A Multi-Commodity Network Flow Problem with Applications to the Network Equilibrium Problem

by Larry J. LeBlanc*, Edward K. Morlok ,** William P. Pierskalla***

Introduction

We consider the following multi-commodity network flow problem. Given a network with n nodes, let nodes 1, 2, ..., p, $p \leq n$ be a set of nodes which are either supply points or demand points or both. Typically, p = n, i.e., every node is a supply point and/or a demand point. In addition, every node in the network will be considered a transshipment point. Let A denote the set of arcs in the network and let D be a p x p matrix whose i, j entry indicates the number of units which must flow between nodes i and j. To write the conservation of flow equations insuring that D(j, s) units flow from node j to node s, we introduce the following notation:

x_{ij}^{s} - flow along arc (i, j) with destination s

x_{ij} - total flow along arc (i, j)

---

* Department of Computer Science and Operations Research, Southern Methodist University, Dallas, Texas.

** Department of Civil and Urban Engineering, University of Pennsylvania, Philadelphia, Pennsylvania.

*** Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois.

We then must have

$$D(j,s) + \sum_i x_{ij}^s = \sum_k x_{jk}^s \qquad \begin{matrix} j = 1,\ldots,n \\ s = 1,\ldots,p \\ j \neq s \end{matrix}$$

These constraints state that for every node j, for every destination s, the flow which originates at node j destined for node s plus the transiting flow destined for s must equal the total flow which leaves node j destined for s. Thus, for each destination node s we have a system of conservation of flow equations; one equation for all nodes j in the network, $j \neq s$. We may omit the equation for $j = s$, since it is generally true that if we require the correct amount of flow to leave each supply point, and if flow is conserved at every intermediate node, then the equation requiring the flow to arrive at the demand point is redundant. Let the cost of shipping x_{ij} units along arc (i, j) be $f_{ij}(x_{ij})$. The cost function $f_{ij}(\cdot)$ is a function of the total flow along the arc, i.e., the sum of all the commodities flowing along the arc. The multi-commodity transshipment problem which we address is then

$$\text{(NLP)} \qquad \text{MIN} \sum_{(i,j)\in A} f_{ij}\left(\sum_{s=1}^{p} x_{ij}^s\right) \tag{1}$$

$$\text{s.t.} \quad D(j,s) + \sum_i x_{ij}^s = \sum_k x_{jk}^s \qquad \begin{matrix} j = 1,\ldots,n \\ s = 1,\ldots,p \\ j \neq s \end{matrix} \tag{2}$$

$$x_{ij}^s \geq 0 \quad (i,j)\in A \quad s = 1,\ldots,p \tag{3}$$

The (NLP) problem above is a special case of the multi-commodity minimum cost flow problem. Each demand point (or destination) in the underlying network has a set of demands which can only be satisfied by flow from specified supply

points (or origins). For example, at destination j, the demand for goods from node i is D(i, j); this demand can be satisfied only by flow originating at node i. The total flow into node j is not comingled as is the case in the classical transshipment problem. Thus, to provide that each destination receives the correct amount of flow from the appropriate origins, we label each unit of flow along the arcs of the network with its respective ultimate destination.

Observe that it is not uncommon for each node in the network to serve as an origin, destination and transshipment point for flow between other origin-destination pairs.

A more intuitive, although computationally more difficult, form for the conservation of flow equations, is as follows: Define

x_{ij}^{rs} - flow along arc (i,j) which originates at node r and is destined for node s

x_{ij} - total flow on arc (i,j); $x_{ij} = \sum_{rs} x_{ij}^{rs}$

Our problem is then

$$\text{MIN} \sum_{(i,j)\in A} f_{ij} \left(\sum_{r,s} x_{ij}^{rs} \right) \tag{1'}$$

$$\sum_k x_{ij}^{rs} - \sum_k x_{jk}^{rs} = \begin{cases} -D(r,s) & j = r \\ 0 & j \neq r,s \\ D(r,s) & j = s \end{cases} \quad \begin{array}{ll} j = 1,\ldots,n & \\ r,s = 1,\ldots,p & \\ & r \neq s \end{array} \tag{2'}$$

$$x_{ij}^{rs} \geq 0 \quad (ij)\in A \quad r,s = 1,\ldots,p \quad r \neq s \tag{3'}$$

Constraints (2') state that the flow into node j traveling from node r to node s equals the flow out of node j traveling from r to s (unless j = s). For any specified

network, the constraints (2') are clearly much greater in number than the constraints (2).

An Example: The Network Equilibrium Problem

The network equilibrium problem, or the equilibrium traffic assignment problem, may be stated as follows: We are given a set of streets and zones representing a particular urban area, and we have estimates of how much traffic must travel between each pair of zones. We wish to determine how this traffic will distribute itself over the streets of the city. Typically, we have projections of the traffic between each zone pair in some future period, and we wish to determine if the existing street network can handle the increased traffic without excessive congestion. A system of streets and expressways may be modeled by a network whose nodes represent major intersections and interchanges. 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, while 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 specifying the expected number of trips between the various zones during the time period being studied. This matrix is usually called a trip table; the i,j entry equals the number of vehicles which must depart origin i to arrive at destination j. Each zone is identified with a node or nodes in the network. A node will not necessarily have any demand for trips associated with it - the node may simply represent an intersection of two streets. All traffic which 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 the 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. 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. The parameters of the travel time function are determined by its 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.

Suppose then that we have a fixed network for an urban area, and a trip table indicating the projected number of trips between each origin destination pair in some future period. Assume that it has been proposed that some additional arcs be constructed, such as a new expressway or a bridge across a river. The fixed network will include these proposed arcs. We wish to determine how many vehicles will use the new arcs to see if their construction is justified. In general, we wish to determine how the flows will distribute themselves over all of the arcs of the network. This is the equilibrium traffic assignment problem, i.e. given a trip table and a network for a particular urban area, determine how many vehicles will flow along each arc of the network.

One of the most common behavioral assumptions made in traffic assignment algorithms is that each user of the urban network will take the path of least resistance from his origin to his destination. It is known that travel time is a significant factor to the majority of travelers when they choose their routes between their respective origins and destinations, and most applications

of traffic assignment utilize travel time as the basic measure of travel [Comsis, 1972]. Distance is a less important criterion, but tension also plays an important role in the selection of a route. Michaels [1960, 1962, 1965] has concluded that it is the net impedance or total disutility - a composite measure of travel time, tension and other factors - that drivers seek to minimize. For example, if a traveler is faced with a choice between two routes of equal travel time, he may prefer not to choose the one which is characterized by heavy stop-and-go traffic. Tension is primarily caused by the grade or curvature of the street and variations in street surface, etc. In the remainder of this paper, the terms travel time and travel cost will be used to mean time, tension, distance, dollars or some weighted combination of these.

The assumption that each driver takes the path of least resistance between his origin and destination gives rise to the concept of network equilibrium. A set of flows along the arcs of the network is said to be at equilibrium if the following two conditions are satisfied for every origin-destination pair, r - s.

(1) If two or more routes between node r and node s are actually traveled, then the cost to each traveler between r and s must be same for each of these routes.

(2) There does not exist an alternative unused route between nodes r and s with less cost than that of the routes which 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. Although this may not be completely true in reality, it is assumed that those drivers using more costly routes constitute a negligible portion of the total. 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. An equilibrium is the aggregate result of individual decisions.

At an equilibrium, no single driver can reduce his own cost by choosing an alternative route in the network.

If the travel time of every arc were constant, independent of the level of flow along the arc, then we could solve shortest route problems between each origin-destination pair to find the equilibrium flows. However, the assumption of constant travel time ignores the effect that congestion of an arc has on travel time. In the more realistic case of increasing, nonlinear travel time functions, the interaction among drivers makes the problem of finding the equilibrium very complex. The two equilibrium conditions above are equivalent to Wardrop's first principle, the principle of equal travel times for all users [Wardrop, 1952]. His second principle, that of overall minimization, leads to a different assumption of driver behavior. Wardrop's second principle states that flows are distributed over the arcs of the network in such a manner that the sum of the travel times for all users is minimized. Models based on the assumption of equilibrium are often referred to as "user-optimal" models, while models based on Wardrop's second principle are referred to as "system optimal" models. The second behavioral assumtion, which uses the system optimal flows to approximate the equilibrium flows, typically appears in urban transportation problems more general than the traffic assignment problem.

The network flow problems usually treated in the Operations Research literature are representative of system optimal models. Typical of these problems where a decision maker chooses the flows on each arc to minimize the cost incurred by the system (the sum of the individual costs) is the problem of minimizing the shipping costs between the factories and warehouses of a company.

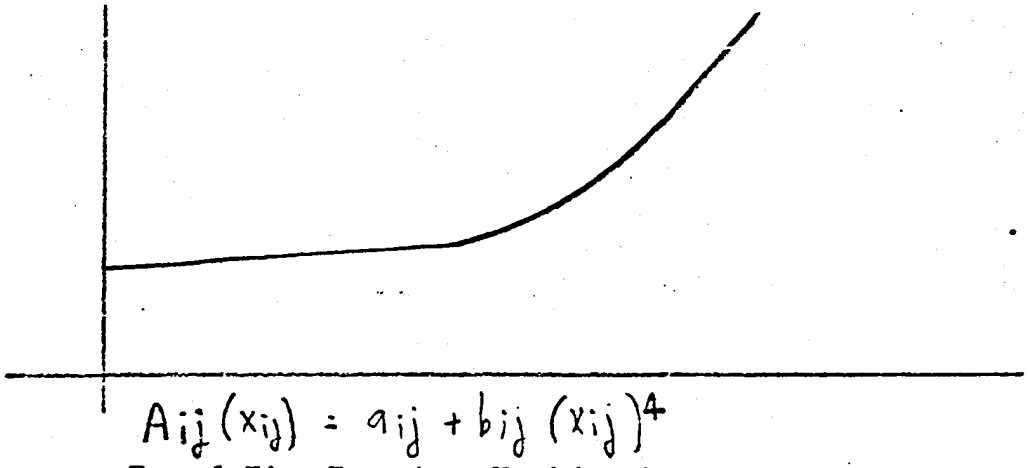
The fundamental difference between the equilibrium flows and the system optimal flows is that at the system optimal flows, some users of the network may incur an unnecessarily high cost, allowing the majority of the users to have a greatly reduced cost. If this reduces the company's total shipping cost, then this would certainly be optimal. In the urban transportation context, this type of cost reduction does not occur. Each individual user of the network chooses his own path, and he wishes only to minimize his own travel time. In the equilibrium assignment problem, we must find the pattern of traffic flows which results from many individuals competing for transportation between each pair of nodes in the network; that is, the set of flows satisfying the equilibrium conditions (1) and (2), regardless of what the sum of the individual costs is. However, as will be seen shortly, the equilibrium problem is equivalent to a mathematical programming problem in the usual sense. Since there is really no central controller to direct the flows in an urban transportation road network, the concept of network equilibrium seems to be an accurate model of the system, and thus the great majority of existing solution techniques are based on the assumption of equilibrium.

A Nonlinear Programming Model of the Traffic Assignment Problem

Consider a fixed network with n nodes, and assume that nodes 1, 2, . . . ,p $p \leq n$ are origins and destinations. Define A to be the set of arcs (i,j) in the network. Let x_{ij} denote the total flow along arc (i, j), let $x_{ij}^{\sim}$ denote the flow along arc (i, j) with destination s and let x_{ij}^{rs} denote the flow along arc (i, j) with origin r and destination s. Obviously, $x_{ij}^{\cdot} = \sum_{r,s=1}^{p} x_{ij}^{rs}$ and $x_{ij}^{\sim} = \sum_{r=1}^{p} x_{ij}^{rs}$. Let $A_{ij}(x_{ij})$ equal the travel time experienced by each user of arc (i, j) when x_{ij} units of vehicles flow along the arc. For examples, if arc (i, j) is one mile long, and the speed per vehicle is thirty miles per hour when the volume of flow is x_{ij}, then $A_{ij}(x_{ij})$ = 2 minutes. We

is an increasing function; it is taken to be nonlinear because of the effects of congestion on the travel time for each user of any arc.

We assume that the $A_{ij}(\cdot)$ are increasing functions with continuous derivatives. This assumption is not all restrictive--the U. S. Bureau of Public Roads uses polynomial functions which satisfy these properties. These functions are of the form in Figure 1; a_{ij} and b_{ij} are empirically determined parameters for each arc which are computed from its length, speed limit and number of lanes and traffic lights [Comsis, 1972]. The shape of the average cost function $A_{ij}(\cdot)$ 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 average travel time functions used in this paper will be those of the U.S.B.P.R., although other functions could be easily substituted.



Travel Time Functions Used by the U.S.B.P.R.

Figure 1

Now define $f_{ij}(x_{ij}) = \int_0^{x_{ij}} A_{ij}(t)dt = a_{ij}x_{ij} + (b_{ij}/5)(x_{ij})^5$. Then the optimal flow values for the problem

$$\text{(NLP)} \quad \text{Min } f(x) = \text{Min} \sum_{(i,j)\in A} f_{ij}\left(\sum_{s=1}^{p} x_{ij}^{s}\right)$$

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

$$\text{s.t.} \quad D(j,s) + \sum_{i} x_{ij}^{s} = \sum_{k} x_{jk}^{s} \qquad j = 1,\ldots,n \quad s = 1,\ldots,p \quad j \neq s \qquad (2)$$

$$x_{ij}^{s} \geq 0 \quad (i,j)\in A; \quad s = 1,\ldots,p \qquad (3)$$

constitute the equilibrium flows. The summations in (2) are over $i \in B(j)$ and $k \in A(j)$; we will drop this cumbersome notation. Problem (NLP) is closely related to the work done by Kirchoff in electrical networks. The proof is based on the Kuhn-Tucker conditions; see [Beckmann, 1956], [LeBlanc, 1973].

The objective function (1) contains many cross products involving fifth degree terms. There is one constraint (2) for every node j, for every destination s in the network, $j \neq s$. There is one variable for each arc, for each destination in the network. Thus if we wish to find the equilibrium on a 500 node, 2,000 arc grid network in which each node is an origin and a destination, then problem (NLP) has $500 \cdot 499 = 249{,}500$ conservation of flow constraints and $2000 \cdot 500 = 1{,}000{,}000$ variables and non-negativity constraints. In this paper, we will present an efficient method for solving such large scale versions of (NLP). First we give a lemma which is fundamental to the remainder of this paper.

Lemma II.1 If $A_{ij}(\cdot)$ is a nondecreasing continuously differentiable function on $[0,\infty)$, then the objective function (1) is convex with respect to the x_{ij}^{rs}.

Proof: It is only necessary to show that each f_{ij} is convex with respect to the $x_{ij} = \sum_{rs} x_{ij}^{rs}$. Then, since x_{ij} is a linear function of the x_{ij}^{rs}, each f_{ij} will be convex with respect to the x_{ij}^{rs}. Now $f_{ij}'(x_{ij}) = A_{ij}(x_{ij})$, and since A_{ij} was assumed continuously differ-

entiable and nondecreasing--the travel time per user does not get smaller as a road gets more congested--then $A'_{ij}(x_{ij}) = f''_{ij}(x_{ij})$ is non-negative on $[0,\infty)$. Therefore, each f_{ij} is a convex function of the x^{rs}_{ij}. Since the sum of convex functions is convex, the proof is completed.

Therefore, any local optimal solution to (1), (2), (3) is also a global optimal solution, and hence is the desired equilibrium flow. When $A_{ij}(\cdot)$ is strictly increasing, the objective function (1) is strictly convex with respect to the x_{ij}, but is not strictly convex with respect to the x^{rs}_{ij}. Although the optimal solution to the NLP problem above may not be unique, the total flows along each arc at equilibrium (e.g., the $x_{ij} = \sum_{rs} x^{rs}_{ij}$) are unique in the strictly increasing case.

An Algorithm for the Network Equilibrium Problem

Now if x^1 is a set of flows which satisfies constraints (2) and (3'), then we can use a first order Taylor's Expansion for the objective function $f(\cdot)$ to write, for any y,

$$f(y) = f(x^1) + \nabla f(x^1 + \theta(y - x^1))\cdot(y - x^1) \quad \text{some } \theta \in [0,1]$$

Here, the gradient of f is being evaluated at some point between x^1 and y. A convenient linear approximation to f(y) is to let θ equal zero

$$f(y) \approx f(x^1) + \nabla f(x^1)\cdot(y - x^1) \tag{4}$$

If we solve the linear programming problem in y

$$\text{(P)} \qquad \underset{y \geq 0}{\text{Min}} \ \nabla f(x^1)\cdot y + \left[f(x^1) - \nabla f(x^1)\cdot x^1\right]$$

$$\text{s.t. } D(j,s) + \sum_i y^s_{ij} = \sum_k y^s_{jk} \qquad \begin{matrix} j = 1, \ldots, n \\ s = 1, \ldots, p \\ j \neq s \end{matrix} \tag{5}$$

getting y^1 as the optimal solution, then y^1 is also a feasible solution to (NLP), since the constraints are identical for both problems. Note that the term $[f(x^1) - \nabla f(x^1) \cdot x^1]$ is a constant, so we may omit it from the linear programming problem. The direction $d^1 = y^1 - x^1$ is then a good direction along which to minimize f [Zangwill, 1969]. Since the feasible region (2), (3) is convex, any point on the line segment between x^1 and y^1 will also satisfy the constraints. Thus to minimize f in the direction d^1, we solve the one dimensional problem

$$\begin{aligned} &\text{Min} \quad f(x^1 + ad^1) \\ &a \in [0,1] \end{aligned} \tag{6}$$

It is well known that a Bolzano search is an efficient technique for solving this problem [Zangwill, 1969]. After solving the one dimensional problem, we get a new feasible point, x^2.

Lemma $\lim_{n \to \infty} x^n = x^*$, where x^* is optimal for (NLP).

See Zangwill [1969] for a proof of this. This algorithm of iteratively solving one dimensional searches and linear programming problems which minimize successively better approximations to f near x^* is known as the Frank-Wolfe algorithm. One nice aspect of this algorithm is that at every iteration we have a lower bound on the optimal value of the NLP problem. By convexity, we have

$$f(x^*) \geq f(x^k) + \nabla f(x^k) \cdot (x^* - x^k)$$

where x^* is the optimal solution, and x^k is the feasible solution at iteration k. Also, we have that

$$f(x^k) + \nabla f(x^k)\cdot(x^* - x^k) \geq f(x^k) + \nabla f(x^k)\cdot(y^k - x^k)$$

since y^k minimizes $\nabla f(x^k)\cdot y$ for every feasible y. Therefore $f(x^k) + \nabla f(x^k)\cdot(y^k - x^k)$ is a lower bound on $f(x^*)$, for every k. One possible stopping criterion is to stop when the current functional value is within a given epsilon of this lower bound.

Unfortunately, if we wish to find the equilibrium on the 500 node network mentioned above, then the problem (NLP) and consequently each subproblem, has a prohibitively large number of constraints and variables. Not only are the linear programming subproblems apparently unsolveable, but also the 1,000,000 variables in the example present a serious challenge for the one dimensional search. Even a highly efficient technique such as the Golden Section search will require a considerable amount of time, and yet networks of this size are not uncommon in an urban context.

We will now develop an efficient algorithm for solving the linear programming subproblem (P) and the one diminsional searches (6). After developing the algorithm, numerical results will be presented. Omitting the constant terms in the objective function in (P), we have

$$\text{Min } \nabla f(x^1)\cdot y = \text{Min} \sum_{ijs} \frac{\partial f(x^1) y_{ij}^s}{x_{ij}^s}$$

Now let us look at $\nabla f(x)$. We see that

$$\frac{\partial f(x)}{\partial x_{ij}^s} = \frac{\partial \sum_{kl} f_{kl}(x_{kl})}{\partial x_{ij}^s} = \frac{\partial f_{ij}(x_{ij})}{\partial x_{ij}^s} = \frac{\partial f_{ij}(x_{ij})}{\partial x_{ij}} \cdot \frac{\partial x_{ij}}{\partial x_{ij}^s}$$

where $x_{ij} = \sum_{s=1}^{p} x_{ij}^s$. The second equality is true because all of the cost functions except f_{ij} are independent of x_{ij}^s. The third equality is an application of the chain rule. Clearly, $\frac{\partial x_{ij}}{\partial x_{ij}^s} = 1$. Therefore, we have that

$$\frac{\partial f(x)}{\partial x_{ij}^s} = A_{ij}\left(\sum_s x_{ij}^s\right)$$

by definition of f_{ij} and continuity of the A_{ij}. Notice that this is independent of s. Now define

$$c_{ij} = A_{ij}(x_{ij})\big|_{x=x^1} .$$

This is the average time or cost on arc (i,j)--the time experienced by each user of arc (i,j)--when the flow in the network is equal to the fixed amount x^1. The linear programming subproblem is then

$$\underset{y \geq 0}{\text{Min}} \sum_{(i,j) \in A} \sum_{s=1}^{p} c_{ij} y_{ij}^s = \underset{y \geq 0}{\text{Min}} \sum_{s=1}^{p} \sum_{(i,j) \in A} c_{ij} y_{ij}^s$$

$$\text{s.t. } (5) \qquad \text{s.t. } (5)$$

$$\geq \sum_{s=1}^{p} \underset{y \geq 0}{\text{Min}} \sum_{(i,j) \in A} c_{ij} y_{ij}^s \qquad (7)$$

$$\text{s.t. } (5)$$

The right hand side of (7) constitutes a lower bound on the optimal

value of the linear programming problem since the minimum of a sum is always greater than or equal to the sum of the minima. Note that there are no capacities on the arcs in problem (P). The constraints are only non-negativity and conservation of flow. It was not necessary to introduce capacities into the original NLP problem because the travel time for each user of any arc becomes very large when the flow exceeds the level for which the arc was designed. By definition of an equilibrium, no one will voluntarily accept a higher travel time than necessary; thus at equilibrium, flows will necessarily avoid a high state of congestion on any arc (unless every arc in the network is congested). Since the optimal solution to (1), (2), (3) is the desired equilibrium, there is therefore no reason to introduce capacities for any of the arcs; it is the non-linearity of the travel time functions which obviates the need for capacities on any arc.

Since the Frank-Wolfe algorithm is an iterative procedure, the subproblem (7) will be solved many times. At each iteration k, the vector of objective function coefficients is $\nabla f(x^k)$, where x^k is the k^{th} feasible solution. This changes at every iteration; hence we must iteratively solve problem (7). For every iteration, the linear programming subproblem (7) has the very special form

$$\underset{y \geq 0}{\text{Min}} \quad \sum_{(i,j)\in A} c_{ij}y^1_{ij} + \ldots + \sum_{(i,j)\in A} c_{ij}y^s_{ij} + \ldots + \sum_{(i,j)\in A} c_{ij}y^p_{ij}$$

$$\text{s.t.} \quad -\sum_i y^1_{ij} + \sum_k y^1_{jk} = D(j,1)$$

$$j = 2, 3, \ldots, n$$

$$-\sum_i y^s_{ij} + \sum_k y^s_{jk} = D(j,s)$$

$$j = 1, 2, \ldots, n, j \neq s$$

$$-\sum_i y^p_{ij} + \sum_k y^p_{jk} = D(j,p)$$

$$j = 1, 2, \ldots, n, j \neq p$$

where n is the number of nodes in the network. There are no linking constraints at all in this problem; the variables within any block do not interact with the variables in any other block. Therefore the problem reduces to n separate linear programming problems; by solving each "block" problem separately from the others, we get a feasible solution to the linear programming problem, and thus the lower bound (7) is achieved.

An additional important simplification occurs due to the structure of each of the above blocks. We can think of the problem in block s as minimizing total shipping cost from every node in the network to node s, where c_{ij} is the linear shipping cost of arc (i,j). It is not necessary to use the simplex method to solve the problem--this is the simplest of all transportation problems. Using the D(j,s) as

quantities to be shipped from node j to node s and the c_{ij} as the length of the arc (i,j), we can find the shortest route between nodes j and s, for all j. Since there are no capacities on any of the arcs, we send all of the quantity D(j,s) along this shortest path. Since the shortest route algorithm due to Dijkstra [Dreyfus, 1969] finds the shortest route between any given node and all other nodes, the optimal solution for block s can be found by solving one shortest route problem. Certainly the above procedure does not violate the conservation of flow requirement (5) or the non-negativity requirement. This procedure is summed up by the following theorem.

THEOREM II.1 At each iteration k of the Frank-Wolfe algorithm the optimal solution to subproblem k, problem (P), is given by $y_{ij}^{s} = \sum_{r} y_{ij}^{rs}$, where

$$y_{ij}^{rs} = \begin{cases} D(r,s) \text{ if arc } (i,j) \text{ is on the shortest path between } r \text{ and } s \text{ using the } c_{ij} = A_{ij}(x_{ij})\big|_{x=x^k} \text{ as the lengths of the arcs } (i,j). \\ 0 \text{ if arc } (i,j) \text{ is not on the shortest path} \end{cases}$$

Proof: Assume that at the optimal solution to the subproblem, all the flow does not follow the minimum path or least cost path between at least one origin-destination pair; let this pair be r and s. If we change this flow so that all of the flow does follow the shortest path, then we get a smaller cost for this origin-distination pair. Since all costs are linear in the subproblems, adding additional flow to any arc has no affect on the cost of other users. Their unit cost is c_{ij}, independent of the level of flow on the arc, so

their total cost remains the same. We then have a smaller value of the objective function, contradicting the assumed optimality.

Observe that with this approach to solving the sub-linear programming problem (7), we do not have to write out any of the tens of thousands of conservation of flow and non-negativity constraints. The constraints are implicitly satisfied--we simply observe that we have the best solution which satisfies them. Conservation of flow is satisfied by definition of a path between two nodes; obviously this procedure does not yield negative flows.

Thus we can solve the NLP problem by a sequence of shortest route problems and one dimensional searches. Since the subproblems are linear, all of the existing theory of network decomposition [Hu, 1970] can be used to solve the shortest routes in each subproblem. Numerical results are given in the next section.

An initial set of feasible flows is required to begin the algorithm. Fortunately, in a problem such as this one, it is relatively easy to get a good starting solution; we may estimate the equilibrium flows (i.e., the optimal solution to (NLP)) by dividing the flow required for each origin-destination pair among several promising routes between the origin and destination. In general, the better this initial estimate of the equilibrium point is, the fewer the number of iterations required will be. Since we are more interested in the optimal flows than the optimal value to the (NLP) problem, the best stopping rule is to stop when the maximum percentage change between the components of two consecutive flow vectors is less than some specified amount. In summary, the algorithm is as follows; we begin with the index $k = 0$.

1. Let the current feasible solution be x^k.
2. Compute $A_{ij}(x_{ij})\big|_{x=x^k} \equiv c_{ij}$. Set $y_{ij} = 0$, all (i,j). Set s = 1.
3. Find the shortest route between every node in the network and node s (i.e., solve one shortest route problem) using the c_{ij} as distances.
4. For each origin r in the network, perform the Fortran operation $y_{ij} = y_{ij} + D(r,s)$ for every arc (i,j) on the shortest route between node r and node s.
5. If s is not equal to the number of destinations, set s = s + 1 and go to step 3; otherwise go to step 6.
6. Now the y_{ij} are the solution to subproblem k. Denote by y^k the vector of the y_{ij}. Minimize the function f along the line segment between x^k and y^k, using a one-dimensional search technique. Let the minimizing point be x^{k+1}.
7. Test the stopping criterion, and go to step 2 if it fails.

As stated previously, the exceeding large number of decision variables in this problem can cause computational difficulty in step 6. The number of variables will usually number approximately one million on a 500 node network. However, only the variables $x_{ij} = \sum_{s=1}^{p} x_{ij}^s$ affect the objective function; the objective function is uniquely determined by $\sum_{s=1}^{p} x_{ij}^s$, regardless of the values of the individual x_{ij}^s. Obviously, the x_{ij} are far fewer in number than the variables x_{ij}^s. Each time we solve the subproblem (7) in y, we can find the optimal $y_{ij} = \sum_{s=1}^{p} y_{ij}^s$ without storing any of the y_{ij}^s by using counters, y_{ij}, to store the cumulative flow on each arc. We

do this in the following manner. First compute the y_{ij}^1 by sending the flow from each node to node 1 along the shortest paths. Next perform the Fortran operation $y_{ij} = y_{ij} + y_{ij}^1$ for all arcs (i,j) in the network, and then compute the y_{ij}^2. We again increment the counters, and repeat the process, computing y_{ij}^3, y_{ij}^4, ... , and then y_{ij}^p, incrementing y_{ij} each time. This procedure gives the optimal y_{ij} for (7), although the optimal y_{ij}^s will not be known. Given these y_{ij}, the one dimensional search will yield new values for the flows by solving the problem (6). After performing this one-dimensional search, we will not know the values of the x_{ij}^s, but only the values of the total flow, the x_{ij}. But we do not need the x_{ij}^s in implementing this algorithm, since the x_{ij} alone uniquely determine $\nabla f(x^{k+1})$, the new vector of arc lengths for solving (7) again. Using this procedure on the 500 node, 2,000 arc network discussed above, the one dimensional search ~~need~~ only work with 2,000 variables instead of 1,000,000. This is a significant improvement--a bisecting search of a polynomial function of 2,000 variables takes less than half a second on the CDC 6400.

It is important to realize that solving the NLP (1), (2), (3) with any straight forward algorithm would certainly require storing each individual decision variable, x_{ij}^s. These variables x_{ij}^s had to be introduced in the NLP to assure that the correct number of units flowed between each origin-destination pair. Multi-commodity conservation of flow equations require that flow be identified by its ultimate destination. Only by exploiting the structure of the NLP was explicit consideration of each decision variable obviated.

We thus have the ~~remarkable~~ conclusion that this (NLP) problem can be solved by the procedure developed above without ever writing down any of the tens of thousands of constraints and without ever storing the values of the individual decision variables.

Discussion and Comparison of the Algorithm

A common problem in any algorithm which utilizes derivatives of a function is the excessive computation time required to precisely calculate these derivatives. In this particular model, the problem is non-existent--since we are minimizing the integral of the average cost function, the derivatives are explicitly available. In fact, derivatives are easier to compute than functional values, since they contain only fourth powers, and the objective function contains fifth powers. Accuracy is limited only by the accuracy of the actual parameters, a_{ij} and b_{ij}, input to the model.

Another difficulty inherent to feasible direction algorithms such as the Frank-Wolfe occurs when the objective function assumes a minimum point in the interior of the feasible region. If x^* is an interior minimum point, then $\nabla f(x^*) = 0$. In a small neighborhood of x^*, the individual components of the gradient, because they are near zero, may oscillate positive and negative. Thus the geometric direction indicated by the gradient may change very rapidly as the sequence of points nears the optimal solution. This can cause the points to zig zag around the optimal solution, retarding the speed of convergence. Fortunately, the gradient of the objective function (1) will not be zero at the optimal solution, since the function is strictly increasing and has no stationary point in the feasible region. Thus the NLP (1), (2), (3) appears to be well suited to a feasible directions approach.

In very large linear programming problems, there is frequently a certain amount of round-off error introduced into the optimal solution

given by the simplex method. This type of error is primarily caused by multiplication and division operations. The error ~~should be almost~~ is non-existent in the solution given by Theorem II.1, since shortest route algorithms typically require only addition and comparison operations.

The most time consuming portion of the algorithm described above is in finding all shortest paths in the network, i.e., in solving subproblem (P). We solve subproblem (P) to determine a direction in which to minimize the objective function (1). It is intriguing to note that the computational requirements of (P) are identical to one iteration of one of the most common simulation traffic assignment procedures used today. This is the iterated capacity restraint algorithm [Comsis, 1962], [Hershdorfer, 1966]. The rationale behind this simulation procedure is taken from a game theoretic interpretation of the network equilibrium problem. Associated with each origin in the network is a player who tries to choose a set of routes such that the correct number of vehicles will travel from his origin to each destination at minimum travel time for the vehicles associated with his origin. Since 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 in the network. Each player chooses his routes in turn; after all the players have made their decisions, the resulting travel times are revealed to all the players, and they again take turns in revising their routes [Hershdorfer, 1966].

The computational requirements of this rather colorful interpretation of the network equilibrium problem are as follows. At each iteration of the procedure,

the length of each arc is computed as the average travel time function evaluated at the current flow level on the arc. Next, shortest routes between each origin and destination are determined so that each player can direct his vehicles along the shortest path to their destination. The algorithm developed in this paper also iteratively computes the length of each arc as the arc's average travel time function evaluated at the current flow on that arc and finds all shortest paths. The fundamental difference between the two procedures is that our proposed algorithm minimizes the objective function along this generated direction to compute a new vector of flows. The simulation procedure in [Hershdorfer, 1966] on the other hand, uses this n-dimensional direction itself as a new vector of flows. This leads to completely distinct flow vectors. These two approaches to the network equilibrium 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 proven to be the equilibrium. Nevertheless, the only difference in computational requirements is in the one dimensional search, a negligible procedure.

Numerical Results

The algorithm developed above was programmed in Fortran on a CDC 6400 computer; a small test network of ten arcs and four nodes was used for debugging purposes. Each of the four nodes was considered to be an origin and a destination. The network, trip table, and travel time functions were chosen so that the equilibrium flows could be claculated by hand. The (NLP) problem for this network consisted of 40 variables x_{ij}^{s}, $s = 1,2,3,4$ for each of the ten arcs; $4 \cdot 3 = 12$ conservation of flow constraints; and 40 non-negativity constraints. Approximately ten iterations of the procedure were required for a reasonable degree of

of accuracy - about 5% of the flow variables. Computing time (central processing unit) was approximately one second.

The algorithm was then run on the 76 arc, 24 node network in Figure 2. The trip table and parameters for the average travel time functions are given in Table 1. The (NLP) problem for this network had 1824 variables: x_{ij}^s, s = 1,2,..., 24, for each of the 76 arcs, (i,j); (24)·(23) = 552 conservation of flow constraints; and 1824 non-negativity constraints. Since this is a general nonlinear programming problem, it is impossible to determine the exact solution in a finite amount of time. However, Table 2 shows the sequence of flow vectors generated: there are 76 components, one for each arc. After 20 iterations, only 2 variables changed by more than 5%; the majority changed by less than 2%. Thus the vector x^{20} appears to be a highly accurate estimate of the equilibrium solution. Computing time for twenty iterations, excluding 3 seconds of compilation time, was nine seconds. If the termination rule had been to stop when the maximum percentage change in components was less than 8%, the procedure would have terminated after 16 iterations. After sixteen iterations, the maximum percentage change in the components was 7.7%; computing time was seven seconds.

Each subproblem for this network decomposed into 24 shortest route problems. Since every node is both an origin and a destination, all shortest routes in the network are required, and the multi-terminal shortest route algorithm of Floyd [Dreyfus, 1969] was used.

A very encouraging result was the extremely small computing time for solving (NLP). This same problem for the 76 arc, 24 node network was also solved by linearizing the objective function (1) and using

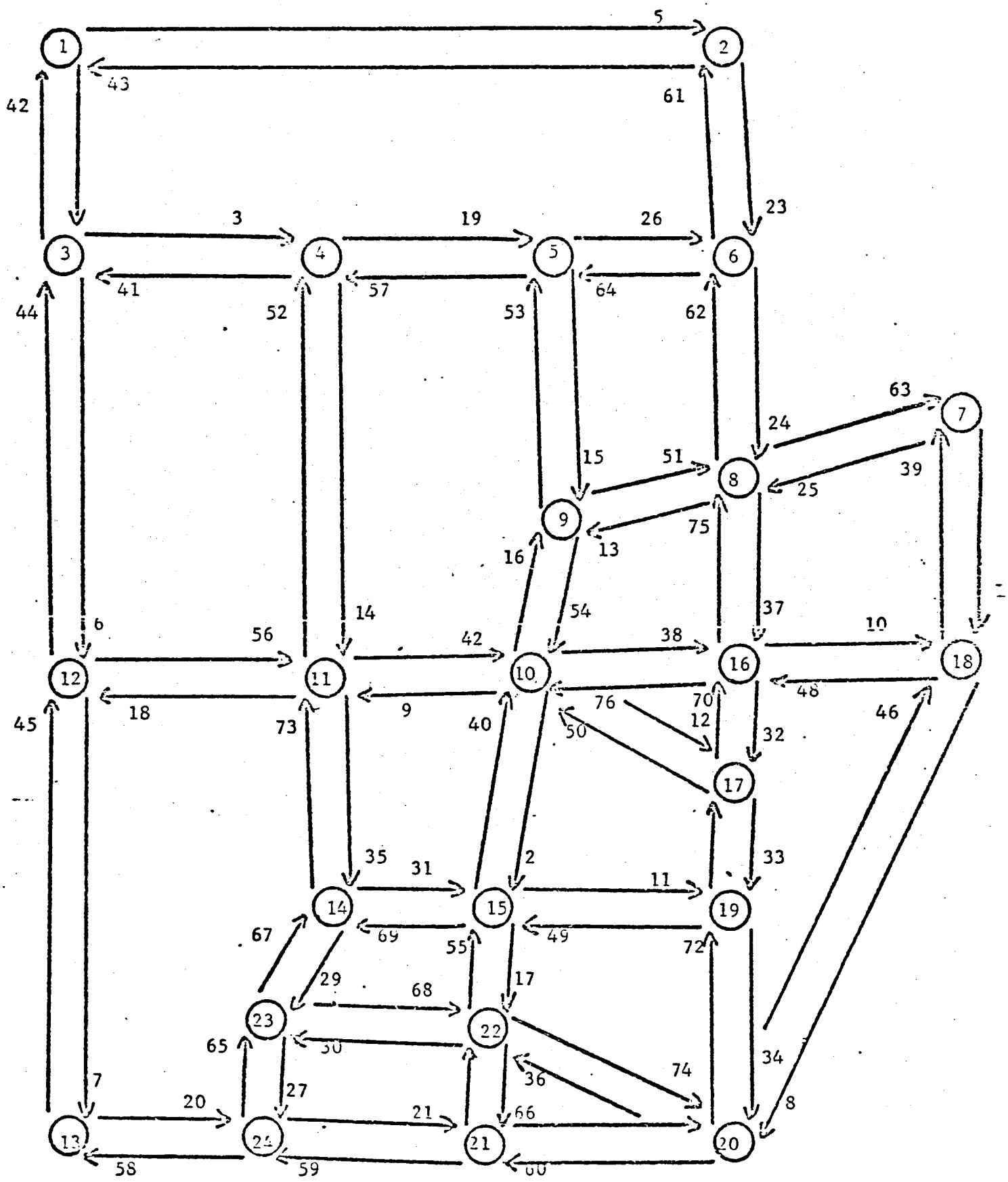


Figure 2--Test Network for the Equilibrium Problem

	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	0.0	.1	.2	.6	.2	.4	.5	.8	.6	1.4	.6	.3	.6	.3	.5	.6	.5	.2	.3	.3	.1	.4	.3	.1
2	.1	0.0	.1	.3	.1	.5	.2	.5	.3	.6	.2	.2	.3	.1	.2	.4	.3	.1	.1	.2	.1	.2	.1	.1
3	.2	.1	0.0	.3	.1	.3	.1	.2	.2	.3	.3	.3	.2	.1	.1	.2	.1	.0	.1	.1	.1	.1	.1	.1
4	.6	.3	.3	0.0	.5	.5	.5	.7	.8	1.2	1.5	.7	.6	.5	.5	.8	.5	.1	.3	.4	.2	.4	.5	.3
5	.3	.1	.1	.5	0.0	.3	.2	.6	.8	1.0	.6	.2	.2	.2	.3	.6	.3	.1	.2	.2	.1	.2	.2	.1
6	.4	.5	.3	.5	.3	0.0	.4	.8	.4	.8	.4	.3	.3	.2	.3	1.0	.6	.1	.3	.4	.1	.3	.2	.1
7	.5	.2	.1	.5	.2	.4	0.0	1.0	.6	1.9	.5	.8	.5	.3	.5	1.4	1.0	.2	.5	.6	.3	.6	.2	.1
8	.8	.5	.2	.7	.3	.8	1.1	0.0	.9	1.6	.9	.6	.6	.4	.7	2.2	1.4	.3	.7	.9	.4	.6	.4	.2
9	.6	.3	.2	.8	.8	.4	.6	.9	0.0	2.8	1.5	.7	.6	.6	1.0	1.5	1.0	.2	.5	.7	.4	.7	.6	.2
10	1.4	.6	.3	1.2	1.0	.8	1.9	1.6	2.8	0.0	4.0	2.1	1.9	2.1	4.0	4.4	3.9	.7	1.8	2.6	1.3	2.7	1.8	.9
11	.6	.2	.3	1.5	.6	.4	.5	.9	1.5	4.0	0.0	1.4	1.0	1.6	1.5	1.4	1.0	.2	.5	.7	.5	1.1	1.4	.6
12	.3	.2	.3	.7	.2	.3	.8	.6	.7	2.1	1.5	0.0	1.4	.7	.8	.7	.7	.2	.3	.5	.4	.8	.7	.5
13	.6	.3	.2	.6	.2	.3	.5	.6	.6	1.9	1.0	1.4	0.0	.6	.7	.7	.6	.1	.4	.7	.6	1.3	.8	.8
14	.3	.1	.1	.5	.2	.2	.3	.4	.6	2.2	1.6	.7	.6	0.0	1.3	.7	.7	.1	.4	.5	.4	1.2	1.1	.4
15	.5	.2	.1	.5	.3	.3	.5	.7	1.0	4.0	1.5	.8	.7	1.3	0.0	1.3	1.6	.3	.8	1.1	.8	2.6	1.0	.5
16	.6	.4	.2	.8	.6	1.0	1.4	2.2	1.5	4.4	1.4	.7	.7	.7	1.3	0.0	2.8	.5	1.4	1.7	.6	1.2	.6	.3
17	.5	.3	.1	.5	.3	.6	1.0	1.4	1.0	3.9	1.0	.7	.6	.7	1.5	2.8	0.0	.7	1.7	1.7	.7	1.7	.6	.3
18	.2	.1	.0	.1	.1	.1	.2	.3	.2	.7	.2	.2	.1	.1	.3	.5	.7	0.0	.4	.5	.1	.4	.1	.1
19	.3	.1	.1	.3	.1	.3	.5	.7	.5	1.8	.5	.3	.4	.4	.8	1.4	1.7	.4	0.0	1.3	.5	1.3	.4	.2
20	.3	.2	.1	.4	.2	.4	.6	.9	.7	2.6	.7	.5	.7	.5	1.1	1.7	1.7	.5	1.3	0.0	1.3	2.5	.7	.5
21	.1	.1	.1	.2	.1	.1	.3	.4	.4	1.3	.5	.4	.6	.4	.8	.6	.7	.1	.5	1.3	0.0	1.9	.7	.6
22	.4	.2	.1	.4	.2	.3	.6	.6	.7	2.7	1.1	.8	1.3	1.2	2.6	1.2	1.7	.4	1.3	2.5	1.9	0.0	2.2	1.2
23	.3	.1	.1	.5	.2	.2	.2	.4	.6	1.8	1.4	.7	.8	1.1	1.0	.6	.6	.1	.4	.7	.7	2.2	0.0	.8
24	.2	.1	.1	.3	.1	.1	.1	.2	.2	.9	.6	.5	.8	.4	.5	.3	.3	.1	.2	.5	.6	1.2	.8	0.0

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

Table 1-- Trip Table and Arc Parameters for Network in Figure 3

NORM OF X SUB 1 MINUS X SUB 0 EQUALS 49.18 MAXIMUM PERCENTAGE CHANGE OF COMPONENTS IS 67.8 PER CENT

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
13.1 8.8 7.1 5.4 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
17.2 21.5 15.9 9.4 2.9 14.5 14.2 9.4 17.4 14.6 16.1 14.9 16.9 12.7 19.4 26.4 25.2 12.8 21.0
13.1 8.8 7.1 5.4 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

NORM OF X SUB 2 MINUS X SUB 1 EQUALS 30.42 MAXIMUM PERCENTAGE CHANGE OF COMPONENTS IS 46.6 PER CENT

21.3 22.2 18.9 6.8 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
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
21.4 22.2 18.9 6.8 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
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

NORM OF X SUB 3 MINUS X SUB 2 EQUALS 15.87 MAXIMUM PERCENTAGE CHANGE OF COMPONENTS IS 39.4 PER CENT

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
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
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
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

NORM OF X SUB 4 MINUS X SUB 3 EQUALS 21.05 MAXIMUM PERCENTAGE CHANGE OF COMPONENTS IS 50.0 PER CENT

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
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
19.9 25.7 20.0 12.0 7.1 19.8 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
12.2 11.5 11.5 8.0 15.5 16.0 9.5 11.5 12.6 10.5 8.9 12.0 14.4 13.5 14.3 13.3 8.8 9.9 11.1

NORM OF X SUB 5 MINUS X SUB 4 EQUALS 13.71 MAXIMUM PERCENTAGE CHANGE OF COMPONENTS IS 32.1 PER CENT

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
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
20.5 25.8 18.1 12.2 6.9 19.7 17.3 22.9 20.7 22.6 19.7 10.5 8.3 9.0 19.2 26.2 22.0 10.6 21.0
12.6 9.9 9.5 7.3 13.7 16.0 8.3 11.8 12.1 9.7 10.9 10.8 13.9 11.8 11.7 11.6 11.7 8.2 13.0

NORM OF X SUB 6 MINUS X SUB 5 EQUALS 21.28 MAXIMUM PERCENTAGE CHANGE OF COMPONENTS IS 100.0 PER CENT

16.4 24.4 23.4 10.1 6.0 22.0 17.5 21.7 17.0 18.7 19.2 9.3 8.5 8.6 20.4 28.7 18.4 9.3 24.6
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
16.4 26.4 23.4 10.1 6.0 22.1 17.5 21.7 17.0 18.7 19.2 9.3 8.6 8.6 20.5 28.7 18.4 9.3 24.6
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

NORM OF X SUB 7 MINUS X SUB 6 EQUALS 12.44 MAXIMUM PERCENTAGE CHANGE OF COMPONENTS IS 41.1 PER CENT

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 8.5 18.5 27.1 18.4 9.0 21.8
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
17.2 25.9 20.7 10.9 6.4 21.2 16.9 22.1 20.5 19.9 18.8 9.3 8.6 8.6 18.5 27.1 18.4 9.0 21.3
11.4 10.7 8.1 7.1 14.1 15.1 9.9 10.2 10.6 10.7 9.6 9.9 13.2 11.3 8.7 9.9 9.0 9.1 12.9

Table 2--Sequence for Flow Vectors for Figure 3

NORM OF X SUB 8 MINUS X SUB 7 EQUALS 9.04								MAXIMUM PERCENTAGE CHANGE OF COMPONENTS IS 21.6 PER CENT										
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
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
17.4	25.4	19.3	11.3	6.9	19.1	17.2	23.3	19.5	18.8	20.5	9.2	9.0	7.9	18.5	25.9	19.4	9.4	20.6
12.3	10.1	8.2	7.3	13.4	14.5	8.5	10.5	9.9	9.1	11.0	11.2	11.9	10.8	11.1	11.1	8.7	9.2	11.2
NORM OF X SUB 9 MINUS X SUB 8 EQUALS 9.68								MAXIMUM PERCENTAGE CHANGE OF COMPONENTS IS 35.4 PER CENT										
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
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
16.7	25.8	18.3	10.3	6.3	17.1	16.7	22.8	19.3	19.1	18.8	8.9	7.2	7.3	17.2	25.1	20.8	9.6	20.8
12.3	11.1	7.9	7.1	14.7	13.7	10.2	8.9	10.3	8.4	10.3	8.8	12.9	11.1	8.2	9.7	9.6	9.5	12.3
NORM OF X SUB 10 MINUS X SUB 9 EQUALS 6.10								MAXIMUM PERCENTAGE CHANGE OF COMPONENTS IS 16.3 PER CENT										
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
12.1	10.6	7.8	6.7	13.6	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	11.9
17.6	24.9	19.0	10.7	6.1	17.3	16.7	22.6	19.4	19.5	19.3	8.9	8.6	7.4	18.4	25.3	20.1	8.9	21.4
12.1	10.6	7.8	6.6	13.8	15.1	9.5	9.4	9.9	8.8	10.4	10.5	12.3	10.9	9.2	11.2	9.4	8.6	11.6
NORM OF X SUB 11 MINUS X SUB 10 EQUALS 9.42								MAXIMUM PERCENTAGE CHANGE OF COMPONENTS IS 25.0 PER CENT										
17.5	25.9	17.2	11.9	7.6	15.1	16.1	23.3	20.6	19.1	19.7	8.8	7.3	6.8	17.4	24.8	19.6	9.6	20.0
11.9	11.1	8.1	7.3	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
17.6	25.7	16.8	11.9	7.6	15.5	16.7	23.2	19.8	19.0	19.7	8.8	7.3	6.7	17.2	24.6	18.0	9.7	19.8
12.4	11.5	7.9	7.9	14.6	14.4	8.7	8.4	9.7	9.6	8.7	8.4	11.3	9.4	11.0	9.3	8.8	8.7	11.3
NORM OF X SUB 12 MINUS X SUB 11 EQUALS 5.42								MAXIMUM PERCENTAGE CHANGE OF COMPONENTS IS 16.0 PER CENT										
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
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
17.4	25.8	18.2	12.0	7.2	15.8	16.4	22.8	19.4	19.0	19.6	8.7	8.2	7.5	18.3	24.5	18.9	8.9	20.7
12.1	10.8	7.5	7.4	13.8	14.2	8.3	9.0	9.5	9.2	10.2	10.0	11.6	9.7	10.2	10.4	9.2	9.0	11.3
NORM OF X SUB 13 MINUS X SUB 12 EQUALS 4.62								MAXIMUM PERCENTAGE CHANGE OF COMPONENTS IS 13.9 PER CENT										
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
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
17.1	24.7	17.8	11.4	6.8	15.2	16.2	22.5	19.2	18.7	19.4	8.7	7.7	7.1	17.7	24.3	19.2	9.1	20.7
12.2	9.9	7.7	7.2	14.3	14.0	9.2	8.2	8.7	8.8	10.0	9.6	12.2	10.2	9.7	11.0	8.6	9.2	11.9
NORM OF X SUB 14 MINUS X SUB 13 EQUALS 3.93								MAXIMUM PERCENTAGE CHANGE OF COMPONENTS IS 9.6 PER CENT										
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
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
17.9	25.1	18.3	11.3	6.8	15.5	16.2	22.3	18.8	19.0	19.7	8.7	8.3	7.0	18.6	24.6	19.4	8.7	21.2
12.2	10.6	7.6	7.2	13.8	14.9	8.7	8.9	9.3	8.5	10.5	9.9	12.1	10.2	9.7	10.7	8.5	8.5	11.5

Table 2 Continued

NORM OF X SUB 15 MINUS X SUB 14 EQUALS							4.48	MAXIMUM PERCENTAGE CHANGE OF COMPONENTS IS							11.4 PER CENT		
17.5	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
11.9	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
17.5	25.4	17.8	10.9	6.5	14.9	16.2	22.5	20.2	18.8	19.6	8.7	8.0	6.9	18.1	24.5	18.7	9.0	21.0
12.0	11.0	7.5	7.0	14.1	14.4	9.1	8.7	9.7	9.0	9.6	10.3	12.1	10.2	9.0	10.2	8.4	8.4	12.0
NORM OF X SUB 16 MINUS X SUB 15 EQUALS							3.21	MAXIMUM PERCENTAGE CHANGE OF COMPONENTS IS							7.7 PER CENT		
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
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
17.4	25.4	18.3	10.9	6.6	15.1	16.0	22.4	19.3	18.5	19.8	8.7	8.5	7.1	19.0	24.5	18.9	8.6	21.4
12.0	10.5	7.4	7.0	13.6	14.3	8.6	9.0	9.4	8.9	10.4	9.8	11.9	10.1	9.5	10.4	8.3	8.6	11.7
NORM OF X SUB 17 MINUS X SUB 16 EQUALS							4.15	MAXIMUM PERCENTAGE CHANGE OF COMPONENTS IS							11.2 PER CENT		
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
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
17.5	25.2	18.1	11.2	6.3	14.4	15.8	22.7	19.6	18.5	19.6	8.7	7.9	8.0	17.4	23.0	19.3	8.9	20.5
11.9	10.9	7.5	6.7	13.9	14.1	9.2	8.5	9.5	8.4	10.5	10.3	12.0	10.0	9.2	10.4	8.3	8.8	11.8
NORM OF X SUB 18 MINUS X SUB 17 EQUALS							2.73	MAXIMUM PERCENTAGE CHANGE OF COMPONENTS IS							7.9 PER CENT		
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
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
17.5	25.2	18.3	11.0	6.3	14.6	15.9	22.5	18.9	18.2	19.7	8.7	8.5	7.8	18.1	23.5	18.8	8.7	20.8
12.0	10.5	7.4	6.8	13.6	14.2	8.8	8.3	9.3	8.9	10.1	10.1	12.0	10.0	9.2	10.4	8.2	8.9	11.7
NORM OF X SUB 19 MINUS X SUB 18 EQUALS							3.00	MAXIMUM PERCENTAGE CHANGE OF COMPONENTS IS							8.2 PER CENT		
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
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
17.5	25.0	18.1	10.7	6.2	14.3	16.1	22.6	19.6	18.2	19.5	8.7	8.0	7.4	17.7	23.5	19.2	9.1	20.8
12.1	11.1	7.4	6.8	14.0	14.2	9.3	8.8	9.8	8.3	11.0	9.9	12.1	10.0	8.9	10.3	8.1	8.9	12.0
NORM OF X SUB 20 MINUS X SUB 19 EQUALS							2.38	MAXIMUM PERCENTAGE CHANGE OF COMPONENTS IS							6.9 PER CENT		
17.4	25.9	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
11.9	10.5	7.5	6.5	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	11.9
17.5	25.1	18.4	10.7	6.2	14.4	16.0	22.6	19.0	18.0	19.6	8.7	8.3	7.3	18.4	23.8	18.9	8.9	21.2
12.0	10.6	7.6	6.7	13.7	14.1	9.0	8.5	9.5	8.5	10.5	9.9	11.8	10.0	9.4	10.4	8.0	9.0	11.8

Table 2 Continued

the OPTIMA linear programming package for the CDC 6400. Computing time was significantly higher--11 minutes and 40 seconds. Equally encouraging was the very small increase in the number of iterations required by the Frank-Wolfe algorithm for the two example problems above. In Table 3 below, we see that there were 12 conservation of flow constraints and 40 non-negativity constraints for the first network. The larger network resulted in 552 conservation of flow equations and 1,824 variables and non-negativity constraints. Nevertheless, the required number of iterations increased from 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 the number of constraints in the NLP problem. Increasing the number of nodes by a factor of 6 doubled the number of iterations; this indicates that the algorithm will be efficient for problems as large as several hundred nodes.

	Nodes	Arcs	Conservation of Flow Constraints	Variables (Non-negativity constraints	Iterations for 5% accuracy	Computing Time (Seconds)
Problem 1	4	10	12	40	10	1
2	24	76	552	1824	20	9

Table 3

Conclusion

The primary significance of the solution technique developed in this paper is that it requires only the solution of a sequence of shortest route problems (computing time for the one dimensional searches was insignificant). The computing time for finding an approximate solution was less than that required by the simplex method by orders of magnitude

even on a fairly small network. For larger problems the savings would be even greater, since for multi-commodity network problems the number of constraints grows as the square of the number of nodes in a network. At no time are any of the conservation of flow and non-negativity constraints used explicitly in the technique of this paper. It is well known that problems on networks with several hundred nodes can be efficiently solved. Also, preliminary computational results indicate that the number of shortest route subproblems (i.e., the number of iterations) for a network equilibrium problem with several hundred nodes will not be excessive. Thus the solution approach appears very promising for large network equilibrium problems.

REFERENCES

1. Beckmann, M.; McGuire, C.B. and Winsten, C.; Studies in the Economics of Transportation, New Haven, Connecticut: Yale University Press, 1956.

2. Comsis Corporation, Traffic Assignment: Methods-Applications-Products. Prepared for Urban Planning Division, Office of Planning, Federal Highway Administration, Preliminary--Subject to Review, 1972.

3. Dreyfus, S.E., "An Appraisal of Some Shortest Path Algorithms", Operations Research, 394-412, 1969.

4. Hershdorfer, A.M., "Predicting the Equilibrium of Supply and Demand: Location Theory and Transportation Network Flow Models", TRF --Papers, 131-143, 1966.

5. LeBlanc, L.J., Mathematical Programming Algorithms for Large Scale Network Equilibrium and Network Design Problems, Ph.D. Dissertation, Department of Industrial Engineering and Management Sciences, Northwestern Univ., 1973.

6. Luenberger, D.G., Introduction to Linear and Nonlinear Programming, Addison-Wesley, Reading, Mass., 1973.

7. Michaels, R.M., "Driver Tension Responses Generated on Urban Streets", Public Roads, 53-58, 71, 1960.

8. Michaels, R.M., "The Effects of Expressway Design on Driver Tension Responses", Public Roads, 107-112, 1962.

9. Michaels, R.M., "Attitudes of Driver Determine Choice Between Alternate Highways", Public Roads, 225-236, 1965.

10. Wardrop, J.G., "Some Theoretical Aspects of Road Traffic Research", Proc. Inst. Civ. Eng. 1, Part II, 325-378, 1952.

11. Zangwill, W.I., Nonlinear Programming: A Unified Approach, Prentice-Hall, Inc., 1969.