CENTER FOR
PARALLEL OPTIMIZATION

A STRUCTURED INTERIOR POINT METHOD

by

Gary L. Schultz and Robert R. Meyer

Computer Sciences Technical Report #934

May 1990

A Structured Interior Point Method

Gary L. Schultz Robert R. Meyer

Abstract

We develop a structured interior point method for block angular optimization and describe the convergence properties of the method. Excellent computational results are presented for a class of large-scale linear programming models. These models are multicommodity flow problems that arise from an Air Force MAC application and generate problems as large as 100,000 rows and 300,000 columns.

This work was supported in part by NSF grant CCR-8709952 and AFOSR grant 89-0410.

1 Block Angular Optimization

A block angular optimization problem is defined as the following:

Given a smooth, convex function $c : \mathbb{R}^N \rightarrow \mathbb{R}$,
find a minimizer x^* of c subject to

$$\begin{pmatrix} A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_K \\ D & & & \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_K \end{pmatrix} \begin{matrix} = \\ = \\ \vdots \\ = \\ \leq \end{matrix} \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_K \\ d \end{pmatrix} \qquad (1)$$

and bounds $\ell \leq x \leq u$.

The dimension of the kth block A_k is $M_k \times N_k$ and we define $M := \sum_k M_k$ and $N := \sum_k N_k$. The dimension of D is $J \times N$. Note that linear multicommodity problems are an important subclass of this class of problems.

We assume that solving a set of K linear subproblems:

$$\left.\begin{array}{ll} \text{minimize} & \tilde{c}_k x_k \\ \text{subject to} & A_k x_k = \tilde{b}_k \\ & \tilde{\ell}_k \leq x_k \leq \tilde{u}_k \end{array}\right\} \qquad \text{for } k = 1, \ldots, K \tag{2}$$

is easy relative to solving the entire original problem (1). This is quite reasonable for most applications, since the problems (2) are independent subproblems, and may, therefore, be solved in parallel. This is particularly appropriate in a MIMD computational environment. Secondly, some types of structure in the A_k may be exploited which could not be directly used if (1) were solved with the entire set of constraints. An example of this is the multicommodity flow problem, where each A_k is a node-arc-incidence matrix. In this case (2) may be solved with a special purpose network code. And thirdly, we note that the difficulty of solving most linear programs (in some sense the easiest of optimization problems) in practice increases as a quadratic or cubic function with the size of the problem, so that it is much more efficient to solve a set of small problems than a single aggregate problem.

2 The Decomposition Scheme

In this section we describe a scheme that allows us to deal with the block constraints explicitly and the coupling constraints implicitly via a barrier function. Define the set of feasible points for the block constraints by

$$\mathcal{B} := \{x | Ax = b \text{ and } \ell \leq x \leq u\},$$

and the set of feasible points for the coupling constraints by

$$\mathcal{C} := \{x | Dx \leq d\}.$$

The algorithm begins by finding x^0 as the solution of a relaxed problem

$$\underset{x}{\text{minimize}} \tilde{c}x \text{ subject to } x \in \mathcal{B}. \tag{3}$$

Here $\tilde{c}$ could approximate the gradient of the original cost function $\nabla c(\tilde{x})$ at some point $\tilde{x}$, but we require only that $x^0 \in \mathcal{B} \cap \operatorname{dom} c$. The case where $\operatorname{dom} c \neq \mathbb{R}^N$ will not be considered further in this paper, as it produces technical difficulties without being enlightening. In subsequent iterations, the algorithm solves additional subproblems with block constraints, and then does a smaller (typically K-dimensional) search to coordinate the solutions of the subproblems, forming a new point in $\mathcal{B}$. During this process information about $\mathcal{C}$ is introduced into the objective function by using a barrier function.

We shall introduce the barrier function, discuss its fundamental properties and show how to obtain feasible points in section 2.1. Section 2.2 will show how to generate a sequence of feasible points that approximates the solution. Section 2.3 then summarizes the decomposition method.

2.1 Shifting Barriers to Obtain Feasibility

Define the *shifted logarithmic barrier function*

$$\rho : \mathbb{R}^N \times \mathbb{R}_{>0} \times \mathbb{R}^J \to \mathbb{R} \cup +\infty$$

by

$$\rho(x, \tau, \theta) := \begin{cases} -\tau \sum_{j=1}^{J} \ln(\theta_j - D_j x) & \text{if } \theta > Dx \\ +\infty & \text{otherwise} \end{cases}$$

and let

$$f(x, \tau, \theta) := c(x) + \rho(x, \tau, \theta)$$

denote the original objective function augmented by the shifted logarithmic barrier function. Also define the corresponding *barrier problem* as

$$\mathcal{P}(\tau, \theta): \qquad \underset{x}{\text{minimize}}\, f(x, \tau, \theta) \text{ subject to } x \in \mathcal{B},$$

i.e., $\mathcal{P}(\tau, \theta)$ represents the optimization problem with parameters $\tau > 0$ and $\theta \in \mathbb{R}^J$. ρ was defined so that $\rho(\cdot, \tau, \theta)$ is a barrier function modeling the constraints $Dx < \theta$. Thus, for $\theta = d$, we have $\operatorname{dom} \rho(\cdot, \tau, d) = \operatorname{int} \mathcal{C}$. Allowing $\theta \neq d$ has the property of "shifting" the barrier, hence the name.

Once an initial point $x^0 \in \mathcal{B}$ has been found by solving the relaxed problem (3), the parameters θ^1 and τ^1 are chosen so that $\tau^1 > 0$ and $\theta^1 > Dx^0$. This will have the effect of making $x^0 \in \mathcal{B}$ an interior point

of $\operatorname{dom} f(\cdot, \tau^1, \theta^1)$. We then compute x^1 by approximately minimizing the barrier problem $\mathcal{P}(\tau^1, \theta^1)$. In general, if $Dx^i < d$, we have produced a feasible point. If not, then we choose θ^{i+1} as described below while maintaining $\tau^{i+1} = \tau^i$, then set $i \leftarrow i+1$ and do the process again. We will prove that if a point $x \in \mathcal{B} \cap \operatorname{int} \mathcal{C}$ exists, then *such a point is generated in a finite number of iterations*, under appropriate assumptions given below.

Suppose first that our θ are chosen so that

$$Dx^i < \theta^{i+1} \leq \theta^i \qquad \text{and} \qquad \theta^i \geq d. \tag{4}$$

This implies that $\theta^\infty := \lim_{i \to \infty} \theta^i$ is well defined. We make one more assumption on our choice of θ:

$$\text{either} \quad \theta^\infty = d \quad \text{or} \quad \exists j \text{ such that } \liminf_i \left(\theta^i - D_j x^i\right) = 0. \tag{5}$$

In order to develop expressions for derivatives of ρ needed in the convergence proof, let

$$q(x, \tau, \theta) := \left(\frac{\tau}{\theta_1 - D_1 x}, \cdots, \frac{\tau}{\theta_J - D_J x} \right) \tag{6}$$

and

$$Q(x, \tau, \theta) := \begin{pmatrix} \dfrac{\tau}{(\theta_1 - D_1 x)^2} & 0 & \cdots & 0 \\ 0 & \dfrac{\tau}{(\theta_2 - D_2 x)^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \dfrac{\tau}{(\theta_J - D_J x)^2} \end{pmatrix}.$$

Provided $Dx < \theta$, we have

$$\begin{aligned} \nabla_x \rho(x, \tau, \theta) &= q(x, \tau, \theta) D & (7) \\ \nabla_{xx} \rho(x, \tau, \theta) &= D^\top Q(x, \tau, \theta) D & (8) \\ \nabla_\theta \rho(x, \tau, \theta) &= -q(x, \tau, \theta), \text{ and} & (9) \\ \nabla_{\theta\theta} \rho(x, \tau, \theta) &= Q(x, \tau, \theta). & (10) \end{aligned}$$

A function ϕ is said to be *essentially smooth* (see section 26 of Rockafellar [10]) if $\operatorname{dom} \phi \neq \emptyset$, ϕ is differentiable on $\operatorname{dom} \phi$, and $\lim |\phi(x^i)| = +\infty$ for

all sequences $\{x^i\} \subset \operatorname{dom}\phi$ converging to a point $\bar{x} \in \operatorname{bdy}\operatorname{dom}\phi$. We shall denote the restriction of a mapping ϕ to a set S by $\phi|_S$ and if S is a subspace of T, the quotient space of S in T is denoted by T/S. Where y is a point and S is a set, we use $y + S$ to denote $\{y + s | s \in S\}$.

THEOREM 1. *Let ϕ and ψ denote the mappings*

$$x \overset{\phi}{\mapsto} \rho(x, \tau, \theta) \qquad \text{and} \qquad \theta \overset{\psi}{\mapsto} \rho(x, \tau, \theta),$$

respectively, where $\tau > 0$. Let y be any fixed vector in $\mathbb{R}^N$ and let $S := y + \ker D$ and $S^{\div} := y + (\mathbb{R}^N / \ker D)$, so that S and $S^{\div}$ are translated subspaces. Then the following are true:

(i)	$\operatorname{dom}\phi = \{x \mid Dx < \theta\}$	*(vi)*	$\operatorname{dom}\psi = \{\theta \mid Dx < \theta\}$
(ii)	*ϕ is essentially smooth*	*(vii)*	*ψ is essentially smooth*
(iii)	*ϕ is convex*	*(viii)*	*ψ is antitone*
(iv)	*$\phi|_S$ is constant*	*(ix)*	*ψ is strictly convex*
(v)	*$\phi|_{S^{\div}}$ is strictly convex*		

Proof: Clearly (i) and (vi) hold. Then (ii) and (vii) hold from the definition. Since $q(x, \tau, \theta) > \mathbf{0}$ whenever $Dx < \theta$ and $\tau > 0$, (9) shows (viii) to be true. Since $Q(x, \tau, \theta)$ is positive definite whenever $Dx < \theta$ and $\tau > 0$ we see from (8) and (10) that (iii) and (ix) are true. Notice that ρ is constant on $S = y + \ker D$, so that (iv) holds, and we have shown all but (v)

For $x \in S^{\div} \cap \{x | Dx < \theta\}$ there is a 1-to-1 correspondence with $\{s | s < \theta\}$ given by $s = Dx$. (The reader may prove this by showing that the linear mapping $x \mapsto Dx$ is a bijection from $S^{\div}$ to range D.) Therefore, it suffices to show that the mapping

$$s \mapsto -\tau \sum_j \ln(\theta_j - s_j)$$

is strictly convex for $s < \theta$. This is true because its Hessian is given by $Q(x, \tau, \theta)$, which is positive definite. ∎

We let $\mathcal{X}(\tau, \theta) \subset \mathcal{B}$ denote the set of minimizers of $\mathcal{P}(\tau, \theta)$ as a function of $\tau > 0$ and θ, Also let $f^*(\tau, \theta) := f(x, \tau, \theta)$ for $x \in \mathcal{X}(\tau, \theta)$ be the optimal value function. In the case where $\operatorname{dom} f(\cdot, \tau, \theta) \cap \mathcal{B} = \emptyset$, we define $\mathcal{X}(\tau, \theta) = \emptyset$ and $f^*(\tau, \theta) = +\infty$. The following shows that $\mathcal{X}$ is nonempty in the other case.

THEOREM 2. *Suppose $\tau > 0$ and $\theta \in \mathbb{R}^J$ are fixed. If there is some point $z \in \operatorname{dom} f(\cdot, \tau, \theta) \cap \mathcal{B}$, then $\mathcal{X}(\tau, \theta) \neq \emptyset$ and $f^*(\tau, \theta) < +\infty$.*

Proof: Since $\mathcal{B}$ is compact and the level set $\{x | f(x, \tau, \theta) \leq f(z, \tau, \theta)\}$ is closed, the intersection of these two sets is compact. Since $f(\cdot, \tau, \theta)$ is a continuous function, it must attain its minimum on this compact intersection. Clearly this minimum $f^*(\tau, \theta)$ is $< +\infty$. ∎

The next result is quite easy to prove. It is stated as a lemma so that we may refer to it later.

LEMMA 3. *Suppose $\tau > 0$ and the sequences $\{x^i\} \subset \mathbb{R}^N$ and $\{\theta^i\} \subset \mathbb{R}^J$ satisfy $\theta^i_j > D_j x^i$ for each $i = 0, 1, \ldots$ and $j = 1, \ldots, J$. Then the following are equivalent:*

(i) $\rho(x^i, \tau, \theta^i)$ *is bounded for all* $i = 0, 1, \ldots$
(ii) $\|q(x^i, \tau, \theta^i)\|$ *is bounded for all* $i = 0, 1, \ldots$
(iii) $\theta^i_j - D_j x^i$ *is bounded away from 0 for all* $i = 0, 1, \ldots$ *and* $j = 1, \ldots, J$

Proof: One may see from the definitions of ρ and q that (iii) is equivalent with both (i) and (ii). ∎

THEOREM 4. *Suppose $c(\cdot)$ is bounded from below on $\mathcal{B}$. Let θ^{i+1} be chosen to satisfy (4) and (5). Let $x^i \in \mathcal{B}$ be computed so that*

$$f(x^i, \tau, \theta^i) \leq f^*(\tau, \theta^i) + \beta \tag{11}$$

for a constant $\beta > 0$. If a point $z \in \mathcal{B} \cap \operatorname{int} \mathcal{C}$ exists, such a point will be found in a finite number of steps. On the other hand, if no such z exists, then $f(x^i, \tau, \theta^i) \to +\infty$.

(Note that by using a procedure for convex programming that generates lower bounds by solving linear programs, we may actually compute each x^i in a finite number of steps.)

Proof: Assume such a z exists. Then

$$f^*(\tau, \theta^i) \leq f(z, \tau, \theta^i) \leq f(z, \tau, d) < +\infty \quad \forall i$$

where the second inequality follows from antitonicity in θ (part (viii) of theorem 1). Therefore, for all i, $f(x^i, \tau, \theta^i)$ is bounded above by $\beta + f(z, \tau, d)$. By

lemma 3 and because $c(x^i)$ is bounded from below, $\theta_j^i - D_j x^i \geq \gamma > 0 \ \forall i, j$. Therefore, for some finite $\hat{\imath}$, $d_j - D_j x^{\hat{\imath}} \geq \gamma/2 > 0 \ \forall j$, and a point with $Dx^{\hat{\imath}} < d$ has been found in a finite number of iterations. If no such z exists, then (4) and (5) imply that $\theta_j^i - D_j x^i \to 0$ for at least one $j \in \{1, \ldots, J\}$. Lemma 3 then shows that $f(x^i, \tau, \theta^i) \to +\infty$. ∎

Assume from now on that the x^i are computed so that $Dx^i < \theta^i$. We will now give a particular method for computing θ. This is a very simple scheme and more complicated ones may be easily developed. After computing $x^0 \in \mathcal{B}$, set

$$\theta_j^1 \leftarrow \begin{cases} d_j & \text{if } D_j x^0 < d_j \\ D_j x^0 + \Theta & \text{if } D_j x^0 \geq d_j \end{cases} \tag{12}$$

where $\Theta > 0$ is a constant. In general, after computing x^i, set

$$\theta_j^{i+1} \leftarrow \begin{cases} d_j & \text{if } D_j x^i < d_j \\ \lambda_\theta D_j x^i + (1 - \lambda_\theta)\theta_j^i & \text{if } D_j x^i \geq d_j \end{cases} \tag{13}$$

where $\lambda_\theta \in (0, 1)$ is a constant.

THEOREM 5. *Suppose the x^i are computed so that $Dx^i < \theta^i$. If the θ are computed by the rules (12) and (13), then (4) and (5) are satisfied.*

Proof: We show the result for each component j. If $D_j x^i \geq d_j$, then $\theta_j^{i+1} = \lambda_\theta D_j x^i + (1 - \lambda_\theta)\theta_j^i$. Condition (4) follows from this since $D_j x^i < \theta_j^i$ and $\lambda_\theta \in (0, 1)$. If $D_j x^i < d_j$, then $\theta_j^{\hat{\imath}} = d_j$ for all $\hat{\imath} > i$ and again condition (4) follows. If, for some finite i, $D_j x^i < d_j$, then condition (5) follows. In the other case, ($D_j x^i \geq d_j \ \forall i$) the limit

$$\theta_j^\infty := \lim_{i \to \infty} \theta_j^i$$

is well defined since the sequence is monotonic and bounded. Then (13) shows that for any $s_j^\infty \in \text{acc}\{D_j x^i\}$ we have

$$\theta_j^\infty = \lambda_\theta s_j^\infty + (1 - \lambda_\theta)\theta_j^\infty.$$

Therefore $s_j^\infty = \theta_j^\infty$, which shows that (5) holds. ∎

2.2 ε-Optimal Solutions

In this section we assume that $\theta = d$ and that we have found some point in $\mathcal{B} \cap \operatorname{int} \mathcal{C}$. We develop a method that converges to a feasible point whose objective value is within a pre-specified optimality tolerance ε. As before, we let $\mathcal{X}(\tau, d)$ denote the set of minimizers of $\mathcal{P}(\tau, d)$.

The following result (see Fiacco and McCormick [3] or page 341 of McCormick [9] for related results) gives us a useful bound on the error of $x \in \mathcal{X}(\tau, d)$ as compared with an optimal solution of (1).

THEOREM 6. *Suppose $\tau > 0$, $x \in \mathcal{X}(\tau, d)$, and x^* is an optimal solution of the original problem (1). Then*

$$c(x) - \varepsilon \leq c(x^*) \leq c(x),$$

for $\varepsilon = \tau J$, where J is the number of rows of D.

Proof: The second inequality follows from the feasibility of x and the optimality of x^*. The KKT conditions show that solving the barrier problem $\mathcal{P}(\tau, d)$ exactly gives

$$r = \nabla_x f(x, \tau, d) + pA,$$

and complementary slackness

$$\max\{r_{k,n}(x_{k,n} - \ell_{k,n}), r_{k,n}(x_{k,n} - u_{k,n})\} = 0 \ \forall k, n.$$

(The vector r is so named because it is the "reduced cost" vector.) Define $q^\top \in \mathbb{R}^J$ as in (6). We note that (x, p, q, r) is feasible for the Wolfe dual of the original problem (1). The Wolfe dual may be stated

$$\begin{aligned} \underset{x,p,q,r}{\text{maximize}} \quad & c(x) + p(Ax - b) + q(Dx - d) + r_+(x - \ell) + r_-(u - x) \\ \text{subject to} \quad & \\ & \nabla c(x) + pA + qD = r \\ & q \geq \mathbf{0} \\ & \max\{r_{k,n}(x_{k,n} - \ell_{k,n}), r_{k,n}(x_{k,n} - u_{k,n})\} = 0 \ \forall k, n. \end{aligned}$$

It is well known that the objective value of the Wolfe dual is a lower bound on $c(x^*)$. The feasibility of x and the complementarity of x and r show that the objective function of the Wolfe dual is equal to

$$c(x) + q(Dx - d) = c(x) + \sum_{j=1}^{J} \frac{\tau(D_j x - d_j)}{d_j - D_j x} = c(x) - \tau J$$

which completes the proof. ∎

Instead of letting $\tau^i \downarrow 0$ we use a sequence $\{\tau^i\}$ generated by the recurrence

$$\tau^{i+1} \leftarrow \max\{\lambda_\tau \tau^i, \tau_{\text{inf}}\}$$

where $\tau^0, \tau_{\text{inf}} > 0$ and $\lambda_\tau \in [0, 1)$. Thus, doing an infinite number of steps gives a sequence with $\text{acc}\{x^i\} \subset \mathcal{X}(\tau_{\text{inf}}, d)$. Using the theorem, we may choose τ_{inf} *a priori* so that $c(x)$ differs from $c(x^*)$ by not more that a pre-specified tolerance for $x \in \mathcal{X}(\tau_{\text{inf}}, d)$. Since $\tau^i = \tau_{\text{inf}}$ for i sufficiently large, we give a method that is able to generate a sequence of points whose accumulation points are in $\mathcal{X}(\tau_{\text{inf}}, d)$. Thus, for large τ, we need not worry about staying close to the solution set $\mathcal{X}(\tau, d)$, which is a generalization of the central path in interior point methods (in fact it reduces to the central path if D is square and invertible). Instead, we merely choose a parameter λ_τ that gives good empirical performance. Note that the sequence $\tau^i = \tau_{\text{inf}}$ $\forall i$ will suffice in theory. In practice we begin with $\tau^1 \gg \tau_{\text{inf}}$ because $\mathcal{P}(\tau, d)$ is ill conditioned for $\tau \approx 0$. This path following idea may be viewed as a heuristic method for reducing the problems caused by ill-conditioning.

2.3 The Three Phase Method

The method we have been developing fits naturally into a three phase framework. We assume that we are given the objective function c and the constraint sets $\mathcal{B}$ and $\mathcal{C}$. Also we are given constants Θ, λ_θ, λ_τ, τ_{inf}, and τ^1 as introduced above.

The algorithm we use is described in figure 1. Note that both the FEASIBILITY PHASE and the REFINE PHASE contain the procedure

$$\text{Generate } x^i \text{ as an } \textit{approximate} \text{ solution of } \mathcal{P}(\tau^i, \theta^i).$$

We shall now discuss how this is implemented.

Note that the sequences $\{\theta^i\}$ and $\{\tau^i\}$ generated by the REFINE PHASE of this method have the properties that $\theta^i = d$ and $\tau^i = \tau_{\text{inf}}$ for all i sufficiently large. Thus, for the REFINE PHASE, we use only *one step* of some iterative method for convex programming to generate x^{i+1} from x^i. The resulting sequence $\{x^i\}$ will then have $\text{acc}\{x^i\} \subset \mathcal{X}(\tau_{\text{inf}}, d)$. For the FEASIBILITY PHASE, one step *may* suffice, but the hypotheses of theorem 4 (finite feasibility) would not necessarily hold.

RELAXED PHASE

$i \leftarrow 0$

Compute x^0 as the solution of the "relaxed" problem (3)

If we determine that $\mathcal{B} = \emptyset$

Then terminate with the message

''the block constraints are infeasible''

Set θ^1 as in (12)

If $x^0 \in \operatorname{int} \mathcal{C}$ go to the REFINE PHASE

Otherwise go to the FEASIBILITY PHASE

FEASIBILITY PHASE

$i \leftarrow i + 1$

Generate x^i as an *approximate* solution of $\mathcal{P}(\tau^i, \theta^i)$

Set θ^{i+1} as in (13)

$\tau^{i+1} \leftarrow \tau^i$

If $x^i \in \mathcal{B} \cap \operatorname{int} \mathcal{C}$

Then go to the REFINE PHASE

Otherwise repeat the FEASIBILITY PHASE

REFINE PHASE

$i \leftarrow i + 1$

Generate x^i as an *approximate* solution of $\mathcal{P}(\tau^i, \theta^i)$

Set $\theta^{i+1} \leftarrow d$

$\tau^{i+1} \leftarrow \max\{\lambda_\tau \tau^i, \tau_{\inf}\}$

Repeat the REFINE PHASE

Figure 1: The three phase method.

3 Solving Barrier Problems

We assume in this section that $\tau > 0$ and θ are given, and that at the current iteration i we are given $x^i \in \mathcal{B}$ such that $f(x^i, \tau, \theta)$ is finite. Our method consists of computing search directions separately for each block and then coordinating them. The method we use is a generalization of the Frank-Wolfe method that uses a trust region and takes advantage of the block structure of the constraints by using a multi-dimensional search rather than a line search.

3.1 Search Directions

Suppose we are given a current point $x^i \in \mathcal{B}$. Linearizing $\mathcal{P}(\tau, \theta)$ and adding a trust region $R(x^i)$ gives the following problem:

$$\underset{y}{\text{minimize}} \nabla_x f(x^i, \tau, \theta) y \text{ subject to } y \in \mathcal{B} \cap R(x^i) \tag{14}$$

which is of the form (2) with $\tilde{c} = \nabla_x f(x^i, \tau, \theta)$ and $R(x^i) = \{y | \tilde{\ell} \leq y \leq \tilde{u}\}$. Let y^i denote the solution of (14) and define $\delta y^i := y^i - x^i$. Recall our assumption that (14) is easy to solve and may be solved in parallel by decomposing into K independent subproblems.

The purpose of the trust region is to take into account the poles of the barrier function. We use the following choice for the trust region $R(\cdot)$. Recall that the objective function is

$$f(x^i + \delta y, \tau, \theta) = c(x^i + \delta y) - \tau \sum_j \ln \left(\theta_j - D_j(x^i + \delta y)\right)$$

from which we immediately see that

$$x^i + \delta y \in \operatorname{dom} f \iff (\forall j) \quad D_j \delta y < \theta_j - D_j x^i.$$

Suppose we were to let only one component (the (k, n) component, say) of δy be nonzero. Then $x^i + \delta y \in \operatorname{dom} f$ only if $D_{j,k,n} \delta y_{k,n} < \theta_j - D_j x^i$. Therefore, we use the "one-sided" trust region R defined by

$$R(x^i) := \left\{x = x^i + \delta y | D_{j,k,n} \delta y_{k,n} \leq \max\left\{\hat{\delta}, \gamma(\theta_j - D_j x^i)\right\} \quad \forall j, k, n\right\}$$

where $\hat{\delta} > 0$ and $\gamma \in (0, 1]$ are constants.

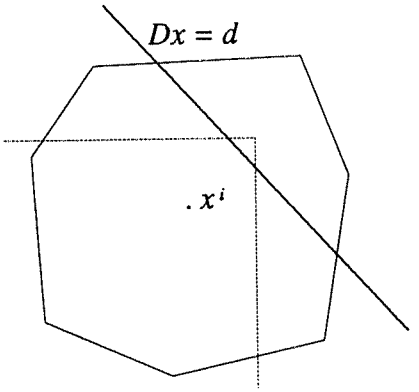

Figure 2: A typical trust region ($\mathcal{B}$ is represented by the solid polygon).

More generally, the trust region property that we require is the following:

ASSUMPTION 7. *Let $z \in \mathcal{B}$ and the sequence $\{x^i\} \subset \mathcal{B}$, and define*

$$\alpha^i := \max_{0 \le \alpha \le 1} \{\alpha | x^i + \alpha(z - x^i) \in R(x^i)\}.$$

Then $\liminf_{i \to \infty} \alpha^i > 0$.

If this assumption is satisfied, there is always some room to move in any direction from x^i without being constrained by the trust region. Assumption 7 is clearly true for our choice of R above because $\hat{\delta} > 0$ and $\mathcal{B}$ is bounded.

3.2 Coordination

The Frank-Wolfe method would use a one-dimensional search along the ray δy^i from x^i to y^i. If we search over a larger region we may expect to do better. The block structure of the original problem makes a multi-dimensional search quite natural. More precisely, we will show that the block structure of A means that we may solve a K dimensional optimization problem with simple bounds in order to "coordinate" the subproblem solutions. In this section $f(\cdot)$ and $\nabla f(\cdot)$ are used in place of $f(\cdot, \tau, \theta)$ and $\nabla_x f(\cdot, \tau, \theta)$ since τ and θ are fixed.

Let Y^i denote the $N \times K$ matrix of block search directions:

$$Y^i := \begin{pmatrix} \delta y_1^i & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \delta y_2^i & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \delta y_K^i \end{pmatrix}$$

so that the kth column of Y^i corresponds to the kth block of the update δy^i. The coordination problem that we consider generates w^i as an approximate solution of

$$\underset{w}{\text{minimize}} f(x^i + Y^i w) \text{ subject to } \ell \leq x^i + Y^i w \leq u. \tag{15}$$

We will then choose $x^{i+1} = x^i + Y^i w^i$ for our next iterate. Note that $x^{i+1} \in \mathcal{B}$ if $x^i \in \mathcal{B}$ because Y^i is in the null-space of the equations defining $\mathcal{B}$. Also, it is easy to enforce $x^{i+1} \in \operatorname{dom} f$ since $x^i \in \operatorname{dom} f$. Our approximate solutions of (15) will satisfy

$$f(x^{i+1}) - f(x^i) \leq \mu \left[\min_{x \in \text{ray}[x^i, y^i]} f(x) - f(x^i) \right] \tag{16}$$

where $\mu \in (0,1)$ and where the minimization is taken over the line segment from x^i to y^i. (We consider below computationally tractable methods for guaranteeing satisfaction of (16) in a finite number of iterations.)

We now show that this method converges to solutions of $\mathcal{P}(\tau, \theta)$ where τ and θ are fixed. From (16) we may deduce that the algorithm is monotonic, i.e. $f(x^{i+1}) \leq f(x^i)$. Consider a subsequence $\{x^{\sigma(i)}\} \subset \{x^i\}$ that has $x^{\sigma(i)} \to \bar{x}$ and $y^{\sigma(i)} \to \bar{y}$. This subsequence must exist because $\mathcal{B}$ is compact. Also define $\overline{\delta y} := \bar{y} - \bar{x}$ so that $\delta y^{\sigma(i)} \to \overline{\delta y}$. Pick any $\lambda \in (0,1]$. Then

$$\begin{aligned} f(x^{\sigma(i)} + \lambda \delta y^{\sigma(i)}) - f(x^{\sigma(i)}) &\geq \left[\min_{x \in \text{ray}[x^{\sigma(i)}, y^{\sigma(i)}]} f(x) - f(x^{\sigma(i)}) \right] \\ &\geq \frac{1}{\mu} \left[f(x^{\sigma(i)+1}) - f(x^{\sigma(i)}) \right]. \end{aligned}$$

Since $f(\cdot)$ is bounded from below on $\mathcal{B}$, taking the limit as $i \to \infty$ gives $f(\bar{x} + \lambda \overline{\delta y}) - f(\bar{x}) \geq 0$, and taking the limit of this as $\lambda \downarrow 0$ gives

$$0 \leq \nabla f(\bar{x}) \overline{\delta y} = \nabla f(\bar{x}) (\bar{y} - \bar{x}). \tag{17}$$

Suppose now that $z \in \mathcal{B}$ is an arbitrary point. Since R satisfies assumption 7,

$$\alpha^i = \max_{0 \le \alpha \le 1} \left\{ \alpha | x^i + \alpha(z - x^i) \in R(x^i) \right\}$$

has $\liminf \alpha^i > 0$. Then

$$\nabla f(x^i)\alpha^i(y^i - x^i) \le \nabla f(x^i)\alpha^i(z - x^i)$$

by the definition of y^i. By taking the limit, and using (17) and the convexity of f

$$0 \le \nabla f(\bar{x})(\bar{y} - \bar{x}) \le \nabla f(\bar{x})(z - \bar{x}) \le f(z) - f(\bar{x}).$$

This shows that $\bar{x}$ is a global minimizer of f over $\mathcal{B}$ and hence $\bar{x} \in \mathcal{X}(\tau, \theta)$.

Except in special cases, the minimum in condition (16) may not be computed in a finite number of steps. We therefore substitute a computable lower bound.

Let $\phi(w) = f(x^i + Y^i w)$, so that the coordinator problem (15) may be stated

$$\underset{w}{\text{minimize}}\, \phi(w) \text{ subject to } \underline{w} \le w \le \overline{w} \tag{18}$$

for bounds $\underline{w}$ and $\overline{w}$ that correspond to the bounds of (15). (Note that $\underline{w} \le \mathbf{0}$ and $\overline{w} \ge \mathbf{1}$.) Since ϕ is convex,

$$\phi(\hat{w}) + \nabla\phi(\hat{w})(w - \hat{w}) \le \phi(w)$$

for all w. Restricting the linearization on the left to $w \in \text{ray}[\mathbf{0}, \mathbf{1}]$ (which corresponds to $x \in \text{ray}[x^i, y^i]$ in the original space) we will obtain a lower bound on the minimum in (16). Clearly this linearization restricted to a line segment will achieve its minimum at an endpoint. Therefore, defining

$$\psi(\hat{w}) := \phi(\hat{w}) + \min\left\{\nabla\phi(\hat{w})(\mathbf{0} - \hat{w}), \nabla\phi(\hat{w})(\mathbf{1} - \hat{w})\right\},$$

we have shown

$$\psi(\hat{w}) \le \min_{w \in \text{ray}[\mathbf{0},\mathbf{1}]} \phi(w) = \min_{x \in \text{ray}[x^i, y^i]} f(x).$$

Thus, instead of satisfying (16) directly, we instead require

$$\phi(w^i) - \phi(\mathbf{0}) \le \mu \left[\psi(w^i) - \phi(\mathbf{0})\right], \tag{19}$$

in order to accept w^i. A w^i satisfying (19) may, therefore, be computed in a finite number of iterations for $\mu < 1$. Then setting $x^{i+1} \leftarrow x^i + Y^i w^i$ implies that (16) is satisfied.

In practice, we have found that it is better to choose w^i to satisfy both (19) and

$$\left\| \text{reduced gradient of } \phi(w^i) \right\| \leq \mu' \left\| \text{reduced gradient of } \phi(\mathbf{0}) \right\|$$

(in some norm $\|\cdot\|$) for some $\mu' \in (0,1)$. That is, we accept w^i as a solution to the coordinator problem if it has sufficient decrease to ensure convergence (i.e. (19) is satisfied) *and* the reduced gradient is sufficiently reduced in norm.

The coordinator problem may be enriched by utilizing one or more sets of "prior" updates as well as the "current" updates Y^i. Since all updates lie in the null space of the equality constraints of $\mathcal{B}$, we need only take into account the bounds $\ell \leq x \leq u$. We shall discuss the case where our prior updates are merely the set of updates Y^{i-1} from the previous iteration. Let v be the vector of weights for the prior update Y^{i-1}, analogous to the role of w for the current update Y^i. The coordination constraints are then

$$\ell \leq x^i + Y^{i-1} v + Y^i w \leq u, \tag{20}$$

which no longer reduce to simple bounds on v and w. However, we may easily compute sets of separate bounds on v and w that form an "inner approximation" of (20) in the sense that any (v, w) feasible for the bounds must also be feasible for (20). To do this, choose a constant $\xi \in (0,1)$ and constrain w by $\xi \underline{w} \leq w \leq \xi \overline{w}$, where $\underline{w}$ and $\overline{w}$ are obtained exactly as above. Having set these constraints, it is easy to compute $\underline{v}$ and $\overline{v}$ such that

$$\xi \underline{w} \leq w \leq \xi \overline{w} \text{ and } \underline{v} \leq v \leq \overline{v} \qquad \Rightarrow \qquad (v, w) \text{ satisfies (20).}$$

The convergence proof is extended in a straightforward manner to cover generalized coordination procedures of this type. This approach is related to the PARTAN method of combining update information (see Himmelblau [6] and Lee *et al* [8]), and also to restricted simplicial decomposition (see Lawphongpanich and Hearn [7]).

4 Numerical Results for a Large-Scale Application

This section describes our test problems and documents the success we have had using our decomposition algorithm. Section 4.1 describes our code and

the key parameters used. In section 4.2 we describe the problems that motivated this decomposition algorithm. Finally, section 4.3 shows the timing results on the large-scale problems.

4.1 Description of the Code and Parameters

We ran our code on two machines: a DECstation 3100 running the ULTRIX operating system, and a 20 processor Sequent Symmetry S81 running the DYNIX operating system. Most of the code was written in C, with the portion used to solve the network subproblems being written in FORTRAN (see further discussion below). The C programming language was chosen primarily because of its ability to work properly with modern data structures. The code was compiled with the default code optimization (`-O1`). Double precision was used for all calculations.

The following parameter values were used in our runs:

- We ran the method for 50 iterations in all cases. The answers we obtained matched known good approximations of optimal values to between 5 and 7 digits. Moreover, improvement in $c^T x^i$ tends to become small after 50 iterations in our implementation.

- $\lambda_\theta = 0.9$
  We found that if this parameter is smaller, the method does not find feasible solutions as quickly. Larger values of λ_θ don't seem to cause numerical problems for our test problems.

- $\lambda_\tau = 0.5$
  Smaller values tend to introduce the problems of ill-conditioning earlier, so that convergence is hampered. Larger values require too many iterations until τ^i is sufficiently small.

- $\tau^0 = 10$
  We normalize the cost coefficients so that $\|c\|_\infty = 1$, making τ^0 ten times the maximum absolute value of the cost coefficients.

- $\varepsilon = \tau_{\text{inf}} J = 10^{-8}$
  (See theorem 6.) Eight places of accuracy in the objective function is a fairly ambitious goal for problems as large as our test problems. This was not always achieved in 50 iterations.

The code takes full advantage of the network structure of the A_k, since solving multicommodity networks was the initial goal of this work. The code also takes advantage of the special structure of the matrix D. Typically for multicommodity networks, a constraint $D_j x \leq d_j$ represents a physical situation where the flow of a given "topological arc" (an arc appearing at most once in each commodity) can only handle a certain capacity of flow. In this case $D_{j,k,n} \in \{0, 1\}$, and for each j and k, there is at most one $\hat{n}$ such that $D_{j,k,\hat{n}} = 1$. Our code has a $J \times K$ array of pointers to this $\hat{n}$ (it stores 0 if no such $\hat{n}$ exists). This saves space because J and K are usually much smaller than N.

To solve the network subproblems we use a modified version of RNET, written in FORTRAN by Grigoriadis and Hsu [5]. RNET is an implementation of the network simplex method. This code was modified by the authors to work in double precision rather than integer arithmetic, and to use parameters to specify input data. We begin RNET with an all artificial basis at each iteration. The parameters given to RNET are mostly determined by the suggestions in Grigoriadis and Hsu [5], except for the following:

`BT` (number of bits in one integer word)
This was removed because we deal with floating point data.

`MXIT` == $15M_k$ (maximum number of iterations)
Some of the larger problems do a huge number of pivots toward the end of the computation on some commodities. This indicates that some of these networks are very difficult to solve.

For the trust region in subsection 3.1 we set $\gamma = 0.7$ and $\hat{\delta} = 10^{-8}$. We found that $\gamma \in [0.5, 0.9]$ worked reasonably well. We also found, somewhat to our surprise, that using smaller $\hat{\delta}$ seemed to make the algorithm perform better. Using a smaller $\hat{\delta}$ means that we let the current point get closer to the boundary of the trust region, possibly at the expense of being able to move less within the null space of D. We found that $\hat{\delta} = 0$ (in which case our convergence proof fails) worked just fine in practice.

The coordinator algorithm uses an active set method in conjunction with the steepest descent direction. We stop when both the function and the norm of the projected gradient have been sufficiently decreased, as is explained at the end of subsection 3.2. The code uses $\mu = 0.4$ and $\mu' = 0.03$. If we stop when the function values have been sufficiently decreased, but ignore the projected gradient, then the method converges in theory, but in practice it

seems somewhat problematic. Using a larger μ' would allow the algorithm to terminate, when in fact the coordinator could probably find a significantly better point at low cost. We run the coordinator algorithm for at most 15 iterations within each major iteration.

As part of this coordinator method we need to use a one-dimensional line search method. The one-dimensional line search algorithm we used is (2.6.4) in Fletcher [4]. Special structure of the objective function allows us to use Newton's method in place of the usual minimization of a quadratic or cubic interpolant. Parameters were set to attain a line search of medium accuracy.

4.2 Patient Distribution System Problems

The test problems we used were obtained from the CINCMAC analysis group of the Military Airlift Command (MAC) at Scott Air Force Base (Chmielewski [2]). The model is called the Patient Distribution System (PDS) and is a logistics model designed to help make decisions about how well MAC can evacuate patients from Europe. The PDS problems are a class of problems; PDS-$\mathcal{D}$ denoting the problem that models a scenario lasting $\mathcal{D}$ days. PDS-$\mathcal{D}$ well-defined for integers $\mathcal{D} \in [1, 85]$. As $\mathcal{D}$ becomes larger, the size of PDS-$\mathcal{D}$ grows quite large, as may be seen in table 1. The PDS problems are linear multicommodity network flow problems, which are block angular linear programs where each A_k is a node-arc-incidence matrix. (See figure 3.)

These PDS problems have received considerable attention lately, partly because they are a real-world application, and partly because they seem to be quite challenging. For example, in Carolan *et al* [1] the KORBX system was used at Scott AFB to solve numerous challenging problems, including some of the smaller PDS problems. It took the KORBX system (using default parameters) between 3.3 hours and 4.5 hours to solve PDS-10. The KORBX system had a very difficult time, however, solving PDS-20. In fact, only one out of the four KORBX codes finished within 24 hours on PDS-20.

4.3 Results of Numerical Experiments

We shall now present performance results of our codes on a subset of the PDS problems. We were interested in two things when beginning these tests. First we wanted to develop algorithms that compute approximate solutions

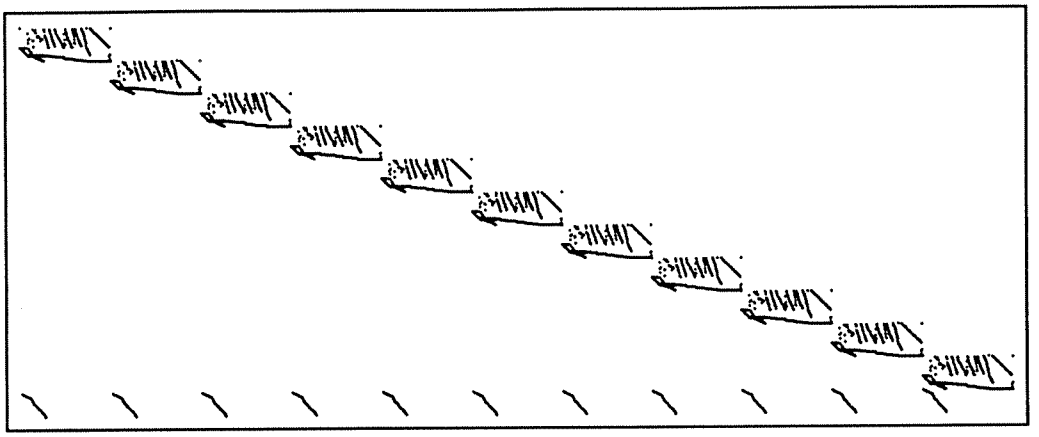

Figure 3: Sparsity structure of the constraint matrix for PDS-01.

Problem Name	max size of block $\max_k M(k)$	$\max_k N(k)$	coupling J	dimension of A M	N
PDS-01	126	339	87	1,386	3,729
PDS-02	252	685	181	2,772	7,535
PDS-03	390	1117	303	4,290	12,287
PDS-05	686	2,149	553	7,546	23,639
PDS-06	835	2,605	696	9,185	28,655
PDS-10	1,399	4,433	1,169	15,389	48,763
PDS-20	2,857	10,116	2,447	31,427	105,728
PDS-30	4,223	15,126	3,491	46,453	154,998
PDS-40	5,652	20,698	4,672	62,172	212,859
PDS-50	7,031	26,034	5,719	77,341	270,095
PDS-60	8,423	31,474	6,778	92,653	329,643
PDS-70	9,750	36,262	7,694	107,250	382,311

Table 1: Sizes of some of the PDS problems. We also remind the reader that each PDS problem has eleven blocks (i.e. $K = 11$).

to multicommodity network flow problems quickly. Second we want our method to compute accurate solutions. Our method does not verify that the solution is accurate. We compute duals on the network constraints and the dual estimates q (defined in (6)), and these will give lower bounds much the same as in the proof of theorem 6. When we have checked these lower bounds, however, they are not even as good as the lower bound given by solving the relaxed problem (3).

The tables that follow contain timings and optimal objective function values for the PDS problems we solved. The column of the table labeled "relaxed" contains statistics for computing the solution to (3). The column labeled "feasible" contains statistics for computing a feasible point. The column labeled "final" contains statistics for computing the final approximation of the optimal solution. The row of the table labeled "iter" is the number of iterations the method has taken to attain a given phase. The row labeled "obj $\times 10^{-10}$" is the value of $10^{-10} c^\top x^i$ where x^i is the current point and c is the *original* cost vector ($\|c\|_\infty$ is *not* necessarily 1). A row labeled "DECstation" shows the performance on the DECstation 3100 . A row labeled "Sequent($\wp$)" shows the performance on the Sequent Symmetry using $\wp$ processors. All times reported are wall clock time.

PDS-01			
phase	relaxed	feasible	final
total iterations	0	11	50
objective $\times 10^{-10}$	2.9033	2.9096	2.9084
DECstation	1.4sec	18sec	1min 30sec
Sequent(11)	1.1sec	12sec	1min 4sec
PDS-02			
phase	relaxed	feasible	final
total iterations	0	12	50
objective $\times 10^{-10}$	2.8758	2.8876	2.8858
DECstation	2.9sec	42sec	3min 17sec
Sequent(11)	1.5sec	22sec	2min 9sec
PDS-03			
phase	relaxed	feasible	final
total iterations	0	13	50
objective $\times 10^{-10}$	2.8442	2.8622	2.8597
DECstation	4.7sec	1min 20sec	5min 47sec
Sequent(11)	2sec	39sec	3min 38sec
PDS-05			
phase	relaxed	feasible	final
total iterations	0	16	50
objective $\times 10^{-10}$	2.7824	2.8125	2.8054
DECstation	15sec	3min 14sec	11min 28sec
Sequent(11)	4.2sec	1min 33sec	6min 43sec
PDS-06			
phase	relaxed	feasible	final
total iterations	0	16	50
objective $\times 10^{-10}$	2.7526	2.7846	2.7761
DECstation	17sec	4min 17sec	15min 4sec
Sequent(11)	4.6sec	1min 56sec	8min 44sec

Table 2: Timing and objective value results for small PDS problems.

PDS-10			
phase	relaxed	feasible	final
total iterations	0	16	50
objective $\times 10^{-10}$	2.6333	2.6857	2.6727
DECstation	30sec	8min 15sec	28min 31sec
Sequent(11)	9sec	3min 57sec	16min 39sec
PDS-20			
phase	relaxed	feasible	final
total iterations	0	14	50
objective $\times 10^{-10}$	2.3342	2.4069	2.3822
DECstation	2min 22sec	33min 19sec	2hr 12min
Sequent(11)	40sec	9min 44sec	50min 43sec
PDS-30			
phase	relaxed	feasible	final
total iterations	0	12	50
objective $\times 10^{-10}$	2.0284	2.1818	2.1390
DECstation	4min 38sec	1hr 9min	5hr 23min
Sequent(11)	1min 40sec	16min 43sec	1hr 48min
PDS-40			
phase	relaxed	feasible	final
total iterations	0	14	50
objective $\times 10^{-10}$	1.7188	1.9452	1.8866
DECstation	8min 53sec	2hr 39min	10hr 27min
Sequent(11)	3min 45sec	32min 32sec	2hr 54min

Table 3: Timing and objective value results for large PDS problems.

PDS-50			
phase	relaxed	feasible	final
total iterations	0	13	50
objective $\times 10^{-10}$	1.5002	1.7336	1.6625
DECstation	13min 28sec	3hr 50min	16hr 46min
Sequent(11)	4min 35sec	54min 2sec	5hr 30min
PDS-60			
phase	relaxed	feasible	final
total iterations	0	13	50
objective $\times 10^{-10}$	1.2159	1.5288	1.4462
DECstation	18min 40sec	5hr 27min	24hr 6min
Sequent(11)	5min 38sec	1hr 19min	6hr 55min
PDS-70			
phase	relaxed	feasible	final
total iterations	0	16	50
objective $\times 10^{-10}$	0.9309	1.3191	1.2311
Sequent(11)	8min 12sec	2hr 9min	9hr 24min

Table 4: Timing and objective value results for very large PDS problems.

5 Summary and Conclusions

We have developed a structured interior point method for block angular problems. This three-phase approach takes advantage of the constraint structure by keeping the block-structured constraints explicitly and using barrier functions to model the coupling constraints. This allows us to generate a starting point for a shifted barrier problem and to obtain search directions quickly and in parallel. We also discussed techniques for coordinating these search directions to assure convergence to an ε-optimal point. This algorithm is particularly well-suited to multicommodity flow problems, since the subproblems are then linear single-commodity networks. Computational experience with a class of real-world multicommodity problems arising from an Air Force MAC application indicates that the method is very efficient, even for problems with hundreds of thousands of variables.

References

[1] W.J. Carolan, J.E. Hill, J.L. Kennington, S. Niemi, and S.J. Wichmann. An empirical evaluation of the KORBX algorithms for military airlift applications. *Operations Research*, 38(2):240–248, March-April 1990.

[2] Major R. Chmielewski. Private Communication, March 1989.

[3] A.V. Fiacco and G.P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques.* John Wiley and Sons, 1968.

[4] R. Fletcher. *Practical Methods of Optimization.* John Wiley and Sons, second edition, 1987.

[5] M.D. Grigoriadis and Tau Hsu. *RNET, The Rutgers Minimum Cost Network Flow Subroutines, Users Documentation.* Department of Computer Science, Hill Center for the Mathematical Sciences, Rutgers University—Busch Campus, New Brunswick, New Jersey 08903, 3.6 edition, October 1979.

[6] D.M. Himmelblau. *Applied Nonlinear Programming.* McGraw-Hill, New York, 1972.

[7] S. Lawphongpanich and D. W. Hearn. Restricted simplicial decomposition with application to the traffic assignment problem. Technical Report 83-8, Industrial and Systems Engineering Department, University of Florida, Gainesville, Fla. 32611, Sep. 1983.

[8] D.N. Lee, K.T. Medhi, J.L. Strand, R.G. Cox, and S. Chen. Solving large telecommunications network loading problems. *AT&T Technical Journal*, 68(3):48–56, May-June 1989.

[9] G.P. McCormick. *Nonlinear Programming, Theory, Algorithms, and Applications.* John Wiley and Sons, 1983.

[10] R.T. Rockafellar. *Convex Analysis.* Princeton University Press, 1970.