SOLVING LARGE TELECOMMUNICATIONS NETWORK LOADING PROBLEMS

Dah Nain Lee, Karen T. Medhi, John L. Strand, Roger G. Cox, and Stephen Chen

Telecommunications network loading problems focus on routing traffic through networks so as to satisfy the demand forecast between all pairs of nodes. The objective is to minimize the total routing costs subject to satisfying capacity constraints associated with the individual links. This paper presents a nonlinear programming formulation for the multiperiod network loading problem, a problem that appears in many guises throughout network planning. Such problems are also known in the operations research literature as (multiperiod) multicommodity network flow problems. The resulting optimization problem has a very large number of variables and constraints. We describe the implementation of a nonlinear optimization algorithm, the Frank-Wolfe method with PARTAN extension and projective gradient acceleration, on a parallel vector processor for solving these large dynamic network loading problems. Our computational experience shows that our algorithm solves large, realistic loading problems very efficiently and is serving as a valuable tool for telecommunications network planning.

Dah Nain Lee, ***Karen T. Medhi***, ***John L. Strand***, ***Roger G. Cox***, *and* ***Stephen Chen*** *are with AT&T Bell Laboratories at Holmdel, New Jersey. Mr. Lee is a member of technical staff in the Integrated Network Planning Department. He develops models and solution methods for designing the AT&T transmission facility network. He received a B.S. in structure engineering from National Cheng Kung University, an M.S. in mathematics from Kansas State University, and a Ph.D. in operations research and numerical methods from Carnegie Mellon University. He joined AT&T in 1982. Ms. Medhi is a member of technical staff in the Advanced Decisions Support Systems Department. She worked on network modeling and algorithm development while a member of the Integrated Network Planning Department. Currently, she works on* (continued on page 56)

Introduction

Telecommunications network loading problems concentrate upon routing traffic through the network in order to satisfy expected demands between all pairs of nodes. If the capacities of the links connecting the nodes are unlimited, one can simply route the demand for any given pair of nodes on the cheapest path connecting this pair. However, since capacity constraints are given for each link, the loading problem is a complex optimization problem. Such problems are also known as multicommodity network flow problems.[1]

In this paper we focus on the multiperiod loading problem, in which the demands and capacities are changed over time. That is, the loading process is dynamic. Such problems are very large in size and have numerous applications in telecommunications network planning. See, for example, References 2 and 3. We present a nonlinear programming formulation and describe the implementation of a nonlinear optimization algorithm, the Frank-Wolfe method with PARTAN extension and projective gradient acceleration, on a parallel vector processor for solving this problem. Our computational results show that our algorithm solves large, realistic loading problems efficiently and effectively. It is serving as a valuable tool in performing realistic analyses for large-scale telecommunications network planning.

We first introduce some terminology. In a telecommunications network context, a *node* corresponds to an office or (in the case of aggregated networks) a set of offices in a specified region. Each *link* represents a transmission route joining two nodes, e.g., a fiber-optic cable. Often there will be multiple routes between two nodes to represent multiple distinct technologies on a single route. In these cases there will be multiple links between the same node pair. Each link has a (time-varying) capacity. A link joining a node n_a and another node n_z is said to be *incident* to n_a and n_z. Two links are said to be *adjacent* if they are both incident to the same node. A *path* from a node n_a to another node n_z can be defined as a sequence of distinct links $(m_1, m_2, \ldots, m_s)$, where m_i and m_{i+1} are *adjacent* for $i = 1, 2, \ldots, s-1$. A *demand* is the expected change in the required number of circuits of a certain type between specified nodes at a specified time. The cost of loading a unit of demand between a pair of nodes consists of multiplex costs at the nodes and the cost of the necessary transmission media on each link traversed by the path. The network loading problem is to assign demands to paths so that link capacity constraints are satisfied if possible and so that the total network costs are minimized.

In the next section we present the mathematical formulation for the network loading problem. Then, in "Solution Technique," we describe the solution method and the implementation technique for solving the problem. The section "Performance" shows some computational results. We close with some concluding remarks.

The Loader Formulation

The Multicommodity Network Model. In this section, we describe our multicommodity flow formulation. Let n_a and n_z be two distinct nodes in a telecommunications network. A demand for circuits between n_a and n_z can be viewed as a demand for a flow of a commodity along a path connecting n_a and n_z in the network. The total flow between n_a and n_z is simply the sum of flows on all paths connecting n_a and n_z. The flow on all paths that use a given link cannot exceed the link capacity. This is the arc-path formulation of flow.[4–6] The formulation can be generalized to treat more than one commodity (i.e., more than one demand pair) and is then termed multicommodity flow. A node-arc formulation uses, in addition to link capacity constraints, flow conservation equations for each commodity.[4,5] In general, an arc-path formulation has fewer constraints than the node-arc formulation at the expense of having an enormously large number of variables (possible paths.) However, in telecommunications applications only a small subset of the paths needs to be considered as viable routes—for example, only paths with, say, at most 10 links, or only paths that pass a certain quality criterion.

We first consider a formulation for one time period. We assume that, for a time period, there are K demand pairs in the network and that there are a total of P paths connecting the end points of the demand pairs. We also assume that these paths use a total of M links. Let

P_k = set of paths available to demand pair k
Q_m = set of paths passing through link m
u_m = capacity of link m
d_k = total amount of demand for demand pair k

Let x_p be the amount of demand loaded on path p; also let α_p denote the cost of routing one demand unit on path p. We now state the following program:

$$\text{Minimize} \quad \sum_{p=1}^{P} \alpha_p x_p \tag{1a}$$

$$\text{subject to} \quad \sum_{p \in P_k} x_p = d_k \qquad k = 1, \ldots, K \tag{1b}$$

$$\sum_{p \in Q_m} x_p \le u_m \qquad m = 1, \ldots, M \tag{1c}$$

$$x_p \ge 0 \qquad p = 1, \ldots, P \tag{1d}$$

The objective function (1a) represents the total path cost. System (1b) requires that the path flows sum to the required values for each demand pair k, and system (1c) requires that the load on each link does not exceed the link capacity.

This model is easily extended to handle multiple time periods. We let T be the number of time periods in the planning horizon. Let

- x_p^t = incremental flow on path p in time period t; $x_p^t \ge 0$
- d_k^t = incremental demand requirements for demand pair k in time period t; $d_k^t \ge 0$
- Q_m^t = set of paths through link m in time period t
- P_k^t = set of paths available to demand k in time period t
- u_m^t = capacity of link m in time period t; u_m^t is nondecreasing with t

We seek loads and path flows so that node-to-node demand requirements are met for each time period t and demand pair k, while total discounted routing costs are minimized. Therefore we have

$$\text{Minimize} \sum_{t=1}^{t=T} \sum_{p=1}^{p=P} \omega_t \alpha_p x_p^t \tag{2a}$$

$$\text{subject to} \sum_{p \in P_k^t} x_p^t = d_k^t \qquad k = 1, \ldots, K; t = 1, \ldots, T \tag{2b}$$

$$\sum_{s=1}^{t} \sum_{p \in Q_m^s} x_p^s \le u_m^t \qquad m = 1, \ldots, M; t = 1, \ldots, T \tag{2c}$$

$$x_p^t \ge 0 \qquad p = 1, \ldots, P; t = 1, \ldots, T \tag{2d}$$

where ω_t is a discount factor for each time period t.

We note that this is a simple extension of the one-time-period model. System (2b) is the same as system (1b). System (2c) reflects a cumulative load on a link. In other words, circuits loaded in time period s on link m remain on link m through the entire planning horizon (time periods $s+1, s+2, \ldots, T$).

A basic assumption of our model is that there is no disconnection of loads; in other words $d_k^t \ge 0$ and $x_k^t \ge 0$ for every k and every t. Note that incremental demand $d_k^t < 0$ implies that previously routed demands must be altered and that we must allow $x_p^t < 0$. To ensure that the decrease occurs only in paths with sufficient demand on them, we must replace the nonnegativity constraints (2d) on x_p^t by

$$\sum_{s=1}^{t} x_p^s \ge 0 \qquad p = 1, \ldots, P; t = 1, \ldots, T \tag{2e}$$

This guarantees feasibility, but it opens up the possibility of demand rearrangements. (Any feasible combination of x_p's, $p \in P_k$, summing to zero, could be added to the solution to reflect a set of rearrangements of demand k.) Although rearrangements are an option to consider in network optimization, this formulation does not model their costs. In the remaining sections of this paper we assume no rearrangements are allowed and we are modeling a growing network according to function (2a) and system (2b) with $d_k^t \ge 0$.

Linear Programming Model. We now state *generically* the following linear program in a compact matrix form for network loading function:

$$\begin{aligned} \text{Minimize} \quad & \mathbf{cx} \\ \text{subject to} \quad & A\mathbf{x} = \mathbf{d} \\ & B\mathbf{x} \le \mathbf{u} \\ & \mathbf{x} \ge \mathbf{0} \end{aligned}$$

Here $\mathbf{x}$ is a vector of path loads over all time periods. The matrix A is a demand-path incidence matrix over all demands over all time periods [i.e., system (2b)]. The vector $\mathbf{d}$ is a vector of point-to-point demands. The matrix B is a link-path incidence matrix over all time periods [system (2c)]. The vector $\mathbf{u}$ is the link capacities for all links over all time periods.

The loading model above may be infeasible for a capacitated facility planning problem. That is, the current capacity available may not accommodate all the expected demand. In this case, we would like to load as much of the demand as possible in a least-cost way. Let y_k^t be the unsatisfied demand increment for demand pair k at time period t, and let y be the corresponding vector of y_k^t's. We assign a high cost to the y_k^t variables so that the optimal solution will load as many demands as possible on real paths. The augmented model is as follows:

$$\begin{aligned} \text{Minimize} \quad & \mathbf{cx} + \mathbf{gy} \\ \text{subject to} \quad & A\mathbf{x} + \mathbf{y} = \mathbf{d} \\ & B\mathbf{x} \le \mathbf{u} \\ & \mathbf{x}, \mathbf{y} \ge \mathbf{0} \end{aligned} \qquad (3)$$

where $\mathbf{g}$ is the cost vector associated with $\mathbf{y}$. We assume that the cost parameters g_k^t are sufficiently high.

We emphasize here again that only a subset of paths is considered for our model. Indeed, because of the great size of the problem, the approach would be impractical otherwise without resorting to column generation techniques.

Solution Technique

Large network loading problems normally are solved by using sophisticated heuristic methods. Usually, planners modify the results manually to achieve an acceptable "suboptimal" solution. Other approaches could include a linear programming package, using the simplex method or a Karmarkar-based algorithm. Although these algorithms could perhaps be specialized to the structure of equation (3),[7] given the sizes of realistic loading problems, the computation time is expected to be too large for real-time "what if" analysis.

Nonlinear Penalty Formulation. We develop an alternative approach. The underlying algorithm is the Frank-Wolfe algorithm. The algorithm minimizes a convex nonlinear objective function subject to linear constraints by a sequence of linear programming approximations.[8]

We note that the demand-path matrix A is a nicely structured matrix which is completely decoupled by a demand pair. Physically, this means that a path can go between one and only one set of terminal pairs. If capacity was not an issue, global optimization would be equivalent to finding the best route for each terminal pair independent of other terminal pairs; that is, the problem would be separable.

The inequalities $B\mathbf{x} \le \mathbf{u}$ couple the problem together. Physically, this means that paths between different terminal pairs may share certain links. The idea of the nonlinear model is to relax these coupling constraints and take advantage of the nice structure of the demand constraints. To do this we penalize the objective function for violating the link capacity constraints. Define

$$h_i(x) = \begin{cases} 0 & \text{if } \mathbf{B}_i\mathbf{x} - u_i \le 0 \\ G(\mathbf{B}_i\mathbf{x} - u_i) & \text{if } \mathbf{B}_i\mathbf{x} - u_i > 0 \end{cases}$$

where $\mathbf{B}_i$ is the ith row of the matrix B [see equation (3)] and G exceeds all the elements of the vector $\mathbf{g}$.

Let

$$h(x) = \sum_{i=1}^{i=M\times T} h_i(x)$$

Thus we have an intuitively appealing "pseudo"

penalty function. We then solve the following convex program:

$$\begin{aligned} \text{Minimize} \quad & \mathbf{cx} + \mathbf{gy} + h(\mathbf{x}) \\ \text{subject to} \quad & A\mathbf{x} + \mathbf{y} = \mathbf{d} \\ & \mathbf{x}, \mathbf{y} \geq \mathbf{0} \end{aligned} \tag{4}$$

If G exceeds all the elements of $\mathbf{g}$, then the optimal solution of equation (4) will satisfy the capacity constraints $B\mathbf{x} \leq \mathbf{u}$.

Let us first introduce the essential components of the Frank-Wolfe algorithm. We will then explain some adaptations to our problem. The Frank-Wolfe algorithm is known to be slow near the solution.[9] Therefore, we added some extensions to make the algorithm more robust. These extensions include the classic PARTAN extension and a projected gradient push. We will conclude this section by discussing these enhancements.

Suppose we want to solve the nonlinear convex programming problem:

$$\begin{aligned} \text{Minimize} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{5}$$

where $f(\mathbf{x})$ is convex and differentiable. We assume the problem is feasible. Let $\mathbf{x}^0$ be an initial feasible point. If we have $\mathbf{x}^i$, we find $\mathbf{x}^{i+1}$ by solving the linear program:

$$\begin{aligned} \text{Minimize} \quad & \nabla f(\mathbf{x}^i)\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{6}$$

where $\nabla f(\mathbf{x}^i)$ is the gradient of the function $f(\mathbf{x})$ evaluated at $\mathbf{x}^i$. Suppose $\mathbf{y}^i$ solves the above linear program. Then

$$\mathbf{x}^{i+1} = \mathbf{x}^i + \lambda^i(\mathbf{y}^i - \mathbf{x}^i) \tag{7a}$$

where λ^i is defined to be

$$\min_{0 \leq \lambda \leq 1} f(\mathbf{x}^i + \lambda(\mathbf{y}^i - \mathbf{x}^i)) \tag{7b}$$

Therefore we solve a sequence of linear programs that approximate the nonlinear problem around the current iterate. Note that $f(\mathbf{x}^i)$ is an upper bound on the solution of the problem since $\mathbf{x}^i$ is feasible. We can also compute a lower bound at every iteration. If $\mathbf{x}^*$ is the solution to equation (5), we have

$$f(\mathbf{x}^*) \geq f(\mathbf{x}^i) + \nabla f(\mathbf{x}^i)(\mathbf{x}^* - \mathbf{x}^i) \geq f(\mathbf{x}^i) + \nabla f(\mathbf{x}^i)(\mathbf{y}^i - \mathbf{x}^i) \tag{7c}$$

where the first inequality is due to convexity and the second inequality is due to $\mathbf{y}^i$ minimizing the subproblem. The right-hand side of the inequality is easily computed at every iteration. Therefore at every iteration we have a lower and upper bound on the optimal objective function value. When these bounds are sufficiently close to each other, the algorithm terminates.

Returning to the penalty objective function, we first note that the nonlinear function we propose to minimize in equation (4) is nondifferentiable. Therefore we smooth the objective function as follows:

$$h_i(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbf{B}_i\mathbf{x} - u_i \leq -\Delta \\ G(\mathbf{B}_i\mathbf{x} - u_i + \Delta)^2/(4\Delta) & \text{if } -\Delta \leq \mathbf{B}_i\mathbf{x} - u_i \leq \Delta \\ G(\mathbf{B}_i\mathbf{x} - u_i) & \text{if } \mathbf{B}_i\mathbf{x} - u_i > \Delta \end{cases}$$

where Δ is some positive number. This function is differentiable and convex. Therefore it satisfies the criteria needed to apply the Frank-Wolfe algorithm.

The key to the efficiency of this algorithm is

solving the linear programming (LP) subproblem quickly. In general, this may not be possible, but for our application we have a special structure. In fact, the LP subproblem is trivially solved by inspection. This follows because the demand-path incidence matrix is completely decoupled. Therefore the solution is found by loading all the demands on the cheapest path for each demand pair.

The time-consuming portion of the algorithm is finding λ. We use the "golden section" search to approximate λ. This is a search technique that is guaranteed to trap the minimum of a convex function within an interval of a prescribed length by making successive function evaluations. For a more detailed description of the search technique, see Reference 10.

Algorithm Extensions. The Frank-Wolfe algorithm is known to have slow convergence near the solution. The problem lies in the fact that, near the solution, the directions become nearly orthogonal to the direction of steepest descent. This occurs because the linear programming subproblems have solutions at vertices while the actual nonlinear solution may have a solution far from the vertices. Therefore, the algorithm has a zigzag pattern as it moves in directions back and forth between a set of vertices. The two extensions we have added are designed to try to increase the set of possible directions.

The PARTAN extension is a generalization of the parallel tangent method for unconstrained optimization, and is used to avoid zigzagging. A general description of this method can be found in Reference 11. Briefly stated, equation (7a) is replaced by

$$\boldsymbol{\omega} = \mathbf{x}^i + \lambda^i(\mathbf{y}^i - \mathbf{x}^i)$$

where λ^i is defined to be

$$\min_{0 \le \lambda \le 1} f(\mathbf{x}^i + \lambda(\mathbf{y}^i - \mathbf{x}^i))$$

Now define the search direction $\mathbf{d}^i = \boldsymbol{\omega} - \mathbf{x}^{i-1}$. Then

$$\mathbf{x}^{i+1} = \mathbf{x}^i + \lambda^i \mathbf{d}^i$$

where λ^i is chosen to solve the one-dimensional minimization problem

$$\min_{0 \le \lambda \le \tilde{\lambda}} f(\mathbf{x}^i + \lambda \mathbf{d}^i)$$

and $\tilde{\lambda}$ is the largest step possible without violating the constraints. Note that in our application this amounts to preserving the nonnegativity, since the equality constraints will be preserved everywhere along that line.

The other extension we add is a projected gradient push. It can be shown that when we start on a face that contains $\mathbf{x}^*$, Frank-Wolfe will converge linearly. Thus, the idea of the projected gradient push is to push the iterate to a face using a projected gradient direction, and then restart Frank-Wolfe. Therefore, given the current iterate, we identify a set of active constraints—in other words, constraints which must be satisfied with equality. This set will include the demand requirements. We will also hold those variables that are zero at the current iterate to be zero. Define $\tilde{A}$ to be the matrix A with certain columns deleted. In particular, we delete the columns associated with variables that we will hold fixed at zero. Then define $\nabla f(\mathbf{x}^i)$ as the gradient of $f(\mathbf{x})$ at $\mathbf{x}^i$ without the variables that are held fixed. Then we define

$$\mathbf{d}^i = -[\mathbf{I} - \tilde{A}^T(\tilde{A}\tilde{A}^T)^{-1}\tilde{A}]\nabla f(\mathbf{x}^i)$$

The right-hand side is the vector $\nabla f(\mathbf{x}^i)$ projected onto the null space of $\tilde{A}$.

Then we set

$$\mathbf{x}^{i+1} = \mathbf{x}^i + \lambda^i \mathbf{d}^i$$

where λ^i is defined to be

$$\min_{0 \le \lambda \le \tilde{\lambda}} f(\mathbf{x}^i + \lambda \mathbf{d}^i)$$

and $\tilde{\lambda}$ represents the largest step possible without violating the nonnegativity constraints.

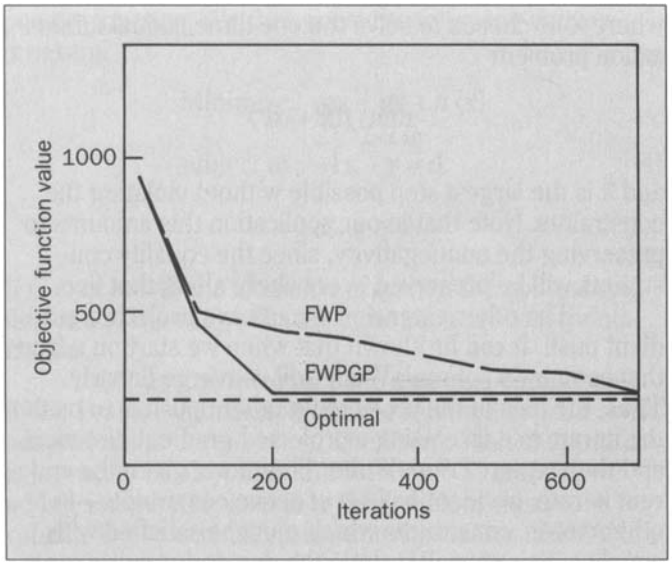

Figure 1. Comparisons between FWP and FWPGP for a small problem.

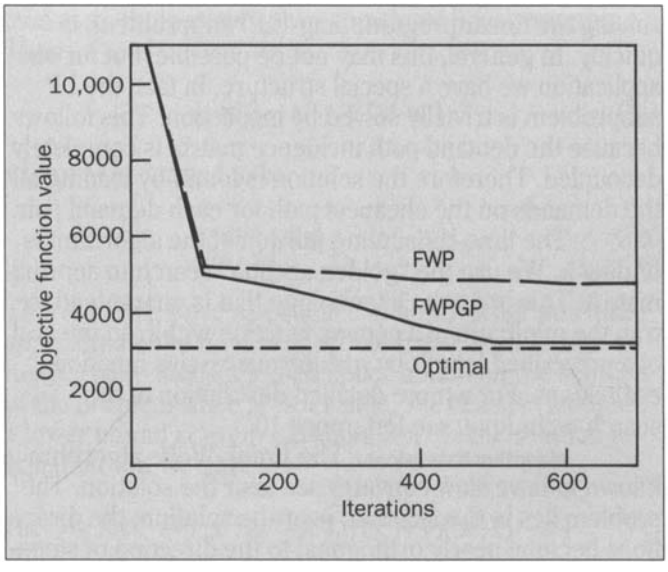

Figure 2. Comparisons between FWP and FWPGP for a larger problem.

The idea of this hybrid approach is to try to accelerate the Frank-Wolfe algorithm by giving it a restart. The Frank-Wolfe algorithm is very fast until the zigzagging pattern begins. The projected descent direction is better than the Frank-Wolfe direction when the algorithm is near the solution. Therefore, the idea is to start out with a Frank-Wolfe algorithm and later switch to the projected gradient technique. However, if we fail to identify the correct active set, convergence could be slow for a straight projected gradient method. Therefore, we return to the Frank-Wolfe algorithm after taking the projected step. If the projected step has pushed the iterate onto a face that contains the solution, Frank-Wolfe will converge linearly. Otherwise, we continue the hybrid approach.

This approach turns out to be very efficient for the loader model because of the special structure of the demand path matrix. Generally this direction would be expensive to compute because we must find the inverse of a matrix that takes $O(n^3)$ steps, where n is the dimension of the matrix $\tilde{A}\tilde{A}^T$. However, because the matrix is completely decoupled, $\tilde{A}\tilde{A}^T$ is a diagonal matrix, so the inverse of $\tilde{A}\tilde{A}^T$ is readily obtained.

Performance

In this section we present some computational results. These results are obtained on the AT&T KORBX® system, a processor with eight vector central processing units (CPUs) operating in parallel. We conducted two sets of tests of the algorithm. We conducted tests on randomly generated networks with known solutions and on realistic network data from previous planning runs. We first describe the randomly generated tests.

The random examples test the robustness of the algorithm to handle changing characteristics in the

Table I. Comparison of FWPGP Loader with a Commonly Used Heuristic

Measure	Test 1		Test 2		Test 3	
	FWPGP	Heuristic	FWPGP	Heuristic	FWPGP	Heuristic
Normalized loaded demand	107.5	100	105	100	105	100
CPU minutes	0.5	298	2.5	NA*	5	300

*Not available.

network. For instance, a network with much spare capacity is easier to solve than a tightly constrained network. We used a random network generator to give us the network structure, including links, paths, and terminal pairs with their associated demands. We then used linear programming theory to generate path costs and link capacities in order to guarantee the optimality of a certain prescribed solution. We ran these problems using both the Frank-Wolfe method with PARTAN extension (FWP) and the Frank-Wolfe method with PARTAN and projected gradient push (FWPGP). The stopping criterion was chosen, using equation (7c), so that the algorithm terminates when the objective function value is within 3 percent of the optimal. Figures 1 and 2 compare the objective function convergence between FWP and FWPGP. Figure 1 is a 10-node problem with 165 total paths and 30 links over 3 time periods. Figure 2 is a 50-node problem with 2115 paths and 200 links over 5 time periods. We can see that the projected gradient push greatly improves convergence. In fact, FWP would have been unacceptable by itself in the larger problem. Here, an iteration means a Frank-Wolfe iteration that includes the PARTAN extension and a projected gradient push, if needed. Also note that the CPU time spent on the projected gradient computation is relatively negligible since the only work involved is to find the inverse of the diagonal matrix $\tilde{A}\tilde{A}^T$.

The algorithm was also tested on three separate sets of data from three telecommunications network loading problems. Table I compares our loader with a commonly used heuristic that loads demands *sequentially* on judiciously selected paths. We compare two performance measurements: normalized loaded demand and computation time.

Test 1 has 3900 total paths and 590 links. Test 2 has approximately 10,000 paths and 1000 links. Test 3 has 15,700 paths and 5300 links. In all three problems, not all of the demand could be loaded (i.e., there is no solution with $\mathbf{y} = \mathbf{0}$). This is typically the case in realistic loading problems. The Frank-Wolfe method consistently loaded more demand and is much faster. The loaded demand numbers were normalized to be 100 for the heuristic. The percent of demand loaded by FWPGP was between 5 to 7.5 percent larger than that of the heuristic. As a result, we expect substantial construction savings from our approach. We used the same stopping criterion for the algorithm, i.e., when the objective function value is within 3 percent of the optimal. Regarding the required CPU time, FWPGP solved these large problems very fast, in 0.5 to 5 minutes. Note that the heuristic requires substantially more computation time, since the whole loading problem has to be solved several times in order to achieve satisfactory results.

Conclusion

In this paper we described a variant of the Frank-Wolfe algorithm for solving large-scale telecommunication network loading problems. By exploiting the structure of our arc-path formulation for the loader, *global* and *near optimal* solutions were obtained in relatively short computer time with minimum effort.

An unstated benefit from the Frank-Wolfe

method is the speedy near-optimal solution usually obtained. Speedy near-optimality is a more relevant function of a loader than true optimality, given the uncertainties of future demand and cost data. It would be valuable to examine what other types of constraints, encountered in different loading problems, can be handled by variants of our algorithm. For example, diversity constraints, which limit the percent of any demand pair routed through any link, can readily be handled. We are currently pursuing such extensions and other open issues in order to develop more powerful loaders that can support a variety of network planning functions.

We conclude with an observation shared with LeBlanc et al.,[12,13] that the Frank-Wolfe algorithm has also been successfully used by transportation planners.

References

1. A. A. Assad, "Multicommodity Network Flows—A Survey," *Networks*, Vol. 8, No. 1, Spring 1978, pp. 37-91.
2. B. Yaged, "Minimum Cost Routing for Dynamic Network Models," *Networks*, Vol. 3, No. 3, Fall 1973, pp. 193-224.
3. C. E. Clark and J. L. Strand, "Application of the Karmarkar Algorithm and Expert System Technology to Transmission Network Planning," *Conference Record*, Vol. 2, IEEE/IEICE Global Telecommunications Conference, Tokyo, 1987, p. 270.
4. L. R. Ford and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, New Jersey, 1962.
5. K. Murty, *Linear and Combinatorial Programming*, John Wiley and Sons, New York, 1976.
6. R. E. Gomory and T. C. Hu, "An Application of Generalized Linear Programming to Network Flows," *Journal of the Society for Industrial and Applied Mathematics*, Vol. 10, No. 2, June 1962, pp. 260-283.
7. S. Chen and D. N. Lee, "Supercomputers and an Efficient Implementation of Karmarkar's Algorithm," *National Meeting on Numerical Analysis*, Society for Industrial and Applied Mathematics, Denver, 1987.
8. M. Frank and P. Wolfe, "An Algorithm for Quadratic Programming," *Naval Research Logistics Quarterly*, Vol. 13, 1987, pp. 95-110.
9. P. Wolfe, "Convergence Theory in Nonlinear Programming," *Integer and Nonlinear Programming*, J. Abadie, ed., North Holland Publishing, New York, 1970.
10. P. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, New York, 1981.
11. D. M. Himmelblau, *Applied Nonlinear Programming*, McGraw-Hill, New York, 1972.
12. L. J. LeBlanc, E. K. Morlok, and W. P. Pierskalla, "An Efficient Approach to Solving the Road Network Traffic Assignment Problem," *Transportation Research*, Vol. 9, 1975, pp. 309-318.
13. M. Florian, "Nonlinear Cost Network Models in Transportation Analysis," Publication 287, Center for Transportation Research, University of Montreal, March 1983.

Biographies (continued)

core algorithm development for the AT&T KORBX® system. She received a B.A. in mathematics from St. Olaf College, an M.S. in mathematics from the University of Illinois, Urbana, and a Ph.D. in industrial engineering from the University of Wisconsin, Madison. She joined AT&T in 1987. Mr. Strand is a distinguished member of technical staff in the Integrated Network Planning Department. He works on architecture, design, and development of new network planning tools. He has an A.B. degree in economics from Harvard College and a Ph.D. in mathematics from the University of California, Berkeley. He joined AT&T in 1968. Mr. Cox is a supervisor in the Integrated Network Planning Department. He manages development of network decision support systems and consults on network design and deployment. He received a B.A. in mathematics and an M.S.Eng. in public decision making, both from The Johns Hopkins University, and a Ph.D. in transportation and logistics from Cornell University. He joined AT&T in 1984. Mr. Chen is head of the Integrated Network Planning Department. He works on planning for the AT&T domestic and overseas networks. He has a B.S. in mathematics from the University of Pennsylvania and an M.S. in computer science from the University of California, Berkeley. He joined AT&T in 1970.

(Manuscript received January 23, 1989)