A Decomposition-Based Pricing Procedure for Large-Scale Linear Programs: An Application to the Linear Multicommodity Flow Problem

John W. Mamer • Richard D. McBride
Anderson Graduate School of Management, University of California, Los Angeles, Box 951481, 110 Westwood Plaza, Los Angeles, California 90095-1481
Marshall School of Business, University of Southern California, Los Angeles, California 90089-0809
john.mamer@anderson.ucla.edu • mcbride@rcf.usc.edu

We propose and test a new pricing procedure for solving large-scale structured linear programs. The procedure interactively solves a relaxed subproblem to identify potential entering basic columns. The subproblem is chosen to exploit special structure, rendering it easy to solve. The effect of the procedure is the reduction of the number of pivots needed to solve the problem. Our approach is motivated by the column-generation approach of Dantzig-Wolfe decomposition. We test our procedure on two sets of multicommodity flow problems. One group of test problems arises in routing telecommunications traffic and the second group is a set of logistics problem which have been widely used to test multicommodity flow algorithms.
(*Linear Programming*; *Multicommodity Flows*; *Optimization*)

1. Introduction

In this paper we propose an iterative partial pricing scheme for solving large-scale linear programs using the simplex method. The approach solves a relaxed subproblem to identify the columns to be considered during the pricing step of the simplex method. The goal of the procedure is to find an efficient set of candidate columns to enter the basis without incurring the cost of pricing all of the nonbasic columns in the problem at each iteration. Our scheme is motivated by the column-generation approach of Dantzig-Wolfe decomposition. We test our procedure on a set of publicly available multicommodity flow problems.

Our subproblem is identical to the usual decomposition subproblem; however, we replace the usual (partial) master problem with a "column restricted" version of the original problem. By "column restricted" we mean a version of the original problem with all of the original rows and a subset of the original columns. The dual solution from this restricted problem is used to construct the subproblem in which a subset of constraints (usually "complicating" constraints) are relaxed and the objective function modified so that violations of these constraints are penalized according to the values of the dual variables. Given an optimal solution to the subproblem, a new restricted problem is created from the columns corresponding to the nonzero components of this solution to the subproblem (in the case of an extreme point optimal solution, the basic columns) together with the basic columns of the current restricted problem. The restricted problem is resolved, yielding a

0025-1909/00/4605/0693$05.00
1526-5501 electronic ISSN

new dual solution. Constructing the restricted problem in this fashion assures on the one hand that a basic feasible solution to the original problem is maintained at each iteration (since the current basic columns are maintained), and on the other hand that the size of the problem to which we must apply the simplex method is quite small. One can think of the columns added to the restricted problem in each iteration as candidate entering basic variables. The procedure terminates when none of the columns obtained from the subproblem price out in the restricted problem. We assume that the problem is structured in such a way that a very easily solved subproblem can be identified. In many instances, the subproblem decomposes further into a set of smaller problems.

In its broadest outline our procedure owes much to Dantzig-Wolfe decomposition. In the usual approach to Dantzig-Wolfe decomposition (also called "price-directed" decomposition (see Nazareth 1987), structure in the constraints of the problem is used to identify an easily solved subproblem by dualizing a subset of the constraints. Our procedure is similar to Dantzig-Wolfe decomposition in this respect. However, instead of using the subproblem to produce an extreme point of the relaxed polytope for inclusion in a master problem (as in Dantzig-Wolfe decomposition), we use the optimal basic columns of the subproblem to identify columns to be included in a restricted version of the original problem. The restricted version of the original problem is solved to obtain an improved primal feasible basic solution to the original problem, and new dual prices to use in the next subproblem.

This work was motivated by our experience solving a very specialized multicommodity flow formulation of the message routing problem; in particular, the path-flow algorithm for solving this formulation. The path-flow approach to the multicommodity flow problem reformulates the problem so that the variables represent flows along paths from source to sink, not flows along individual arcs. Since the number of path flows is very large, column generation is used to reduce the computational effort. Ahuja et al. (1993, p. 665) give an interpretation of this approach within the framework of Dantzig-Wolfe decomposition. Under this scheme, the algorithm iterates between a subproblem that identifies potentially improving paths and a restricted (path-flow) master problem. Because of the simple structure of the original problem, the subproblems are readily solved by the shortest path algorithm. In McBride and Mamer (1997b), the subproblem solutions are used to guide pivot selection for a specialized version of EMNET (a partitioned basis embedded network simplex algorithm (McBride 1985)) applied to the arc-flow formulation of the message routing problem. Our experience showed a dramatic reduction in the number of pivots needed to solve the multicommodity flow problem, and a concomitant reduction in solution time (when compared to EMNET solving the multicommodity formulation directly). It was rather surprising to us that this column selection procedure afforded such large improvements in the solution time, especially in light of the extent to which EMNET has already been specialized to solve multicommodity flow problems. It was this experience that encouraged us to attempt to generalize the technique to other structured linear programming problems.

Our procedure is also similar to the "dynamic simplex algorithm" described by Padberg (1995). In this variant of the simplex method, a sequence of smaller problems are solved which lead to a solution to the original problem. The smaller problems are obtained from the original problem by dropping nonbasic variables and nonbinding constraints. Periodically, all rows and columns are scanned for possible inclusion into the master problem. In contrast, our procedure uses all rows and solves a surrogate problem to determine which columns to include in the restricted problem at each iteration.

Additionally, Valerio de Carvalho (1997) has used surrogate problems to determine which columns to include in a branch and bound solution to the bin-packing problem. In this work a subproblem is solved to identify columns to be included in a cutting stock formulation of the bin-packing problem.

In §2 of this paper we specify the procedure in detail and develop the relationship between our proposed procedure and the classical Dantzig-Wolfe decomposition algorithm, and the common partial pricing heuristics for the simplex method. In §3 of this paper

we give some indication of the salient issues faced during the implementation of our procedure.

An initial set of numerical results are contained in §4 of this paper. We test our ideas on two sets of problems. The first are the message routing problems described above. The second are a set of publicly available multicommodity flow problems. These multicommodity flow problems offer a more stringent test of our ideas since the resulting subproblems correspond to network flow problems instead of shortest path problems (the simplifying assumptions of the message routing problem are not met for these problems), and thus require more time to solve. The results we report for this latter set of problems improve upon our best solution times (reported in McBride and Mamer 1997a).

This paper, together with McBride and Mamer (1997a) and (1997b) are part of an ongoing interest of ours in using techniques pioneered in the context of decomposition algorithms to improve the performance of very large scale implementations of the simplex method.

2. Decomposition-Based Pricing

Consider a linear program of the following form:

$$V = \min \mathbf{cx},$$

$$\mathbf{Nx} = \mathbf{b},$$

$$\mathbf{Ax} \leq \mathbf{r},$$

$$\mathbf{x} \geq \mathbf{0}, \tag{1}$$

where the constraint matrix $\mathbf{N}$ possesses some special structure that makes the relaxed problem, obtained by dropping the constraints $\mathbf{A}$, especially easy to solve (network flow, maximum flow, assignment, shortest path, etc.). We assume that N has m rows and A has h rows, each matrix possesses n columns where, in practice, $n \gg m$. We also assume that the combined constraint array has full rank.

The dual of Problem (1) is given by:

$$\max \mathbf{ub} - \mathbf{vr},$$

$$\mathbf{uN} - \mathbf{vA} \leq \mathbf{c},$$

$$\mathbf{v} \geq \mathbf{0}. \tag{2}$$

Suppose that (1) is bounded and a feasible basis has been identified. Denote by $\mathscr{C}$ the set of column indices of (1) and by $\mathscr{B} \subseteq \mathscr{C}$ a subset of the columns of (1). Consider the restricted problem

$$V' = \min \mathbf{c}'\mathbf{x},$$

$$\mathbf{N}'\mathbf{x} = \mathbf{b},$$

$$\mathbf{A}'\mathbf{x} \leq \mathbf{r},$$

$$\mathbf{x} \geq \mathbf{0}, \tag{3}$$

where $\mathbf{N}'$ and $\mathbf{A}'$ are the submatrices of $\mathbf{N}$ and $\mathbf{A}$ obtained by dropping all columns not in $\mathscr{B}$ (the row dimension of (3) is the same as the original problem (1), the column dimension is $|\mathscr{B}|$) and listing the columns in order of their original indices. Let $\mathbf{c}'$ denote the cost vector consisting of the costs corresponding to columns in $\mathscr{B}$. We will denote the jth columns of $\mathbf{A}$ and $\mathbf{N}$ by $\mathbf{A}_j$ and $\mathbf{N}_j$, respectively. Suppose that the restricted Problem (3) has a basic feasible solution. It will be convenient, and cause no confusion, to speak of sets of columns and sets of column indices interchangeably. Let $\mathbf{x}'$ denote the optimal primal solution and $\mathbf{u}'$ and $\mathbf{v}'$ the (row vector) of dual variables corresponding to the constraints $\mathbf{N}'$ and $\mathbf{A}'$, respectively. Any basis for (3) will consist of columns of $\begin{bmatrix}\mathbf{N}' \\ \mathbf{A}'\end{bmatrix}$ and columns corresponding to the slack variables for the constraints $\mathbf{Ax} \leq \mathbf{r}$. In this paper we will use the term "basic columns" of (3) to mean the columns of (3) in the basis, excluding the columns added in the solution of (3) to represent slack variables. We will denote by $\mathscr{B}'$ the set columns of (3) in the optimal basis. To avoid proliferation of notation, here and for the remainder of this paper, we make the implicit assumption that the dimension of the generic variable $\mathbf{x}$ specified in the statement of an optimization problem is of appropriate dimension.

Consider the partial dual of Problem (1), given by:

$$V_{SP}(\mathbf{v}') = \min(\mathbf{c} + \mathbf{v}'\mathbf{A})\mathbf{x},$$

$$\mathbf{Nx} = \mathbf{b},$$

$$\mathbf{x} \geq \mathbf{0}. \tag{4}$$

It follows from the theory of Lagrangian relaxation (see Ahuja et al. 1993) that $(V_{SP}(\mathbf{v}) - \mathbf{vr}) \leq V$ for any $\mathbf{v} \geq 0$. Since (3) results from (1) by dropping columns, we are assured that $V \leq V'$. Problems (4) and (3) provide convenient upper and lower bounds on the optimal value. Let $\hat{x}$ denote an optimal primal solution to (4). Let $\hat{\mathcal{B}}$ denote the set of columns in $\mathcal{C}$ corresponding to nonzero entries in $\hat{x}$.

The Decomposition Pricing Procedure (DPP)

0. *Initialization*. Find a feasible basis for (1). Let $\mathcal{B}$ be the set of basic columns of (1) corresponding to the initial basis. Formulate the restricted problem and solve it. Let $\mathbf{x}'$, $\mathbf{u}'$, $\mathbf{v}'$ denote optimal primal and dual basic solutions to this restricted problem; let $\mathcal{B}'$ denote the set of indices of columns in the of the restricted problem in the optimal basis.

1. Solve the relaxed subproblem (4). Denote the indices of the columns corresponding to nonzero components of the optimal solution to (4) by $\hat{\mathcal{B}}$. If $\mathbf{u}'\mathbf{N}_j - \mathbf{v}'\mathbf{A}_j \leq c_j$ for all $j \in \hat{\mathcal{B}}$, then an optimal solution to (1) can be obtained from $\bar{\bar{x}}'_j = x'_j$ *for* $j \in \mathcal{B}'$, $\bar{\bar{x}}'_j = 0$ $j \in \mathcal{C} - \mathcal{B}'$. Otherwise, go to Step 2.

2. Construct a new restricted problem consisting of the columns $\mathcal{B} = \mathcal{B}' \cup \hat{\mathcal{B}}$ (the basic columns from the restricted problem together with the columns in $\hat{\mathcal{B}}$). Solve this new restricted problem. Replace $\mathbf{x}'$, $\mathbf{u}'$, $\mathbf{v}'$ with the primal and dual solutions to the new restricted problem and replace $\mathcal{B}'$ with the optimal basic columns of the new restricted problem. Go to Step 1.

We maintain at each iteration a feasible basic solution to the restricted Problem (3). In Step 2, the optimal basic solution to the current restricted problem can be used as an advanced starting basis for the next restricted problem. On the other hand, we do not assume that the subproblem is solved via the simplex method, or that a basic optimal solution to the subproblem is available. This allows the latitude to solve the subproblem by whatever algorithm is convenient. If Subproblem (4) is to be solved via the simplex method, the procedure can be modified to use the optimal basis of the restricted problem as an advanced starting solution to the subproblem. To see how this can be done, consider the restricted problem, as it is solved with the simplex method, augmented with slack variables.

$$V' = \min \mathbf{c}'\mathbf{x},$$

$$\mathbf{N}'\mathbf{x} = \mathbf{b},$$

$$\mathbf{A}'\mathbf{x} + \mathbf{I}\mathbf{s} = \mathbf{r},$$

$$\mathbf{x} \geq \mathbf{0}. \tag{5}$$

In the above formulation, I is an $h \times h$ identity matrix. Let B denote the optimal basis to (5). It follows from the nonsingularity of B that by a suitable permutation of the columns of B we can obtain a partition of B as follows:

$$\mathbf{B} = \begin{bmatrix} \mathbf{N}'_1 & \mathbf{N}'_2 & \mathbf{0} \\ \mathbf{A}'_1 & \mathbf{A}'_2 & \mathbf{E} \end{bmatrix},$$

where $\mathbf{N}'_1$ is an $m \times m$ matrix, and $\mathbf{E}$ is a matrix with h rows with a single 1 in each column corresponding to the slack variables in the basis. The nonsingularity of $\mathbf{B}$ guarantees the nonsingularity of $\mathbf{N}'_1$. The matrix $\mathbf{N}'_1$ will be used as an advanced starting basis for the subproblem in Step 1 of DPP. The vectors of dual variables $\mathbf{u}'$, and $\mathbf{v}'$ corresponding to $\mathbf{B}$ necessarily satisfy the relation,

$$\mathbf{u}'\mathbf{N}'_1 - \mathbf{v}'\mathbf{A}'_1 = \mathbf{c}'_1$$

or

$$\mathbf{u}'\mathbf{N}'_1 = \mathbf{c}'_1 + \mathbf{v}'\mathbf{A}'_1$$

where $\mathbf{c}'_1$ is partitioned consistently with $\mathbf{A}'_1$. Thus $\mathbf{N}'_1$ is a dual basis and $\mathbf{u}'$ the corresponding dual solution to (4). We can construct an initial primal feasible starting solution by taking the primal variables associated with $\begin{bmatrix} \mathbf{N}'_2 \\ \mathbf{A}'_2 \end{bmatrix}$, and treating them as nonbasic variables standing above 0. This amounts to the "pegged variable" technique described by Nazareth (1987). We use a variant of the convex cost simplex algorithm to implement this feature (in §3 of this paper). With this adjustment, the subproblem can be started from the optimal basis of the restricted problem in each iteration.

When starting with an advanced basis derived from the restricted problem, the first iteration of the simplex method in the subproblem will look for columns such

that $\mathbf{u}'\mathbf{N}_j > \mathbf{c}_j + \mathbf{v}'\mathbf{A}_j$, which is exactly what would happen if we were to allow the restricted problem to scan all of the columns of the original problem. The first column found in this way, when added to the restricted problem will bring about a gain (absent degeneracy). After the first pivot of the subproblem, this may no longer be true, since with each iteration the subproblems dual variables $\mathbf{u}$ will change. Hence, we have no immediate guarantee that columns encountered in later iterations of the simplex method applied to the subproblem will "price out" according to optimal dual solution of the restricted problem.

The purpose of this digression into the mechanics of the revised simplex method is to illuminate a refinement to the DPP procedure that we have found to be very effective in practice.

The Decomposition Pricing Procedure with Advanced Start (DPP-AS)

0. *Initialization*. Same as DPP

1.1. Identify a starting basis for Subproblem (4). If this basis is optimal in the subproblem, stop, then the solution to (3) is optimal for (1), otherwise go to Step 1.2.

1.2. Solve the relaxed Subproblem (4). Denote the indices of the columns corresponding to nonzero components of the optimal solution of (4) by $\hat{\mathcal{B}}$. Let i^* denote the column number of the first column to enter the basis in the solution of (4). If $\mathbf{u}'\mathbf{N}_j - \mathbf{v}'\mathbf{A}_j \leq c_j$ for all $\mathbf{j} \in \hat{\mathcal{B}}$, then an optimal solution to (1) can be obtained from $\bar{\bar{x}}'_j = x'_j$ for $j \in \mathcal{B}'$, $\bar{\bar{x}}'_j = 0$, $j \in \mathcal{C} - \mathcal{B}'$. Otherwise, go to Step 2.

2. Construct a new restricted problem consisting of the columns $\mathcal{B} = \mathcal{B}' \cup \hat{\mathcal{B}} \cup \{i^*\}$ (the basic columns from the restricted problem plus the columns in $\hat{\mathcal{B}}$, plus the first column to enter the basis in the subproblem). Solve this new restricted problem. Replace $\mathbf{x}'$, $\mathbf{u}'$, $\mathbf{v}'$ with the primal and dual solutions to the new restricted problem and replace $\mathcal{B}'$ with the optimal basic columns of the new restricted problem. Go to Step 1.

Convergence

For procedure DPP-AS convergence follows from the standard proof of convergence for the simplex method. The first step of the simplex method applied to the advanced basis obtained from the restricted problem is to find a column j such that $\mathbf{u}'\mathbf{N}_j > c_j + \mathbf{v}'\mathbf{A}_j$, which is exactly the requirement for an improving column in the restricted problem. If this column is added to the restricted problem, then (absent degeneracy) an improvement will be obtained in the next restricted problem, leading to an improved basic solution, and hence a new basis. Finite convergence follows as in the simplex algorithm.

Procedure DPP does not use an advanced basis to solve the subproblem, indeed it does not even require an optimal basic solution to the subproblem. Unless we explicitly add a pricing scan to Step 1 of the algorithm, and add at least one "priced out" (a column for which $\mathbf{u}'\mathbf{N}_j > c_j + \mathbf{v}'\mathbf{A}_j$) column from Step 1 to the restricted problem in Step 2, we cannot guarantee that every column will be inspected using the current duals from the restricted problem at Step 1. As it is specified, the procedure only looks at the columns corresponding to nonzero variables in an optimal solution to the subproblem. This makes establishing finite convergence slightly more complicated than for DPP-AS.

At each iteration of DPP, we add to the restricted problem the columns corresponding to the nonzero variables in an optimal solution to the subproblem. If one of these columns is found to "price out" in the restricted problem, then there will be an improvement and a change of basis in the restricted problem. If this does not happen, then all of the columns corresponding to nonzero variables in the optimal solution to the subproblem have positive reduced costs, and the current solution to the restricted problem is an optimal basic solution. This last fact is established in Proposition 1.

Proposition 1. *If for some basic optimal solution to the restricted Problem* (3) *none of the columns associated with nonzero variables in an optimal solution to the relaxed Subproblem* (4) *have negative reduced costs, then the optimum has been achieved*: *The current solution to the restricted problem, embedded in a* 0*-vector of appropriate dimension* (*as in Step* 1 *of the procedure*), *is optimal for the original primal problem*.

The general approach of proof is to establish that if all of the columns corresponding to positive variables

in the optimal solution to the subproblem have positive reduced costs, then the current solution to the restricted problem, embedded in a 0-vector of appropriate dimension, is an optimal solution to the subproblem, and then to use the properties of the Lagrangian dual to show that it must also be an optimal solution to the original problem. A detailed proof is given in the appendix to this paper. With this result in hand, convergence follows as in the usual simplex method from the fact that there are only a finite number of basic solutions to the original Problem (1).

In Step 2 of both the DPP and DPP-AS, we may decide to add more than just those columns that correspond to the nonzero components of the optimal solution to the subproblem. For example, if the subproblem is solved via the simplex method we could decide to add a collection of columns taken from the set of all columns that "priced out" during the solution of the subproblem. If we do not have access to a basic solution to the subproblem, or are not using the simplex method to solve the subproblem, then we might use other heuristics to choose which columns to add to the restricted problem. For example, one could add a complete or partial scan of the original columns using the current dual variables from the restricted problem to find a set of "promising" columns in addition to the columns revealed by the subproblem. The tradeoff is between the good that these columns might do in finding the optimal solution, versus the increases in computational cost brought about by a larger restricted Problem (3). The minimal requirement for convergence is that we add those columns corresponding to the positive variables in the optimal solution to the subproblem. In practice we have found that for the multicommodity flow problem it is helpful to include the first few pivot columns from the solution to the subproblem in addition to the strictly positive variables from the optimal solution to the subproblem.

The premise upon which our procedure is based is that the restricted problem is considerably smaller than the original multicommodity flow problem, and therefore may be solved much more quickly. The subproblem, on the other hand, possesses the same number of columns as the original problem but fewer (and well structured) constraints, rendering it easy to solve. We attempt to solve the original problem by iterating between a pair of easier problems.

Initial Basic Solution and Degeneracy

Our procedure requires an initial basic solution to the original linear programming problem. There are a variety of ways to do this. We discuss the specific choices we made in our implementation in §3.

Cycling can only happen when no strictly positive improvement is made in the restricted Problem (3). If we apply any of the standard anticycling rules (Bland 1977) to the restricted problem, and do not drop any column from the reformulation of the restricted problem in Step 2 until a strictly positive improvement has been encountered in the restricted problem, then the algorithm cannot cycle. Of course, this approach may cause the restricted problem to grow quite large.

Comparisons with Dantzig-Wolfe Decomposition

It is instructive to contrast our procedure with classical Dantzig-Wolfe decomposition. Let $\eta_1, \ldots, \eta_M$ denote the extreme points of the polytope

$$\mathcal{P} = \{\mathbf{x}: \mathbf{N}\mathbf{x} = \mathbf{b}, \mathbf{A}\mathbf{x} \leq \mathbf{r}, \mathbf{x} \geq \mathbf{0}\}.$$

Let $\xi_1, \ldots, \xi_{\bar{M}}$ denote the extreme points of the polytope

$$\mathcal{Q} = \{\mathbf{x}: \mathbf{N}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}.$$

Let us assume for simplicity that the polytope $\mathcal{Q}$ is bounded hence any point in $\mathcal{Q}$ can be written as a convex combination of the extreme points ξ_i. The Dantzig-Wolfe decomposition algorithm hinges on the fact that since $\mathcal{P} \subseteq \mathcal{Q}$ any point in $\mathcal{P}$ can be written as a convex combination of the extreme points of $\mathcal{Q}$.

At any given iteration we have available a subset of the extreme points of $\mathcal{Q}$, $\xi_1, \ldots, \xi_{\bar{M}'}$ where $\bar{M}' \leq \bar{M}$. The partial master problem,

$$\min \sum_{i=1}^{\bar{M}'} \mathbf{c}\xi_i z_i,$$

$$\sum_{i=1}^{\bar{M}'} \mathbf{A}\xi_i z_i \leq \mathbf{r},$$

$$\sum_{i=1}^{\bar{M}'} z_i = 1,$$

$$\mathbf{z} \geq \mathbf{0} \tag{6}$$

yields a multiplier vector $\mathbf{v}$ for the constraints $\sum_{i=1}^{\bar{M}'} \mathbf{A}\xi_i z_i \leq \mathbf{r}$. In each iteration of the decomposition algorithm, an extreme point is generated by solving a subproblem,

$$V_{SP}(\mathbf{v}) = \min(\mathbf{c} + \mathbf{vA})\mathbf{x},$$

$$\mathbf{Nx} = \mathbf{b},$$

$$\mathbf{x} \geq \mathbf{0}. \tag{7}$$

Linear programming duality assures us that when the extreme point identified by the subproblem does not improve the master problem, then the optimum has been obtained.

Note that the Dantzig-Wolfe Subproblem (7) is identical to the subproblem used in Step 2 of our pricing procedure (which is why we call our procedure "decomposition-based pricing"). However, instead of solving the partial master Problem (6), we solve a restricted version of the original Problem (3). We can put our restricted problem in the same terms as the corresponding Dantzig-Wolfe master problem. Let $\mathcal{B}_0$ denote the set of currently active columns (the columns of the most recently solved restricted problem). Let $\psi'_1, \ldots, \psi'_{M'}$ denote the extreme points of the polyhedron defined by the restricted problem,

$$\mathcal{P}' = \{\mathbf{x}\colon \mathbf{Nx} = \mathbf{b},\ \mathbf{Ax} \leq \mathbf{r},\ x_j = 0,\ j \notin \mathcal{B}_0,\ \mathbf{x} \geq \mathbf{0}\}.$$

Clearly $\mathcal{P}' \subseteq \mathcal{P}$ and any extreme point of the restricted problem is an extreme point of the original problem. The restricted problem chooses an extreme point from $\mathcal{P}'$. Step 2 of DPP assures that the optimal extreme point for the current restricted problem will remain in the set of extreme points of the next restricted problem. The optimal objective value of the restricted problem decreases on each iteration (ignoring degeneracy, of course). At each iteration the DPP adds to the restricted problem a set of extreme points *of the original problem* defined by the new columns and the current basis. In contrast, the Dantzig-Wolfe algorithm, as it is usually described, maintains a set of extreme points *to the relaxed subproblem* and adds only one new extreme point at each iteration (the algorithm can be modified to add several such extreme points as they become available during the solution to the subproblem).

The decomposition-based pricing procedure is likely to work well in the same situations in which the decomposition algorithm would work well; specifically, those problems whose constraint matrices can be partitioned into an "easy" subproblem and "complicating" side constraints. In principle, there is no reason why the partition of the problem could not be done on a completely ad hoc basis, thereby capitalizing on what ever structure is available in each problem instance. The efficacy of the method will depend on the extent to which the subproblems that are identified can be solved more rapidly than the complete problem.

3. Implementation

There are many aspects of our algorithm that will depend on the type of problem being solved. The choice of how to solve the subproblem, how to find an initial basic feasible solution, and how to solve the restricted problem will depend to some extent on the problem structure.

At a minimum, our scheme requires two solvers, a solver for the restricted problem and a solver for the subproblem. We used as our implementation platform the EMNET embedded network simplex solver to solve the restricted problem. The EMNET code uses a primal simplex algorithm with a partitioned basis factorization and maintains the problem representation as well as the working matrices in memory. The program has been specialized to solve networks with side constraints and side variables. Our experience with this code has been quite good; McBride and Mamer (1997a) give comparisons between EMNET, interior point methods, standard simplex method, and decomposition algorithms, as well as a description of some of the enhancements contained in EMNET. The subproblems, on the other hand, were solved with special purpose algorithms. In the case of the message routing problem, a standard shortest path algorithm was used and for the multicommodity flow problem a version of the GENNET algorithm of Brown and

McBride (1984), implemented within EMNET, was employed.

Since the apparent efficacy of our procedure improves as the time taken to solve the restricted problem increases relative to the subproblem (this is so since our procedure trades off additional pivots in the subproblem against pivots on the restricted problem), we felt that using EMNET as the implementation platform offered a conservative measure of the impact of our proposed procedure. Since EMNET, applied directly, can solve the multicommodity flow problem quickly (with considerably better performance than either general purpose simplex and interior point codes, and certainly on a par with most specialized codes), comparing the solution time using our procedure with the solution time using EMNET alone would give a fair indication of the efficacy of our proposal. Our intent is to compare our procedure to what amounts to the "state of the art" in multicommodity flow algorithms. Having said this, we hasten to remind the reader that the specification of our procedure is quite general and it could easily be implemented using any linear programming solver.

The default pricing procedure for EMNET is a "candidate queue" pricing mechanism. Columns that are to be considered for entry into the basis are placed in a queue, called the "candidate queue." On each pass through the pricing step of the simplex algorithm, the reduced costs for some of the columns in the candidate queue are calculated, columns that no longer have negative reduced cost are removed from the queue, and a pivot column is chosen. When the number of columns in the queue with negative reduced costs falls below some critical threshold, then the queue is refreshed (a pricing scan is done to identify new columns for the candidate queue), and the cycle repeats. There are many possible approaches to deciding which columns should be placed in the candidate queue list. One heuristic, commonly used in network simplex implementations, is to compute the reduced costs for the arcs (columns) entering a collection of nodes, and to place in the candidate queue those yielding negative reduced costs. There are many possible approaches to choosing the entering basic variable. Nazareth (1987, Ch. 7) offers a lucid overview of the most commonly used techniques.

For our implementation of the decomposition-based pricing procedure, we co-opted the pricing strategy of EMNET to implement the solution to the restricted problem. After each solution of the subproblem, we add to the candidate queue those columns corresponding to basic columns from the solution to the restricted problem in the previous iteration, and those columns corresponding to positive variables in the optimal solution to the subproblem in the current iteration. EMNET then solves the original problem, but without refreshing the candidate queue, until a scan of the candidate queue reveals that there are no improving columns to be found. We neither add columns to nor drop columns from the candidate queue during this solution process. This effectively restricts EMNET to pivoting on those columns included in the current restricted problem. Building the restricted problem in this fashion, by maintaining the original problem data structure, but restricting the columns eligible for pricing, has the disadvantage of slightly increasing the computational effort required in the basis update after each pivot (because the entire problem is maintained in memory), but offers the advantage of being able to switch from one restricted problem to the next and from the restricted problem to the subproblem very quickly.

As mentioned in §2, when constructing the restricted problem we need not limit ourselves to only those columns corresponding to nonzero variables in the optimal solution to the subproblem. After experimenting with several different heuristics for choosing columns to include in the restricted problem, we found that the best performance was obtained by adding the first few columns to enter the basis during the solution of the subproblem (under DPP-AS the very first column to enter the basis of the subproblem is guaranteed to have a negative reduced cost in the restricted problem). The first few entering basic columns, whether or not they remained basic in the subproblem, seemed to help in the restricted problem. For some of the test problems we added several hundred additional columns to the restricted problem in this fashion. We experimented briefly with not

solving the subproblem to optimality on each iteration, but rebuilding the restricted problem once a significant gain in the subproblem had been made. This variation on our procedure did not improve our solution times for our sample of test problems. One can easily imagine many variations on our basic procedure, and we have had time enough to explore only a few of them. We have not had the opportunity to fine-tune our procedure. Indeed, it was a surprise to us that the procedure worked so well from our original implementation. With some additional time spent tuning the procedure and testing more sophisticated heuristics for adding and dropping columns from the restricted problem, we imagine that it would be possible to improve on the results reported in §4 of this paper.

Another important implementation decision for the DPP-AS procedure is how to create an initial starting basis for the subproblem from the optimal basis of the restricted problem. In §2 we showed that it is, in principle, possible to recover a starting basis to the subproblem from the restricted problem; however, that basis will have nonbasic variables that are not at their bounds (since the subproblem is of smaller row dimension than the restricted problem). One approach is to use the "pegged-variable" technique mentioned in §2. We chose instead to make use of the convex cost mechanism built into EMNET. The convex cost simplex algorithm allows for piecewise linear convex costs, each cost is specified by a series of cost rates and intervals, the rates are nondecreasing and the intervals are contiguous. In a basic solution, the nonbasic variables may be either at their lower bounds, their upper bounds, or at a cost "breakpoint." We make use of this feature by creating two segments, with identical costs, which meet at the initial value of the nonbasic variable. The problem is then solved from the initial basis using the convex cost simplex method. McBride and Mamer (1997a) offer a more complete explanation of this technique.

Starting the problem poses another major implementation decision. There are many ways to produce the initial basic solutions; one possibility is the classical Phase I/Phase II algorithm. We used two approaches. Starting from the un-side constrained network solution we either used the allocation heuristic proposed by McBride and Mamer (1997a), or we used a variant of the "Big M" technique: Artificial variables for infeasible side constraints constraints were assigned large costs so as to drive them out of the basis.

4. Numerical Experiments

We consider two sets of problems. The first set of problem instances is derived from the message routing problem. The problem instances solved here were generated from a program made available to us by Chris Hane, and problems generated in this fashion have been used by other authors to test algorithms for solving the message routing problem (see Barnhart et al. 1994). Our experience with the specialization of the EMNET code for this problem are reported in detail in McBride and Mamer (1997b). We reproduce some of the results of that analysis here to highlight the effect of the decomposition-based pricing technique. The second set of problems were standard directed multicommodity flow problems, some of which are available from the NETLIB archive, and from generators available on the Internet. Our computing environment consists of a workstation based on a 500 MHz DEC Alpha CPU running Microsoft Windows NT 4.0 and a PC with a 300 MHz Pentium II CPU running Windows 95.

The message routing problem can be stated (see McBride and Mamer 1997b) as:

$$\min \sum_{k=1}^{K} \sum_{(i,j)\in E} c_{ij}^k |x_{ij}^k|,$$

$$\sum_{\{j:(i,j)\in E\}} x_{ij}^k - \sum_{\{j:(j,i)\in E\}} x_{ji}^k = b_i^k, \quad i \in N,\ k = 1, \ldots, K,$$

$$\sum_k |x_{ij}^k| \le u_{ij}, \quad \forall\ (i, j) \in E. \tag{8}$$

In this formulation N denotes a set of nodes, $E \subseteq N \times N$ the set of arcs, and $k = 1, \ldots, K$ an enumeration of the messages to be sent. x_{ij}^k denotes the flow of message k from node i to node j. Positive values represent flows from i to j and negative values flows from j to i. Each message is s^k units of flow between its origin and its destination (thus, $b_i^k = s^k$ if node i is the

Table 1 Message Routing Test Problems

Nodes/ Commodity	Arcs/Commodity	# Comm.	Total Nodes	Total Arcs	Number of Side Constraints
30	86	90	2,700	7,740	86
40	121	175	7,000	21,175	121
50	201	302	15,100	60,702	201
60	268	447	26,820	119,796	268
70	368	622	43,540	228,896	368
80	485	784	72,720	380,240	485
90	595	1,006	90,540	598,570	595
100	737	1,227	122,700	904,299	737

origin node for message k, $b_i^k = -s^k$ if node i is the destination node for message k, and $b_i^k = 0$ otherwise). Since arc capacity is used up by flow in either direction, the joint capacitation (representing the available bandwidth) is a function of the *absolute value* of the flow variables.

As documented in McBride and Mamer (1997b) we approached this problem by generalizing the "convex cost" approach of Fourer (1985, 1988) (specialized to network problems by Murthy and Helgason 1993) to handle absolute values in the objective function and absolute values in the constraints. With this modification, EMNET was able to solve the problem without reformulating it as a directed multicommodity flow problem. We then applied the decomposition-based pricing heuristic to this modification of EMNET. The subproblem consisted of the original problem with the side constraints dualized. Since there were no arc capacities, and each message had a single source and a single sink (origin and destination), the subproblems were solved as undirected shortest path problems. The arcs in the shortest path correspond to the nonzero columns in the optimal basis to the subproblems. Table 1 gives the sizes of the problems solved. Table 2 documents the impact of the decomposition-based pricing algorithm on solution time and number of simplex pivots. These problems were solved using the 500 MHz Alpha processor.

Table 2 shows clearly the decrease in both solution time and number of pivots (on the restricted problem) brought about by our procedure. The column labeled "Pivots: Direct Solution" and "Solution Time: Direct Solution" give the number of pivots and solution time for EMNET to solve the problem using only the modification needed to handle undirected arcs (but a standard pricing scheme). The columns labeled "Pivots: Decomposition Pricing" and "Solution Time: Decomposition Pricing" give the number of pivots and solution time for exactly the same algorithm, using the decomposition pricing procedure. In the case of the message routing problem, since the subproblems were solved using a shortest path algorithm, we did not start them using an advanced basis extracted from the restricted problem.

Our second set of problems follows the pattern of the standard directed multicommodity flow formulation:

$$\min \sum_{k=1}^{K} \sum_{(i,j)\in E} c_{ij}^k x_{ij}^k,$$

Table 2 Pivot Counts and Solution Times (in Seconds) Message Routing Test Problems

Nodes/ Commodity	Pivots: Direct Solution	Pivots: Decomposition Pricing	Solution Time: Direct Solution 500 MHz DEC Alpha	Solution Time: Decomposition Pricing 500 MHz. DEC Alpha
30	3,506	304	0.39	0.18
40	9,176	577	1.14	0.29
50	24,611	1,070	3.19	1.00
60	49,720	1,740	9.74	1.91
70	90,706	2,911	26.33	4.13
80	151,824	4,002	69.31	8.20
90	223,250	4,912	205.21	31.31
100	370,932	7,839	563.35	55.42

$$\sum_{\{j:(i,j)\in E\}} x_{ij}^k - \sum_{\{j:(j,i)\in E\}} x_{ji}^k = b_i^k, \quad i \in N,\ k = 1, \ldots, K, \tag{9}$$

$$\sum_k a_{ij}^k x_{ij}^k \le r_{ij}, \quad \forall\ (i, j) \in S, \tag{10}$$

$$l_{ij}^k \le x_{ij}^k \le u_{ij}^k, \quad \forall\ (i, j) \in Ek = 1, \ldots, K. \tag{11}$$

In (10), $S \subseteq E$ represents the set of side-constrained arcs, and l_{ij}^k and u_{ij}^k represent upper and lower bounds on the flow of commodity k on arc (i, j).

The general structure of these problems follows that given in (1), where **N** represents the network constraints (9), and **A** the complicating side constraints (often called "joint capacity constraints") (10). Removing the joint capacity constraints (10) allows the problem to be solved as a collection of separate single commodity flow problems.

For these problems we were able to make use of the partitioned basis representation maintained by EMNET and easily obtained a starting basis for the subproblem from the optimal basis of the restricted problem. This allowed us to use the DPP-AS procedure as described in §2.

Table 3 identifies the PDS problems we solved, and gives their sizes. Table 4 identifies[1] the Mnetgen[2] (mnet), dimacs2pprn[3] (dmx), and JLF (the Chen5 and 15.term.o problems were selected from the JLF family of problems) problems and gives their sizes. Also included with each generator are computational results comparing several solution approaches and algorithms.

Table 5 gives the solution time for each PDS problem using EMNET without the pricing procedure and with the pricing procedure. We note that the times given for EMNET improve significantly on those reported in McBride and Mamer (1997a); this reflects major refinements in the code as well as a faster CPU (these times were obtained using the 300 MHz Pentium II CPU). We record in the column entitled "restricted problem pivots" the number of pivots of the restricted problem needed in the solution process. We did not count the number of pivots used in solving the subproblems since the network simplex algorithm pivots many times faster than does EMNET when working with side constraints. The appropriate comparison of the work done by the simplex algorithm is to compare the restricted problem pivots with the number of pivots needed to solve the problem using EMNET without the pricing procedure. It is immediately apparent from Table 5 that the pricing procedure dramatically reduces the number of pivots needed to solve the problem. For the PDS family of problems the reduction in solution time ranged from −0.13% and 47%, greater decreases in solution time seem to be associated with increased problem size.

Table 3 **PDS Multicommodity Flow Problems Solved (11 Commodities)**

Problem	Nodes	Side Const.	Total Rows	Total Columns
PDS-10	15,389	1,169	16,558	49,932
PDS-20	31,427	2,447	33,874	108,175
PDS-30	46,453	3,491	49,944	158,489
PDS-40	62,172	4,672	66,844	217,531
PDS-50	77,341	5,719	83,060	275,814
PDS-60	92,653	6,778	99,431	336,421
PDS-70	107,250	7,694	114,944	390,005
PDS-80	120,879	8,302	129,181	434,580
PDS-85	127,556	8,557	136,113	455,488

The PDS family of problems has been used by many authors to test multicommodity flow algorithms. McBride and Mamer (1997a) review in detail the results and the algorithms used to solve these problems. Table 6 helps place our results into perspective with other published results. In this table, Z1 and Z8 stand for the results reported in Pinar and Zenios (1992) using a decomposition algorithm (with one and eight processors respectively), M stands for Marsten et al. (1990) using an interior point algorithm, S & M stands for Schultz and Meyer (1991) using a decomposition algorithm, KBX stands for Carolan et al. (1990) using an interior point algorithm on the KORBX

[1] Generators for these families of problems can be found at the website maintained by Antonio Frangioni, http://www.di.unipi.it/di/groups/optimize/Data/MMCF.html.

[2] Problems corresponding to lines 205, . . . , 216 in the computational result files in the "results" subdirectory (this subdirectory is created when the generator is installed) associated with this generator were selected.

[3] Problems corresponding to lines 41, . . . , 48, in the file "mmcfb" of the results subdirectory associated with RmfGen generator.

Table 4 Mnetgen, JLF, and dimacs2ppprn Multicommodity Flow Problems Solved

Problem	Nodes	Side Const.	Total Rows	Total Columns	Commodities
mnet205	65,536	315	65,851	185,319	256
mnet206	65,536	316	65,852	185,338	256
mnet207	65,536	314	65,850	185,339	256
mnet208	65,536	673	66,209	185,770	256
mnet209	65,536	685	66,221	185,884	256
mnet210	65,536	682	66,218	185,709	256
mnet211	65,536	831	66,367	361,031	256
mnet212	65,536	880	66,416	361,062	256
mnet213	65,536	825	66,361	360,873	256
mnet214	65,536	1,769	67,305	361,810	256
mnet215	65,536	1,772	67,308	361,824	256
mnet216	65,536	1,802	67,338	361,866	256
dmx41	2,048	2,240	4,288	11,200	4
dmx42	8,192	2,240	10,432	38,080	16
dmx43	32,768	2,240	35,008	145,600	64
dmx44	131,072	2,240	133,312	575,680	256
dmx45	4,096	4,544	8,640	22,720	4
dmx46	16,384	4,544	20,928	77,248	16
dmx47	65,535	4,544	70,080	295,360	64
dmx48	262,144	4,544	266,688	1,167,808	256
chen5	650	242	892	5,932	10
15.term.0	4,275	202	4,477	7,345	15

Table 5 Comparison of Pricing Strategies with PDS Multicommodity Flow Problems

	Normal Pricing		Decomposition Pricing		
Problem	Equivalent Pivots with Side Constraints	Solution Time (300 MHz)	Restricted Problem Pivots	Solution Time (300 MHz)	Percent Improvement
PDS-10	2,122	5.17	1,535	5.82	−0.13
PDS-20	13,151	60.64	6,666	46.68	23
PDS-30	26,139	159.50	14,620	146.92	8
PDS-40	53,981	495.10	32,877	416.06	15
PDS-50	77,104	751.11	39,786	590.29	20
PDS-60	126,673	1,567.46	56,568	940.16	40
PDS-70	113,036	1,629.70	54,274	992.83	39
PDS-80	166,030	2,933.90	70,405	1,557.19	47
PDS-85	160,996	2,680.92	69,622	1,442.51	46

system, and L & R stands for Lustig and Rothberg (1996) who used a parallelized version of the CPLEX interior point algorithm. The final column, labeled EMNET, offers our times for these problems. Since the times recorded in each column of Table 6 were obtained on a different processor, they are not directly comparable. At best they give some general indication of the efficiency of the algorithms compared.

Table 7 gives the results for the selected problems from the Mnetgen, dimacs2pprn, and JLF families of problems. In Table 7 an asterisk (*) means that the code was not able to solve the problem, usually due to

Table 6 Comparison Timings in Seconds

Problem	Z1	Z8	M	S & M	KBX	L & R	EMNET
PDS-10	408	96	1,521	999	11,880		5.82
PDS-20	1,946	740	15,972	3,043	63,720	409	46.68
PDS-30	7,504	2,566		6,480			146.92
PDS-40				10,440			416.06
PDS-50				19,800			590.29
PDS-60				24,900			940.16
PDS-70				33,840			992.83
PDS-80							1,557.19
PDS-85							1,442.51

Z1:	Pinar and Zenios (1992), Cray-YMP, one processor. Approx. 5 digits of accuracy.
Z8:	(1992), Cray-YMP, eight processors. Approx. 5 digits of accuracy. Time reported is elapsed time.
M:	Marsten et al. (1990), Cray-YMP, one processor.
S & M:	Schultz and Meyer (1991), Sequent 11 processors, stopped after 50 iterations. Time reported is elapsed time. Three digits of accuracy for PDS-30 to PDS-50, two digits of accuracy for PDS-60 and PDS-70.
KBX:	Carolan et al. (1990), KORBX, 14 processors. Time reported is elapsed time.
L & R:	Lustig and Rothberg (1996), Silicon Graphics Power Challenge with 16 processors.
EMNET	Solved with EMNET and the Decomposition Based pricing heuristic on a PC with a 300 MHz Pentium II processor.

insufficient memory. Cplex is the well-known commercial code. PPRN is a pure network primal partitioning simplex implementation (see Castro 1994 and Castro and Nabona 1996). IPM is a specialized Interior Point code from Castro and Nabona (1996). The column EMNET(decom) gives the times for EMNET with the DPP-AS pricing procedure, and the column EMNET(Cand.que) gives the times for EMNET applied without the pricing procedure using its default candidate que pricing strategy. The Cplex, PPRN, and IPM results were obtained on a HP 9000/712-80 workstation by Antonio Frangioni.[4] Frangioni (1997) also provides some excellent solution times for MMCFB, a dual bundle algorithm. For the current set of test results, the solutions obtained may be slightly primal infeasible (although typically very nearly feasible solutions are found) and the amount of infeasibility may vary from problem to problem. In view of this fact, we decided that it was not appropriate to compare the MMCFB results directly with the results of EMNET, CPLEX, PPRN, IPM. The 300 MHz Pentium II CPU is believed to be about four times faster than the HP workstation. The results for this second set of test problem were considerably more variable, ranging from a low of −126% and a high of +84%. Only on the rather difficult problems: mnet214, mnet215, and mnet216, did the DPP-AS procedure make a clear improvement in solution time. It averaged an improvement of eight times for mnet214 and mnet215 and 12 times for the mnet216.

[4] See the website http://www.di.unipi.it/di/groups/optimize/Data/MMCF.html.

5. Conclusion

We think of our approach as a pricing strategy implemented within the framework of the simplex method, rather than a decomposition algorithm. We use the representation of the problem available from the revised simplex method to structure and organize the problem. The solution to the subproblem is used to make enhanced pricing choices within the simplex method. The mathematical description of the procedure is written in terms of a sequence of restricted problems, but in practice, the restricted problems are

Table 7 Timing Comparisons with Mnetgen, JLF, and dimacs2ppprn Multicommodity Flow Problems

Problem	Cplex	PPRN	IPM	EMNET(decom)	EMNET(Cand. que)
mnet205	5,982.5	22,093.9	3,829.53	109.96	78.44
mnet206	4,704.42	18,069.7	3,007.22	95.19	64.10
mnet207	1,832.34	8,289.06	3,202.78	80.58	50.32
mnet208	16,430.9	35,131.9	7,085.33	205.26	152.04
mnet209	20,976.0	33,614.6	7,200.41	235.63	207.29
mnet210	11,262.0	14,129.7	5,941.38	140.66	102.17
mnet211	*	39,383.2	16,210.7	286.33	299.84
mnet212	*	54,120.0	15,930.3	400.29	451.98
mnet213	*	45,213.7	21,352.7	390.30	460.72
mnet214	>300,000.	>600,000.	41,257.1	2,936.10	17,816.24
mnet215	>300,000.	>600,000.	37,742.6	2,390.25	14,594.42
mnet216	>300,000.	>600,000.	44,608.2	2,760.77	33,381.30
dmx41	6.41	12.08	35.58	1.27	0.94
dmx42	43.3	275.89	166.14	2.63	1.87
dmx43	261.62	4,492.95	754.25	18.89	8.35
dmx44	*	81,419.10	4,236.76	186.15	128.85
dmx45	14.02	34.84	101.49	2.31	1.92
dmx46	74.82	1,286.26	514.74	6.93	5.00
dmx47	978.31	22,511.3	3,411.42	56.14	42.08
dmx48	*	>400,000	*	616.82	448.19
chen5	48.74	48.12	5.24	5.87	4.11
15.term.0	2.72	10.23	30.35	0.93	0.55

implemented via the candidate queue mechanism of (or, more generally, the pricing mechanism) of the EMNET algorithm. The basic idea is quite simple and could, in principle, be implemented with any linear programming solver (extreme point or interior point). We chose the EMNET code for our test case for two reasons. First of all, it has a demonstrated track record on multicommodity flow problems and therefore offers a stringent test of our ideas. Secondly, it offers a stable platform with readily available source code.

This work lies within a larger stream of work (McBride and Mamer 1997a, 1997b) in which we attempt to use ideas pioneered in the context of decomposition algorithms to enhance the performance of the simplex method. This work flows from two observations. The first is that with the advent of cheap memory and very fast inexpensive CPUs it is possible to execute the revised simplex method on problems that could previously only have been attempted using a decomposition algorithm. This ability is further enhanced (in our view) by the use of partitioned basis factorization techniques, which offer a very terse representation of the simplex basis. Naturally, our greatest success in this endeavor has occurred in the solution of highly structured problems (such as the multicommodity flow problem). The second observation is that for very large problem instances the simplex method suffers from many of the same problems as the decomposition algorithms, but also offers a level of numerical robustness not available in the decomposition algorithms. Our goal was to try to blend the best of both types of algorithms. The decomposition pricing scheme uses a decomposition-style subproblem to offer guidance in pivot selection so as to overcome the simplex algorithm's tendency to "bog down" (to make many small, or nonimproving, pivots) on very large problems.

The decomposition pricing procedure may also ameliorate the effects of degeneracy. In a typical multicommodity flow problem, many degenerate pivots are made while trying to identify alternative improving paths through the flow network. Such

pivots are time consuming even for a specialized algorithm like EMNET. Under our decomposition pricing scheme, the subproblem optimal solution identifies a complete tentative improving path that is passed to the restricted problem. The small size of the restricted problem offers it many fewer pivot choices and allows it to pivot more quickly in comparison to trying to solve the original problem directly with the EMNET solver.

Another, perhaps less apparent, advantage of our procedure is that it produces a primal basic feasible solution to the problem. Decomposition techniques often produce only approximately feasible solutions and interior point algorithms also produce optimal but nonextremal solutions. The basic feasible solution has much to recommend it. It offers a good starting point for integer optimization to accommodate, for example, such issues as sole-source requirements or fixed charges within the flow network.

Appendix

For ease of reference, recall our original problem:

$$V = \min \mathbf{cx},$$
$$\mathbf{Nx} = \mathbf{b},$$
$$\mathbf{Ax} \le \mathbf{r},$$
$$\mathbf{x} \ge \mathbf{0}. \tag{1}$$

Its dual is given by:

$$\max \mathbf{ub} - \mathbf{vr},$$
$$\mathbf{uN} - \mathbf{vA} \le \mathbf{c},$$
$$\mathbf{v} \ge \mathbf{0}. \tag{2}$$

The relaxed subproblem is:

$$V_{SP}(\mathbf{v}) = \min(\mathbf{c} + \mathbf{vA})\mathbf{x},$$
$$\mathbf{Nx} = \mathbf{b},$$
$$\mathbf{x} \ge \mathbf{0}. \tag{4}$$

We require two lemmas establishing the relationships between the optimal solutions to (1) and (4). Lemmas 1 and 2 are contained in Ahuja et al. (1993 pp. 605–607). We reproduce them here for continuity.

Lemma 1. *If* $\mathbf{x}^*$ *and* $(\mathbf{u}^*, \mathbf{v}^*)$ *are optimal primal, and dual solutions to* (1), *respectively, then* $\mathbf{x}^*$ *is an optimal solution to* (4) *when* $\mathbf{v} = \mathbf{v}^*$.

Proof. By construction, $\mathbf{x}^*$ is a feasible solution to (4). Since $\mathbf{u}^*$ and $\mathbf{v}^*$ are feasible solutions to (2), we have $\mathbf{u}^*N \le \mathbf{c} + \mathbf{v}^*\mathbf{A}$, hence $\mathbf{u}^*$ is a feasible solution to the dual of (4). It remains to show that their optimal values are equal. Since $\mathbf{x}^*$ and $(\mathbf{u}^*, \mathbf{v}^*)$ are primal and dual optimal for (1), we must have $\mathbf{cx}^* = \mathbf{u}^*\mathbf{b} - \mathbf{v}^*\mathbf{r}$. Moreover, $\mathbf{x}^*$ and $\mathbf{v}^*$ satisfy complementary slackness: $\mathbf{v}^*(\mathbf{Ax}^* - \mathbf{r}) = 0$. Thus,

$$\mathbf{u}^*\mathbf{b} = \mathbf{cx}^* + \mathbf{v}^*\mathbf{r} = \mathbf{cx}^* + \mathbf{v}^*\mathbf{Ax}^* = (\mathbf{c} + \mathbf{v}^*\mathbf{A})\mathbf{x}^*,$$

which establishes that the primal and dual values are equal. □

Lemma 2. *If* $\mathbf{x}^*$ *is an optimal solution to* (4) *with* $\mathbf{v} = \mathbf{v}^*$, *and is feasible for* (1), *and* $\mathbf{x}^*$ *and* $\mathbf{v}^*$ *satisfy complimentary slackness, i.e.* $\mathbf{v}^*(\mathbf{Ax}^* - \mathbf{r}) = 0$, *then* $\mathbf{x}^*$ *is an optimal solution to* (1).

Proof. For any feasible solution $\mathbf{x}$, to (1), and nonnegative $\mathbf{v}$, $\mathbf{cx} + \mathbf{v}(\mathbf{Ax} - \mathbf{r}) \le \mathbf{cx}$. Thus $V_{SP}(\mathbf{v}) - \mathbf{vr} \le V$. Since $\mathbf{x}$ is a feasible solution to (1), $\mathbf{cx} \ge V$. Complementary slackness ensures that $\mathbf{v}(\mathbf{Ax} - \mathbf{r}) = 0$. Putting these facts together yields:

$$V \le \mathbf{cx} = \mathbf{cx} + \mathbf{v}(\mathbf{Ax} - \mathbf{r}) = (\mathbf{c} + \mathbf{vA})\mathbf{x} - \mathbf{vr} = V_{SP}(\mathbf{v}) - \mathbf{vr} \le V.$$

This establishes the optimality of $\mathbf{x}$ in (1). □

With these results in hand we can now prove the basic optimality condition for our procedure.

Proposition 1. *If for some basic optimal solution to the restricted problem* (3) $(\mathbf{x}')$, $(\mathbf{u}', \mathbf{v}')$, *none of the columns associated with nonzero variables in an optimal solution to the relaxed subproblem* (4) *have negative reduced costs, then the optimum has been achieved: the current solution to the restricted problem, embedded in an* 0*-vector of appropriate dimension* (*as in Step* 1 *of the procedure*) *is optimal for the original primal problem.*

Proof. Recall that $\hat{\mathbf{x}}$ denotes the optimal solution to the subproblem (4) (in Step 1 of the algorithm). By assumption, for each j such that $\hat{x}_j \ne 0$, we have $c_j - \mathbf{u}'\mathbf{N}_j + \mathbf{v}'\mathbf{A}_j \ge 0$. Thus, none of the columns corresponding to nonzero variables in the optimal solution to (4) "price out" in (3). Formulate a new restricted problem with column index set equal to the basic columns from the restricted problem plus all of the columns from the subproblem corresponding to nonzero variables in the optimal solution $\mathcal{B}'' = \mathcal{B}' \cup \hat{\mathcal{B}}$:

$$V'' = \min \mathbf{c}''\mathbf{x},$$
$$\mathbf{N}''\mathbf{x} = \mathbf{b},$$
$$\mathbf{A}''\mathbf{x} \le \mathbf{r},$$
$$\mathbf{x} \ge \mathbf{0}. \tag{12}$$

Define a primal solution to (12):

$$\bar{x}'_j = \begin{bmatrix} x'_j, & j \in \mathcal{B}', \\ 0, & j \notin \mathcal{B}', \quad \text{and} \quad j \in \mathcal{B}'', \end{bmatrix}$$

and define a primal solution to (1):

$$\bar{\bar{x}}'_j = \begin{cases} x'_j, & j \in \mathcal{B}', \\ 0 & j \notin \mathcal{B}' \quad \text{and} \quad j \in \mathcal{C}. \end{cases}$$

The vectors $\mathbf{x}'$, $\bar{\mathbf{x}}'$, and $\bar{\bar{\mathbf{x}}}'$ agree on their components corresponding to the columns $\mathcal{B}'$ (the basic components of $\mathbf{x}'$); their dimensions have been adjusted to be conformable with problems (3), (12), and (1), respectively. We will show that $\bar{\bar{\mathbf{x}}}'$ is an optimal solution to (1). By construction, we have:

$$\mathbf{c}'\mathbf{x}' = \mathbf{c}''\bar{\mathbf{x}}' = \mathbf{c}\bar{\bar{\mathbf{x}}}', \tag{13}$$

$$\mathbf{A}'\mathbf{x}' = \mathbf{A}''\bar{\mathbf{x}}' = \mathbf{A}\bar{\bar{\mathbf{x}}}' \leq \mathbf{r}, \tag{14}$$

and

$$\mathbf{N}'\mathbf{x}' = \mathbf{N}''\bar{\mathbf{x}}' = \mathbf{N}\bar{\bar{\mathbf{x}}}' = \mathbf{b}. \tag{15}$$

Note that the the embedding preserves complementary slackness, i.e., $\mathbf{v}'(\mathbf{A}'\mathbf{x}' - \mathbf{r}) = \mathbf{v}'(\mathbf{A}''\bar{\mathbf{x}}' - \mathbf{r}) = \mathbf{v}'(\mathbf{A}\bar{\bar{\mathbf{x}}}' - \mathbf{r}) = 0$. It also immediate that $\bar{\mathbf{x}}'$ is an optimal solution to (12) and $(\mathbf{u}', \mathbf{v}')$ is an optimal dual solution to (12). To see this, note that since $(\mathbf{u}', \mathbf{v}')$ is an optimal dual solution for (3), $\mathbf{u}'\mathbf{N}' + \mathbf{v}'\mathbf{A}' \leq \mathbf{c}'$. By assumption, for any $j \in \mathcal{B}$, $\mathbf{u}'\mathbf{N}_j + \mathbf{v}'\mathbf{A}_j \leq c_j$, hence $(\mathbf{u}', \mathbf{v}')$ are dual feasible for (12). By (14) and (15), $\bar{\mathbf{x}}'$ is a feasible solution to (12). However, $\mathbf{c}''\bar{\mathbf{x}}' = \mathbf{c}'\mathbf{x}' = \mathbf{u}'\mathbf{b} - \mathbf{v}'\mathbf{r}$ where the first equality follows from (13) and the second from the fact that $\mathbf{x}'$ and $(\mathbf{u}', \mathbf{v}')$ are an optimal primal-dual pair for (3). The optimality of $\bar{\mathbf{x}}'$ for (12) follows from strong duality.

The remainder of the proof entails showing that $\bar{\bar{\mathbf{x}}}'$ is an optimal solution to (4). Lemma 2 will then imply that it is an optimal solution to (1). To this end, define the partial dual of (12):

$$V''_{SP}(\mathbf{v}') = \min(\mathbf{c}'' + \mathbf{v}'\mathbf{A}'')\mathbf{x},$$

$$\mathbf{N}''\mathbf{x} = \mathbf{b},$$

$$\mathbf{x} \geq \mathbf{0}. \tag{16}$$

Since (16) results from (4) by removing only the columns associated with the zero components of an optimal solution, the optimal values of the two optimization problems are equal. We establish this formally. Since (16) results from (4) by removing columns, $V_{SP}(\mathbf{v}') \leq V''_{SP}(\mathbf{v}')$.

Define $\hat{x}''_i = \hat{x}_i i \in \mathcal{B}''$; since all of the columns corresponding to nonzero columns of $\hat{\mathbf{x}}$ are in $\mathcal{B}''$, we have $\mathbf{N}''\hat{\mathbf{x}}'' = \mathbf{b}$ and $(\mathbf{c}'' + \mathbf{v}'\mathbf{A}'')\hat{\mathbf{x}}'' = (\mathbf{c} + \mathbf{v}'\mathbf{A})\hat{\mathbf{x}}$. Thus $\hat{\mathbf{x}}'$ is a feasible solution to (16), and yields an objective function value equal to the optimal value of (4). Thus

$$V''_{SP}(\mathbf{v}') \leq (\mathbf{c}'' + \mathbf{v}'\mathbf{A}'')\hat{\mathbf{x}}'' = (\mathbf{c} + \mathbf{v}'\mathbf{A})\hat{\mathbf{x}} = V_{SP}(\mathbf{v}'). \tag{17}$$

Hence $V_{SP}(\mathbf{v}') = V''_{SP}(\mathbf{v}')$.

We established above that $\bar{\mathbf{x}}'$ and $(\mathbf{u}', \mathbf{v}')$ are optimal primal and dual solutions to (12). Lemma 1 applied to (12) and (16) implies that $\bar{\mathbf{x}}'$ is an optimal solution to (16). By (15) $\bar{\bar{\mathbf{x}}}'$ is a feasible solution to (4). Coupling this last fact with (13), (15), and (17) yields

$$(\mathbf{c} + \mathbf{v}'\mathbf{A})\bar{\bar{\mathbf{x}}}' = (\mathbf{c}'' + \mathbf{v}'\mathbf{A}'')\bar{\mathbf{x}}' = V''_{SP}(v') = V_{SP}(v').$$

Hence, $\bar{\bar{x}}'$ is an optimal solution to (4).

We have now established that $\bar{\bar{\mathbf{x}}}'$ is feasible for (1), optimal for (4), and satisfies complementary slackness by Lemma 2. It is an optimal solution to (1). □

We do not assume that the solution to the subproblem (4) is a basic solution, only that it is an optimal solution. In practice, the subproblem may be solved by the simplex method, yielding an optimal basic solution. However, in some instances it may be convenient to find an optimal solution to the subproblem which is not basic. In general, any convenient algorithm may be used to solve the subproblem; however, a basic optimal solution has much to recommend it, because it has relatively few nonzero components (at most a number equal to the dimension of the basis), and since each nonzero component may result in the addition of a column to the restricted problem, it makes sense to examine extreme point solutions to the subproblem.

PROPOSITION 2. *Assuming nondegeneracy, and the availability of an initial feasible basic solution to* (1), *the procedure described in Steps* 0, 1, *and* 2 *converges after a finite number of iterations to an optimal basic solution to* (1).

PROOF. Starting with an initial basic feasible solution (Step 0), at each iteration of the algorithm a basic feasible solution to (1) is obtained. Each time the algorithm enters at Step 2, at least one potential improving column is identified and added to the columns of the current restricted problem. This assures (again, ignoring degeneracy) that a positive improvement will be made on the first pivot of the new restricted problem, and an improved basic feasible solution will be identified at the end of Step 2. If no improving columns are identified at the end of Step 1, then, by Proposition 1, an optimal solution is at hand. Since there are only a finite number of basic solutions, the procedure must terminate after a finite number of iterations. □

References

Ahuja, R., T. Magnanti, J. Orlin. 1993. *Network Flows*. Prentice Hall, Englewood Cliffs, NJ.

Barnhart, C., E. Johnson, C. Hane, G. Sigismondi. 1994. An alternative formulation and solution strategy for multicommodity network flow problems. *Telecommunications Systems* **3** 239–258.

Bland, R. G. 1977. New finite pivoting rules for the simplex method. *Math. Oper. Res.* **2** 103–107.

Brown, G. G., R. McBride. 1984. Solving generalized networks. *Management Sci.* **30** 1497–1523.

Carolan, W., J. Hill, J. Kennington, S. Niemi, S. Wichmann. 1990. An empirical evaluation of the KORBX algorithms for military airlift applications. *Oper. Res.* **38** 240–248.

Castro, J. 1994. *PPRN* 1.0 User's Guide. DR 94/06, Statistics and Operations Research Dept., Univeristat Politecnica de Catalunya, Barcelona, Spain.

——, N. Nabona. 1996. Primal-dual interior point method for multicommodity network flow with side constraints and comparison with alternative methods. Unpublished manuscript, Statistics and Operations Research Dept., Universitat Politecnica de Catalunya, Barcelona, Spain.

——, ——. 1996. An implementation of linear and nonlinear multicommodity network flows. *European J. Oper. Res.* **92**(1) 37–53.

Fourer, R. 1985. A simplex algorithm for piecewise-linear programming I: Derivation and proof. *Math. Programming* **33** 281–315.

Fourer, R. 1988. A simplex algorithm for piecewise-linear programming II: Derivation and proof. *Math. Programming* **41** 204–233.

Frangioni, A. 1997. Dual-ascent methods and multicommodity flow problems. Ph.D. Thesis TD 5/97, Università di Pisa-Genova-Udine, Italy.

Hane, C. 1996. Private communication.

Lustig, I., Rothberg, E. 1996. Gigaflops in linear programming. *Oper. Res. Letters* **18**(4) 157–165.

Marsten, R., R. Subramanian, I. Lustig, D. Shanno. 1990. Interior point methods for linear programming: Just call Newton, Lagrange, and Fiacco and McCormick! *Interfaces* **20** 105–116.

McBride, R. 1985. Solving embedded generalized network problems. *European J. Oper. Res.* **21** 82–92.

——, J. Mamer. 1997a. Solving multicommodity flow problems with a primal embedded network simplex algorithm. *INFORMS J. Comput.* **9** (Spring) 154–163.

——, ——. 1997b. Solving the undirected multicommodity flow problem using a shortest path based pricing algorithm. Working paper, USC, Los Angeles, CA.

Murthy, R., R. Helgason. 1993. A direct simplex algorithm for network flow problems with piecewise linear costs. Department of Computer Science and Engineering, Southern Methodist University, Dallas, TX.

Nazareth, J. 1987. *Computer Solutions of Linear Programs*. Oxford University Press, Oxford, UK.

Padberg, Manfried. 1995. *Linear Optimization and Extensions*. Springer-Verlag, New York.

Pinar, M., S. Zenios. 1992. Parallel decomposition of multicommodity network flows using a linear-quadratic penalty algorithm. *ORSA J. Comput.* **4** 235–248.

Premoli, A. 1986. Piecewise-linear programming: The compact (CPLP) algorithm, *Math. Programming* **36** 210–227.

Schultz, G., R. Meyer. 1991. An interior point method for block angular optimization. *SIAM J. Optim.* **1** 583–602.

Valerio De Carvalho, J. M. 1997. Exact solution of bin-packing problems using column generation and branch-and-bound. Working Paper, Dept. Producoa e Sistemas, Univerxidade do Minho, 4719 Broga, Portugal.

Accepted by Thomas M. Liebling; received April, 1998. This paper was with the authors 11 months for 2 revisions.