AN ACCURATE AND EFFICIENT APPROACH TO EQUILIBRIUM TRAFFIC ASSIGNMENT ON CONGESTED NETWORKS

Larry J. LeBlanc, Southern Methodist University;
Edward K. Morlok, University of Pennsylvania; and
William P. Pierskalla, Northwestern University

This paper describes a new solution technique for the equilibrium traffic assignment problem. After existing methods of solution are reviewed and difficulties that have been experienced with current techniques are discussed, a mathematical programming model for the equilibrium traffic assignment problem is presented. The solution technique for this programming model is one that has been proved to converge every time and that rapidly closes in on the equilibrium flows without excessive computational requirements. It is noted that the computational requirement of the proposed approach is very similar to those of currently used solution methods, which clearly indicates the feasibility of using the proposed approach to find the equilibrium flows on networks with hundreds of nodes. Numerical results for the proposed solution technique on a test network having 76 arcs and 24 nodes are given. A computing time (central processing unit) of 6 seconds on the CDC 6400 computer is reported for accurately computing the equilibrium flows on the test network.

•THIS PAPER describes an efficient method for finding the equilibrium traffic flows on urban transportation networks. The problem is as follows: We are given a system of streets and zones representing a particular urban area, and we have estimates of the number of travelers (amounts of flow) who will drive between each pair of zones. It is well known (2) that the travel time along any street experienced by each driver depends on the number of vehicles flowing along the street. We assume that each driver will take the shortest (quickest) route between his origin and destination, and we wish to determine the traffic density on each street that results from the interaction among drivers as they congest the streets by traveling to their destinations. An equilibrium exists when a driver (increment of flow) cannot reduce his travel time by switching to another route between his origin and destination. Thus we wish to determine how the traffic between the zone pairs will be distributed over the streets of the city.

The equilibrium traffic assignment problem is an especially important one inasmuch as every metropolitan area experiences to some degree the serious problem of traffic congestion, notably during peak hours of movement. To improve an urban transportation system to meet projected demands for trips between each pair of zones in some future period requires that a model be developed for testing the proposed improvements. Alternatively, we may wish to determine whether the existing system can accommodate future increases in traffic without excessive congestion.

A system of streets and expressways is usually modeled by a network whose nodes represent major intersections and interchanges; the nodes are connected by directed arcs so that a two-way street is modeled by two arcs in opposite directions. The network is generally used to represent only the major streets of an urban area, whereas minor roads such as side streets in housing areas are usually not included.

An urban area is typically divided into zones. We assume that a matrix is available that specifies the expected number of trips between the various zones during the time

Publication of this paper sponsored by Committee on Transportation Systems Design.

period being studied. This matrix is called a trip table; the (i,j) entry equals the number of vehicles that must depart origin i to arrive at destination j. Each zone is identified by a node in the network. A node will not necessarily have any demand for trips associated with it; it may simply represent an intersection of two streets. All traffic that leaves any zone emerges into the network through its associated node or nodes, and traffic entering the zone leaves the network through the associated nodes. In this way, origin-destination estimates based on urban zones are transferred into origin-destination estimates based on nodes in the network. The model assumes that all traffic enters and leaves the network through the nodes.

The travel time experienced by a user of any road or arc, called the average travel time function or the volume-delay curve, is a known function of the total volume of flow along the road. Let $A_{ij}(x_{ij})$ denote the travel time experienced by each user of arc (i,j) when x_{ij} units of vehicles flow along the arc. For example, if arc (i, j) is 1 km long and vehicle speed is 30 km/h when the volume of flow on the arc is $\bar{x}_{ij}$, then $A_{ij}(\bar{x}_{ij}) = 2$ min. Almost all recent studies have recognized the effect that congestion of an arc has on travel time and have used nonlinear, increasing travel time functions. We assume that $A_{ij}(x_{ij})$ has continuous derivatives; this assumption is not at all restrictive. FHWA uses polynomial functions that have this property.

The travel time functions used by FHWA are shown in Figure 1 (2). The shape of the function $A_{ij}(x_{ij})$ is intuitive. As in the figure, the travel time per user increases very slowly at first; it remains almost constant for low levels of flow. However, as the flow begins to reach the level for which the arc (street) was designed, the travel time experienced by each user begins to increase rapidly. The a_{ij} and b_{ij} are empirically determined parameters for each arc, which depend on the arc's length, speed limit, and number of lanes and traffic lights. If there is a significant delay in making a left turn at an intersection (node), then turn penalties can be incorporated by using dummy arcs to represent the delay in making the turn.

Wardrop (10) has formulated two conditions that together formally characterize a network equilibrium. A set of flows along the arcs of a network is said to be at equilibrium if the following two conditions are satisfied for every origin-destination r-s pair:

1. If two or more routes between nodes r and s are actually traveled, then the cost to each traveler between r and s must be the same for each of these routes; and
2. There does not exist an alternative unused route between nodes r and s with less cost than that of the routes that are traveled.

The assumption is made that each user of the network seeks to minimize his own travel cost and that he experiments with different routes, eventually finding the least cost one. It is clear that, if 1 or 2 were not true, some drivers would switch to the cheaper routes, congesting them and causing a new flow pattern to evolve. Equilibrium is the aggregate result of individual decisions; at equilibrium, no single driver can reduce his own cost by choosing an alternative route in the network.

EXISTING SOLUTION TECHNIQUES FOR THE EQUILIBRIUM TRAFFIC ASSIGNMENT PROBLEM

The majority of solution techniques used today for finding the equilibrium flows on a network are simulation models, heuristic in nature, involving the concept of the shortest route between two nodes. One of the earliest techniques for this traffic assignment problem is called the "all-or-nothing" assignment technique. This method assumes that the travel time experienced by each driver on any arc in the network is a simple constant, independent of the flow level along the arc, and thus it completely ignores the very real problem of traffic congestion. The all-or-nothing technique is first to determine the shortest path between each origin-destination pair and then to assign all of the trips between this node pair to the shortest path. The all-or-nothing assignment technique is unstable: A slight change in the demand matrix can cause radical changes in the predicted arc volumes. Changing the demand has caused an arc's volume to change from the heaviest in the network to too few trips to justify its con-

Figure 1. Travel time functions used by the U.S. BPR.

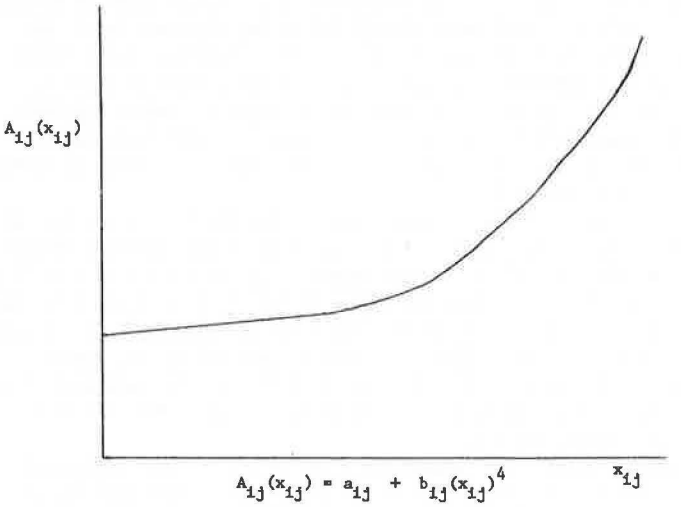
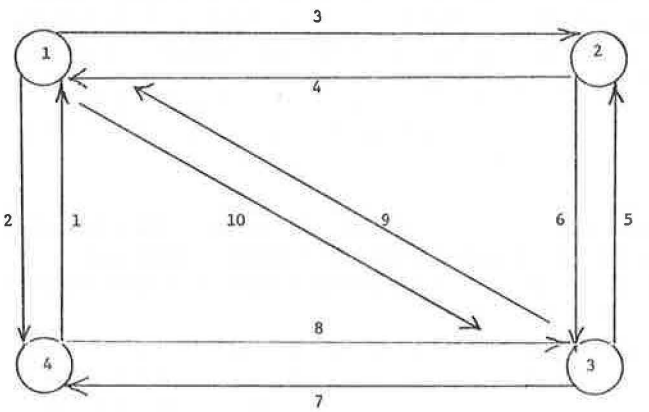


Figure 2. Test network.



struction. In view of the approximate nature of the demand estimates and of the travel time functions, this instability can seriously hinder the model's utility (3).

A natural extension of the all-or-nothing technique is to recognize the effect of congestion on the travel time on any arc. The effect of congestion is incorporated in the capacity restraint simulation models; these are traffic assignment procedures in which the travel time along an arc is adjusted according to a functional relationship between the design capacity and the volume of flow along the arc. An example of a capacity restraint model is the traffic assignment technique developed by the Chicago Area Transportation Study. In the CATS technique, one node is randomly selected from the network, and the minimum paths are determined from this node to all other nodes. All trips from the selected node are then assigned to the corresponding minimum paths. The network is then updated with new travel times calculated for the arcs in the minimum paths. The procedure is repeated with the random selection of another node and the computation of the new shortest routes.

The CATS procedure does not involve any iteration because the network is updated before the computation of each set of shortest routes, and thus the travel times are directly related to the volumes of flow along the arcs (6). However, the CATS procedure gives different flow patterns, depending on the order of selection of the origin nodes (2). Also the flow pattern produced by CATS does not really ensure that all users follow the shortest path between their origin and destination. For example, if node s is randomly selected during the early stages of the procedure, then all trips originating at node s are assigned to the shortest paths between node s and each destination by using the current arc travel times. But other nodes are subsequently generated, their trips are assigned to routes in the network, and the arc travel times are changed. The result is frequently that the paths used by travelers originating at node s are no longer the shortest paths to their destinations.

In an attempt to find the true equilibrium, iterative procedures are often used. Iterative procedures are simply continuations of the previous model; after all nodes have been generated, the model continues to generate nodes again. The rationale of these iterative models is similar to Charnes' game theoretic interpretation of the problem. Charnes associated with each origin a player who tries to choose a set of routes such that the correct number of vehicles will travel from the origin to each destination at minimum travel time. Because vehicles from the various origins interact, the travel times as seen by a given player depend on the actions of the other players. Thus an iterative technique is used to determine the equilibrium flows on the network. Each player chooses his routes in turn; after all the players have made their decisions, the resulting times along each arc are revealed to all the players, and they again take turns in revising their routes (4).

If the procedure described above is iterated enough times, the sequence of flow vectors may converge to an equilibrium. One possible termination criterion is to stop when the maximum percentage change between the components of two consecutive flow vectors is less than some specified amount. However, this iterative technique does not always produce a convergent sequence; examples are known where the sequence oscillates around a flow pattern that is not in equilibrium. In actual applications of large-scale problems, the practice is usually to terminate after four iterations (2).

The incremental assignment technique is a variation of the all-or-nothing method in which only a small increment of the total number of trips between any two nodes is assigned to the minimum path between the two nodes. In this technique, a node pair is randomly selected and the shortest path between these two nodes is determined. The length of each arc is set equal to the value of the arc's volume-delay function evaluated at the current level of flow along the arc; initially, the flow is zero. Then a small percentage of the total required flow is sent along this path, the flow level for each arc in the shortest path is incremented, and the new lengths of the arc are determined. Another node pair is then selected, and the process repeats itself until all traffic has been assigned. This represents an attempt to load the network in a balanced manner so that all arcs of the network approach the fully loaded condition at the same time. Thus the effect of congestion becomes more significant, and there is a better chance of achieving the conditions of network equilibrium. However, the incremental assign-

ment technique is time-consuming from a computational point of view inasmuch as a shortest route problem must be solved many times for each distinct origin-destination pair in the network (5). Also, it has not been proved that the incremental assignment technique converges to the equilibrium flows; thus for a particular problem, it may not produce the equilibrium flows.

As this brief review has indicated, there are a number of solution techniques in use for the traffic assignment problem. However, there have been difficulties with existing solution techniques, as a recent study in Lancaster, Pennsylvania (9), has shown. The study used data for the existing network in an attempt to replicate observed flows along the arcs by the unrestrained all-or-nothing technique and capacity restraint versions of this procedure involving one, two, three, and four iterations. To determine which of these five algorithms produced flows most closely resembling the observed flows, we made extensive comparisons of each assignment algorithm with the observed ground counts. A chi-square index was used as one means of comparison:

$$\sum_{i=1}^{n} (G_i - A_i)^2 / n$$

where

G_i = observed flow on street i,
A_i = flow on arc i predicted by the assignment algorithm, and
n = number of arcs in the network.

The authors used the chi-square index as an intuitive means of comparison; however, they report that, if the values of the chi-square were used in a statistical test, then "... all of the assignments would be rejected according to a Chi-Square test, since all of the values are significantly different from the ground count." As given below, the chi-square index actually increased after the first iteration, and, even after four iterations, it was still larger than the index of iteration number one:

Technique	Chi-Square Value
Unrestrained assignment	19,035
Iteration No. 1	12,597
Iteration No. 2	16,616
Iteration No. 3	14,599
Iteration No. 4	14,187

No mention of a confidence level is given. The authors' conclusion emphasizes their problem with lack of convergence: "It is not recommended that additional iterations of capacity restraint be made utilizing the same function or model because it has been concluded that the fourth iteration is only the third best assignment."

A chi-square test was also used in the National Cooperative Highway Research Program study (7) to check the all-or-nothing algorithm and several variations of the capacity restraint algorithms. Here it is also reported that "all of the values are significantly different from the ground count estimates, indicating that the difference in assignment is more than can be expected by chance alone."

Another criterion for comparing the output of each algorithm with the observed flows is that of total vehicle-miles in the network. In their report (7), Huber, Boutwell, and Witheford tested the ground count vehicle-miles and each algorithm's predicted vehicle-miles for the Pittsburgh network. The hypothesis tested was that the observations from the ground count and from four algorithms are all from the same normally distributed population. No justification of the assumption of normality is given. The authors again conclude that the hypothesis must be rejected. In fact, they reach the remarkable conclusion that all of the assignment algorithms are equally poor, stating that "... the various assignment techniques gave results which were closer to each other than to the ground count results." This clearly indicates the need for an improved equilibrium traffic assignment algorithm.

In the next section we will present a different model for the equilibrium traffic assignment problem. A solution technique that has been proved to converge every time and that rapidly closes in on the equilibrium flows without excessive computational requirements will then be described.

MATHEMATICAL PROGRAMMING MODEL FOR LARGE-SCALE NETWORK EQUILIBRIUM PROBLEMS

Consider a fixed network with n nodes, and assume that nodes 1, 2, . . ., p, $p \leq n$ are origins and destinations. Define A as the set of arcs (i, j) in the network. Let x_{ij} denote the total flow along the arc (i, j), let x_{ij}^s denote the flow along arc (i, j) with destination s, and let D(r, s) denote the fixed amount of flow required between nodes r and s. Obviously,

$$x_{ij} = \sum_{s=1}^{p} x_{ij}^s$$

As above, we let the average travel time function for arc (i, j) be denoted by $A_{ij}(x_{ij})$. Now define

$$f_{ij}(x_{ij}) \equiv \int_0^{x_{ij}} A_{ij}(t)dt$$

Using the definition $A_{ij}(x_{ij})$ in Figure 1, we see that

$$f_{ij}(x_{ij}) = \left[a_{ij}\left(\sum_{s=1}^{p} x_{ij}^s \right) + (b_{ij}/5)\left(\sum_{s=1}^{p} x_{ij}^s \right)^5 \right]$$

Then the optimal solution for the nonlinear programming problem

$$\text{(NLP) min} \sum_{(i,j)\in A} f_{ij}\left(\sum_{s=1}^{p} x_{ij}^s \right) = \min \sum_{(i,j)\in A} \left[a_{ij}\left(\sum_{s=1}^{p} x_{ij}^s \right) + (b_{ij}/5)\left(\sum_{s=1}^{p} x_{ij} \right)^5 \right] \tag{1}$$

$$\text{s.t.} \sum_{i\in B(j)} x_{ij}^s + D(j, s) = \sum_{k\in A(j)} x_{jk}^s \tag{2}$$

for j = 1, . . ., n and s = 1, . . ., p and

$$x_{ij}^s \geq 0 \quad (i, j) \in A \tag{3}$$

for s = 1, . . ., p constitutes the equilibrium flows. The objective function, Eq. 1, is the sum of the integrals of the average cost functions. In Eq. 2, B(j) is the set of nodes with arcs leading into node j (before j) and A(j) is the set of nodes with arcs leading into them from j (after j). The constraints of Eq. 2 are conservation of flow equations that state that, for each destination s, the sum of the flows into each node destined for s plus the flow originating at that node destined for s equals the sum of the flows out of that node destined for s. Constraints of Eq. 3 are simply the nonnegativity requirements.

Problem NLP is closely related to the work done by Kirchoff in electrical networks. Beckmann (1) seems to have been the first to apply the idea to transportation networks;

he proves that the solution to NLP is the desired equilibrium. Unfortunately, this problem appears very much harder to solve than current simulation models. The number of constraints in NLP is enormous—the number of conservation of flow constraints of Eq. 2 equals the product of the number of nodes in the network with the number of destinations in the network. Thus this nonlinear programming problem for a network with 200 nodes, 100 of which are destinations, would have 20,000 conservation of flow constraints. In addition there are the nonnegativity constraints. Potts and Oliver (8) state that computational success has been limited to small versions of problem NLP, and so the approach appears useless for realistically sized equilibrium traffic assignment problems.

However, a rigorous examination and exploitation of the structure of problem NLP reveal that this is not at all the case. LeBlane (5) used the Frank-Wolfe algorithm (11) to solve the equilibrium traffic assignment problem, e.g., problem NLP. This solution technique has proved to be remarkably accurate and efficient. In the Frank-Wolfe algorithm, Eq. 1 is replaced with a very simple linear approximation, and the linear programming problem of minimizing this linear approximation subject to Eqs. 2 and 3 is solved. The optimal solution to this linear programming problem is then used to define a search direction in which to minimize Eq. 1; the result of this search is an estimate of the equilibrium flows. After the search is completed in this generated direction, a new linear approximation is obtained, the linear programming problem is resolved with the new objective function to obtain a different direction of search, and a better estimate of the equilibrium flows is obtained by searching in this direction. The algorithm continues to iterate in this manner, solving one-dimensional searches and linear programming problems that minimize successively better linear approximations to the nonlinear objective function of Eq. 1.

An alternative procedure for solving NLP would be the usual method of linearization—approximate Eq. 1 with a piecewise linear function and use the simplex method to solve the resulting linear programming problem. The solution of problem NLP by linearization has been attempted in the past. Because linearization uses a more accurate linear approximation and hence does not solve a sequence of linear programming problems, it may seem that linearization is more efficient than the iterative technique from a computational point of view. However, this is not the case. Both solution techniques were coded on the CDC 6400 computer for a test network, and the computing time for the iterative technique was less than that of the usual linearization procedure by orders of magnitude. These numerical results are reported in the next section.

The key reason for the computational success of the Frank-Wolfe algorithm described above is that each of the linear programming problems has such an extremely simple structure that it can be solved by a shortest route algorithm. This means that all of the conservation of flow equations and nonnegativity constraints can be ignored; they are automatically satisfied by definition of a route between two nodes. It is well known in operations research literature that the computational requirements of a shortest route algorithm are trivial as compared to the requirements of a linear programming problem. The net result is that when problem NLP is solved by the Frank-Wolfe algorithm, the computational requirements of several shortest route problems and one-dimensional searches are vastly less than the computational requirements of the simplex method for solving NLP by linearization.

In the preceding section, a currently used iterative simulation technique based on Charnes' game theoretic interpretation of the equilibrium traffic assignment problem was described. LeBlanc (5) showed that each iteration of this simulation technique involves solving a shortest route problem that is identical to the shortest route problem solved at each iteration of the algorithm suggested in this paper for solving problem NLP. Thus we have the remarkable conclusion that the computational requirements of the proposed approach are not significantly different from the computational requirements of currently used simulation techniques for the equilibrium traffic assignment problem; so it is obvious that the proposed approach will be efficient for large problems.

The basic difference between the simulation technique and the algorithm suggested in this paper is that the proposed algorithm solves the shortest route problem to determine a direction of search and then minimizes the objective function in this direction to

obtain a new estimate of the equilibrium flows. The simulation procedure, on the other hand, uses the solution to this same shortest route problem itself as a new vector of flows. This leads to completely distinct flow vectors. These two approaches to the problem are fundamentally different. One is a simulation technique based on heuristic assumptions about the system; it frequently does not converge. The other is a rigorous application of a convergent algorithm to an NLP problem whose optimal solution is proved to be the equilibrium.

COMPUTATIONAL RESULTS

The Frank-Wolfe algorithm described was programmed in FORTRAN IV on a CDC 6400 computer; the test network shown in Figure 2 was used initially for debugging of the computer programs only. The network and travel time functions were chosen so that the equilibrium flows could be determined by inspection. Because the Frank-Wolfe algorithm converges to the equilibrium solution only after an infinite number of iterations, the primary concern was to determine how many iterations of the procedure are required for a reasonably accurate answer. The results are given in Table 1; after eight or 10 iterations, the flow values are probably more accurate than the data that are input to this type of model.

The algorithm was then run on a larger network consisting of 76 arcs and 24 nodes, each of which was both an origin and a destination. This network was used to model Sioux Falls, South Dakota, a city of approximately 125,000 residents. The network, trip table, and volume-delay functions are shown in Figures 3 and 4. Because the trip table in Figure 4 is symmetric—the number of trips between node i and node j equals the number of trips between nodes j and i—the equilibrium flow values will also be symmetric. In other words the equilibrium flow on arc (i, j) will be equal to the flow on arc (j, i), and thus we really need compute only 38 flow values. In this problem, one unit of flow was chosen to be 1,000 vehicles per day. Problem NLP for this network had 1,824 variables, 576 conservation of flow constraints, and 1,824 nonnegativity constraints. Because this is a general nonlinear programming problem, it is impossible to determine the exact solution in a finite amount of time. However, examination of the sequence of flows in Figure 5 shows that, after 20 iterations, only two variables changed by more than 5 percent; the majority changed by less than 2 percent. Thus the final flow vector appears to be a highly accurate estimate of the equilibrium solution. Computing time for 20 iterations, excluding 3 seconds of compilation time, was 6 seconds. If the termination rule had been to stop when the maximum percentage change in components was less than 8 percent, the procedure would have terminated after 16 iterations. After 16 iterations, the maximum percentage change in the components was 7.7 percent; computing time was 5 seconds.

The most encouraging computational result was the very small increase in the number of iterations required by the Frank-Wolfe algorithm for the two example problems. There were 12 conservation of flow constraints and 40 nonnegativity constraints for the initial network used for debugging, whereas the problem for the larger network in Figure 3 consisted of 576 conservation of flow equations and 1,824 nonnegativity constraints. Nevertheless, the required number of iterations increased from approximately eight or 10 to only 16 or 20. The number of iterations appears to be related to the number of nodes in the underlying network rather than to the number of constraints in the NLP problem. Increasing the number of nodes by a factor of six only doubled the number of iterations; this indicates that the algorithm will be computationally efficient for problems as large as several hundred nodes.

As mentioned earlier, problem NLP can also be solved by a more common form of linearization in which Eq. 1 is approximated by a piecewise linear function. This approach was also attempted on problem NLP for the network in Figure 3. However, the computing time was much greater: For this technique, the Optima package on the CDC 6400 required 700 seconds to solve NLP—more than 100 times as long as the Frank-Wolfe technique.

Figure 3. Sioux Falls network.

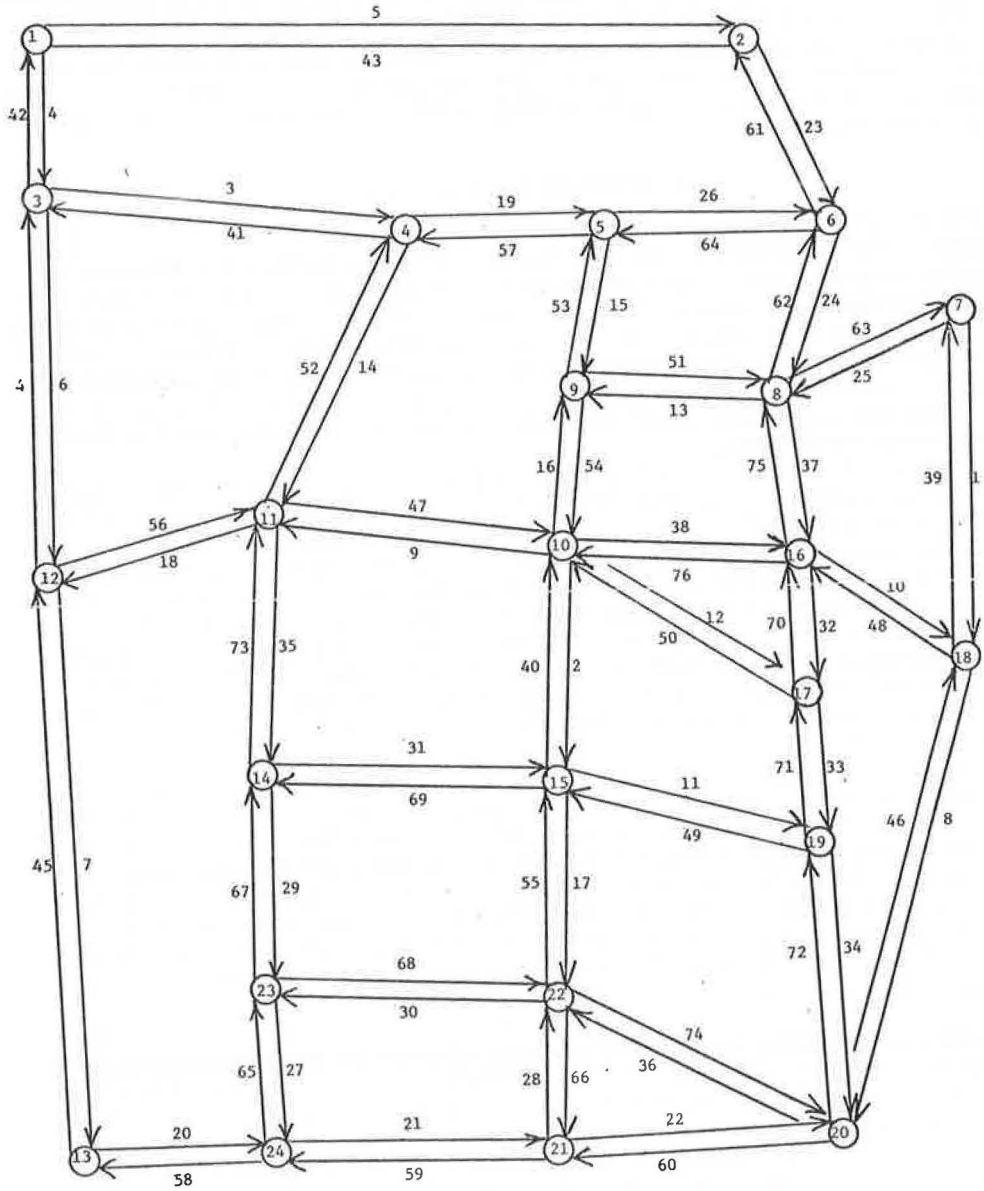


Table 1. Sequential estimates of equilibrium flows on 10 arcs of Figure 1.

Sequential Vectors of Equilibrium Flow	Arc 1	2	3	4	5	6	7	8	9	10
$x^{(1)}$	37.2	28.3	37.2	28.3	28.3	37.2	37.2	28.3	14.4	14.4
$x^{(2)}$	32.2	32.6	32.2	32.6	32.6	32.2	32.2	32.6	20.6	20.6
$x^{(3)}$	33.9	29.9	33.9	29.9	29.9	33.9	33.9	29.9	20.4	20.4
$x^{(4)}$	32.0	31.6	32.0	31.6	31.6	32.0	32.0	31.6	22.6	22.6
$x^{(5)}$	29.9	33.0	29.9	33.0	33.0	29.9	29.9	33.0	22.2	22.2
$x^{(6)}$	31.0	31.8	31.0	31.8	31.8	31.0	31.0	31.8	23.6	23.6
$x^{(7)}$	32.5	29.9	32.5	29.9	29.9	32.5	32.5	29.9	23.0	23.0
$x^{(8)}$	31.7	30.6	31.7	30.6	30.6	31.7	31.7	30.6	23.0	23.9
$x^{(9)}$	29.9	32.1	29.9	32.1	32.1	29.9	29.9	32.1	23.3	23.3
$x^{(10)}$	31.7	29.9	31.7	29.9	29.9	31.7	31.7	29.9	22.7	22.7
$x^{(11)}$	30.0	31.4	30.0	31.4	31.4	30.0	30.0	31.4	22.3	22.3
$x^{(12)}$	31.2	30.0	31.2	30.0	30.0	31.2	31.2	30.0	22.0	22.0
$x^{(13)}$	30.0	31.0	30.0	31.0	31.0	30.0	30.0	31.0	21.8	21.8
$x^{(14)}$	31.0	30.0	31.0	30.0	30.0	31.0	31.0	30.0	31.6	21.6
$x^{(15)}$	30.0	30.9	30.0	30.9	30.9	30.0	30.0	30.9	21.5	21.5
Equilibrium flow	30.0	30.0	30.0	30.0	30.0	30.0	30.0	30.0	20.0	20.0

Figure 4. Trip table and arc parameters for Sioux Falls network.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	0	1	1	5	2	3	5	8	5	13	5	2	5	3	5	5	4	1	3	3	1	4	3	1
2	1	0	1	2	1	4	2	4	2	6	2	1	3	1	1	4	2	0	1	1	0	1	0	0
3	1	1	0	2	1	3	1	2	1	3	3	2	1	1	1	2	1	0	0	0	0	1	1	0
4	5	2	2	0	5	4	4	7	7	12	14	6	6	5	5	8	5	1	2	3	2	4	5	2
5	2	1	1	5	0	2	2	5	8	10	5	2	2	1	2	5	2	0	1	1	1	2	1	0
6	3	4	3	4	2	0	4	8	4	8	4	2	2	1	2	9	5	1	2	3	1	2	1	1
7	5	2	1	4	2	4	0	10	6	19	5	7	4	2	5	14	10	2	4	5	2	5	2	1
8	8	4	2	7	5	8	10	0	8	16	8	6	6	4	6	22	14	3	7	9	4	5	3	2
9	5	2	1	7	8	4	6	8	0	28	14	6	6	6	9	14	9	2	4	6	3	7	5	2
10	13	6	3	12	10	8	19	16	28	0	40	20	19	21	40	44	39	7	18	25	12	26	18	8
11	5	2	3	15	5	4	5	8	14	39	0	14	10	16	14	14	10	1	4	6	4	11	13	6
12	2	1	2	6	2	2	7	6	6	20	14	0	13	7	7	7	6	2	3	4	3	7	7	5
13	5	3	1	6	2	2	4	6	6	19	10	13	0	6	7	6	5	1	3	6	6	13	8	8
14	3	1	1	5	1	1	2	4	6	21	16	7	6	0	13	7	7	1	3	5	4	12	11	4
15	5	1	1	5	2	2	5	6	10	40	14	7	7	13	0	12	15	2	8	11	8	26	10	4
16	5	4	2	8	5	9	14	22	14	44	14	7	6	7	12	0	28	5	13	16	6	12	5	3
17	4	2	1	5	2	5	10	14	9	39	10	6	5	7	15	28	0	6	17	17	6	17	6	3
18	1	0	0	1	0	1	2	3	2	7	2	2	1	1	2	5	6	0	3	4	1	3	1	0
19	3	1	0	2	1	2	4	7	4	18	4	3	3	3	6	13	17	3	0	12	4	12	3	1
20	3	1	0	3	1	3	5	9	6	25	6	5	6	5	11	16	17	4	12	0	12	24	7	4
21	1	0	0	2	1	1	2	4	3	12	4	3	6	4	8	6	6	1	4	12	0	18	7	5
22	4	1	1	4	2	2	5	5	7	26	11	7	13	12	26	12	17	3	12	24	18	0	21	11
23	3	0	1	5	1	1	2	3	5	18	13	7	8	11	10	5	6	1	3	7	7	21	0	7
24	1	0	0	2	0	1	1	2	2	8	6	5	7	4	4	3	3	0	1	4	5	11	7	0

ARC	A	B	ARC	A	B	ARC	A	B	ARC	A	B
1	.02	.00000001	20	.04	.00000893	39	.02	.00000001	58	.04	.00000893
2	.06	.00000027	21	.03	.00000790	40	.06	.00000027	59	.03	.00000790
3	.04	.00000007	22	.06	.00001373	41	.04	.00000007	60	.06	.00001373
4	.04	.00000002	23	.05	.00001241	42	.04	.00000002	61	.05	.00001241
5	.06	.00000002	24	.02	.00000521	43	.06	.00000002	62	.02	.00000521
6	.04	.00000002	25	.03	.00000119	44	.04	.00000002	63	.03	.00000119
7	.03	.00000001	26	.04	.00001001	45	.03	.00000001	64	.04	.00001001
8	.04	.00000002	27	.02	.00000451	46	.04	.00000002	65	.02	.00000451
9	.05	.00000075	28	.02	.00000401	47	.05	.00000075	66	.02	.00000401
10	.03	.00000003	29	.04	.00001020	48	.03	.00000003	67	.04	.00001020
11	.03	.00000010	30	.04	.00000960	49	.03	.00000010	68	.04	.00000960
12	.08	.00001930	31	.05	.00001085	50	.08	.00001930	69	.05	.00001085
13	.10	.00002306	32	.02	.00000401	51	.10	.00002306	70	.02	.00000401
14	.06	.00001550	33	.02	.00000554	52	.06	.00001550	71	.02	.00000554
15	.05	.00000075	34	.04	.00000958	53	.05	.00000075	72	.04	.00000958
16	.03	.00000012	35	.04	.00001061	54	.03	.00000012	73	.04	.00001061
17	.03	.00000052	36	.05	.00001130	55	.03	.00000052	74	.05	.00001130
18	.06	.00001550	37	.05	.00001157	56	.06	.00001550	75	.05	.00001157
19	.02	.00000003	38	.04	.00001080	57	.02	.00000003	76	.04	.00001080

Figure 5. Sequence of vectors of flow on (a) arcs 1 through 19 and (b) arcs 20 through 38.

(a)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	17.2	21.5	15.9	9.4	2.9	14.5	14.1	9.4	17.4	14.6	16.1	14.9	16.9	12.7	19.4	26.4	25.2	12.8	21.0
2	21.3	22.2	18.9	8.8	3.4	16.7	14.6	17.6	18.5	23.6	16.2	11.2	12.7	9.5	21.6	28.6	21.1	9.7	24.3
3	21.9	26.4	17.2	9.6	4.4	16.8	15.0	20.0	20.4	24.1	15.9	12.3	10.9	9.2	19.8	26.8	25.0	10.8	22.3
4	19.9	25.7	20.0	11.9	7.1	19.7	16.3	21.5	17.4	20.3	20.8	10.8	8.1	9.4	20.6	28.3	21.7	7.2	23.3
5	20.4	25.8	18.1	12.1	6.9	19.7	17.3	22.9	20.7	22.6	19.7	10.5	8.3	9.0	19.2	26.2	22.1	10.6	21.0
6	16.4	26.4	23.4	10.1	6.0	22.0	17.5	21.7	17.6	18.7	19.2	9.3	8.5	8.6	20.4	28.7	18.4	5.3	24.6
7	17.2	25.9	20.7	10.9	6.4	21.2	16.9	22.0	20.5	19.9	18.8	9.3	8.6	8.6	18.5	27.1	18.4	9.0	21.8
8	17.4	26.0	19.3	11.3	6.9	19.1	17.2	23.2	18.9	18.8	20.5	9.2	9.0	7.9	18.5	25.9	19.4	9.4	20.6
9	16.7	24.5	18.4	10.3	6.3	16.9	16.5	22.8	20.6	19.1	18.8	8.9	7.2	7.6	17.2	25.0	20.6	9.6	20.7
10	17.6	25.1	19.3	10.6	6.1	17.0	16.7	22.8	19.3	19.6	19.0	8.9	8.5	7.3	18.8	25.6	19.9	9.3	21.8
11	17.6	25.9	17.2	11.9	7.6	15.1	16.1	23.3	20.8	19.1	19.7	8.8	7.3	6.8	17.4	24.8	19.6	9.6	20.0
12	17.3	25.7	18.1	11.9	7.2	15.8	16.3	22.8	19.6	19.1	19.4	8.8	8.2	7.3	18.5	24.7	19.1	8.8	20.9
13	17.3	24.5	17.9	11.4	6.7	15.1	16.2	22.9	19.3	18.9	19.4	8.8	7.5	7.2	17.8	24.3	19.2	9.2	20.7
14	17.9	25.1	18.4	11.3	6.7	15.3	16.1	22.6	18.8	19.3	19.6	8.8	8.3	7.0	18.9	24.9	19.1	8.7	21.4
15	17.6	25.4	18.2	11.0	6.4	14.6	16.0	22.4	19.2	19.0	19.5	8.7	7.8	7.9	17.2	23.4	19.0	9.2	20.4
16	17.5	25.2	18.6	11.0	6.4	14.9	16.0	22.5	18.9	18.9	19.7	8.7	8.4	7.7	18.4	23.8	19.1	8.7	21.1
17	17.5	25.2	17.6	10.7	6.8	14.4	16.0	22.7	20.1	18.8	19.5	8.7	8.1	7.4	17.9	23.7	18.4	9.1	20.5
18	17.5	25.2	18.0	10.7	6.6	14.6	15.9	22.5	19.4	18.3	19.7	8.7	8.5	7.3	18.6	24.0	18.8	8.7	20.9
19	17.5	25.5	18.0	10.5	6.3	14.2	15.8	22.8	19.3	18.3	19.5	8.7	7.9	7.1	18.1	23.8	19.2	8.9	21.0
20	17.4	24.9	18.1	10.5	6.3	14.5	15.9	22.7	19.1	18.1	19.6	8.7	8.3	7.2	18.4	23.8	18.9	8.6	21.1

(b)

	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
1	13.1	8.8	7.1	8.4	11.8	14.6	7.3	10.6	9.4	14.0	13.8	11.6	19.2	23.7	10.3	12.7	10.6	16.0	11.1
2	12.1	11.8	12.8	6.0	14.6	18.2	10.2	11.7	11.3	11.0	10.4	11.6	20.5	17.9	12.6	10.0	8.0	12.1	16.1
3	12.2	10.4	11.0	6.4	14.8	18.5	10.1	12.2	11.1	10.4	12.3	10.2	18.4	15.3	10.7	10.0	13.2	10.3	13.8
4	12.2	11.5	11.5	7.9	15.5	16.0	9.5	11.6	12.6	10.5	8.9	12.0	14.4	13.5	14.3	13.3	8.8	9.9	11.1
5	12.6	9.9	9.4	7.3	13.7	16.0	8.3	11.8	12.1	9.7	10.9	10.8	13.9	11.8	11.8	11.6	11.7	8.1	13.0
6	12.3	12.3	8.9	7.0	15.5	15.0	11.6	10.4	11.4	10.2	7.1	11.2	14.5	11.7	8.3	8.9	8.6	10.7	11.5
7	11.4	10.7	8.1	7.0	14.1	15.1	9.9	10.3	10.6	10.7	9.6	9.9	13.2	11.3	8.7	9.9	9.0	9.1	12.9
8	12.3	10.1	8.2	7.2	13.3	14.6	8.5	10.5	9.9	9.2	11.0	10.7	11.9	10.8	11.1	10.6	8.7	9.2	11.2
9	12.2	11.3	7.9	7.1	14.7	13.7	10.2	9.3	10.5	8.7	10.4	10.0	12.9	11.1	8.2	11.2	9.6	9.5	12.3
10	12.1	10.6	7.8	6.7	13.8	15.0	9.5	9.4	9.9	9.0	10.2	9.8	12.3	10.9	9.0	10.6	9.4	8.6	11.8
11	11.9	11.1	8.1	7.8	14.6	14.4	8.7	8.6	9.5	8.3	10.1	10.8	11.3	9.4	11.0	10.5	8.8	8.7	11.4
12	12.0	10.3	7.8	7.4	13.8	14.2	8.3	8.6	9.3	9.0	9.9	10.1	11.7	9.8	10.1	10.4	8.9	9.0	11.5
13	12.1	11.4	7.7	7.2	14.2	13.8	9.1	9.7	10.1	8.8	11.5	9.7	11.8	9.9	9.3	11.1	8.7	9.0	11.9
14	12.0	10.9	7.7	7.1	13.7	14.8	8.6	9.3	9.7	8.9	10.6	9.5	11.8	9.9	9.4	10.7	8.4	8.3	11.7
15	11.8	10.7	7.5	6.9	14.1	14.6	9.3	8.6	9.5	8.9	9.7	10.4	12.2	10.4	9.0	10.2	8.3	8.9	11.8
16	11.9	10.4	7.5	6.8	13.6	14.5	8.8	8.8	9.4	8.7	10.5	10.1	11.8	10.1	9.5	10.4	8.3	9.0	11.6
17	12.0	10.9	7.5	7.3	14.0	14.3	8.7	8.4	9.6	9.2	9.6	9.3	11.9	10.0	9.1	10.2	8.2	8.9	12.0
18	11.9	10.9	7.5	7.1	13.6	14.3	8.5	8.6	9.6	8.9	10.4	10.1	12.0	10.1	9.2	10.4	8.3	9.0	11.7
19	11.8	10.1	7.5	6.9	14.0	14.1	9.2	8.1	9.0	8.4	10.2	9.4	12.0	10.0	8.9	9.9	8.2	8.9	12.0
20	11.9	10.5	7.5	6.8	13.7	14.1	8.8	8.4	9.1	8.5	10.3	10.1	11.7	10.0	9.4	10.6	8.2	9.0	11.8

CONCLUSIONS AND RECOMMENDATIONS

In this paper we have addressed the problem of finding the equilibrium traffic flows on urban networks. In particular, we have looked at the computational aspects of large-scale equilibrium problems. The algorithm that was discussed above promises to be efficient for finding the equilibrium on a network with hundreds of nodes, since its most difficult computational requirements are identical to those of the iterated capacity restraint simulation models of traffic assignment currently used. And yet the above algorithm is a rigorous one; it is proved theoretically that it always converges to the exact equilibrium. This algorithm has demonstrated its capability by solving a large-scale nonlinear programming problem (576 linear constraints and 1,824 variables and nonnegativity constraints) in 6 seconds on the CDC 6400 computer. Even this small computing time could certainly be reduced by examining the computer program in detail and by making it more sophisticated and more efficient. Further research is needed, however, to determine exactly how large a network can be handled in a reasonable amount of computer time. This question is of particular interest to transportation planners inasmuch as the assumption that a true equilibrium is achieved is almost universally used in system models used to support such planning. Current methods of network equilibrium analysis are known to be inaccurate (in that they often do not converge to an equilibrium) and very costly, making the potential payoff from research on better methods very substantial. The problem of selecting a suitable test network and appropriate data must also be addressed. Finally, suitable comparisons for the outputs of different assignment models must be chosen.

REFERENCES

1. Beckmann, M., McGuire, C. B., and Winsten, C. Studies in the Economics of Transportation. Yale Univ. Press, 1956.
2. Comsis Corporation. Traffic Assignment: Methods-Applications-Products. Office of Planning, Federal Highway Administration, preliminary rept., 1972.
3. Dial, R. B. Probabilistic Assignment: A Multipath Traffic Assignment Model Which Obviates Path Enumeration. Dept. of Civil Engineering, Univ. of Washington, PhD dissertation, 1970.
4. Hershdorfer, A. M. Predicting the Equilibrium of Supply and Demand: Location Theory and Transportation Network Flow Models. TRF Papers, 1966, pp. 131-143.
5. LeBlanc, L. J. Mathematical Programming Algorithms for Large Scale Network Equilibrium and Network Design Problems. Dept. of Industrial Engineering and Management Sciences, Northwestern Univ., PhD dissertation, 1973.
6. Martin, B. V., and Manheim, M. L. A Research Program for Comparison of Traffic Assignment Techniques. Highway Research Record 88, 1965.
7. Huber, M. J., Boutwell, H. B., and Witheford, D. K. Comparative Analysis of Traffic Assignment Techniques With Actual Highway Use. NCHRP Rept. 58, 1968.
8. Potts, R. B., and Oliver, R. M. Flows in Transportation Networks. Academic Press, New York, 1972.
9. Starasinic, F. N., and Schuster, J. J. Comparative Analysis of Capacity Restraint Traffic Assignments. Traffic Engineering, 1972, pp. 48-52.
10. Wardrop, J. G. Some Theoretical Aspects of Road Traffic Research. Proc. Institution of Civil Engineers, Vol. 1, Part II, 1952, pp. 325-378.
11. Zangwill, W. Nonlinear Programming: A Unified Approach. Prentice-Hall, 1969.