An Algorithm for the Traffic Assignment Problem

Andrés Weintraub
Departamento de Industrias Universidad de Chile, Santiago, Chile
Jaime González
Departamento de Matemáticas Universidad de Chile, Santiago, Chile

An optimization algorithm for the traffic assignment problem is presented. It is based on determining, for the flow corresponding to each origin at a time, a set of disjoint circuits with the most negative total marginal cost, followed by a unidimensional optimization along these circuits. The proposed method was compared to other well known approaches on randomly generated networks. In most cases the proposed method proved to be highly efficient, particularly when appropriate tighter stopping rules were used. In this sense, it was seen that when stopping rules were based on insufficiently tight tolerances, substantial errors could appear in the equilibrium traffic assignment.

1. INTRODUCTION

This paper considers the problem of determining an equilibrium traffic assignment when demands are fixed. A procedure based on an approach different from those presently available is presented which determines optimal solutions for the equivalent problem: multicommodity network flows of minimum cost on an uncapacitated network with convex, continuous derivative cost functions. The method is based on determining, for a given commodity at a time a set of disjoint circuits with minimal total marginal cost (calculated using partial derivatives), followed by a unidimensional optimization (Fibonacci search) along these circuits.

In Sec. 2 the traffic assignment problem is described, with the introduction of some known optimization techniques used in the solution of this problem [1-3].

In Sec. 3 the proposed algorithm is presented, along with some needed proofs.

In Secs. 4 and 5 the algorithm is detailed and a flow chart of it is analyzed.

Finally (Sec. 6), a comparison of algorithms is presented. Based on computational experience, the proposed algorithm appeared to be highly efficient. The advantage of the proposed algorithm increased as the tolerances for stopping rules were tightened. A point is made in reference to stopping rules, where it is seen that stopping rules not tight enough may often lead to substantial errors in the evaluation of equilibrium traffic assignments.

© 1980 by John Wiley & Sons, Inc.
0028-3045/80/0010-197$01.30

2. TRAFFIC ASSIGNMENT PROBLEM

The problem of determining an equilibrium traffic assignment has been well studied. For a network of roads and zones representing given areas, an estimate is made of the travel demand between given pairs of nodes called origins and destinations. Associated with each zone is an origin and/or destination node. All traffic originating or terminating in this zone is assumed to enter or leave the network at its associated nodes. Each pair represents the total traffic between two zones. It is of interest to know how the traffic generated between each pair of nodes at given times will distribute itself. It should be noted that travel demand is not independent of traffic assignment. Congestion may reduce it. This problem has been approached in two forms: (1) sequentially determining traffic assignment and travel demand until an equilibrium is reached, or (2) solving both problems simultaneously as in [6, 17] where the problem is solved through a combination of a fixed demand algorithm and Bender's decomposition. The decision making of the driver can take two forms. According to Wardrop's principles [4, 5] the equilibrium in one case (principle I) is obtained when each driver takes the path of least cost available to him (private or user optimization) while in the other case (principle II) an overall minimum cost for all drivers is sought (social or system optimization).

The cost of flow along a path is obtained simply by adding the cost corresponding to the road segments along the path. Principle II implies centralized decision making, a situation which hardly occurs in reality. However, a simple change in the cost function transforms a problem from user optimized to system optimized [4, 5].

Note that in a user optimized equilibrium, all paths between an origin-destination pair carrying flow between the corresponding pair have equal average cost, while all paths not carrying flow corresponding to this pair have a higher or equal average cost. If this were not the case, drivers could decrease their cost by unilaterally switching paths, a contradiction with an equilibrium situation. A similar analysis, based on the Kuhn-Tucker conditions, is valid for a system optimized equilibrium, when marginal costs are considered [4, 5].

Several assumptions are made in the determination of equilibrium points:

Assumption (a). Drivers along the roads perceive impedances or costs, which depend on travel time, mileage, driving tension, etc.

This "generalized" cost or disutility has been studied and functions can be established to determine the cost perceived by a driver on each segment of road as a function of total flow on that segment. As congestion augments, disutility will naturally tend to augment. Experimentally, curves relating travel cost and road flow have proved to be nonlinear and strictly increasing [7].

Assumption (b). Cost functions are known to drivers and they make rational decisions.

Consider now the following multicommodity network flow problem. Let E be a directed network of m nodes and n arcs. An arc (i, j) joins nodes i and j. x_{ij}^q is the value of flow on arc (ij) corresponding to commodity q, which indexes the originating node of flow on the arcs.

Then, find flows $\{x_{ij}^q\}$ that minimize $\Sigma_{ij} c_{ij}(\Sigma_q x_{ij}^q)$ such that

$$\sum_j x_{ij}^q - \sum_j x_{ji}^q = \begin{cases} f^q, & i = q, \\ 0, & i \neq q, i \notin D_q, \\ -f_i^q, & i \neq q, i \in D_q, \end{cases} \tag{1}$$
$$x_{ij}^q \geqq 0 \quad \forall (ij), q,$$

where D_q is the set of destinations of commodity flow q, and the c_{ij} are convex functions. Naturally,

$$f^q = \sum_{i \in D_q} f_i^q.$$

If we call

$$x_{ij} = \sum_q x_{ij}^q,$$

the total flow on arc (ij), the objective function becomes min $\Sigma\, c_{ij}(x_{ij})$. In the traffic assignment situation, road segments are viewed as arcs, and origins, destinations, and road intersections as nodes. Modelling vehicles as flow on the arcs, it is simple to see that the solution to the multicommodity network flow problem described in (1), where we associate a commodity to the flow originating at each origin, corresponds to the solution of a system optimized traffic assignment with traffic flow considered as continuous.

In ref. [12] it is shown that a sufficient condition for approximating the real problem, where flows have integer values (number of vehicles), to one where flows are considered continuous is that varying the flow along any route by one unit does not measurably alter the cost corresponding to the other vehicles in that route.

Several algorithms have been proposed to determine equilibrium solutions through solving a problem of the form of (1). Among the best known we can mention are (A) Leventhal, Nemhauser, and Trotter, which is based on a column generation approach [1]; (B) Nguyen, which is based on a convex simplex approach [2]; and (C) LeBlanc, Morlok, and Pierskalla, which is based on a specialized Frank-Wolfe approach [3].

The algorithms proposed in (A), (B), and (C) share several features. In every iteration, each algorithm uses first a shortest-route-type algorithm to generate a direction of variation of flow for a new solution and a unidimensional optimization procedure is carried out next to determine an optimal flow reallocation in that direction.

The difference in efficiency among them will be given then by the amount of work required to generate the direction of variation and the number of iterations required to reach an acceptable solution. Since (1) is a nonlinear program, an exact solution cannot be obtained and stopping rules have to be defined consistent with the degree of accuracy desired. In Sec. 6 the problem of adequate stopping rules is discussed.

3. PROPOSED ALGORITHM

An algorithm is presented to solve Problem (1).

We will call the flow corresponding to each origin a commodity q, $q = 1, Q$. Given any feasible solution to (1): $x = \{x_{ij}^q\}$, $q = 1, Q$, we will define the marginal cost in the direction (ij) of an arc joining nodes i and j for commodity q as

$$c_{ij}'^q = \begin{cases} c_{ij}'(x_{ij}) & \text{if the direction of the arc is } (ij), \\ -c_{ji}'(x_{ji}) & \text{if the direction of the arc is } (ji) \text{ and } x_{ji}^q > 0, \\ \infty & \text{if the direction of the arc is } (ji) \text{ and } x_{ji}^q = 0. \end{cases}$$

The algorithm determines at each iteration, for a given commodity q, a set of disjoint circuits such that the total sum of marginal costs on all arcs is most negative. (A set of disjoint circuits is such that to each node in the network corresponds exactly one incoming and outgoing arc. This includes self-loops. This set will be called a circuit-set.) Along the direction given by these circuits a unidimensional search determines the actual flow that is optimum to redirect along each circuit. A flow dX^q in commodity q along a circuit will be called circuit-flow. A flow DX^q in commodity q along a circuit-set will be called circuit-set flow.

Given a solution $X = (x_{ij}^q)$ to problem (1), a circuit-flow $dX^q = (dx_{ij}^q)$ along a circuit C^k will be called negative if

$$\sum_{(ij)\in C^k} c_{ij}(x_{ij} + dx_{ij}^k) < \sum_{(ij)\in C^k} c_{ij}(x_{ij}),$$

where dx_{ij}^k is positive if the circuit flow dX has the same direction as x_{ij} on arc (ij), negative otherwise.

Analogously, a circuit-set flow $DX^q = (Dx_{ij}^q)$ along a circuit-set CS^q will be called negative for a given solution $X = \{x^q\}$ to (1) if

$$\sum_{(ij)\in CS^q} c_{ij}(x_{ij} + Dx_{ij}^q) < \sum_{(ij)\in CS^q} c_{ij}(x_{ij}).$$

Lemma 1. The set of circuits in a network E is a subset of the set of circuit-sets.

Proof. Given any circuit, a circuit-set is easily constructed using this circuit and self loops on all remaining nodes of the network. ■

Lemma 2. Given a solution X, negative circuit flows exist iff negative circuit-set flows exist.

Proof. By Lemma 1, and considering that in any negative circuit-set flow at least one of the disjoint circuits of the circuit-set must have negative circuit flow. ■

Theorem 1. A solution to (1) is optimal iff no negative circuit flow exists in any commodity when the marginal costs are considered for each arc [4, 5].

Corollary 1. A solution X to (1) is optimal iff no negative circuit-set flow exists in any commodity, when marginal costs are considered for each arc.

Proof. By Theorem 1 and Lemma 2. ■

4. DETERMINATION OF A NEGATIVE SET OF CIRCUIT-SET FLOWS FOR THE MARGINAL COST MATRIX IN A COMMODITY q

Consider the following problem in commodity q:

$$\begin{gathered}\sum_i x_{ij} = 1,\\ \sum_j x_{ij} = 1,\\ 0 \leqq x_{ij} \leqq 1,\\ \text{Min} \sum_{i \neq j} c_{ij}'^q x_{ij}.\end{gathered} \tag{2}$$

Problem (2) corresponds to an assignment problem on a network E_1 where the costs assigned to each pair (ij) are the marginal costs $c_{ij}'^q$ for commodity q when $i \neq j$, and 0 if $i = j$ (self-loops).

This assignment problem approach has been used previously for the one-commodity network flow problem [8]. The algorithm used is a dual one first proposed by Edmonds and Karp [9]. The upper bound of additions and comparisons required is $O(m^3)$, but in practice the algorithm is considerable faster, depending on the density of the network [10, 19]. An alternative approach (not tested) uses the primal methods developed for this special form of the transportation problem [14]. In both cases, it is more efficient to start each assignment problem with the solution of the previous iteration.

Note that a solution to Problem (2) corresponds to a circuit-set, with minimal total marginal cost in commodity q.

5. FLOW CHART OF THE PROPOSED ALGORITHM

Step 0. Determine a feasible solution to problem (1). Set $q = 1$ and $F = 0$. F corresponds to the number of consecutive times a commodity has been examined with no "significant" improvement in the objective function.

Step 1. Consider commodity q. Determine the marginal cost matrix for that commodity.

Step 2. Determine a circuit-set CS^q in commodity q of minimal marginal cost (assignment problem).

Step 3. For each circuit C_r^q in the circuit set CS^q not a self-loop, determine, within a certain tolerance, an optimal flow reallocation leading to an improvement in the objective function: dC_r^q.

Step 4. If

$$\sum_r dC_r^q > \epsilon,$$

ϵ a defined tolerance, take $F = 0, q = q + 1$ $(q = 1$ if $q + 1 > Q)$. Go to 1. If

$$\sum_r dC_r^q \leqq \epsilon,$$

take $F = F + 1$. If $F < Q$, take $q = q + 1$ $(q = 1$ if $q + 1 > Q)$ and go to 1. If $F = Q$, end.

Step 0 can be determined using an incremental shortest route approach. That is, starting with flow $F_0^q = 0$, for each commodity $q = 1, Q$, determine the shortest route for every origin-destination pair (O^q, D^q), with costs $c'_{ij}(0)$ in every arc. Assign, for every commodity q a fraction F_1^q (say 10%) of the total flow along that route.

New costs are calculated on arcs based on the flows $F_0^q + F_1^q$ and shortest routes are again determined. A new fraction F_2^q of the flow of commodity q is assigned to the new shortest route between all pairs (O_q, D_q), $q = 1, Q$. This process is continued until all the flow in all commodities is allocated.

This approach has been also used to determine approximate traffic assignments before the introduction of optimization approaches. Another approach used for the tests done is to determine just a reasonable feasible starting flow assignment with a simple, quick procedure (all or nothing type allocation).

Step 1 is a straight forward calculation. Step 2 is described in Sec. 4.

In Step 3 a one-dimensional optimization process is required to determine k,

$$\left(k \leqslant \min_{(ji)\in C_r^q} x_{ji}^q \right) \min \left[\sum_{(ij)\in C_r^q} (c_{ij}(x_{ij} + k) - c_{ij}(x_{ij})) + \sum_{(ji)\in C_r^q} (c_{ij}(x_{ij} - k) - c_{ij}(x_{ij})) \right]$$

for each circuit $C_r^q \in CS^q$.

A Fibonacci search was implemented to solve this problem and used in all methods tested.

Step 4 is the implementation of the stopping rule. If in sequence the improvement in the objective function in all commodities is below a given tolerance ϵ, the solution is considered to be close enough to an optimal. A formal stopping rule can be used based on accepted tolerances in flow and cost measures as done in [2] (and leading to lower bounds). A flow will be considered an equilibrium solution if, for each commodity, to any path carrying a flow of that commodity greater than ϵ_1 there corresponds the same sum of costs along the arcs, within a deviation ϵ_2. Here ϵ_1, ϵ_2 are parameters for tolerances in, respectively, measuring flows significantly different from zero and differences in costs perceived by drivers. A simple labeling routine can be used to check if this condition is satisfied.

6. COMPARISON OF METHODS

The method proposed above (Method D) was compared with the three other methods mentioned in Sec. II; Leventhal et al. (Method A), Nguyen (Method B), and LeBlanc et al. (Method C). Each method was programmed according to the steps suggested in the corresponding paper. For Methods A and D, the stopping rule is based on having marginal or total improvements in the objective function below a given tolerance. In method B, the stopping rule is based on acceptable tolerances for errors in flow values on arcs and total times along paths (dual variables). In Method C the stopping rule is given by considering that from one iteration to the next, the flow on no arc varies by more than a given percentage (LeBlanc et al. suggest 5%). A lower bound for an optimal solution can be determined based on convexity of the objective function.

After several tests, the stopping rule procedure decided on for the purpose of comparison for Methods A to D was to first run Method C with its stopping rule, and then run Methods A, B, and D, with a stopping rule so that in each case the objective value of the solution was as close as possible to the one of Method C.

Table I defines the problems tested. Problems 1-11 were generated randomly, based on the number of nodes and arcs for the network. The number of origins defines the number of commodities involved in Problem (1). The cost functions were of the form $c_{ij}(x) = a_{ij}x + b_{ij}x^5$. In problems 1-11 approximately 25% of the nodes were origins. Problems 12 and 13 are replications of Problems 4 and 3 in which the number of origins was doubled. Problems 14 and 15 had only the cost function generated randomly while the network structure followed a road network pattern, designed deterministically. Problems 16 and 17 are substantially larger. Results of test runs made on Problems 1-15 are detailed in [11]. Leventhal et al. was the least efficient, by a large margin. Nguyen also appeared to be less efficient than Methods C and D, but in this case, it is clear that significant savings can be obtained through variations in the design of steps of the algorithm, in particular in terms of the strategy for going from one commodity to the next.

This would explain results given in [13], where for one case study (city of Winnipeg) Nguyen was about 40% faster than LeBlanc et al.

We present now a condensed comparison of Methods C and D.

TABLE I. Description of problems.

Problem	Nodes	Arcs	Origins
1	9	16	2
2	9	16	2
3	12	30	3
4	12	30	3
5	12	30	3
6	15	40	4
7	15	40	4
8	20	68	5
9	20	110	5
10	30	100	8
11	30	100	8
12	12	30	6
13	12	30	6
14	12	34	12
15	9	24	4
16	100	361	8
17	200	726	12

Table II shows comparisons of Methods C and D for the Problems 1-11. Comparisons are given in terms of number of iterations (Fibonacci searches), CPU seconds on an IBM 370-145 to reach the tolerances of 5% and 1% of the LeBlanc et al. stopping rule, and the ratios of CPU seconds per iteration.

It can be seen that:

(a) The proposed method was faster than LeBlanc et al. in most cases.

(b) The advantage of the proposed method over LeBlanc et al. increases as the tolerance becomes tighter. Problems of slow convergence for the Frank-Wolfe Algorithm, in particular in late stage iterations, are well known. In [16], a modified version of this approach was presented which provided, in the major case study, an improvement of about 20% in CPU time.

(c) For the smaller tolerance, there seems to be a clearer trend in terms of increasing the relative efficiency of the proposed method as the size of the network grows.

Problems 12 and 13 were run to test the effect of increasing the number of commodities (Table III). Theoretically, due to the fact that the proposed method works separately on each commodity while that of LeBlanc et al. works globally on the problem, there should be comparative advantages for LeBlanc et al. as the number of commodities increases.

In the tests of Problems 4-12, 3-13, and Problem 14, while not sufficient for conclusive evidence, there was such an indication for the case of 100% origins, but not for 50% origins.

TABLE II. Comparison of Methods C and D.

		Method C LeBlanc et al.			Method D Weintraub and Gonzalez		
Problem	Stopping Rule	Iterations	CPU sec	Iteration Time	Iterations	CPU sec	Iteration Time
1	5%	6	0.73	0.12	3	0.52	0.17
	1%	18	2.76	0.15	8	1.60	0.20
2	5%	6	0.88	0.15	2	0.39	0.19
	1%	12	2.11	0.18	5	1.23	0.25
3	5%	30	6.72	0.22	19	5.33	0.28
	1%	86	22.88	0.27	35	10.38	0.30
4	5%	14	3.39	0.24	23	6.0	0.27
	1%	17	4.57	0.27	23	6.3	0.28
5	5%	23	5.56	0.24	20	5.58	0.28
	1%	69	19.61	0.28	23	7.15	0.31
6	5%	104	35.85	0.35	23	7.64	0.33
	1%	223	84.72	0.38	61	21.16	0.35
7	5%	157	52.73	0.34	19	7.30	0.38
	1%	201	71.29	0.35	50	20.89	0.42
8	5%	36	19.70	0.55	15	5.79	0.39
	1%	180	120.35	0.67	19	8.56	0.45
9	5%	23	20.08	0.87	70	35.77	0.51
	1%	63	57.69	0.92	88	48.60	0.55
10	5%	36	34.25	0.95	17	10.60	0.62
	1%	69	70.78	1.03	18	11.84	0.66
11	5%	19	17.32	0.91	17	10.29	0.60
	1%	87	50.32	0.58	45	26.82	0.59

TABLE III. Comparison of Methods C and D (CPU sec).

Stopping Rule	LeBlanc et al.	Weintraub and Gonzalez
Problem 3		
5%	6.72	5.33
1%	22.88	10.38
Problem 4		
5%	3.39	6.01
1%	4.57	6.30
Problem 12		
5%	11.02	20.56
1%	24.76	82.44
Problem 13		
5%	4.31	2.24
1%	27.84	3.43

Problems 14 and especially 15 (Table IV) had the objective of seeing if the randomness of the generation of networks rather than using typical road patterns influenced results. There was an indication here that having 100% origin nodes was significant, rather than the pattern of the network.

Problems 16 and 17 (100 and 200 nodes) were run to make a comparison of the algorithms on larger problems. Due to computer cost considerations, these problems were run only up to "close to" the 5% stopping rule, i.e., Method C was stopped when there remained a relatively small number of arcs (about 6% of the total number of arcs) on which the change of flow from one iteration to the next varied by more than 5%.

The results given in Table V show that for both cases the proposed method was faster. Note that the relaxation in the stopping rule for this problem should favor

TABLE IV. Comparison of Methods C and D (CPU sec).

Stopping Rule	LeBlanc et al.	Weintraub and Gonzalez
Problem 14		
5%	5.10	20.17
1%	10.08	23.01
Problem 15		
5%	5.11	1.97
1%	25.24	3.21

TABLE V. Comparison of Methods C and D (CPU sec).[a]

	LeBlanc et al.			Weintraub and Gonzalez		
Problem	Iterations	CPU (sec)	Time/ Iteration	Iterations	CPU (sec)	Time/ Iteration
16	170	602.7	3.54	149	205.3	1.38
17	175	1384.0	7.91	170	634.2	3.70

[a]Stopping rule "close to 5%."

LeBlanc et al. since the experience in smaller problems is that the proposed method improves in the comparison with respect to LeBlanc et al. as the stopping rule becomes tighter, and this was also observed to occur in these two problems.

There are well known difficulties in comparing algorithms through CPU time, even if all methods are programmed by the same person with same intended effort and using common subroutines (Fibonacci searches).

Total CPU times were in some cases sensitive to Fibonacci tolerances (a smaller interval requires more evaluations but leads to fewer total iterations). Only a few tests were done to determine good interval strategies for the unidimensional search.

For LeBlanc et al. the intervals chosen were 0.025 until the 5% tolerance was reached and 0.005 from then to the end. For the proposed method tolerances were about 0.001. These intervals are considered in a unidimensional search of total length 1.

To determine shortest routes, Djistra's algorithm [15] was used for all problems, except 14, on which Floyd's algorithm [15] was used. No improvements developed for these algorithms [15] were tested.

Modifications could improve the programs. On the one hand, CPU time spent on unidimensional searches of the polynomial functions could be reduced by replacing the Fibonacci algorithm with Bolzano searches, which work with the derivatives of the cost functions. This change would be more effective for LeBlanc et al., as in general the method of LeBlanc et al. requires a higher expenditure of effort for unidimensional searches than the proposed method (up to 70% of total CPU time in Fibonacci searches for LeBlanc et al.). This is because in the method of LeBlanc et al., the order of magnitude of evaluations (multiplications) is about the number of arcs, while in the proposed method it is about the number of nodes. The proposed method uses a smaller tolerance interval, but this effect is of lesser importance.

On the other hand, there is obviously room for improvement in the determination of direction of variation of flow, with more careful programming, and in the proposed method this part uses up a larger fraction of total CPU time than in LeBlanc et al. In particular, the assignment problem subroutine used in the proposed method, which is considerably more complex than Djistra's, could probably be substantially improved, given that only a fair effort was put into its programming.

Tests were made independently [19], where pure assignment problems were run without taking advantage of starting with "good" initial solutions, as is done in the presented algorithm. Results indicate that for networks of densities similar to the ones common in traffic assignment problems, CPU times grow only somewhat faster than linearly with the number of nodes. Thus, using the assignment algorithm as a sub-

TABLE VI. Comparison of Methods C and D.[a]

Problem		LeBlanc et al.				Weintraub and Gonzalez			
		Z	CPU (sec)	$\bar{e}/\bar{f}(\%)$	$DZ/Z^*(\%)$	Z	CPU (sec)	$\bar{e}/\bar{f}(\%)$	$DZ/Z^*(\%)$
1	5%	13.7×10^4	0.73	6.2	3.0	13.7×10^4	0.52	3.0	2.8
	1%	13.3×10^4	2.76	2.0	0.3	13.3×10^4	1.60	1.2	0.2
	Final	13.3×10^4	15.05	0.2	0.0	13.3×10^4	5.15	0.0	0.0
3	5%	49.8×10^4	6.72	1.7	0.2	49.8×10^4	5.33	1.1	0.2
	1%	49.8×10^4	22.88	0.5	0.0	49.8×10^4	10.38	0.3	0.0
	Final	49.7×10^4	37.25	0.3	0.0	49.7×10^4	31.0	0.0	0.0
8	5%	31.8×10^3	19.70	27.7	42.5	31.8×10^3	5.79	29.1	42.3
	1%	28.6×10^3	120.35	20.5	27.8	27.6×10^3	8.56	23.8	23.4
	Final	27.7×10^3	307.01	19.8	23.8	22.4×10^3	154.36	0.0	0.0
10	5%	21.0×10^4	34.65	15.4	17.2	20.7×10^4	10.60	17.8	15.3
	1%	20.7×10^4	73.02	13.5	15.2	20.6×10^4	11.84	17.5	15.0
	Final	19.9×10^4	487.55	9.0	10.7	17.9×10^4	279.08	0.0	0.0
11	5%	7226.	17.32	7.2	2.7	7223	10.29	5.8	2.6
	1%	7071.	50.33	4.6	0.5	7072	26.82	2.2	0.5
	Final	7036.	130.73	3.4	0.0	7036	80.10	0.0	0.0
13	5%	34.9×10^5	4.31	3.6	1.0	34.8×10^5	2.24	3.4	0.7
	1%	34.7×10^5	27.84	1.6	0.3	34.7×10^5	3.43	2.1	0.3
	Final	34.6×10^5	60.00	0.6	0.1	34.5×10^5	29.38	0.0	0.0
15	5%	638.	5.11	12.2	24.6	641	1.97	19.7	25.2
	1%	543.	25.24	3.8	6.0	533	3.21	8.3	4.0
	Final	538	30.09	3.2	5.1	512	9.33	0.0	0.0

[a]Evaluation of errors in objective value and arc flow assignments vs. CPU times: ($\bar{e}/\bar{f}$, % error in arc flows; DZ/Z^*, % error in objective value; Z, objective value at that stage). Final solutions determined by tight stopping rule for Method D. (ϵ about 0.1% of z, for 2 successive cycles.)

routine to determine circuits on which flow should be varied should not cause CPU time problems in larger networks.

A point that was considered of interest was to determine the error corresponding to solutions that stop at the 5% and 1% tolerances. Table VI presents the results for Problems 1, 3, 8, 10, 11, 13, 15. The final solution considered was given by a tight stopping rule for Method D.

Errors can be measured in two forms: (i) by the difference in objective value between an intermediate solution (z) and a final (z^*), or (ii) by the differences in arc flow assignments between an intermediate solution and a final solution.

The error in objective values was measured as a percentage based on $Dz/z^* = (z - z^*)/z^*$.

The error in arc flows was measured in percent based on

$$\bar{e}/\bar{f}, \text{where } \bar{e} = \sum_{(ij)} (f_{ij} - f_{ij}^*)$$

and

$$\bar{f} = \sum_{(ij)} f_{ij}^*,$$

for f_{ij} flow on arc (ij) at an intermediate solution, f_{ij}^* at the final.

We can see that in almost half the problems tested, in particular the larger ones, the errors at the 5% and even 1% tolerance level were quite substantial, possibly unacceptable for normal accuracy requirements. In some cases the percent error was higher measured in flows, in others as an objective value. These results would indicate the need to exert care in the tolerances accepted as stopping rules. Note that these values are estimates of the errors, since the solutions obtained were not exact optima.

The problems of slow convergence for LeBlanc et al. can be seen particularly in problems 8 and 10 in Table VI, where the deviations from a final solution were still very substantial, and decreasing slowly, when the runs were stopped, indicating an extremely high projected CPU time for reaching an acceptable solution.

The proposed method requires a far larger amount of computer memory than LeBlanc et al., thus needing auxiliary memory. As is indicated in [13], this increases computer costs somewhat and also requires more sophisticated programming. However, the efficiency in terms of CPU time of the proposed method can compensate for this factor and make this approach interesting.

References

[1] T. L. Leventhal, G. L. Nemhauser, and L. E. Trotter, "A Column Generation Algorithm for Optimal Traffic Assignment," *Transp. Sci.*, 7, 168–176 (1973).

[2] S. Nguyen, "An Algorithm for the Traffic Assignment Problem," *Transp. Sci.*, 8, 203–216 (1974).

[3] L. J. LeBlanc, E. K. Morlok, and W. P. Pierskalla, "An Efficient Approach to Solving the Road Networks Equilibrium Traffic Assignment Problem," *Transp. Res.*, 9, 309–318 (1975).

[4] S. C. Dafermos and F. T. Sparrow, "The Traffic Assignment Problem for a General Network," *J. Res. Nat. Bur. Stand. Sec. B*, 37(2) 1969.

[5] R. B. Potts and R. M. Oliver, *Flows in Transportation Networks*, Academic, New York, 1972.
[6] M. Florian, S. Nguyen and J. Ferland, "On the Combined Distribution Assignment Problem," *Transp. Sci.*, **9**, 43-53 (1975).
[7] R. M. Michaels, "Attitudes of Driver Determine Choice Between Alternate Highways," *Public Roads*, **34**, 225-236 (1965).
[8] A. Weintraub, "A Primal Algorithm to Solve Network Flow Problems with Convex Costs," *Manag. Sci.*, **21** (1), 87-97 (1974).
[9] J. Edmonds and R. Karp, "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems," *J. ACM.*, **19**, 248-264 (1972).
[10] A. Weintraub, "The Shortest and K-Shortest Routes as Assignment Problems," *Networks*, 3, 61-73 (1973).
[11] A. Weintraub and J. Gonzalez, "An Efficient Algorithm for the Traffic Assignment Problem," Working Paper 77/07/C, Departmento de Industrias, University of Chile (1972). Presented at the TIMS/ORSA Meeting, Atlanta, GA, November 1977.
[12] A. Weintraub, "Optimal Flows and Games: The Multicommodity Flow Problem in Integers," Working Paper 76/21/C, Departmento de Industrias, University of Chile (1976).
[13] M. Florian and S. Nguyen, "An Application and Validation of Equilibrium Trip Assignment Methods," *Transp. Sci.*, **10**, 374-390 (1976).
[14] R. S. Barr, F. Glover, and D. Klingman, "The Alternating Basis Algorithm for Assignment Problems," *Math. Prog.*, **13**, 1-13 (1977).
[15] G. H. Bradley, "Survey of Deterministic Networks," AIIE Trans. 7 N, 222-234 (1975).
[16] M. Florian, "An Improved Method for a Multicommodity Convex Cost Flow Problem," presented at the ORSA/TIMS Meeting, Atlanta, GA, November, 1977.
[17] M. Florian and S. Nguyen, "A Method for Computing Network Equilibrium with Elastic Demands." *Transp. Sci.*, **8**, 321-332 (1974).
[18] S. Nguyen, "A Unified Approach to Equilibrium Methods for Traffic Assignment," presented at the International Symposium of Traffic Equilibrium Methods, Montreal 1975.
[19] A. Weintraub, and F. Barahona, "A Dual Algorithm for the Assignment Problem," Working Paper 79/02/C, Departamento de Industrias, Universidad de Chile.

Received January 1978
Revised October 1979