LIDS-P-1696 August 1987

Relaxation Method for Linear Programs with Side Constraints

by

Paul Tseng †

Dimitri P. Bertsekas †

Abstract

We propose a new method for solving linear programs with side constraints. This method extends those proposed in [2] and [15] and uses decomposition to exploit special structures in the side constraints. Application to network flow problems with side constraints is discussed.

*Work supported by the National Science Foundation under grant NSF-ECS-8519058 and by the Army Research Office under grant DAAL03-86-K-0171. The first author is also supported by Canadian NSERC under Grant U0505.

†The authors are with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139.

1. Introduction

Consider a linear program with m variables and n + n′ equality constraints. We denote the first n rows of the constraint matrix by E (E is a n × m real matrix) , the remaining n′ rows of the constraint matrix by D (D is a n′ × m real matrix) , the jth variable by x_j, and the per unit cost of the jth variable by a_j. The problem then has the form:

$$\textit{Minimize} \quad \sum_{j=1}^{m} a_j x_j \tag{P}$$

$$\textit{subject to} \quad \sum_{j=1}^{m} e_{ij} x_j = 0 \quad \forall\, i=1,2,\ldots,n, \tag{1}$$

$$\sum_{j=1}^{m} d_{kj} x_j = b_k \quad \forall\, k=1,2,\ldots,n', \tag{2}$$

$$l_j \le x_j \le c_j \quad \forall\, j=1,2,\ldots,m, \tag{3}$$

where b_k denotes the right hand side of the kth constraint, for k = 1,2,...,n′, the scalars l_j and c_j denote respectively the lower and the upper bound for the jth variable, e_{ij} denotes the (i,j)th entry of E and d_{Kj} denotes the (k,j)th entry of D. For simplicity we assume the first n constraints all have zero right hand side (we can convert any problem having nonzero right hand side to this form by introducing dummy variables). Note the constraints (1) and (2) can be expressed in vector form by Ex = 0 and Dx = b respectively.

A special case of (P), network flow problems with linear side constraints (i.e. E is the node-arc incidence matrix for a network), arise frequently in the areas of manufacturing, traffic flow, and data communication. Conventional methods for solving such problems include Dantzig-Wolfe decomposition and Bender's decomposition. For the special case of multicommodity network flow problems, specialized solution methods such as the primal simplex method using basis partitioning [7] and the primal dual method of

Jewell [6] have been developed. For a survey of multicommodity network flow methods see Assad [1] and Kennington [7].

Let us rewrite **(P)** as the following convex program

$$\textit{Minimize} \quad h(x) + \sum_{i=1}^{n} \delta(d_i) \qquad \textbf{(P')}$$

$$\textit{subject to} \quad x \in X,$$

where

$$h(x) \equiv \begin{cases} a^T x & \textit{if} \quad x \in H, \\ +\infty & \textit{otherwise}, \end{cases}$$

$$\delta(\xi) \equiv \begin{cases} 0 & \textit{if} \quad \xi = 0, \\ +\infty & \textit{otherwise}, \end{cases}$$

$$H \equiv \{ x \mid l \leq x \leq c \ , \ Dx = b \}, \tag{4}$$

and

$$X \equiv \{ x \mid Ex = 0 \}. \tag{5}$$

Note that h is a proper convex and lower semicontinuous function. To simplify the presentation we make the following standing assumptions :

Assumption A: **(P)** is feasible, (i.e. $H \cap X \neq \emptyset$).

Assumption B: D is of full row rank and each extreme point of H satisfies exactly m–n' inequality constraints with equality.

Assumption B implies that, given any linear function mapping $\mathbb{R}^m$ into $\mathbb{R}$, if the minimization of this function over H has an optimal dual solution then this dual solution is unique ([13], pp. 579). Since H is closed, bounded and [cf. Assumption A] nonempty it follows that the minimization of any linear function over H always has an unique optimal dual solution. For any $k \geq 1$, any $\xi \in \mathbb{R}^k$ whose ith component we

denote ξ_i and any subset B of {1,2,..,k}, we will use ξ_B to denote the vector whose components are ξ_i, $i \in B$. For any matrix W with k columns, whose columns are indexed from 1 to k, and any subset B of {1,2,..,k}, we will use W_B to denote the matrix comprised of the columns of W whose indexes are in B.

In this paper we present a new dual descent method for solving **(P)** that extends the relaxation methods in [2] and [15] by incorporating decomposition to handle the side constraints. This method, which we also call relaxation method, emphasizes descent along coordinate directions and, on network flow problems (with side constraints), can be implemented using primarily network data structures; these are key features contributing to the success of the method on pure network flow problems without side constraints [3]. Although this method may be interpreted as a primal dual method in that it maintains both primal and dual variables at each iteration and terminates when the primal variables are feasible, it is quite different from the primal dual method of Jewell [6]. Our paper proceeds as follows: in §2 we formulate the dual of **(P)** as an unconstrained minimization problem and describe the associated optimality conditions; in §3 we study ways of generating dual descent directions, which correspond to the elementary vectors of a certain extended dual space, using the painted index algorithm of Rockafellar [13]; in §4 we describe the relaxation method; in §5 we describe a modification to the relaxation method that ensures finite convergence – one is an out-of-kilter-like modification and the other is based on the notion of ε-complementary slackness; in §6 we discuss implementation of the relaxation method for network flow problems with side constraints; and in §7 we present our conclusion and discuss extensions.

2. Dual Problem and Optimality Conditions

Using Fenchel's Duality Theorem ([10], Ch. 31) the dual program of **(P')** is

$$\begin{aligned} &\textit{Minimize} && h^*(\tau) \\ &\textit{subject to} && \tau \in X^{\perp}, \end{aligned}$$

where

$$h^*(\tau) = \max_{x \in H} \ (\tau - a)^T x \ , \tag{6}$$

$$X^{\perp} = \{ \tau \mid \tau = E^T p \text{ for some } p \} . \tag{7}$$

h* is a piecewise linear convex function since, for a given τ, $h^*(\tau)$ is the maximum of a finite set (corresponding to the set of extreme points of H) of linear functions of τ. Using (4), (6) and linear programming duality [4] we have

$$h^*(\tau) = \begin{array}{ll} max & (\tau-a)^T x \\ subject\,to & Dx = b\ ,\ l \le x \le c \end{array} = \min_{\Phi} \left\{ \begin{array}{ll} max & (\tau-a)^T x + \Phi^T D x - \Phi^T b \\ subject\,to & l \le x \le c \end{array} \right\}$$

$$= \min_{\Phi} \left\{ \begin{array}{ll} max & (\tau + D^T\Phi - a)^T x \\ subject\,to & l \le x \le c \end{array} - \Phi^T b \right\} = \min_{\Phi} \left\{ \sum_{j=1}^{m} g_j(\tau_j + D_j^T \Phi) - \Phi^T b \right\}, \tag{8}$$

where D_j denotes the jth column of D and

$$g_j(\eta) \equiv \begin{cases} (\eta - a_j) l_j & \text{if } \eta \le a_j \\ (\eta - a_j) c_j & \text{if } \eta > a_j \end{cases} . \tag{9}$$

Using (9) we can expand the last term in (8) to obtain

$$h^*(\tau) = \min_{\Phi} \rho(\tau,\Phi) , \tag{10a}$$

where

$$\rho(\tau,\Phi) \equiv l^T(\tau + D^T\Phi - a) + (c-l)^T[\tau + D^T\Phi - a]^+ - \Phi^T b . \tag{10b}$$

and $[z]^+$ denotes the orthogonal projection of z onto the positive orthant. Assumption B ensures that for a given τ, the argument that achieves the minimum in (10a) is unique.

Using (6) and (7), we can reformulate the above dual program as an unconstrained minimization problem

$$\begin{array}{ll} Minimize & q(p) \\ subject\ to\ no\ constraint\ on\ p & \end{array} , \tag{D}$$

where q denotes the convex piecewise linear function

$$q(p) \equiv h^*(E^T p) . \tag{11}$$

We will call p, the Lagrange multiplier associated with the linear homogeneous constraints (1), the price vector. Since [cf. Assumption B] the argument of the minimization in (10a) is uniquely determined by τ, we can define a point-to-point mapping $\phi: \mathbb{R}^n \to \mathbb{R}^{n'}$ by

$$\phi(p) \equiv arg \min_{\phi} \rho(E^T p, \phi) . \tag{12}$$

Then using (10a), (10b), (11) and (12) we can express q(p) explicitly as a function of p :

$$q(p) = l^T(E^T p + D^T\phi(p) - a) + (c-l)^T[E^T p + D^T\phi(p) - a]^+ - \phi(p)^T b . \tag{13}$$

We will borrow the terminology in [13] for network programming and call the vector t satisfying

$$t \equiv E^T p + D^T\phi(p) \tag{14}$$

the tension vector associated with p.

Instead of (10b) and (12), we can use (5) and (7) to express $\phi(p)$ explicitly in terms of the optimal basis for the linear program

$$q(p) = \begin{array}{ll} Max & (E^T p - a)^T x \\ subject\ to & D x = b \\ & l \le x \le c. \end{array} \tag{15}$$

From classical linear programming theory we know

$$\phi(p) = (D_B^{-1})^T(a_B - E_B^T p), \tag{16}$$

where D is partitioned into $[\, D_B \;\; D_N \,]$ such that D_B is an optimal basis matrix for (15) and a and E are correspondingly partitioned into $a = (a_B, a_N)$ and $E = [\, E_B \;\; E_N \,]$. The dual problem **(D)** has two advantages – (i) the maximization problem (15) typically decomposes into many small easy-to-solve subproblems (as in the case where **(P)** is a multicommodity network flow problem) and (ii) being unconstrained, **(D)** is amenable to solution by a coordinate descent type method.

We now give the necessary and sufficient conditions for a primal dual pair (x, p) to be optimal for **(P)** : Let t denote the tension vector associated with p [cf. (14)] and define each column j (j = 1,2,...,m) to be

inactive	if	$t_j < a_j$,
balanced	if	$t_j = a_j$,
active	if	$t_j > a_j$.

Then (x, p) is said to satisfy the complementary slackness condition, CS for short, if x achieves the maximum in (15), or equivalently if x satisfies the following optimality conditions :

$$\begin{aligned} & x_j = l_j && \forall \text{ inactive } j, \\ & l_j \le x_j \le c_j && \forall \text{ balanced } j, \\ & x_j = c_j && \forall \text{ active } j \\ \text{and} \quad & Dx = b. \end{aligned} \tag{CS}$$

For each primal vector x, we define the deficit of row i to be

$$d_i = \sum_{j=1}^{m} e_{ij} x_j .$$

Let d be the vector with coordinates d_i (in vector form d = Ex). We define the total deficit for x to be

$$\sum_{i=1}^{n} |d_i|$$

(the total deficit is a measure of how close x is to satisfying (1)). Then a primal dual pair (x, p) is optimal for (P) and (D) respectively if and only if (x, p) satisfies CS and d = 0.

3. Dual Descent Directions and the Modified Painted Index Algorithm

We first compute, for a given price vector p and a direction u, the directional derivative of q at p in the direction of -u. Using (13) we obtain

$$q'(p;-u) = - \sum_{\substack{t_j < a_j \text{ or} \\ t_j = a_j, v_j > 0}} l_j v_j - \sum_{\substack{t_j > a_j \text{ or} \\ t_j = a_j, v_j < 0}} c_j v_j + \Psi^T b , \tag{17}$$

where q'(p;-u) denotes the directional derivative of q at p in the direction -u

$$q'(p;-u) \equiv \lim_{\lambda \downarrow 0} \frac{q(p-\lambda u) - q(p)}{\lambda} ,$$

t is given by (14),

$$v \equiv E^T u + D^T \Psi \tag{18}$$

and Ψ is the unique vector satisfying

$$\Phi(p - \lambda u) = \Phi(p) - \lambda \Psi \qquad \textit{for } \lambda > 0 \textit{ sufficiently small}. \tag{19}$$

Such Ψ can be computed using the optimal basis from the maximization problem (15) with the cost vector perturbed in the direction $-E^T u$ (see Appendix A on computing Ψ).

Let x satisfy CS with p, and let d = Ex. Then we have (using (18) and the definition of CS)

$$x^T v = d^T u + (Dx)^T \Psi = d^T u + b^T \Psi \tag{20}$$

and

$$x_j = \begin{cases} l_j & \text{if } t_j < a_j \\ c_j & \text{if } t_j > a_j \end{cases}. \tag{21}$$

Combining (17), (20) and (21) we obtain :

$$q'(p; -u) = -C(u,p),$$

where

$$C(u,p) = d^T u + \sum_{t_j = a_j,\, v_j > 0} (l_j - x_j) v_j + \sum_{t_j = a_j,\, v_j < 0} (c_j - x_j) v_j . \tag{22}$$

Equation (22) is interesting in that its right hand side depends on the side constraints solely through the vector v [cf. (14), (18) and (19)]. To evaluate C(u,p) we first compute the vector Ψ satisfying (19) and then use (18) and (22).

Clearly q is piecewise linear since q is the composition of a piecewise linear function, namely h^*, with a linear function [cf. (11)]. Then following an argument similar to that for Propostion 2 in [15] we obtain :

Proposition 1

$$q(p - \lambda u) = q(p) - \lambda q'(p; -u) \qquad \forall\, \lambda \in [0, \alpha),$$

where

$$\alpha = \inf_{v_j (a_j - t_j) < 0} \left\{ \frac{t_j - a_j}{v_j} \right\} = \inf \left\{ \inf_{v_j > 0 \; j \, active} \frac{t_j - a_j}{v_j} \,,\; \inf_{v_j < 0 \; j \, inactive} \frac{t_j - a_j}{v_j} \right\} \tag{23}$$

(α is the stepsize at which some column becomes balanced.), t is given by (14) and v is given by (18), (19).

How can we generate dual descent directions for **(D)**? The basic idea is to apply, for a given non-optimal primal dual pair satisfying CS, a modified version of the painted index algorithm of Rockafellar ([13], Ch. 10G) to either (a) generate a dual descent direction, or (b) change the primal vector so as to reduce the total deficit while maintaining CS. In what follows we assume the reader is familiar with the notion of elementary vectors, Tucker tableaus and painted index algorithm as described in Ch. 10 of [13].

Modified Painted Index Algorithm

Given primal dual pair (x, p) satisfying CS, let $t = E^T p + D^T \phi(p)$ and $d = Ex$. If $d = 0$ then (x, p) satisfies the optimality conditions so x and p solves **(P)** and **(D)** respectively. If $d \neq 0$, then we select some row s for which $d_s \neq 0$. In the description that follows we assume $d_s > 0$. The case where $d_s < 0$ may be treated in an analogous manner.

We apply the painted index algorithm, with s as the lever index and using Bland's anticycling rule, to the extended linear homogeneous system (whose columns are indexed from 1 to $n + n' + m$)

$$\begin{array}{ccc} {\scriptstyle 1,\ldots,n} & {\scriptstyle n+1,\ldots,n+n'} & {\scriptstyle n+n'+1,\ldots,n+n'+m} \\ \left[\begin{array}{ccc} -I & 0 & E \\ 0 & -I & D \end{array}\right] & \left[\begin{array}{c} w \\ y \\ z \end{array}\right] & = 0, \end{array} \tag{24}$$

where index i (corresponding to w_i), i = 1,2,...,n, is painted

white if $d_i > 0$,

black if $d_i < 0$,

red if $d_i = 0$,

index $n + k$ (corresponding to y_k), k = 1,2,...,n', is painted

red

and index $n + n' + j$ (corresponding to z_j), j = 1,2,...,m, is painted

green if j balanced and $l_j < x_j < c_j$,

black if j balanced and $l_j = x_j < c_j$,

white if j balanced and $l_j < x_j = c_j$,

red if j not balanced

or if j balanced and $l_j = x_j = c_j$.

Furthermore, we (i) use as the initial Tucker tableau one for which s is in row position and (ii) assign the lowest priority to index s (this ensures that s is always in row position, as is shown in Appendix B of [15]). At each iteration of the painted index algorithm we check if a dual descent direction can be obtained from the current Tucker tableau as follows :

We say a component of (w, y, z) is a column (row) variable if its corresponding index is in column (row) position of the tableau. We denote

a_{si} = entry in row indexed by s of tableau corresponding to column variable w_i.

a_{sk} = entry in row indexed by s of tableau corresponding to column variable y_k.

a_{sj} = entry in row indexed by s of tableau corresponding to column variable z_j.

Applying the General Basis Theorem ([13], pp. 457) to the extended linear homogeneous system (24) we obtain the elementary vector (u, π, v) of $C^{\perp}$ using s (normalized so $u_s = 1$), where

$$C \equiv \left\{ (-d, -w, x) \;\middle|\; \begin{bmatrix} E \\ D \end{bmatrix} x = \begin{bmatrix} d \\ w \end{bmatrix} \right\}, \qquad C^{\perp} \equiv \left\{ (u, \pi, v) \;\middle|\; v = E^T u + D^T \pi \right\}, \tag{25}$$

that corresponds to the current tableau to be

$$u_i = \begin{cases} 1 & \text{if } i = s \\ -a_{si} & \text{if } w_i \text{ is a column variable} \\ 0 & \text{otherwise} \end{cases}, \tag{26}$$

$$\pi_k = \begin{cases} a_{sk} & \text{if } y_k \text{ is a column variable} \\ 0 & \text{otherwise} \end{cases}, \tag{27}$$

$$v_j = \begin{cases} a_{sj} & \text{if } z_j \text{ is a column variable} \\ 0 & \text{otherwise} \end{cases}. \tag{28}$$

If C(u,p) > 0 then -u is a dual descent direction and the algorithm terminates. Note from (22) that if the tableau is such that its row indexed by s is compatible ([13], pp. 475) then -u is a dual descent direction. This is because our choice of index painting and the definition of a compatible row imply

$d_s > 0$ and $a_{si} d_i \le 0$ for all i such that w_i is a column variable,

$x_j = c_j$ for all j such that z_j is a column variable, n + n' + j is red or black, and $a_{sj} < 0$ (29)

and $x_j = l_j$ for all j such that z_j is a column variable, n + n' + j is red or white, and $a_{sj} > 0$,

which in view of the definition of CS implies that π satisfies

$$\phi(p-\lambda u) = \phi(p) - \lambda\pi \quad \textit{for } \lambda>0 \textit{ sufficiently small}.$$

So using the definition of C(u,p) [cf. (14), (18), (19) and (22)] we have

$$C(u,p) = d^T u + \sum_{t_j=a_j,\, v_j>0} (l_j - x_j)v_j + \sum_{t_j=a_j,\, v_j<0} (c_j - x_j)v_j$$

whose right hand side according to (26) , (28) and (29) is positive.

We know that the painted index algorithm using Bland's priority rule terminates finitely with either a compatible row using s or a compatible column using s (see [13], Ch. 10H). Thus we must either find a dual descent direction -u for which C(u,p) > 0 [cf. discussion above] or else find a Tucker tableau containing a compatible column using s. In the latter case, an incremental change towards primal feasibility is performed as follows :

Let r^* denote the index of the compatible column.

Let a_{ir^*} denote the entry in the compatible column corresponding to row variable w_i , let a_{kr^*} denote the entry in the compatible column corresponding to row variable y_k , and let a_{jr^*} denote the entry in the compatible column corresponding to row variable z_j .

Case 1 If r^* is of the form $r^* = i$ for some $i \in \{1,\ldots,n\}$ and r^* is black then set

$$w_i^* \leftarrow \begin{cases} 1 & \text{if } i = r^* \\ a_{ir^*} & \text{if } i \text{ is basic} \\ 0 & \text{else} \end{cases}, \quad y_k^* \leftarrow \begin{cases} a_{kr^*} & \text{if } n+k \text{ is basic} \\ 0 & \text{else} \end{cases}, \quad z_j^* \leftarrow \begin{cases} a_{jr^*} & \text{if } n+n'+j \text{ is basic} \\ 0 & \text{else} \end{cases}.$$

Case 2 If r^* is of the form $r^* = n + n' + j$ for some $j \in \{1,\ldots,m\}$ and r^* is black then set

$$w_i^* \leftarrow \begin{cases} a_{ir^*} & \text{if } i \text{ is basic} \\ 0 & \text{else} \end{cases}, \quad y_k^* \leftarrow \begin{cases} a_{kr^*} & \text{if } n+k \text{ is basic} \\ 0 & \text{else} \end{cases}, \quad z_j^* \leftarrow \begin{cases} 1 & \text{if } n+n'+j = r^* \\ a_{jr^*} & \text{if } n+n'+j \text{ is basic} \\ 0 & \text{else} \end{cases}.$$

Case 3 If r^* is of the form $r^* = i$ for some $i \in \{1,\ldots,n\}$ and r^* is white then set

$$w_i^* \leftarrow \begin{cases} -1 & \text{if } i = r^* \\ -a_{ir^*} & \text{if } i \text{ is basic} \\ 0 & \text{else} \end{cases}, \quad y_k^* \leftarrow \begin{cases} -a_{kr^*} & \text{if } n+k \text{ is basic} \\ 0 & \text{else} \end{cases}, \quad z_j^* \leftarrow \begin{cases} -a_{jr^*} & \text{if } n+n'+j \text{ is basic} \\ 0 & \text{else} \end{cases}.$$

Case 4 If r^* is of the form $r^* = n + n' + j$ for some $j \in \{1,...,m\}$ and r^* is white then set

$$w_i^* \leftarrow \begin{cases} -a_{ir^*} & \text{if } i \text{ is basic} \\ 0 & \text{else} \end{cases} \quad y_k^* \leftarrow \begin{cases} -a_{kr^*} & \text{if } n+k \text{ is basic} \\ 0 & \text{else} \end{cases} \quad z_j^* \leftarrow \begin{cases} -1 & \text{if } n+n'+j = r^* \\ -a_{jr^*} & \text{if } n+n'+j \text{ is basic} \\ 0 & \text{else} \end{cases}.$$

That w^*, y^*, and z^* so defined satisfy $w^* = Ez^*$, $y^* = Dz^*$ follows from applying the General Basis Theorem ([13], pp. 457) to the extended linear homogeneous system (24) (with (w^*, y^*, z^*) normalized so $w^*_{r^*} = 1$ in cases 1, 2 and $w^*_{r^*} = -1$ in cases 3 and 4). Furthermore, our choice of index painting, together with compatibility of the column indexed by r^*, guarantees that, for $\mu > 0$ sufficiently small, $x + \mu z^*$ satisfies CS with p and that $x + \mu z^*$ has strictly smaller total deficit than x. Note that since both x and $x + \mu z^*$ satisfy CS we always have $y^* = D(x + \mu z^*) - Dx = 0 - 0 = 0$ (this can also be seen from the fact that the indexes $n + 1, ..., n + n'$, are always painted red).

In summary, the modified painted index algorithm either produces a dual descent direction $-u$ given by (26) or produces a primal direction z^* as defined above that can be used to reduce the total deficit. In the special case where E is the node-arc incidence matrix for an ordinary network, we can perform the pivoting operation more efficiently by exploiting the network structure (see §6).

To determine whether $-u$ given by (26) is a dual descent direction may be computationaly expensive. In general if s is not a compatible row then the vector π given by (27) does not satisfy

$$\phi(p - \lambda u) = \phi(p) - \lambda\pi \quad \text{for } \lambda > 0 \text{ sufficiently small}.$$

(an example of this is easy to construct). For this reason $C(u,p)$ cannot in general be determined directly from the entries of the Tucker tableau and must be computed using (14), (18), (19) and (22). A compromise is to instead compute a lower bound on $C(u,p)$. It can be shown (see Lemma 1 in §5.2) that, for any $\pi \in \mathbb{R}^{n'}$, the following quantity

$$C(u,\pi,p) \equiv u^T d + \sum_{\substack{v_j > 0 \\ j \text{ balanced}}} v_j(l_j - x_j) + \sum_{\substack{v_j < 0 \\ j \text{ balanced}}} v_j(c_j - x_j) ,$$

where $v \equiv E^T u + D^T \pi$, x is a primal vector satisfying CS with p, and d = Ex, is a lower bound on C(u,p) (this in particular implies C(u,p) = $\max_\pi$ C(u,π,p) = C(u,ψ,p), where ψ is given by (19)). It follows if π is chosen by (27) then C(u,π,p) is directly computable from the Tucker tableau, with u given by (26) and v given by (28). Furthermore, if the lever row that generates u is compatible then C(u,π,p) = C(u,p) in which case the bound is tight. Thusfore we can use C(u,π,p) instead of C(u,p) in the modified painted index algorithm to reduce the computational effort per pivot.

One particular choice of the initial Tucker tableau that has proven to be computationally successful on problems without side constraints is

$$\begin{bmatrix} E \\ D \end{bmatrix},$$

for which the indexes 1 to n + n′ are basic. Since the direction associated with the lever row s of this tableau [cf. (26)] is the sth coordinate vector in $\mathbb{R}^n$, this implementation may be viewed as a generalized coordinate descent or relaxation implementation whereby coordinate directions are given priorities as candidate for dual descent. Computational tests showed that on network flow problems (without side constraints) the coordinate directions typically contribute between 80 to 90 percent of the improvements in the dual functional [16].

4. The Relaxation Method

Based on the discussions in §3, we can now formally describe the relaxation method for **(P)** and **(D)**. Each iteration of the method begins with a primal dual pair (x, p) satisfying CS and returns another pair (x′, p′) satisfying CS for which either (i) $q(p') < q(p)$ or (ii) $q(p') = q(p)$ and (total deficit of x′) < (total deficit of x).

Relaxation Iteration

Step 0 Given primal dual pair (x, p) satisfying CS. Denote $d = Ex$.

Step 1 If $d = 0$ then x is primal feasible and we terminate the method. Otherwise choose a row s for which d_s is nonzero. For convenience we assume $d_s > 0$. The case where $d_s < 0$ may be treated analogously.

Step 2 Apply the modified painted index algorithm with s as the lever index to the extended system

$$\begin{bmatrix} -I & 0 & E \\ 0 & -I & D \end{bmatrix} \begin{bmatrix} w \\ y \\ z \end{bmatrix} = 0$$

as described in §3. If the algorithm terminates with a dual descent direction -u we go to Step 4. Otherwise the algorithm terminates with a compatible column using s, in which case we go to Step 3.

Step 3 (Primal Rectification Step)

Let

$$\mu = \min \left\{ \min_{z_j^* > 0} \frac{c_j - x_j}{z_j^*}, \min_{z_j^* < 0} \frac{l_j - x_j}{z_j^*}, \min_{w_i^* \neq 0} \frac{-d_i}{w_i^*} \right\},$$

where z^*, w^* are computed as discussed at the end of §3. Set $x' \leftarrow x + \mu z^*$ and $p' \leftarrow p$ (The choice of μ is the largest for which CS is maintained and the magnitude of each deficit is monotonically decreased).

Step 4 (Dual Descent Step)

Determine a stepsize λ^* for which

$$q(p-\lambda^* u) = \min \{q(p-\lambda u) \mid \lambda > 0\} .$$

Set $p' \leftarrow p - \lambda^* u$ and compute x' that satisfies CS with p'.

Validity and Finite Termination of the Relaxation Iteration

We will show that all steps in the relaxation iteration are executable, that the iteration terminates in a finite number of operations, and CS is maintained. Since the modified painted index algorithm (with Bland's priority pivoting rule) is finitely terminating, the relaxation iteration must then terminate finitely with either a primal rectification step (Step 3) or a dual descent step (Step 4). Step 3 is clearly executable and finitely terminating. Step 4 is executable for if there does not exist a line minimization stepsize λ^* in the direction -u then since -u is a dual descent direction at p, the convexity of q would imply

$$q'(p-\lambda u;-u) < 0 \quad \text{for all } \lambda > 0 .$$

This, in view of the piecewise linear nature of q, implies q'(p-λu;-u) is bounded away from 0 and therefore

$$\lim_{\lambda \to \infty} q(p-\lambda u) = -\infty .$$

This contradicts Assumption A. CS is trivially maintained in Step 4. In Step 3, the only change in the primal or dual vector comes from the change in the value of some primal variable(s) whose corresponding column is balanced. Since the amount of change μ is chosen such that each such primal variable satisfies the capacity constraints (2) it follows that CS is maintained.

Implementation of the line minimization in Step 4

Perhaps the most efficient scheme for implementing the line minimization of Step 4 is to move along the breakpoints of q, in the descent direction -u, until the directional derivative becomes nonnegative. This scheme also allows us to efficiently update the value of C(u,p). Algorithmically it proceeds as follows :

Step 4a Start with p and u such that $C(u,p) > 0$.

Step 4b If $C(u,p) \leq 0$ then exit (line minimization is complete). Else compute α using (23) (α is the stepsize to the next breakpoint of q from p in the direction -u). Then move p in the direction -u to a distance of α and update t and x :

Decrease p_i by αu_i $\forall$ i.

Set $x_j \leftarrow l_j$ $\forall$ balanced j such that $v_j > 0$.

Set $x_j \leftarrow c_j$ $\forall$ balanced j such that $v_j < 0$.

Decrease t_j by αv_j $\forall$ j.

Update ψ and v using (18), (19). Upate C(u,p) using (22).

Return to Step 4b.

5. Finite Convergence of the Relaxation Method

The relaxation method that consists of successive iterations of the type described in §4 is not guaranteed to converge to an optimal dual solution. We distinguish the following two difficulties :

(a) Only a finite number of dual descents may take place because all iterations after a finite number end up with a primal rectification step.

(b) An infinite number of dual descents take place, but the generated sequence of dual costs does not converge to the optimal cost.

Difficulty (a) may be bypassed by choosing an appropriate priority assignment of the indexes in the relaxation iterations, similar to Proposition 3 in [15]:

Proposition 2 If in the relaxation method the green indexes are assigned the highest priorities and the black and white indexes belonging to {1,2,...,n}, except for the lever index, are assigned the second highest priorities, then the number of primal rectification steps between successive dual descent steps is finite.

Proof : This is a special case of Proposition 3 in [15] where the indexes $n + 1$ up to $n + n'$ are always painted red.

Difficulty (b) can arise as shown by the example in [16], Appendix G. To bypass difficulty (b) we will consider two types of modification to the relaxation iteration of §4 and show that the number of dual descents is finite under either type.

5.1. Finite Convergence using an Out-of-Kilter modification

This modification is reminiscent of the out-of-kilter method and is easily implementable:

1. In the modified painted index algorithm, perform a dual descent only if the lever row in the Tucker tableau is compatible, and then use as stepsize that given by (23) (i.e. the stepsize to the first breakpoint of q from p in the direction $-u$ given by (26)).
2. We assign higher priorities to black or white indexes of the form i, $i \in \{1,...,n\}$, except for the lever index, over black or white indexes of the form $n + n' + j$, $j \in \{1,...,m\}$.
3. If the previous iteration terminated with a dual descent step then use the same lever index as in the previous iteration and use the final tableau from the previous iteration as the initial tableau .

The out-of-kilter modification however places two major limitations on the method – (i) dual descent cannot be performed until lever row becomes compatible, and (ii) the same lever index is used between consecutive primal rectifications. For this reason we may wish to implement this modification only after sufficient large number of dual descents have been performed. Our proof of finite convergence is similar to that used by Rockafellar for his general out-of-kilter method (see [13], Ch. 11K). Our condition for convergence differs from his in that we do not require the same lever index (which always corresponds to a row of E in our case) be used at successive iterations until the corresponding deficit reaches zero, but in its place we require modification 2.

To prove finite convergence we need, in addition to Proposition 2, the following:

Proposition 3 The number of dual descent steps between successive primal rectification steps is finite.

Proof of Proposition 3 is given in Appendix B (it is similar to the proof of Proposition 2 but applied to the dual of (24)).

Now suppose the relaxation method does not terminate finitely. Then Propositions 2 and 3, together with modification 1, imply that compatible lever rows are found in an infinite number of relaxation iterations. It can be seen [cf. the primal rectification algorithm of ([13], pp. 485) and (25)] that there exists a Tucker tableau representing C and $C^{\perp}$ having a compatible row indexed by s if and only if the following linear system in (w, y, z)

$$\begin{bmatrix} -I & 0 & E \\ 0 & -I & D \end{bmatrix} \begin{bmatrix} w \\ y \\ z \end{bmatrix} = 0,\ y = 0,\ w_s < 0,\ \left\{ \begin{array}{l} w_i = 0 \text{ if } d_i = 0 \\ w_i \geq 0 \text{ if } d_i < 0 \\ w_i \leq 0 \text{ if } d_i > 0 \end{array} \right\},\ \left\{ \begin{array}{ll} z_j = 0 & \text{if } t_j \neq a_j \\ z_j \geq 0 & \text{if } t_j = a_j, x_j = l_j \\ z_j \leq 0 & \text{if } t_j = a_j, x_j = c_j \end{array} \right\},$$

where x denotes the current primal vector and t denotes the tension vector given by (14), is inconsistent. This implies the deficit of s cannot be decreased in magnitude without strictly increasing the magnitude of other deficit(s), while fixing x_j at l_j for all inactive j and fixing x_j at c_j for all active j. On the other hand, modification 1 implies the primal vector does not change during each dual descent (see note below) so the deficits are monotonically decreasing in magnitude (since the deficits are decreasing in magnitude during each primal rectification step). It then follows that no compatible lever row can repeat with the same combination of lever index, set of inactive j's and set of active j's – a contradiction since the number of such combinations is finite.

Since the relaxation method maintains complementary slackness at all iterations and terminates only when the deficit of all rows are zero, the final primal dual pair produced by the method must be optimal (since the final primal vector is feasible for (P)). Thus we have the main result of this section:

Proposition 4 If in the relaxation method the green indexes are assigned the highest priorities, the black and white indexes belonging to {1,2,...,n}, except for the lever index, are assigned the second highest priorities, and the out-of-kilter modification is implemented, then the method converges finitely.

It should be noted that using the same lever index as in the previous iteration when the previous iteration terminated with a dual descent is a necessary condition for the method to be finitely convergent. If a different lever index is used then the method may perform an infinite number of successive dual descents, even if the final tableau from the previous iteration is used as the initial tableau for the current iteration (see [16], Appendix G for such an example).

Note: The deficits do not change during each dual descent step because when the descent direction -u is given by a compatible row [cf. (26)] we have

$$\begin{aligned} x_j \in (l_j, c_j) \quad &\Rightarrow \quad v_j = 0, \\ x_j = l_j \quad &\Rightarrow \quad v_j \geq 0, \\ x_j = c_j \quad &\Rightarrow \quad v_j \leq 0, \end{aligned}$$

for all balanced j, where (x, p) denotes the current primal dual pair and v is given by (27) and (28). Then (29) implies for any price vector on the line segment joining p and $p - \alpha u$, where α is given by (23), the corresponding tension vector is on the line segment joining t and $t - \alpha v$, where t denotes tension vector associated with p (since π given by (27) in this case satisfies (19) for all $\lambda \in [0,\alpha]$). It follows from the choice of α [cf. (23)] that x satisfies CS with this price vector so x is unchanged.

5.2. Finite Convergence using the ε-CS Modification

In this section we consider a modification based on the notion of ε-complementary slackness [15] that places virtually no additional restriction on the execution of the relaxation iterations.

Let ε be a fixed nonnegative scalar and define, for each price vector p, column j to be

$\underline{\varepsilon\text{-inactive}}$ if $t_j < a_j - \varepsilon$,

$\underline{\varepsilon\text{-balanced}}$ if $a_j - \varepsilon \leq t_j \leq a_j + \varepsilon$,

$\underline{\varepsilon\text{-active}}$ if $t_j > a_j + \varepsilon$,

where t is the tension vector associated with p [cf. (14)]. Then a primal dual pair (x, p) is said to satisfy $\underline{\varepsilon\text{-complementary slackness}}$ if

$$\begin{aligned} &x_j = l_j && \forall\ \varepsilon\text{-inactive } j, \\ &l_j \leq x_j \leq c_j && \forall\ \varepsilon\text{-balanced } j, \\ &x_j = c_j && \forall\ \varepsilon\text{-active } j, \\ &Dx = b. \end{aligned} \tag{ε-CS}$$

For each p, u in $\mathbb{R}^n$ and π in $\mathbb{R}^{n'}$ we define

$$C^{\varepsilon}(u,\pi,p) \equiv \sum_{a_j - t_j > \varepsilon} v_j l_j + \sum_{\substack{|a_j - t_j| \leq \varepsilon \\ v_j > 0}} v_j l_j + \sum_{a_j - t_j < -\varepsilon} v_j c_j + \sum_{\substack{|a_j - t_j| \leq \varepsilon \\ v_j < 0}} v_j c_j - \pi^T b, \tag{30a}$$

where

$$v = E^T u + D^T \pi. \tag{30b}$$

We can express $C^{\varepsilon}(u,\pi,p)$ in a form analogous to (22). Let x be any primal vector that satisfies ε-CS with p and let d = Ex. Then we have [cf. (30a), (30b) and the definition of ε-CS]

$$C^{\varepsilon}(u,\pi,p) = \sum_{j=1}^{m} v_j x_j + \sum_{\substack{|a_j - t_j| \leq \varepsilon \\ v_j > 0}} v_j (l_j - x_j) + \sum_{\substack{|a_j - t_j| \leq \varepsilon \\ v_j < 0}} v_j (x_j - c_j) - \pi^T b$$

$$= (u^T E + \pi^T D)x + \sum_{\substack{|a_j - t_j| \leq \varepsilon \\ \pi_j > 0}} v_j (l_j - x_j) + \sum_{\substack{|a_j - t_j| \leq \varepsilon \\ \pi_j < 0}} v_j (x_j - c_j) - \pi^T b$$

$$= u^T d + \sum_{\substack{v_j > 0 \\ j\ \varepsilon\text{-balanced}}} v_j (l_j - x_j) + \sum_{\substack{v_j < 0 \\ j\ \varepsilon\text{-balanced}}} v_j (c_j - x_j), \tag{31}$$

where the last equality follows from the fact that $Dx-b=0$ and $d=Ex$. It appears computing $C^{\varepsilon}(u,\pi,p)$ using (31) is more efficient than using (30a). Furthermore (31) makes it evident that, for each primal dual pair (x, p) satisfying ε-CS, if in the painting of the modified painted index algorithm we replace CS by ε-CS then each compatible lever row would yield a u and a π [cf. (26) and (27)] for which $C^{\varepsilon}(u,\pi,p) > 0$ (since the corresponding v is given by (28) then (29) implies $u^Td > 0$ and $x_j = l_j$ ($x_j = c_j$) if $v_j > 0$ ($v_j < 0$) and j is ε-balanced). This observation motivates the following modified relaxation iteration:

Relaxation Iteration with ε-CS modification

Same as the relaxation iteration described in §4 but with CS replaced by ε-CS and C(u,p) replaced by $C^{\varepsilon}(u,\pi,p)$ in the modified painted index algorithm, where π is computed using (27).

We have shown that every compatible row yields a u and a π for which $C^{\varepsilon}(u,\pi,p) > 0$. Therefore each relaxation iteration with ε-CS modification must either find a u and a π for which $C^{\varepsilon}(u,\pi,p) > 0$ or find a compatible column using the lever index. As was shown at the end of §3, a compatible column using the lever index yields a (-w, -y, z) in C for which moving x in the direction z would maintain ε-CS with p while strictly decreasing the total deficit. Therefore if a compatible column using the lever index is found we perform a primal rectification step just as before.

Proposition 2 shows the number of primal rectification steps between consecutive dual descents is finite. Therefore to prove finite termination of the relaxation method with the ε-CS modification it suffices to show the number of descents along directions -u given by (26), for which $C^{\varepsilon}(u,\pi,p) > 0$ for some π, is finite. To do this we first show that $C^{\varepsilon}(u,\pi,p) > 0$ implies -u is a dual descent direction at p and along -u the line minimization stepsize is lower bounded by a positive scalar multiple of ε :

Lemma 1 Let ε be a nonnegative scalar. Then for any $p\in\mathbb{R}^n$, $u\in\mathbb{R}^n$, and $\pi\in\mathbb{R}^{n'}$, $q'(p-\alpha u;-u) \le -C^{\varepsilon}(u,\pi,p)$ for all $\alpha\in[0,\varepsilon/\theta(u)]$ where $\theta(u) \equiv \max\{\|v\|_\infty \mid v \text{ given by (18) and (19) for some } p\}$.

The proof of Lemma 1 is given in Appendix C. Lemma 1 says that if $C^{\varepsilon}(u,\pi,p) > 0$ for some π then -u is a dual descent direction at p and furthermore the line minimization stepsize in the direction of -u is at least $\varepsilon/\theta(u)$. This is rather surprising since, for ε positive, v as given by (18) and (19) may change as the price

vector moves along the line segment between p and $p-\varepsilon u/\theta(u)$ while the rate of descent in the direction -u [cf. (17) or (22)] depends critically on v. In other words, as the price vector moves along the direction -u from p, there may occur a change in the optimal basis to the subproblem (15) arbitrarily close to p (in Appendix D we give an example illustrating this). At such a point, ψ and v as given by (18) and (19) would change, and it is not obvious that this change would not make the directional derivative positive.

Now consider the case where ε is positive (note in the case where $\varepsilon = 0$ the relaxation iteration using ε-CS modification reduces to the relaxation iteration of §4). Since the number of distinct dual descent directions -u used by the ε-CS relaxation method is finite (recall that u is given by the row entries of some Tucker tableau [cf. (26)] and the number of distinct Tucker tableaus is finite), the number of distinct values of rate of descent [cf. (17), (18) and (19)] is finite and so it follows the rate of each dual descent is lower bounded by a positive constant. By Lemma 1, the line search stepsize at each dual descent step is lower bounded by the positive scalar ε/θ where $\theta \equiv \max\{\theta(u) \mid u \text{ given by (26)}\}$. Therefore we can lower bound the improvement in the dual functional q per dual descent step by a positive constant (which depends on the problem data and ε only) and it follows the total number of dual descents is finite. This observation together with Proposition 2 yield the main result of this section:

Proposition 5 If the conditions of Proposition 2 are met then the relaxation method that uses the ε-CS modification, with ε being any fixed positive scalar, terminates finitely with a primal dual pair (x, p) satisfying ε-CS and $x \in X$.

Proof : We have already shown that, under the stated conditions, the relaxation method terminates finitely. Since ε-CS is maintained at all iterations and the method terminates only if the deficit of all rows are zero, the final primal dual pair (x, p) produced by the method must satisfy ε-CS and $x \in X$. Q.E.D.

We note that in the ε-CS relaxation method we can check for a dual descent direction (in the modified painted index algorithm) by reading the values of u , π , and v directly from the row entries of the Tucker tableau [cf. (26), (27) and (28)] and then evaluate $C^{\varepsilon}(u,\pi,p)$ using (30a), (30b) or (31) (in fact if we use (31) we do not need to compute π) .

We sense that for ε small, a primal dual pair (x,p) satisfying ε-CS and $x \in X$ would be close to being optimal. We make this notion precise below (compare to Proposition 6 in [15]):

Proposition 6 If (x, p) satisfies ε-complementary slackness and $x \in X$ then

$$0 \leq a^T x + q(p) \leq \varepsilon \sum_{j=1}^{m} (c_j - l_j). \tag{32}$$

Proof: Using $Ex = 0$ and $Dx = b$ we have

$$a^T x = a^T x - p^T E x - \phi(p)^T D x + \phi(p)^T b. \tag{33}$$

Define

$$\bar{a} \equiv a - E^T p - D^T \phi(p).$$

Then (33) and the definition of ε-complementary slackness imply

$$a^T x = \phi(p)^T b + \sum_{\bar{a}_j > \varepsilon} \bar{a}_j l_j + \sum_{\bar{a}_j < -\varepsilon} \bar{a}_j c_j + \sum_{-\varepsilon \leq \bar{a}_j \leq \varepsilon} \bar{a}_j x_j. \tag{34}$$

On the other hand [cf. (13)]

$$q(p) = -\phi(p)^T b - \sum_{\bar{a}_j > 0} \bar{a}_j l_j - \sum_{\bar{a}_j < 0} \bar{a}_j c_j. \tag{35}$$

Combining (34) with (35) and we obtain

$$a^T x + q(p) = \sum_{0 < \bar{a}_j \leq \varepsilon} \bar{a}_j (x_j - l_j) + \sum_{-\varepsilon \leq \bar{a}_j < 0} \bar{a}_j (x_j - c_j)$$

from which it follows

$$a^T x + q(p) \leq \varepsilon \sum_{j=1}^{m} (c_j - l_j)$$

and the right hand inequality in (32) is established. To prove the left hand inequality we note that by definition

$$q(p) = \begin{array}{ll} Max & p^T E \xi - a^T \xi \\ subject\ to & \xi \in H \end{array},$$

where H is given by (4), from which it follows

$$q(p) \geq \begin{array}{ll} Max & p^T E \xi - a^T \xi \geq -a^T x, \\ subject\ to & \xi \in H, E\xi = 0 \end{array}$$

where the last inequality holds since $x \in H$ and $Ex = 0$. Q.E.D.

A primal dual pair satisfying the conditions of Propositon 9 may be viewed as an optimal solution to a perturbed problem whereby each cost coefficient a_j is perturbed by an amount not exceeding ε. Since we are dealing with linear programs, it is easily seen that if ε is sufficiently small then every solution of the perturbed primal problem is also a solution of the original primal problem. Therefore, for sufficiently small ε, the relaxation method with the ε-CS modification terminates in a finite number of iterations with an optimal solution to **(P)**. The required size of ε for this to occur may be estimated by

$$min \ \{ \ a^T x - a^T x^* \mid x \ a \ basic \ feasible \ solution \ of \ (\mathbf{P}), \ a^T x - a^T x^* \neq 0 \ \}$$

divided by $\Sigma_j (c_j - l_j)$, where x^* denotes any optimal solution to **(P)**.

6. Application to Network Flow Problems with Side Constraints

In this section we consider the special case of (P) where E is the node-arc incidence matrix for a directed network. We show that by exploiting the network structure we can efficiently implement the tableau pivots.

To improve the efficiency of tableau pivoting we will use the factorization scheme suggested by Rockafellar ([13], Ch. 10F) for the two tier linear homogeneous system

$$\begin{bmatrix} F \\ F_0 \end{bmatrix} x = 0, \tag{36}$$

where

$$F_0 \ x = 0 \tag{37}$$

will be called the auxiliary system. Consider any Tucker tableau representing (36)

$$\begin{bmatrix} A' \\ A'' \end{bmatrix},$$

where we have partitioned the tableau row-wise to correspond to a partition of its basis such that the indexes corresponding to the rows of A" form a basis for the auxiliary system (37). Thus if we let B", B' and

N denote the subset of indexes that are respectively basic for (37), nonbasic for (37) but basic for (36), and nonbasic for (36), then

$$x_{B'} = A'x_N \ , \ x_{B''} = A''x_N \ , \ x_{B''} = A'_0 x_{B'} + A_0 x_N , \tag{38}$$

where

$$[\ A'_0 \quad A_0\ ]$$

denotes the Tucker tableau representing (37) for which B" is a basis and whose columns are partitioned to correspond to B' and N respectively. Combining the first and the last equality in (38) we obtain

$$x_{B''} = (A'_0 A' + A_0)x_N ,$$

implying

$$A'' = A'_0 A' + A_0 . \tag{39}$$

Equation (39) allows us, instead of maintaining A' and A", to maintain A', A_0' and A_0 and compute the columns of A" only when needed. This is computationally attractive if A" is a dense matrix while A_0' and A_0 are sparse matrices. Note when a pivot involves making an element of B' nonbasic, only A' needs to be updated, while if a pivot involves making an element of B" nonbasic then after updating A' it will be necessary to also update A_0' and A_0. In the latter case it may be necessary to also exchange an element of B' with the index just pivoted into B" so to maintain B" as a basis for the auxiliary system (37). We will apply the factorization scheme to the system (24) with F = [0 -I D] and F_0 = [-I 0 E] . This choice of factorization allows us to efficiently store A_0' and A_0 in terms of a spanning tree on a network and to avoid storing A", which has a large number of rows and is not necessarily sparse. Note we may alternately apply the factorization scheme with F = [-I 0 E] and F_0 = [0 -I D] , although this does not appear to yield any interesting result.

We will assume E is the node-arc incidence matrix for a connected, directed ordinary network G, whose nodes are numbered from 1 to n and whose arcs are numbered from n + n' + 1 to n + n' + m. In other words,

$$e_{ij} = \begin{cases} 1 & \text{if there exists an arc } n+n'+j \text{ leaving node } i, \\ -1 & \text{if there exists an arc } n+n'+j \text{ entering node } i, \\ 0 & \text{otherwise.} \end{cases}$$

Let us add to G a node n + 1 and an arc i joining node n + 1 to each node i (i = 1,2,...,n) and call the resulting network G' (we will refer to the nodes and the arcs by their numbers). We first give a network characterization of [A_0' A_0]. Since [A_0' A_0] is a Tucker representation of

$$\begin{bmatrix} -I & 0 & E \end{bmatrix} \begin{bmatrix} w \\ y \\ z \end{bmatrix} = 0 \tag{40}$$

(the columns of [-I 0 E] we index from 1 to n + n' + m) and its dual, whose basis is characterized by a spanning tree T on G' rooted at node n + 1 (see [13], Ch. 10C), we can compute the nonzero columns of [A_0' A_0] directly using T. More precisely, the arcs in T form B" and the arcs not in T form (B'UN)\{indexes corresponding to the zero columns in (40)}. Entries in the nonzero columns of [A_0' A_0] are given by (letting a_{ij}^0 denote the entry correspond to arc i in T and arc j not in T)

$$a_{ij}^0 = \begin{cases} 1 & \text{if } i \text{ is in unique cycle of } T \cup \{j\} \text{ and is oriented in same direction as } j, \\ -1 & \text{if } i \text{ is in unique cycle of } T \cup \{j\} \text{ and is oriented in opposite direction as } j, \\ 0 & \text{otherwise.} \end{cases}$$

We will assume T is stored in an array whose kth entry records the arc joining node k to its immediate predecessor in T. To compute a_{ij}^0 for each arc i in T and each arc j not in T requires tracing backwards along the path from the root to each end node of j in T to see if the unique cycle of $T \cup \{j\}$ contains i. The greatest work however comes from performing a pivot where we exchange an element of B" for an element of N and the resultant B" does not form a basis for the auxiliary system (40), in which case we have to find an element of B" and an element of B' whose exchange would make B" a basis. More precisely, let the arc of G' that is to be pivoted out of B" into N (out of N into B") be denoted i (j) and suppose the unique cycle in $T \cup \{j\}$ does not contain i. Then there exist an arc k in B' that connects the two components of $T \setminus \{i\}$ (since (B'UB"U{j})\{i} forms a basis for (36), it must contain a subset that is a basis for (40)). Then (B"U{k})\{i} forms a basis for (40) and (B'UB"U{j})\{i} forms a basis for (36). However, to find such k requires searching through the elements of B' to find an arc that joins the two components of $T \setminus \{i\}$. A simple scheme for this is to, for each element k in B', trace backwards along the path in T from the root to each end node of arc k to see if the unique cycle of $T \cup \{k\}$ contains i (if yes, then we use k). However, if n' = (cardinality of B') is large or if T has large depth then more sophisticated data structures and algorithms would be needed. Techniques used by dual simplex methods may be useful here because finding a k that connects the two components of $T \setminus \{i\}$ is analogous to a dual simplex pivot.

We remark that D may possess special structure which makes solving (15) easier. One such example is the multicommodity network flow problem [1], [7] for which E has the form

$$E = \begin{bmatrix} E'' & & & \\ & E'' & & \\ & & \ddots & \\ & & & E'' \end{bmatrix},$$

where E", say n" by m", is the node-arc incidence matrix for some directed network and D has the form

$$D = \left[\begin{array}{cccc} I & I & \ldots & I \end{array} \right],$$

where I denotes the m" x m" identity matrix. Subproblem (15) then separates into m" disjoint subproblems, where the jth (j = 1,2,...,m") subproblem has the form

$$\begin{aligned} Max \quad & \sum_{r=1}^{K} ((E_j'')^T p^r - a_j^r) x_j^r \\ subject\ to \quad & x_j^1 + x_j^2 + \ldots + x_j^K = b_j \\ & l_j^r \leq x_j^r \leq c_j^r \quad , \ r=1,\ldots,K, \end{aligned}$$

and $K \equiv m/m''$ denotes the total number of distinct commodities, E_j'' denotes the jth column of E", x_j^r denotes the flow of the rth commodity on arc j, p^r denotes the price vector associated with the constraint $E''x^r = 0, \ldots$ etc. In general, if D is block diagonal with K blocks along its diagonal then (15) decomposes into K subproblems.

7. Conclusion and Extensions

We have described a dual descent method that extends the linear programming method in [15] to incorporate decomposition. Two types of modifications were proposed to ensure finite convergence of the method – one an out-of-kilter type and the other based on ε-CS. In the special case of network flow problems with side constraints the method can be efficiently implemented using network data structures.

Our method can also be extended to directly handle inequality side constraints of the form

$$D x \leq b$$

(instead of first transforming the inequalities into equalities). In this case the dual variables associated with the side constraints would be constrained to be nonnegative. Although our method may be interpreted as a primal dual method, it is quite different from that of Jewell [6] for multicommodity network flow. His method solves the restricted primal subproblem, itself a multicommodity network flow problem, by enumerating all possible flow augmenting paths and then solving the path formulation of the restricted primal subproblem (this method of solving the subproblem is equivalent to Dantzig-Wolfe decomposition

[4] with an artificial starting basis for the master problem and with only the columns of negative reduced cost, corresponding to the flow augmenting paths, being used in the master problem). In contrast our method emphasizes dual coordinate descent, does not solve any restricted primal subproblem, and generates descent directions by applying the painted index algorithm of Rockafellar.

Solution of the dual problem in the unconstrained form of **(D)** has been well studied. However in most of the approaches people used variants of either the subgradient method or the steepest descent method (i.e. the primal dual method) or methods of descent along edges of epi(q). Each of these methods has some serious drawback – the first is not a dual descent method, the second requires large computational effort to find each descent direction, and the third requires many dual descents when epi(q) has many edges. Our method on the other hand is a dual descent method that places very few restrictions on the choice of descent directions other than that they be generated from the Tucker representations of certain extended subspace. This dual descent emphasis contrasts our method with methods, such as the primal dual method, that emphasize moving the primal vector towards feasibility.

Appendix A

In this appendix we demonstrate how, for each p and u, the ψ satisfying (19), i.e.

$$\phi(p-\lambda u) = \phi(p) - \lambda\psi \qquad \textit{for } \lambda > 0 \ \textit{ sufficiently small,}$$

may be computed, where $-\phi(p)$ denotes the optimal dual solution to the following subproblem [cf. (15)]

$$\begin{aligned} &\textit{Maximize} && (E^T p - a)^T x \\ &\textit{subject to} && Dx = b\,, \quad l \le x \le c\,. \end{aligned} \tag{Q(p)}$$

Let p' denote p perturbed in the direction -u and let D be partitioned into $[D_B\ D_N]$ where B denotes the optimal basis for Q(p'). The perturbed problem Q(p') is no harder to solve than Q(p) using standard perturbation techniques such as lexicographical ordering. Then, for sufficiently small positive λ, B is an optimal basis for Q(p–λu) and the corresponding optimal dual variable is given by

$$\phi(p-\lambda u) = (D_B^{-1})^T r_B\,, \tag{A.1}$$

where r denotes the negative of the cost vector of Q(p– λu), i.e.

$$r = a - E^T(p-\lambda u) \tag{A.2}$$

and r is partitioned into (r_B, r_N) corresponding to the partitioning of D into $[D_B\ D_N]$. Combining (A.1) with (A.2) we obtain

$$\phi(p-\lambda u) = (D_B^{-1})^T(a - E_B^T(p-\lambda u)) = (D_B^{-1})^T(a-E_B^T p) + \lambda(D_B^{-1})^T E_B^T u = \phi(p) + \lambda(E_B D_B^{-1})^T u\,,$$

where E is partitioned into $[E_B\ E_N]$ corresponding to the partitioning of D into $[D_B\ D_N]$. It follows

$$\psi = -(E_B D_B^{-1})^T u\,.$$

In this way we can update ϕ(p–λu) recursively as λ increases from zero by updating ψ at every λ for which the optimal basis for Q(p– λu) changes.

Appendix B

In this appendix we prove Proposition 3 - that the number of dual descent steps between consecutive primal rectifications is finite under the out-of-kilter modification of §5.1. We will argue by contradiction. Suppose an infinite number of dual descents occur between some two consecutive primal rectifications. Since the primal variable x does not change during this infinite sequence of iterations [cf. note at end of §5.1], each of which terminates in a dual descent, the only indexes that can change colour during these iterations are those indexes $n+n'+j$ for which $x_j = l_j$ (each such index can change colour between red and black) and those indexes $n+n'+j$ for which $x_j = c_j$ (each such index can change colour between red and white). The colour of the other indexes remain unchanged. Let s denote the lever index used during these iterations [cf. modification 3]. s must be painted either black only or white only during all these iterations and for simplicity assume s is painted white (analogous argument holds for case where s is black). In the modified painted index algorithm, since each green index once pivoted into row position remains in row position from then on, each red index once pivoted into column position never returns to row position painted red, each red index remaining in row position never changes colour and the final tableau of each iteration is used as the initial tableau for the iteration that follows [cf. modification 3], we conclude the following :

Observation 1 After a while, no pivot involves making a basic red or green index nonbasic or a nonbasic red or green index basic.

Now consider what occurs at the end of each relaxation iteration. The price vector changes value and, as a result of this change, at least one nonbasic red index of the type $n+n'+j$, where $j \in \{1,...,m\}$, becomes either black or white (at the same time some nonbasic black or white index of the same type may become red). Then at the next relaxation iteration a pivot is performed to make one such index basic (since the same lever index s is used). We thus conclude the following :

Observation 2 Immediately before each dual descent there is at least one nonbasic index of the type $n+n'+j$, for some $j \in \{1,...,m\}$. At the beginning of each iteration one such nonbasic index becomes basic.

In what follows we will assume that enough time has passed so the statement in Observation 1 holds. Then during each pivot either (a) a basic black or white index of the type i, where $i \in \{1,...,n\}$, is made nonbasic or (b) a basic black or white index of the type $n + n' + j$, where $j \in \{1,...,m\}$, is made nonbasic. Since the number of nonbasic indexes of the type $n + n' + j$, where $j \in \{1,...,m\}$, is not increased in case (a) it follows from Observation 1 and Observation 2 that there must be an infinite number of pivots for the form (b) (otherwise Observation 1 implies that after a while every pivot must be making nonbasic black or white indexes of the type i (for some $i \in \{1,...,n\}$) basic, which contradicts Observation 2).

Now consider the time just before a pivot of the form (b) takes place. Let r denote the nonbasic index that is to be made basic during the pivot step (r is either black or white by Observation 1) and, for each basic index w, let a_{wr} denote the entry in row indexed by w and column indexed by r of the current Tucker tableau. We need to consider two cases: case (1) $r = i^*$ for some $i^* \in \{1,...,n\}$ and case (2) $r = n + n' + j^*$ for some $j^* \in \{1,...,m\}$. In case (1) we obtain, applying Lemma B (stated at the end of this appendix) with

$$W = \begin{bmatrix} I & 0 & E \\ 0 & I & D \end{bmatrix} \quad and \quad v = (p, \phi(p), t),$$

that

$$-p_{i^*} = \sum_{\substack{w=i \\ for\ some\ i \in \{1,..,n\}}} a_{wr} p_i + \sum_{\substack{w=n+n'+j \\ for\ some\ j \in \{1,..,m\}}} a_{wr} t_j + \sum_{\substack{w=n+k \\ for\ some\ k \in \{1,..,n'\}}} a_{wr} \Phi_k(p),$$

where $\phi(p)$ is given by (12) or (16) and t is given by (14).

Using Observation 1 we obtain $a_{wr} = 0$ for all basic red indexes w, so the equation above is equivalent to

$$-p_{i^*} = \sum_{\substack{w=i \\ for\ some\ i \in \{1,..,n\} \\ w\ black\ or\ white}} a_{wr} p_i + \sum_{\substack{w=n+n'+j \\ for\ some\ j \in \{1,..,m\} \\ w\ not\ red}} a_{wr} t_j . \tag{B.1}$$

Since in a pivot step of the form (b) we make nonbasic a basic black or white index of the type $n + n' + j$, for some $j \in \{1,...,m\}$, which [cf. modification 2] has lower priority than any basic black or white index of the type i, for some $i \in \{1,...,n\}$, it follows from the pivoting rule in [13], pp. 476 (assuming for simplicity that i^* is white)

$$either \quad a_{wr} \geq 0 \ and \ w \ is \ white \quad or \quad a_{wr} \leq 0 \ and \ w \ is \ black, \tag{B.2}$$

for all basic indexes w such that $w = i$ for some $i \in \{1,...,n\}$.

For notational convenience define $a_{i*r} = 1$. We can then write (B.1) as (using the fact that $t_j = a_j$ if $n + n' + j$ is not painted red)

$$\sum_{\substack{w=i \\ \text{for some } i\in\{1,..,n\} \\ w \text{ black or white}}} -a_{wr}p_i = \sum_{\substack{w=n+n'+j \\ \text{for some } j\in\{1,..,m\} \\ w \text{ not red}}} a_{wr}a_j . \tag{B.3}$$

Each term $-a_{wr}p_i$ on the left hand side of (B.3) monotically increases during each dual descent ((26) and (29) imply each p_i can only decrease (increase) during a dual descent if i is painted white (black)). Furthermore since r indexes the pivoting column and s indexes the lever row, it must be $a_{sr} \neq 0$, so $-a_{sr}p_s$ strictly increases during the dual descent (since p_s strictly decreases at each dual descent). It follows the left hand side of (B.3) strictly increases at each dual descent. Since (B.3) holds whenever the same lever index, Tucker tableau and index painting occur in the iterations while the right hand side of (B.3) is constant with respect to the same lever index,Tucker tableau and index painting, it follows the same lever index, Tucker tableau and index painting cannot recur in the iterations. This is a contradiction since the lever index is fixed in our argument while the number of distinct Tucker tableaus and index paintings is finite. In case (2) a contradiction is similarly obtained using an analogous argument. Q.E.D.

Lemma B Consider a real matrix W of full row rank and a Tucker tableau A representing the subspace pair $\{ z \mid Wx = 0 \}, \{ v \mid v = W^T u \}$. Then for any vector v satisfying $v = W^T u$ for some u, we have $-v_j = v_B{}^T A_j$ for all nonbasic indexes j, where v_j denotes the jth component of v, A_j denotes the column of A indexed by j, B denotes the basis associated with A and v_B denotes the vector with components v_j, $j \in B$.

Proof : From the definition of Tucker tableaus ([13], Ch. 10) we have $A = -W_B{}^{-1}W_N$, for some partition of W into $[W_B\ W_N]$ for which W_B is invertible. Using the definition of basic and nonbasic indexes we correspondingly partition v into (v_B, v_N), where v_N denotes the vector with components v_j, j nonbasic. It follows $(v_B, v_N) = W^T u = (W_B{}^T u, W_N{}^T u)$, so for all nonbasic indexes j,

$$v_j = W_j^T u = u^T W_j = (u^T W_B)(W_B{}^{-1} W_j) = (v_B^T)(-A_j) .$$

Q.E.D.

Appendix C

In this appendix we prove that if ε is a nonnegative scalar then for any $p\in\mathbb{R}^n$, $u\in\mathbb{R}^n$, and $\pi\in\mathbb{R}^{n'}$, $q'(p-\alpha u;-u) \le -C^\varepsilon(u,\pi,p)$ for all $\alpha\in[0,\varepsilon/\theta(u)]$ where $\theta(u) \equiv \max\{\|v\|_\infty \mid v$ given by (18) and (19) for some $p\}$.

Proof: Let y be a primal vector that satisfies ε-CS with p, let α be a fixed stepsize in $[0,\varepsilon/\theta(u)]$, and let x be a primal vector that satisfies CS with $p-\alpha u$. For notational simplicity let $p' = p-\alpha u$ and let $t' = E^T p' + D^T\phi(p')$, $t = E^T p + D^T\phi(p)$. Then [cf. (30a) and (30b)]

$$C^\varepsilon(u,\pi,p) = \sum_{a_j - t_j > \varepsilon} v_j l_j + \sum_{\substack{|a_j - t_j|\le\varepsilon \\ v_j>0}} v_j l_j + \sum_{a_j - t_j < -\varepsilon} v_j c_j + \sum_{\substack{|a_j - t_j|\le\varepsilon \\ v_j<0}} v_j c_j - \pi^T b, \tag{C.1}$$

where

$$v = E^T u + D^T \pi, \tag{C.2}$$

and also [cf. (17)]

$$-q'(p';-u) = \sum_{a_j - t_j > 0} v_j l_j + \sum_{\substack{a_j - t_j = 0 \\ v_j>0}} v_j l_j + \sum_{a_j - t_j < 0} v_j c_j + \sum_{\substack{a_j - t_j = 0 \\ v_j<0}} v_j c_j - \Psi^T b, \tag{C.3}$$

where

$$v = E^T u + D^T \Psi, \tag{C.4}$$

and Ψ satisfies

$$\phi(p'-\lambda u) = \phi(p') - \lambda\Psi \qquad \textit{for } \lambda>0 \textit{ sufficiently small}.$$

Using (C.1), (C.2) and the fact

$$y_j = \begin{cases} l_j & \forall j \ni a_j - t_j > \varepsilon \\ c_j & \forall j \ni a_j - t_j < -\varepsilon \end{cases} \qquad \textit{and} \quad Dy - b = 0 \tag{C.5}$$

we obtain

$$C^\varepsilon(u,\pi,p) = \sum_{a_j - t_j > \varepsilon} v_j l_j + \sum_{|a_j - t_j|\le\varepsilon} v_j y_j + \sum_{a_j - t_j < -\varepsilon} v_j c_j - \pi^T b$$

$$+ \sum_{\substack{|a_j - t_j|\le\varepsilon \\ v_j>0}} v_j (l_j - y_j) + \sum_{\substack{|a_j - t_j|\le\varepsilon \\ v_j<0}} v_j (c_j - y_j)$$

$$= \sum_{a_j - t_j > \varepsilon} (E_j^T u) l_j + \sum_{|a_j - t_j| \le \varepsilon} (E_j^T u) y_j + \sum_{a_j - t_j < -\varepsilon} (E_j^T u) c_j$$

$$+ \sum_{\substack{|a_j - t_j| \le \varepsilon \\ v_j > 0}} v_j (l_j - y_j) + \sum_{\substack{|a_j - t_j| \le \varepsilon \\ v_j < 0}} v_j (c_j - y_j) \,. \qquad \text{(C.6)}$$

On the other hand x satisfies CS with p' and therefore

$$x_j = \begin{cases} l_j & \forall j \ni a_j - t'_j > 0, \\ c_j & \forall j \ni a_j - t'_j < 0, \end{cases} \qquad and \quad Dx - b = 0 \,. \qquad \text{(C.7)}$$

From the definition of $\theta(u)$ we have

$$a_j - t'_j > 0 \quad if \; a_j - t_j > \varepsilon,$$
$$a_j - t'_j < 0 \quad if \; a_j - t_j < -\varepsilon,$$

which together with (C.7) yields

$$x_j = \begin{cases} l_j & \forall j \ni a_j - t_j > \varepsilon \\ c_j & \forall j \ni a_j - t_j < -\varepsilon \end{cases} \qquad and \quad Dx - b = 0 \,. \qquad \text{(C.8)}$$

Then combining (C.3), (C.4) with (C.8) we obtain

$$-q'(p'; -u) = \sum_{a_j - t_j > \varepsilon} (E_j^T u) l_j + \sum_{|a_j - t_j| \le \varepsilon} (E_j^T u) x_j + \sum_{a_j - t_j < -\varepsilon} (E_j^T u) c_j \,. \qquad \text{(C.9)}$$

Combining (C.6) with (C.9) and we obtain

$$C^{\varepsilon}(u,\pi,p) + q'(p'; -u) = \sum_{|a_j - t_j| \le \varepsilon} (E_j^T u)(y_j - x_j) + \sum_{\substack{|a_j - t_j| \le \varepsilon \\ v_j > 0}} v_j (l_j - y_j) + \sum_{\substack{|a_j - t_j| \le \varepsilon \\ v_j < 0}} v_j (c_j - y_j) \,.$$

Since b-Dy = 0 and b-Dx = 0 we have D(y-x) = 0, so $\pi^T D(y-x) = 0$, and using (C.5) and (C.8) we obtain

$$0 = \sum_{|a_j - t_j| \le \varepsilon} (D_j^T \pi)(y_j - x_j) \,. \qquad \text{(C.10)}$$

Adding (C.10) to $C^{\varepsilon}(u,\pi,p) + q'(p';-u)$ gives

$$C^{\varepsilon}(u,\pi,p) + q'(p'; -u) = \sum_{|a_j - t_j| \le \varepsilon} v_j (y_j - x_j) + \sum_{\substack{|a_j - t_j| \le \varepsilon \\ v_j > 0}} v_j (l_j - y_j) + \sum_{\substack{|a_j - t_j| \le \varepsilon \\ v_j < 0}} v_j (c_j - y_j)$$

$$= \sum_{\substack{|a_j - t_j| \le \varepsilon \\ v_j > 0}} v_j (l_j - x_j) + \sum_{\substack{|a_j - t_j| \le \varepsilon \\ v_j < 0}} v_j (c_j - x_j) . \qquad \text{(C.11)}$$

The right hand side of (C.11) is nonpositive and so it follows

$$q'(p'; -u) \le -C^{\varepsilon}(u, \Pi, p) .$$

Q.E.D.

Appendix D

In this appendix we give a numerical example to illustrate the essential features of the relaxation method of §4 using the ε-CS modification of §5.2. We will show that, during a dual descent step, even if v as given by (18) and (19) changes at stepsizes that are arbitrarily close to zero, the line minimization stepsize remains bounded away from zero.

Consider the following linear program :

$$\begin{aligned} &\textit{Minimize} && -x_2 \\ &\textit{subject to} && -x_1 - 2x_2 = 0 \\ &&& x_1 + x_2 + x_3 = M/2 \end{aligned}$$

$$-M \le x_1 \le M\ ,\ -M \le x_2 \le M\ ,\ 0 \le x_3 \le M\ ,$$

where M is a sufficiently large scalar. To put into the form of **(P)**, we set a = (0, -1, 0), E = [-1 -2 0], D = [1 1 1], b = M/2, $l_1 = l_2 = -M$, $l_3 = 0$, $c_1 = c_2 = c_3 = M$.

This problem is primal feasible since (-M, M/2, M) is a feasible primal vector. Let $\varepsilon = \frac{1}{2}$ and let the initial price vector $p = -\delta$, where δ is a positive scalar $< \frac{1}{2}$.

Relaxation iteration 1

The subproblem (15) associated with the above linear program is

$$\begin{aligned} &\textit{Maximize} && \delta x_1 + (1+2\delta)x_2 \\ &\textit{subject to} && x_1 + x_2 + x_3 = M/2 \end{aligned}$$

$$-M \le x_1 \le M\ ,\ -M \le x_2 \le M\ ,\ 0 \le x_3 \le M,$$

which has column 1 in the optimal basis, an optimal primal solution of x = (-M/2, M, 0) and optimal dual solution of $\phi(p) = -\delta$ with associated reduced cost $a - t = a - E^T p - D^T \phi(p) = (0, -1-\delta, \delta)$. Then

d = Ex = M/2–2M = -3M/2. Columns 1 and 3 are ε-balanced and column 2 is ε-active so the initial Tucker tableau is :

			z_1	z_2	z_3
			g	r	b
lever row ⟶	w_1	b	-1	-2	0
	y_1	r	1	1	1

r = red
b = black
g = green

The corresponding u, π, and v are respectively [cf. analog of (26)-(28) for black lever index] -1, 0 and (1, 2, 0). Using (31) we have

$$C^e(u,\pi,p) = ud + v_1(l_1 - x_1) = \frac{3}{2}M + (-\frac{M}{2}) = M > 0,$$

and therefore -u = 1 is a dual descent direction at p.

Now we perform a line minimization in the direction of -u. We first compute ψ and v given respectively by (19) and (18). Using Appendix A we obtain $\psi = -uE_B D_B^{-1} = -uE_1 = -1$ so that $v = E^T u + D^T \psi = (1, 2, 0) - (1, 1, 1) = (0, 1, -1)$. The rate of descent is given by [cf. (17)]

$$q'(p;-u) = \psi b - v_2 c_2 - v_3 l_3 = -\frac{1}{2}M - (M) + (0) = -\frac{3}{2}M.$$

The first breakpoint of q in the direction -u from p occurs at stepsize [cf. (23)]

$$\alpha = min\left\{ \frac{a_1 - t_1}{-v_1}, \frac{a_3 - t_3}{-v_3} \right\} = min\{\delta + 1, \delta\} = \delta.$$

Set $p \leftarrow p - \alpha u = 0$

At p = 0 the subproblem (15) with p perturbed in the direction of -u has column 3 in the optimal basis, an optimal primal solution of x = (-M, M, M/2) and optimal dual solution of φ(p) = 0 with associated reduced cost a–t = (0, -1, 0). Since column 3 is in optimal basis we have $\psi = -uE_B D_B^{-1} = -uE_3 = 0$ and $v = E^T u + D^T \psi = (1, 2, 0) - (0, 0, 0) = (1, 2, 0)$. The rate of descent is then given by

$$q'(p;-u) = \psi b - v_1 l_1 - v_2 c_2 = 0 - (-M) - 2(M) = -M$$

and therefore -u is still a direction of descent. The second breakpoint moving in the direction -u occurs at stepsize [cf. (23)]

$$\alpha = min \left\{ \frac{a_2 - t_2}{-v_2} \right\} = \frac{1}{2} .$$

Set $p \leftarrow p - \alpha u = \frac{1}{2}$.

At $p = \frac{1}{2}$ the subproblem (15) with p perturbed in the direction of -u has column 2 in the optimal basis, an optimal primal solution of $x = (-M, M/2, M)$ and optimal dual solution of $\phi(p) = 0$ with associated reduced cost $a - t = (\frac{1}{2}, 0, 0)$. Since column 2 is in optimal basis then $\psi = -uE_B D_B^{-1} = -uE_2 = -2$ and $v = E^T u + D^T \psi = (1, 2, 0) - (2, 2, 2) = (-1, 0, -2)$. The rate of descent is then

$$q'(p; -u) = \psi b - v_1 l_1 - v_3 c_3 = -M + (-M) + 2M = 0 ,$$

which is nonnegative and so line minimization terminates. The current deficit is $Ex = -(-M) - 2(M/2) = 0$ so relaxation method terminates.

The important observation to make in this example is that the first breakpoint in a dual descent step at which v changes value can occur arbitrarily close to the starting price vector (in our example δ can be arbitrarily small) yet despite this change the line minimization stepsize is still bounded away from zero [cf. Lemma 1].

References

[1] Assad, A. A., "Multicommodity Network Flows - A Survey", *Networks*, Vol. 8, pp. 37-91 (1978).

[2] Bertsekas, D. P., "A Unified Framework for Minimum Cost Network Flow Problems," *Mathematical Programming*, Vol. 32, pp. 125-145 (1985).

[3] Bertsekas, D. P. and Tseng, P., "Relaxation Methods for Minimum Cost Ordinary and Generalized Network Flow Problems", LIDS Report P-1462, Mass. Institute of Technology (May 1985; revised September 1986), *Operations Research J.* (to appear).

[4] Dantzig, G. B., *Linear Programming and Extensions*, Princeton Univ. Press, Princeton, N.J. (1963).

[5] Golden, B. and Magnanti, T.L., *Network Optimization*, currently in draft form.

[6] Jewell, W. S., "A Primal Dual Multicommodity Flow Algorithm", Operations Research Center Report 66-24, University of California, Berkeley (1966).

[7] Kennington, J. L., "A Survey of Linear Cost Multicommodity Network Flows", *Operations Research*, Vol. 26, No. 2, pp. 209-236 (1978).

[8] Magnanti, T., "Optimization for Sparse Systems", in *Sparse Matrix Computations* (J. R. Bunch and D. J. Rose, eds), Academic Press, New York, pp. 147-176 (1976).

[9] Magnanti, T. and Golden, B., "Deterministic Network Optimization: A Bibliography", *Networks*, Vol. 7, pp. 149-183 (1977).

[10] Rockafellar, R. T., *Convex Analysis*, Princeton Univ. Press (1970).

[11] Rockafellar, R. T., "Monotropic Programming: Descent Algorithms and Duality", in *Nonlinear Programming 4*, by O. L. Mangasarian, R. Meyer, and S. Robinson (eds.), Academic Press, pp. 327-366 (1981).

[12] Rockafellar, R. T., "The Elementary Vectors of a Subspace of R^N", in *Combinatorial Mathematics and Its Applications*, by R. C. Bose and T. A. Dowling (eds.), The Univ. of North Carolina Press, Chapel Hill, N. C., pp. 104-127 (1969).

[13] Rockafellar, R. T., *Network Flows and Monotropic Programming*, Wiley-Interscience (1983).

[14] Tarjan, R. E., *Data Structures and Network Algorithms*, Society for Industrial and Applied Mathematics, Philadelphia (1983).

[15] Tseng, P. and Bertsekas, D. P., "Relaxation Methods for Linear Programming Problems," *Mathematics of Operations Research*, Vol. 12, pp. 1-28 (1987).

[16] Tseng, P., "Relaxation Methods for Monotropic Programming Problems," Ph. D. Thesis, Operations Research Center, MIT (1986).