Interior point methods for large-scale linear programming[1]

John E. Mitchell, Kris Farwell, and Daryn Ramsden
Mathematical Sciences
Rensselaer Polytechnic Institute
Troy, NY 12180
{mitchj,farwek,ramsdd}@rpi.edu

August 16, 2004

Abstract

We discuss interior point methods for large-scale linear programming, with an emphasis on methods that are useful for problems arising in telecommunications. We give the basic framework of a primal-dual interior point method, and consider the numerical issues involved in calculating the search direction in each iteration, including the use of factorization methods and/or preconditioned conjugate gradient methods. We also look at interior point column generation methods which can be used for very large scale linear programs or for problems where the data is generated only as needed.

Keywords: Interior point methods, preconditioned conjugate gradient methods, network flows, column generation.

[1]Research supported in part by NSF grant numbers DMS-0317323 and CMS-0301661. The research of the second author was supported by the NSF through the VIGRE program, grant number DMS-9983646.

1 Introduction

The performance of solvers for linear programming problems has improved dramatically in recent years. A user of linear programming packages now has available the option of using sophisticated interior point methods developed in the last twenty years. In addition, the development of interior point methods has spurred considerable successful research into efficient implementations of the simplex algorithm. It appears that different methods are better for different problems, with interior point methods possibly becoming a better choice as problem size grows. In addition to size, the structure of the linear programming problem is a major determinant as to which algorithm should be chosen. The simplex algorithm is discussed elsewhere in this book, so we focus on interior point methods.

The use of an interior point method is a good choice for general large-scale linear programming problems. We introduce interior point algorithms in §2, and discuss the computational issues that arise in §3.

For problems with well-defined structure, it is often true that an algorithm can be developed which will successfully exploit the structure. For example, the network simplex algorithm exploits the nature of basic feasible solutions to network flow problems. Interior point methods can also be refined to solve network flow problems efficiently, and we survey preconditioned conjugate gradient approaches to these problems in §4. The structure of multicommodity network flow problems can also be exploited in a carefully designed interior point method, as discussed in §5.

Linear programming problems often arise as subproblems of other problems. For example, integer programming problems can be solved using branch-and-cut, and multicommodity problems can be formulated with a huge number of variables and then attacked using a column generation approach. We survey the use of interior point methods in column generation and constraint generation settings in §6.

Conclusions are offered in §7.

2 Primal-dual interior point methods

Several excellent textbooks on interior point methods were published in the 1990's, all of which discuss the material in this section in far greater detail. In particular, the reader is referred to Roos et al. [47], Vanderbei [50], Wright [51], and Ye [54]. Several papers also surveyed primal-dual interior point methods, and our presentation is closest to that in Andersen et al. [2].

We take the following to be our standard primal-dual LP pair:

$$\begin{array}{lrcl} \min & c^T x & & \\ \text{subject to} & Ax & = & b \\ & x+s & = & u \\ & x, s & \geq & 0 \end{array} \qquad (P)$$

and

$$\begin{array}{lcccccccc} \max & b^T y & & & - & u^T w & & \\ \text{subject to} & A^T y & + & z & - & w & = & c \\ & & & & & z, w & \geq & 0 \end{array} \qquad (D)$$

Here, A is an $m \times n$ matrix, c, x, z, and w are n-vectors, and b and y are m-vectors. We assume that the rank of A is m, that the feasible region of (P) is bounded, and that (P) and (D) each have a bounded set of optimal solutions.

An interior point method is an iterative scheme for solving (P) and (D) with each iterate strictly satisfying the nonnegativity restrictions. The primal-dual interior point barrier method can be motivated by setting up the subproblem

$$\begin{array}{lrcl} \min & c^T x & - & \mu(\sum_{i=1}^n \ln(x_i) + \sum_{i=1}^n \ln(s_i)) \\ \text{subject to} & Ax & = & b \\ & x+s & = & u \end{array} \qquad (P(\mu))$$

where μ is a positive scalar. As μ is varied, the solution to $(P(\mu))$ traces out the *central path*. The limiting solution to $(P(\mu))$ as $\mu \to \infty$ is the *analytic center* of the feasible region of (P). The algorithm finds an approximate solution to $(P(\mu))$ for a particular μ, reduces μ, and repeats until a sufficiently accurate solution to (P) is found. As $\mu \to 0$, the solution to $(P(\mu))$ converges to the analytic center of the set of optimal solutions to (P).

The Karush-Kuhn-Tucker optimality conditions for $(P(\mu))$ can be written

$$Ax = b \qquad (1)$$

$$x + s = u \qquad (2)$$

$$A^T y + z - w = c \qquad (3)$$

$$Zx = \mu e \qquad (4)$$

$$Ws = \mu e \qquad (5)$$

$$x, s, z, w \geq 0 \qquad (6)$$

where Z and W denote diagonal matrices containing the entries of z and w respectively on the diagonal, and e denotes the vector of all ones of the appropriate dimension. The matrices S and X are defined similarly to Z and W.

Given an iterate $(\bar{x}, \bar{s}, \bar{y}, \bar{z}, \bar{w})$ strictly satisfying the nonnegativity constraint (6), one approach to Newton's method for finding a solution to the barrier problem is to solve the following system to find a direction $(\Delta x, \Delta s, \Delta y, \Delta z, \Delta w)$:

$$\begin{bmatrix} A & 0 & 0 & 0 & 0 \\ 0 & 0 & A^T & I & -I \\ I & I & 0 & 0 & 0 \\ Z & 0 & 0 & X & 0 \\ 0 & W & 0 & 0 & S \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta s \\ \Delta y \\ \Delta z \\ \Delta w \end{bmatrix} = \begin{bmatrix} r_b \\ r_c \\ r_u \\ \tau\mu e - XZe \\ \tau\mu e - SWe \end{bmatrix} \tag{7}$$

where τ is a scalar between 0 and 1, the diagonal matrices X, Z, S, and W are defined at the current iterate, and

$$r_b = b - A\bar{x} \tag{8}$$

$$r_c = c - A^T\bar{y} - \bar{z} + \bar{w} \tag{9}$$

$$r_u = u - \bar{x} - \bar{s} \tag{10}$$

Choosing different values of τ gives different directions. Taking $\tau = 0$ corresponds to a pure descent direction with no centering component, and this is known as the affine direction or the predictor direction. Choosing $\tau = 1$ and $\mu = \frac{1}{2n}(\bar{x}^T\bar{z} + \bar{s}^T\bar{w})$ is a pure centering direction, and it results in a direction that leaves the duality gap $x^Tz + s^Tw$ unchanged.

The system (7) can be converted into an equivalent system of equations with the making of the following eliminations:

$$\Delta z = X^{-1}(\tau\mu e - XZe - Z\Delta x) \tag{11}$$

$$\Delta s = u - x - s - \Delta x \tag{12}$$

$$\Delta w = S^{-1}(\tau\mu e - SWe - Wr_u + W\Delta x) \tag{13}$$

the system becomes

$$\begin{bmatrix} -D^{-2} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} r_c - X^{-1}(\tau\mu e - XZe) + S^{-1}(\tau\mu e - SWe - Wr_u) \\ r_b \end{bmatrix} \tag{14}$$

where

$$D^2 = (X^{-1}Z + S^{-1}W)^{-1}. \tag{15}$$

Equation (14) is known as the *augmented system* of equations. All interior point methods have to solve a system essentially of this form, possibly with a different diagonal matrix and with a different right hand side vector. The system can be reduced further to give the *normal equations*, through the elimation of Δx. In particular, multiplying the first set of equations by AD^2 and adding the second set gives:

$$(AD^2A^T)\Delta y = r_b + AD^2(r_c - X^{-1}(\tau\mu e - XZe) + S^{-1}(\tau\mu e - SWe - Wr_u)) =: g. \quad (16)$$

Note that we define g to represent the right hand side of the normal equations. The *augmented system* and the *normal equations* lend their names to the two main approaches to solving for the vector of Newton directions. These approaches are discussed in the next section.

Once this vector of directions has been computed, a new feasible point can be found by scaling each of the steps down (to maintain feasibility) before adding them to the current feasible solution. At this point μ is updated and the procedure is repeated. Predictor-corrector algorithms are the current methods of choice. These methods alternate between a predictor step with $\tau = 0$ and a corrector step with $\tau = 1$. The algorithm can be implemented in such a way that the work of calculating both steps is not much greater than the work of calculating either step individually — see the next section.

3 Computational considerations for general linear programs

Computational considerations can intuitively be divided up into three categories, namely 1) those related to initialization, 2) those related to finding and scaling the vector of newton directions and 3) those related to termination. As in the preceding section, we will adhere most closely to Andersen et al. [2].

Initialization of an implementation consists of a presolving stage and finding an initial point. Presolving is the process by which the matrix A is examined to see if there are any straightforward measures that can be taken to make the problem easier to solve. For instance, the process may unearth duplicate rows or columns, in which case the corresponding dual or primal variables respectively can be combined, or it may find, with minimal effort, that certain variables should have fixed values or that some constraints are redundant. Also it is possible that inefficient formulation of the problem may result in zero columns, meaning we can simply push the corresponding variable to its upper or lower bound depending on whether or not its coefficient in the objective function is negative or positive. Measures such as these drastically reduce the workload in the body of the algorithm and overall it can be said that a preprocessing stage is well worth the work required. A comprehensive treatment of presolving has been done by Andersen and Andersen. [1].

Finding an initial point can be done by solving the quadratic program below, which has a solution given by an explicit formula:

$$\begin{array}{lrcl} \min & c^T x & - & \frac{\sigma}{2}(x^T x + s^T s) \\ \text{subject to} & Ax & = & b \\ & x + s & = & u \end{array}$$

The formula may result in values for x and s that may be negative and as such they can be pushed toward positivity. Parallel to this a solution (y, z, w) for the dual can be constructed with $y = 0$.

As mentioned before there are two approaches to solving for the vector of Newton directions, namely the *augmented systems* approach and the *normal equations* approach. The choice of approach is based on the relative density of the matrix AD^2A^T when compared to the matrix A. If the former is relatively dense in comparison to the latter, formulation and usage of the *normal equations* is unnecessarily expensive. We can now look at the solving of the *normal equations*. If we let C be the coefficient matrix in equation (16) after a possible symmetric permutation of its rows and columns, under the assumption that A is of full row rank, we have that C is positive definite and we can use Cholesky factorization to find a lower triangular matrix L such that $C = PAD^2A^TP^T = LL^T$ where P is a permutation matrix. Systems of equations of the form $C\Delta y^* = r$ can then be solved (y^* representing an appropriate renumbering of the components of y to correspond with the permutations made to AD^2A^T), by solving the following systems in turn :

$$\begin{array}{rcl} L\gamma & = & \nu \\ L^T \Delta y^* & = & \gamma \end{array}$$

The stability of the calculation is independent of the choice of permutation and thus much of the task at hand is the choice of an appropriate permutation which makes L reasonably sparse. There are two main ordering heuristics 1) the minimum degree and 2) minimum local fill-in heuristics [51].

Using the minimum degree ordering involves looking at the remaining matrix and choosing as the pivot element, the diagonal element that has the least number of nonzeroes (we call this its degree, d) in its row/column as the new pivot element at each step of the Cholesky algorithm. This is motivated by the fact that the algorithm generates d^2 nonzeroes in the update matrix and as such d should be kept as small as possible.

Minimum local fill-in is the main competing heuristic and is motivated by the fact that the minimum degree figures out how many elements are actually changed in the elimination steps as opposed to figuring out how many go from being zero to nonzero (a subset of the former category). Many times it is possible to choose a pivot element that results in a more sparse factorization at the expense of carrying out more analytical work beforehand. Choosing the right variant of minimum local fill-in can possibly lead to results that are competitive with that using the minimum degree heuristic.

The normal equations approach is particularly susceptible to weak performance in situations where A has dense columns and thus generates dense blocks in the AD^2A^T matrix. One simple possible remedy is to examine both of the factorizations, AA^T and A^TA. In situations that A has both columns and rows which are dense, this trick loses its effectiveness. At this point in the implementation more sophisticated methods can be employed, but it is often more appropriate to resort to working with the *augmented system*.

The augmented system approach utilizes the Bunch-Parlett factorization algorithm to get the following:

$$\begin{bmatrix} -D^{-2} & A^T \\ A & 0 \end{bmatrix} = L\Lambda L^T \tag{17}$$

where Λ is a block diagonal matrix. This formulation is more robust as far as coping with the effects of dense columns of A. Also while the matrix is prone to being ill-conditioned it is relatively easy to make judgements on the quality of the solution vector.

Once a solution vector has been found it is important to scale it before adding it to the current iterate. Conventionally the maximum value of a multiplier of the solution vector that allows for maintained feasibility is scaled down by a predetermined constant between 0 and 1 to obtain the size of the step taken in the direction of the solution vector.

There are many possible predictor-corrector variants. One possibility is to first solve (7) with $\tau = 0$, then modify the right hand side of (7) as if the predictor step had been taken to the boundary of the feasible region, and then calculate the corrector step with $\tau = 1$ and with a guess for the new value of μ. If μ^0 denotes the old value of μ and μ^{aff} denotes the value obtained at the boundary of the feasible region after the affine step, Mehrotra [33] suggests taking $(\frac{\mu^{aff}}{\mu^0})^3\mu^0$ as the guess for the new value of μ in the calculation of the corrector step. The direction taken from the current iterate is the solution to this corrector system of equations. Note that the diagonal scaling matrices in the calculation of the corrector step are the same as those in the predictor step, so it is only necessary to factorize the matrix AD^2A^T once for each pair of a predictor and corrector step. This idea of calculating two vectors Δy with the same factorization of AD^2A^T can be extended naturally to finding multiple corrections to the affine direction, giving *higher order* methods. These higher order methods may reduce the number of iterations required to solve the problem slightly, with each iteration marginally more expensive, and they can lead to considerable overall reductions in computational time.

As far as termination criteria are concerned, we have only to monitor the duality gap and feasibility at each iterate until they are sufficiently small to meet the prescribed required precision. For a required precision of 8 digits, (a very common standard), the conditions are as follows:

$$\frac{||Ax-b||}{1+||b||} \leq 10^{-8} \text{ and } \frac{||x+s-u||}{1+||u||} \leq 10^{-8} \tag{18}$$

$$\frac{||A^Ty+z-w-c||}{1+||c||} \leq 10^{-8} \tag{19}$$

$$\frac{|c^Tx-(b^Ty-u^Tw)|}{1+|b^Ty-u^Tw|} \leq 10^{-8} \tag{20}$$

The last of these conditions is often the most stringent and difficult to satisfy and consequently in most cases it is adequate simply to check only its validity. Some implementations may even feature a termination phase in which an optimal vertex/basis is recovered. This is most notably implemented in CPLEX and is comparable to $O(m)$ simplex iterations in terms of complexity. A strongly polynomial algorithm for the recovery of a primal-dual optimal basis given optimal solutions to both the primal and dual problems was provided by Megiddo [32].

4 Preconditioned conjugate gradient methods for network flow problems

A standard implementation of an interior point method, using a complete factorization of AD^2A^T, is not competitive with the network simplex method for solving network flow problems. An alternative is to use a preconditioned conjugate gradient algorithm to find Δy in (16). The structure of the problem allows the construction of preconditioners that work very well.

In the set of equations (16) for interior point methods

$$AD^2A^T\Delta y = g \tag{21}$$

multiplying by the preconditioner M^{-1} gives

$$M^{-1}AD^2A^T\Delta y = M^{-1}g \tag{22}$$

where M is a symmetric and positive definite matrix. The matrix M should be chosen appropriately so that equation (22) can be solved more easily than (21). It should also make the matrix $M^{-1}AD^2A^T$ well-conditioned, with few extreme eigenvalues. These two requirements for M are somewhat in conflict. For example, taking $M = AD^2A^T$ makes $M^{-1}AD^2A^T = I$, so it minimizes the condition number of $M^{-1}AD^2A^T$; however, determining the right hand side of (22) is as hard as solving the original system. It has been observed in practice that the number of conjugate gradient steps depends on the number of distinct eigenvalues, in fact typically on the number of distinct clusters of eigenvalues. Thus, it is desirable to pick M in order to cluster the eigenvalues of $M^{-1}AD^2A^T$. The Preconditioned Conjugate Gradient Algorithm for solving (21) is summarized in Figure 1. (Far more information on preconditioned conjugate gradient methods and related topics in numerical linear algebra can be found in the texts by Demmel [14] and Trefethen and Bau [49].)

For the minimum cost network flow problem on a graph $G = (\mathcal{V}, \mathcal{A})$ with vertices $\mathcal{V}$ and arcs $\mathcal{A}$, A is the node-arc incidence matrix (after deleting one row for each component, to ensure A has full row rank). At an interior point, the matrix AD^2A^T has nonzeroes on the diagonal and in position (i, j) for each $(i, j) \in \mathcal{A}$, assuming no numerical cancellation. With a preconditioned conjugate gradient algorithm, it is typically sufficient to solve (16) approximately as part of an infeasible interior point algorithm. The desired accuracy of the conjugate gradient algorithm can be chosen to depend on the primal residual r_b, defined in (8).

At a nondegenerate basic feasible solution, the elements of D corresponding to nonbasic variables are zero, so we have

$$AD^2A^T = BD_B^2B^T$$

where B denotes the columns of A corresponding to the basic variables and D_B the corresponding elements of D. At such a point, $M = BD_B^2B^T$ would be a perfect

1. Take Δy_0 equal to some initial guess.
2. $r_0 := g - AD_k^2 A^T \Delta y_0$.
3. $z_0 := M^{-1} r_0$.
4. $p_0 := z_0$.
5. $i = 0$.
6. While stopping criteria is not met
7. $q_i := AD^k A^T p_i$.
8. $\alpha_i := {z_i}^T r_i / {p_i}^T q_i$.
9. $\Delta y_{i+1} := \Delta y_i + \alpha_i p_i$.
10. $r_{i+1} := r_i - \alpha_i q_i$.
11. Solve $M z_{i+1} := r_{i+1}$.
12. $\beta_i := {z_{i+1}}^T r_{i+1} / {z_i}^T r_i$.
13. $p_{i+1} := z_{i+1} + \beta_i p_i$.
14. $i = i + 1$.
15. End While
16. $\Delta y := \Delta y_i$.

Figure 1: Preconditioned conjugate gradient algorithm for solving $AD^2 A^T \Delta y = g$.

preconditioner, in the sense that the conjugate gradient method would require just one step. Further, determining the right hand side of (22) is also straightforward: the basic feasible solutions are spanning trees, so finding $M^{-1} g$ requires only linear time. Therefore, once the interior point method gets close to an optimal solution, it is hoped that a preconditioner based on a nearby spanning tree would be a good choice, so preconditioners that use spanning trees are the most relevant. Some examples of the possible choices for preconditioners are the diagonal, the maximum spanning tree, or the diagonally compensated maximum spanning tree preconditioner.

The **Diagonal Preconditioner** was used by Resende and Veiga [45] in a dual affine algorithm for network flow problems. It is defined as:

$$M = \operatorname{diag}(AD^2 A^T) \tag{23}$$

This particular preconditioner is commonly used in the first few iterations of the preconditioned conjugate gradient algorithms, when it is most effective, an empirical observation somewhat supported by theoretical results in [24]. The diagonal preconditioner is known as the Jacobi preconditioner in numerical linear algebra, and it reduces the condition number of $M^{-1} AD^2 A^T$ to within a factor of $|\mathcal{V}|$ of its minimum value.

The **Maximum Spanning Tree preconditioner** is determined by using the current solution to select weights for the arcs of the graph and then finding a maximum spanning tree. It is defined as

$$M = A_{\mathcal{T}} {D_{\mathcal{T}}}^2 {A_{\mathcal{T}}}^T \tag{24}$$

where $A_{\mathcal{T}}$ contains the columns of A corresponding to the edges of the maximum spanning tree of $G = (\mathcal{V}, \mathcal{A})$ with

$$D_{\mathcal{T}} = \mathrm{diag}(d_{t_1}, ..., d_{t_{m-1}}) \tag{25}$$

where $t_1, ..., t_{m-1}$ are the edge indices of the maximum spanning tree and D is defined in (15). The edge weights for the spanning tree problem need to be defined appropriately. Portugal et al [42] suggest using the edge weight vector:

$$w = D_{\mathcal{T}}^2 e \tag{26}$$

where e is a vector of ones.

The **Diagonally Compensated Maximum Spanning Tree preconditioner** uses the matrix M defined as follows

$$M = A_{\mathcal{T}} D_{\mathcal{T}}^2 {A_{\mathcal{T}}}^T + \phi \ \mathrm{diag}(A_{\tilde{\mathcal{T}}} D_{\tilde{\mathcal{T}}}^2 {A_{\tilde{\mathcal{T}}}}^T) \tag{27}$$

where $G' = (\mathcal{V}, \mathcal{T})$ is maximum spanning tree of $G = (\mathcal{V}, \mathcal{A})$, $\tilde{\mathcal{T}} = \mathcal{A} - \mathcal{T}$ and ϕ is a nonnegative parameter. The diagonally compensated maximum spanning tree preconditioner is reduced to the maximum spanning tree preconditioner when $\phi = 0$. Some suggestions for ϕ used by Mehrotra and Wang in [34] are $\phi = 1$, $\phi = 0.1 \cdot \frac{\min(D_k)}{\max(D_k)}$, and $\phi = 10 \cdot \frac{\min(D_k)}{\max(D_k)}$.

Judice et al. [24] state and prove many theorems regarding the condition numbers of these preconditioners. They show that the condition number of AD^2A^T is bounded above by a function of the condition number of $D_{\mathcal{T}}^2$, where $D_{\mathcal{T}}$ consists of the elements of D corresponding to the edges of a maximum spanning tree for $G = (\mathcal{V}, \mathcal{A})$ with edge weights D. If the algorithm is converging to a primal nondegenerate basic feasible solution, then this gives a bound on the condition number of AD^2A^T. They are able to obtain stronger bounds on the condition number of $M^{-1}AD^2A^T$ with the use of the two tree-based preconditioners than the bound they obtain for either AD^2A^T itself or $M^{-1}AD^2A^T$ with the diagonal preconditioner. Further, they show that even in the presence of primal degeneracy of the optimal solution, the condition number of $M^{-1}AD^2A^T$ is uniformly bounded using the tree preconditioners, once the duality gap becomes small.

There are two stopping criteria for the Conjugate-Gradient Method suggested by Portugal et al in [42]. The first stopping criteria works well for the beginning iterations:

$$\| r_i \| \;\leq\; \beta_0 r_b \tag{28}$$

where r_i is the residual at the i^{th} iteration, x_k is the k^{th} interior point solution and β_0 is suggested to be 0.0999 in [42]. This stopping criterion may be too conservative in later iterations, once r_b is small, requiring too many conjugate gradient iterations.

The second stopping criteria exploits the fact that Δy is a direction, and that a dual steplength is still to be chosen. Therefore the magnitude of Δy is unimportant; what is important is that Δy point in approximately the right direction, that is, the angle θ between

$$AD^2A^T\Delta y_i \text{ and } g \tag{29}$$

should be small. This leads to the criterion:

$$| \, 1 - \cos\theta \, | \;<\; \epsilon^k_{cos} \tag{30}$$

where ϵ^k_{cos} is the tolerance for the interior point iteration k. Also, ϵ^k_{cos} can be tightened by multiplying by $\Delta\epsilon_{cos} < 1$ at each iteration. The angle θ can be computed

$$\cos\theta = \frac{| \, g^T(AD^2A^T)\Delta y_i \, |}{\| \, g \, \| \cdot \| \, (AD^2A^T)\Delta y_i \, \|} \tag{31}$$

or approximated (since calculating (31) is as expensive as a conjugate gradient iterate) by

$$\cos\theta \approx \frac{| \, g^T(g - r_i) \, |}{\| \, g \, \| \cdot \| \, (g - r_i) \, \|} \tag{32}$$

where r_i is the residual at the i^{th} iteration. This approximation works well when solving network linear programs.

The stopping criteria for the interior point method has two types of conditions. The first is the primal-basic (PB) stopping rule and the second the maximum flow (MF) stopping criterion; both are summarized in [42] and [44]. Both criteria use basis identification techniques, with the (PB) method exploiting the spanning tree found in the calculation of the preconditioner and the (MF) method using a method from the interior point literature. For a survey of basis identification techniques in interior point methods, see El-Bakry et al [16].

The PB stopping rule uses the tree found in the preconditioner and takes the edges of the tree to be the basic variables. Let $\mathcal{T}$ be the index set of the edges of the maximum spanning tree used in the creation of (25). Let

$$\Omega^+ = \left\{ i \in \{1,2,...,n\} \backslash \mathcal{T} : \frac{x_i}{z_i} > \frac{s_i}{w_i} \right\} \tag{33}$$

and set these nonbasic edges of the tree to their upper bounds. The basic variables must satisfy

$$A_{\mathcal{T}} x^*_{\mathcal{T}} = b - \sum_{i \in \Omega^+} u_i A_i \tag{34}$$

If this set of equations has a solution $0 \le x^*_{\mathcal{T}} \le u$ then $x^*_{\mathcal{T}}$ is a basic feasible solution. Let

$$F = \{ i \in \mathcal{T} : 0 < x^*_i < u_i \}. \tag{35}$$

be the set of edges where the dual slacks are zero. By orthogonally projecting the current dual solution onto the set F, we preserve complementary slackness by solving

$$\min_{y^* \in \mathbb{R}^m} \{ \parallel y^* - y^k \parallel : A_F^T y^* = c_F \} \tag{36}$$

which can be solved efficiently for network flow problems.

A feasible dual solution (y^*, z^*, w^*) can be found by changing the dual slacks

$$w^*_i = \begin{cases} -\delta_i & \text{if } \delta < 0 \\ 0 & otherwise \end{cases}$$

$$z^*_i = \begin{cases} 0 & \text{if } \delta < 0 \\ \delta_i & otherwise \end{cases}$$

where $\delta_i = c_i - A^T_{.i} y^*$. We can stop when $c^T x^* - b^T y^* + u^T w^* = 0$ because (x^*, s^*) is a primal feasible solution and (y^*, w^*, z^*) is a dual feasible solution. The idea of projecting an interior point onto the boundary in this manner was proposed by Ye [53] as a means to get finite convergence of interior point algorithms.

The other stopping criteria (MF) involves a different edge indicator, namely let edge i be inactive at its lower bound when

$$\tfrac{x_i}{z_i} < \xi \quad \text{and} \quad \tfrac{s_i}{w_i} > \xi^{-1}$$

and edge i be inactive at its upper bound when

$$\tfrac{x_i}{z_i} > \xi^{-1} \quad \text{and} \quad \tfrac{s_i}{w_i} < \xi$$

The other edges are then active. Using these active edges, we can form the maximum weighted spanning forest. Like before we can project to find y^* as in (36). However now we must also build a primal feasible x^*. The forest is extended slightly to $\overline{F} = \{i \in \{1, 2, ..., n\} : |c_i - A_{\cdot i}^T y^*| < \epsilon_r\}$ for some small tolerance ϵ_r (eg, $\epsilon_r = 10^{-8}$) and then the nonbasic variables are set:

$$x_i^* = \begin{cases} 0 & \text{if } i \in \Omega^- = \left\{j \in \{1, 2, ..., n\} \backslash \overline{F} : c_j - A_{\cdot j}^T y^* > 0\right\} \\ u_i & \text{if } i \in \Omega^+ = \left\{j \in \{1, 2, ..., n\} \backslash \overline{F} : c_j - A_{\cdot j}^T y^* < 0\right\} \end{cases}$$

Flow on the edges in $\overline{F}$ produces a *restricted network* which must satisfy

$$A_{\overline{F}} x_{\overline{F}} = b - \sum_{i \in \Omega^+} u_i A_i \tag{37}$$

and

$$0 \leq x_i \leq u_i, \ i \in \overline{F} \tag{38}$$

If a feasible primal solution exists on the restricted network then it is also complementary to y^*. As the algorithm proceeds, more edges will be declared inactive, resulting in a sparser spanning forest if the optimal solution is degenerate. In the limit, the algorithm will identify edges with flow strictly between the bounds, but some of these edges might not be included in the spanning forest if multiple primal optimal solutions exist. The expansion to $\overline{F}$ is necessary to recover these edges. Finding a feasible flow in the restricted network requires solving a maximum flow problem.

In summary, when using interior point methods to solve network flow problems, a preconditioned conjugate gradient algorithm can be used to approximately solve the direction finding subproblem at each iteration. It is recommended to start with the diagonal preconditioner, since it is easy to compute for beginning iterations, and then switch to a more complicated preconditioner based on spanning trees. The latter preconditioners have nice theoretical properties as the optimal solution is approached.

The primal-dual infeasible interior point method for network flow problems, PDNET, is available online at

http://www.research.att.com/˜mgcr/pdnet/

Much of this code is written in FORTRAN, and is described in detail in Patrício et al [41].

5 Multicommodity network flow problems

In this section, we consider multicommodity problems. These problems are considered in more detail in two other chapters of this book. In a multicommodity problem, the resources of the network are shared among several different competing commodities. For example, multiple different calls must be routed in a telephone network, and each call can be regarded as a commodity. It is not appropriate to aggregate the calls from a particular customer, rather the calls between each pair of customers must be considered separately. Similarly, in a traffic network, the set of vehicles moving from a particular origin to a particular destination should be considered a commodity.

If there are no capacity restrictions on the nodes and arcs of the graph then the multicommodity problem separates into several single commodity problems which can be solved independently. We will assume in this section that each arc e has capacity u_e. For simplicity, we will not assume capacities on the nodes. Let $\mathcal{K}$ denote the set of commodities. Let b^k and c^k denote the demand and cost vectors, respectively, for commodity k. For simplicity, we assume that the node-arc incidence matrix A is identical for each commodity. The basic minimum cost multicommodity network flow problem can be formulated as follows, with variables x_e^k equal to the amount of commodity k flowing along arc e:

$$\begin{array}{llcll} \min & \sum_{k\in\mathcal{K}} {c^k}^T x^k & & & \\ \text{subject to} & A x^k & = & b^k & \forall k \in \mathcal{K} \\ & \sum_{k\in\mathcal{K}} x_e^k & \leq & u_e & \forall e \in \mathcal{A} \\ & x^k & \geq & 0 & \forall k \in \mathcal{K} \end{array} \tag{39}$$

In contrast with the single commodity case, the constraint matrix in this formulation is not totally unimodular, so the optimal solution may not be integral. This formulation may have a huge number of variables for even a reasonably realistic multicommodity problem, since the number of variables is the product of the number of arcs and the number of commodities. Nonetheless, the constraint matrix has a structure that can be exploited to devise efficient interior point solvers, with better complexity bounds than with a naive invocation of an interior point method; see, for example, Kapoor and Vaidya [28, 29] and Kamath and Palmon [27]. As with single commodity problems, preconditioned conjugate gradient approaches have been used successfully for multicommodity problems. See, for example, Choi and Goldfarb [12], Yamakawa et al [52], Júdice et al [25], and Castro et al [9, 10]. Resende and Veiga [46] survey interior point approaches to multicommodity network flow problems.

We consider the algorithm of Castro [9] in a little more detail, in order to give a flavor of the numerical linear algebra techniques that can be used to speed up solution of multicommodity network flow problems. The crucial issue is the solution of (16).

For the formulation (39), (16) takes the block form

$$\begin{bmatrix} B & C \\ C^T & F \end{bmatrix} \begin{bmatrix} \Delta y_1 \\ \Delta y_2 \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} \tag{40}$$

where B is a block diagonal matrix, F is a symmetric matrix, C is a matrix, and Δy_1, Δy_2, g_1 and g_2 are vectors. This system can be solved by forming the Schur complement

$$H := F - C^T B^{-1} C, \tag{41}$$

solving

$$H\Delta y_2 = \tilde{g}_2 \tag{42}$$

for an appropriate $\tilde{g}_2$, and then solving

$$B\Delta y_1 = \tilde{g}_1 \tag{43}$$

for an appropriate $\tilde{g}_1$. For a general network, H has the same nonzero structure as $A^T(AA^T)^{-1}A$, so it is dense and (42) is hard to solve directly. Thus, Castro proposes solving (42) using a preconditioned conjugate gradient approach, with preconditioners based on a power series expansion.

An alternative approach to multicommodity network flow problems is a column generation method involving a path decomposition. This is useful when there is a large number of commodities; typically, the direct approach is better when the number of commodities is limited [9]. Without loss of generality, we assume that each commodity has a single source and a single sink (otherwise, the commodities can be disaggregated). Further, we assume that no two commodities have the same source and sink (otherwise the commodities can be aggregated). A set P_k of paths for each commodity $k \in \mathcal{K}$ are generated. Flow x_p is routed along these paths, and we let w_p be the cost of routing one unit along path p. The multicommodity network flow problem can then be written as

$$\begin{array}{lrcll} \min & \sum_{k\in\mathcal{K}} \sum_{p\in P_k} w_p x_p & & & \\ \text{subject to} & \sum_{p\in P_k} x_p & = & d_k & \forall k \in \mathcal{K} \\ & \sum_{p\in P_k, k\in\mathcal{K}: e\in p} x_p & \leq & u_e & \forall e \in \mathcal{A} \\ & x_p & \geq & 0 & \forall e \in \mathcal{A} \end{array} \tag{44}$$

where d_k denotes the demand for commodity k. If the sets P_k contain all possible paths then this is an exact formulation, but of course this is typically not computationally feasible due to the number of paths. Thus, this formulation is used in a column generation approach, with paths generated as needed in a subproblem. The path generation subproblem is a shortest path problem, with edge weights coming from the dual problem. This column generation approach has been used with the linear programs (44) solved using simplex (for example, Chardaire and Lisser [11]) and when the linear programs have been solved with an interior point method (for example, [11] and Goffin et al [19]). We discuss interior point column generation methods in more detail in §6.

6 Interior point column generation methods

Solving very large linear programming problems directly is sometimes computationally intractable. In this case, a column generation approach may be attractive, and we describe interior point column generation methods in this section. Integer programming problems can be solved using cutting plane methods. Adding a cutting plane in the primal linear programming problem is equivalent to adding a column in the dual problem, so the methods discussed in this section are also applicable to cutting plane algorithms.

6.1 Motivation

As already mentioned in §5, path-based formulations of multicommodity network flow problems are typically solved using column generation methods. Goffin et al [19] and Chardaire and Lisser [11] have both investigated interior point column generation methods for the solution of such problems. Gondzio and Kouwenberg [21] have solved large problems in financial optimization using interior point column generation methods; these problems involve assets that flow from one class to another over time, and hence the problem formulation has similarities with network flow problems and hence with problems in telecommunications.

Other applications of interior point column generation approaches to problems on networks and to integer programming problems include the solution of crew scheduling problems by Bixby et al [8], max cut problems by Mitchell [35] and linear ordering problems by Mitchell and Borchers [37]. Further, Goffin et al have solved Lagrangian relaxations of various problems, including integer programming problems, using interior point column generation methods; see, for example, Elhedhli and Goffin [17].

In telecommunications, many integer programming and / or column generation formulations of network design problems have been posed in the literature. See, for example, Atamtürk [3], Barahona [4], Bienstock and Muratore [6], Dahl and Stoer [13], Grötschel et al [22, 23], Myung et al [40], Raghavan and Magnanti [43], and Sherali et al [48]. Frequency assignment problems for cellular networks can also be cast as integer programming problems, as in, for example, Eisenblätter [15].

The introduction of stochasticity into linear telecommunications problems leads to growth in the size of the linear programs, so again a column generation approach may well be necessary. The classical L-shaped method for stochastic programs is a column generation method, of course. See Birge and Louveaux [7] and Kall and Wallace [26] for more information on stochastic programming. Robust optimization is another approach to handling uncertainty, and many robust optimization models are constructed as second order cone programs or semidefinite programming problems, and then solved using interior point methods; see Ben-Tal and Nemirovskii [5] for example.

6.2 Mechanics

Interior point column generation methods differ from simplex-based column generation approaches in one crucial respect: the linear programming subproblems are not solved to optimality at each stage, but solved approximately. This leads to the generation of dual cuts that cut off a larger proportion of the dual space, at least theoretically (see Goffin and Vial [20] and Mitchell [36] for recent surveys of the theoretical performance of interior point column generation methods).

When columns are added, it is desirable to return to the interior of the feasible region in order to allow fast convergence to the next approximate solution. This can be done by using a direction originally proposed in Mitchell and Todd [39], which can be motivated through consideration of Dikin ellipsoids.

The current relaxation can be written

$$\begin{array}{llcl} \min & c^T x & & \\ \text{subject to} & Ax & = & b \\ & x & \geq & 0 \end{array} \qquad \begin{array}{llcl} \max & b^T y & & \\ \text{subject to} & A^T y & \leq & c \end{array} \tag{45}$$

The complete problem can be written

$$\begin{array}{lcccl} \min & c^T x & + & h^T z & \\ \text{subject to} & Ax & + & Hz & = \; b \\ & & & x, z & \geq \; 0 \end{array} \qquad \begin{array}{llcl} \max & b^T y & & \\ \text{subject to} & A^T y & \leq & c \\ & H^T y & \leq & h \end{array} \tag{46}$$

The number of columns in H may be very large, possibly infinite. As the algorithm proceeds, the current relaxation is modified by adding columns from H to A (and dropping columns from A also, if the columns no longer appear to be useful). Any dual point y that is feasible in the complete problem (46) provides a lower bound on the optimal value. Any point x feasible in the current primal relaxation (45) is also feasible in (46) and so provides an upper bound on the optimal value. The column generation process can be stopped once these two bounds are close enough.

An interior point column generation algorithm approximately solves the current relaxation (45), calls an oracle to search for violated dual constraints, adds a subset of the primal columns / dual constraints found by the oracle, finds a new interior point to restart, and repeats the process. If the oracle is unable to find violated dual constraints, another interior point iteration is taken, and the oracle is called again.

There are two disadvantages to the use of interior point column generation methods. First, it is harder to exploit a warm start with an interior point method than with simplex. For smaller problems, when only a few columns are generated, the simplex algorithm can reoptimize far faster than an interior point method. However, this disadvantage is reduced for larger problems and when large numbers of columns are generated at once. In such a situation, the stronger columns generated by the interior point method can lead to faster convergence than when using the simplex method; see, in particular, [8, 37].

The second disadvantage is that the attempt to generate columns from more central points may fail to find violated dual constraints, so time is wasted in the calls to the oracle. This failure to find a violated dual constraint doesn't imply that the problem has been solved (as would be the case if the oracle was always accurate and if the current relaxation was solved to optimality). The remedy is to take another interior point iteration and try again. The required accuracy for the subproblem can be tightened as the algorithm proceeds in order to reduce the occurrence of this situation. Further, the failure of the oracle to find a violated dual constraint does provide some useful information: the dual solution must be feasible in the complete problem, so a bound on the optimal value is obtained, provided the oracle is correct in stating that there are no violated dual constraints.

It is of interest to note two ideas that appear both here and in §4, in totally different contexts. In particular, in both interior point column generation methods and in interior point algorithms for network flows, it suffices to solve the current problem approximately. Secondly, the accuracy to which the subproblem is solved is updated dynamically, depending on the progress of the algorithm, with more accurate solutions desired as the solution to the overall problem is approached.

More details of the implementation of interior point column generation methods can be found in Mitchell et al [38], as well as in the references given above.

7 Conclusions and Extensions

Interior point algorithms have found broad applicability in large scale linear programs arising in telecommunications, including successful implementations of interior point methods using preconditioned conjugate gradient algorithms for network flow problems. The use of interior point methods in column generation settings allows the possibility of solving very large linear programming problems. Interior point methods have been extended to other optimization problems such as semidefinite programming problems, second order cone programming problems, and general convex programming problems and these will allow the solution of more sophisticated models of problems in telecommunications. For example, network design problems have been solved by Lisser and Rendl using a semidefinite programming approach [30], and robust optimization problems such as antenna array design can be modeled as second order cone programs (for example, Lobo et al [31], Ben-Tal and Nemirovskii [5]). As a final note, interior point methods for nonlinear programming problems are surveyed by Forsgren et al [18].

References

[1] E. D. Andersen and K. D. Andersen. Presolving in linear programming. *Mathematical Programming*, 71:221–245, 1996.

[2] E. D. Andersen, J. Gondzio, C. Mészáros, and X. Xu. Implementation of interior point methods for large scale linear programming. In T. Terlaky, editor, *Interior Point Methods in Mathematical Programming*, chapter 6, pages 189–252. Kluwer Academic Publishers, 1996.

[3] A. Atamtürk. On capacitated network design cut-set polyhedra. *Mathematical Programming*, 92(3):425–452, 2004.

[4] F. Barahona. Network design using cut inequalities. *SIAM Journal on Optimization*, 6:823–837, 1996.

[5] A. Ben-Tal and A. Nemirovski. Robust optimization — methodology and applications. *Mathematical Programming*, 92(3):453–480, 2003.

[6] D. Bienstock and G. Muratore. Strong inequalities for capacitated survivable network design problems. *Mathematical Programming*, 89(1):127–147, 2000.

[7] J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer, New York, 1997.

[8] R. E. Bixby, J. W. Gregory, I. J. Lustig, R. E. Marsten, and D. F. Shanno. Very large-scale linear programming: a case study in combining interior point and simplex methods. *Operations Research*, 40:885–897, 1992.

[9] J. Castro. A specialized interior point algorithm for multicommodity flows. *SIAM Journal on Optimization*, 10(3):852–877, 2000.

[10] J. Castro and A. Frangioni. A parallel implementation of an interior-point algorithm for multicommodity network flows. In *Vector and Parallel Processing VECPAR 2000*, volume 1981 of *Lecture Notes in Computer Science*, pages 301–315. Springer-Verlag, 2001.

[11] P. Chardaire and A. Lisser. Simplex and interior point specialized algorithms for solving nonoriented multicommodity flow problems. *Operations Research*, 50(2):260–276, 2002.

[12] I. C. Choi and D. Goldfarb. Solving multicommodity network flow problems by an interior point method. In T. F. Coleman and Y. Li, editors, *Large-Scale Numerical Optimization*, pages 58–69. SIAM, Philadelphia, PA, 1990.

[13] G. Dahl and M. Stoer. A cutting plane algorithm for multicommodity survivable network design problems. *INFORMS Journal on Computing*, 10:1–11, 1998.

[14] J. W. Demmel. *Applied Numerical Linear Algebra*. SIAM, Philadelphia, PA, 1997.

[15] A. Eisenblätter. *Frequency Assignment in GSM Networks: Models, Heuristics, and Lower Bounds*. PhD thesis, TU-Berlin and Konrad-Zuse-Zentrum für Informationstechnik, Berlin, 2001.

[16] A. S. El–Bakry, R. A. Tapia, and Y. Zhang. A study of indicators for identifying zero variables in interior–point methods. *SIAM Review*, 36:45–72, 1994.

[17] S. Elhedhli and J.-L. Goffin. The integration of interior-point cutting plane methods within branch-and-price algorithms. *Mathematical Programming*, 100(2):267–294, 2004.

[18] A. Forsgren, P. E. Gill, and M. H. Wright. Interior methods for nonlinear optimization. *SIAM Review*, 44(4):525–597, 2002.

[19] J.-L. Goffin, J. Gondzio, R. Sarkissian, and J.-P. Vial. Solving nonlinear multicommodity network flow problems by the analytic center cutting plane method. *Mathematical Programming*, 76:131–154, 1997.

[20] J.-L. Goffin and J.-P. Vial. Convex nondifferentiable optimization: a survey focussed on the analytic center cutting plane method. *Optimization Methods and Software*, 17(5):805–867, 2002.

[21] J. Gondzio and R. Kouwenberg. High-performance computing for asset-liability management. *Operations Research*, 49(6):879–891, 2001.

[22] M. Grötschel, C. L. Monma, and M. Stoer. Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints. *Operations Research*, 40(2):309–330, 1992.

[23] M. Grötschel, C. L. Monma, and M. Stoer. Polyhedral and computational investigations for designing communication networks with high survivability requirements. *Operations Research*, 43(6):1012–1024, 1995.

[24] J. J. Júdice, J. M. Patrício, L. F. Portugal, M. G. C. Resende, and G. Veiga. A study of preconditioners for network interior point methods. *Computational Optimization and Applications*, 24:5–35, 2003.

[25] J. J. Júdice, L. F. Portugal, M. G. C. Resende, and G. Veiga. A truncated interior point method for the solution of minimum cost flow problems on an undirected multicommodity network. In *Proceedings of the First Portuguese National Telecommunications Conference*, pages 381–384, 1997. In Portuguese.

[26] P. Kall and S. W. Wallace. *Stochastic Programming.* John Wiley, Chichester, UK, 1994. Available online from the authors' webpages.

[27] A. P. Kamath and O. Palmon. Improved interior point algorithms for exact and approximate solution of multi-commodity flow problems. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 502–511, January 1995.

[28] S. Kapoor and P. Vaidya. Fast algorithms for convex programming and multicommodity flows. *Proceedings of the 18th annual ACM symposium on the theory of computing*, pages 147–159, 1988.

[29] S. Kapoor and P. Vaidya. Speeding up Karmarkar's algorithm for multicommodity flows. *Mathematical Programming*, 73:111–127, 1996.

[30] A. Lisser and F. Rendl. Graph partitioning using linear and semidefinite programming. *Mathematical Programming*, 95(1):91–101, 2003.

[31] M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret. Applications of second-order cone programming. *Linear Algebra and its Applications*, 284(1–3):193–228, 1998.

[32] N. Megiddo. On finding primal- and dual-optimal bases. *ORSA Journal on Computing*, 3:63–65, 1991.

[33] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2(4):575–601, 1992.

[34] S. Mehrotra and J.-S. Wang. Conjugate gradient based implementation of interior point methods for network flow problems. In L. Adams and J. L. Nazareth, editors, *Linear and nonlinear conjugate gradient-related methods*, pages 124–142. AMS/SIAM, 1996.

[35] J. E. Mitchell. Computational experience with an interior point cutting plane algorithm. *SIAM Journal on Optimization*, 10(4):1212–1227, 2000.

[36] J. E. Mitchell. Polynomial interior point cutting plane methods. *Optimization Methods and Software*, 18(5):507–534, 2003.

[37] J. E. Mitchell and B. Borchers. Solving linear ordering problems with a combined interior point/simplex cutting plane algorithm. In H. L. Frenk *et al.*, editor, *High Performance Optimization*, chapter 14, pages 349–366. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000.

[38] J. E. Mitchell, P. M. Pardalos, and M. G. C. Resende. Interior point methods for combinatorial optimization. In D.-Z. Du and P. M. Pardalos, editors, *Handbook of Combinatorial Optimization*, volume 1, pages 189–297. Kluwer Academic Publishers, 1998.

[39] J. E. Mitchell and M. J. Todd. Solving combinatorial optimization problems using Karmarkar's algorithm. *Mathematical Programming*, 56:245–284, 1992.

[40] Y.-S. Myung, H.-J. Kim, and D.-W. Tcha. Design of cummunication networks with survivability constraints. *Management Science*, 45(2):238–252, 1999.

[41] J. Patrício, L. F. Portugal, M. G. C. Resende, G. Veiga, and J. J. Júdice. Fortran subroutines for network flow optimization using an interior point algorithm. Technical Report TD-5X2SLN, AT&T Labs, Florham Park, NJ, March 2004.

[42] L. Portugal, M. Resende, G. Veiga, and J. Júdice. A truncated primal-infeasible dual-feasible network interior point method. *Networks*, 35:91–108, 2000.

[43] S. Raghavan and T. L. Magnanti. Network connectivity. In M. Dell'Amico, F. Maffioli, and S. Martello, editors, *Annotated bibliographies in combinatorial optimization*, pages 335–354. John Wiley, Chichester, 1997.

[44] M. G. C. Resende and G. Veiga. An efficient implementation of a network interior point method. In D.S. Johnson and C.C. McGeogh, editors, *Network Flows and Matching: First DIMACS Implementation Challenge,*, pages 299–348. American Mathematical Society, 1993. DIMACS Series on Discrete Mathematics and Theoretical Computer Science, vol. 12.

[45] M. G. C. Resende and G. Veiga. An implementation of the dual affine scaling algorithm for minimum cost flow on bipartite uncapacitated networks. *SIAM Journal on Optimization*, 3:516–537, 1993.

[46] M. G. C. Resende and G. Veiga. An annotated bibliography of network interior point methods. *Networks*, 42:114–121, 2003.

[47] C. Roos, T. Terlaky, and J.-Ph. Vial. *Theory and Algorithms for Linear Optimization: An Interior Point Approach*. John Wiley, Chichester, 1997.

[48] H. D. Sherali, J. C. Smith, and Y. Lee. Enhanced model representations for an intra-ring synchronous optical network design problem allowing demand splitting. *INFORMS Journal on Computing*, 12(4):284–298, 2000.

[49] L. N. Trefethen and D. Bau. *Numerical Linear Algebra*. SIAM, Philadelphia, PA, 1997.

[50] R. J. Vanderbei. *Linear Programming: Foundations and Extensions*. Kluwer Academic Publishers, Boston, 1996. Second Edition: 2001.

[51] S. Wright. *Primal-dual interior point methods*. SIAM, Philadelphia, 1996.

[52] E. Yamakawa, Y. Matsubara, and M. Fukushima. A parallel primal-dual interior point method for multicommodity flow problems with quadratic costs. *Journal of the Operations Reseach Society of Japan*, 39(4):566–591, 1996.

[53] Y. Ye. On the finite convergence of interior-point algorithms for linear programming. *Mathematical Programming*, 57(2):325–335, 1992.

[54] Y. Ye. *Interior Point Algorithms: Theory and Analysis*. John Wiley, New York, 1997.