers will have to combine . . . techniques and judgement to suit the problem at hand.

Below, we show a network abstraction procedure that satisfies our specifications outlined by Equations 7 through 12 for a typical transportation network design problem.

NETWORK AGGREGATION ALGORITHM

As mentioned, a network abstraction procedure typically starts with zonal aggregation. In transportation analysis, the grouping of contiguous nodes k and l together is more often than not decided exogenously, mainly by political, geographical, and other considerations. This is distinctly different from scientifically motivated "error-bound" procedures that require that "topologically similar" nodes be aggregated together (*7*)—a process that may require "regrouping" a posteriori for the express purpose of tightening error bounds.

Although zonal aggregation can be accomplished quite readily here via Equation 7, link aggregation needs some explanation. According to Chan (*5,15,16*), link aggregation can be performed in three phases after an initial traffic assignment is made in the detailed network.

Phase 1. Categorization

The links aligned along the minimum paths between each detailed O-D pair, R_{kl}, are categorized first according to groupings, as illustrated in Figure 1. We identify two general classes of flow paths: (a) interzonal flow paths, such as that from node 2 to 7, which goes through aggregate zones I, II,

and III, and (b) intrazonal flow paths, such as that from node 2 to 3 where the entire path is contained in aggregate zone I.

The categorization of links in a path is then broken down into aggregate link classifications according to where the flow path is coming from and where it is leading. For example, in Figure 1, the flow path 2–7 is broken down in the following manner: The first category contains detailed links that carry the egress flow from aggregate zone I to II, which we denote as $E^{I\,II}$. Referring to Figure 1 again, it is found that links (2,1) and (1,3) are the detailed egress links that fall into this category. As a result, the set $E^{I\,II}$ contains (2,1) and (1,3) as elements. The second category contains detailed links that carry the line-haul traffic from aggregate zone I to zone II: $L^{I\,II}$. It corresponds to detailed link (3,5). Similarly, the bypass link from I to III—$^{I}B^{III}$—contains (5,4). The line-haul link $L^{II\,III}$ contains (4,7), and, finally, the *access* link $A^{II\,III}$ is made up of node 7 only. The reader may notice that each column in the summary Table 1 corresponds to an aggregate link F_a^{IJ}, each identified by IJ and aggregate link type a such as line-haul, access, and so on. In the case of a bypass link, an additional superscript specifies the zone in which the line-haul traffic passes through, $^{K}F_a^{IJ}$.

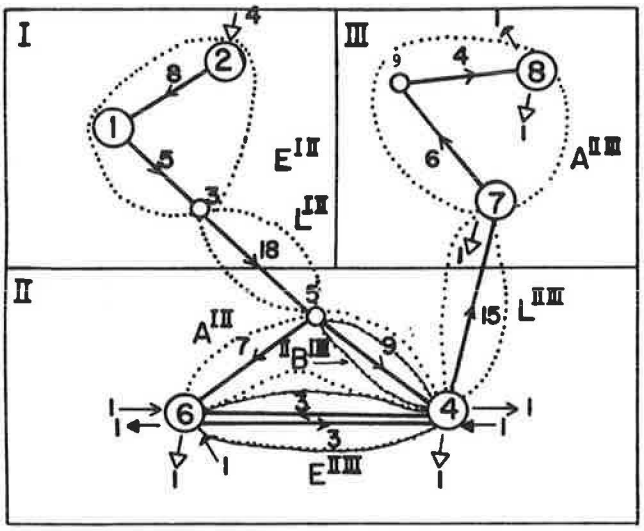

FIGURE 1 Example network.

In the entries of Table 1, the links on each path from origin k to destination l are partitioned into groups $|a_{kl}|$ according to whether they serve a line-haul, access/egress, or intraflow function:

$$|a_{kl}| \leftarrow \{(i,j)\}_{kl} \tag{13}$$

As suggested earlier, an intraflow is the traffic that originates and terminates within an aggregate zone. An example can be found in aggregate zone II between nodes 4 and 6. Unlike interzonal flows, aggregate links that carry intraflows in zone I are simply identified by F_I.

TABLE 1 LINK AGGREGATION

Y_1: $\Delta\hat{L}^{I\,II}=2$ Y_2: $\hat{A}^{I\,II}=4$ Y_3: $\Delta^{II}\hat{B}^{I\,III}=1$

$b_1=2$ $b_2=2.5$ $b_3=1$

O-D Demands	$E^{I\,II}$	$L^{I\,II}$	$A^{I\,II}$	$E^{II\,III}$	$^{II}B^{I\,III}$	$L^{II\,III}$	$A^{II\,III}$
$v^{2\,4}=1$	$f_{2\,1}+f_{1\,3}=13$	$f_{3\,5}=18$	$f_{5\,4}=9$				
$v^{2\,6}=1$	$f_{2\,1}+f_{1\,3}=13$		$f_{5\,6}=7$				
$v^{2\,7}=1$	$f_{2\,1}+f_{1\,3}=13$	$f_{3\,5}=18$			$f_{5\,4}=9$	$f_{4\,7}=15$	$f_7=0$
$v^{2\,8}=1$	$f_{2\,1}+f_{1\,3}=13$	$f_{3\,5}=18$			$f_{5\,4}=9$	$f_{4\,7}=15$	$f_{7\,9}+f_{9\,8}=10$
$v^{6\,8}=1$				$f_{6\,4}=3$		$f_{4\,7}=15$	$f_{7\,9}+f_{9\,8}=10$
Link times $^{K}\hat{F}^{IJ}$	$\hat{E}^{I\,II}=13$	$\hat{L}^{I\,II}=18$	$\hat{A}^{I\,II}=8$	$\hat{E}^{II\,III}=3$	$^{II}\hat{B}^{I\,II}=9$	$\hat{L}^{II\,III}=15$	$\hat{A}^{II\,III}=6\ 2/3$

	I_{II}	i_{II}
$v^{4\,6}=1$	$f_4=0$	$f_{4\,6}=3$
$v^{6\,4}=1$	$f_{6\,4}=3$	$f_4=0$
Link times	$\hat{I}_{II}=1\ 1/2$	$\hat{i}_{II}=1\ 1/2$

Phase 2. Summation

In this phase, the travel times of a serial string of links belonging to the same aggregate link grouping $|a_{kl}|$ are summed. Referencing Figure 1 again as an example, in the $E^{\text{I II}}$ grouping and for flows between 2 and 7, we find links (2,1) and (1,3) with travel times $f_{2\ 1} = 8$ and $f_{1\ 3} = 5$, respectively. The summation phase calls for the addition of these two elements, producing the aggregate link time $\hat{F}^a_{kl} = 13$, which is tabulated in Table 1. In general, detailed link times f_{ij} in the chain from detailed zone k to zone l are summed to become $\hat{F}^a_{kl}$:

$$\hat{F}^a_{kl} = \sum_{ij \in |a_{kl}|} f_{ij} \tag{14}$$

Phase 3. Weighting

Up to the last step of our aggregation procedure, we have in general several aggregate-time quantities $\hat{F}^a_{kl}$ under each aggregate link F^{IJ}_a (or ${}^K F^{IJ}_a$), as illustrated in Table 1. Our final objective, as the reader may expect, is to derive a single link time for each aggregate link F^{IJ}_a. This is accomplished by computing the weighted average (or convex combination) of chains in parallel, where the detailed O-D flows are used as weights (normalized by the total flow volume on the aggregate link). The weighted average becomes the required link travel time $\hat{F}_a$.

Take the example of the F^a_{kl}'s under $A^{\text{II III}}$ in Table 1. There is one trip on $F^a_{2\ 7}$, one trip on $F^a_{2\ 8}$, and one trip on $F^a_{6\ 8}$, resulting in a total of three trips on the aggregate link $A^{\text{II III}}$. Weighting $\hat{F}^a_{2\ 7}$, $\hat{F}^a_{2\ 8}$ and $\hat{F}^a_{6\ 8}$ equally by ⅓, ⅓, and ⅓, respectively, we obtain 6⅔, which is the link time for $A^{\text{II III}}$. In general,

$$\hat{F}_a = \sum_{kl \in |a|} w^a_{kl} \hat{F}^a_{kl}$$

where

$$w^a_{kl} = \frac{v^{kl}}{V_a} \tag{15}$$

and $|a|$ = the set of O-Ds that use aggregate link F^{IJ}_a (or ${}^K F^{IJ}_a$).

Finally, the portion of internal circulation mixed with line-haul traffic in an aggregate zone is to be assigned uniformly to all the access/egress and bypass links of the zone concerned. We denote this type of internal links I_K, where the subscript K denotes the aggregate zone in which the link can be found. The portion of internal circulation flowing on exclusive right-of-way (ROW) links, on the other hand, should be modeled as a separate intra-ROW link. We denote this type of internal links i_K. For example, shown in Table 1 is one internal trip from 4 to 6, using link (4, 6) in i_{II}. Likewise, the other internal trip from 6 to 4 uses link (6, 4) in I_{II}. The link aggregation procedure described above yields $I_{\text{II}} = 1½$ and $i_{\text{II}} = 1½$. For clarity, readers may wish to consult Figure 1 as they go through Table 1.

Aside from computational advantages, the aggregation procedure presented here has obvious functional advantages. The categorization of the detailed links in each aggregate zone into line-haul, access/egress, bypass, and internal circulation groups facilitates transportation analysis, as we can now conveniently refer to a generic class of detailed links by the particular function they perform (*15*). Thus a transportation planner can specify his/her improvement strategy in terms of the function performed by each aggregate link. For example, if egress from zone I to zone II is to be improved, $E^{\text{I II}}$ is specified as a candidate project.

While the example illustrates only single-path assignments, it is clear that the multipath assignment case represents a simple extension. Instead of weights w^a_{kl} defined for each O-D entry v^{kl}, it is now generalized to $w^a_{kl}(q)$, representing the O-D flows $v^{kl}(q)$ "fanning out" into the qth path. Thus the example can be carried forward without loss of generality (*18*).

AGGREGATE TREE SEARCH

Once a network is abstracted, the hierarchical search algorithm finds the optimal network design through a tree search procedure. A branch-and-bound algorithm will be used to search for the best link improvements to the abstracted network. To fix ideas, the algorithm here follows the classic tree search logic for binary variables, using the simple logic of "branching from the minimal lower bound." Other variants of the tree search—such as branch-and-backtrack—can be built upon the basic concepts here (*19*) and will be illustrated in sequel.

Bounding Rule

Every time a branch is made on the search tree, we evaluate the vehicle-minutes of travel resulting from improving a link or several links corresponding to the odd numbered node to the left and even numbered node to the right, respectively (see Figure 2). At each of these nodes one can write the following inequality (or bounds) for system-optimizing traffic assignments that are supposed to be performed:

$$E^c < E^n < E^o \tag{16}$$

where

E^o = vehicle-minutes of travel congestion before link shortening;
E^n = "arithmetic update" on travel congestion assuming flows do not shift paths after link improvement (hence only those that used the link benefit from travel-time reduction); and
E^c = travel congestion if there is a shift of flow paths.

If we use the same set of symbols, but underline them to denote the corresponding bounds in the aggregate search tree, we can write:

$$\underline{E}^c < \underline{E}^n < \underline{E}^o. \tag{17}$$

Now we will trace out the relationship between the $\underline{E}$'s in the aggregate network and the E's in the detailed network. Given that each aggregate network is derived on the basis of a fixed group of detailed links and some outdated detailed

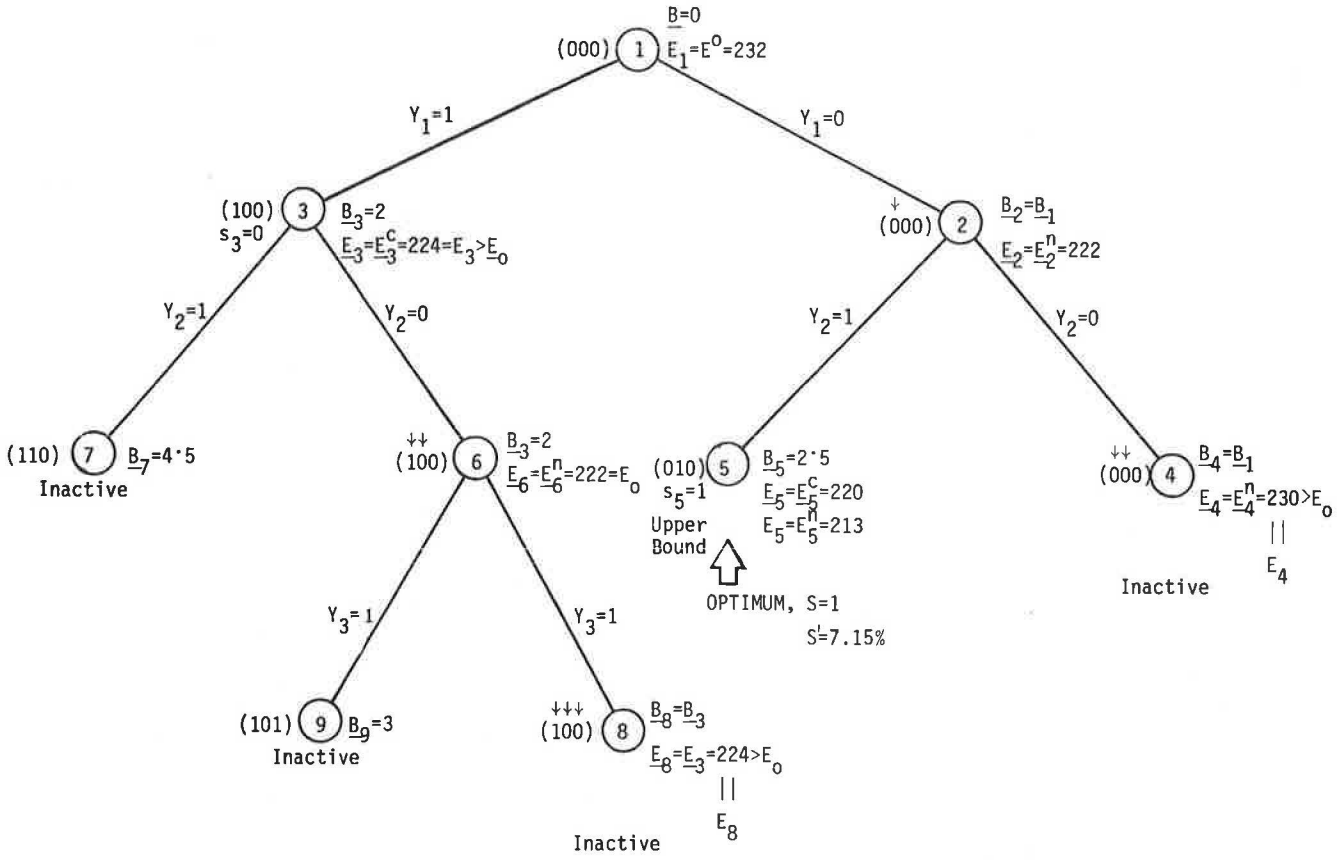

FIGURE 2 Aggregate branch-and-bound tree.

flow pattern further up the tree, one can show that at a particular node of the tree search (proof of this set of inequalities is shown in the Appendix):

$$\underline{E}^n \geq E^n \text{ and} \tag{18}$$

$$\underline{E}^c \geq E^n \tag{19}$$

where $\geq$ reads "is *likely to be* greater than or equal to."

Combining inequalities 16 through 19, the complete bounding relation can be written as

$$E^o = \underline{E}^o > \underline{E}^n > \underline{E}^c \geq E^n > E^c \tag{20}$$

Hence

$$\underline{E}_i \geq E_i \tag{21}$$

which says that, compared with the detailed assignment made at each node i of the aggregate tree, the aggregate system-optimizing travel congestion for the same node cannot be better. In other words, $\underline{E}$ is the upper bound for E.

Applying this bound to extend the "feasibility exclusion rule" of tree search, one can gain some efficiency at the bounded nodes by observing that as long as $\underline{E}^n$ is less than or equal to the maximally tolerable congestion E_o, we do not need to perform any traffic reassignment or "calibrate" the aggregate network against the detailed one. (*Calibration* is defined as performing a traffic assignment on the current detailed network and performing the network abstraction procedure again. More is said about calibration in the following section.) On the other hand, if $\underline{E}^n > E_o$ at node i, we examine the aggregate network and carry out postoptimality procedures to obtain $\underline{E}^c$. Either of the following cases may happen at a node i:

$$\underline{E}_i = \underline{E}_i^c \gtreqless E_o$$

1. If $>$, we calibrate by doing a detailed traffic assignment to obtain E_i.
2. If $\leq$, we keep on branching from this bounded node i.

Exclusion takes place only when the detailed reassignment indicates that $E_i > E_o$.

The exclusion criterion just described would guarantee a comprehensive search space in our aggregate tree search procedure, because we do not prune our tree by exclusion prematurely. A bounded node is excluded only if the detailed assignment indicates that no feasible solution can be obtained no matter what values the "free" variables pick up.

Equivalence Between Search in the Aggregate and Disaggregate Networks

The problem of disaggregation arises when one wants to translate network improvements in the aggregate network back to the detailed. Ideally speaking, one wishes to have the same network design as a detailed analysis, even though the analysis was actually performed in the aggregate search tree.

If, in the process of investigating congestion reduction in the aggregate search tree, we decided on shortening the travel

time on link F_a^{IJ} by $\Delta\hat{F}_a$, it gets translated to a corresponding set of detailed links $\{\Delta f_{ij}\}_a$:

$$\Delta\hat{F}_a \rightarrow \{\Delta f_{ij}\}_a \tag{22}$$

To evaluate Δf_{ij}s, we examine $\{F_{kl}^a\}$, the set of parallel chains defined in Equations 13 and 14. Because of our convex-combination definitions in aggregation, the amount of link shortening in each of the chain F_{kl}^a is the same:

$$\Delta\hat{F}_{kl}^a = \Delta\hat{F}_a \tag{23}$$

A series of link times $[f_{ij}]_{kl}^a$ may be contained in F_{kl}^a, according to Equation 14. This set of linear algebraic equations is therefore to be solved to obtain Δf_{ij}:

$$\sum_{ij \,\varepsilon\, |a_{kl}|} \Delta f_{ij} = \Delta\hat{F}_{kl}^a \qquad \forall a,\ \forall kl \tag{24}$$

It should be pointed out that the solutions of this set of equations are by no means unique. More often than not, they are indeterminate. Mathematical techniques alone are not able to resolve this problem satisfactorily. But irrespective of the arbitrary judgments made in solving Equation 24, the aggregate results of investment decisions are similar, as is shown below.

An essential part of making investment decisions in the aggregate space is to establish the equivalency between the aggregate and detailed networks. In other words, we wish to show the invariance properties both in the static networks and as we perform the search dynamically.

If the shortest paths do not shift, it can be shown that the total vehicle minutes of travel are conserved:

$$\begin{aligned}
\sum_a V_a\hat{F}_a &= \sum_a V_a \frac{\sum_{kl \,\varepsilon\, |a|} v^{kl}\hat{F}_{kl}^a}{V_a} && \text{via Equation 15}\\
&= \sum_a \sum_{kl \,\varepsilon\, |a|} v^{kl}\hat{F}_{kl}^a\\
&= \sum_a \sum_{kl \,\varepsilon\, |a|} v^{kl} \sum_{ij \,\varepsilon\, |a_{kl}|} f_{ij} && \text{via Equation 14}\\
&= \sum_a \sum_{kl \,\varepsilon\, |a|} \sum_{ij \,\varepsilon\, |a_{kl}|} v_{ij}^{kl} f_{ij}\\
&= \sum_{kl} \sum_{ij} v_{ij}^{kl} f_{ij} && (25)
\end{aligned}$$

A similar invariance relationship is maintained in project disaggregation. The equivalence of congestion reduction can be written as

$$\begin{aligned}
V_a\Delta\hat{F}_a &= \sum_{kl \,\varepsilon\, |a|} v^{kl}\,\Delta\hat{F}_a\\
&= \sum_{kl \,\varepsilon\, |a|} v^{kl}\,\Delta\hat{F}_{kl}^a && \text{via Equation 13}\\
&= \sum_{kl \,\varepsilon\, |a|} v^{kl} \sum_{ij \,\varepsilon\, |a_{kl}|} \Delta f_{ij}\\
&= \sum_{kl \,\varepsilon\, |a|} \sum_{ij \,\varepsilon\, |a_{kl}|} v_{ij}^{kl}\,\Delta f_{ij} && (26)
\end{aligned}$$

Although the discussions have been concentrating on disaggregation, the same invariance relationship can be developed for aggregation. Instead of specifying an aggregate link for improvement one can specify a set of detailed links, which are collapsed into the aggregate space prior to tree search.

While one can guarantee $E^o = \underline{E}^o$ by prohibiting path changes in the aggregate network, the invariance properties described above may not be guaranteed as one performs the tree search, which by its very nature causes path shifts. If too many path shifts occur, the weights used in network aggregation change according to Equation 15. This means that the original aggregate network may no longer be an accurate representation of the detailed. Under these circumstances, a calibration of the aggregate network is required, where a detailed traffic assignment is performed upon which a new abstract network is built.

The following heuristics should serve as guidelines for deciding when to calibrate at a node of the aggregate tree:

1. Using the project disaggregation rule, one can define the set of detailed links to be shortened and the amount for each aggregate investment project $\Delta\hat{F}_{IJ}$. Inspect (rather than actually compute) the routing matrices of the detailed network to estimate the number of detailed links that would cause flow shift; call this number e. (The postoptimality procedure of Murchland (*20*), for example, allows for a quick inspection of whether the shortening of a link introduces any flow shift.) If the flow shift is "significant," calibrate; otherwise, proceed with reassignment in the aggregate network.
2. Record the number of odd-numbered nodes in the aggregate tree that have been generated without calibration. This number, b, together with an estimate of the average number of detailed links that could cause path shifts at each odd-numbered node, e, gives a measure of the inaccuracy. If the inaccuracy exceeds a tolerance limit, calibrate; otherwise, proceed.
3. If the aggregate reassignment indicates a shift of flow but the detailed indicates otherwise, record the number of aggregate O-D flow path shifts, d. Together with b and e, d constitutes a measure of the inaccuracy of the aggregate tree search. Again, if the inaccuracy exceeds the tolerance limit, calibrate; otherwise, proceed.

To sum up the preceding three guidelines, we can define an error function at each node k of the aggregate tree as

$$s_k = be + d \tag{27}$$

The quality of a solution can be measured by the cumulative inaccuracies from the root node of the tree. The inaccuracy introduced by skipping calibration at a few odd-numbered nodes where detailed flows have been rerouted cannot be nullified by a calibration performed at the end of these "skips." We have to measure the inaccuracy of the solution by summing all the error functions at all calibration points skipped along the path, from the "root" of the tree to the solution at odd-numbered node K, where the quality of the solution, S, is to be measured.

$$S = \sum_{k \,\varepsilon\, (1 \rightarrow K)} s_k \tag{28}$$

One can normalize S by the total number of detailed links λ and the number of nonzero entries in the aggregate O-D matrix, P:

$$S' = \frac{S}{\lambda + P} \tag{29}$$

Such a percentage gives a rough estimate on the fraction of the total links and O-D pairs that have been affected by the path changes.

Following the same line of argument, the budget expenditure defined for an aggregate project changes as calibration takes place. The reason for this change is that the set of detailed links contained within the aggregate link changes over time as a consequence of path changes. Aggregate project costs, therefore, have to be redetermined at each calibration procedure by summing the costs of the current set of corresponding detailed projects. The point to be noted, however, is that this cost redefinition does not interfere with the additivity (hence, the monotonicity) of the budget function B_k.

Unfortunately, one cannot guarantee optimality in the aggregate search procedure—as the reader may have concluded already. Depending on how often we calibrate (i.e., regrouping the detailed links to a different aggregate link), the result of the optimization routine would conceivably be different. Even when calibration is performed at each node, the final detailed project selection would still depend on the order in which the $0 - 1$ decision variables Y_i are introduced into the tree. The reason for that relationship is that the disaggregation of aggregate link improvement to the detailed level would depend on the current grouping of detailed links to aggregate links, and the grouping is again a function of the order in which Y_i's are introduced.

Branch-and-Bound Algorithm

We are now ready to formalize the tree-search algorithm.

Step 0. Perform network aggregation at the root node $r = 1$. Define for this node $\overline{Y} = (\overline{0})$, and label it with objectives function $\underline{B} = \underline{B}_1 = 0$.

Step 1. *Bound*: Out of the set of active nodes, find the node l with the smallest objective function $\underline{B}_l$ (i.e., the lower bound T). Node l is the bounded node. If $r \neq 1$, set $r = r + 2$.

Step 2. If an active node j has $\underline{E}_j \leq \underline{E}_o$, a reassignment is performed at the detailed level, yielding a new system cost E_j. Compute the error function s_j, which indicates the need for calibration. Then set upper bound $U = \underline{B}_j$. Put node j on an inactive status. All active, feasible nodes with $\underline{B}_i \geq U$ are dominated and declared inactive. If there are no more active nodes, terminate the algorithm. The optimal solution $\underline{B}_j^* = U$ has been found.

Step 3. *Branch*. Branch from the bounded node l, creating node $r + 1$ to the right and $r + 2$ to the left. Set a free variable $Y_k = 0$ on the right branch and $Y_k = 1$ on the left branch. At node $r + 1$, an arithmetic update is performed, with the free variables set to 1. If $\underline{E}_{r+1} > \underline{E}_o$, obtain E_{r+1}. If $E_{r+1} > \underline{E}_o$, node $r + 1$ has been fathomed and termed inactive. Otherwise, set $\underline{B}_r = \underline{B}_l$. At node $r + 2$, compute $\underline{B}_{r+2} = \overline{\beta}^T \overline{Y}_{r+2}$. Go back to Step 1.

NUMERICAL EXAMPLE

Take the base network shown in Figure 1, which has a total congestion cost of E^o 232 vehicle-minutes in both the aggregate and detailed networks. We will define three aggregate projects, which improve the line-haul, access, and bypass functions rendered by the network, respectively:

Project 1—shorten $L^{\mathrm{I\,II}}$	by $\Delta\hat{L}^{\mathrm{I\,II}}$	= 2 minutes
Project 2—shorten $A^{\mathrm{I\,II}}$	by $\Delta\hat{A}^{\mathrm{I\,II}}$	= 4 minutes, and
Project 3—shorten ${}^{\mathrm{II}}B^{\mathrm{I\,III}}$	by $\Delta^{\mathrm{II}}\hat{B}^{\mathrm{I\,III}}$	= 1 minute.

The costs for Projects 1, 2, and 3 are estimated from their detailed counterparts as 2, 5, and 1 units, respectively. The objective of tree search is to find the lowest-budget network improvement plan, with the tolerable congestion level E_o set at 222 vehicle-minutes.

In Figure 2, we perform the branch-and-bound tree-search algorithm step by step, as illustrated:

- *Initialization*: At node 1, $\overline{Y}_1 = (0,0,0)$ at budget level $\underline{B} = 0$, and $\underline{E}_1 = E^o = 232$. We branch to nodes 2 and 3, corresponding to $Y_1 = 0$ and 1, respectively.
- *Nodes 2 and 3*: Node 2 is the bounded node, with a lower budget $\underline{B}_2 = \underline{B}_1 = 0$ (compared with a $\underline{B}_3$ of 2 at node 3). At node 2, variable Y_1 is fixed at zero value, leaving only Y_2 and Y_3 free. An arithmetic update is performed at node 2, with the free variable set to 1: $\underline{E}_2 = \underline{E}_1 - V_2\Delta\hat{F}_2 - V_3\Delta\hat{F}_3 = 232 - 2 \times 4 - 2 \times 1 = 222 = \underline{E}_2^n$. We branch from node 2 to nodes 4 and 5.
- *Nodes 4 and 5*: At the bounded node 4, an arithmetic update is performed: $\underline{E}_4^n = \underline{E}_1 - V_3\Delta\hat{F}_3 = 232 - 2 \times 1 = 230 > E_o$. According to the logic described in the bounding rule portion of the aggregate tree search section, the true state of the system has to be assessed. A reassignment is made on the aggregate network, resulting in $\underline{E}_4 = \underline{E}_4^n$. This prompts another reassignment on the detailed network corresponding to $\Delta f_{5\,4} = 1$, resulting in $E_4 = \underline{E}_4 > E_o$. We exclude further branching from node 4. Between terminal nodes 3 and 5, node 3 is the bounded node because it carries a lower B of 2. A reassignment at node 3 yields $\underline{E}_3 = 224 = E_3$, with an error function $s_3 = b \cdot e = 1 \times 0 = 0$. We branch to nodes 6 and 7.
- *Nodes 6 and 7*: Between 5, 6, and 7, node 6 is the bounded node. An arithmetic update yields $E_6^n = E_3 - V_3\Delta F_3 = 224 - 2 = 222 = E_o$. Further branching is to be performed from this node.
- *Nodes 8 and 9*: Bounded node 8 carries $\underline{E}_8 = 224 = \underline{E}_3 > E_o$. We exclude further branching from this node after checking that $E_8 = \underline{E}_8 = 224 > E_o$.

Out of the three terminal nodes (5, 7 and 9), node 5 is the bounded one because it carries the least budget. The reassigned system cost $\underline{E}_5 = \underline{E}_5^c = 220$. The flows that formerly traversed ${}^{\mathrm{II}}B^{\mathrm{I\,III}}$ are now rerouted using $A^{\mathrm{I\,II}}$ and $E^{\mathrm{II\,III}}$ as bypass links. In the detailed network $E_5 = E_5^n = 213$. The error function $s_5 = b \cdot e + d = 1 \times 0 + 1 = 1$.

- *Termination*: Node 5 is a feasible node with a budget objective function of 2.5; $B_5 = 2.5$ would be an upper bound to help reject further branching from any nodes with B's higher than 2.5. By this rule, nodes 7 and 9 are rejected. The optimum of improving $\hat{L}^{\mathrm{I\,II}}$ [or link (3,5)] by 2 min is found. The error function measures

$$S = \sum_{t \in (1 \to 5)} s_t = 1$$

or a percentage error of $S' = 1/(10 + 4) = 7.15$ percent.

CASE STUDY

We further illustrate the hierarchical search algorithm through a case study of the 1978 network from Taipei metropolitan area, Taiwan, Republic of China, consisting of 49 zones (see Figure 3). To show the versatility of aggregate tree search, the following network design formulation, rather than the one used in the numerical example, is used:

$$\text{minimize } E(\bar{y}) = \min \sum_{kl} \sum_{ij \in R^{kl}} (f_{ij} - \Delta f_{ij} y_{ij})\, x_{ij}^{kl} \tag{30}$$

subject to the node-arc incidence relationship of Equation 5 and the budget constraint

$$\bar{\beta}^T \bar{y} \leq B \tag{31}$$

Essentially, this is the inverse of the formulation in Equations 1 through 6, wherein the best congestion reduction is to be achieved within the budget allowance. In Equation 30, the travel time function f_{ij} assumes the form

$$f_{ij} = f_{ij}^o \left(\frac{1 - h\rho_{ij}}{1 - \rho_{ij}} \right) \tag{32}$$

where ρ_{ij} is the volume/capacity ratio or

$$\rho_{ij} = \frac{x_{ij}}{c_{ij}} \tag{33}$$

Typical capacities (c) range from 1,250 to 3,400 passenger-car-equivalents per hour in the study area, covering local streets through superhighways; h is a calibration constant with typical values of .4, .5, and .6.

The network aggregation algorithm reduces the number of zones from 49 to 14 (see Figure 3), nodes from 155 to 76, and links from 568 to 222. It follows exactly the same procedure as the previous example except that a multipath assignment is used as described earlier.

Three line-haul link-improvement projects were identified.

Project 1: $\Delta\hat{L}^{\text{VII XII}} = 5$ min at a cost of 2 units,

Project 2: $\Delta\hat{L}^{\text{XIV V}} = 5\frac{1}{2}$ min at a cost of 1.5 units, and

Project 3: $\Delta\hat{L}^{\text{XIV I}} = 8\frac{1}{2}$ min at a cost of 2.5 units.

A total budget of four units is imposed.

Branch-and-backtrack is used in the tree search rather than branch-and-bound. For simplicity, the tree search was con-

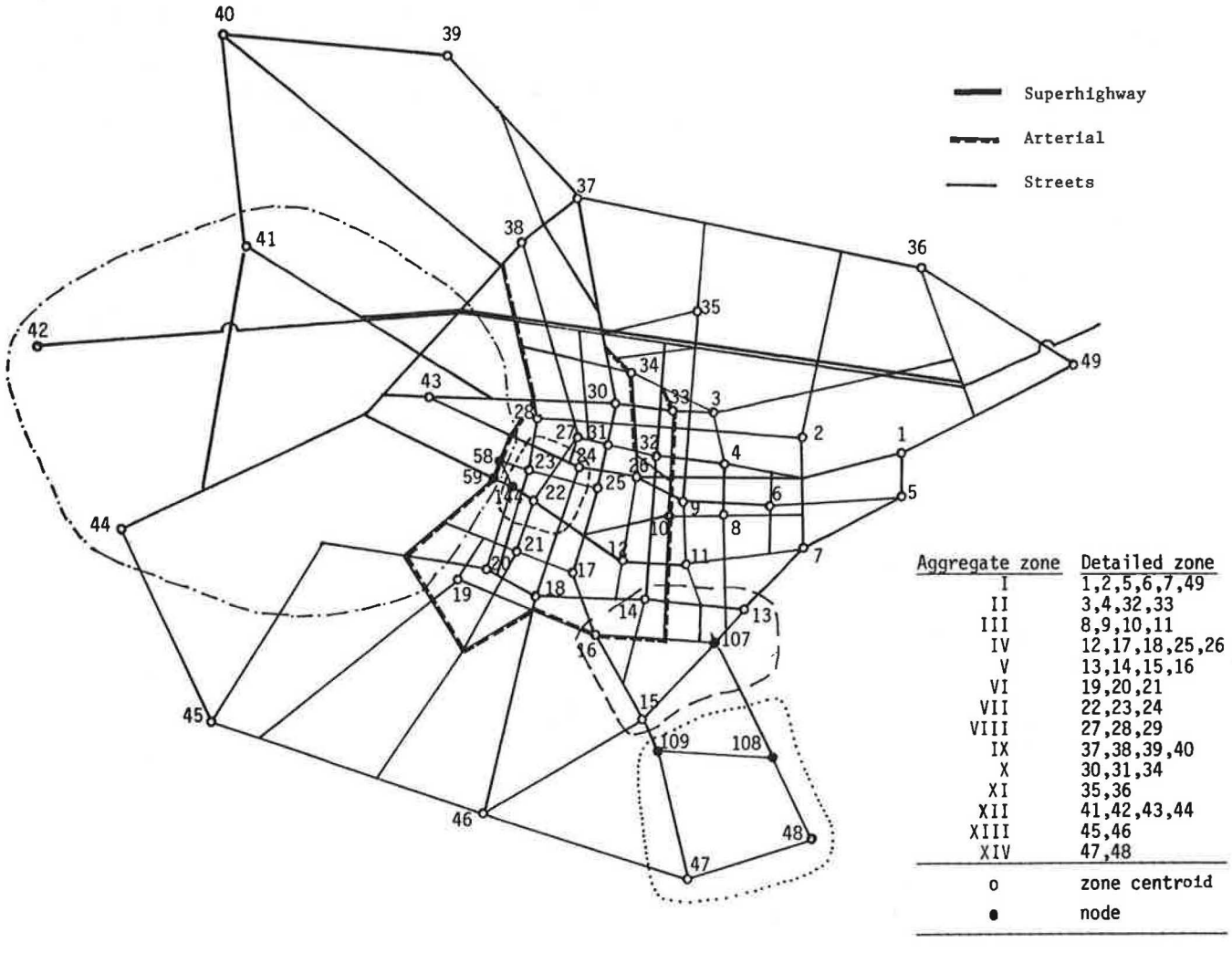

FIGURE 3 Taipei metropolitan network.

ducted entirely in the aggregate network without calibration. Instead of branching from the lowest bound, the branch-and-backtrack method keeps on branching from the latest active node (*19*):

- $r = 1$: Initialize at $\overline{Y}_1 = (0,0,0)$, with $E_1 = 6.6919 \times 10^6$ vehicle-minutes. Branch to nodes 2 and 3, corresponding to $Y_1 = 1$ and 0, respectively. Because $\underline{B}_2 = b_1 = 2 < B$, node 2 is declared active. On the other hand, as $\underline{B}_3 = b_2 + b_3 = 4 \leq B$, node 3 is a feasible solution. The reassignment performed at node 3 yields $\underline{E}_3 = 6.5112 \times 10^6$ vehicle-minutes, which is a better level of system congestion than $\underline{E}_1$. Set $U = \underline{E}_3$ and declare node 3 as inactive. Branch from node 2.
- $r = 3$: Nodes 4 and 5 are obtained corresponding to $Y_2 = 1$ and 0. Because $\underline{B}_4 = b_1 + b_2 = 3.5 < B$, node 4 is active. Similarly, node 5 is also active because $\underline{B}_5 = b_1 + b_3 = 4.5 > B$.
- $r = 5$: Branch from node 5 to nodes 6 and 7. Because $\underline{B}_6 = b_1 + b_3 = 4.5 > B$, declare node 6 to be inactive. On the other hand, $\underline{B}_7 = b_1 = 2 < B$, node 7 is feasible. Traffic assignment shows that $\underline{E}_7 = 6.6386 \times 10^6 > U$. Node 7 is declared inactive; node 4 is the only place where branching can take place.
- $r = 7$: Branch from node 4 to 8 and 9. As $\underline{B}_8 = b_1 + b_2 + b_3 = 6 > B$, node 8 is declared inactive. $\underline{B}_9 = b_1 + b_2 = 4 \leq B$ and $\underline{E}_9 = U = 6.5712 \times 10^6 = \underline{E}^*$. We stop branching because the optimum has been found, corresponding to improving $L^{\text{VII XII}}$ and $L^{\text{XIV V}}$.

For actual implementation purposes, we disaggregate $L^{\text{VII XII}}$ into the parallel chains of (144, 59) and (23, 58) and $L^{\text{XIV V}}$, into parallel chains of (108, 107) and (109, 15). From Equation 14:

$$\Delta\hat{F}^{L}_{22\ 44} = \Delta f_{144\ 59} = 5, \text{ and}$$

$$\Delta\hat{F}^{L}_{23\ 43} = \Delta f_{23\ 58} = 5;$$

yielding the obvious answer of a travel time reduction of 5 min for both detailed links (144,59) and (23,58). Likewise,

$$\Delta\hat{F}^{L}_{48\ 13} = \Delta f_{108\ 107} = 5.5 \text{ and}$$

$$\Delta\hat{F}^{L}_{47\ 15} = \Delta f_{109\ 15} = 5.5;$$

which means a reduction of 5.5 min for both (108,107) and (109,15).

This case study illustrates the practical value of the hierarchical search algorithm in solving realistic size network problems. The city of Taipei, with its population of 2.4 million, was analyzed using a FORTRAN IV traffic assignment code on IBM OS/360. In spite of the use of this relatively outdated machine, the multipath assignment in the detailed network took less than 5 min to cover all the computational trials, yielding an adopted h value of 0.6 in Equation 32. The network design tree search on the aggregate network was also simple enough to be conducted by hand as already shown. It is estimated that an eightfold savings of computational requirement was achieved. Most of this is the result of network abstraction, which reduces the number of links/nodes, hence expediting traffic assignment and tree search in an exponential manner. Because no calibration was performed, however, there is no measurement on the quality of the approximate solution.

COMPARISON OF NETWORK ABSTRACTION WITH NETWORK EXTRACTION

To assess further the afore-described abstraction algorithm in network design applications, a comparison is made with the previously mentioned network extraction algorithm of Haghani and Daskin (*3*). A common square-grid network of 25 nodes and 40 links as shown in Chan (*5*) is analyzed, in which a kth best path traffic assignment (*21*) is performed.

In this controlled experiment, the second network design formulation (Equations 30 and 31) is used, wherein the budget constraint allows for the implementation of only one of the two candidate projects. The first project reduces the cost of two detailed links from 15 to 10 and from 10 to 5 min of travel time, respectively. The second project reduces one single detailed link by three separate amounts: 15 to 9 (Case A), 15 to 8 (Case B), and 15 to 5 (Case C). Results of this experiment are shown in Table 2.

The results show that investment decisions made on both the abstracted network and the extracted network agree to a large extent with those on the detailed network. Two of the three link improvement decisions are the same between the detailed analysis and each aggregate algorithm. The abstraction algorithm performs better in estimating both the total congestion (in vehicle-minutes) and the congestion reduction of Project 1. The extraction algorithm, on the other hand, yields identical congestion reduction as in the detailed analysis for Project 2.

Although this constitutes only a limited experimentation, a few observations can be made.

1. Because of the "invariance" property of abstraction, traffic assignment in the abstracted network appears to yield a total system congestion closer to the detailed network than to the extracted network.
2. Depending on the candidate links for network improvement, the congestion reduction effect may be estimated to be different by the two aggregation schemes. In the case of a

TABLE 2 ABSTRACTION AND EXTRACTION COMPARED

	Detailed Network	Abstracted Network	Extracted Network
Total congestion (vehicle-min)	2,385[a]	2,178[a]	2,125
Congestion reduction for project 1 (vehicle-min)	145	134	130
Congestion reduction for project 2 (vehicle-min)			
Case A	138	94	138
Case B	161	109	161
Case C	230	157	230
Selected project (vehicle-min)			
Case A	1	1	2
Case B	2	1	2
Case C	2	2	2

[a]The two figures are not the same due to path shifts in the abstracted network. By definition, figures are identical without reassignment.

relatively insignificant candidate link, abstraction yields better results. For a major link, on the other hand, extraction algorithm works better. This shows that abstraction is a more "balanced" aggregation algorithm in which the properties of all links—both major and minor—are included in the aggregate network, whereas extraction tends to favor major links to the minor ones, inasmuch as the algorithm explicitly retains major links and discards minor ones.

3. Owing to their different premises, one should not expect the abstraction algorithm to yield identical results as extraction, although there should be some similarity between analyses performed on the aggregated networks and the detailed networks, irrespective of the aggregation method.

CONCLUSION

In this paper, we applied a hierarchical search algorithm to solve network design problems for three spatially abstracted networks. The branch-and-bound and branch-and-backtrack techniques, respectively, were used in the first two formulations of the problem, assuming the objective function of least budget and least travel cost, respectively. These techniques result in a greatly reduced search space, as well as a functional grouping of the detailed links into access/egress, line-haul, and bypass categories. The latter allows network improvement projects to be specified in terms of the corresponding aggregate links that are identified by the function they perform.

The hierarchical search algorithm—combining network abstraction with tree search—was also shown to possess tighter bounding criteria than tree search alone, thus accelerating computational efficiency. Equivalency was established between the aggregate space and the detailed space, including certain invariance properties, such as conservation of vehicle-minutes of travel between the abstracted and detailed networks. The inaccuracy introduced by the approximate optimization procedure was measured by an "error function," showing the solution's percentage divergence from the global optimum. Aside from a numerical example, a case study was taken from the metropolitan area of Taipei, Taiwan, Republic of China, to illustrate the usefulness of the algorithm. For example, the case study shows that computational requirement is reduced by a factor of 8 (*22*).

To the writers' best knowledge, and in their opinion, the hierarchical search algorithm is the first "scientific" attempt to establish equivalency between abstracted and detailed network design decisions. The only required system behavior is monotonicity of two figures of merit as the tree search is being conducted; in our formulation, these are cumulative project expenditure and system congestion level. As such, it is an extremely flexible technique to take care of this class of NP-hard problems. For example, it is conceivable that elastic demands can be tackled as long as monotonicity is maintained in the figures of merit chosen.

The abstracted network is, in essence, a convex combination of the link and node attributes of the original network. This convexity property is exploited to the fullest in establishing the equivalency already mentioned, including the conservation of system travel between the abstracted and detailed networks during aggregation and disaggregation. Should no path shift at all during the search, strict equivalency is guaranteed. The network design methodology becomes approximate when paths do shift as a result of reassignment or network improvement, and the divergence from global optimum in this case is then measured by an error function (in percentages), as alluded to earlier.

The abstraction method was compared with extraction on the third and last network in our research. It was found that both yield investment strategies approximating the detailed network design model, although there is little correlation between the two aggregation approaches. Because of its invariance property, abstraction appears to estimate the total system congestion (in vehicle-minutes) more accurately. There is little distinction between the two, however, in estimating the specific congestion-reduction associated with link improvements.

Although the preceding findings represent modest progress, much work remains to be done in the hierarchical search algorithm. Among them are the following:

1. Further computational experience can be gained by relating the "error function" to the sequence with which link improvement projects are introduced into the search tree and the frequency of calibration. The objective is to find a strategy that minimizes the error in aggregate search.
2. Another set of experiments can be performed to clarify the trade-off between the level of abstraction and the inaccuracy introduced into the solution. The objective is to know the "appropriate" level of aggregation for a given problem.
3. "Branching from the minimal lower bound or the latest active node" is used throughout this paper. Although it served to introduce the hierarchical search algorithm, more efficient tree searches along the line of work by Chan (*19*) can be used to gain even better computational efficiency through the use of double bounds.
4. In spite of the insights gained in our experiments, additional tests can obviously be conducted to compare the performance of an abstraction approach with the extraction approach in network design.

It will be useful eventually to generalize the algorithm to perform user-optimizing network designs, in addition to the system-optimizing ones performed here—although this is by no means a straightforward extension inasmuch as monotonicity properties are no longer guaranteed in the tree search due to Braess's Paradox.

ACKNOWLEDGMENTS

The authors acknowledge the support of the U.S. Department of Transportation, U.S. Department of Defense, General Motors, and the Chinese Technical-Service Council. They are grateful to the many colleagues and friends, including M. L. Manheim, J. H. Stafford, T. L. Loung, and many others at the Air Force Institute of Technology, Massachusetts Institute of Technology, and National Taiwan University, who were of aid. T. S. Shen's contribution toward this paper was performed while he was working in Tapei, Taiwan (*22*). Naturally, the individual authors, rather than their affiliations, are solely responsible for the statements made here.

APPENDIX: Proof that the System Cost in an Aggregate Search, $\underline{E}$, is the Upper Bound for the Corresponding Statistic, E, in the Detailed Space

There are two possible consequences of shortening an aggregate link at node k of the tree search: either a path shift occurs or it does not—that is

$$\underline{E}_k = \underline{E}_k^c \text{ or}$$

$$\underline{E}_k = \underline{E}_k^n.$$

To show the inequality between $\underline{E}_k$ and E_k, one has four possible pairwise relationships to consider.

1. $\underline{E}^n - E^n$;
2. $\underline{E}^n - E^c$;
3. $\underline{E}^c - E^n$; and
4. $\underline{E}^c - E^c$.

Our objective is to show that in all four cases, $\underline{E} \geq E$.

Case 1. When there are no path shifts, shortening the aggregate link will, by virtue of Equation 26, yield the same vehicle-minute reduction as shortening the corresponding set of detailed links; hence $\underline{E}^n = E^n$.

Case 2. This follows from the result of the first case. Because $\underline{E}^n = E^n$ and $E^n > E^c$ from Equation 16, it follows that $\underline{E}^n > E^c$.

Case 3. A detailed link may get categorized under more than one aggregate link, as can be seen in Equation 13 and in Table 1 for $f_{6\ 4}$. Disaggregation of an aggregate link improvement into the detailed network at a node of the aggregate tree means that each improved detailed link (i, j) benefits diverse O-Ds (including intra flows) that use (i, j) according to Equation 30, even though no path shifts occur by definition. For the aggregate link, on the other hand, functional restrictions are placed in the flow paths given that each aggregate link is derived on the basis of some outdated flow pattern further up the tree following Equations 13 through 15. Even though paths shift in the abstracted network owing to the link improvement, the limited amount of flexibility in the outdated network does not allow as great a vehicle-minutes of travel reduction as the corresponding arithmetic update in the detailed network where each and every unit of reduction is accounted for. This means that the aggregate travel congestion is likely to be greater than or equal to that of the detailed after link improvement as illustrated in node 5 of Figure 2.

$$\underline{E}^c \geq E^n.$$

Case 4. This case follows when one combines the result of the preceding finding with Equation 16 (or $E^n > E^c$). Therefore

$$\underline{E}^c > E^c.$$

Conclusion: In all cases, $\underline{E} \geq E$.

REFERENCES

1. R. Wong. Worst-Case Analysis of Network Design Problem Heuristics. *SIAM Journal*, Vol. 1, No. 1, 1980.
2. Y. Chan, K. Follansbee, M. Manheim, and J. Mumford. Aggregation in Transportation Networks: An Application of Hierarchical Structure. Research Report R68-47. Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge, 1968.
3. A. Haghani and M. Daskin. Network-Design Application of an Extraction Algorithm for Network Aggregation. In *Transportation Research Record 944*, TRB, National Research Council, Washington, D.C., 1984, pp. 37–60.
4. P. Bovy and G. Jansen. Network Aggregation Effects upon Equilibrium Assignment Outcomes: An Empirical Investigation. *Transportation Science*, Vol. 17, No. 3, 1983, pp. 240–261.
5. Y. Chan. A Method to Simplify Network Representation in Transportation Planning. *Transportation Research*, Vol. 10, 1976, pp. 179–191.
6. R. Eash, K. Chou, Y. Lee, and D. Boyce. Equilibrium Traffic Assignment on an Aggregated Highway Network for Sketch Planning. In *Transportation Research Record 944*, TRB, National Research Council, Washington, D.C., 1984, pp. 30–37.
7. P. Zipkin. Bounds for Aggregating Nodes in Network Problems. *Mathematical Programming*, Vol. 19, 1980, pp. 155–177.
8. J. R. Evans. The Multi-Commodity Assignment Problem—A Network Aggregation Heuristic. *Computers and Mathematics with Applications*, Vol. 7, No. 2, 1981, pp. 187–194.
9. M. Abdulaal and L. LeBlanc. Continuous Equilibrium Network Design Models. *Transportation Research*, Vol. 13B, 1979, pp. 19–32.
10. T. Friesz. Transportation Network Equilibrium, Design and Aggregation: Key Developments and Research Opportunities. *Transportation Research A*, Vol. 19A, No. 516, 1985, pp. 413–427.
11. C. Suwansirikul, T. Friesz, and R. Tobin. Equilibrium Decomposed Optimization: A Heuristic for the Continuous Equilibrium Network-Design Problem. *Transportation Science*, Vol. 21, No. 4, 1987.
12. R. Barton and D. Hearn. Network Aggregation in Transportation Planning. U.S. Department of Transportation, Mathematica, Princeton, N.J., 1979.
13. Y. Chan. Optimal Travel Time Reduction in a Transport Network: An Application of Network Aggregation and Branch-and-Bound Techniques. Research Report R69-39. Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge, (Clearinghouse AD 693-095), 1969.
14. G. Dantzig, S. Maier, and Z. Lansdowne. The Application of Decomposition to Transportation Network Analysis. Report DOT-TSC-OST-76-26. U.S. Department of Transportation, Washington, D.C., 1978.
15. Y. Chan. A Comparative Analysis of Network Aggregation Approaches. Presented at the National Joint Meeting of the Operations Research Society of America and the Institute of Management Science, San Francisco, 1977.
16. F. Ou and Y. Chan. An Aggregate Representation of Areawide Urban Networks. *Proc., 11th Annual Pittsburgh Conference on Modelling and Simulation*, University of Pittsburgh, 1980, pp. 1115–1119.
17. Y. Chan and F. Ou. Equilibration Procedure to Forecast Areawide Travel. *Transportation Engineering*, Vol. 112, No. 6, 1986, pp. 557–575.
18. N. M. Mahaba. *Network Aggregation*. Paper for OPER 7.67—Network and Combinatorial Optimization, Department of Operational Sciences, Air Force Institute of Technology, Wright-Patterson, Ohio, 1988.
19. Y. Chan. The Transportation Network Investment Problem: A Synthesis of Tree-Search Solution Algorithms. *Transportation Research Record 1074*, TRB, National Research Council, Washington, D.C., 1986, pp. 32–40.
20. J. D. Murchland. A Fixed Matrix Method for All Shortest Distances in a Directed Graph and for the Inverse Problem. Doctoral thesis. University of Karlsruhe, West Germany, 1970.
21. M. Pollack. Solutions of the kth Best Route—A Review. *Journal of Mathematical Analysis and Applications*, Vol. 3, 1961, p. 547.
22. T. S. Shen. Application of Traffic Assignment and Network Aggregation in an Urban Area. Master's thesis. National Taiwan University, Taipei, Taiwan, Republic of China, 1981.

Efficient Algorithm for Locating a New Transportation Facility in a Network

HUEL-SHEN TSAY AND LIANG-TAY LIN

The single-location problem is to locate a new transportation facility in a network that can serve all customers at the minimum distance or cost. There are four types of single-location problems. The absolute 1-center problem is considered in this paper. By definition, in that problem, the customers are on any vertex and the center may be a vertex or a point on an edge. There are two previous methods for finding the absolute 1-center: (a) the Hakimi method (1965) and (b) the Minieka method (1981). They considered all possible links of a network to determine the best candidate point. Later, Larson and Odoni proposed a shortcut to reduce the number of links needed for calculation. In this paper, a new shortcut with a stricter bound is first proposed to find the absolute 1-center directly. The Larson and Odoni shortcut is then introduced and integrated with the Minieka method to form a combined method. Finally, a new method is developed to find the absolute 1-center based on a spanning tree that is obtained from that of the vertex to all shortest distances. The number of iterations needed to perform the analysis is in proportion to the number of vertices instead of edges for any given network. To make a consistent comparison, four different methods have been programmed and tested with several networks. The results show that the new method or the new shortcut is fast and powerful in finding the absolute 1-center location. They provide the same solutions and belong to polynomial time-bounded algorithms. Therefore, we recommend use of the new method or shortcut for locating a new facility if the absolute 1-center problem is considered in a network.

In selecting the optimal facility, location plays a vital role in the fields of transportation, communications, and distribution management. Applications may include transit stops, fire stations, warehouses or plant locations, post offices, schools, and public buildings. A major concern in location models is to find the optimal placement of facilities on a network so the cost of locating, operating, and providing service is minimized. Here, the cost of serving customers can be defined as the cost incurred between customers and the assigned depot; it refers to the transportation cost that is primarily due to the distance traveled to and from the depot location. Therefore, the back-and-forth distance between two nodes is an important component in determining the location of new facilities.

Generally speaking, network location research can be categorized into two types: single-depot location and multiple-depot locations. The single-depot location problem considers locating only one depot in the network, either to minimize loss or to maximize benefit or to provide good service to customers. This facility and its customers may be located at the vertex (node) or anywhere along two vertices. The multiple-depot location problem, on the other hand, finds locations for more than one depot to serve all customers with an objective of minimizing total related cost, minimizing the maximum travel distance, or providing the best service.

Graduate School of Transportation and Communication Management Science, National Cheng Kung University, Taiwan, Republic of China.

Because locating a new transportation facility in a given network is our main consideration, it is necessary to know the differences between various types of single-location problems. There are four major types of single-location problems shown in Table 1. From Table 1, the vertex also represents nodes, and each link or edge has an infinite number of possible point locations. The vertex 1-center and general 1-center location problems have been solved and programmed through efficient methods (*1,2*). These are all polynomial, time-bounded algorithms.

The absolute 1-center problem is defined as a point located such that the maximum distance from this facility to any node is minimized. This new location can be anywhere on a link (edge) or at a node (vertex). Basically, it is a problem of one point serving multiple nodes. One application of the absolute 1-center problem, for example, locates a fire station in a rural community in a manner that minimizes the maximum response time from the station to any farmhouse. It was first presented by Hakimi (*3,4*). The literature on network location problems has grown rapidly since the appearance of Hakimi's paper (*5*). The Hakimi method, for each link, constructs upper envelopes continuously to compute the intersecting points from all nodes in the network. From all feasible intersecting points, we choose the best local minimum for the corresponding link. Once all links have been examined, the best among all such local minima is selected as the absolute 1-center of a given network. Its solution is more difficult and complex than that of either the vertex 1-center or general 1-center problem. In this paper, four methods for solving the absolute 1-center problem are extensively discussed and compared.

The general absolute 1-center location problem is, among four types of single-location problems, the most difficult to solve. This is a problem of one point serving an infinite number of customer points on each link. Recently, some algorithms have been developed and proved to be effective (*1,2,6*). Because the absolute 1-center is our focus, the general absolute 1-center location problem is not discussed here.

LARSON AND ODONI SHORTCUT

Because the Hakimi method requires the examination of each link before the best absolute center in a network is chosen, the number of calculations grows rapidly and sometimes becomes unacceptably large as the number of links increases.

TABLE 1 FOUR TYPES OF SINGLE LOCATION PROBLEMS

Type	Facility Location	Customer Locations
Vertex 1-center	Vertex	Vertex
General 1-center	Vertex	Link*
Absolute 1-center	Link*	Vertex
General Absolute 1-center	Link*	Link*

* Represents infinite possible points located on each link or edge.

Some links, in fact, cannot further improve the optimal solutions. Larson and Odoni proposed a shortcut to reduce the computational effort required to obtain the absolute 1-center (7). That shortcut takes advantage of the fact that it is simple to find the optimal solution of a vertex 1-center problem in a given network. This solution is then treated as the upper bound value to identify those links that actually cannot improve the final result. This shortcut is represented by the following equation:

$$\frac{m(r,a) + m(s,b) - l(r,s)}{2} < m(i^*) \tag{1}$$

where

$m(r,a)$ = the distance required for node r to serve the farthest node a in the network;
$m(s,b)$ = the distance required for node s to serve the farthest node b in the network;
$l(r,s)$ = link distance between nodes r and s; and
$m(i^*)$ = the optimal solution of vertex 1-center.

It implies that the Hakimi method can be applied to those links that do not violate Equation 1. So, for a link (r,s) that satisfies Equation 2:

$$\frac{m(r,a) + m(s,b) - l(r,s)}{2} \geq m(i^*) \tag{2}$$

The local 1-center of this link (r,s) cannot further improve the vertex 1-center solution $m(i^*)$. The fact is that the maximum distance associated with the vertex 1-center must be greater than or equal to the corresponding distance for the absolute 1-center (7). In other words, if Equation 2 is satisfied, the link (r,s) need not be examined further. Through such a test, considerable computational effort will be reduced. But the number of computations that can actually be saved depends on the network configuration. It is difficult to predict a specific number of reductions if the shortcut is applied. However, this shortcut shows its ability to eliminate several unnecessary calculations.

A NEW SHORTCUT

A new shortcut is proposed in this section. Nodes a and b are assumed to be the farthest nodes that can be reached by nodes r and s shown on Figure 1a. Then, we have

$$\overline{ra} = m(r,a)$$
$$\overline{sb} = m(s,b)$$

There exists one point p on the path r-a that makes $\overline{pa} = m(i^*)$. Similarly, there is another point q on $\overline{sb}$ with the property of $\overline{qb} = m(i^*)$. Equation 1 can be rearranged in the following form:

$$\frac{m(r,a) - m(i^*) + m(s,b) - m(i^*) - l(r,s)}{2} \leq 0 \tag{3}$$

Then, based upon the preceding definitions, we have

$$\overline{rp} + \overline{sq} - l(r,s) \leq 0 \tag{4}$$

It means that any link in the network satisfying Equation 4 may be able to improve the final solution of the absolute 1-center. Only such a link will be considered in making further calculations. From Figure 1b, x and y are defined as:

$$\{x \mid \overline{xr} + \overline{ra} = \overline{xs} + \overline{sa}\ ,\ x \in \text{link } (r,s)\} \tag{5}$$
$$\{y \mid \overline{ys} + \overline{sb} = \overline{yr} + \overline{rb}\ ,\ y \in \text{link } (r,s)\} \tag{6}$$

Let x' and y' have this relationship:

$$\overline{x'r} = 2\overline{xr} + \overline{rp}$$
$$\overline{y's} = 2\overline{ys} + \overline{sq}$$

As far as $\Delta\ rsa$ (Figure 1b) is concerned, the distance from x passing through node r to node a should be equal to the distance traveling from x through nodes s to a. Suppose x is the absolute 1-center of a given network; path x-r-a will have the longest distance. This value can be further decreased if the absolute 1-center is not located on x. There are two possibilities. The first way considers the center located on the left-hand side of x. In such circumstances, the best location obviously belongs to node r. The distance from node r to the farthest node a is $m(i^*)$ plus $\overline{rp}$, $\overline{ra} = m(i^*) + \overline{rp}$. It is greater than or at most equal to $m(i^*)$ and may not be the best choice. Another possibility is to move the center to the right-hand side of x. The distance from x' to a becomes $m(i^*)$ if only $\overline{rp}$ distance units are shifted from x to x'. Furthermore, once the length of link (r,s) is larger than $\overline{x'r}$ (i.e., $l(r,s) \geq 2\overline{xr} + \overline{rp}$,

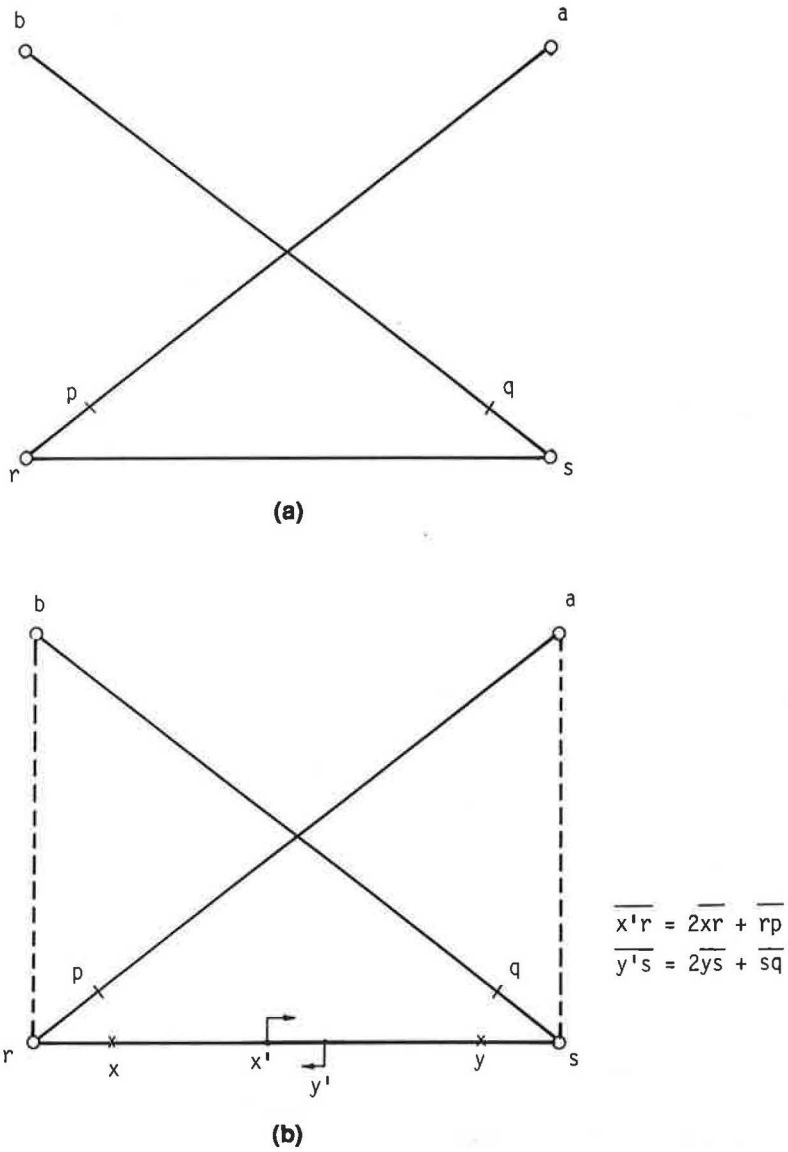

FIGURE 1 (a) Graphic representation of service from link (r,s). (b) Graphic representation of different service range for link (r,s).

the service distance from x' to node a will be less than or equal to $m(i^*)$. It is also necessary to make $l(r,s) \geq 2\overline{ys} + \overline{sq}$ to have the same property. Besides, x' must lie in the left side of y' to guarantee that the shortest distance from x' to node a or from y' to node b is smaller than $m(i^*)$. Thus,

$$l(r,s) \geq 2\overline{xr} + \overline{rp} + 2\overline{ys} + \overline{sq} \tag{7}$$

$$l(r,s) \geq 2\left[\frac{m(r,a) + l(r,s) + m(s,a)}{2} - m(r,a)\right] + m(r,a) - m(i^*) + 2\left[\frac{m(s,b) + l(r,s) + m(r,b)}{2} - m(s,b)\right] + m(s,b) - m(i^*)$$

$$l(r,s) \geq l(r,s) + m(s,a) - m(i^*) + l(r,s) + m(r,b) - m(i^*)$$

$$\frac{m(s,a) + m(r,b) + l(r,s)}{2} \leq m(i^*) \tag{8}$$

$m(r,b)$ = the distance from node r to the farthest node b (for s) and

$m(s,a)$ = the distance from node s to the farthest node a (for r).

Any link that violates Equation 8 cannot improve the final solution of absolute 1-center and will not be further considered.

Because

$$m(r,a) \leq m(s,a) + l(r,s)$$

$$m(s,b) \leq m(r,b) + l(r,s)$$

$$\frac{m(r,a) + m(s,b) - l(r,s)}{2} \leq \frac{m(s,a) + m(r,b) + l(r,s)}{2} \leq m(i^*) \tag{9}$$

It is noted that the proposed shortcut has a more strict bound than the Larson and Odoni shortcut does based on Equa-

tion 9. If the proposed shortcut given in Equation 8 is considered, then the local 1-center of link (r,s) is located on the middle point of $\overline{x'y'}$. Its location has DI distance units from node r.

$$\begin{aligned} DI &= \overline{x'r} + 1/2[l(r,s) - \overline{x'r} - \overline{y's}] \\ &= l(r,s) + m(s,a) - m(i^*) + [l(r,s) \\ &\quad - 2l(r,s) - m(s,a) \\ &\quad - m(r,b) + 2m(i^*)] \\ &= 1/2[l(r,s) + m(s,a) - m(r,b)] \end{aligned} \tag{10}$$

The service distance of this local 1-center to the farthest node is

$$\begin{aligned} SS &= m(i^*) - 1/2[l(r,4s) - \overline{x'r} - \overline{y's}] \\ &= m(i^*) - 1/2[l(r,s) - 2l(r,s) - m(s,a) \\ &\quad - m(r,b) + 2m(i^*)] \\ &= 1/2[m(s,a) + m(s,b) + l(r,s)] \end{aligned} \tag{11}$$

Based on the preceding discussion, the proposed shortcut can be performed as follows. First, for any link in the network, we check whether it satisfies Equation 8. If the answer is yes, then Equations 10 and 11 will be applied to find the local 1-center of that link. Otherwise, the link need not be further considered. After all links have been examined, the location and service distance of absolute 1-center for the given network can easily be determined. The foregoing procedure is rather simple and makes it easy to obtain the final solution without using elaborate computations. Comparisons of this new shortcut with other methods are given later.

MINIEKA METHOD

A polynomial time algorithm for finding the absolute 1-center of a network was proposed by Minieka (*8*). This algorithm is combinatorial in nature and requires only knowledge of the shortest path distances between all pairs of nodes. Conceptually, it is different from the Hakimi method. Consider p on a link (r,s) as one point on the link (r,s) that is p units from r and $l(r,s) - p$ units from s, where $0 \leq p \leq l(r,s)$. Those nodes that are best reached from p by traveling through node r are set in node set R. Similarly, others best reached through node s belong to set S. On the basis of this definition, the Minieka method for finding p^*, the local 1-center on a link (r,s), follows these steps:

Step 1: Obtain the shortest matrix between all nodes through any efficient algorithm.

Step 2: Place all nodes in R, and arrange the sequence of nodes according to the order of their distance from node r, with the most distant node first. Compare the maximum distance from node r to all other nodes of the network with the link length $l(r,s)$ and then store the higher value as the first point-to-node distance.

Step 3: Remove from R and place into S the node that is currently most distant from node r.

Step 4: Compare the distance from node s to the node that has the maximum distance in R with the largest value of the current set S. If this new distance is smaller than the existing maximum distance, go to Step 3; otherwise go to Step 5.

Step 5: Calculate the maximum distance needed from both sets R and S, using the equation

$$\text{MD} = [d(r,z_i) + d(s,z_k) + l(r,s)]/2 \tag{12}$$

where

MD = the current maximum distance needed to serve customers in the both sets R and S;
$d(r,z_i)$ = maximum distance from node r to node z_i in the current set R;
$d(s,z_k)$ = maximum distance from node s to node z_k in the current set S; and
$l(r,s)$ = actual link distance between nodes r and s.

Step 6: Compute the p^* by subtracting $d(r,z_i)$ from MD.

Step 7: Go to Step 3 until all other nodes have been examined and moved to set S. Compare the length of link (r,s) with the maximum distance from S, and then store the higher value as a MD with p^* equal to the length of link (r,s).

Step 8: Choose the smallest MD and its related p^* value among all candidates. This is the local 1-center of link (r,s).

The foregoing procedure can be used for finding the local 1-center of link (r,s). Obviously, it is also applicable to all other links. Thus, the local centers of other links are found through the same steps. After all links have been examined, the best absolute 1-center of the network is determined simply by choosing the minimum among all local 1-center candidates. This method performs the preceding steps easily and can be used to solve large network problems. Its computational effort mainly lies in obtaining the all-to-all shortest-distance matrix. Therefore, this is a polynomial time-bounded algorithm and is easy to program.

A COMBINED METHOD

Although the Minieka method is efficient in computing the local 1-center on a link, it still requires much effort to examine all links of a given network if no bounding technique is applied. For the Larson and Odoni shortcut, considerable reduction in computational effort can be achieved by omitting many unnecessary links before searching for the absolute 1-center. After the shortcut is applied, however, the inefficient Hakimi method is used to find local centers for those critical links that do not violate Equation 1. Therefore, it becomes feasible to combine the Minieka method with the Larson and Odoni shortcut to reduce further the number of calculations and computer time. The basic idea of this combination is simply to consider the Larson and Odoni shortcut first in deleting links that cannot improve the solution. Then only those links satisfying Equation 1 are examined and calculated to determine their local centers using the Minieka method. It is expected that the computational effort will be reduced through this combined method. The steps of this combined method are summarized next.

Step 1: Obtain the shortest distance matrix between all nodes.

Step 2: Apply the Larson and Odoni shortcut to delete those links that satisfy Equation 2.

Step 3: Use the Minieka method to find the local 1-center for each critical link and store it as a candidate.

Step 4: Repeat Step 3 until all critical links have been examined.

This combined method takes advantages of the most efficient parts of the Minieka method and the Larson and Odoni shortcut. Tests of a newly developed computer program show that the program works well and reduces some computer time. These tests are discussed more extensively later.

A NEW METHOD

In this section, a new method for finding the absolute 1-center is proposed. The solution is obtained from a spanning tree based on the vertex's one-to-all shortest distances. It first considers the longest and second longest distances of the spanning tree from each node in a network (*9,10*). For each such tree, the local 1-center is found. Then the minimum of local centers is selected as the absolute 1-center for the entire network. Because the new method finds the local 1-center from the spanning tree of each node, the maximum number of iterations needed to perform the computation is in proportion to the number of nodes, instead of links, for the given network. In other words, conceptually, the new method can reduce computer time more than the previous method if larger networks are considered. The steps included in this new method are as follows:

Step 1: Obtain the shortest path for each vertex to all other nodes.

Step 2: Find the farthest node i and second longest distance node j to form the nonoverlap distance $x(i,j)$ according to the minimum spanning tree of each node from Step 1.

Step 3: Determine the service distance of local 1-center for each node and store it as a candidate:

$$1/2[\max x(i,j)] \tag{13}$$

Step 4: Repeat Steps 2 to 3 until all nodes have been examined.

Step 5: Choose the minimum value among all candidates. This is the absolute 1-center for the given network.

It is easy to perform the preceding steps by using the graphic method manually. Steps 2 and 3 need to be modified, however, if the new method is to be programmed. After several tests, it is found that the local 1-center of the designated node may not always be located on the path that includes the longest and second longest distances rooted at each node. More verifying steps must be added to obtain a better solution of the local 1-center. The best way to perform this analysis is to check all connecting links from that node. This can be observed in Figure 2a. Suppose node I, with the longest path I-A and the second longest path I-B, is under consideration. The pivotal local 1-center of node I is located on M with MI distance from node I. Link (I,K) represents one connecting links originating from node I. The service distance SS of the initial local 1-center based on the previous steps equals $\frac{1}{2}(IA + IB)$. If the shortest distances from node K to nodes A and B satisfy Equation 14,

$$\text{Max } [D(K,A),D(K,B)] = GL < SS \tag{14}$$

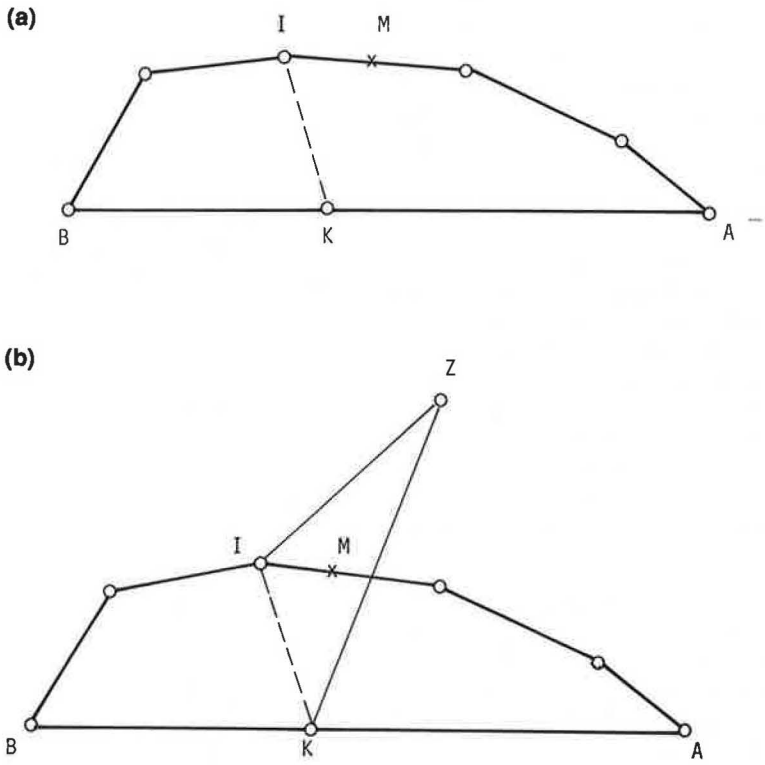

FIGURE 2 (a) The farthest node *A* and second longest distance node *B* for node *I* and one of connecting links (*I*,*K*). (b) Checking steps for connecting links rooted from node *I*.

where

$D(K,A)$ = the shortest distance from nodes K to A;
$D(K,B)$ = the shortest distance from nodes K to B; and
GL = higher value of $D(K,A)$ and $D(K,B)$,

then it is possible for the local 1-center to be located on the connecting link (I,K) instead of the original shortest path. What would be the most desired location of local 1-center for node I? Before performing more analyses, we denote W as the nodes that are different from nodes I, K, A, and B but satisfy the following two conditions.

$$D(K,W) \geq GL$$

$$D(I,W) + D(I,K) \geq GL$$

where

$D(K,W)$ = the shortest distance from nodes K to W;
$D(I,W)$ = the shortest distance from nodes I to W; and
$D(I,K)$ = the shortest distance from nodes I to K.

Let Z be the node that has the largest value among all $D(I,W)$. The new local 1-center stays on link (I,K) if Equation 15 is met.

$$D(I,Z) + D(I,K) \geq GL \tag{15}$$

Otherwise, the local 1-center remains at the node K. The service distance and location of local 1-center for node I become SS and p^*, respectively.

$$SS_1 = 1/2[D(I,Z) + D(I,K) + GL] \tag{16}$$

$$p^* = SS_1 - D(I,Z) \tag{17}$$

After obtaining the new SS_1, if SS_1 is smaller than SS, then SS_1 will substitute SS as the new service distance of local 1-center. The location of this local 1-center is located on the connecting link (I,K) with p^* distance units from node I. Otherwise, the SS value still represents the service distance of local 1-center. After all connecting links have been examined, the smallest value among all SS is chosen. The smallest value and its corresponding location p^* are considered the local 1-center of node I.

To put the preceding discussion into sequential steps, we substitute the following Steps 1a through 5a for Steps 2 and 3 and add Steps 6a to 8a for checking connecting links. Before describing these steps, let $c(i,j)$ be the shortest distance from nodes i to j and $b(i,j) = A$ be the nearest node number on the shortest path from node i to all other nodes j; $g(i,j)$ represents the largest value among all $c(i,j)$, and $h(i,j')$ denotes the second largest value in the all remaining $c(i,j)$. B gives the node letter j that has the second longest distance from node i. Besides, $N(i)$ shows the node letter i currently under consideration.

Step 1a: List all $c(i,j)$ and $b(i,j) = A$ for node i;

Step 2a: Find the largest value $g(i,j)$ among $c(i,j)$ and its nearest node letter A from node i to all other nodes;

Step 3a: Determine the second largest value among remaining $c(i,j)$ that the nearest node number is not A and denote it as $h(i,j')$ and $b(i,j') = B$;

Step 4a: Calculate $Q(i)$ through the following equation:

$$Q(i) = \frac{g(i,j) + h(i,j')}{2} \tag{18}$$

Step 5a: If $[Q(i) - h(i,j')]$ is less than or equal to $c(i,A)$, then $Q(i)$ is the local 1-center distance for node i. Determine the suitable location p^* for $Q(i)$ and go to Step 6a. If $[Q(i) - h(i,j'] > c(i,A)$, go back to Step 4;

Step 6a: Check one connecting link from node i to node k:

$$GL = \text{Max}[D(k,A),D(k,B)]$$

If $GL \leq Q(i)$, go to Step 7a; otherwise, go to Step 8a;

Step 7a: Find z that satisfies the following two conditions and has the largest value:

$$D(i,z) + D(i,k) \geq GL \text{ and } D(k,z) \geq GL$$

If there is no z available, go to Step 8a.

$$Q'(i) = 1/2[D(i,z) + D(i,k) + GL]$$

If $Q'(i) < Q(i)$, then $Q(i) = Q'(i)$, $p^* = Q'(i) - D(i,z)$. Go to Step 8a.

Step 8a: Check other connecting links originating from node i. If all links have been examined, go to Step 4; otherwise, go to Step 6a.

EXAMPLE

Find the absolute 1-center of the network shown in Figure 3 using the new method. This example requires that the shortest distance from each node to all vertices be calculated in the first step. Then, the spanning tree of the designated node based on the shortest distance from each node to all vertices is calculated in the second step. The spanning tree of the designated node based on the shortest distance is then obtained. Figure 4 shows the spanning tree of node 7. After several checking steps, the initial local 1-center becomes the center of node 7. From this figure, it can be seen that the distance between nodes 5 and 10 is 155. Thus, the initial local 1-center for node 7 equals 77.5 units, according to Equation 13. This center will be located on link (7,8) at a distance of 0.5 units from node 7. Similarly, the initial local 1-center of node 6 can be easily obtained from Figure 5a. This initial local 1-center has 62.5 distance units and stays on link (6,10) with a distance

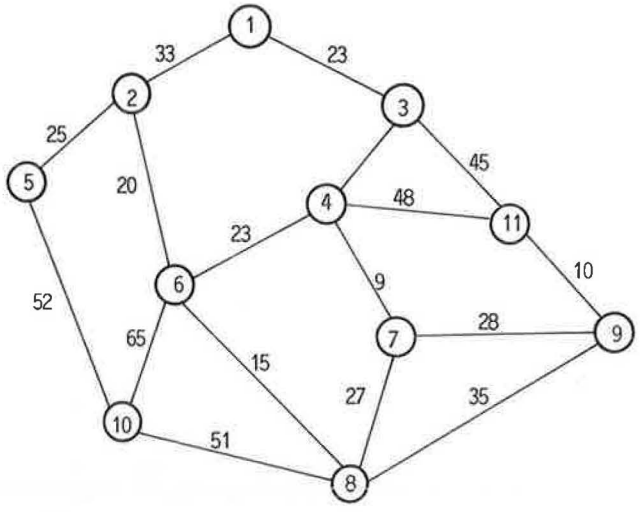

FIGURE 3 Distance and configuration of given network.

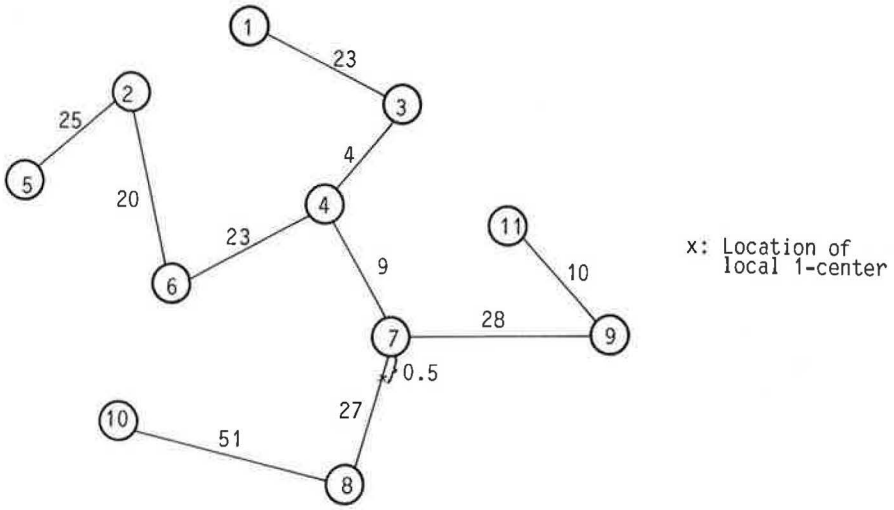

FIGURE 4 Spanning tree of node 7 and its location of local 1-center.

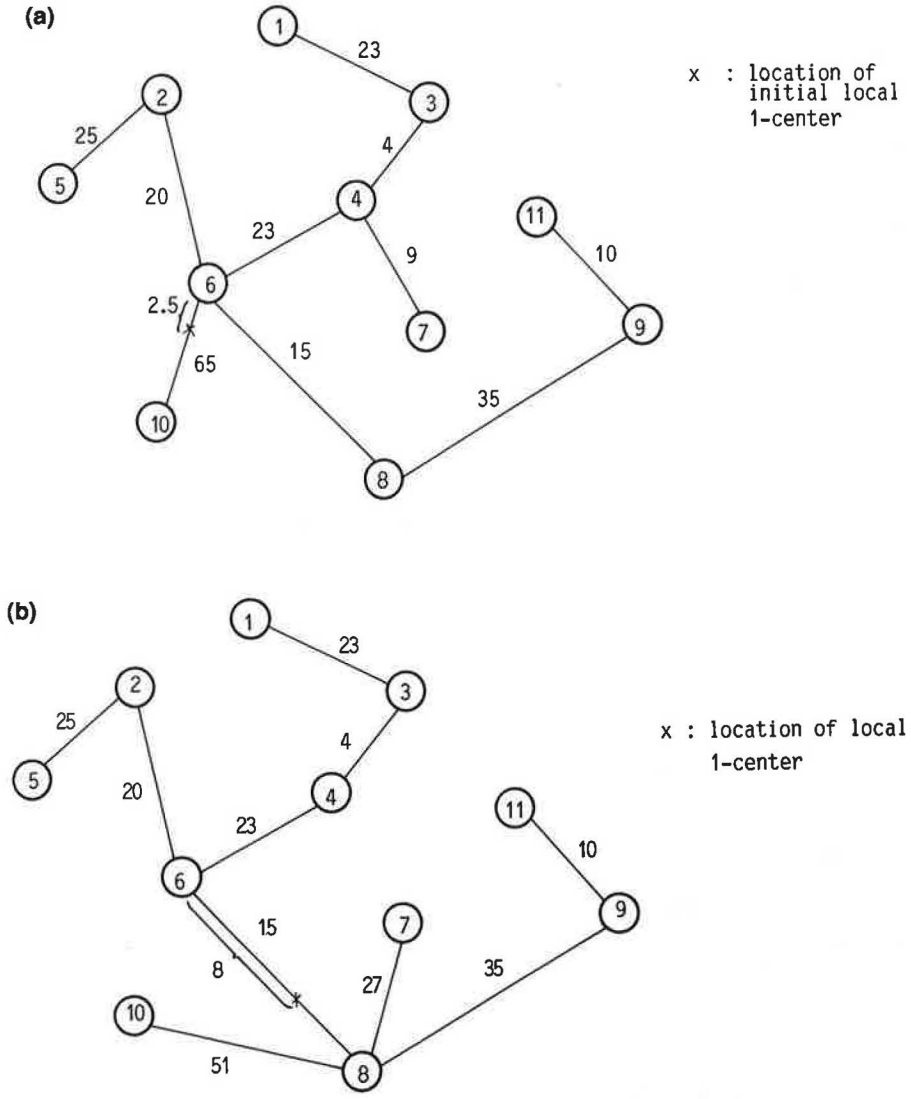

FIGURE 5 (a) Initial local 1-center of node 6 with 62.5 units service distance. (b) Location of local 1-center for node 6 with 58 units.

of 2.5 units from node 6. It is not the best solution for node 6. In fact, the local 1-center of node 6 is located on link (6,8) with 58 distance units from node 6 to serve all other nodes by applying Steps 6a to 8a to examine each connecting link rooted from node 6. Its final location can be seen from Figure 5b. This location and service distance also represents the absolute 1-center for the given network. Therefore, from this example, it is important to examine all connecting links of any designated node before its local 1-center is finally determined.

Another way of searching the local 1-center of each node is simply to apply the given Steps 1a to 8a. A complete computational procedure for node 7 is shown on Table 2 based on these steps. Definitions of all variables in the table are referred to this section. Table 3 gives the computational result of local 1-center for node 6. This case provides the user a better solution after checking each connecting link originated from node 6. It is noted that the results obtained from Table 3 are the same as those shown in Figure 5b.

COMPARISON OF FOUR METHODS

Thus far, four different methods for finding the absolute 1-center have been discussed: the new shortcut, the Minieka method, the combined method, and the new method. As far as computer time and computational complexity are concerned, it is necessary to understand the capabilities and limitations of these four methods. One analytic strategy is to apply four methods to the same given network. To make such a comparison, nine networks with different numbers of nodes and links are selected. In these networks, an absolute 1-center will be sought such that the maximum service distance from this center to all nodes is minimized. The absolute 1-center can be located anywhere on a link or at a node.

Four computer programs have been developed separately for the four methods. Each program reads the same network input file and prints the output in an identical format. Each program was run on a PC/AT with a math coprocessor 80287-10. For each network, the final distance and location

TABLE 2 COMPUTATION OF LOCAL 1-CENTER FOR NODE 7 BY APPLYING STEPS 1a TO 8a

Node i	Node j										
N(7)=7	1	2	3	4	5	6	7	8	9	10	11
c(7,j)=	36	52	13	9	77	32	0	27	28	78	38
b(7,j)=	4	4	4	4	4	4	7	8	9	8	9

A. Search the initial local 1-center:

g(7,10) = 78, A = 10, h(7,5) = 77, B = 5

b(7,10) = 8 $\neq$ b(7,5) = 4

$$Q(7) = \frac{g(7,10) + h(7,5)}{2} = 77.5$$

Q(7) - h(7,5) = 77.5 - 77 = 0.5 < c(7,8) = 27, O.K.

The initial local 1-center of node 7 is located on link (7,8) and 0.5 distance units from node 7.

B. Check each connecting link

(1) link (7,4), k = 4

GL = max[D(4,10) = 87, D(4,5) = 68] = 87 $\nleq$ Q(7)

(2) link (7,9), k = 9

GL = max[D(9,10) = 86, D(9,5) = 95] = 95 $\nleq$ Q(7)

No connecting links can provide a better solution, so the local 1-center of node 7 is still located on link (7,8) and 0.5 distance units from node 7.

TABLE 3 COMPUTATION OF LOCAL 1-CENTER FOR NODE 6 BY APPLYING STEPS 1a TO 8a

Node i	Node j										
N(6)=6	1	2	3	4	5	6	7	8	9	10	11
c(6,j)=	50	20	27	33	45	0	42	15	50	65	60
b(6,j)=	4	2	4	4	2	6	4	8	4	10	8

A. Search the initial local 1-center:

g(6,10) = 65, A = 10, h(6,11) = 60, B = 11

b(6,10) $\neq$ b(7,11)

$$Q(6) = \frac{g(6,10) + h(6,11)}{2} = 62.5$$

Q(6) - h(6,11) = 62.5 - 60 = 2.5 < c(6,10) = 60, O.K.

The initial local 1-center of node 6 is located on link (6,10) and 2.5 distance units from node 6.

B. Check each connecting link:

(1) link (6,2), k = 2

GL = max[D(2,10) = 85, D(2,11) = 80] = 85 $\not\leq$ Q(6)

(2) link (6,4), k = 4

GL = max[D(4,10) = 87, D(2,11) = 47] = 87 $\not\leq$ Q(6)

(3) link (6,8), k = 8

GL = max[D(8,10) = 51, D(2,11) = 47] = 51 < Q(6)

One connecting link (6,8) may provide a better solution.

More checking steps need to be undertaken.

D(6,1) + D(6,8) = 65 $\geq$ GL = 51

D(8,1) = 63 $\geq$ GL = 51

Q'(6) = [D(6,1)+D(6,8)+GL] = [50+15+51] = 58 $\leq$ Q(6)

P* = Q'(b) - D(b,1) = 58 - 50 = 8 $\leq$ c(6,8) = 15 O.K.

Thus, the local 1-center of node 6 is located on link (6,8) and 8 distance units from node 6.

of the absolute 1-center are the same according to the output of the four computer programs. They all provide the best solution. Comparisons of computer time used and the numbers of links and nodes considered by each method are summarized in Table 4. It can be seen that the combined method is more efficient than the Minieka method, because the former skips many unnecessary links before searching the local 1-center. The new method and the new shortcut are obviously better than the combined method. Both the new method and the shortcut use almost the same computer running time. For a network with 80 nodes and 141 links, the new method and shortcut need only 45 percent of the Minieka's computer time and 61 percent of the time required by the combined method. The results show that the new method and the shortcut are computationally fast and powerful if larger networks are considered. Also, both new methods can be categorized as polynomial time-bounded algorithms.

CONCLUSIONS

The combined method finds the absolute 1-center with fewer link computations than the Minieka method does, if the latter is assumed to examine all links. The computational complexity of this technique relies on the efforts of finding the all-to-all shortest distance paths and requires $0(N^3)$ calculations. Hence, the combined method is a polynomially bounded algorithm and requires less computational effort. The proposed new

TABLE 4 COMPARISONS OF FOUR METHODS BY RUNNING ON PC/AT WITH A MATH COPROCESSOR 80287-10

Network Number	No. of Nodes	No. of Links	Minieka Method (sec)	Combined Method (sec)	New Shortcut (sec)	New Method (sec)
1	15	27	3.6	4.5	4.1	3.7
2	17	36	4.7	5.3	4.5	4.0
3	25	50	9.9	9.5	7.5	7.2
4	30	40	13.1	12.4	10.0	9.5
5	40	56	27.7	23.3	17.5	17.0
6	45	95	44.0	32.8	22.6	22.2
7	50	74	51.7	41.0	28.4	28.0
8	65	100	109.5	82.2	53.4	53.0
9	80	141	207.4	148.0	90.4	90.0

shortcut gives a stricter bound than does the Larson and Odoni shortcut. After this new shortcut is applied, the location and its service distance to the local 1-center for the desired link can be obtained directly. The new method finds the absolute 1-center by considering the number of nodes instead of the number of links in a network. Although the combined method has reduced the number of links needed in calculating the local 1-center, the number of remaining links, in most cases, is still greater than the number of nodes considered for the given network. Therefore, after several tests, it can be concluded that the new method and the new shortcut are faster and more powerful than the combined method or the Minieka method, especially for a large network. On this basis, the new method or the new shortcut is recommended for use in locating a new transportation facility if the absolute 1-center location problem is being considered.

ACKNOWLEDGMENT

The authors would like to express their deepest thanks for the assistance of Gwo-Feng Yang during the phases of developing computer programs and preparing this final manuscript.

REFERENCES

1. H. S. Tsay. *The Combined Facility Location and Vehicle Routing Problem: Formulation and Solution*. Ph.D. dissertation. Purdue University, West Lafayette, Ind., Aug. 1985.
2. H. S. Tsay. New Efficient Algorithms for Solving 1-Center Location Problems. *Transportation Planning Journal*, Vol. 15, No. 2, June 1986, pp. 207–220. (Printed in Taiwan)
3. S. L. Hakimi. Optimum Distribution of Switching Centers and the Absolute Centers and Medians of a Graph. *Operations Research*, Vol. 12, 1964, pp. 450–459.
4. S. L. Hakimi. Optimum Distribution of Switching Centers in a Communication Network and Some Related Graph Theoretic Problems. *Operations Research*, Vol. 13, 1965, pp. 462–475.
5. B. C. Tansel, R. L. Francis, and T. J. Lowe. Location on Networks: Parts I and II. *Management Science*, Vol. 29, 1983, pp. 482–511.
6. J. D. Fricker and H. S. Tsay. A Polynomial Time Algorithm to Solve the General Absolute 1-Center Problem. Presentation at ORSA/TIMS Joint National Meeting, Atlanta, Ga., Oct. 1985.
7. R. C. Larson and A. R. Odoni. *Urban Operations Research*. Prentice-Hall, Englewood Cliffs, N.J., 1981, pp. 439–442.
8. E. Minieka. A Polynomial Time Algorithm for Finding the Absolute Center of a Network. *Networks*, Vol. 11, 1981, pp. 351–355.
9. G. Y. Handler and P. B. Mirchandani. *Location on Networks: Theory and Algorithms*. MIT Press, Cambridge, Mass., 1979.
10. H. S. Tsay and L. T. Lin. A New Method for Finding Absolute 1-Center in a Network. Presented at 1987 TIMS/ORSA Joint National Meeting, New Orleans, La., May 1987.

Locomotive Scheduling Under Uncertain Demand

SCOTT SMITH AND YOSEF SHEFFI

Each day, railroads face the problem of allocating power to trains. Often, power requirements for each train are not known with certainty, and the fleet of locomotives may not be homogeneous. To deal with both of these complications, we formulate a multicommodity flow problem with convex objective function on a time-space network. The convex objective allows us to minimize expected cost under uncertainty by penalizing trip arcs likely to have too little power. Our solution heuristic sends locomotives down the shortest paths (based on marginal arc costs) in the time-space network and then attempts to improve interchanges of locomotives around cycles. Two lower bounds are also developed by relaxing the multicommodity aspect of the problem. In 19 test problems, ranging in size from 15 to 404 arcs, the heuristic performed well, with short running times and costs averaging within 3 percent of the best of the two lower bounds developed.

A problem frequently faced by transportation carriers is the allocation of a fixed supply of vehicles to a given schedule. Examples include the allocation of locomotives to freight trains, of buses to transit routes, and of airplanes to flights. These examples have the following features in common:

1. There is a published or "committed to" schedule of services that have to be carried out;
2. The supply of vehicles to trips can be represented as an integer, multicommodity minimum cost flow problem over a network of trip, layover, and storage arcs. The problem has multicommodity aspects because the vehicle fleet is not homogeneous; for example, locomotives may have different power ratings and airplanes may be of different sizes. (Naturally, however, there are some important differences between the modes. For example, in the rail mode, two or more locomotives typically are used to meet demand for a given trip, whereas only one airplane is used for a single airline trip);
3. Even though the schedule is fixed, the demand for service may vary. In the rail context, the tonnage of a given train is variable. In the bus or airline context, the number of passengers on a given trip will vary. Further, it may sometimes be desirable not to meet all the demand, for example, by having standees on buses, or refusing airline reservations, or leaving cars behind for the next freight train to pick up.

This paper formulates this allocation problem and suggests solution techniques in the context of rail. First, background information on both the formulation of the problem and past research in this area is presented. Second, the problem is formulated as a mathematical program. Third, a fast heuristic solution technique is presented. Finally, the results of the heuristic are compared with lower bounds obtained through various relaxations. The techniques presented here explicitly consider uncertainty in locomotive demand and are able to deal with locomotives of different power ratings.

Department of Civil Engineering, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, Mass. 02139.

BACKGROUND

This section looks at the network representation of the locomotive scheduling problem. This formulation underlies virtually all other attempts in the literature to develop a solution for this problem. Some of that research is reviewed in the second part of this section.

Time-Space Representation

The rail scheduling problem is typically formulated as a minimum cost flow problem on a time-space network, which is a graph of locations versus time on which activities are plotted (Figure 1). Each node in this network represents a terminal (yard) at a point in time, and arcs are of the following types:

1. Trip arcs represent trains between the upstream terminal node and the downstream terminal node that the arc connects.

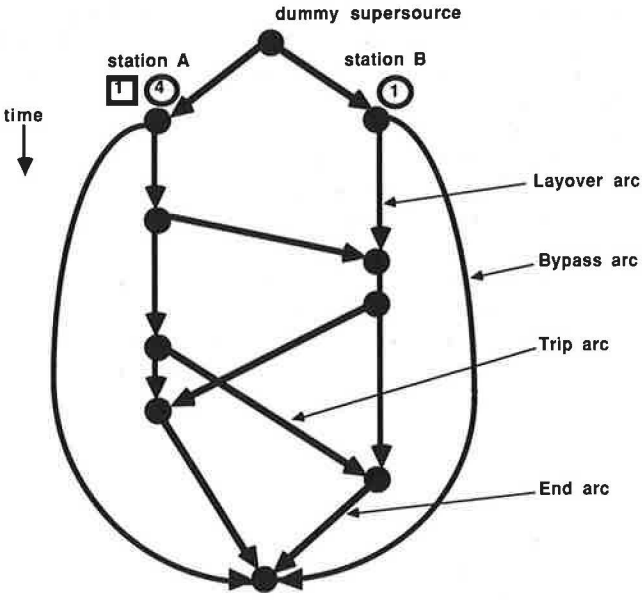

FIGURE 1 Sample time-space network.

There is a power requirement for each trip, which may be represented either by a lower bound on the arc flow of locomotives or by a penalty function that increases greatly as this flow falls below the minimum desired value;

2. Layover arcs represent short-term storage at a terminal. They have a lower bound of 0 and some fixed cost per unit;

3. Bypass arcs represent long-term storage of unneeded units and have very low cost per unit; and

4. End arcs represent locomotive requirements at each terminal at the end of the planning horizon. Any practical time-space representation has a finite planning horizon. Therefore, if this horizon is short, end effects must be considered. In the model here, we want to know how many locomotives are needed at each terminal at the end of day, week, or whatever period we are modeling. Thus, the end arcs will have either cost functions or lower bounds similar to those for trip arcs.

This is a multicommodity network flow problem with either a "bundle constraint" in the lower bound for each arc or a penalty term in the cost function that "bundles" the commodities. We flow locomotive units of various types through the network, but a minimum level of motive power must be met for each arc.

Past Work

Comprehensive reviews of rail scheduling are contained in two papers by Assad (*1,2*) and one by Peterson (*3*). Some of the earliest analytical work in locomotive assignment is that of Bartlett in 1957 (*4*), who presented a pairing algorithm to assign vehicles to a fixed schedule. Later, McGaughey et al. (*5*) described the distribution of locomotives and cabooses with a time-space network model. They used an out-of-kilter algorithm to find the optimal flow of units through a single-commodity network with a fixed lower bound on the power supplied to each arc. In 1976 Florian et al. (*6*) considered the multicommodity aspect of locomotive scheduling, with fixed lower bounds. They used Bender's decomposition to solve this multicommodity flow problem and reported good results with medium-size (about 200 train movements) problems but had less success with larger problems. In 1980, Booler (*7*) formulated the same multicommodity flow problem but obtained an integer solution using a heuristic method based on linear programming.

All of this work assumes deterministic, known lower bounds on the power flows. Furthermore, there has been only limited success with multicommodity flows, particularly with large problems, as already mentioned. As the first step to the explanation of our approach to the problem, the next section formulates the locomotive allocation problem as a mathematical program. Later we assume that the lower bound is not known with certainty, and we reformulate the problem using a penalty function.

FORMULATION

This section starts with the "traditional" mathematical programming formulation of the problem. It then incorporates the uncertainty in locomotive requirements directly into the formulation using several probability density functions of the power needs. The formulation and notation that follow relate to the time-space representation of the locomotive assignment problem.

Define the following:

i = arc number in the time-space network,
j = locomotive type,
x_{ij} = flow of locomotive type j on arc i,
H_j = horsepower rating of locomotive type j
s_i = horsepower flow on arc i ($s_i \equiv \Sigma_j H_j x_{ij}$),
x_j = vector of locomotive flows of type j on all arcs,
$C_i(s_i)$ = general operating cost function on arc i,
c_i = operating cost per unit HP flow on arc i,
l_i = demand for power on trip arc i (this may be either a deterministic value or a random variable),
F_i = cumulative distribution function for l_i,
f_i = probability density function for l_i,
μ_i = average demand for power (expected value of l_i),
σ_i = standard deviation of demand for power,
$P_i(s_i)$ = general penalty cost function for power shortfall on link i,
p_i = penalty per unit of power shortfall on arc i,
$Z_i(s_i)$ = cost function on arc i (including operating cost and penalty),
N = node arc incidence matrix for the time-space network, and
b_j = vector of sources and sinks for locomotive type j.

We first formulate the problem with deterministic lower bounds on the power requirements, and then show how these lower bounds can be modeled as random variables. That formulation leads to the use of penalty functions in the objective of the mathematical program. In all cases, we assume that the lower bound is expressed in terms of horsepower, so that combinations of locomotive types with the same total power rating are interchangeable. The mathematical formulation is

$$\min \Sigma_i c_i (\Sigma_j H_j x_{ij}) \tag{1}$$

subject to

$$\Sigma_j H_j x_{ij} \geq l_i \text{ for all } i \tag{2}$$

$$Nx_j = b_j \text{ for all } j \tag{3}$$

$$x_{ij} \text{ integer and } \geq 0 \tag{4,5}$$

This is a multicommodity minimum cost network flow problem. The objective is to assign all locomotive types j to the network, whose node-arc incidence matrix is N, at minimum cost. The various locomotive types cannot be assigned separately because they all contribute to the power on each train link. This bundling of locomotive types appears in the lower bound constraint 2.

Recall that the original problem calls for uncertainty in demand. Therefore, the fixed lower bound formulation of Equations 1 through 5 may not be realistic. This is because a fixed lower bound can be thought of as an infinite cost penalty on flows below it. Such a cost function for an arc with a lower bound of 5000 and cost per unit flow of c is shown in Figure 2. According to this cost function, 4999 HP on this train has an infinite cost whereas 5000 HP has the lowest cost, clearly an unrealistic situation in a world where the 5000-HP

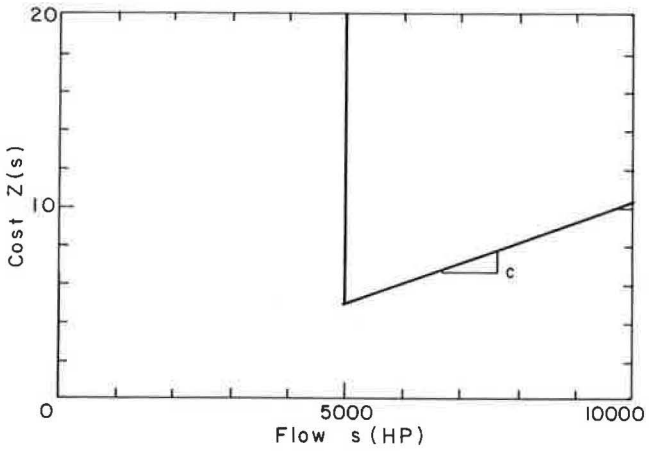

FIGURE 2 Arc cost function with deterministic lower bound.

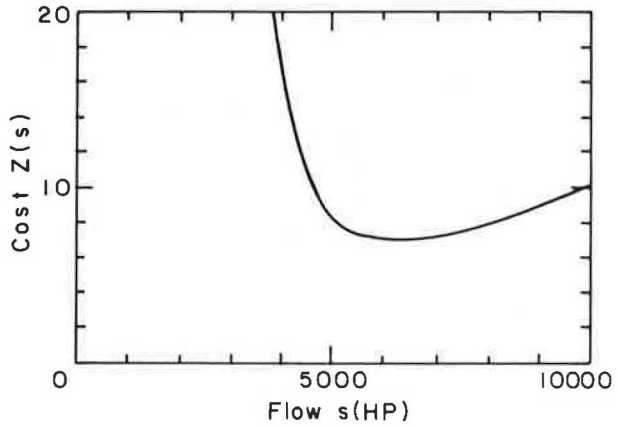

FIGURE 3 Arc cost function with uncertain power demand.

requirement is actually a random variable, which can attain values lower than 5000 HP. A more realistic cost function (Figure 3) might be obtained by reasoning as follows:

Our forecast demand is 5000 HP but we know the forecast might be off by as much as 1000 HP. Therefore, our safest assignment would be to have 6000 HP on this train; 5000 HP would probably work; 4000 HP would be unacceptable. However, we do not want a fixed lower bound of 6000 HP because 5000 HP may be all the power we have available.

To yield a cost function that looks like the one shown in Figure 3, we move constraint 2 into the objective with a penalty term. By doing so, we acknowledge that (a) power requirements may vary in a random manner and (b) the lower bound on power is not a hard-and-fast rule; rather, there is a trade-off between service quality and the amount of power supplied. This formulation provides a more realistic representation and, arguably, makes the problem easier to solve. Thus, rather than having the demand for power, l_i, as a fixed lower bound, it is modeled as a random variable.

The shape of the cost function derived this way depends on three elements:

1. The operating cost of additional power; we assume this is some function, $C_i(s_i)$ of the power supplied, s_i. This cost is assumed independent of the demand, l_i, and locomotive type;
2. The distribution of the demand for power, l_i. The probability density function for l_i is denoted by f_i and its cumulative distribution by F_i. This distribution has mean μ_i and variance σ_i^2; and
3. The magnitude and shape of the penalty, given a shortage. This will have the form $P_i(l_i - s_i)$ (recall that $s_i = \Sigma_j H_j x_{ij}$).

In the equations that follow, the subscript i is dropped to make the notation easier to read.

The cost as a function of power supply, s, is $Z(s)$. Given the probability density function of l, this cost can be expressed as follows:

$$Z(s) = C(s) + \int_s^\infty P(l - s)\, f(l)\, dl \tag{6}$$

As a tractable approximation to the normal distribution, we assume that l follows a logistic distribution where

$$F(l) = [1 + \exp((a/\sigma)(\mu - l))]^{-1} \tag{7}$$

$$f(l) = (a/\sigma)\exp((a/\sigma)(\mu - l)) \times [1 + \exp((a/\sigma)(\mu - l))]^{-2} \tag{8}$$

where $a \equiv \pi/\sqrt{3} \approx 1.81$

The cost function, $Z(s)$, now becomes

$$Z(s) = cs + p\int_s^\infty lf(l)\, dl - ps\int_s^\infty f(l)\, dl$$

$$= cs + p\int_s^\infty lf(l)\, dl + ps(F(s) - 1) \tag{9}$$

After integrating (by parts) the cost becomes

$$Z(s) = cs + (p\sigma/a)\log(1/F(s)) \tag{10}$$

In the logistic distribution mentioned above, negative demand is theoretically possible. However, the parameters are such that this is not a problem in practice.

Cost functions were also derived for uniform and gamma distributed demands, with some examples plotted in Figure 4. Note the following points:

1. All functions approach the $\sigma = 0$ case asymptotically as s becomes either very large or very small with respect to μ;
2. There is not much difference between the cases with logistic, uniform, and gamma distributions. This is reassuring, because it indicates that the exact shape of the distribution for demand may not matter much, and we can use the logistic distribution to form a tractable cost function. Although the gamma distribution is probably the most realistic representation of l (it never has values below zero), it does not yield a closed form for $Z(s)$ and, consequently, is difficult to work with;
3. All functions have the desired shape (as in Figure 3). In the remainder of the paper, we assume that the demand for power l is logistically distributed with mean μ and standard deviation σ.

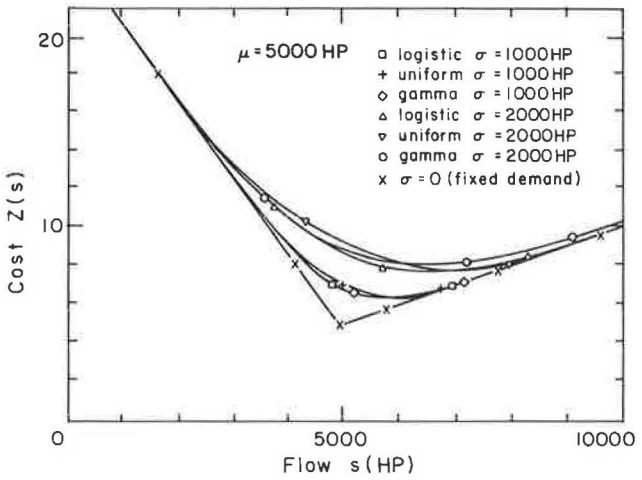

FIGURE 4 Arc cost function under various demand distributions.

The problem we are solving can now be summarized as follows: Minimize the total cost of flowing power on all arcs, subject to the following constraints:

1. Flow conservation constraints are met for each power class and
2. The number of locomotives on each arc is integer and nonnegative.

In other words:

$$\min \Sigma_i\{c_i s_i + (p_i\sigma_i/a) \log(1 + \exp[(a/\mu_i)(\mu_i - s_i)])\} \quad (11)$$

$$\text{subject to } Nx_j = b_j \text{ for all } j \quad (12)$$

$$x_{ij} \text{ integer and } \geq 0 \quad (13,14)$$

(As before, s_i is defined as $\Sigma_j H_j x_{ij}$ and $a = 1.81$.)

This representation, in addition to having the advantages mentioned earlier, also lends itself well to the treatment of uncertain end effect arcs, which can be modeled like trip arcs with high σ. The next section looks at solution approaches for this problem.

SOLUTION APPROACHES

We have formulated a nonlinear, multicommodity integer network flow problem. Exact solution techniques for such a problem are likely to be neither easy nor fast. This problem, however, is similar to the (multicommodity) traffic assignment equivalent program. The traffic assignment problem deals with the assignment of an origin-destination trip matrix to a transportation network so as to minimize each user's travel time (or cost). In traffic assignment, the arcs have a fuzzy upper bound that arises from highway congestion effects, whereas our problem has a fuzzy lower bound arising from power shortfalls. Both problems can be formulated as a solution to a convex program over network flows. The heuristic used to solve the locomotive scheduling borrows from both the incremental assignment method and the Frank-Wolfe algorithm used to solve the traffic assignment problem.

Review of Incremental Assignment and Frank-Wolfe

In incremental assignment, we start with zero flow on the network and choose a number of increments, n. The algorithm, then, in each of n iterations, greedily assigns $1/n$ of the total flow along each shortest path from origin to destination. Because the cost function is nonlinear, these shortest paths may change with each iteration (*8*). Although this algorithm can be set to maintain integrality of the solution, has intuitive appeal, and is easy to implement, it has a number of shortcomings. First, it does not always work, as shown through counterexamples by Ferland et al. (*9*). Second, if it is to produce reasonable solutions, the number of increments may have to be very large, thus unduly increasing the running time.

The Frank-Wolfe algorithm (*8*) is a feasible direction method and therefore starts with an initial feasible solution and moves to improved solutions, maintaining feasibility throughout. It does this by developing linear approximations to the objective function and by solving linear subproblems to find the correct distances to move in improving directions. With cost minimization and a convex objective function, the Frank-Wolfe method does converge to the optimal solution and is easy to implement on networks. Furthermore, a lower bound to the optimal solution is provided at each iteration. Flows, however, are split between paths; thus integrality is lost. In most traffic assignment problems, convergence is rapid (about five iterations) but may be slowed if the solution is in a highly nonlinear portion of the objective function (*10*).

The Two-Commodity Heuristic Approach

The heuristic presented here obtains a feasible solution to the problem through incremental assignment, and then obtains improvements through a feasible direction method. Unlike Frank-Wolfe, it maintains integrality and exploits the integrality of the problem by moving one locomotive unit at a time, thus obviating the need for line searches in the feasible direction method.

The heuristic runs in two phases. First, it loads the network by assigning one unit at a time to shortest paths. This is referred to as the GREEDY phase. Second, after the network is loaded, it attempts improvements by sending flows around augmenting negative cycles in an INTERCHANGE phase. These augmenting cycles are similar to the augmenting paths of maximum flow algorithms in that they include both forward and reverse arcs; thus flow can be removed from an arc when going against the flow direction. Both phases are outlined in more detail below:

GREEDY Phase

Step 0. Initialization. Start with zero flow, and compute arc marginal costs at zero flow.

Step 1. Send one unit down the shortest path from any source to the supersink; update arc flows. (Note that the order

in which units are selected will affect the outcome. For the experiments here, the largest units were arbitrarily selected first).

Step 2. Recompute arc marginal costs along that path.

Step 3. If all units have been sent, go to the INTERCHANGE phase, otherwise, go to Step 1.

Note that for large problems, this phase can be speeded up by sending more than one unit at first. Also, saving the shortest path tree rooted at the sink node and reoptimizing it after every assignment (rather than recomputing the shortest path at each iteration) offers another opportunity to speed up this phase of the heuristic.

INTERCHANGE Phase

Step 0. Identify arcs that are candidates for improvement. In the present implementation these are arcs with large negative marginal cost (i.e., trip arcs with insufficient power).

Step 1. Search for a flow-augmenting negative cycle involving some candidate arc. If no negative cycle can be found in the network, stop. Otherwise, go to Step 2.

Step 2. Interchange flows around this cycle and update arc marginal costs. Go back to Step 1.

The interchanges performed in the second phase are generally more complicated than simply sending one unit of flow around the cycle. This is because the interchanges often involve minor HP changes and thus may involve the exchange of two locomotive types. For example, if our two locomotives types have 2000 HP and 3000 HP, respectively, two interchanges that would produce a small horsepower change would be:

1. Add one high-power and remove one low-power unit on the arc that needs additional power (net change of 1000 HP) or
2. Add two low-power and remove one high-power unit (net change of 1000 HP).

Within the heuristic, these interchanges are performed in the following manner:

1. Create an ordered list of arcs that will benefit from more power.
2. Attempt to find an improving interchange involving one of the arcs on the ordered list. This is done as follows:
   - 2a. Select the first interchange type.
   - 2b. REPEAT.
     - Select the first arc.
     - REPEAT.
       - Try to find an improving interchange (flow-augmenting negative cycle) with this arc and interchange type. If one is found, go to Step 3. Otherwise, select the next arc
     - UNTIL all arcs examined.
     - Consider the next interchange type.
     - UNTIL all interchange types have been considered.
3. If we have found an interchange
   - Perform the interchange and update arc marginal costs.
   - Update the ordered list of arcs, go back to Step 2.

   Otherwise, we terminate, because no interchange can be found.

Example

Consider a two-node network with two trip arcs and one bypass arc (Figure 5). The arc costs are shown in Table 1. The locomotive supply includes one high-powered (3000 HP) and two low-powered (2000 HP) locomotives. The greedy phase of the heuristic performs as follows:

1. Send the high-powered unit down arc 1.
2. Send a low-powered unit down arc 2.
3. Send a low-powered unit down arc 1.

We now have 5000 HP on arc 1 and 2000 HP on arc 2. Arc 2 is short of power.

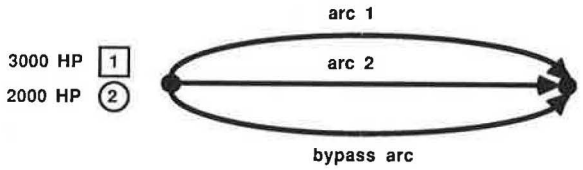

FIGURE 5 Sample network.

TABLE 1 COSTS FOR THE EXAMPLE NETWORK

	arc 1	arc 2	bypass
Parameters			
μ	4200	3000	0
σ	1000	1000	0
c	1	1	0
p	10	10	0
	Costs		
Flow (HP)	arc 1	arc 2	bypass
0	42002	30024	0
1000	33016	21146	0
2000	24102	12837	0
3000	15596	6829.	0
4000	8919.	4837.	0
5000	6166.	5146.	0
6000	6208.	6024.	0
7000	7034.	7003.	0
8000	8005.	8000.	0

Moving to the interchange phase, we see that the high-powered unit on arc 1 can be exchanged with the low-powered unit on arc 2. After this exchange is performed, we are finished, as no more improving interchanges can be seen.

The progress of the heuristic in terms of the arc flows and the objective function is plotted in Figure 6. The method works by first moving in big jumps (whole units) toward the optimal solution, then refining the solution by making smaller jumps (interchanges).

Advantages and Disadvantages of the Heuristic Approach

This double-phase heuristic has several advantages. First, it maintains feasibility throughout. Second, by always moving in an improving direction, the method is intuitively appealing. Therefore, it may lend itself well to interactive use. Third, it is easy to incorporate other side constraints into the framework of this heuristic. Some of these are the following:

1. Prohibition of certain locomotive types from certain sections of track,
2. Assigning newer, more reliable, locomotives to high-priority trains, and
3. Sending locomotives to home shops for scheduled maintenance.

Finally, the heuristic is also quite fast and produces close to optimal results in several test problems.

The disadvantages of this method are, first, its heuristic nature: optimality is not guaranteed. In addition, the complexity of the interchange phase increases as the number of the commodities is increased beyond two. This was not a problem in the case study reported later but may present difficulty in other applications.

LOWER BOUNDS

Several lower bounds were derived to test the performance of the heuristic. A lower bound may be (a) an optimal solution to a relaxed version of the primal problem, (b) a dual feasible solution, or (c) some combination of the foregoing, such as a dual feasible solution to a relaxed version of the primal. Two lower bounds were derived for the problem discussed here.

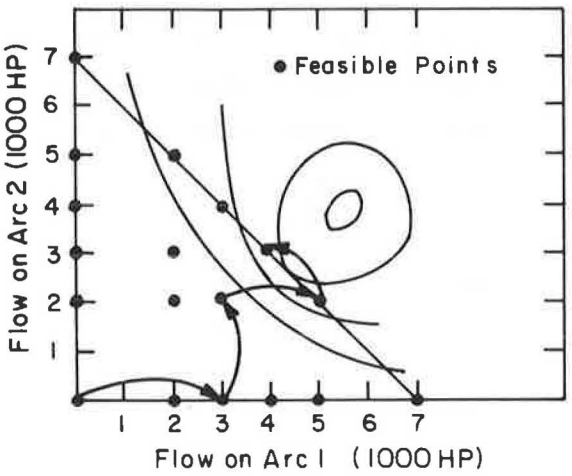

FIGURE 6 Progress to the best heuristic solution.

Frank-Wolfe Relaxation

By relaxing the integrality constraint in the original problem, we obtain a single-commodity (horsepower) network flow problem with convex objective function. This can be solved with the Frank-Wolfe (convex combinations) method already reviewed. The Frank-Wolfe method provides both a feasible solution and lower bound on the relaxed problem at every iteration. This lower bound on the relaxed problem will, naturally, also provide a lower bound on the original minimization problem in the following manner:

$$\text{heuristic solution} \geq \text{optimal solution}$$
$$\geq \text{optimal solution to relaxed problem}$$
$$\geq \text{lower bound to relaxed problem}$$

Unfortunately, a complete relaxation of the integrality constraint in this manner may lead to a large gap between the optimal solution and the optimal solution to the relaxed problem. Such a gap makes it difficult to evaluate the performance of the heuristic.

Greatest Common Factor (GCF) Relaxation

This relaxation is based on the following observation: Any feasible solution will have a horsepower flow in each arc that is a multiple of the greatest common factor (GCF) of the horsepower ratings. For example, if there are two locomotive types rated at 2000 HP and 3000 HP, the flow on each arc will be a multiple of 1000 HP. If there are three locomotive types with ratings of 1750 HP, 2000 HP, and 3000 HP, the flow on each arc must be a multiple of 250 HP. Any other horsepower flow is infeasible because it cannot possibly be produced as a combination of locomotive flows.

We can use this observation to transform the original network problem with convex nonlinear cost function to a conventional linear network flow problem with integral upper bounds on the arc flows. This latter problem is easy to solve. The steps in the transformation are

1. Let b = GCF of the locomotive power ratings.
2. Transform the cost function by making it piecewise linear with breakpoints at multiples of b. See Figure 7. Note that the cost function remains convex and we change its value only at points that cannot be generated by any combination of locomotives.
3. Create n arcs (one for each division in the cost function) for each original arc in the network (Figure 8). (We do not need to add additional constraints because the cost function is still convex, thus the arcs will be loaded in the correct order).
4. Because our sources, sinks, and bounds are all multiples of b, we can scale flows down by a factor of b without losing integrality. The solution to this problem, when scaled back up, will be a multiple of b.

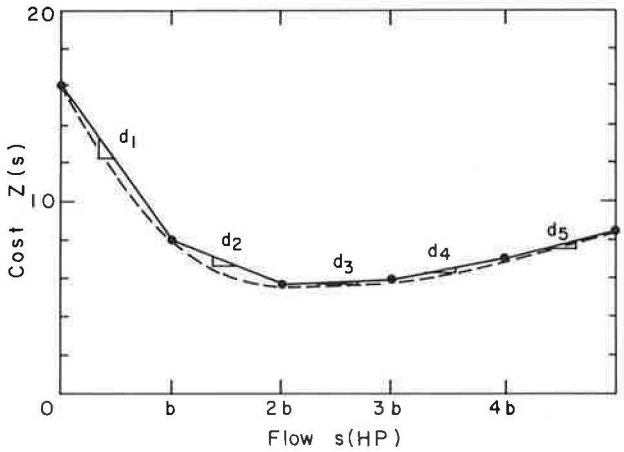

FIGURE 7 Piecewise linearization of the arc cost function.

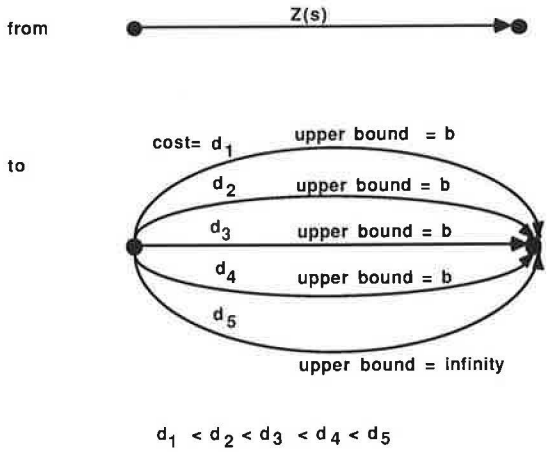

FIGURE 8 Transformation to a piecewise linear cost function.

5. We now have a conventional minimum cost network flow problem that will provide a valid lower bound on the original problem because (a) all feasible solutions to the original problem are feasible in this problem and (b) the cost function was changed only at infeasible points in the original problem. Note, however, that a feasible solution in this problem may not be feasible in the original problem. An example would be an arc flow that is greater than zero but less than the power rating of the smallest locomotive.

TEST PROBLEMS AND RESULTS

The heuristic was implemented in FORTRAN on a MicroVax II running MicroVMS 4.5 and tested on 19 problems of four time-space network configurations (Table 2). The smallest of these networks is shown in Figure 1, and Problems L1–L8 were drawn from an actual 3-day train schedule for the Grand Trunk Western Railroad.

The logistic form of the cost function was used in all cases. The power requirements varied from 3000 HP to 15,000 HP on the trip arcs, and locomotive supplies were fixed to be "barely adequate." The basic networks had $p/c = 10$, $c = 0.5$, $\sigma/\mu = 0.20$, but these were systematically varied in some of the test problems. Table 2 shows these parameters and results for the various networks. The heuristic-optimal values of the objective function behaved reasonably, with the following configurations having increased costs over the baseline (Problems S1, L1):

- Higher penalty term: Increasing the penalty term tenfold approximately doubled the objective function (Problems S3, L3).
- Higher standard deviation: Increasing σ/μ from 0.1 to 1 also increased the cost substantially (Problems S5, L5).
- Lower power supply: Because the initial power supply was "barely adequate," reducing it increased costs somewhat as more trains were underpowered (Problems S6, L6).

We would expect average running times to be a function of both the number of locomotives supplied and of the size of the network. In the cases here, as the networks became larger, running times seemed to be $O(nt)$ where n is the number of nodes and t the number of locomotives. They were reasonable in all cases, ranging from 0.5 sec for the smallest network to 63 sec for the largest. This is acceptable because it is envisioned that in an operating environment, the model will be run about once per 8-hr shift rather than continuously.

The numerical results were normalized to the best lower bound found, which was the GCF lower bound. The results of the heuristic were, on average, within 3 percent of this bound. These normalized results are shown in Table 2. Problems with a flat objective function (low p/c and high σ/μ – problems S2, S5, L2, L5) tended to perform better with results, on average, within about 1 percent of the lower bound. Conversely, problems with a highly nonlinear objective (S3, S4, L3, L4) give results that were on average only within 6–7 percent of the lower bound. The Frank-Wolfe algorithm also tended to have poor convergence on these problems.

FURTHER WORK

We have developed a model that deals explicitly with the uncertainty in power requirements. Moreover, the heuristic used to solve this model is promising because it is both fast and fairly accurate. Further research should focus on improvements to the heuristic and incorporation of schedule variability.

The present implementation of the heuristic does not optimize speed. Some improvements, mentioned earlier, include sending more than one unit at a time in the early stages of the heuristic, and keeping and reoptimizing a shortest path tree rooted at the supersink, rather than recalculating shortest paths for each iteration. Another improvement to the heuristic would be the incorporation of additional side constraints and provision for more than two commodities.

Of possibly greater interest is the incorporation of schedule variability. Although the model now assumes a fixed schedule, one way to do this would be to incorporate the heuristic into an interactive system that displays where and when shortages of locomotives are likely to occur, and then allowing the user to adjust the schedule accordingly before running the heuristic

TABLE 2 TEST RESULTS

Trial	Trips	arcs	nodes	kHP	p/c	σ/μ	GCF Cost	Normalized Costs: greedy	int	FW	FWLB	GCF
T1	3	15	10	13	10	0.1	18.8	1.017	1.017	0.998	0.951	1
T2	3	15	10	13	10	0.1	18.6	1.069	1.028	0.991	0.981	1
S1	9	42	25	52	10	0.1	123	1.025	1.018	0.995	0.984	1
S2	9	42	25	52	5	0.1	105	1.016	1.013	0.999	0.991	1
S3	9	42	25	52	50	0.1	249	1.066	1.044	0.988	0.978	1
S4	9	42	25	52	10	0.05	105	1.053	1.047	0.994	0.881	1
S5	9	42	25	52	10	1	254	1.002	1.000	1.000	0.996	1
S6	9	42	25	37	10	0.1	184	1.038	1.000	0.999	0.995	1
S7	9	42	25	67	10	0.1	110	1.065	1.008	0.996	0.979	1
M1	35	153	88	137	10	0.1	1156	1.034	1.011	1.013	0.976	1
M2	35	153	88	168	10	0.1	1085	1.035	1.031	1.004	0.976	1
L1	102	404	239	283	10	0.1	1531	1.065	1.035	1.020	0.933	1
L2	102	404	239	283	5	0.1	1191	1.027	1.021	1.004	0.968	1
L3	102	404	239	283	50	0.1	3598	1.133	1.104	1.039	0.914	1
L4	102	404	239	283	10	0.05	1414	1.117	1.067	1.034	0.886	1
L5	102	404	239	283	10	1	2475	1.014	1.010	1.005	0.992	1
L6	102	404	239	253	10	0.1	1577	1.062	1.048	1.017	0.960	1
L7	102	404	239	309	10	0.1	1484	1.068	1.046	1.015	0.927	1
L8	102	404	239	406	10	0.1	1454	1.041	1.020	1.009	0.919	1
							average	1.050	1.030	1.006	0.957	1

Trips = number of trips in this network

kHP = total horsepower supply (thousands HP)

p/c = ratio of penalty to cost term

σ/μ = coefficient of variation for power demand

GCF Cost = Total cost (thousands $) for GCF relaxation

greedy = total cost after GREEDY phase / GCF Cost

int = total cost after INTERCHANGE phase / GCF Cost

FW = total cost of Frank-Wolfe solution / GCF Cost

FWLB = total cost of Frank-Wolfe lower bound / GCF Cost

GCF = total cost of GCF relaxation / GCF Cost

run time = total running time for the heuristic, excluding input/output and computation time for relaxations.

again. However, to adjust schedules within the algorithm will require consideration of systemwide train scheduling and customer demand, both of which are very difficult to quantify.

ACKNOWLEDGMENT

This research was funded by the Grand Trunk Western Railroad.

REFERENCES

1. A. A. Assad. Models for Rail Transportation. *Transportation Research*, Vol. 14A, 1980, pp. 205–220.
2. A. A. Assad. Analytical Models in Rail Transportation: An Annotated Bibliography. *INFOR*, Vol. 19, No. 1, 1981, pp. 59–80.
3. E. R. Peterson. Operations Research and Rail Transportation. In *Scientific Management of Transport Systems* (N. K. Jaiswal, ed.), Elsevier North-Holland, New York, 1981, pp. 37–51.
4. T. E. Bartlett. An Algorithm for the Minimum Number of Transport Units to Maintain a Fixed Schedule. *Naval Research Logistics Quarterly*, Vol. 4, No. 2, 1957, pp. 139–149.
5. R. S. McGaughey, K. W. Gohring, and R. N. McBrayer. Planning Locomotive and Caboose Distribution. *Rail International*, Vol. 4, 1973, pp. 1213–1218.
6. M. Florian, G. Bushnell, J. Ferland, G. Guerin, and L. Nastansky. The Engine Scheduling Problem in a Railway Network. *INFOR*, Vol. 14, No. 2, 1976, pp. 121–138.
7. J. M. P. Booler. The Solution of a Railway Locomotive Scheduling Problem. *Journal of the Operational Research Society*, Vol. 31, 1980, pp. 943–948.
8. Y. Sheffi. *Urban Transportation Networks*. Prentice-Hall, Englewood Cliffs, N.J., 1985.
9. J. A. Ferland, M. Florian, and C. Achim. On Incremental Methods for Traffic Assignment. *Transportation Research*, Vol. 9, 1975, pp. 237–239.
10. S. E. Mimas. *Equilibrium Traffic Assignment Models for Urban Networks*. Master's thesis. Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge, 1984.

System-Optimal Trip Scheduling and Routing in Commuting Networks

GANG-LEN CHANG, HANI S. MAHMASSANI, AND MICHAEL L. ENGQUIST

A time-space network formulation is presented for the system-optimal assignment to departure times and routes of traffic flows from multiple origins to a common destination. Time is discretized, and congestion is represented using simplified deterministic queuing stations. The solution minimizes total travel time in the system subject to arrivals at the destination taking place within a specified time interval. Alternatively, a formulation is presented for the minimization of a total cost measure consisting of a weighted sum of the users' travel time and schedule delay. The solution can be obtained using efficient and widely available pure network optimization algorithms. A numerical application is presented to illustrate the methodology, including a network generator developed for this purpose.

Peak-period congestion continues to be a severe daily annoyance in most metropolitan areas where large volumes of commuters desiring to arrive at their destinations within a narrow time interval compete for limited transportation system capacity. No major innovations for combating congestion seem to have emerged in the past 15 yr. Recently, the potential of advanced information and communication technology for congestion control appears to have rekindled interest and effort in this problem. However, the design and evaluation of various control strategies require deeper understanding of the systems' complex nature and methodologies with the capability to deal effectively with time-dependent flows in congested networks.

Several contributions have addressed the problem of finding a time-dependent flow pattern that satisfies dynamic user equilibrium conditions in an idealized system consisting of a single route containing a bottleneck and connecting a single origin-destination pair (*1–9*). Extensions have included multiple routes and alternative assumptions on the system's configuration or behavioral mechanisms underlying tripmakers' decisions (*10–14*). The day-to-day dynamics of the interaction between commuter decisions and congestion in a traffic system have also received some attention recently, using simulation experiments (*15*) and observational studies (*16–19*). Relatively little attention has been directed toward the problem of solving for time-dependent traffic patterns that are in some sense optimal from a total system cost standpoint.

Previous studies dealing with time-varying system-optimal traffic patterns have followed one of two lines: (a) optimizing the traffic-generation patterns in a given system with a single route (and one bottleneck) or (b) assigning known time-dependent flows from multiple origins to a single destination to the links of a network so as to minimize total system cost (travel time). Research along the first line consists of analytical derivations or discussions of system-optimal departure patterns, in connection with the aforementioned dynamic user equilibrium studies in an idealized system in which a number of commuters from the same origin are trying to arrive at a common destination at the same time (*6, 9*). Extension to the scenario of staggered work hours has been described elsewhere (*20*). Discussions of system-optimal departure patterns are also given by Hendrickson et al. (*21*), Fargier (*5*), and Newell (*14*).

Contributions along the second line of research are limited to situations where the time-dependent departures from multiple origins to a single destination are known, and congestion is modeled using link performance functions intended for static traffic assignment applications. As such, these are direct extensions of the standard system-optimal network assignment formulations. Merchant and Nemhauser (*22, 23*) formulated the problem as a discrete time, nonlinear but nonconvex math program where the objective is to minimize the total travel time spent by the given trips in the network. A recent paper by Carey (*24*) reformulates that problem as a convex nonlinear program, which is of course more attractive computationally than the previous formulation, and discusses possible extensions to more general situations.

The present paper addresses a more general system-optimal state, which includes not only the assignment of known time-varying flows to the links of a commuting network but also the determination of the corresponding optimal time-dependent traffic-generation patterns from the various origins, given constraints on desired arrivals at the destination. The paper presents a methodology for the system-optimal assignment of commuters to departure times and routes subject to specified constraints on acceptable arrivals. It consists of a time-space network formulation that can easily be solved using existing efficient network flow programming codes. The scope is still limited to commuting systems with a single major destination, such as a CBD or other large industrial or business employment center, but allows for multiple routes and multiple origins. It is intended primarily as a tool to explore the potential benefits that could be achieved from information-related and demand-side strategies aimed at reducing congestion.

The next section presents the conceptual framework and principal features of the proposed approach, followed by a detailed formulation for the time-space network of principal activities for a simple commuting system with a single route and a single origin. Extension of the formulation to more general situations with multiple routes and multiple origins is

G.-L. Chang, Department of Civil Engineering, University of Texas at Arlington, Arlington 76010. H. S. Mahmassani, Department of Civil Engineering, University of Texas at Austin, Austin 78712. M. L. Engquist, Management Science and Information Systems, University of Texas at Austin, Austin 78712.

discussed in a later section, followed by presentation of a numerical illustration. Finally, concluding comments and possible extensions are addressed.

MODEL FORMULATION

This section presents the key features of the time-space network formulation, including the representation of the traffic system. The context considered here is a commuting corridor surrounded by residential areas. For convenience and ease of presentation, we start with the simplest scenario, shown in Figure 1, where only one highway facility exists in the corridor for use by residents from adjoining areas in their daily commute to the same work destination. Concern here is primarily with the inbound, or home-to-work, direction.

For the purpose of analytical representation, the highway facility is conceptually divided into a number of sections with each including, at most, one entry ramp. The time spent in any section or ramp depends on the facility's service characteristics and the generated time-dependent flow patterns. In this formulation, the entire system is viewed as a network of queuing stations. Using a simplified representation adopted in several papers dealing with dynamic traffic assignment (*7, 10, 11, 25*), each highway section can be viewed essentially as a potential bottleneck with a given service rate (capacity). If the flow is less than this service rate, then only the free-flow travel time is incurred on the corresponding section; otherwise, a waiting time in queue is incurred, representing the excess travel time resulting from congestion. Likewise, entrance and exit ramps can also be modeled as typical, deterministic queuing stations with service rates depending on each ramp's physical capacity and control system (e.g., in the event of ramp metering). Further detail is given hereafter.

Because we are dealing with only one day's process at a time, the system is considered for a given duration that includes the earliest and latest possible (and meaningful) departure times. Time is discretized into equal intervals of a suitable small length Δt (in the order of a few minutes). The network formulation of the system-optimal dynamic assignment problem can be obtained by analogy to the time-space relation of individual vehicles traveling from the origin to the destination. The network is akin to a trans-shipment problem where it is desired to send flow units (tripmakers) from a set of supply points (origins) to a demand point (destination) at minimum cost using a network of arcs and nodes. The arcs correspond to activities, generally involving an expenditure of time or other cost, incurred on a per-flow unit basis; whereas the nodes correspond to the beginning and/or end of activities. The activities here include but are not limited to movement on physical highway links.

There are basically three categories of activities in the formulation: departure from origins, travel on links (including queuing at bottlenecks), and arrival at the common destination. Described are the formulation of each category as a separate subnetwork and then their integration to form the entire network for a given problem. The presentation is primarily graphical in nature; the notational convention used in the network formulation follows the work of Klingman and coworkers (*26*) and is summarized in Figure 2.

Formulation of the Departure Activity

For a given origin (supply) node, each discrete departure time alternative (of length Δt) is represented by a node, as shown in Figure 3, with the earliest and latest possible departure times denoted respectively by nodes DA_1 and DA_n, and the intermediate nodes labeled sequentially. The total number of commuters originating from location A constitutes the total supply (in trans-shipment terminology) for node A, which is connected to each of the associated departure nodes by a unique outgoing arc. The flow on arc (A, DA_k), $k = 1, \ldots, n$, obtained in the solution corresponds to the number of users that depart from A in the kth time slice; the set of these flows thus represents the optimal departure pattern of users at this location. The departure time here is taken at the entry of the highway facility. Thus arc (A, DA_k) corresponds to local travel from origin A to the facility. For simplicity, but without loss of generality, we assume that the time cost of this travel is a constant, T, associated with each arc (A, DA_k), $k = 1, \ldots, n$. This cost is not necessary from the perspective of model operation, however, because users at a given origin are uniquely assigned to an entry point. A more detailed formulation could let this assignment be determined in the optimal solution.

Commuters may have to join a queue or be otherwise delayed at the entry point. The horizontal arc emanating from each departure node (see Figure 3) is designated to carry only those commuters actually entering the highway in that given interval. The upper bound of flow through each arc, denoted as $C1$, is used to control the maximum entry rate, reflecting either physical capacity restrictions or the effect of traffic control devices. The associated arc cost $T1$ represents the travel time to the next "state," a congested location in this case. Because of the preceding capacity constraint, commuters departing simultaneously (i.e., in the same time slice) may not all be allowed onto the facility at the same time. Thus each departure node DA_k is connected to the next departure alternative DA_{k+1} by arc (DA_k, DA_{k+1}), shown vertically in Figure 3. This arc will carry the excess number of commuters at DA_k who could not be served in a given time slice. The resulting waiting time is then captured by the arc cost,

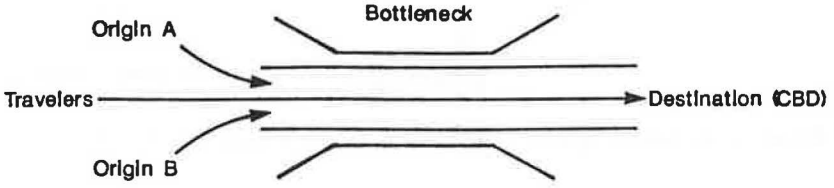

FIGURE 1 An idealized commuting system for analysis.

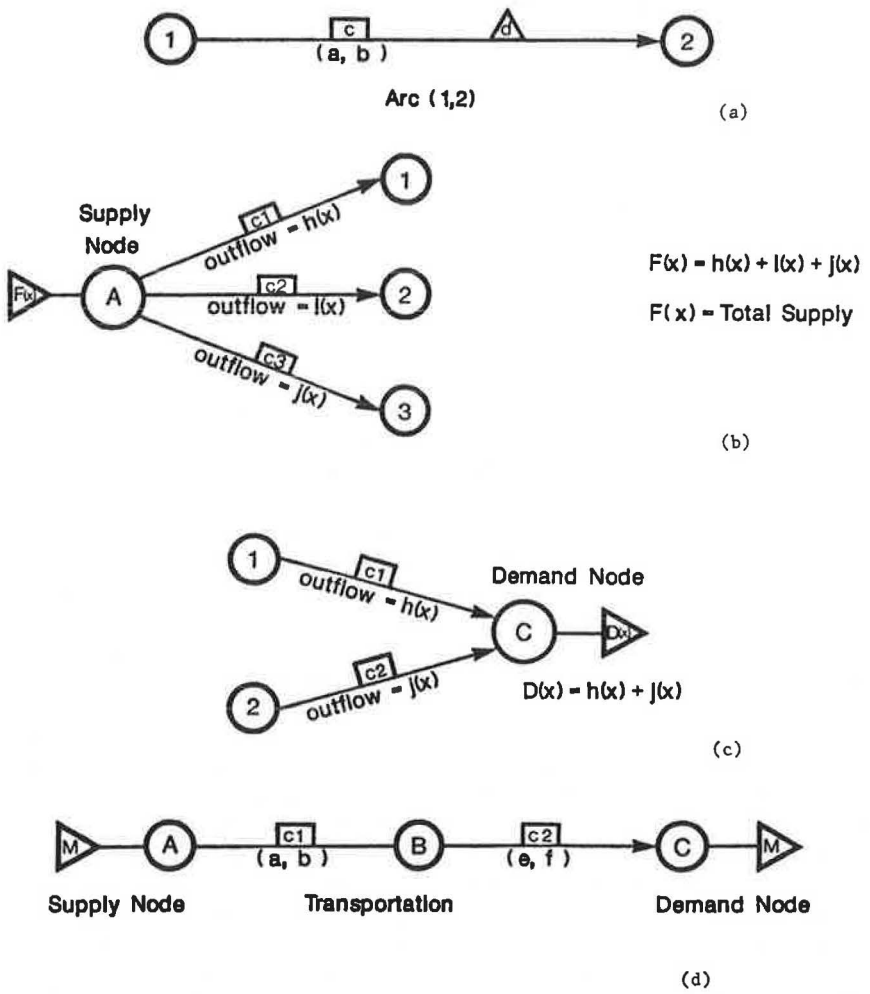

FIGURE 2 Notation for the network formulation.

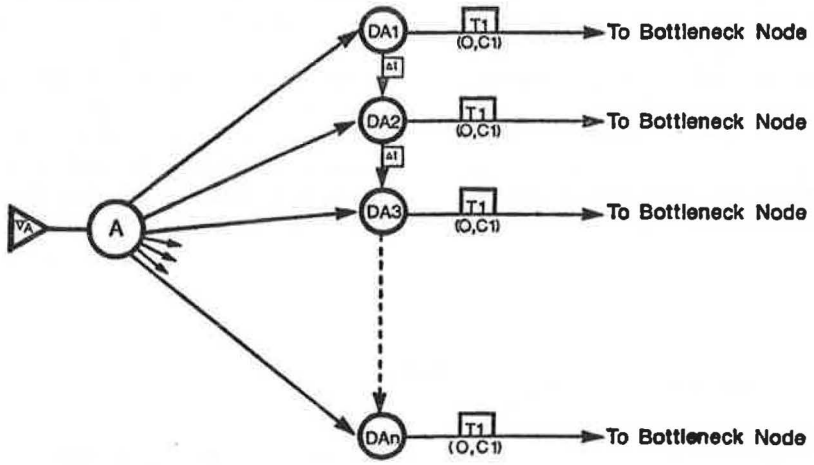

FIGURE 3 Graphical representation of the departure activity subnetwork.

Δt, equal in magnitude to the unit departure time slice. An upper bound on the flow on these waiting arcs can be specified to reflect physical storage capacity limitations or policy decisions.

Other trip generation sources, such as origin B in Figure 2, can also be represented in a similar manner.

Formulation of Congested Locations

It should first be noted that this representation is not intended to capture the details of the traffic flow phenomena taking place on the facility or the formation and dissipation of physical queues in the system. It is principally an approach to calculate realistic travel times under congested conditions for each link in the context of a pure network formulation of the system-optimal, time-varying assignment problem. There may not necessarily be a physical queue of stopped vehicles in the actual system, even if traffic is highly congested. Instead, users may be forced to slow down along some sections with particularly high concentrations. As noted in the previous section, such congested locations are modeled as queuing stations and are formulated as follows.

Two sets of nodes and associated connecting arcs are proposed to model congested locations. As illustrated in Figure 4, the first set of nodes, denoted as BI_k, $k = 1, \ldots, n$, is used to represent the arrival at the bottleneck, with nodes BI_1 and BI_n representing the earliest and latest arrival times, respectively. Each node in the set $\{BI_k, k = 1, \ldots, n\}$ is connected by an arc (BI_k, BO_k), shown horizontally in Figure 4, to a unique corresponding node in the second set $\{BO_k, k = 1, \ldots, n\}$ designated to model the exit from the bottleneck. The upper bound on flow in each of these arcs is defined by the bottleneck's service rate S (i.e., the number of users allowed to go through in a given time slice Δt). The associated arc cost $T2$ is the time through the section in the absence of congestion. As in the formulation of the departure activity, vertical arcs (in Figure 4) are specified from each node BI_k to node BI_{k+1}, $k = 1, \ldots, n$, to carry the queuing flow, with arc cost again equal to the unit waiting time Δt. Note that no such arcs are shown in Figure 4 between consecutive BO_k nodes because queuing should not occur immediately upon exit from the bottleneck.

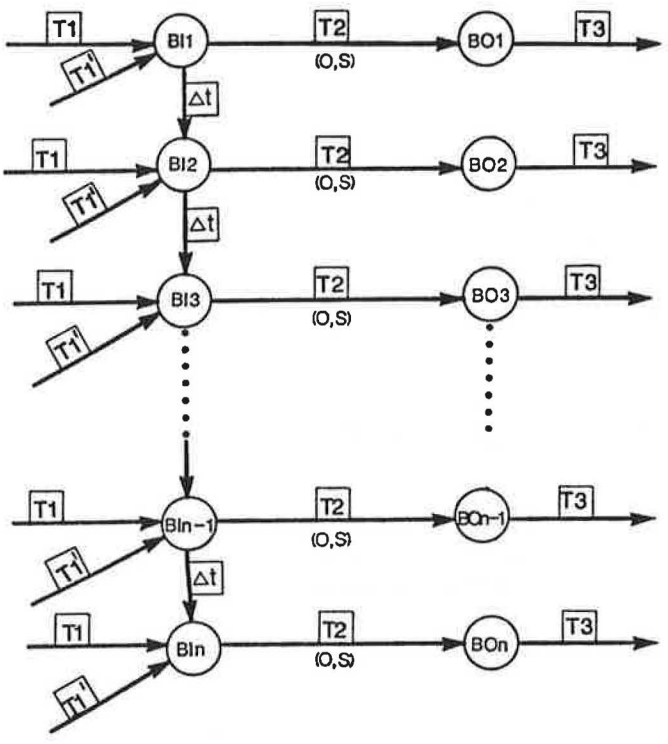

FIGURE 4 Graphical representation of the bottleneck area subnetwork.

Finally, all arcs incident to the BI_k nodes, except for the queue-carrying vertical arcs, are intended to carry commuters arriving at the bottleneck. Because each of these arcs originates at a departure node, the specification of the associated arc cost and upper bound on flow must be consistent with that specified in the departure formulation. Likewise, the arcs leaving each of the BO_k nodes carry the flow departing from the bottleneck. No upper bounds are shown for these arcs in Figure 4 because the flow has already been regulated by the service rate S of the bottleneck, although we may want to specify such upper bounds for more general systems.

Formulation of the Arrival Process

The formulation of the arrival process subnetwork depends on the explicit definition of the system optimum sought. So far, we have implied that the desired solution would minimize total system cost, calculated as the sum of all arc costs incurred by the assigned flows. These arc costs have in turn been specified as either uncongested travel times or delays due to congestion at bottlenecks. We need to address further the costs contributed to the objective function incurred in conjunction with the arrival process, as well as the constraints that need to be satisfied by this process. Now considered here are two basic alternative formulations reflecting different assumptions about the users' preferences or cost function: a satisfying formulation and a utility maximization one. We also describe how variants can be modeled.

Before describing these two formulations and the underlying assumptions, it is useful to consider, qualitatively, the nature of the departure patterns that can be expected in the solution. First, it must be recognized that it is generally not feasible for all users to arrive simultaneously (in a single time interval Δt) at the desired destination. There is a minimum duration for the arrival period that is governed by the capacity of the bottlenecks. If users were allowed to arrive at any time before the official work start time, then one can almost always find a solution that minimizes the total travel time in the system and that involves absolutely no queuing (i.e., all the vertical arcs in the network formulation would have zero flows). Unfortunately, such a solution would likely exhibit so much spread in the departure (and arrival) pattern that it would be meaningless. In other words, we would have a trivial problem if there were no constraints on either the range of possible departures or the range of possible arrivals and if travel time were the sole consideration in the objective function.

The first meaningful formulation we consider here constrains all arrivals to take place within a specified time band. It is consistent with empirical evidence that workers like to allow some extra time prior to the official work start time (*18*,

27); as such they may be indifferent to arrivals if they are within a reasonable time band. Referring to a recent paper by Mahmassani and Chang (12), it can also be noted that this formulation would yield the "best" departure pattern among the multiple patterns that satisfy boundedly rational user equilibrium conditions for a given value of the indifference band, assumed to be identical across users.

The second formulation places a penalty on the time between actual arrival at the destination and the work start time, also referred to as schedule delay. Thus the user's utility function would include both the travel time and the schedule delay, the latter multiplied by a weight reflecting its valuation relative to travel time. This type of function would be consistent with the classical microeconomic view of this problem, as presented by Vickrey (9) and by Hendrickson and Kocur (7). The solution would involve a trade-off between travel time and schedule delay, which would lead to spread-out departures and arrivals and thus high schedule delays. We next describe the network representation of the two cases, starting with the satisfying formulation.

In all cases, we define a set of arrival nodes D_r, $r = 1, \ldots, n$, that define the arrival time alternatives, generally corresponding to the departure nodes DA_1 through DA_n. The satisfying feature is included in the formulation by specifying the subset of consecutive nodes from D_k to D_n as the acceptable range of arrival times, as shown in Figure 5. All commuters are supposed to traverse at least one of those nodes to end their trips. Each of these nodes is connected to a supersink (or total demand) node DE, the common destination, by arcs (D_k, DE), with upper bound on flow denoted by $C3$ in Figure 5. This value may be the same across these arcs, representing the physical constraint for the arrival rate, or may vary across arcs to reflect the operation of traffic control devices. Each of the feasible arrival nodes is connected to the next one by a vertical arc with cost Δt to convey the queuing flow.

Unlike nodes D_k through D_n, nodes D_1 through D_{k-1} are not connected to the supersink. Vertical arcs with very high costs (M) are specified between each pair (D_i, D_{i+1}), $i = 1, \ldots, k-1$. This will prevent flows in the network from taking paths ending in an unsatisfactory arrival time (i.e., outside the band) unless there is no feasible solution for the specified arrival time band. Obviously, nonzero flow on any of the "big M" arcs in the final solution will be a sign of unfeasibility, which could be resolved by widening the acceptable arrival band to include additional arrival nodes.

Given the foregoing formulations of the three principal activities, the network for the entire system can be constructed through careful integration of the three subnetworks, as shown in Figure 6 for the idealized commuting system of Figure 1. We next describe how the formulation of the arrival process can be modified to represent the utility maximization case.

Utility Maximization Formulation

As noted previously, the total trip cost of commuters depends, under this rule, on the specification of the utility function. A commonly used specification in this context involves a trade-off between trip time and schedule delay, of the form:

$$TC_{i,t} = (a \cdot TR_{i,t}) + (\delta \cdot b \cdot SDE_{i,t}) + (1 - \delta) \cdot c \cdot SDL_{i,t} \quad (1)$$

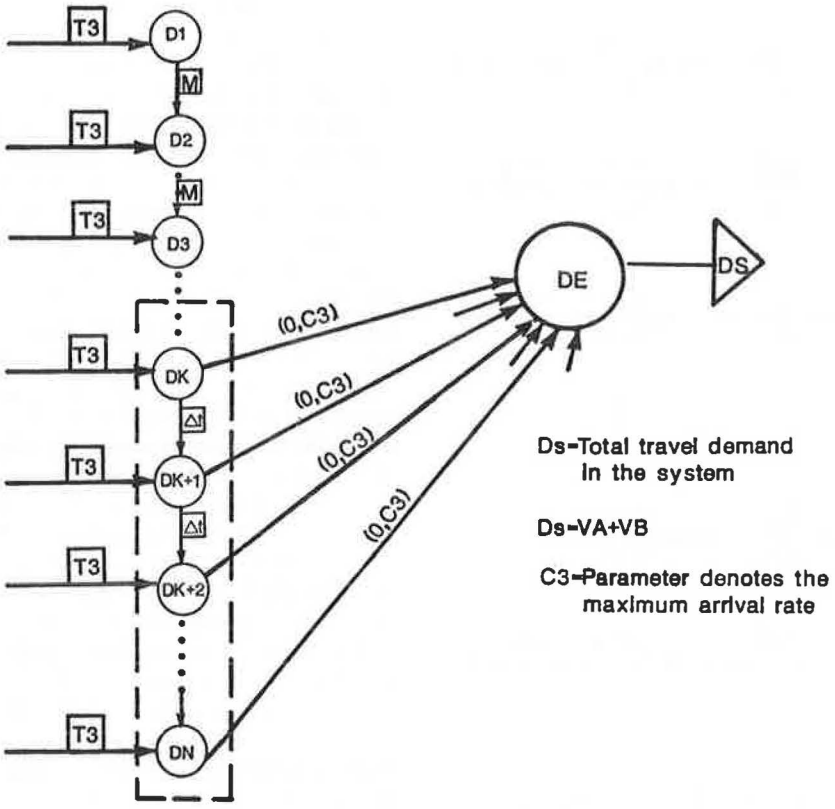

FIGURE 5 Graphical representation of the satisfying formulation for the arrival process subnetwork.

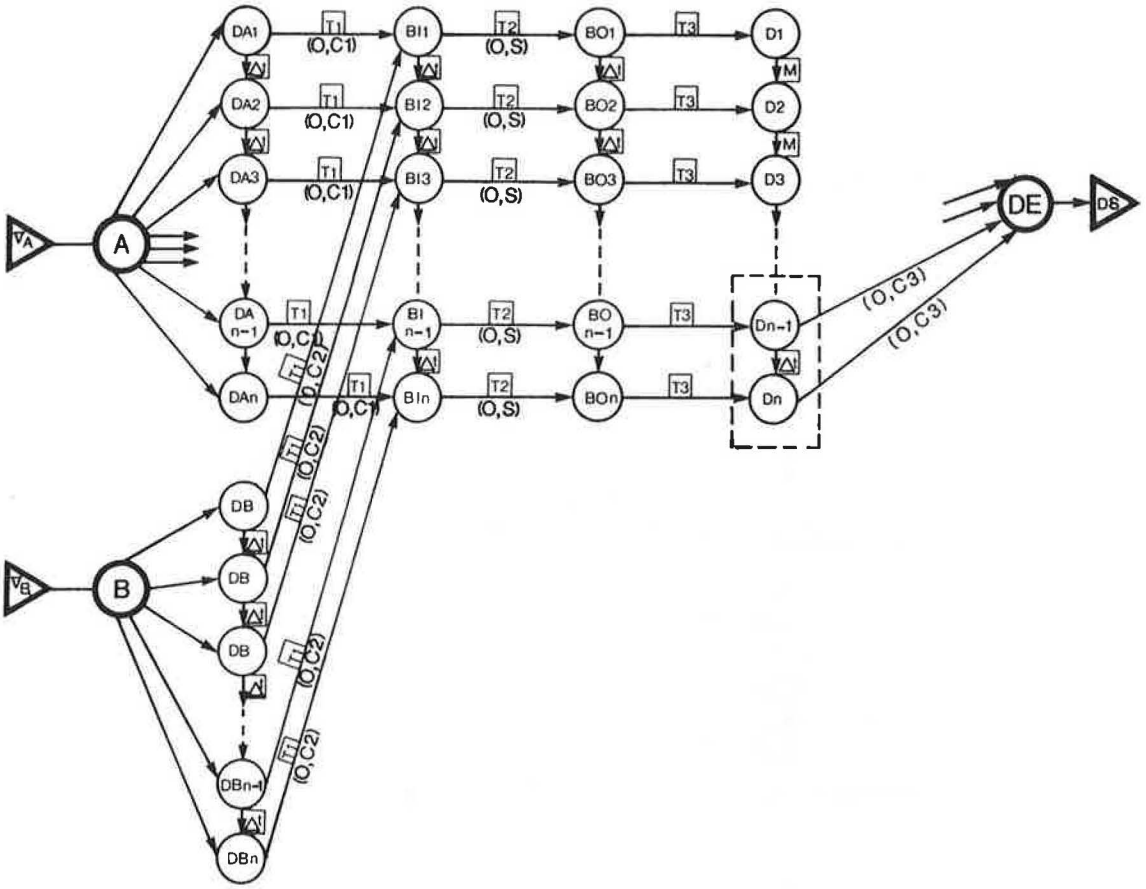

FIGURE 6 Example network formulation for the idealized commuting system.

where

$TC_{i,t}$ and $TR_{i,t}$ = the total travel cost and travel time, respectively, incurred by flow unit i departing at time t;

$SDE_{i,t}$ and $SDL_{i,t}$ = the schedule delay for early and late arrival, respectively, relative to the desired arrival time;

a, b, and c = parameters capturing the disutility of a unit of travel time, schedule delay for early and late arrival, respectively (it is convenient to set $a = 1$ and scale b and c accordingly); and

δ = a binary variable equal to 1 for early arrival and to 0 for late arrival.

We assume hereafter that all users are identical in terms of the parameters of the preceding function. To capture this trade-off between schedule delay and travel time, the network formulation of the arrival process can be modified as shown in Figure 7. Let node AR_n denote the work starting time; the other arrival time nodes form two groups: AR_1 to AR_{n-1}, and LAR_1 to LAR_m, which correspond to early and late arrivals, respectively. Only node AR_n is connected to the total demand node DE to force all flows, except those arriving at the AR_n node via a horizontal travel arc, to traverse the needed number of queuing arcs to reach AR_n before they can terminate their trips. The summation of the costs incurred on these arcs yields the schedule delay cost. In this formulation, the specification of the arc cost consists of the time slice Δt multiplied by an appropriate factor consistent with the underlying utility function (Equation 1); for instance, in Figure 7, the multipliers EC and LC are equal to b/a and c/a, respectively.

The solution of the minimum cost trans-shipment problem under the preceding specification of the arc costs will thus be optimal for the system in terms of minimizing the total disutility of system users. Several variants are possible here, such as constraining all arrivals to occur within a particular time band. In this case, a large number M can be imposed on all vertical queuing arcs with at least one end outside the band and the schedule delay costs on those entirely within the band (still only node AR_n would be connected to DE). Alternatively, one can represent a utility function combining the behavioral features of both the satisfying and utility maximizing formulation. In particular, an indifference band of acceptable arrivals can be specified where all nodes in the indifference subset are connected to DE and no cost is associated with the vertical arcs connecting nodes in that subset. Vertical arcs outside this band will, however, be assigned a cost equal to the schedule delay disutility (but not large M).

The values of the relative weights of the various cost components would of course have to be determined outside this particular methodology. One use of this formulation is that is allows the systematic investigation of the impact of these relative valuations on the character of the optimal solution and the associated total system costs. However, the assumption of identical valuation across users may be too strong for practical applications.

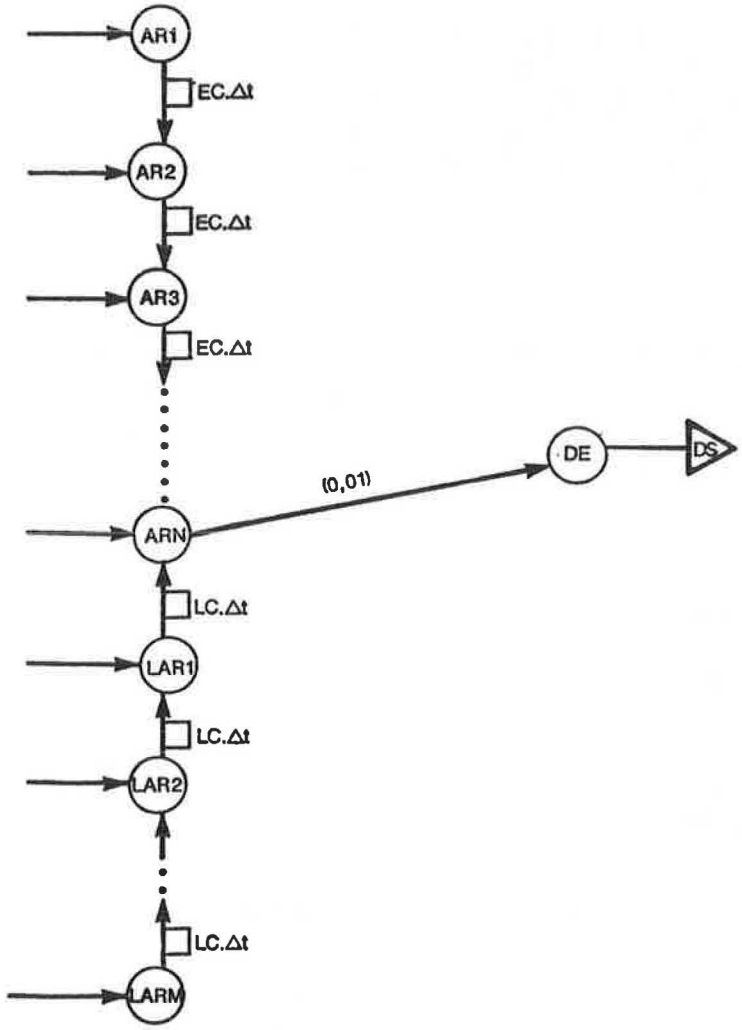

FIGURE 7 Network formulation of the arrival activity under the utility-maximizing decision rule.

EXAMPLES OF MORE GENERAL SYSTEMS

In this section we describe the representation, in the context of the preceding network modeling framework, of more general situations encountered in commuting systems. Still dealing with multiple origins, single destination systems, we first consider multiple bottlenecks (in series) along a single route, then multiple parallel routes. These types of systems have also been considered by Ben-Akiva et al. (*10, 11*) in their study of stochastic user equilibrium time-dependent flows.

Case 1: Multiple Bottlenecks Along a Single Route

Figure 8 depicts an example commuting system with two congested sections, BA and BB, where commuters departing from origin A have to traverse both sections, whereas those from downstream origin B encounter only the second bottleneck, BB. The network formulation for this problem is shown in Figure 9. Two sets of nodes $\{DA_i, i = 1, \ldots, n\}$ and $\{DB_j, j = 1, \ldots, n\}$, as defined previously, represent the feasible departure time alternatives of commuters from origins A and B, respectively. The first bottleneck BA is modeled by a set of node pairs, with each pair $\{(BA_k \text{ and } BA_k^*), k = 1, \ldots, n\}$ as described in the previous section. In the same manner, activities in the second bottleneck are represented by the set of node pairs, $\{(BB_k, BB_k^*), k = 1, \ldots, n\}$. The cost and upper bound associated with each arc are defined as shown in Figure 9, in a manner similar to the basic model of the previous section. Note that the set of arcs $\{(BA_k^*, BB_k), k = 1, \ldots, n\}$ corresponds to travel between the end of the first bottleneck section and the beginning of the second; no upper bounds on flow on these arcs need to be specified as these flows are regulated by the upstream bottleneck and no additional generation takes place in that sector. For the same reason, no vertical arcs connect the BA_k^* nodes. Finally, the arrival process follows the satisfying formulation illustrated in Figure 5, where the set of nodes $\{AR_t, t = 1, \ldots, n\}$ corresponds to the array of possible arrival times and the subset of those connected to the total demand node represents the presumed acceptable arrival interval. It should be mentioned that the possible departure periods for the two origins A and B are assumed to have an identical length and thus an equal number of nodes, for clarity of presentation. This is not

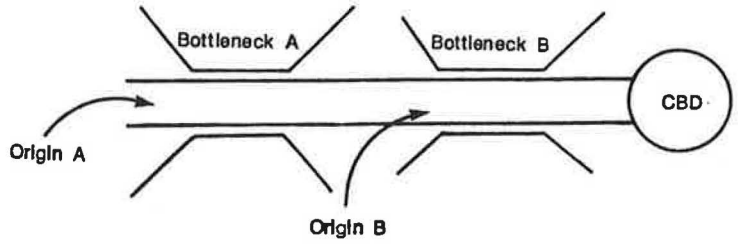

FIGURE 8 Commuting system with two bottlenecks along the single route.

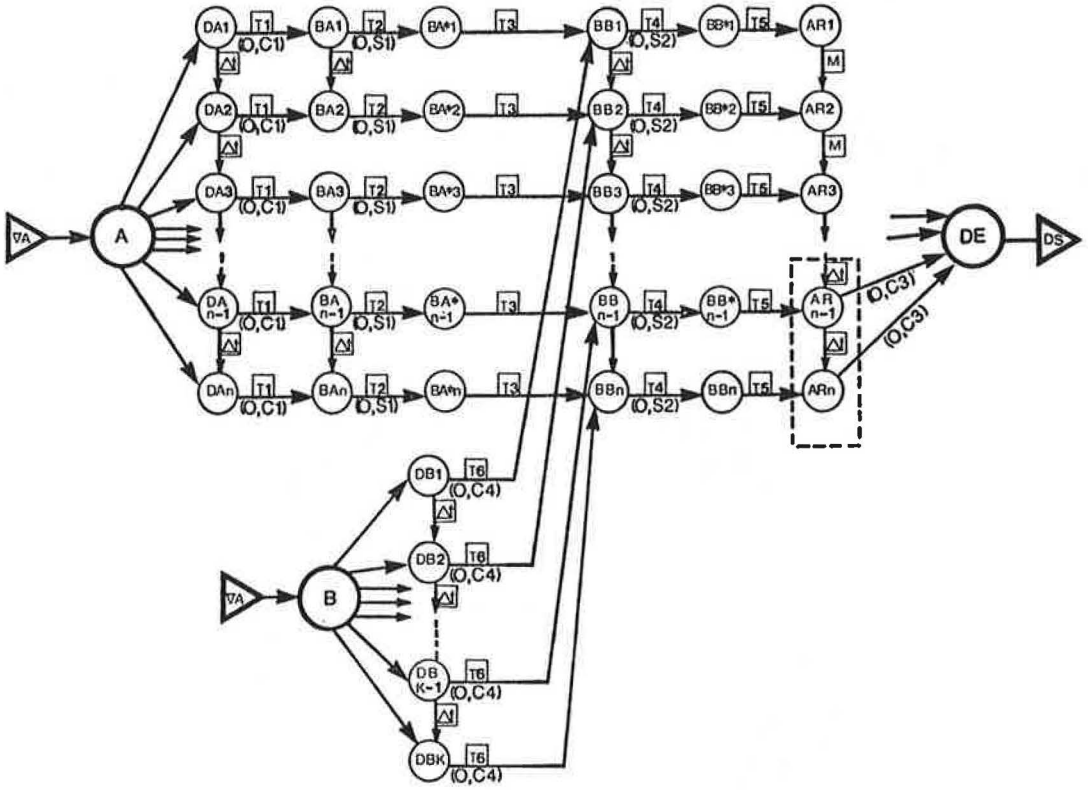

FIGURE 9 Network formulation for commuting system of Figure 8.

necessary, however, as long as they are properly connected to the rest of the network.

With the formulation of Figure 9, the system-optimal departure distribution patterns can be solved using any existing minimum cost, linear network code that implements the network simplex algorithm or its variants. See Kennington and Helgason (*28*) for a discussion of these algorithms.

Case 2: Multiple Parallel Routes

In the commuting system of Figure 10, there are three parallel routes, each containing two bottleneck sections, and commuters can choose their departure time as well as their route. To construct the network formulation of such a system, we can essentially follow the same procedure as in Case 1, with each route being formulated independently as one subnetwork. Then all subnetworks are tied together at both the common supply nodes and arrival nodes.

Figure 11 illustrates the resulting network formulation for this system. Nodes DA_k, DB_k, and DC_k, $k = 1, \ldots, n$ denote the feasible departure period of commuters from location A to travel through Routes A, B, and C, respectively. Node pairs $(A1_k, A1_k^*)$ and $(A2_k, A2_k^*)$, $k = 1, \ldots, n$, represent Bottlenecks 1 and 2, respectively, on Route A. Node pairs $(A3_k, A3_k^*)$ and $(A4_k, A4_k^*)$, $k = 1, \ldots, n$ represent Bottlenecks 3 and 4, respectively, on Route B. Node pairs $(A5_k, A5_k^*)$ and $(A6_k, A6_k^*)$, $k = 1, \ldots, n$ correspond to Bottlenecks 5 and 6, respectively, on Route C. The arrival process is represented as before with a common set of nodes D_1 to D_n for the arrival period, with the subset D_k to D_n defining the acceptable arrival time band.

Again, it should be noted that, for convenience of presentation, the feasible departure periods for the three routes in Figure 11 are assumed to consist of the same number n of time intervals. It is possible to let the feasible departure period vary from route to route. However, attention should be given to the formulation of the arrival period if the length of depar-

ture period is specified differently for different routes or if travel times in the absence of congestion on each route are not identical. Then the arrival time nodes D_1 and D_n should correspond to the earliest and latest possible arrival times, respectively, for any of the possible departure alternatives, on any route and from any origin.

The commuting activities from origin B can also be formulated in the same manner, but the complete graphical representation is not incorporated in Figure 11 for clarity. With the complete network thus formulated, the minimum cost flow pattern in the network will yield the system-optimal assignment to both routes and departures times. In this example, we have considered only nonoverlapping routes. However, a more general transport network can also be modeled in this framework, although the clarity of the graphical presentation would suffer markedly. A numerical example is presented in the next section along with some comments on implementation.

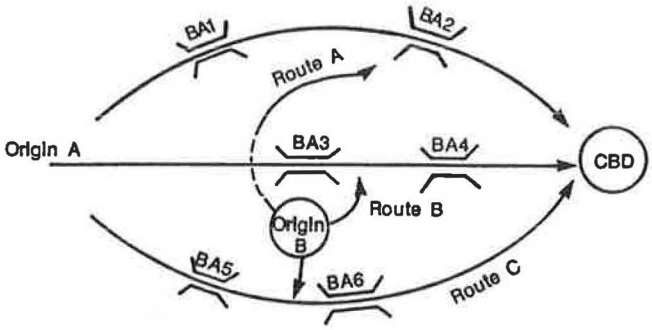

FIGURE 10 Commuting system with multiple bottlenecks on parallel routes.

NUMERICAL APPLICATION

To illustrate some of the issues involved in the application of the methodological framework discussed in this paper and the type of results one can expect, we describe an application to the commuting system shown in Figure 12a. The system is similar to that in Figure 10 in that it consists of two origins (A and B) with access to two parallel highway facilities to the common CBD destination. Each route contains two "bottleneck" sections, the first of which is traversed only by tripmakers from origin A. A constant access time of 5 min is assumed from each origin to the corresponding entry point on the highway facility. Figure 12b shows the characteristics (travel time, capacity per Δt) of each spatial link. Each node

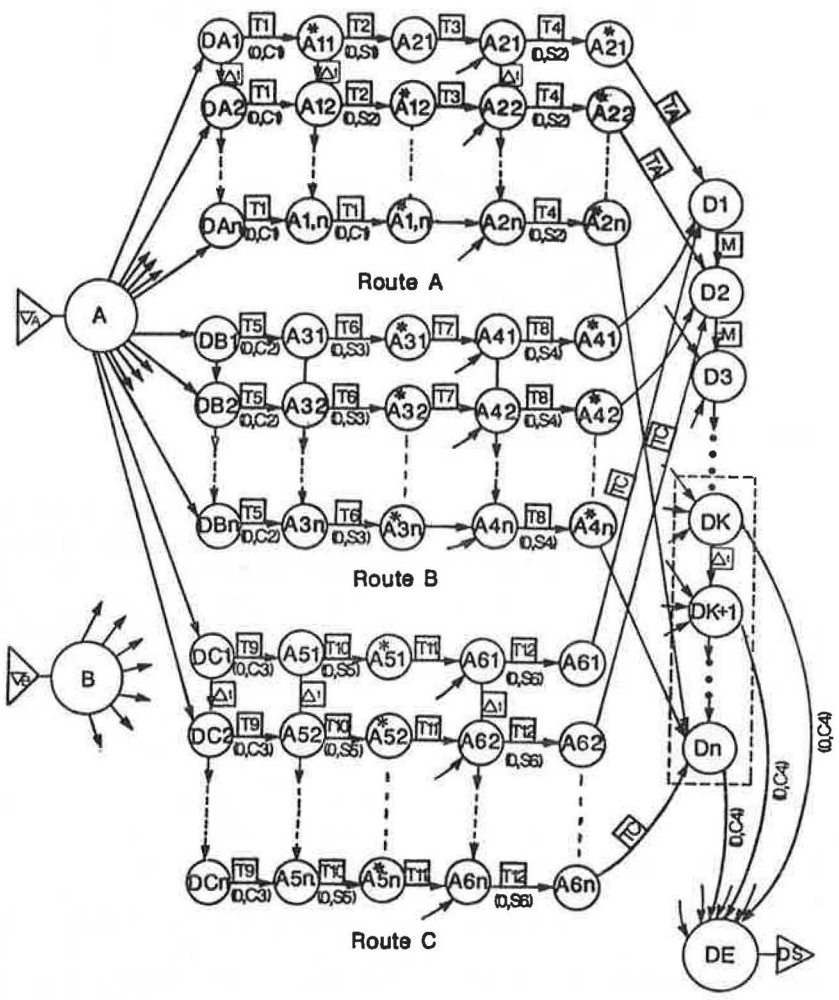

FIGURE 11 Network formulation for commuting system of Figure 10.

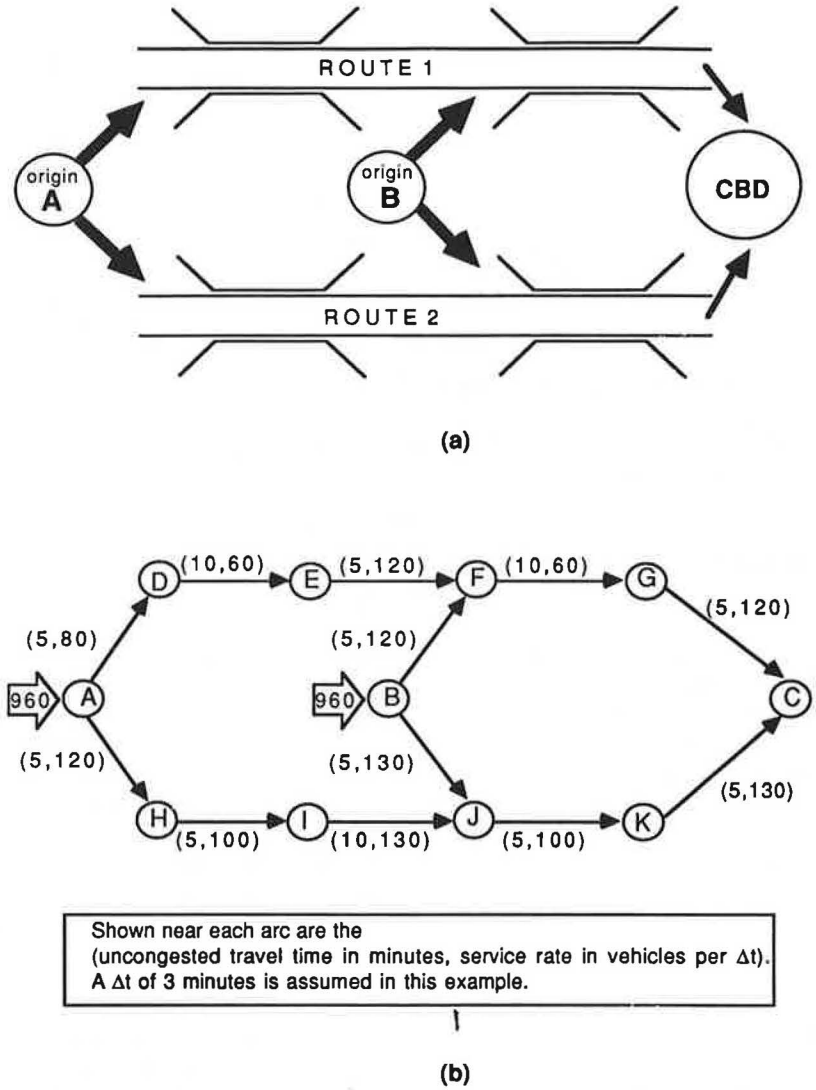

FIGURE 12 Commuting context and data for numerical example.

is assumed to generate a total of 960 vehicle trips during the commuting period.

The network formulation involves adding the time dimension to model waiting times due to congestion and formulating the departure and arrival processes. It should be apparent that developing and coding the time-space network can be a rather time-consuming task. As this network exhibits an obvious repetitive structure, however, this task can be very effectively supported by a network generating code. We have developed such a network generator for commuting systems involving multiple parallel routes with multiple origins. The program is interactive and requires simple input on the number of origins, number of routes, number of spatial nodes, operational characteristics of each spatial arc (i.e., the highway sections and ramps), total trips from each origin, size of the time slice Δt, as well as the range of possible departure times from each origin and acceptable arrival time band (for the satisfying formulation described earlier). This obviously greatly simplifies the practical use of this formulation, as the network can now be generated in an interactive session that requires only a few minutes. The network is then ready for solution by any pure network optimization code. These codes are known for their efficient execution and can easily handle networks with tens of thousands of arcs, thereby alleviating concern about the size of the network needed to model even relatively small physical commuting networks.

For the example under consideration, we have executed the algorithm for three different lengths (in minutes) of the acceptable arrival band: 15, 36, and unconstrained (i.e., all arrival time nodes in the range considered are connected to the total demand sink node). The latter case is included to provide a benchmark for comparing the effect of tightening or relaxing the size of the acceptable arrival (indifference) band on the departure patterns. It was assumed that 8:00 was the common work start time, thus the indifference band would correspond to 7:45–8:00 A.M., 7:24–8:00 A.M., and anytime before 8:00 A.M., respectively. The case with 15 min is not feasible, because that would imply a combined arrival rate much in excess of the capacity of the bottlenecks on the two routes. Actually, 36 min is the minimum feasible arrival period, yielding a total system cost of 52,800 min and a uniform arrival pattern of 160 arrivals per Δt (equal to 3 min in this example; see Figure 12). Because this solution involves no queuing, it cannot be improved on, as evidenced by the solution for the

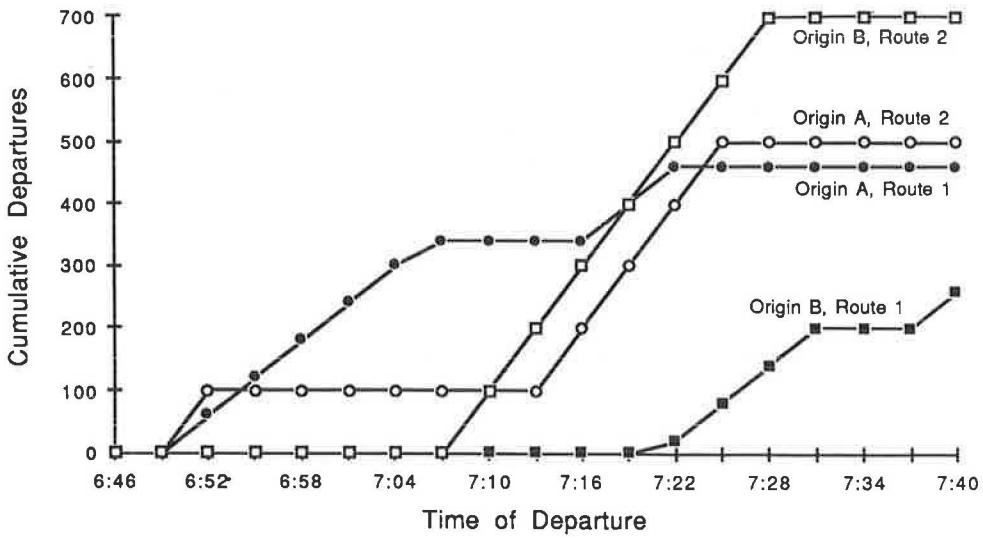

FIGURE 13 Cumulative number of departures from each origin by route.

unconstrained case, which yields more spread out departure and arrival patterns but at the same system cost.

The solution of the network optimization problem also includes the departure pattern from each node (i.e., the set of flows on the arcs connecting each origin to the possible departure time alternatives on each route), the arrival pattern at the CBD, the flows on the vertical queuing links, as well as all the link flows, in addition to the value of the objective function at optimality. The departure patterns from both origin A and origin B for each route are illustrated for the 36-min arrival period in Figure 13.

CONCLUDING COMMENTS

In this paper, a network formulation framework was proposed to solve for the system-optimal time-varying flows in urban commuting networks, yielding optimal departure patterns from each origin on each route as well as the dynamic assignment of traffic to the network's components. The solution of the formulated problem can take advantage of state-of-the-art, large-scale network optimization algorithms. Of course, the representation of the traffic phenomena that may be occurring on the facilities is admittedly crude and simplified; however, this has been a problem in much of the network traffic assignment work, for the static case and particularly for the time-varying formulations. We feel that some compromises in representation, when applied judiciously to preserve the character of the system insofar as the phenomena of interest are concerned, are worth the resulting relative ease of the solution procedure and thus the ability to explore and gain insight into the various aspects of this problem.

It should further be noted that the work presented here is not motivated by a desire to force people to leave at specified times and on preset routes, or by a naive presumption that they would comply if told to do so. Rather, it is intended to generate a benchmark, an "ultimate" solution against which to compare the effectiveness of various strategies, such as, for example, flexible work arrival times. Furthermore, it can be a useful tool to examine the potential of information-related strategies, whereby users could be guided toward the optimal solution. Of course, economists hold the view that one could approach the desired state through pricing; this strategy is not a particularly strong motivator for this work. Another appropriate application of this methodology is the development of contingency evacuation plans for use during some emergency, such as a hurricane or an incident at a nuclear power plant, or for military purposes.

Several improvements and extensions of the methodological framework can be considered. In terms of system representation, extension to the many origins to many destinations case would be most desirable. However, the penalty is rather severe as the problem would then exhibit the features of a capacitated multicommodity problem, which requires additional assumptions for proper resolution, in addition to the obvious increase in the level of complexity required in the representation. Improvements in terms of traffic modeling are certainly possible, but one would then have to sacrifice the easy-to-solve pure network formulation.

ACKNOWLEDGMENTS

This paper is based on research funded by National Science Foundation grant to the University of Texas at Austin. The authors are grateful to Robert Herman for his encouragement and contribution to their efforts in this general problem area. This work has also profited from useful discussion with Andre de Palma of Northwestern University.

REFERENCES

1. A. S. Alfa and D. L. Minh. A Stochastic Model for the Temporal Distribution of Traffic Demand—the Peak Hour Problem. *Transportation Science*, Vol. 13, No. 4, 1979, pp. 315–324.
2. M. Ben-Akiva, M. Cyna and A. de Palma. Dynamic Model of Peak Period Congestion. *Transportation Research*, Vol. 18B, 1984, pp. 339–355.
3. C. F. Daganzo. The Uniqueness of a Time-Dependent Equilibrium Distribution of Arrivals at a Single Bottleneck. *Transportation Science*, Vol. 19, No. 1, 1985, pp. 29–37.

4. A. de Palma, M. Ben-Akiva, C. Lefevre, and N. Litinas. Stochastic Equilibrium Model of Peak Period Traffic Congestion. *Transportation Science*, Vol. 17, No. 4, 1983, pp. 430–453.
5. P. N. Fargier. Effects of the Choice of Departure Time on Road Traffic Congestion: Theoretical Approach. In *Proc., Eighth International Symposium on Transportation and Traffic Theory* (V. F. Hurdle et al., eds.), University of Toronto Press, Toronto, 1983, pp. 223–263.
6. J. V. Henderson. Road Congestion: A Reconsideration of Pricing Theory. *Journal of Urban Economics*, Vol. 1, 1974, pp. 346–365.
7. C. Hendrickson and G. Kocur. Schedule Delay and Departure Time Decisions in a Deterministic Model. *Transportation Science*, Vol. 15, 1981, pp. 62–77.
8. M. J. Smith. The Existence of a Time-Dependent Equilibrium Distribution of Arrivals at a Single Bottleneck. *Transportation Science*, Vol. 18, No. 4, 1984, pp. 385–394.
9. W. S. Vickrey. Congestion Theory and Transport Investment. *American Economic Review*, Vol. 59, 1969, pp. 251–261.
10. M. Ben-Akiva, A. de Palma, and P. Kanaroglou. Effects of Capacity Constraints on Peak-Period Traffic Congestion. *Transportation Research Record 1085*, TRB, National Research Council, Washington, D.C., 1986, pp. 16–26.
11. M. Ben-Akiva, A. de Palma, and P. Kanaroglou. Dynamic Model of Peak Period Traffic Congestion with Elastic Arrival Rates. *Transportation Science*, Vol. 20, No. 3, 1986, pp. 164–181.
12. H. S. Mahmassani and G. L. Chang. On Boundedly Rational User Equilibrium in Transportation Systems. *Transportation Science*, Vol. 21, No. 2, 1987, pp. 89–99.
13. H. S. Mahmassani and R. Herman. Dynamic User Equilibrium Departure Time and Route Choice on Idealized Traffic Arterials. *Transportation Science*, Vol. 18, No. 4, 1984, pp. 362–384.
14. G. F. Newell. The Morning Commute for Nonidentical Travelers. *Transportation Science*, Vol. 21, No. 2, 1987, pp. 74–88.
15. H. S. Mahmassani and G. L. Chang. Experiments with Departure Time Choice Dynamics of Urban Commuters. *Transportation Research B*, Vol. 20B, No. 4, 1986, pp. 297–320.
16. G. L. Chang. Departure Time Choice Dynamics in Urban Transportation Networks. Ph.D. dissertation. Department of Civil Engineering, University of Texas at Austin, 1985.
17. H. S. Mahmassani and G. L. Chang. Dynamic Aspects of Departure Time Choice Behavior in a Commuting System: Theoretical Framework and Experimental Analysis. *Transportation Research Record 1037*, TRB, National Research Council, Washington, D.C., 1985, pp. 88–101.
18. H. S. Mahmassani, G. L. Chang, and R. Herman. Individual Decisions and Collective Effects in a Simulated Traffic System. *Transportation Science*, Vol. 20, No. 4, 1986, pp. 258–271.
19. H. S. Mahmassani and C.-C. Tong. Availability of Information and Dynamics of Departure Time Choice: Experimental Investigation. *Transportation Research Record 1085*, TRB, National Research Council, Washington, D.C., 1986, pp. 33–46.
20. J. V. Henderson. The Economics of Staggered Work Hours. *Journal of Urban Economics*, Vol. 9, 1981, pp. 349–364.
21. C. Hendrickson, D. Nagin, and E. Plank. Characteristics of Travel Time and Dynamic User Equilibrium for Travel-to-Work. In *Proc., Eighth International Symposium on Transportation and Traffic Theory*, (V. F. Hurdle et al., eds.), University of Toronto Press, Toronto, 1983, pp. 321–347.
22. D. K. Merchant and G. L. Nemhauser. A Model and an Algorithm for the Dynamic Traffic Assignment Problem. *Transportation Science*, Vol. 12, 1978, pp. 183–199.
23. D. K. Merchant and G. L. Nemhauser. Optimality Conditions for a Dynamic Traffic Assignment Model. *Transportation Science*, Vol. 12, 1978, pp. 200–207.
24. M. Carey. Optimal Time-Varying Flows on Congested Networks. *Operations Research*. Vol. 35, No. 1, 1987, pp. 58–69.
25. V. F. Hurdle. Equilibrium Flows on Urban Freeways. *Transportation Science*, Vol. 15, No. 3, 1981, pp. 255–293.
26. D. Klingman, N. Phillips, D. Steiger, R. Wirth, R. Padman, and R. Krishnan. An Optimization-Based Integrated Short-Term Refined Petroleum Product Planning System. *CBDA* 123. Center for Business Decision Analysis, Unversity of Texas at Austin, 1985.
27. C. Hendrickson and E. Plank. The Flexibility of Departure Times for Work Trips. *Transportation Research A*, Vol. 18A, No. 1, 1984, p. 25–36.
28. J. L. Kennington and R. V. Helgason. Algorithms for Network Programming. John Wiley & Sons, New York, 1980.

An Application of Optimal Control Theory to Dynamic User Equilibrium Traffic Assignment

BYUNG-WOOK WIE

Optimal control theory is applied to the problem of dynamic traffic assignment, corresponding to user optimization, in a congested network with one origin-destination pair connected by N parallel arcs. Two continuous time formulations are considered, one with fixed demand and the other with elastic demand. Optimality conditions are derived by Pontryagin's maximum principle and interpreted as a dynamic generalization of Wardrop's first principle. The existence of singular controls is examined, and the optimality of singular controls is assured by the generalized convexity conditions. Under the steady-state assumptions, a dynamic model with elastic demand is shown to be a proper extension of Beckmann's equivalent optimization problem with elastic demand. Finally, the derivation of the dynamic user optimization objective functional is demonstrated, which is analogous to the derivation of the objective function of Beckmann's mathematical programming formulation for user equilibrium.

The objective of this paper is to explore the application of optimal control theory to the problem of dynamic traffic assignment corresponding to user optimization. Two continuous time optimal control problems will be formulated, one with fixed demand and the other with elastic demand. The present paper is concerned with dynamic extensions of the steady-state network equilibrium model, particularly Beckmann's equivalent optimization problem, which is a mathematical programming formulation (*1*). This formulation is based on the steady-state assumptions:

(1). The average arc travel cost is some known function of the total traffic flow traversed during the period of analysis;
(2). Travel demands associated with each origin-destination (O-D) pair are constant over time; and
(3). Flow entering each arc is always equal to flow leaving that arc during the period of analysis.

Hence, the relaxation of the steady-state assumptions lead to the problem of dynamic traffic assignment in which the network characteristics are explicit functions of time.

A pioneering research in dynamic traffic assignment was accomplished by Merchant and Nemhauser (*2–4*). They formulated the model as a discrete time, nonlinear, and nonconvex mathematical program corresponding to system optimization in a multiple-origin single-destination network. They showed that the Kuhn-Tucker optimality conditions can be interpreted as a generalization of Wardrop's second principle, which requires equalization of certain marginal travel costs for all the paths that are being used. The behavior of their dynamic model was also examined under the steady-state assumptions, and as a result the model was proven to be a proper generalization of the conventional static system optimal traffic assignment model.

The algorithmic question of implementing the Merchant-Nemhauser (M-N) model was resolved by Ho (*5*). He showed that, for a piecewise linear version of the M-N model, a global optimum is contained in the set of optimal solutions of a certain linear program. He also presented a sufficient condition for optimality, which implies that a global optimum can be obtained by successively optimizing at most $N + 1$ objective functions for the linear program, where N is the number of time periods in the planning horizon.

Recently Carey (*6*) resolved a hitherto open question as to whether the M-N model satisfies a constraint qualification. It was shown that the M-N model does in fact satisfy a constraint qualification, which establishes the validity of the optimality analysis presented by Merchant and Nemhauser (*4*). More recently, Carey (*7*) reformulated the M-N model as a convex nonlinear mathematical program. As a consequence, the new formulation could have analytical, computational, and interpretational advantages in comparison with the original M-N model. In particular, the Kuhn-Tucker conditions are both necessary and sufficient to characterize an optimal solution; in the M-N model, however, they are not sufficient because the constraint set is not convex.

In contrast with the aforementioned mathematical programming approaches, Luque and Friesz (*8*) provided a new insight into the problem of dynamic traffic assignment through the application of optimal control theory. They formulated the M-N model as a continuous time-optimal control problem corresponding to system optimization. The optimality conditions were derived by applying Pontryagin's maximum principle, and economic interpretation was conducted and compared with those obtained from Merchant and Nemhauser (*4*).

It is worth noting that the Merchant-Nemhauser model and its extended models consider a system-optimized flow pattern that satisfies a dynamic generalization of Wardrop's second principle. In general, a traffic flow pattern obeying Wardrop's second principle minimizes the total transportation cost of the network as a whole, and it can be regarded as the most desirable flow pattern for society. In the present paper, however, we are interested in a user-optimized flow pattern obeying a dynamic generalization of Wardrop's first principle, which

Department of City and Regional Planning, University of Pennsylvania, Philadelphia 19104.

requires equalization of certain unit travel costs for all the paths that are being used. Suppose that travel demands are time-dependent but fixed in a multiple O-D network. The problem of dynamic traffic assignment corresponding to user optimization can be viewed as a noncooperative game between players associated with various O-D pairs and departure times. Wardrop's first principle can then be generalized for dynamic traffic assignment such that:

> Individual drivers attempt to minimize their own travel costs by changing routes. At each instant in time, no one can reduce his or her travel costs by unilaterally changing routes; therefore, the unit travel costs on paths used by drivers who have the same departure time and O-D pair are identical and equal to the minimum unit path costs for that O-D pair.

Our analysis is restricted to the network with one O-D pair that is connected by N parallel arcs, as shown in Figure 1. It is also assumed that there is one transport mode—for example, private automobile. Note that A is the set of directed arcs. We will use index a to denote a directed arc. We will consider a fixed planning horizon of length T; that is, all activities occur at some time $t \in [0, T]$. In the remainder of this paper, traffic flow is defined as the average number of vehicles passing a fixed point of an arc per unit of time, and traffic volume is defined as the total number of vehicles accumulated on arc a at some time $t \in [0, T]$.

Our dynamic model is related to models proposed by Hurdle (*9*), Hendrickson and Kocur (*10*), Mahmassani and Herman (*11*), Mahmassani and Chang (*12*), de Palma et al. (*13*), Ben-Akiva et al. (*14,15*), Smith (*16*), Daganzo (*17*), and Newell (*18*). But our model differs in important aspects, which include its formulation as a continuous time optimal control problem. We do not attempt to compare our model with models proposed by the authors just cited. One may refer to Friesz (*19*) and Alfa (*20*) for literature reviews on the dynamic network equilibrium models proposed to date.

ASSUMPTIONS

Exit Function

The flow leaving arc $a \in A$ is a function of the traffic volume accumulated on that arc at time $t \in [0, T]$. The exit functions $g_a[x_a(t)]$ are concave, differentiable, nondecreasing, and nonnegative for all $x_a(t) \geq 0$, with the additional restriction that $g_a(0) = 0$ (Figure 2).

Demand Function

Denote by $\theta[t,D(t)]$ the inverse of the travel demand function where $D(t)$ is the travel demand between origin and destination at time $t \in [0,T]$. The function $\theta[t,D(t)]$ is strictly monotone, decreasing, differentiable, and nonnegative for all $D(t) \geq 0$ and has a different function at each time $t \in [0, T]$ for time-dependent elasticity of demand (Figure 3).

Cost Function

The travel cost on arc $a \in A$ is a function of the traffic volume accumulated on that arc at time $t \in [0, T]$. The cost functions $C_a[x_a(t)]$ are convex, differentiable, nondecreasing, and nonnegative for all $x_a(t) \geq 0$. Note that the travel cost on arc $a \in A$ is simultaneously a function of the exit flow of that arc at time $t \in [0, T]$; that is, $C_a[x_a(t)] \equiv C_a\{g_a[x_a(t)]\}$ (Figure 4).

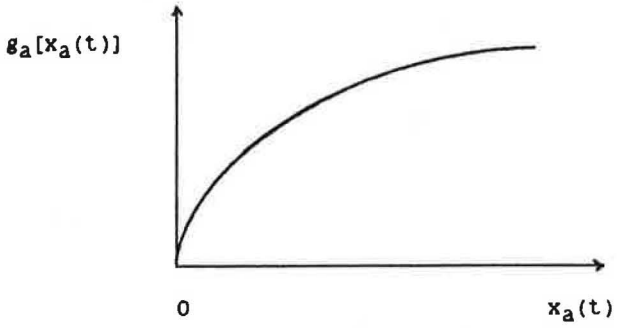

FIGURE 2 Exit function.

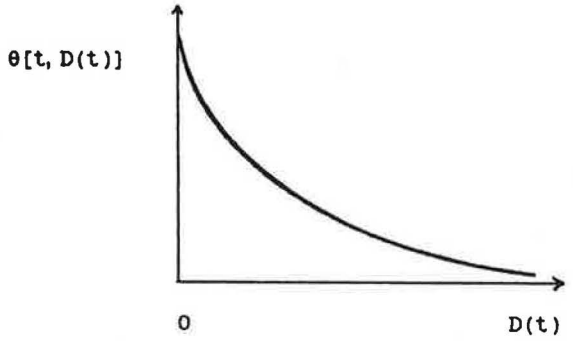

FIGURE 3 Demand function.

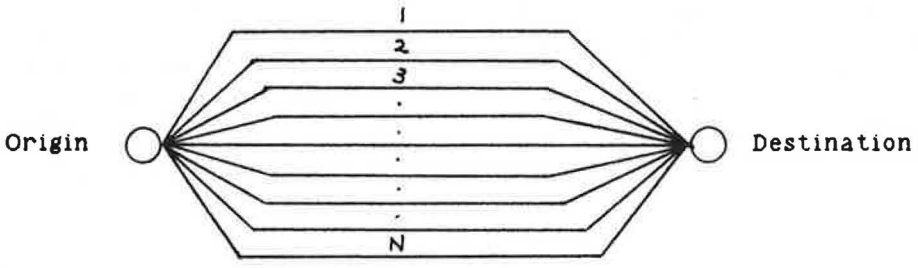

FIGURE 1 Simple network with N parallel arcs.

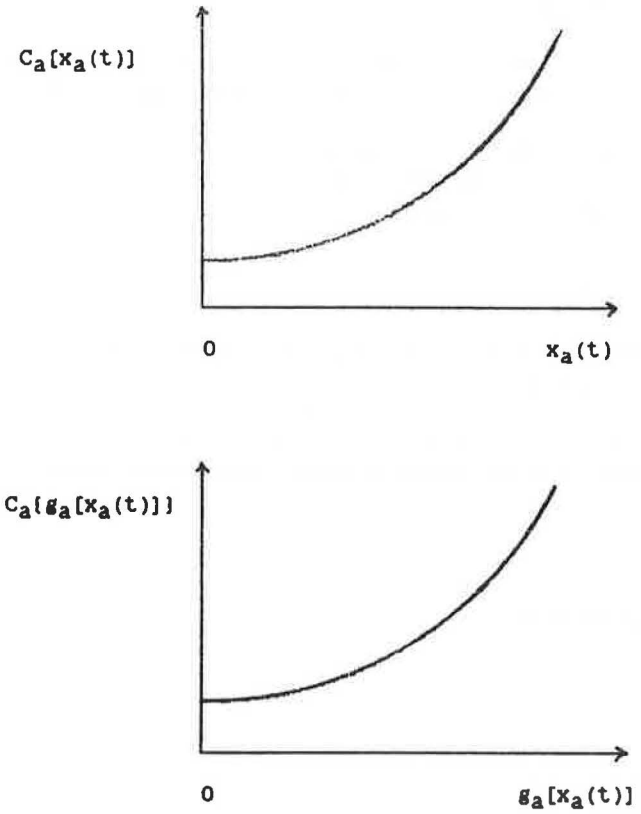

FIGURE 4 Cost function.

DYNAMICS AND CONSTRAINTS

The dynamic evolution of the state of arc $a \in A$ is described by the first-order nonlinear differential equations:

$$\dot{x}_a(t) = \frac{dx_a(t)}{dt} = u_a(t) - g_a[x_a(t)] \qquad \forall\, a \in A \quad t \in [0, T] \qquad (1)$$

where

$x_a(t)$ = the state variable, denoting the traffic volume on arc a at time t;
$u_a(t)$ = the control variable, denoting the flow entering arc a at time t;
$g_a[x_a(t)]$ = the flow leaving arc a at time t; and
$\dot{x}_a(t)$ = the time derivative of the state variable.

Because the state variable is an explicit function of time, $\dot{x}_a(t)$ can be interpreted as the instantaneous rate of change in the traffic volume on arc a with respect to time, which is the difference between inflow and outflow on arc a. Equation 1 is called the state equation in this paper. We can see that the state equation is linear in the control variable and nonlinear in the state variable because of nonlinearity of the exit function $g_a[x_a(t)]$ with respect to the state variable.

For the origin node, the flow conservation constraints can be stated as

$$\sum_{a \in A} u_a(t) = D(t) \qquad \forall\, t \in [0, T] \qquad (2)$$

Equation 2 requires that the number of trips generating at the origin node at time t must be equal to the summation of the control variables over all arcs at time t. Note that $D(t)$ would be exogenously determined in the dynamic model with fixed demand and endogenously determined in that with elastic demand; see following sections of this paper.

In addition, we assume that the traffic volume on arc a is a known positive constant at time $t = 0$:

$$x_a(0) = x_a^0 \qquad \forall\, a \in A \qquad (3)$$

We also ensure that both the state variable and control variables are nonnegative for all arcs and $t \in [0,T]$:

$$x_a(t) \geq 0 \qquad \forall\, a \in A \quad t \in [0, T] \qquad (4)$$

$$u_a(t) \geq 0 \qquad \forall\, a \in A \quad t \in [0, T] \qquad (5)$$

Because the assumption that $g_a(0) = 0$ ensures that the state variables are always nonnegative, we do not subsequently consider constraints (Equation 4) in an explicit manner. For simplicity, we do not impose the upper bound on the control variables as a physical constraint, indicating the maximum inflow admitted to arc a. Define $x \equiv (\ldots, x_a, \ldots)$ and $u \equiv (\ldots, u_a, \ldots)$. To save notational efforts, the following set is used as the set of feasible solutions.

$$\Omega = \{(x,u) : \text{Equations 1, 2, 3, and 5 are satisfied}\} \qquad (6)$$

DYNAMIC USER EQUILIBRIUM TRAFFIC ASSIGNMENT WITH FIXED DEMAND

Model Formulation

Suppose the number of trips generating from the origin at each time $t \in [0, T]$ is fixed and known. We postulate that the following continuous time optimal control problem has a solution that is a user-optimized flow pattern satisfying a dynamic generalization of Wardrop's first principle:

$$\text{Minimize } J_1 = \sum_{a \in A} \int_0^T \int_0^{x_a(t)} C_a(w) g_a'(w)\, dw\, dt$$

$$\text{subject to } (x,u) \in \Omega \qquad (7)$$

The performance index J_1 is the summation of an integrated integral over all arcs in the network. The derivation of J_1 has the same analogy to that of the objective function of Beckmann's equivalent optimization problem with fixed demand. The detailed derivation of J_1 is shown in the appendix. Because the performance index J_1 does not have any intuitive economic interpretation, it should be viewed as a mathematical construction to solve the problem of dynamic user equilibrium traffic assignment. When J_1 achieves its minimum value, the control problem (Equation 7) provides us with a user-optimized traffic flow pattern that is described by the optimal trajectories through time of both the state and the control variables. Note that the control problem (Equation 7) is formulated in the Lagrange form because we do not impose any

state constraint at the terminal time T. We shall suppress the time notation (t) when no confusion arises.

Optimality Conditions

The necessary conditions for an optimal solution of the control problem (Equation 7) can be derived by Pontryagin's maximum principle [Pontryagin et al., (*21*)]. As a first step in analyzing the necessary conditions, we construct the Hamiltonian:

$$H = \sum_{a \in A} \int_0^{x_a} C_a(w) g_a'(w)\, dw + \sum_{a \in A} \gamma_a \cdot [u_a - g_a(x_a)] + \mu \cdot [D - \sum_{a \in A} u_a] + \sum_{a \in A} \beta_a \cdot [-u_a] \quad (8)$$

where $\gamma_a(t)$ is the costate variable associated with the ath state Equation 1; $\mu(t)$ is the Lagrange multiplier associated with the flow conservation constraints at the origin; and $\beta_a(t)$ is also the Lagrange multiplier associated with nonnegativity of the ath control variable.

We can obtain the first-order necessary conditions, also known as the Euler-Lagrange equations in the calculus of variations. The differential equations governing the evolution of the costate variables γ_a are given from the Hamiltonian (Equation 8), which require [see Bryson and Ho, (*22*)]:

$$\frac{\partial H}{\partial x_a} = -\dot{\gamma}_a = [C_a(x_a) - \gamma_a]\, g_a'(x_a) \qquad \forall\, a \in A \quad t \in [0,T] \quad (9)$$

Equation 9 will be called the costate equation. Boundary conditions on the costate variables are obtained by the transversality conditions:

$$\gamma_a(T) = 0 \qquad \forall\ a \in A \quad (10)$$

According to Pontryagin's maximum principle, the Hamiltonian must be minimized at each time $t \in [0, T]$. The Kuhn-Tucker optimality conditions for u_a^* to be an optimal solution that minimizes the Hamiltonian are readily obtained as:

$$\frac{\partial H}{\partial u_a} = 0 = \gamma_a - \mu - \beta_a \qquad \forall\, a \in A \quad (11)$$

$$\beta_a \geq 0 \quad \text{and} \quad \beta_a \cdot u_a = 0 \qquad \forall\, a \in A \quad (12)$$

In the terminology of optimal control theory, $\partial H/\partial u_a$ is often called impulse response function because the gradient of the Hamiltonian with respect to the control variable represents the variation in the performance index J_1 as a consequence of a unit impulse in the corresponding control variable at time t, while holding x_a^o constant and satisfying the state equation (Equation 1). In particular, Equation 12 contains the complementary slackness conditions to take into account nonnegativity of the control variables.

The preceding necessary conditions for optimality may be collected in the following compact form:

$$-\dot{\gamma}_a = [C_a(x_a) - \gamma_a] g_a'(x_a) \qquad \forall a \in A \quad t \in [0, T] \quad (13)$$

$$\gamma_a(T) = 0 \qquad \forall a \in A \quad (14)$$

$$\gamma_a - \mu \geq 0 \qquad \forall a \in A \quad t \in [0, T] \quad (15)$$

$$u_a \cdot (\gamma_a - \mu) = 0 \qquad \forall a \in A \quad t \in [0, T] \quad (16)$$

The optimality conditions (Equations 13–16) can be understood such that if at some time $t \in [0, T]$ $\gamma_a > \mu$ for all $a \in A$, the flow entering arc a is equal to zero, and if $\gamma_a = \mu$, u_a is either zero or singular in nature. Singular control is discussed further in the next section. It is implied that the quantities determining the control variable u_a are the value of $[\gamma_a^n - \mu]$, which is the difference between the costate variable and the Lagrange multiplier. Hence, we may conjecture that the optimality conditions are analogous to the principle of the flow of electricity, in which electric current moves from a node with higher voltage to a node with lower voltage.

The Arrow-Kurz sufficiency theorem (*23*, *24*) ensures that the necessary conditions are also sufficient when the Hamiltonian is convex in the state variables. We can see that the Hamiltonian (Equation 8) is convex in the state variables under the assumptions made previously. Hence, the optimality conditions (Equations 13–16) are necessary and also sufficient.

Singular Controls

Because the Hamiltonian (Equation 8) is linear in the control variable, the gradient of the Hamiltonian with respect to u_a does not depend on the control variable. Therefore, the optimality conditions for u_a^* to be an optimal control that minimizes the Hamiltonian provide no useful information to determine the optimal control in terms of the state and costate variables. In this case we must take successive time derivatives of $\frac{\partial H}{\partial u_a} = \gamma_a - \mu - \beta_a$ and make appropriate substitutions by using the state Equation 1 and the costate Equation 9 until we find an explicit expression for the control variables. The optimal control determined by this procedure is called a singular control. A finite time interval for which a singular control exists is called a singular interval. An extremal arc on which the determinant of the matrix $\frac{\partial}{\partial u_a}\left[\frac{\partial H}{\partial u_a}\right]$ vanishes identically is called a singular arc.

To determine the singular control, we must use the fact that successive time derivatives of the gradient of the Hamiltonian would be also constant and equal to zero on a singular arc. The first and second time derivatives of the gradient of the Hamiltonian with respect to u_a give the following relationship:

$$\dot{\gamma}_a = \dot{\mu} \quad \text{and} \quad \ddot{\gamma}_a = \ddot{\mu} \qquad \forall a \in A \quad t \in [t_1, t_2] \subseteq [0, T] \quad (17)$$

We substitute the costate equation (Equation 9) into the first relationship in Equation 17:

$$(C_a - \gamma_a) g_a' + \dot{\mu} = 0 \quad (18)$$

The second time derivatives of Equation 18 are calculated:

$$(C_a' \dot{x}_a - \dot{\gamma}_a) g_a' + (C_a - \gamma_a) g_a'' \dot{x}_a + \ddot{\mu} = 0 \quad (19)$$

By using the state equation (1), we may manipulate Equation 19 to yield the following expression for the singular control:

$$u_a = \frac{\dot{\mu} g_a' - \ddot{\mu} + [C_a' g_a' + (C_a - \mu) g_a''] g_a}{C_a' g_a' + (C_a - \mu) g_a''}$$

$$\forall\, a \in A \quad t \in [t_1, t_2] \subseteq [0, T] \qquad (20)$$

One may ask whether the singular control given by Equation 20 is optimal or not. To answer this question, we shall derive the necessary conditions for optimality of singular controls. The generalized convexity condition can be obtained elsewhere (*22*):

$$\frac{\partial}{\partial u_a}\left[\frac{d^2}{dt^2}\left[\frac{\partial H}{\partial u_a}\right]\right] = -(C_a - \gamma_a) g_a'' - C_a' g_a' \leq 0$$

$$\forall\, a \in A \quad t \in [t_1, t_2] \subseteq [0, T] \qquad (21)$$

According to the costate equations (Equation 9) and assumptions made earlier, the generalized convexity conditions are satisfied. Hence, we can conclude that the singular control (Equation 20) is optimal.

Dynamic User Equilibrium Principle

The important question now arises as to whether or not a traffic flow pattern, described by time trajectories of the state and control variables as an optimal solution of Equation 7, satisfies a dynamic generalization of Wardrop's first principle. To answer this question, we define the following function by manipulating the costate equation (Equation 9):

$$\Phi_a(t) \equiv C_a(x_a) + \dot{\gamma}_a / g_a'(x_a) \qquad \forall a \in A \quad t \in [0, T] \qquad (22)$$

It is well known that when the performance index J_1 achieves its minimum value J_1^*, we have the following properties (*22*):

$$\gamma_a(t) = \frac{\partial J_1^*}{\partial x_a(t)}$$

$$\dot{\gamma}_a(t) = \frac{d}{dt}\left(\frac{\partial J_1^*}{\partial x_a(t)}\right) \qquad \forall\, a \in A \quad t \in [0, T] \qquad (23)$$

We see that $\dot{\gamma}_a(t)$ is the time rate of change in the value of the performance index J_1 as a consequence of a change in the corresponding state variable $x_a(t)$ along the optimal state trajectory at time $t \in [0, T]$. Therefore, we may interpret $\Phi_a(t)$ as the sum of static and dynamic terms: $C_a(x_a)$ is the unit travel cost on arc a that is equilibrated in the static user optimization problem; and $\dot{\gamma}_a / g_a'$ is regarded as the contribution to arc unit travel cost due to the dynamic nature of our control problem. In the present paper, we call $\Phi_a(t)$ the instantaneous travel cost on arc $a \in A$ at time $t \in [0, T]$.

We are now ready to state and prove the following theorem:

Theorem 1: If at some time $t \in [0, T]$, $u_a > 0$ for all $a \in A$, then $\Phi_a(t) = \inf \{\Phi_a(t) : \forall\, a \in A\}$.

Proof. From the costate equation (9), we see that

$$\Phi_a(t) = C_a(x_a) + \dot{\gamma}_a / g_a'(x_a) = \gamma_a$$

$$\forall\, a \in A \quad t \in [0, T] \qquad (24)$$

However, from Equation 15, we know that

$$\gamma_a \geq \mu \qquad \forall\, a \in A \quad t \in [0, T] \qquad (25)$$

It follows at once from Equations 24 and 25 that

$$\Phi_a(t) \geq \mu \qquad \forall\, a \in A, t \in [0, T] \qquad (26)$$

We also know that if $u_a > 0$ for all $a \in A$, then Equation 25 holds as an equality because of the complementary slackness conditions of Equations 15 and 16. Hence, the theorem is immediately proved.

Theorem 1 tell us that user equilibrium conditions hold at each instant in our dynamic model. Hence, we regard Theorem 1 as a dynamic generalization of Wardrop's first principle, which is termed the Dynamic User Equilibrium Principle in the present paper. But it is restricted to a network with one O-D pair connected by N parallel arcs. This principle can also be restated at each instant $t \in [0, T]$:

$$\Phi_1(t) = \Phi_2(t) = \ldots = \Phi_k(t) \leq \Phi_{k+1}(t) \leq \ldots \leq \Phi_N(t) \qquad (27)$$

$$u_a(t) > 0 \qquad \text{for } a = 1, 2, \ldots, k \qquad (28)$$

$$u_a(t) = 0 \qquad \text{for } a = k+1, \ldots, N \qquad (29)$$

DYNAMIC USER EQUILIBRIUM TRAFFIC ASSIGNMENT WITH ELASTIC DEMAND

Model Formulation

Suppose that travel demands change in response to travel costs between the elements of an origin-destination pair. We postulate that the following continuous, time-optimal control problem has a solution that is a user-optimized flow pattern obeying a dynamic generalization of Wardrop's first principle:

$$\text{Minimize } J_2 = \sum_{a \in A} \int_0^T \int_0^{x_a(t)} C_a(w) g_a'(w)\, dw\, dt - \int_0^T \int_0^{D(t)} \theta(t, y)\, dy\, dt \qquad (30)$$

subject to

$$(x, u) \in \Omega$$

$$D(t) = \sum_{a \in A} u_a(t)$$

where $D(t)$ is the number of trips generating at the origin at time $t \in [0, T]$ and $\theta[t, D(t)]$ is the inverse of the travel demand function. Note that $D(t)$ is determined endogenously in the control problem (Equation 30). The performance index J_2 is decomposed into two terms: the performance index J_1 and an integrated integral of the inverse demand function.

Optimality Conditions

To analyze the necessary conditions, we construct the Hamiltonian:

$$H = \sum_{a\in A} \int_0^{x_a} C_a(w)g_a'(w)\,dw - \int_0^D \theta(t, y)\,dy + \sum_{a\in A} \gamma_a \cdot [u_a - g_a(x_a)] + \sum_{a\in A} \beta_a \cdot [-u_a] \quad (31)$$

It is important to note that the second term of H is strictly concave in the control variable u_a because the integral of a monotone decreasing function is strictly concave. The negative of a strictly concave function is, however, a strictly convex function.

The costate equations and transversality conditions are identical to Equations 9 and 10, respectively. The Kuhn-Tucker optimality conditions for the minimization of the Hamiltonian (Equation 31) with respect to the control variables are obtained:

$$\frac{\partial H}{\partial u_a} = 0 = \gamma_a - \theta[t, \sum_{a\in A} u_a] - \beta_a \qquad \forall\, a \in A \quad (32)$$

$$\beta_a \geq 0 \quad \text{and} \quad \beta_a \cdot u_a = 0 \qquad \forall\, a \in A \quad (33)$$

We may collect the necessary conditions for optimality of the problem of dynamic user equilibrium traffic assignment with elastic demand in the following compact form:

$$-\dot{\gamma}_a = [C_a(x_a) - \gamma_a]g_a'(x_a) \qquad \forall a\in A \quad t\in[0,T] \quad (34)$$

$$\gamma_a(T) = 0 \qquad \forall a\in A \quad (35)$$

$$\gamma_a - \theta \geq 0 \qquad \forall a\in A \quad t\in[0,T] \quad (36)$$

$$u_a \cdot (\gamma_a - \theta) = 0 \qquad \forall a\in A \quad t\in[0,T] \quad (37)$$

It can be understood from the optimality conditions (Equations 34–37) that if at some time $t \in [0, T]$ $\gamma_a > \theta$ for all $a \in A$, the flow entering arc a is equal to zero; and if $\gamma_a = \theta$, then u_a is explicitly determined by the state equation (Equation 1) and the costate equation (9) as a solution of a two-point boundary-value problem. It is worth noting that the control problem (Equation 30) does not have singular controls.

The Arrow-Kurz sufficiency theorem (*23*, *24*) ensures that the optimality conditions (Equations 34–37) are necessary and also sufficient, because the Hamiltonian (Equation 31) is convex in the state variables under the assumptions made previously. In addition, Theorem 1 holds for the dynamic model (Equation 30) except for the fact that $\mu(t)$ is replaced by $\theta[t, D(t)]$ in Equations 25 and 26.

Equivalency Under the Steady-State Assumptions

We wish to assure that the control problem (Equation 30) is a proper dynamic extension of Beckmann's equivalent optimization problem with elastic demand. To do this, we examine the behavior of our dynamic model under the steady-state assumptions, such that the time rate of a change in the traffic volume on each arc would be zero during $[0, T]$ and travel demands would be constant over time.

Through a change of the variables of integration, we may rewrite the first term in the Hamiltonian (Equation 31) and have the following relation:

$$\sum_{a\in A} \int_0^{x_a} C_a(w)g_a'(w)\,dw = \sum_{a\in A} \int_0^{g_a(x_a)} C_a(s)\,ds \quad (38)$$

Let f_a denote the flow on arc a because u_a is always equal to $g_a(x_a)$ under the steady-state assumptions. In addition, the inverse of the demand function is denoted by $\theta(D)$. We are now ready to formulate our dynamic model (Equation 30) as a nonlinear convex mathematical program under the steady-state assumptions as follows:

$$\text{Minimize } Z(\mathrm{f}, D) = \sum_{a\in A} \int_0^{f_a} C_a(s)\,ds - \int_0^D \theta(y)\,dy \quad (39)$$

subject to

$$D = \sum_{a\in A} f_a \quad (40)$$

$$f_a \geq 0 \qquad \forall\, a \in A \quad (41)$$

$$D \geq 0 \quad (42)$$

The Kuhn-Tucker optimality conditions for the problem (Equations 39 through 42) can be readily obtained as

$$f_a\,[C_a(f_a) - \lambda] = 0 \qquad \forall\, a \in A \quad (43)$$

$$C_a(f_a) - \lambda \geq 0 \qquad \forall\, a \in A \quad (44)$$

$$D\,[\lambda - \theta(D)] = 0 \quad (45)$$

$$\lambda - \theta(D) \geq 0 \quad (46)$$

$$f_a \geq 0 \qquad \forall\, a \in A \quad (47)$$

$$D \geq 0 \quad (48)$$

where λ is the Lagrange multiplier interpreted as the minimum travel cost between members of the O-D pair. Because the optimality conditions (Equations 43–48) are identical to user equilibrium conditions, we can conclude that our dynamic model is a proper generalization of Beckmann's equivalent optimization problem with elastic demand. Obviously, the dynamic model (Equation 7) is also a proper extension of the static user equilibrium traffic assignment model with fixed demand.

CONCLUSION

Our analysis has been restricted to a very simple network. Obviously, its further extension would be to have a more complex network with multiple origins and multiple destinations (*25–27*). We have not discussed any computational issues on implementing our dynamic model; such issues are important in assessing the applicability to a realistic network. The existing solution algorithms for dynamic system-optimal

traffic assignment could probably be modified to solve our dynamic model after the discretization of continuous time-optimal control problems (*3*, *5*). We have also assumed that the exit function is nondecreasing; however, it is not true according to traffic flow theory. In fact, an exit function is both increasing and decreasing, and an exit flow is maximized at an optimum density (traffic volume per unit length). Finally, the concept of dynamic user equilibrium made in this paper must be clearly redefined and compared with one that already exists in the transportation literature. An important question would be whether or not our dynamic model with elastic demand is equivalent to a deterministic user equilibrium model of joint route and departure time.

APPENDIX: THE DERIVATION OF THE PERFORMANCE INDEX J_1

Luque and Friesz (*8*) considered the optimal control problem for dynamic system-optimal traffic assignment in a multiple-origin, single-destination network. We need to transform their original formulation into the control problem for a single origin-destination network:

$$\text{Minimize } J_3 = \sum_{a \in A} \int_0^T S_a[x_a(t)]\, dt \tag{A-1}$$

subject to

$$(x, u) \in \Omega$$

where $S_a[x_a(t)]$ is the total travel cost on arc a at time t. The costate equations are obtained:

$$-\dot{\gamma}_a = \frac{\partial H}{\partial x_a} = S_a'(x_a) - \gamma_a g_a'(x_a) \qquad \forall\, a \in A \quad t \in [0, T] \tag{A-2}$$

Then we define the following function:

$$\phi_a(t) = \frac{S_a'(x_a) + \dot{\gamma}_a}{g_a'(x_a)} \qquad \forall\, a \in A \quad t \in [0, T] \tag{A-3}$$

Luque and Friesz (*8*) state that the numerator of Equation A-3 has the units of incremental travel cost per unit increment of traffic volume on arc a, whereas $g_a'(x_a)$ has the units of incremental flow per increment of traffic volume. Equation A-3 expresses incremental travel cost per unit increment of flow; therefore $\phi_a(t)$ can be interpreted as the instantaneous marginal travel cost on arc a at time t. The theorem proved in Luque and Friesz (*8*) enables us to state a dynamic generalization of Wardrop's second principle for all $t \in [0, T]$:

$$\phi_1(t) = \phi_2(t) = \ldots\ldots = \phi_k(t) \leq \phi_{k+1}(t) \leq \ldots\ldots \leq \phi_N(t) \tag{A-4}$$

$$u_a(t) > 0 \qquad \text{for } a = 1, 2, \ldots, k \tag{A-5}$$

$$u_a(t) = 0 \qquad \text{for } a = k+1, \ldots, N \tag{A-6}$$

We can see that the set of arcs is grouped into two subsets: one for arcs with positive inflow and equal instantaneous marginal travel cost, and the other for arcs with zero inflow and travel costs greater than or equal to minimum instantaneous marginal travel cost.

We now hypothesize that the optimal control problem of Equation A-1 with a fictitious performance index $\tilde{J}_3$ determines a dynamic user-optimized traffic flow pattern. The remaining question is how to identify a fictitious performance index $\tilde{J}_3$. To answer this question, we define $\tilde{S}_a[x_a(t)]$ as a fictitious travel cost on arc a when it contains the traffic volume $x_a(t)$ at time $t \in [0, T]$. Provided that the preceding hypothesis is accepted, the following optimal control problem must give a traffic flow pattern obeying the Dynamic User Equilibrium Principle:

$$\text{Minimize } \tilde{J}_3 = \sum_{a \in A} \int_0^T \tilde{S}_a[x_a(t)]\, dt \tag{A-7}$$

subject to

$$(x, u) \in \Omega$$

Then we can readily obtain the fictitious instantaneous marginal travel cost on arc a at time t as

$$\tilde{\phi}_a(t) = \frac{\tilde{S}_a'(x_a) + \dot{\gamma}_a}{g_a'(x_a)} \qquad \forall\, a \in A \quad t \in [0, T] \tag{A-8}$$

For the hypothesis to be true, the following condition must be satisfied:

$$\Phi_a(t) = \tilde{\phi}_a(t) \qquad \forall\, a \in A, t \in [0, T] \tag{A-9}$$

Using Equation 22, we have the following relation:

$$C_a(x_a) + \frac{\dot{\gamma}_a}{g_a'(x_a)} = \frac{\tilde{S}_a'(x_a)}{g_a'(x_a)} + \frac{\dot{\gamma}_a}{g_a'(x_a)} \qquad \forall\, a \in A \quad t \in [0, T] \tag{A-10}$$

Then we manipulate Equation A-10 as follows:

$$\tilde{S}_a'(x_a) = C_a(x_a) \cdot g_a'(x_a) \qquad \forall\, a \in A \quad t \in [0, T] \tag{A-11}$$

where $\tilde{S}_a'(x_a) = \dfrac{d\tilde{S}_a(x_a)}{dx_a}$

Equation A-11 can be rewritten as

$$d\tilde{S}_a'(x_a) = C_a(x_a) \cdot g_a'(x_a) \cdot dx_a \qquad \forall\, a \in A, t \in [0, T] \tag{A-12}$$

Turning A-12 into a definite integral, we get the explicit form of a fictitious travel cost:

$$\tilde{S}_a[x_a(t)] = \int_0^{x_a(t)} C_a(w) g_a'(w)\, dw \qquad \forall\, a \in A \quad t \in [0, T] \tag{A-13}$$

Consequently, we can get the explicit expression of the performance index J_1 by substituting Equation A-13 to Equation A-7:

$$J_1 = \sum_{a \in A} \int_0^T \int_0^{x_a(t)} C_a(w) g_a'(w)\, dw\, dt \qquad \text{(A-14)}$$

GLOSSARY

a: an arc;
A: the set of arcs in the network;
$x_a(t)$: the state variable, indicating the traffic volume accumulated on arc a at time t;
$\dot{x}_a(t)$: the time derivative of the state variable;
$u_a(t)$: the control variable, indicating the traffic flow entering arc a at time t;
$\gamma_a(t)$: the costate variable to take account of the state equation in the minimization of the Hamiltonian;
$\dot{\gamma}(t)$: the time derivative of the costate variable;
$\mu(t)$: the Lagrange multiplier to take account of the flow conservation constraint at the origin node;
$\beta_a(t)$: the Lagrange multiplier to take account of the nonnegativity of the control variables;
J_1: the performance index for dynamic user equilibrium traffic assignment with fixed demand;
J_2: the performance index for dynamic user equilibrium traffic assignment with elastic demand;
H: the Hamiltonian;
$C_a[x_a(t)]$: travel cost on arc a when it contains the traffic volume x_a at time t;
$g_a[x_a(t)]$: the flow leaving arc a when it contains the traffic volume x_a at time t;
$C_a'[x_a(t)]$: the derivative of the travel cost function with respect to the state variable;
$g_a'[x_a(t)]$: the derivative of the exit function with respect to the state variable;
$[0, T]$: the period of analysis, where T is the fixed terminal time;
$D(t)$: the number of trips generating at the origin node at time t;
$\theta[t, D(t)]$: the inverse of the travel demand function;
$\Phi_a(t)$: the instantaneous travel cost on arc a at time t;
$\phi_a(t)$: the instantaneous marginal travel cost on arc a at time t;
$S_a[x_a(t)]$: the total travel cost on arc a when it contains x_a; and
λ: the minimum travel cost between members of an origin-destination pair encountered in a static user equilibrium problem.

REFERENCES

1. M. Beckmann, C. McGuire, and C. Winston. *Studies in the Economics of Transportation.* Yale University Press, New Haven, Conn., 1956.
2. D. K. Merchant. *A Study of Dynamic Traffic Assignment and Control.* Ph.D. dissertation. Cornell University, Ithaca, N.Y., 1974.
3. D. K. Merchant and G. L. Nemhauser. A Model and an Algorithm for the Dynamic Traffic Assignment Problems. *Transportation Science,* Vol. 12, 1978, pp. 183–199.
4. D. K. Merchant, and G. L. Nemhauser. Optimality Conditions for a Dynamic Traffic Assignment Model. *Transportation Science,* Vol. 12, 1978, pp. 200–207.
5. J. K. Ho. A Successive Linear Optimization Approach to the Dynamic Traffic Assignment. *Transportation Science,* Vol. 14, 1980, pp. 295–305.
6. M. Carey. A Constraint Qualification for a Dynamic Traffic Assignment Model. *Transportation Science,* Vol. 20, 1986, pp. 55–58.
7. M. Carey. Optimal Time-Varying Flows on Congested Networks. *Operations Research,* Vol. 35, 1987, pp. 58–69.
8. F. J. Luque, and T. L. Friesz. Dynamic Traffic Assignment Considered as a Continuous Time Optimal Control Problem. Prepared for presentation at the TIMS/ORSA Joint National Meeting, Washington, D.C., May 5–7, 1980.
9. V. F. Hurdle. The Effect of Queueing on Traffic Assignment in a Simple Road Network. *Proc., 6th International Symposium on Transportation and Traffic Theory,* Sydney, Australia, 1974.
10. C. Hendrickson and G. Kocur. Schedule Delay and Departure Time Decisions in a Deterministic Model. *Transportation Science,* Vol. 15, 1981, pp. 62–77.
11. H. S. Mahmassani, and R. Herman. Dynamic User Equilibrium Departure Time and Route Choice on Idealized Traffic Arterials. *Transportation Science,* Vol. 18, 1984, pp. 362–384.
12. H. S. Mahmassani and G-L Chang. Experiments with Departure Time Choice Dynamics of Urban Commuters. *Transportation Research,* Vol. 20B, 1986, pp. 297–320.
13. A. de Palma, M. Ben-Akiva, C. Lefevre, and N. Litinas. Stochastic Equilibrium Model of Peak Period Traffic Congestion. *Transportation Science,* Vol. 17, 1983, pp. 430–453.
14. M. Ben-Akiva, M. Cyna, and A. de Palma. Dynamic model of Peak Period Congestion. *Transportation Research,* Vol. 18B, 1984, pp. 339–355.
15. M. Ben-Akiva, A. de Palma, and P. Kanaroglou. Dynamic Model of Peak Period Traffic Congestion with Elastic Arrival Rates. *Transportation Science,* Vol. 20, 1986, pp. 164–181.
16. M. J. Smith. The Existence of a Time-Dependent Equilibrium Distribution of Arrival at a Single Bottleneck. *Transportation Science,* Vol. 18, 1984, pp. 385–394.
17. C. F. Daganzo. The Uniqueness of a Time-Dependent Equilibrium Distribution of Arrivals at a Single Bottleneck. *Transportation Science,* Vol. 19, 1985, pp. 29–37.
18. G. F. Newell. The Morning Commute for Nonidentical Travelers. *Transportation Science,* Vol. 21, 1987, pp. 74–82.
19. T. L. Friesz. Transportation Network Equilibrium, Design and Aggregation: Key Developments and Research Opportunities. *Transportation Research,* Vol. 19A, 1985, pp. 413–427.
20. A. S. Alfa. A Review of Models for the Temporal Distribution of Peak Traffic Demand. *Transportation Research,* Vol. 20B, 1986, pp. 491–499.
21. L. S. Pontryagin, V. A. Boltyanski, R. V. Gankrelidge, and E. F. Mishenko. *The Mathematical Theory of Optimal Process.* John Wiley & Sons, New York, 1962.
22. A. E. Bryson and Y.-C. Ho. *Applied Optimal Control,* rev. John Wiley & Sons, New York, 1975, pp. 246–270.
23. K. J. Arrow and M. Kurz. *Public Investment, the Rate of Return, and Optimal Fiscal Policy.* Johns Hopkins University Press, Baltimore, Md., 1970.
24. A. Seierstad and K. Sydsaeter. Sufficient Conditions in Optimal Control Theory. *International Economic Review,* Vol. 18, No. 2, 1977, pp. 367–391.
25. B.-W. Wie. *Dynamic User Equilibrium Traffic Assignment in Congested Multiple-Origin Single-Destination Networks.* Working paper. Department of City and Regional Planning, University of Pennsylvania, Philadelphia, 1987.
26. B.-W. Wie. *Dynamic User Equilibrium Traffic Assignment with Elastic Demand in Congested Multiple Origin-Destination Networks.* Working paper. Department of City and Regional Planning, University of Pennsylvania, Philadelphia, 1987.
27. B.-W. Wie. *The Day-to-Day Adjustment Mechanism of Stochastic Route Choice.* Working paper. Department of City and Regional Planning, University of Pennsylvania, Philadelphia, 1987.