Advances in Solving the Multicommodity-Flow Problem

Richard D. McBride

Marshall School of Business
University of Southern California
Los Angeles, California 90089-1421

The multicommodity-flow problem arises in a wide variety of important applications. Many communications, logistics, manufacturing, and transportation problems can be formulated as large multicommodity-flow problems. During the last few years researchers have made steady advances in solving extremely large multicommodity-flow problems. This improvement has been due both to algorithmic and to hardware advances. At present the primal simplex method using the basis-partitioning approach gives excellent solution times even on modest hardware. These results imply that we can now efficiently solve the extremely large multicommodity-flow models found in industry. The extreme-point solution can also be quickly reoptimized to meet the additional requirements often imposed upon the continuous solution. Currently practitioners are using EMNET, a primal basis-partitioning algorithm, to solve extremely large logistics problems with more than 600,000 constraints and 7,000,000 variables in the food industry.

Many large-scale models are formulated as multicommodity-flow problems. "This model arises in a wide variety of application settings in communications, logistics, manufacturing, and transportation as well as in such problem domains as urban housing and food grain export-import" [Ahuja, Magnanti, and

0092-2102/98/2802/0032/$5.00
This paper was refereed.

NETWORKS/GRAPHS—MULTICOMMODITY
TRANSPORTATION—FREIGHT/MATERIALS HANDLING

Orlin 1993]. Researchers usually describe the distribution-management model when they present the classical multicommodity problem. This model describes a separate network flow for each product (commodity). Side constraints are added to the model that limit the total amount of flow between certain origin-destination pairs. In real-world models, these joint capacitation constraints describe inventory-storage, loading-dock, or truck (train) limitations. Practitioners are building and solving formulations of ever-increasing size and complexity that capture economics of scale and coordination among plants and distribution centers. As models have increased in size and complexity, the cost of computation has dropped to a point where it is now feasible to solve very large multicommodity-flow problems on hardware of modest cost. Surveys of the application of multicommodity-flow problems have been published by Assad [1978], Kennington [1978], and Tomlin [1966]. In the food industry, practitioners now routinely solve extremely large multicommodity-flow problems with more than 600,000 constraints and 7,000,000 variables using an enhanced version of EMNET [McBride 1985], a basis-partitioning simplex algorithm. These are huge multi-time-period logistics models with postoptimization required for sole sourcing.

The multicommodity-flow problem is usually presented as having the following form:

$$\begin{aligned} \min_{x \geq 0} \quad & cx \\ & Nx = b \\ & Ax \leq d \end{aligned} \tag{1}$$

where the objective function, cx, is to minimize costs, the network constraints, $Nx = b$, ensure that supply, demand, and shipping-route requirements are satisfied, and $Ax \leq d$ are the joint capacitation constraints.

When product substitutions are not permitted, the network part of the problem decomposes into K disjoint networks. N then has the form

$$N = \begin{bmatrix} N_1 & & & & \\ & N_2 & & & \\ & & \cdots & & \\ & & & N_{K-1} & \\ & & & & N_K \end{bmatrix} \tag{2}$$

there being K commodities. N_i is the network for the ith commodity.

Researchers have studied four different approaches to determine the best way to solve the multicommodity-flow problem. Schultz and Meyer [1991], Zenios and Pinar [1992], and Zenios, Pinar, and Dembo [1995] use decomposition techniques. Carolan et al. [1990], Lustig and Rothberg [1996], Marsten et al. [1990], and Schultz and Meyer [1991] use interior-point algorithms to solve the multicommodity-flow problem. Schultz and Meyer [1991] use both decomposition and interior-point methods. Another approach is to use the solution of the network part of the problem to hot-start the simplex method; McBride and Mamer [1997] give computational results. The fourth approach is to use the simplex method with a partitioned basis. Initial work on the efficient implementation of primal partitioning techniques includes that of Barr et al. [1987], Chen and Saigal [1977], Glover and Klingman [1981], Graves and McBride [1974, 1976], and McBride [1973, 1978a, 1978b, 1980, 1985].

In many of these works, the word *factorization* is used instead of *partitioning*. Work specialized to the multicommodity-flow problem includes that of Ali et al. [1980], Castro and Nabona [1996], Farvolden et al. [1993], Kennington [1977], Jones et al. [1993], Mamer and McBride [1997], McBride [1996], and McBride and Mamer [1995, 1997 (work done in 1991–1993)]. I will review the computational results of some of these researchers and highlight how they have used vectorization, multiple processors, and new faster single processors. I will also review how each has done on solving PDS-20, the one common problem many of these researchers have solved.

I have been able to make reasonable comparison of solution approaches because of the availability of the KEN and PDS problem sets. Many of the above researchers have included results using problems selected from the KEN and PDS problem sets. KEN-11, KEN-13, and KEN-18 are randomly generated test problems using the code MNETGN by Ali and Kennington, and they are available from the NETLIB library. The PDS test problems are created using a generator obtained from Robert Meyer. Meyer obtained the generator from the CINCMAC analysis group of the Military Airlift Command (MAC), now called the Airlift Mobility Command (AMC), at Scott Air Force Base. The patient-distribution system (PDS) model is a logistics model designed to help make decisions about how well MAC (AMC) can evacuate patients from Europe. PDS-D denotes a problem that models a scenario lasting D days, for integers $D \epsilon$ [1,85]. The PDS problems are multicommodity-flow problems that become quite large and more difficult to solve as D becomes larger. PDS-02, PDS-06, PDS-10, and PDS-20 are available from the NETLIB library. Table 1 shows a complete description of the problems.

Decomposition Methods

The smooth-penalty-function algorithm proposed by Zenios, Pinar, and Dembo [1995] uses a linear-quadratic (LQR) function to eliminate the side constraints and produce a differentiable problem. Next simplical decomposition is used to decompose the problem into a set of linear problems, one for each commodity.

This algorithm is very suitable for the vector architecture of the Cray Y-MP. Zenios, Pinar, and Dembo [1995] solved PDS-05 in 93.10 seconds with vectorization and 352.98 seconds without. They obtained performance improvements in the range 1.92–3.76 by using vectorization. Zenios and Pinar [1992] gives the PDS-30 solution time. They appear to use the same algorithm in both papers. On the CRAY Y-MP, PDS-20 took 1,946 seconds with one processor and 740 seconds with eight processors. They obtained approximately five digits of accuracy in their computer runs.

Schultz and Meyer [1991] created a barrier function by placing the side constraints in the objective function. They then used a generalization of the Frank-Wolfe method to linearize the objective function, which decomposes it into an independent problem for each commodity using the structure of (2). Even though the functions are different, there is some similarity with the work of Zenios and Pinar [1992] and Zenios, Pinar, and Dembo

Problem	Number of Nodes	Number of Side Constraints	Total Number of Constraints	Total Number of Variables	Number of Commodities
KEN-11	14,641	53	14,694	21,346	121
KEN-13	28,561	71	28,632	42,659	169
KEN-18	104,976	151	154,127	154,699	324
PDS-01	1,386	87	1,473	3,816	11
PDS-02	2,772	181	2,953	7,716	11
PDS-03	4,290	303	4,593	12,590	11
PDS-05	7,546	553	8,099	24,192	11
PDS-06	9,185	696	9,881	29,351	11
PDS-10	15,389	1,169	16,558	49,932	11
PDS-20	31,427	2,447	33,874	108,175	11
PDS-30	46,453	3,491	49,944	158,489	11
PDS-40	62,172	4,672	66,844	217,531	11
PDS-50	77,341	5,719	83,060	275,814	11
PDS-60	92,653	6,778	99,431	336,421	11
PDS-70	107,250	7,694	114,944	390,005	11
PDS-80	120,879	8,302	129,181	434,580	11
PDS-85	127,556	8,557	136,113	455,488	11

Table 1: Research test set of problems.

[1995]. They used a Sequent computer with 20 processors. During the solution process, they assigned each of the 11 commodities to a separate processor to reduce solution time. They stopped the algorithm after 50 iterations, which resulted in suboptimal solutions for the larger PDS problems. PDS-20 took 3,043 seconds on the Sequent computer. They obtained three digits of accuracy in the objective function for problems PDS-30, PDS-40, and PDS-50 and two digits of accuracy in problems PDS-60 and PDS-70. This is significant since the optimal objective-function value has 11 digits to the left of the decimal point. These results clearly show the slow convergence of decomposition methods.

Interior Point Methods

Lustig and Rothberg [1996] have obtained the best interior-point solution time for PDS-20 by using the parallel version of the CPLEX barrier algorithm running on a Silicon Graphics Power Challenge with 16 processors. They included PDS-20 in their set of test problems because of its large size. Their computation times for PDS-20 are 2,404 seconds for one, 794 seconds for four, 501 seconds for eight, and 409 seconds for 16 processors. Each of the R8000 processors is considered about equal to one CRAY Y-MP processor. On the Power Challenge, each processor has a four Mbyte cache. To get these excellent results, Lustig and Rothberg organized computations very carefully so that the caches had high hit rates.

Previous interior-point solution times for solving the KEN and PDS problems include work done by Carolan et al. [1990] and Marsten et al. [1990]. Marsten et al. obtained these results using OB1 on a CRAY Y-MP with one processor using the

CRAY's vectorization capabilities. OB1 took 15,972 seconds to solve PDS-20. Carolan et al. obtained these results using the KORBX computer. The KORBX had 14 processors with eight used for computation. The other six processors were used for editing and data manipulation. Each of the eight computation processors had its own vectorization capability. PDS-20 took 63,720 seconds on the KORBX system.

Simplex Method with Advanced Basis

Some have thought that just using the advanced-network-starting basis in the regular simplex method would suffice to get good solution times. Without the advanced start, KEN-11 took 450 seconds, while with the advanced start, it took 27 seconds to be solved to optimality. The advanced start can make a big difference but the extremely large size of the LU factorization for the larger multicommodity-flow problems soon has a very debilitating influence upon the effectiveness of the simplex method. PDS-20 took 4,324 seconds using OSL with the advanced-network start on an IBM mainframe. The solution times greatly increase as the total number of constraints increases as seen in the solution times of KEN-18 and PDS-30. PDS-30 ran for three and one half days on the mainframe (getting only a limited use of a CPU) for a total of 45,069 seconds [McBride and Mamer 1997].

The Basis-Partitioning Method

In the basis-partitioning method, the simplex basis is partitioned so that the working explicit inverse is of a dimension at most equal to the number of side constraints. This much smaller inverse is maintained as the simplex method is executed.

The normal solution approach in this method can be described in two steps:
(1) Drop the side constraints and solve the resulting network problems by commodity;
(2) Solve the full problem using the network solution as part of an advanced basis using the partitioned basis.

There have been a number of implementations of this basis partitioning scheme [Barr, Farhangian, and Kennington 1987; Chen and Saigal 1977; Glover and Klingman 1981; and McBride 1978a, 1978b, 1980, 1985.].

Even using the partitioned basis does not guarantee good solution times on extremely large multicommodity-flow problems. The advanced starting basis may not be good enough to considerably reduce the number of pivots when in step 2, which has the more computationally intensive pivots. The dimension of the working inverse may still be very large, requiring a substantial amount of computation per pivot. The pricing strategy used may need to be specialized for this class of problems to further reduce the number of pivots. The work required to update the duals for the network constraints on each pivot may be substantial. PDS-85 has 127,556 network duals. Not every network dual changes on each pivot, but in the usual implementations of the simplex method, the solver must determine on each pivot which ones change and update them. When one deals with these issues properly, one obtains extremely good solution times using modest hardware. In the appendix, I discuss enhancements made to EMNET [McBride 1985] in more detail.

After enhancements to the basic-partitioning-simplex approach, the solution time for PDS-20 is 49 seconds on a machine with a 500 MHZ Alpha processor.

Conclusions

It appears that currently the basis-partitioning implementation of the simplex method is a good way to solve multicommodity-flow problems. This is true for PDS-1 to PDS-85 and also for all of the KEN problems. Super computers with multiple processors, vectorization capabilities, and large amounts of RAM are not needed for the basis-partitioning approach (Table 2). I obtained these results with a Carrera Cobra Alpha AXP 21164 EV-56 at 500 MHZ with 128 megabytes of RAM. These modest hardware requirements will permit companies to efficiently solve extremely large multicommodity-flow problems on workstations already in their possession.

The solution times for EMNET (Table 2) show a near-linear increase in solution times from PDS-30 to PDS-85. This is very promising since it implies that we can now consider industry-size problems. My experience confirms this. For one food company, problem sizes have increased to more than 600,000 constraints and 7,000,000 arcs. The EMNET solution times for the initial continuous LP for these huge problems is less than that needed to solve PDS-80. It seems that many times real industrial problems are easier to solve than the larger PDS problems.

Most multicommodity logistics prob-

Problem	Results from Pinar and Zenios [1992]	Results from Pinar and Zenios [1990]	Results from Marsten et al. [1990]	Results from Schultz and Meyer [1991]	Results from Carolan et al. [1990]	Results from McBride and Mamer [1997]	Results from Lustig and Rothberg [1996]	Results with Enhanced EMNET
Ken-11	16		23		396	27		1.73
Ken-13	138		67		1,242	573		10.32
Ken-18			680		15,840	12,764		46.81
PDS-01	1.86			64				.20
PDS-02			11	129	148			.26
PDS-03	19			218				.48
PDS-05	93	23		403				1.64
PDS-06			295	524	1,692			2.54
PDS-10	408	96	1,521	999	11,880	475		6.70
PDS-20	1,946	740	15,972	3,043	63,720	4,324	409	49
PDS-30	7,504	2,566		6,480		45,069		162
PDS-40				10,440				428
PDS-50				19,800				647
PDS-60				24,900				963
PDS-70				33,840				1,152
PDS-80								1,582
PDS-85								1,678

Table 2: Known results (times in seconds).

lems solved in practice require additional postprocessing of the raw continuous solution to eliminate small truckloads and to consolidate orders. This includes the difficult sole-sourcing requirement. The extreme-point solution obtained using the basis-partitioning approach can be used for efficient postprocessing. While the interior-point approach can take better advantage of multiple processors, it is less efficient when doing postprocessing. Postprocessing capabilities have been added to EMNET for the extremely large logistics model I discussed in the previous paragraph to eliminate small truckloads and consolidate products in orders. The company solves the six-week model five days a week for planning purposes. It uses one of the model solutions for the actual shipment of its products for the next week. I have added postprocessing capabilities to EMNET to sole source a large multicommodity logistics problem for a different customer in the food industry. Unfortunately, for proprietary reasons, I cannot say more about these extensions of EMNET for these commercial applications. The sole-sourcing requirements of these two businesses are quite different, but the extreme-point solutions I obtained using the basis-partitioning approach make this postprocessing much easier to do.

APPENDIX

The heart of the basis-partitioning implementation of the simplex method is the partitioned basis. The basis is partitioned into a network part and a nonnetwork part. The basic network part is further partitioned into a network basis (nonsingular square submatrix) G_1 and the remaining part G_2. Partitioning the basic part of A to conform to the $[G_1, G_2]$ partition yields $[A_1, A_2]$. The full basis partition is as follows:

$$B = \begin{bmatrix} G_1 & G_2 \\ A_1 & A_2 \end{bmatrix}$$

and the partitioned basis inverse is given by

$$B^{-1} = \begin{bmatrix} (I + G_1^{-1}G_2H^{-1}A_1)G_1^{-1} & -G_1^{-1}G_2H^{-1} \\ -H^{-1}A_1G_1^{-1} & H^{-1} \end{bmatrix}$$

where

$$H = A_2 - A_1G_1^{-1}G_2. \qquad (3)$$

The duals are computed according to the standard formula c_BB^{-1} which, when partitioned as above, yields

$$w_2 = (c_2 - c_1G_1^{-1}G_2)H^{-1}, \qquad (4)$$
$$w_1 = (c_1 - w_2A_1)G_1^{-1}, \qquad (5)$$

where the cost vector $c = [c_1, c_2]$ has been partitioned in a fashion conformable to G_1 and G_2. The w_2 values in (4) are the dual variables for the side constraints and the w_1 values in (5) are the dual variables for the constraints corresponding to the nodes in the network. H^{-1}, G_1, and the original problem data are used to execute the simplex method. Here the dimension of the working inverse, H^{-1}, or H in (3) is equal to the number of side constraints.

I will describe the key features and enhancements made to the EMNET [McBride 1985]. These enhancements enable EMNET to achieve the computational results reported here for solving multicommodity-flow problems.

Improving the Advanced Starting Basis

McBride and Mamer [1997] developed a primal allocation decomposition heuristic to move the network solution closer to the optimal solution and to satisfaction of the side constraints. After the network solution is obtained, it checks all of the side constraints for violation. It proportionally assigns the positive variables in the violated side constraints new upper bounds so that if each variable were reduced in value to its new upper bound, the constraints would be satisfied. It enforces the new upper bounds by using higher objective costs imposed using an implicit implementation of piecewise separable convex costs. It reoptimizes the network problem(s). As it imposes the new upper bounds, some variables that were previously zero take on positive values. It makes a new proportional assignment of upper bounds again across all positive variables and again reoptimizes the network problem(s). The process is continued until there is no further improvement in the total infeasibleness of the side constraints. It cuts the infeasibility of the side constraints roughly in half on each iteration for the PDS problems.

Reducing the Size of the Working Inverse, H^{-1}

McBride [1996] introduces a heuristic that controls the dimension of H^{-1} so that it is very close to the number of binding side constraints. This resulted in a 45-percent reduction in solution times for PDS-50 and PDS-60 and a 44-percent reduction for PDS-70 and PDS-80. It reduced the solution time for PDS-85 by 48 percent. The basic idea of the heuristic is to just include the side constraints in H^{-1} when they are violated or are very close to being violated; otherwise they are ignored. Each time the working basis, H^{-1}, is refactored, the side constraints are checked to see if any additional ones should be included in H^{-1}. About 24 percent of the side constraints are binding for PDS-80 and PDS-85 at optimality. For the smaller PDS problems, a slightly larger percentage of the side constraints are binding.

Computing Duals w_1 on Demand

The duals in w_1, the dual variables corresponding to the network constraints, are computed on demand during pricing. Usually only a small percentage of the w_1 duals are needed during a typical pricing step. A careful study of Equation (5) shows that the first time during the pricing step of a pivot that a particular dual is needed one can just proceed up the tree until one encounters a node that already has its dual computed for the pivot or a root node. Then one can proceed back down the back-path computing only the duals on the back-path. One also computes the $(c_1 - w_2A_1)$ values in (5) as needed. When EMNET [McBride 1985] was originally developed, I observed a great reduction in solution times when I added this feature.

Using Decomposition for Pricing

One can reduce the number of more computationally intensive pivots that require the use of H^{-1} with a good pricing strategy. The pivots taken in improving the advanced starting basis are made using the very efficient network technology. When we relax the complicating constraints, $Ax \leq b$, by placing them in the objective function with the dual variables w_2, we create the Lagrangian subproblem. Researchers usually use this Lagrangian subproblem to compute lower bounds for problem (1). Mamer and McBride [1997] show that one can use the solution to the Lagrangian subproblem for guidance in pricing. When N has the structure shown in (2), then the Lagrangian subproblem decomposes into the K smaller subproblems $p_1(w_2), \ldots p_k(w_2)$. One would proceed by solving $p_1(w_2)$ and placing the first few entering variables during the solution process (to ensure and to speed up convergence) and the positive variables in the solution in a candidate queue for pricing. Then one would price the variables in the candidate queue and pivot until no further improvement can be made. Move to the next subproblem $p_2(w_1)$ and repeat the same process. Proceed cycling through the subproblems until optimality. This approach has a resemblance to the classical decomposition methods. A restricted version of the original problem (1) is the master problem and the solution of the Lagrangian subproblem provides pricing guidance rather than a new variable to the master problem. Mamer and McBride [1997] show a reduction in solution times of nearly 50 percent for PDS-70, PDS-80, and PDS-85. The number of step 2 pivots in PDS-85 with this pricing strategy is reduced by more than two thirds. I have obtained similar reductions in solution times for several other proprietary problems.

Using an Efficient Implementation of LU Factorization for H^{-1}

The efficient implementation of LU factorization of H^{-1} [McBride and Mamer 1995] gives EMNET a fast, stable, and efficient representation of H^{-1}.

References

Assad, A. 1978, "Multicommodity network flows: A survey," *Networks,* Vol. 8, No. 1, pp. 37–92.

Ahuja, R.; Magnanti, T.; and Orlin, J. 1993, *Network Flows: Theory, Algorithms, and Applications,* Prentice-Hall, Englewood Cliffs, New Jersey.

Ali, A.; Helgason, R.; Kennington, J.; and Lall, H. 1980, "Computational comparison among three multicommodity network flow algorithms," *Operations Research,* Vol. 28, No. 4, pp. 995–1000.

Barr, R.; Farhangian, K.; and Kennington, J. 1987, "Networks with side constraints: An LU factorization update," *Annals of the Society of Logistics Engineers,* Vol. 1, No. 1, pp. 66–87.

Carolan, W.; Hill, J.; Kennington, J.; Niemi, S.; and Wichman, S. 1990, "An empirical evaluation of the KORBX algorithms for military airlift applications," *Operations Research,* Vol. 38, No. 2, pp. 240–248.

Castro, J. and Nabona, N. 1996, "An implementation of linear and nonlinear multicommodity network flows," *European Journal of Operational Research,* Vol. 92, No. 1, pp. 37–53.

Chen, S. and Saigal, R. 1977, "A primal algorithm for solving a capacitated network flow problem with additional linear constraints," *Networks,* Vol. 7, No. 1, pp. 59–79.

Farvolden, J.M.; Powell, W. B.; and Lustig, I. J. 1993, "A primal partitioning solution for the arc-chain formulation of a multicommodity network flow problem," *Operations Research,* Vol. 41, No. 4, pp. 669–693.

Glover, F. and Klingman, D. 1981, "The simplex son algorithm, for LP/embedded network problems," *Mathematical Programming Study 15* (May), pp. 148–176.

Graves, G. W. and McBride, R.D. 1976, "The factorization approach to large scale linear programming," *Mathematical Programming,* Vol. 10, No. 1, pp. 91–110.

Graves, G. W. and McBride, R. D. 1974, "A dynamic factorization algorithm for general large scale linear programming problems," paper presented at the ORSA/TIMS National Meeting in San Juan, Puerto Rico.

Kennington, J. 1977, "Solving multicommodity transportation problems using a primal partitioning simplex method," *Naval Research Logistics Quarterly,* Vol. 24, No. 2, pp. 309–325.

Kennington, J. 1978, "A survey of linear multicommodity network flows," *Operations Research,* Vol. 26, No. 2, pp. 209–236.

Jones, K. L.; Lustig, I. J.; Farvolden, J. M.; and Powell, W. B. 1993, "Multicommodity network flows: The impact of formulation on decomposition," *Mathematical Programming,* Vol. 62, No. 1, pp. 95–117.

Lustig, I. and Rothberg, E. 1996, "Gigaflops in linear programming," *Operations Research Letters,* Vol. 18, No. 4 (August), pp. 157–165.

Mamer, J. and McBride, R. D. 1997, "A decomposition based pricing procedure for large scale linear programs," working paper, University of Southern California.

Marsten, R.; Subramanian, R.; Lustig, I.; Saltzman, M.; and Shanno, D. 1990, "Interior point methods for linear programming: Just call Newton, Lagrange, and Fiacco and McCormick!," *Interfaces,* Vol. 20, No. 4, pp. 105–116.

McBride, R. D. 1973, "Factorization in large-scale linear programming," Ph D dissertation, University of California, Los Angeles.

McBride, R. D. 1978a, "Dynamic generalized upper bounding," *Operations Research,* Vol. 26, No. 2, pp. 365–369.

McBride, R. D. 1978b, "A spike collective dynamic factorization algorithm for the simplex method," *Management Science,*" Vol. 24, No. 10, pp. 1031–1042.

McBride, R. D. 1980, "A bump triangular dynamic factorization algorithm for the simplex method," *Mathematical Programming,* Vol. 18, No. 1, pp. 49–61.

McBride, R. 1985, "Solving embedded generalized network problems," *European Journal of Operational Research,* Vol. 21, No. 1, pp. 82–92.

McBride, R. 1996, "Progress made in solving the multicommodity flow problem," working paper, University of Southern California.

McBride, R. and Mamer, J. 1995, "Solving multicommodity flow problems," working paper, University of Southern California.

McBride, R. and Mamer, J. 1997, "Solving multicommodity flow problems with a primal embedded network simplex algorithm," *INFORMS Journal on Computing,* Vol. 9, No. 2 (Spring), pp. 154–163.

Pinar, M. and Zenios, S. 1992, "Parallel decomposition of multicommodity network flows using a linear-quadratic penalty-algorithm," *ORSA Journal on Computing,* Vol. 4, No. 3 (Summer), pp. 235–248.

Schultz, G. and Meyer, R. 1991, "An interior point method for block angular optimization," *SIAM Journal on Optimization,* Vol. 1, No. 4, pp. 583–602.

Tomlin, J. 1966, "Minimum cost multicommodity network flows," *Operations Research,* Vol. 14, No. 1, pp. 45–51.

Zenios, S.; Pinar, M.; and Dembo, R. 1995, "A smooth penalty function algorithm for network-structured problems," *European Journal of Operational Research,* Vol. 83, No. 1, pp. 220–236.

Editor's Note: The company mentioned in this paper wishes to remain anonymous. However, personnel at the company including the responsible manager have confirmed to me the veracity of the claims of usage and impact in this paper.