RICE UNIVERSITY

Reoptimization in Interior-Point Methods with Application to Integer Programming

by

Cassandra M. McZeal

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree
Doctor of Philosophy

Approved, Thesis Committee:

Robert E. Bixby, Chairman
Noah Harding Professor Emeritus of
Computational and Applied Mathematics

Yin Zhang, Co-Chairman
Associate Professor of Computational and
Applied Mathematics

Ken Kennedy
Ann and John Doerr Professor in
Computational Engineering

William J. Cook
Noah Harding Professor of Computational
and Applied Mathematics

Richard A. Tapia
Noah Harding Professor of Computational
and Applied Mathematics

Houston, Texas
May, 1999

Abstract

Reoptimization in Interior-Point Methods with Application to Integer Programming

by

Cassandra M. McZeal

This thesis examines current reoptimization techniques for interior-point methods available in the literature and studies their efficacy in a branch-and-bound framework for 0/1 mixed integer programming problems. This work is motivated by the observation that there are instances of integer programming problems where each individual linear program generated in a branch-and-bound tree can be solved much faster by an interior-point algorithm than by a simplex algorithm, in spite of the fact that effective "warm-start" techniques are available for the latter but not for the former. Because of many unresolved issues surrounding effective reoptimization techniques for interior-point methods, interior-point algorithms have not been commonly used as linear programming solvers in a branch-and-bound framework.

In this work, we identify and examine a number of key factors that may affect and even preclude effective reoptimization for interior-point algorithms in the branch-and-bound framework, including change in optimal partition, distance to optimality, and primal infeasibility. We conclude that even though various "warm-start" techniques are capable of reducing the reoptimization cost to some extent, for certain prob-

lem instances a rapid reoptimization can not always be expected from interior-point methods due to their inherent limitations.

Continued research is needed in the direction of the present study in order to provide comprehensive guidelines for the most effective utilization of interior-point algorithms in a branch-and-bound algorithm.

Acknowledgments

Phillipians 4:13

I would like to express my sincere appreciation to the members of my thesis committee for their invaluable guidance and support during my time at Rice University. I thank Dr. Bill Cook for encouraging me to explain my ideas to others and for insightful discussions about integer programming. I thank Professor Richard Tapia and Professor Yin Zhang for their immeasurable assistance developing the ideas in this research. I am also indebted to Professor Zhang for his attention to detail and professionalism that forced me to raise my standards of acceptability. I thank Professor Ken Kennedy for providing insight as to how this work could be developed in a broader framework. I especially thank my primary advisor Professor Bob Bixby for introducing to me the field of Operations Research and encouraging me to "dive in."

Although he was not a part of my committee formally, I owe Dr. Amr El-Bakry an enormous debt of gratitude for his input and interest in my work.

I thank Michael Pearlman for the speed, efficiency, and cheerfulness with which he does his job. I thank Theresa Chatman, Fran Moshiri, Daria Lawrence, and Linda Neyra from the bottom of my heart for their amicable and professional assistance with all of the administrative details of graduate school. I also want to thank Theresa for her generous spirit and her friendship.

To all of my friends while here at Rice, Erica, Jen, Nikki, Rachel, Regina, Kymberly, Donald, Illya, Bill, Gwyneth, Monica, and Tony (we'll always have Baltimore), I thank each and every one of you for providing me with a much needed support system. I thank Jen and Gwyneth in particular for listening to my research-related ideas in their infancy stage. And speaking of infants, I thank Erica for all of the advice, guid-

ance, and baby gear as we both embarked upon the most rewarding journey of our lives.

I want to thank most especially Pamela Williams for leading the way. I cannot begin to enumerate all of the ways that she has positively affected my life both personally and professionally.

I would also like to thank my dear friend Shundria Riddick and her husband Chris for continuing to remind me Whose child I am.

I am forever grateful for the love and support of my family. I thank my parents, Paul and Thelma Moore, for instilling in me a love of mathematics and for their confidence in my abilities. To my in-laws, R.C. and Linda Robinson, I thank you not only for your support, but for taking me into your family as your own without a moment's hesitation. To all my "baby" sisters, Kimberly, Sheila, Dana, and my newest MéCheal, I thank you for believing.

Finally, and most importantly, I thank my husband, Marcellous. Without his abundant assistance (even when he felt overwhelmed), positive attitude, and unwavering faith, none of this would have been possible. Honey, thank you for helping me shout down the walls of Jericho.

This thesis is dedicated to my daughter, Madelyn; the joy in my world.

For Madelyn

Contents

Illustrations

Tables

Chapter 1

Introduction

Integer and combinatorial optimization involves the maximization or minimization of a linear objective function subject to a finite number of linear constraints and subject to integrality restrictions on a subset of the variables. These types of problems are referred to as linear mixed-integer programs (MIPs). Among the many applications of discrete optimization are planning problems such as portfolio analysis, design problems such as transport network design, and political problems such as the division of a region into electoral districts. Because linear mixed-integer programming problems have wide applicability, solving an instance of MIP is of great interest to researchers. Solving a linear mixed-integer programming problem means either to produce an optimal solution or to determine that the problem is infeasible or unbounded. Many algorithms for solving instances of MIP are special purpose, but currently only one algorithm is the basis of many commercial codes: branch and bound.

Branch-and-bound methods generate and solve a sequence of linear programs in order to find the solution to a given linear mixed-integer programming problem. The ability to reoptimize efficiently the next linear program from a previous one is a key issue in a branch-and-bound code. The dual simplex method, used almost exclusively in practice for reoptimization, enables the user to optimize the current linear program, using information from a previous optimization, usually in only a few iterations as compared to optimizing from scratch.

Although the dual-simplex method is typically the algorithm of choice for reoptimizations in a branch-and-bound code, there are instances of linear mixed-integer programming problems in which each linear program generated in a branch-and-

bound algorithm can be solved faster by an interior-point algorithm than by a simplex algorithm. However, the issues surrounding effective reoptimization techniques for interior-point methods in the branch-and-bound framework have not been fully resolved. For example, since it is widely accepted that the optimal solution for one problem cannot be used as the starting point in an interior-point algorithm for a perturbation of that problem, what type of "warm" starting point should be used? Also, once we determine the type of "warm" starting point, is it clear that this point can be found? This thesis examines current reoptimization techniques in the literature for infeasible primal-dual interior-point methods and evaluates their efficacy in the branch-and-bound framework in order to address these questions. The study shows that a rapid reoptimization should not always be expected and identifies a number of factors that affect the efficiency of reoptimization.

The thesis is organized as follows. In the remainder of Chapter 1, Section 1.1 provides formal definitions of a linear mixed-integer programming problem and of related terminology. A complete description of the branch-and-bound algorithm comprises Section 1.2. The infeasible primal-dual interior-point framework studied in this work as well as related terminology are defined in Section 1.3. Chapter 2 provides an overview of reoptimization techniques for interior-point methods from the literature. Chapter 3 describes the warm-start techniques implemented in this work. Chapter 4 describes the set of mixed-integer programming problems tested. Chapter 5 outlines the key factors affecting reoptimization studied in this work and motivates our numerical tests. Chapter 6 describes the numerical tests and gives the results of these tests along with some discussion about the results. Finally, concluding remarks can be found in Chapter 7.

1.1 Linear Mixed-Integer Programming

The linear mixed-integer programming (MIP) problem can be defined as follows:

$$\begin{aligned} \min \quad & c^T x + h^T y \\ \text{st} \quad & Ax + Gy \leq \ b \\ & x \in \mathbf{Z}_+^{n_1} \\ & y \in \mathbf{R}_+^{n_2} \end{aligned} \tag{1.1}$$

where $c \in \mathbf{R}^{n_1}$, $h \in \mathbf{R}^{n_2}$, $b \in \mathbf{R}^m$, $A \in \mathbf{R}^{m \times n_1}$, $G \in \mathbf{R}^{m \times n_2}$.

The set $S = \{x \in \mathbf{Z}_+^{n_1}, y \in \mathbf{R}_+^{n_1}, Ax + Gy \leq b\}$ is called the *feasible region*, and an $(x, y) \in S$ is called a *feasible solution*. If $S \neq \emptyset$, then the problem is said to be feasible. The function $z = c^T x + h^T y$ is the *objective function*, and an $(x, y) \in S$ for which this function is optimized is called an *optimal solution*. If $(x^*, y^*) \in S$ is an optimal solution, then $z^* = c^T x^* + h^T y^*$ is the *optimal value* of the solution. A feasible instance of MIP may also have no optimal solution, a situation that indicates that the problem is unbounded. An instance of MIP is *unbounded* if for every real number there exists a feasible solution for which the objective function value is smaller than that number.

Two special cases of the MIP problem deserve notice. When none of the variables is restricted to hold integer values, then the problem is a *linear programming problem*. Conversely, when all of the variables are restricted to hold integer values, then the problem is a *pure integer programming problem*.

1.2 Branch and Bound

Land and Doig [37] developed the first branch-and-bound algorithm for MIP in 1960. Branch and bound can be described as an enumerative relaxation algorithm for solving instances of MIP. In this algorithm, a divide-and-conquer approach is taken. The feasible region S is divided into k smaller sets such that $S = \cup_{i=0}^{k-1} S^i$, and a relaxation of the problem is solved over each of these sets. The divisions used in a branch-and-bound algorithm can be represented in a branch-and-bound search tree as shown in Figure 1.1. For example, in this tree S^0 is a *parent node* and the smaller sets into which it is divided, S^{01}, S^{02}, and S^{03}, are its *children nodes*. Likewise, S^{02} is a parent node whose children nodes are S^{021} and S^{022}.

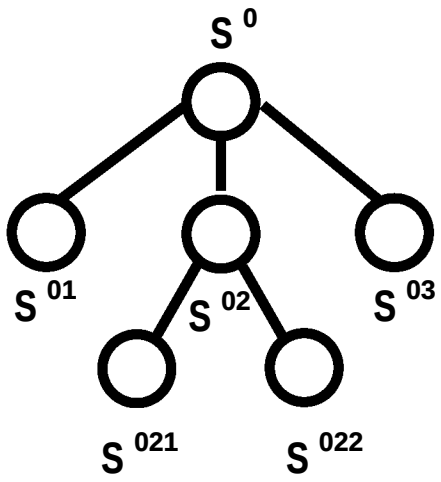

Figure 1.1 A branch-and-bound search tree in which S^0 is a parent node and S^{01}, S^{02}, and S^{03} are its children nodes.

Dividing the feasible region into a number of subsets whose union is the feasible region is valid since

$$\min\{c^T x + h^T y : (x, y) \in S\} = \min_{i=0}^{k-1}\{\min\{c^T x + h^T y : (x, y) \in S^i\}\}.$$

The divisions are used to enforce the integrality of the integer variables so that over each S^i it is not necessary to solve the smaller MIPs created, but only their relaxations, linear programs. The branch-and-bound algorithm used in this work is as follows:

Algorithm 1.1 *(Branch and Bound)*
Given $p = \{\min\ c^T x + h^T y : (x, y) \in S\}$ where $S = \{x \in \mathbf{Z}_+^{n_1}, y \in \mathbf{R}_+^{n_1}, Ax + Gy \leq b\}$, let $z_{up} = +\infty$, $z_{lo}^0 = -\infty$, $P = \{p\}$, and $S^0 = S$. While $P \neq \emptyset$ do the following:

(1) Select and delete a problem p^i from P.

(2) Solve a relaxation of $p^i = \{\min\ c^T x + h^T y : (x, y) \in S^i\}$ for $(x, y)^i$ with optimal value z^i, if one exists.

(3) If p^i is infeasible, then go to step (1). If $z^i \geq z_{up}$, then go to step (1). If x of $(x, y)^i$ is not integral, then go to step (4). Otherwise (p^i is feasible, $z^i < z_{up}$, and x of $(x, y)^i$ is integral), set $z_{up} := z^i$ and delete all problems p^j from P with $z_{lo}^j \geq z_{up}$ and go to step (2).

(4) Let $\{S^{ij}\}_{j=0}^{k-1}$ be a division of S^i. Add problems $\{p^{ij}\}_{j=0}^{k-1}$ to P, where $z_{lo}^{ij} := z^i$. Go to step (2).

At the end of this algorithm, the set P of problems defined over each division is empty, and if $z_{up} < \infty$, then $(x, y)^k$ which yielded z_{up} is an optimal solution. If $z_{up} = \infty$, then the MIP problem is infeasible.

1.3 Infeasible Primal-Dual Interior-Point Method

Although the branch-and-bound algorithm can be stated as a step-by-step process, there are many aspects to consider when formulating a branch-and-bound code. Deciding how to divide the feasible set and in what order to solve the problems

over these divisions are two such considerations. One of the most computationally important aspects involves solving the linear programs.

Consider the following linear programming problem:

$$\begin{aligned} \min \quad & c^T x \\ \text{st} \quad & Ax = b \\ & 0 \leq x \leq u \end{aligned} \tag{1.2}$$

where $x,\ u,\ c \in \mathbf{R}^n,\ b \in \mathbf{R}^m,\ A \in \mathbf{R}^{m \times n}$ $(m < n)$ and A has rank m. This is referred to as the primal problem. By adding the slack variables to the second set of constraints on x, we can convert (1.2) to standard form:

$$\begin{aligned} \min \quad & c^T x \\ \text{st} \quad & Ax = b \\ & x + s = u \\ & x \geq 0 \\ & s \geq 0 \end{aligned} \tag{1.3}$$

where $s \in \mathbf{R}^n$.

A dual of this problem is:

$$\begin{aligned} \max \quad & b^T y - u^T w \\ \text{st} \quad & A^T y + z - w = c \\ & z \geq 0 \\ & w \geq 0 \end{aligned} \tag{1.4}$$

where $y \in \mathbf{R}^m$, and $z, w \in \mathbf{R}^n$. The variables y and w are known as the dual variables and z as the dual slack variable.

The optimality conditions for a solution to the primal and dual problems can be stated as a special case of the optimality conditions for general constrained optimization. These conditions for optimality of a solution, which are called the Karush-Kuhn-Tucker (KKT) conditions, consist of a system of linear-quadratic equations with nonnegativity constraints on some variables. That is, (x^*, s^*) is a solution to (1.3) and (y^*, z^*, w^*) is a solution to (1.4) if and only if the following holds for $(x^*, s^*, y^*, z^*, w^*)$:

$$F(x, s, y, z, w) = \begin{pmatrix} Ax - b \\ x + s - u \\ A^T y + z - w - c \\ XZe \\ SWe \end{pmatrix} = 0$$

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

where $X = diag(x)$, $Z = diag(z)$, $S = diag(s)$, $W = diag(w)$ and e is the n-vector of all ones.

Primal-dual interior-point methods apply variants of Newton's Method to the KKT conditions in order to find primal-dual solutions (x, s, y, z, w). A *primal-dual solution* is a point (x, s, y, z, w) such that (x, s) solves (1.3) and (y, z, w) solves (1.4). These methods also modify search directions and step lengths so that the nonnegativity constraints, $(x, z, s, w) \geq 0$, are strictly satisfied at each iteration. Newton's Method for systems of nonlinear equations can be defined as follows:

Algorithm 1.2 *(Newton's Method for Systems of Nonlinear Equations)*
Given $F : \mathbf{R}^n \to \mathbf{R}^n$ continuously differentiable and $v^0 \in \mathbf{R}^n$. At each iteration k until $||F(v^k)||$ is "small,"

(1) $F'(v^k)\Delta v^k = -F(v^k)$

(2) $v^{k+1} = v^k + \Delta v^k$.

Under standard conditions, we know that Newton's Method is a highly effective method for solving $F(x, s, w, z, y)$. If started close enough (problem dependent) to the solution, Newton's Method gives Q-quadratic convergence. Convergence is not assured in general, however, if Newton's Method is started too far away. The instance of primal-dual interior-point methods used here can be described as the perturbed and damped composite Newton's Method.

The dampening is introduced in equation (2). A step length α^k is used to facilitate global convergence:

$$v^{k+1} = v^k + \alpha^k \Delta v^k.$$

Additionally, by using the composite Newton's Method we can increase the convergence rate of the algorithm to Q-cubic without incurring a substantial amount of extra work. In this variant of Newton's Method, an intermediate point $\hat{v}$ is calculated and the same Jacobian is used twice:

$$F'(v^k)\Delta\hat{v} = -F(v^k), \quad \hat{v} = v^k + \Delta\hat{v}$$

$$F'(v^k)\Delta v^k = -F(\hat{v}), \quad v^{k+1} = \hat{v} + \Delta v^k.$$

Ensuring the nonnegativity of the iterates ($(x, z, s, w) \geq 0$) is an issue that can not be ignored when using Newton's Method to solve linear programming problems. It is well-known that if any variable involved in a complementarity pair is allowed to have value zero, then that variable will stay at zero in all subsequent iterations of Newton's Method (commonly called "sticking to the wall"). Thus, it is necessary not only to maintain nonnegativity, but also strict positivity. Also, it is computationally

advantageous to avoid any variable attaining a positive value that is prematurely too "small." To address this problem, a perturbation parameter μ is introduced in the complementary equations, $XZe = 0, SWe = 0$, at each iteration. Thus, we have

$$XZe - \mu e = 0,$$

$$SWe - \mu e = 0$$

where μ converges to zero in a controlled manner.

The arc of strictly feasible points parameterized by μ is called the *central path* and is defined as follows:

$$\mathcal{C} = \{(x_\mu, s_\mu, z_\mu, w_\mu) | \mu > 0, (x_\mu, s_\mu, z_\mu, w_\mu) \in \mathcal{F}^0, XZe = \mu e, SWe = \mu e\}$$

where

$$\mathcal{F}^0 = \{(x, s, y, z, w) | Ax = b, x + s = u, A^T y + z - w = c, (x, s, z, w) > 0\}.$$

The set $\mathcal{F}^0$ is referred to as the primal-dual strictly feasible set and is a subset of the feasible set

$$\mathcal{F} = \{(x, s, y, z, w) | Ax = b, x + s = u, A^T y + z - w = c\}.$$

A point (x, s, y, z, w) is called *strictly feasible* if it is in $\mathcal{F}^0$. If $\mathcal{C}$ converges to any point as μ converges to zero, then it converges to a primal-dual solution of the linear programming problem.

The *optimal partition* is defined by two index sets $\mathcal{B}$ and $\mathcal{N}$. For any optimal solution $(x^*, s^*, y^*, z^*, w^*)$ to a given linear program, if we let $\hat{x}^* = (x^*, s^*)$ and $\hat{z}^* = (z^*, w^*)$, then

$$\mathcal{B} = \{i \in 1, ..., 2n | \ \hat{x}_i^* \neq 0\}$$

and

$$\mathcal{N} = \{i \in 1, ..., 2n | \ \hat{z}_i^* \neq 0\}.$$

These sets are invariant over the interior of the solution set whenever the solution set is not a singleton.

The sets $\mathcal{B}$ and $\mathcal{N}$ form a partition of the index set $\{1, 2, ..., 2n\}$; each index i belongs to either $\mathcal{B}$ or $\mathcal{N}$ but not both. It is clear from an examination of the complementarity equations of the KKT system

$$Xz = 0$$

$$Sw = 0$$

that $\mathcal{B}$ and $\mathcal{N}$ are disjoint. The result $\mathcal{B} \cup \mathcal{N} = \{1, 2, ..., 2n\}$ is know as the Goldman-Tucker theorem [20]. Primal-dual solutions that are strictly feasible are known as *strictly complementary solutions*, and the Goldman-Tucker theorem [20] guarantees that at least one such solution exists. A primal-dual solution is called *degenerate* if it is not strictly complementary.

Kojima, Mizuno, and Yoshishi [36] first introduced the use of a perturbation in a primal-dual interior-point algorithm. The algorithm they described required a nonnegative feasible starting point, in many cases a prohibitive requirement. However, Lustig, Marsten, and Shanno [43] later show that the algorithm introduced by Kojima, Mizuno, and Yoshishi can be modified so that an infeasible starting point can be used. The state-of-the-art infeasible interior-point algorithm, described by Mehrotra [47], is called the predictor-corrector method and is an enhanced version of the infeasible interior-point algorithm introduced by Lustig, Marsten, and Shanno [43]. Tapia, Zhang, Saltzman and Weiser [57] show that Mehrotra's method can be seen as a modification of the perturbed damped composite Newton's Method.

If we define $\bar{e}$ to be the vector in $\mathbf{R}^{m+4n}$ such that the first $m + 2n$ components are zero and the last $2n$ are one, then the algorithmic framework for the infeasible

primal-dual interior-point method (Mehrotra's predictor-corrector algorithm) can be seen as follows:

Algorithm 1.3 *(Infeasible Predictor-Corrector Interior-Point Algorithm)*
Given $v^0 > 0$, at each iteration k until $||F(v^k)||$ is "small,"

(1) Solve $F'(v^k)\Delta\hat{v} = -F(v^k)$ for $\Delta\hat{v}$.

(2) Set $\hat{v} = v^k + \Delta\hat{v}$.

(3) Choose $\mu^k > 0$.

(4) Solve $F'(v^k)\Delta v^k = -F(\hat{v}) + \mu^k\bar{e}$ for Δv^k.

(5) Choose $\alpha^k \in (0, 1]$.

(6) Set $v^{k+1} = v^k + \alpha^k(\Delta\hat{v} + \Delta v^k)$.

The implementation details of this algorithm such as the starting point, the stopping criteria, and the computation of certain parameters have been intentionally omitted at this point. Chapters 3 and 6 will outline many of the particulars of each of these issues relevant to this work. The next chapter will give some background on some existing methods used to find starting points that utilize information from a previous optimization for interior-point algorithms.

Chapter 2

Reoptimizing the Linear Programming Subproblems

In 1984, Karmarkar [34] introduced the use of interior-point methods as an effective computational tool for solving linear programming problems. Interior-point methods generate iterates that travel through the interior of the feasible region in order to find an optimal solution. Until this time, the simplex method, introduced by Dantzig [11] in 1947, had been the primary method for solving linear programs. Unlike an interior-point method, the simplex method systematically examines the vertices of the feasible region in an effort to find an optimal solution.

Associated with the simplex method (referred to as the primal-simplex method) introduced by Dantzig is the dual-simplex method, introduced by Lemke [39] in 1954, which applies the simplex method to the dual linear programming problem as opposed to the primal linear programming problem. The dual-simplex method can be used to find quickly a new optimal solution after certain changes have been made to a previously solved problem. Fixing to one bound or another a fractional variable in a solution to a parent node problem to create a child node problem, as is done in a branch-and-bound algorithm, is one such change that can be handled efficiently by the dual-simplex method since the parent optimal solution remains dual feasible. However, how to effectively reoptimize using an interior-point algorithm after changing a problem remains unresolved.

In 1996, Gondzio and Terlaky [25] divide the reoptimization techniques attempted for interior-point methods into three categories: shifted-barrier function methods (Freund [16], 1991), modified barrier function methods (Polyak [53], 1992), and per-

turbation methods (Lustig, Marsten and Shanno [43, 44], 1990 and 1994). Additionally, Borchers [6] in 1992 described a method designed specifically for use in a branch-and-bound code. More recently in 1996, Gondzio [23] designed a reoptimization method for a cutting-plane scheme. Each of the next five sections will give an overview of these methods.

2.1 Shifted-Barrier Methods

Freund [16] examines the theoretical efficiency of a shifted-barrier function algorithm for solving

$$\begin{aligned} \min \quad & c^T x \\ \text{st} \quad & Ax = b \\ & x \geq 0. \end{aligned}$$

The shifted-barrier function method described solves a sequence of shifted-barrier problems $\text{Sh}(\epsilon)$ of the form

$$\begin{aligned} \text{Sh}(\epsilon): \quad \min_x \quad & c^T x - \epsilon \sum_{j=1}^{n} ln(x_j + \epsilon h_j) \\ \text{st} \quad & Ax = b \\ & x + \epsilon h > 0 \end{aligned}$$

where h is a given and fixed strictly positive shift vector and for a sequence of values $\epsilon > 0$ that converge to zero. Freund points out that the main advantage of using the shifted-barrier problem $\text{Sh}(\epsilon)$ to solve a linear program is that the algorithm used can be initiated with a "warm start." A "warm start" in this context means a guess of the solution $\bar{x}$ to the linear program that is perhaps not feasible (meaning that either $Ax \neq b$ or $x \ngeq 0$ or both) but is possibly very close to the optimal solution. The author

shows that if one is given an approximate center point of the dual feasible region of the problem to be solved, along with the initial guess $\bar{x}$, then the shift vector h can be chosen in such a way that establishes the theoretical efficiency of the shifted-barrier algorithm.

In a later paper, Freund [17] extends the shifted-barrier function approach and constructs a potential function for which he describes a potential-function reduction algorithm. The potential function is given by

$$F(x, B) = q \ln(c^T x - B) - \sum_{j=1}^{n} \ln(x_j + h_j(c^T x - B))$$

where $q = n + n^{1/2}$, h is a given strictly positive shift vector, and B is a lower bound on the optimal value of the linear program. The algorithm developed solves the linear program directly from an infeasible "warm start." A "warm start" $\bar{x}$ in this context means that $A\bar{x} = b$, but $\bar{x} \ngeq 0$.

In both of these algorithms, Freund attempts to deal with the issue of keeping the iterates nonnegative. Instead of forcing the iterates to be nonnegative at every iteration, he allows them to attain temporarily negative values while gradually moving the iterates back into the positive orthant.

2.2 Modified Barrier Function Method

Like Freund, Polyak [54] devises a Newton Modified Barrier Function Algorithm that also attempts to address the issue of allowing negative iterates before convergence. Here the author replaces the nonnegativity conditions

$$x \geq 0 \qquad \text{and} \qquad z \geq 0$$

with

$$x + \epsilon\, e \geq 0 \qquad \text{and} \qquad z + \epsilon\, e \geq 0$$

where ϵ is some positive scalar and e is the vector containing all ones.

In a later paper [53], the author shows theoretically that for the Newton Modified Barrier Function Algorithm there exists a "hot start" point from which the algorithm has a better rate of convergence for nondegenerate problems. This "hot start" point is not used for reoptimizations, but is realized during the solution process. Also, Jensen, Polyak and Schneur [32] show numerically that the algorithm attains this "hot start" point.

2.3 Perturbation of Optimal Solution

Lustig, Marsten and Shanno [44] examine starting and restarting the infeasible primal-dual interior-point method. For restarting, the problem set they examine is the DINAMICO model from Manne [45]. An example of this model can be found in NETLIB [18] as *stair*. Lustig, Marsten, and Shanno [44] examine solving several linear programs that differ only in the objective function used. The authors solve a base-case linear program and use the solution from this linear program to solve the others. The optimal solution from the base case is perturbed so that any components of nonnegative primal and dual variables that fall below a given threshold ϵ are set to ϵ. (They set $\epsilon = 0.3$.) The algorithm then adjusts μ and solves to optimality. The authors report a fifty-percent or better reduction in the number of iterations required to start from this perturbed solution than from a "cold start."

In a later paper, Lustig, Marsten and Shanno [43] proposed a variant of this earlier algorithm. Again, they propose to shift small nonnegative primal and dual variable values to larger positive values. The authors implement the idea of a shift vector as is used in the shifted-barrier function method by Freund [17]. They derive a specific shift vector to be applied to the nonnegative variables that is based on scaling the constraint matrix A. Additionally, the authors scale the shift vector so that once the

nonnegative iterates are perturbed they lie in a critical neighborhood $\mathcal{N}$ as defined by Kojima, Megiddo, and Mizuno [35].

2.4 Early Termination

In his thesis, Borchers [6] implements an experimental branch-and-bound code that uses a primal-dual interior-point method to solve the linear programming subproblems generated. A heuristic is described for detecting fractional variables before the completion of the optimization process of each relaxation, and a warm-start heuristic based on the work of Lustig, Marsten, and Shanno [44] is used. The fractional variable detection heuristic is based upon Tapia's indicators studied by El-Bakry, Tapia, and Zhang [13]. These indicators are designed to detect the zero/nonzero partition of variables as they converge towards optimality in the primal-dual interior-point algorithm. Borchers uses these indicators to detect whether or not a binary variable x_i and its complement $1 - x_i$ are tending toward a nonzero value. If so, then the author assumes that the variable will be fractional at optimality. Thus, branching will be necessary, and the optimization can be terminated early. In order to end the optimization early, the author also requires that the primal and dual feasibility and duality gap are within a certain range and that the binary variables are neither zero nor one.

The solution to the parent node that is found using the early termination procedure is used as the initial solution for the child node in the branch-and-bound search tree. To reoptimize the child nodes in the branch-and-bound tree, Borchers uses the idea of perturbation introduced by Lustig, Marsten, and Shanno [44]. The centering parameter is then calculated as a function of the infeasibility introduced. In later papers, Mitchell [49, 50, 51] extends this work by showing that the dual slack values generated in an interior-point algorithm can be used to fix binary variables to their

bounds in the same manner that the reduced costs are used to fix variables in the simplex method.

2.5 Centered Iterate

Gondzio [23] develops a procedure for reoptimizing the sequence of linear programs generated by a cutting-plane scheme (see Chvátal [8]). The author suggests that difficulties arising from reoptimizing interior-point methods stem from the use of a near optimal solution as the starting point for the next optimization. Gondzio points out that a near optimal solution is in most cases too near the boundary and therefore makes a particularly difficult starting point. He also suggests that merely perturbing a solution as had been done by Lustig, Marsten, and Shanno [44], Borchers [6], and Mitchell [49, 50, 51] is not sufficient for obtaining a good starting point.

Instead of using the final solution to one optimization as the starting point for the next, Gondzio proposes dividing the solution process into two phases. In the first phase, a point thought to be good for the next reoptimization is identified, and from this point the current optimization is completed. The author refers to the point generated in the first phase as a "close-to-optimality approximate analytic center" because the point is nearly feasible and has well-balanced complementarity products. The next chapter will focus on the details of this method and introduce a method for finding such a point for use within a infeasible predictor-corrector primal-dual interior-point framework.

Chapter 3

Reoptimization Methods Implemented

3.1 Centered Iterate

3.1.1 Background

Gondzio [23] implements a warm-start procedure for use in an infeasible primal-dual interior-point algorithm when solving the restricted master problem from a column generation technique (see Chvátal [8]).

The author is concerned with the following problem:

$$\begin{aligned} \min \quad & \tilde{c}^T \tilde{x} \\ \text{st} \quad & \tilde{A}x = \quad b \\ & \tilde{x} \geq 0 \end{aligned} \tag{3.1}$$

where $\tilde{c}, \tilde{x} \in \mathbf{R}^N$, $b \in \mathbf{R}^m$, and $\tilde{A} \in \mathbf{R}^{m \times N}$. The number of variables, N, is assumed to be much larger than the number of constraints, m. In fact, all the columns in the matrix $\tilde{A}$ may not be given explicitly, and N can be infinite. This problem is referred to as the *full master problem*. To handle the difficulties arising from not having a complete representation of the full master problem, the strategy taken in a delayed column generation technique is to work with only relatively few columns at a time and generate more as needed. Thus, *a restricted master problem*,

$$\begin{aligned} \min \quad & c^T x \\ \text{st} \quad & Ax = \ b \\ & x \geq 0 \end{aligned} \tag{3.2}$$

where $c, x \in \mathbf{R}^n$, $b \in \mathbf{R}^m$, and $A \in \mathbf{R}^{m \times n}$, is solved. In this restricted master problem, $n \ll N$ and A is a submatrix of $\tilde{A}$. Once a solution to this problem is found, information from the dual solution is used to append more columns to A. This process is repeated until an optimal solution is found for the full master problem (3.1) or no further columns can be found.

Adding columns in this way to the restricted master problem (3.2) is equivalent to adding rows (or cutting planes) to the dual problem:

$$\begin{aligned} \max \quad & b^T y \\ \text{st} \quad & A^T y \leq \ c \end{aligned} \tag{3.3}$$

where $c \in \mathbf{R}^n$, $b, y \in \mathbf{R}^m$, and $A \in \mathbf{R}^{m \times n}$. Because a "good" dual solution that is sufficiently close to optimality is needed in order to make wise choices for columns (rows) to add to A (A^T), Gondzio elects to use a primal-dual interior-point algorithm applied to the dual problem within the cutting-plane scheme to solve the full master problem. In a *cutting-plane algorithm*, a linear program is solved and the solution is examined. If the solution does not meet optimality criteria, then constraints or cuts are generated that this solution violates. These constraints are added to the constraint matrix, and the problem is reoptimized. This process is repeated until optimality or until no further constraints can be generated.

Since, like branch and bound, a cutting-plane method involves solving a sequence of linear programs, Gondzio is interested in effective reoptimization techniques for

the primal-dual interior-point algorithm. Unlike other reoptimization techniques, Gondzio's method does not attempt to use an optimal or near-optimal solution (or a perturbation of such a solution) as the starting point. He calls the starting point that he uses a *close-to-optimality approximate analytic center.* Gondzio defines "close to optimality" as a solution that has 3 to 4 digits of accuracy and expects that from such a point optimality would be reached in only a few iterations. By "approximate analytic center" the author is referring to a point that is nearly primal and dual feasible, and the difference between the minimum and maximum of the products in the complementarity equations is not "large."

In order to find a close-to-optimality analytic center, Gondzio proposes to divide the optimization into two phases. In the first phase, the algorithm iterates until a solution that fits the criteria for a close-to-optimality analytic center is found and saved for future use a warm start. In the second phase, the algorithm continues the optimization process until the conditions for optimality of the problem are met.

Gondzio's reoptimization technique is applied in the context of the multiple centrality corrections primal-dual interior-point algorithm [22]. Like Mehrotra's predictor-corrector method introduced in Chapter 1, the multiple centrality corrections algorithm is based on Newton's Method, and the Newton direction taken is computed in two parts. The first part of the step taken in each iteration is the affine-scaling step which does not include the barrier parameter μ. In the second part, μ is chosen and the computation of a "corrector" term for the affine direction is an iterative process that is repeated a number of times that is determined heuristically. These corrector terms are designed to improve the centrality of the next iterate and increase the step sizes in the primal and dual spaces by encouraging iterates not to fall on the central path, but to lie in a loose neighborhood of the central path. Gondzio does this by

changing the perturbation vector μe in the complementarity equations,

$$XZe - \mu e = 0,$$

based on the belief that requiring the iterates to go to the central path is too high a goal to be attainable without a new factorization.

Define $\bar{e}$ to be the vector in $\mathbf{R}^{m+2n}$ such that the first $m+n$ components are zero and the last n are one. Let $\pi(a|B)$ where $a \in \mathbf{R}^p$ and $B \subset \mathbf{R}^p$ be the component-wise projection of a onto B. Also define $C = [\gamma_1, \gamma_2]^p$ as a hypercube in $\mathbf{R}^p$ whose sides are of length $\gamma_2 - \gamma_1$. Using these definitions, the framework for Gondzio's multiple centrality corrections method can be seen as follows:

Algorithm 3.1 *(Multiple Centrality Corrections Interior-Point Algorithm)* Given $(x^0, y^0, s^0) > 0$, at each iteration k until $||F(x^k, y^k, s^k)||_2$ is "small,"

(1) Solve $F'(x^k, y^k, s^k)\Delta_p = -F(x^k, y^k, s^k)$ for Δ_p.

(2) Choose α_p.

(3) Compute Δ and choose α.

While $j < J$ do:

(a) Set $(\hat{x}, \hat{y}, \hat{z}) = (x^k, y^k, s^k) + \alpha_p \Delta_p$.

(b) Choose $\mu > 0$

(c) Set $v_t = \pi(\hat{X}\hat{z}|H) \in \mathbf{R}^n$ where $H = [\beta_{min}\mu, \beta_{max}\mu]^{2n}$

(d) Solve $F'(x^k, y^k, s^k)\Delta_m = -F(\hat{x}, \hat{y}, \hat{z}) + (v_t - \hat{X}\hat{z})\bar{e}$ for Δ_m.

(e) Choose α for the composite direction $\Delta = \Delta_p + \Delta_m$

(f) If $\alpha \geq \alpha_p + \delta$ where δ is an "acceptable" increase in step size, then $j = j + 1$ and $\Delta_p = \Delta$.

(g) Else $\Delta = \Delta_p$ and $j = J$.

(4) Set $(x^{k+1}, y^{k+1}, s^{k+1}) = (x^k, y^k, s^k) + \alpha\Delta$.

For the details of this algorithm including parameter settings, the reader is referred to Gondzio [22].

In order to save a future warm start in the multiple centrality corrections framework, Gondzio modifies Step (2b) of the algorithm. Once the relative duality gap of the current iterate falls below a specified tolerance, μ is set equal to a fraction of the average value of the pairwise complementarity products. The value of μ is not changed until a warm start is found. After μ has been fixed, an iterate is considered a warm-start point when the relative primal and dual feasibilities have fallen below a certain tolerance and when all of the complementarity pairs fall between $\bar{\beta}_{min}\mu$ and $\bar{\beta}_{max}\mu$. Note that $\bar{\beta}_{min}$ and $\bar{\beta}_{max}$ differ from β_{min} and β_{max} used earlier and are, in fact, more restrictive than the earlier parameters since according to Gondzio a tighter neighborhood increases the quality of the warm-start point saved. Also, $\beta_{min} = 1/\beta_{max}$. Once the point to be used for future warm starts has been saved, μ is recomputed as before and the algorithm continues to proceed toward optimality.

3.1.2 Centered Iterate Method Implementations

In this thesis, we are concerned with starting and restarting problems (1.3) and (1.4) within an infeasible predictor-corrector interior-point method. As the previous section detailed, Gondzio's reoptimization technique is applied in the context of his multiple centrality corrections interior-point algorithm. However, we can find within an infeasible predictor-corrector interior-point algorithm a warm-start point that satisfies the criteria for a "good" warm-start point as outlined by Gondzio in [23].

To accomplish this modification, we add to the infeasible predictor-corrector interior-point method a procedure for finding a warm-start point satisfying Gondzio's

criteria. This procedure begins once the relative duality gap of the current iterate falls below a specified tolerance. The perturbation parameter $\bar{\mu}$ is fixed within the procedure, and Newton steps are iteratively taken until a warm-start point is attained. The length of each step is chosen in a line-search with backtracking framework. The merit function ϕ in the line-search algorithm is defined as

$$\phi(x^k, s^k, y^k, z^k, w^k) = 2n\frac{\min(Xz, Sw)}{(x^k)^T z^k + (s^k)^T w^k}$$

Once this warm-start point is found, we return to the infeasible predictor-corrector algorithm and proceed to optimality. Algorithmically, the method can be seen as follows:

Method 3.2 *(Centered Iterate Method 1)*

At each iteration of the infeasible interior-point algorithm, if $|c^T x^k - b^T y^k + u^T w|/(|c^T x^k| + 1) < \epsilon_g$ and a warm-start solution $(x_{ws}, s_{ws}, y_{ws}, z_{ws}, w_{ws})$ has not been saved, then set $\bar{\mu} = \bar{\sigma}((x^k)^T z^k + (s^k)^T w^k)/2n$, let $v^k = (x^k, s^k, y^k, z^k, w^k)$ and do

(1) If

$$||Ax^k - b||_2/(1 + ||x^k||_2) < \epsilon_p,$$

$$||A^T y^k + z^k - w^k - c||_2/(1 + ||y^k||_2 + ||z^k||_2 + ||w^k||_2) < \epsilon_d,$$

$$\beta_{min}\ \bar{\mu} \leq x_i z_i \leq \beta_{max}\ \bar{\mu},$$

and

$$\beta_{min}\ \bar{\mu} \leq s_i w_i \leq \beta_{max}\ \bar{\mu}$$

then set $(x_{ws}, s_{ws}, y_{ws}, z_{ws}, w_{ws}) = (x^k, s^k, y^k, z^k, w^k)$, save the warm start and go to Step 6

(2) Solve for the step Δv^k

$$F'(v^k)\Delta v^k = -F_\mu(v^k)$$

(3) Choose $\tau^k \in (0,1)$ and set $\alpha^k = \min(1, \tau^k \tilde{\alpha}^k)$ where

$$\tilde{\alpha}^k = \frac{-1}{\min((X^k)^{-1}\Delta x^k, (S^k)^{-1}\Delta s^k, (Z^k)^{-1}\Delta z^k, (W^k)^{-1}\Delta w^k)}$$

(4) Let $v^{k+1} = v^k + \alpha_k v^k$. While $\phi(v^{k+1}) < \phi(v^k)$ do

(4.1) $\alpha^k := \rho\alpha^k$ for $\rho \in [l, u]$

(4.2) Go to step 4.

(5) Go to Step 1

(6) Return to interior-point algorithm

As Gondzio suggests in [23] once the relative duality gap,

$$|c^T x^k - b^T y^k + u^T w|/(|c^T x^k| + 1),$$

falls below ϵ_g, $\bar{\mu}$ is set equal to a fraction of the average value of the pairwise complementarity products,

$$\bar{\sigma}((x^k)^T z^k + (s^k)^T w^k)/2n.$$

The perturbation parameter $\bar{\mu}$ stays at this value for subsequent iterations until the relative primal feasibility,

$$||Ax^k - b||_2/(1 + ||x^k||_2),$$

and relative dual feasibility,

$$||A^T y^k + z^k - w^k - c||_2/(1 + ||y^k||_2 + ||z^k||_2 + ||w^k||_2)$$

fall below ϵ_p and ϵ_d, respectively, and the centrality conditions

$$\beta_{min}\ \bar{\mu} \leq x_i z_i \leq \beta_{max}\ \bar{\mu}$$

and

$$\beta_{min}\ \bar{\mu} \leq s_i w_i \leq \beta_{max}\ \bar{\mu}$$

are satisfied.

Our numerical tests showed that the best setting for the parameters ϵ_g and $\bar{\sigma}$ is 1. The tests also show that the primal feasibility requirement should be not be very stringent, so ϵ_p is set to 1. Notice that for warm-start nodes in a branch and bound tree, dual feasibility is satisfied. As suggested by Gondzio, β_{min} and β_{max} are set to $\frac{1}{2}$ and 2, respectively.

In Step 3 of the algorithm, the step length α_k is chosen in each iteration by a ratio test (where τ^0 is as 0.9995) and then scaled so that the next iterate lies inside the neighborhood

$$\mathcal{N} = \{(x, y, z, s, w) : (x, z, s, w) > 0, \min(Xz, Sw) \geq \phi_0 \frac{x^T z + s^T w}{2n}\}$$

where ϕ_0 is chosen by default as 10^{-3}. The set $\mathcal{N}$ is a neighborhood of the central path, and points in $\mathcal{N}$ are guaranteed to be positive.

For the ratio test, we compute $\bar{\alpha}^k$ such that

$$(\bar{x}, \bar{s}, \bar{z}, \bar{w}) = (x^k, s^k, z^k, w^k) + \min(1, \bar{\alpha}^k)(\Delta x^k, \Delta s^k, \Delta z^k, \Delta w^k)$$

is either positive in the case $\bar{\alpha}^k > 1$ or on the boundary of the nonnegative orthant in the case that $\bar{\alpha}^k \leq 1$.

The step length is then scaled by ρ, if necessary, in a backtracking framework such that

$$(x^{k+1}, s^{k+1}, y^{k+1}, z^{k+1}, w^{k+1}) = (x^k, s^k, y^k, z^k, w^k) + \alpha^k(\Delta x^k, \Delta s^k, \Delta y^k, \Delta z^k, \Delta w^k)$$

is in $\mathcal{N}$ where $\alpha^k = \rho\bar{\alpha}^k$. The parameter ρ is decreased, if necessary, by a predetermined factor until

$$\phi(x^{k+1}, s^{k+1}, y^{k+1}, z^{k+1}, w^{k+1}) \leq \phi(x^k, s^k, y^k, z^k, w^k).$$

The next version of the centered iterate idea aims to examine the effect that a change in proximity measure has on the performance of the reoptimization technique. A *proximity measure* is a distance function that measures the distance between a feasible point and either the central path or a point on the central path. Recent work by Gonzàlez-Lima, El-Bakry, and Tapia [28] indicates that the choice of proximity measure can make a computational difference when approaching the central path. To test this idea in the context of reoptimization techniques, we change the central path proximity measure from the one suggested by Gondzio to the l_2 proximity measure

$$\left\| \frac{1}{\bar{\mu}} \begin{pmatrix} Xz \\ Sw \end{pmatrix} - e \right\|_2 .$$

We also change the associated merit function ϕ to

$$\phi(x, s, y, z, w) = \|F_\mu(x, s, y, z, w)\|_2^2.$$

The second centered iterate method can be seen algorithmically as follows:

Method 3.3 *(Centered Iterate Method 2)*

At each iteration of the infeasible interior-point algorithm, if $|c^T x^k - b^T y^k + u^T w|/(|c^T x^k| + 1) < \epsilon_g$ and a warm-start solution $(x_{ws}, s_{ws}, y_{ws}, z_{ws}, w_{ws})$ has not been saved, then set $\bar{\mu} = \bar{\sigma}((x^k)^T z^k + (s^k)^T w^k)/2n$, let $v^k = (x^k, s^k, y^k, z^k, w^k)$, and do

(1) If

$$||Ax^k - b||_2/(1 + ||x^k||_2) < \epsilon_p,$$

$$||A^T y^k + z^k - w^k - c||_2/(1 + ||y^k||_2 + ||z^k||_2 + ||w^k||_2) < \epsilon_d,$$

and

$$\left\| \frac{1}{\bar{\mu}} \begin{pmatrix} Xz \\ Sw \end{pmatrix} - e \right\|_2 \leq \beta$$

then set $(x_{ws}, s_{ws}, y_{ws}, z_{ws}, w_{ws}) = (x^k, s^k, y^k, z^k, w^k)$, save the warm start and go to Step 7

(2) Solve for the step Δv^k

$$F'(v^k)\Delta v^k = -F_\mu(v^k)$$

(3) Choose $\tau^k \in (0,1)$ and set $\alpha^k = \min(1, \tau^k \tilde{\alpha}^k)$ where

$$\tilde{\alpha}^k = \frac{-1}{\min((X^k)^{-1}\Delta x^k, (S^k)^{-1}\Delta s^k, (Z^k)^{-1}\Delta z^k, (W^k)^{-1}\Delta w^k)}$$

(4) Let $v^{k+1} = v^k + \alpha_k v^k$. While $\phi(v^{k+1}) > \phi(v^k) + \gamma\alpha_k \nabla\phi(v^k)^T \Delta v^k$ do

(4.1) $\alpha^k := \rho\alpha^k$ for $\rho \in [l, u]$

(4.2) Go to step 4.

(5) Go to Step 1

(6) Return to interior-point algorithm

As before, in Step 3 of this algorithm the step length α_k is chosen in each iteration by a ratio test. In Step 4 this α_k is then scaled, if necessary, so that the next iterate is deemed "acceptable" by our merit function ϕ.

The step length is scaled by ρ in a backtracking framework such that

$$\phi(v^{k+1}) \leq \phi(v^k) + \gamma\alpha_k \nabla\phi(v^k)^T \Delta v^k$$

where $v^k = (x^k, s^k, y^k, z^k, w^k)$ and $\Delta v^k = (\Delta x^k, \Delta s^k, \Delta y^k, \Delta z^k, \Delta w^k)$.

The parameter settings for ϵ_g, $\bar{\sigma}$, and ϵ_p are the same as in the Centered Iterate Method 1. The parameter β is set to $\frac{1}{2}$.

In the next section we we show how convergence to a "well-centered" warm start can be proven for the Centered Iterate Method 2.

3.1.3 Convergence Theory for Centered Iterate Method 2

In this section, we show that the inner loop defined by the Centered Iterate 2 Method terminates with a point $(x_{ws}, s_{ws}, y_{ws}, z_{ws}, w_{ws})$ such that

$$\left\| \frac{1}{\bar{\mu}} \begin{pmatrix} X_{ws} z_{ws} \\ S_{ws} w_{ws} \end{pmatrix} - e \right\|_2 \leq \beta$$

for fixed $\bar{\mu}$.

First, recall the perturbed KKT conditions for the problems we are considering

$$F_\mu(x, s, y, z, w) = \begin{pmatrix} Ax - b \\ x + s - u \\ A^T y + z - w - c \\ XZe - \mu e \\ SWe - \mu e \end{pmatrix} = 0.$$

We would like to find a point that is nearly on the central path parameterized by μ of the feasible region of the problem that we are solving. That is, we would like a point that is near an $(\bar{x}, \bar{s}, \bar{y}, \bar{z}, \bar{w})$ such that $F_\mu(\bar{x}, \bar{s}, \bar{y}, \bar{z}, \bar{w}) = 0$ for fixed $\bar{\mu}$. As is standard in nonlinear optimization, we transform the problem of solving the system of equations $F_\mu(x, s, y, z, w) = 0$ into the problem of minimizing the function $\|F_\mu(x, s, y, z, w)\|_2^2$ and then apply a globalization strategy within Newton's Method. The globalization strategy that we use is the method of line searches with backtracking. The merit function in the line search with backtracking strategy outlined in Step 4 of the Centered Iterate Method 2 is defined as

$$\phi(x, s, y, z, w) = \|F_\mu(x, s, y, z, w)\|_2^2,$$

and as the following lemma shows the steps taken in the algorithm are a descent direction for this function.

Lemma 3.1 Consider $\mu > 0$ fixed, then $(\Delta x^k, \Delta s^k, \Delta y^k, \Delta z^k, \Delta w^k)^T$ generated by the Centered Iterate Method 2 is a decent direction for the merit function

$$\phi(x, s, y, z, w) = \|F_\mu(x, s, y, z, w)\|_2^2.$$

Proof. Let $v = (x, s, y, z, w)^T$ and $\Delta v^k = (\Delta x^k, \Delta s^k, \Delta y^k, \Delta z^k, \Delta w^k)^T$, then

$$\begin{aligned} \nabla\phi(v)^T(\Delta v^k)^T = & -2(F'_\mu(v)F_\mu(v))^T((F'_\mu(v))^{-1}F_\mu(v)) \\ & = -2(F_\mu(v))^T F_\mu(v) \\ & = -2\|F_\mu(v)\|_2^2 < 0 \end{aligned}$$

and the property follows. □

Next, we need to show that the norm of the steps taken in the algorithm

$$\|(\Delta x^k, \Delta s^k, \Delta y^k, \Delta z^k, \Delta w^k)\|_2$$

are bounded.

Lemma 3.2 Consider $\mu > 0$ fixed and let $\{(x^k, s^k, y^k, z^k, w^k)\}$ be generated by the Centered Iterate Method 2. Assume

(a1) $\frac{\min(\tilde{X}^k\tilde{Z}^k e)}{((\tilde{x}^k)^T\tilde{z}^k)} \geq \frac{\gamma}{2n}$ where $\tilde{x} = (x, s)$ and $\tilde{z} = (z, w)$ for all k and some $\gamma \in (0, 1)$,

(a2) $\{(x^k, s^k, z^k, w^k)\}$ is bounded.

(a3) A has full rank.

Then,

$$\|(\Delta x^k, \Delta s^k, \Delta y^k, \Delta z^k, \Delta w^k)\|_2 \leq C < +\infty$$

for all $k \geq 0$ and some C.

Proof. Consider

$$F'_{\mu}(x,s,y,z,w) = \begin{pmatrix} A & 0 & 0 & 0 & 0 \\ X & S & 0 & 0 & 0 \\ 0 & 0 & A^T & I & I \\ Z & 0 & 0 & X & 0 \\ 0 & W & 0 & 0 & S \end{pmatrix}.$$

which is bounded on $\{(x^k, s^k, z^k, w^k)\}$ since every submatrix is bounded. This along with assumptions **(a1)**, **(a2)**, and **(a3)** implies that $(F'_{\mu}(x,s,y,z,w))^{-1}$ exists and is bounded on the sequence.

Let $v^k = (x^k, s^k, y^k, z^k, w^k)^T$ and $\Delta v^k = (\Delta x^k, \Delta s^k, \Delta y^k, \Delta z^k, \Delta w^k)^T$, then,

$$\begin{aligned} \|\Delta v^k\|_2 &= \| - (F'_{\mu}(v^k))^{-1} F_{\mu}(v^k)\|_2 \\ &\leq \|(F'_{\mu}(v^k))^{-1}\|_2 \|F_{\mu}(v^k)\|_2 \\ &\leq C < +\infty \end{aligned}$$

for all $k \geq 0$ and some C $\square$.

Finally, we can show that if the sequence generated by the inner loop of the algorithm converges, then it converges to a point $(x^k_{\mu}, s^k_{\mu}, y^k_{\mu}, z^k_{\mu}, w^k_{\mu}) \in \mathcal{F}$ and

$$\left\| \begin{pmatrix} X^k_{\mu} Z^k_{\mu} e \\ S^k_{\mu} W^k_{\bar{\mu}} e \end{pmatrix} - \mu e \right\|_2 \to 0$$

Theorem 3.4 Consider $\bar{\mu} > 0$ fixed and let $\{(x^k, s^k, y^k, z^k, w^k)\}$ be generated by the Centered Iterate Method 2 and $\tau \in (0,1)$. Then

$$F_{\mu}(x,s,y,z,w) \to 0$$

Proof. Let $v^k = (x^k, s^k, y^k, z^k, w^k)^T$. It is known (see Dennis and Schnabel [15]) that the backtracking line-search strategy used in the Centered Iterate Method 2 produces

$$\frac{\nabla\phi(v^k)\Delta v^k}{\|\Delta v^k\|_2} \to 0$$

Since

$$\nabla\phi(v^k)^T(\Delta v^k)^T = -2\|F_\mu(v^k)\|_2^2$$

and

$$\|\Delta v^k\|_2 \leq C < +\infty,$$

we have

$$\|F_\mu(v^k)\|_2^2 \to 0.$$

From the definition of $F_\mu(v^k)$,

$$\left\| \begin{pmatrix} X^k Z^k e \\ S^k W^k e \end{pmatrix} - \mu e \right\|_2 \to 0,$$

and the result follows.□

Corollary 3.1 The Centered Iterate Method 2 terminates with a warm-start point $(x_{ws}, s_{ws}, y_{ws}, z_{ws}, w_{ws})$ such that

$$\left\| \frac{1}{\bar{\mu}} \begin{pmatrix} X_{ws} z_{ws} \\ S_{ws} w_{ws} \end{pmatrix} - e \right\|_2 \leq \beta$$

for fixed $\bar{\mu}$.

3.2 Early Termination Method

We also implement a version of the early termination method described in Section 2.4 of this thesis. The early termination method attempts to find a good warm-start

point by ending early the interior-point optimization of the current relaxation at a branch-and-bound node. Mitchell [49] points out that the benefits of this are twofold. Clearly, fewer iterations of the interior-point algorithm are required at each node in the branch-and-bound tree, and secondly it is hoped that an iterate not on the optimal face will be more well-centered.

The drawback of this method is that an interior-point optimization could be terminated too early. That is, the branch-and-bound algorithm may branch on a parent problem that could have been pruned had the interior-point optimization been taken to completion. There are three cases where branching may be too early. The first is when the optimal solution to the relaxation is integral, the second is when the objective function value of the optimal solution to the relaxation is worse than the best integer incumbent, and the last is when the relaxation is infeasible. Mitchell [49] describes safeguards to avoid terminating when the optimal solution is integral and when the optimal solution is worse than the best bound on the integral optimal solution. The early termination method with these safeguards was implemented by Borchers [6]. For the problems that he tested, the number of iterations required for a reoptimization was one-third to one-half fewer than the number of iterations from a cold start.

To avoid branching on a node whose optimal solution is integral, a heuristic is described for detecting fractional variables before the completion of optimization. Tapia's indicators (see El-Bakry, Tapia and Zhang [13]) are used to detect whether or not a binary variable x_i and its complement $1 - x_i$ are tending toward a nonzero value. It is then assumed that the variable x_i at optimality is not 1 or 0.

Recall that the optimal partition is defined by two index sets $\mathcal{B}$ and $\mathcal{N}$ such that for any optimal solution $(x^*, s^*, y^*, z^*, w^*)$ to a given linear program, if we let

$\hat{x}^* = (x^*, s^*)$ and $\hat{z}^* = (z^*, w^*)$, then

$$\mathcal{B} = \{i \in 1, ..., 2n |\ \hat{x}_i^* \neq 0\}$$

and

$$\mathcal{N} = \{i \in 1, ..., 2n |\ \hat{z}_i^* \neq 0\}.$$

Tapia's primal indicators are

$$T_p^x = \frac{x_i^{k+1}}{x_i^k} \quad \text{and} \quad T_p^s = \frac{s_i^{k+1}}{s_i^k},$$

and

$$\lim_{k\to\infty} \frac{x_i^{k+1}}{x_i^k} = \begin{cases} 1 & \text{if} \quad i \in \mathcal{N} \\ 0 & \text{if} \quad i \in \mathcal{B} \end{cases}$$

$$\lim_{k\to\infty} \frac{s_i^{k+1}}{s_i^k} = \begin{cases} 1 & \text{if} \quad i \in \mathcal{N} \\ 0 & \text{if} \quad i \in \mathcal{B} \end{cases}$$

The dual Tapia indicators are

$$T_d^w = \frac{w_i^{k+1}}{w_i^k} \quad \text{and} \quad T_d^z = \frac{z_i^{k+1}}{z_i^k},$$

and

$$\lim_{k\to\infty} (1 - \frac{z_i^{k+1}}{z_i^k}) = \begin{cases} 1 & \text{if} \quad i \in \mathcal{N} \\ 0 & \text{if} \quad i \in \mathcal{B} \end{cases}$$

$$\lim_{k\to\infty} (1 - \frac{w_i^{k+1}}{w_i^k}) = \begin{cases} 1 & \text{if} \quad i \in \mathcal{N} \\ 0 & \text{if} \quad i \in \mathcal{B} \end{cases}$$

Borchers combines the primal Tapia indicator with its corresponding dual indicator. The combination of Tapia's primal indicators for determining that a variable is tending towards zero is

$$T_0 = \frac{x_i^{k+1}}{x_i^k} + |1 - \frac{z_i^{k+1}}{z_i^k}|.$$

Likewise the combination for determining that a variable is converging to a nonzero value is

$$T_1 = \frac{s_i^{k+1}}{s_i^k} + |1 - \frac{w_i^{k+1}}{w_i^k}|.$$

If a binary variable x_i is fractional at optimality, then the slack on the upper bound constraint for x_i, namely s_i, is also fractional. Since the solution satisfies strict complementarity, we also know that the corresponding dual variables, z_i and w_i, must be zero at optimality. Thus if the binary variable x_i is fractional then T_0 and T_1 will both approach two. However, if x_i is not fractional, then T_0 and T_1 should approach zero. Once both T_0 and T_1 are at least 0.9, it is heuristically assumed that they will not approach zero.

To decrease the chance of branching on a node that could be fathomed by bounds, Borchers [6] suggests not terminating the relaxation until the duality gap is within a specified range. Also, so that infeasible nodes are not branched, the relaxation is not terminated until the primal feasibility is below a set tolerance.

Finding a warm-start point for the early termination method can be seen algorithmically as follows:

Method 3.5 *(Early Termination Method)*

At each iteration of the infeasible interior-point algorithm do

(1) If

$$0.0001 \leq x_i \leq 0.9999 \quad \text{for all} \quad i$$

$$T_0 \geq 0.9, \quad \text{and} \quad T_1 \geq 0.9,$$

$$||c - A^T y + w - z||_2/(1 + ||y||_2 + ||w||_2 + ||z||_2) < 10^{-8},$$

$$||b - Ax||_2/(1 + ||x||_2) < 0.10,$$

and

$$|c^T x - (b^T y - u^T w)|/(1 + |b^T y + u^T w|) < 0.05,$$

then save the point (x, s, y, z, w) as a warm start, and terminate the algorithm.

(2) Else return to algorithm.

In our tests, we follow Borchers [6] implementation for finding warm starting points, but we do not terminate early.

3.3 Total Relative Error Method

Our final method, is the total relative error method. This method accepts an iterate as a warm-start point during a parent optimization if the total relative error is less than a specified tolerance. The total relative error is defined as

$$\frac{||Ax - b||_2}{max(1, ||b||_2)} + \frac{||A^T y + z - w||_2}{max(1, ||c||_2)} + \frac{||x + s - u||_2}{max(1, ||u||_2)} + \frac{|c^T x - b^T y + u^T w|}{max(1, |c^T x|, |b^T y - u^T w|)}.$$

When the total relative error is less than 10^{-1}, a warm-start solution for subsequent problems is saved. The motivation behind this method is twofold. First, since the total relative error must be below the stopping tolerance for convergence of the interior-point algorithm used in this work, finding a warm-start point for feasible problems is guaranteed. Secondly, we hope that the total relative error measure characterizes a solution to the parent problem well, and the relatively loose tolerance required for saving the warm-start point will allow for good convergence to optimality for child problems.

Chapter 4

0/1 Mixed Integer Programming Problem Set

4.1 MIPLIB

The majority of the problem instances tested are a part of MIPLIB [5], an electronically available library of both pure and mixed integer programs. MIPLIB is a standard test set for comparing the performance of mixed integer optimization codes. The problem statistics for the MIPLIB problems can be found in the appendix.

4.2 Capacitated Facility Location

The second set of problems is a set of four capacitated facility location problems available from the OR-Library [4]. These problems were also examined in Borchers' thesis [6] to test the interior-point branch-and-bound code he implemented. The problem statistics for the capacitated facility location problems can be found in the appendix.

4.3 Conrail

The instance of MIP that motivated this study comes from Consolidated Rail Corporation (Conrail) The model uses actual data from 1995 and attempts to choose the departure times for 998 trains over 559 yards such that the trains depart at some point during the correct day, and the number of locomotives needed to power the schedule is minimized. The trains are of two types, regular and exchange. Regular trains contain cars, as well as, locomotives, and exchange trains contain only locomotives. The trains run according to a weekly schedule, and there is an adjustable

parameter that controls the number of departure time periods each day. When this parameter is set to three the trains leave every eight hours, when set to twenty-four they leave hourly, and when set to the maximum, ninety-six, trains leave every fifteen minutes. There is a time period parameter for each of the two types of trains in the model. For regular trains the time period parameter is called NumTP. For exchange trains the parameter is called NumEnsTP, and must divide NumTP evenly. The problem statistics for the Conrail problems can be found in the appendix.

The objective is to minimize the total number of locomotives used by the schedule. The possible moves that a train can make on any given day and hour are modeled on a time-space directed graph. The network consists of nodes, $InvNode$, and three types of arcs, X (for regular trains), $XEns$ (for exchange trains), and I (for inventory at each yard). The nodes are indexed on yards, days, and hours. Each arc measures the number of locomotives used to get a train from its origination yard on a particular day at a particular time to its destination yard at the appropriate arrival day and time.

Given that a train t can travel on day d, at hour h, from some yard y, then an arc $X_{t,d,h}$ or $XEns_{t,d,h}$ is placed between node $InvNode_{y,d,h}$ and the node corresponding to the destination yard for train t from yard y, on the estimated arrival day, and at the estimated arrival time. Each arc $X_{t,d,h}$ has an upper bound $MaxPower$. This constant upper bound is the maximum number of locomotives that a regular train can contain. The arcs $I_{y,d,h}$ connect, for a given yard, the previous day and/or hour to the next day and/or hour. The flow on the $I_{y,d,h}$ arcs tells how many locomotives remain in inventory from one time period to another at a yard.

For each yard, the node corresponding to the last hour of the last day of the week is connected by an inventory arc to the first hour of the first day of the week. This forces the weekly time line generated by a solution to be a cycle in the network.

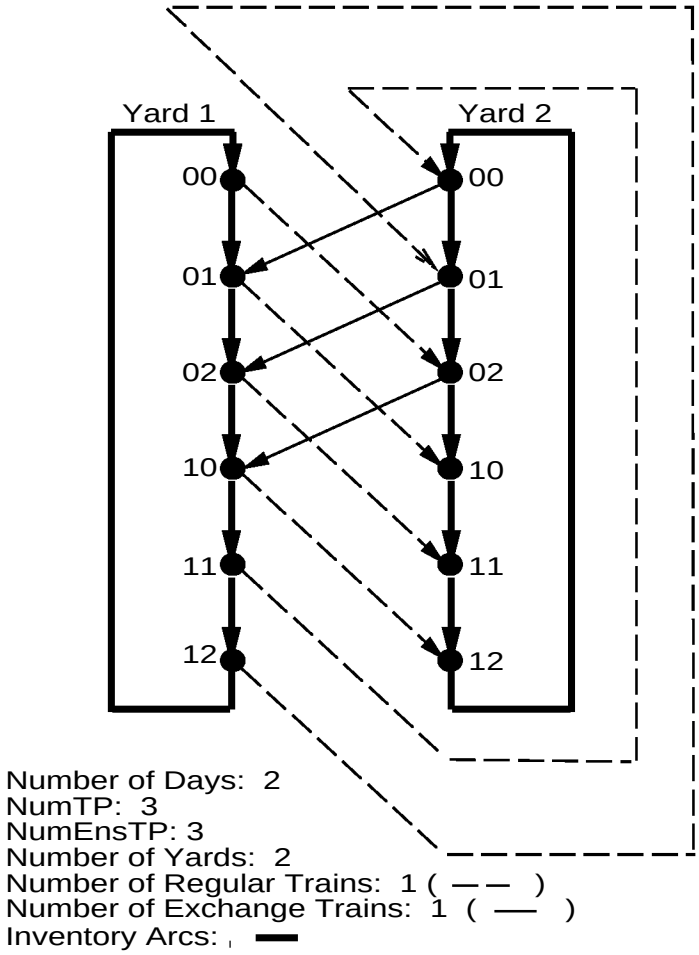

Figure 4.1 Time-Space Network.

In Figure 2.1, we have an example of the time-space directed graph. For ease of exposition, we assume that a week has only two days. Also, assume that we have one regular train that runs daily and one exchange train that runs only on the first day of the "week." The trains originate at one of the two yards, and arrive at the other. There are three time periods each day regular as well as the exchange train. To arrive at its destination yard from its origination yard, the regular train takes two time periods, and the exchange train takes one time period. Also notice, that when a train starts at the end of the week and the day and time needed to finish in that week is exceeded, it ends at the beginning of the week, during the appropriate time period, at its destination yard. That is, Sunday follows Saturday in the time line.

A regular train can travel from its origination to its destination only once during any day, so in addition to the variables corresponding to the arcs in the network, there are also decision variables. The variable $z_{t,h}$ is 1 if a regular train t is used during time period h, and 0 otherwise. Any regular train t is also required to have some minimum number of locomotives, $xPowerNeeds_{td}$, on each day d.

In order to count the number of locomotives used to power the schedule, the objective function tallies the total number of locomotives during a certain time period. Any time period will do, but the model uses time period zero ($h = 0$).

The complete model can be described as follows. The objective is to minimize the number of locomotives

$$\sum_{(t,d,0)\in A} X_{t,d,h} + \sum_{(t,d,0)\in B} XEns_{t,d,h} + \sum_{(y,d,0)\in C} I_{y,d,h},$$

where A and B are the sets of arcs for regular trains and exchange trains, respectively, and C is the set of inventory arcs. We are constrained by flow conservation constraints over the network as well as capacities of size $MaxPower$ on each arc $X_{t,d,h}$.

Each regular train is restricted to run in only one time period,

$$\sum_{h \in H} z_{th} = 1,$$

where H is the set of time periods. Also, for each regular train t we do not allow $X_{t,d,h}$ to hold positive values when z_{th} is zero

$$\sum_{(t,d,h) \in A} X_{t,d,h} \leq |A| * MaxPower * z_{t,h}.$$

Finally, we are constrained by the need for a minimum number of locomotives, $xPowerNeeds_{td}$, to power each regular train on a given day

$$\sum_{(t,d,h) \in A} X_{t,d,h} \geq xPowerNeeds_{td}.$$

4.3.1 Preliminary Numerical Tests

There are 41 problems from the Conrail test set of sizes ranging from 17,442 variables and 6118 constraints to 538,800 variables and 144,595 constraints. Due to the size and the computational effort expended, we report preliminary results for the first 21 problems.

The Conrail problems are instances of integer programming problems where each individual linear program generated in a branch-and-bound tree solves faster using an interior-point algorithm than a simplex-based algorithm when no reoptimization techniques are used for either method. Table 4.3.1 shows the running time in seconds of the root nodes generated in a branch-and-bound tree for the Conrail set. In our numerical tests, we used CPLEX 4.0 on one of four SGI Power Challenge-L multiprocessors. Each multiprocessor consists of four 75 megahertz MIPS 8000 processors, 256 megabytes of memory, 2 gigabytes of SCSI disk, and an 800 megabit HIPPI interface.

Name	Interior Point(Crossover)	Dual Simplex	Primal Simplex
con33	42.60 (13.76)	420.15	388.15
con42	44.80 (16.40)	527.47	628.50
con44	121.51 (20.04)	770.30	809.85
con62	89.78 (32.10)	955.50	1140.19
con63	121.96 (33.57)	1263.39	1307.66
con66	255.30 (42.88)	2443.39	2131.02
con82	129.94 (49.02)	1627.76	1987.94
con84	325.99 (58.22)	2288.16	2746.39
con88	837.71 (80.85)	4314.84	4364.02
con122	327.51 (93.15)	3731.88	4982.25
con123	527.19 (93.47)	4694.94	5602.27
con124	677.74 (104.44)	4937.27	6398.69
con126	1046.65 (128.67)	7429.42	7802.71
con1212	1478.48 (167.30)	n/a	n/a
con242	1478.48 (147.77)	n/a	n/a
con243	1783.73 (92.03)	n/a	n/a
con244	3479.76 (283.95)	n/a	n/a
con246	5931.44 (329.54)	n/a	n/a
con248	9887.12 (n/a)	n/a	n/a
con2412	n/a (n/a)	n/a	n/a
con2424	n/a (n/a)	n/a	n/a

Table 4.1 Solution times (secs) for branch-and-bound search tree root node for CPLEX's interior-point, dual-simplex, and primal-simplex codes where **n/a** indicates that a solution was not found within the time limit given.

4.3.2 Integer Feasible Solution Heuristics

The computational effort expended to solve one node of the branch-and-bound tree generated by the Conrail model can be extensive. For example, more than 10,000 seconds are required to solve the root node only for problems *con2412* and *con2424*. A large branch-and-bound search tree for these problems could be prohibitive. The size of a branch-and-bound search tree is reduced by the presence of appropriate upper and lower bounds on the optimal solution. In order to find good upper bounds, heuristic algorithms are used to generate integer feasible solutions whose objective function value provides an upper bound on an optimal solution's objective function value. The heuristics implemented are based upon fixing the variables z_{th} in the Conrail model so that some subset of the cover constraints

$$\sum_{h \in H} z_{th} = 1.$$

are satisfied.

The first heuristic is a simple rounding heuristic. When the linear programming relaxation is solved using the primal-dual interior-point codes (either CPLEX or LIPSOL), with the exception of the fixed variables, nearly every z_{th} is within a very small amount of 1/NumTP. With a crossover solution from CPLEX, however, we found that more of a distinction could be made between the $|H|$ variables in each row, and our rounding heuristic has greater success. Thus, in our implementation we are given a basic feasible solution x as input to our rounding heuristic. S_R is the current feasible region with the integrality restrictions relaxed, and ϵ is the integrality tolerance (10^{-7}). We saw the most favorable results when we let RND_TOL equal 0.9.

Algorithm 4.1 *LP-Based Rounding Heuristic*

Given x such that $x \in S_R$ and $\epsilon < x_j < 1 - \epsilon$ for some j

While $S_R \neq \emptyset$ do:

(1) If $x_j \geq RND\_TOL$, then set x_j equal to 1.

(2) Solve the resulting LP for x.

(3) If $\epsilon < x_j < 1 - \epsilon$ for some j, then let

$$RND\_TOL = \min(RND\_TOL - 0.1,\ \max_j x_j)$$

Since the linear programming relaxations of the Conrail problems require lengthy computation times to solve, a rounding heuristic that is not based on linear programming relaxations was also implemented. A great deal of the underlying structure of the Conrail problems consists of a network. Since solving the embedded network linear program is a considerably easier task than solving the original LP, the second heuristic takes advantage of this structure in order to find an integer feasible solution.

The cover constraints

$$\sum_{h \in H} z_{th} = 1,$$

enforce the condition that each regular train run only at one specific time period. Notice that if we are given to which time period each train is assigned (any such assignment will do, not necessarily an optimal one), then we can quite readily compute the number of locomotives necessary to power such a schedule. The problem reduces to an upper-bounded transshipment problem (or a minimum cost network flow problem) (see Chvátal [8]), and any fixing produces an integer feasible solution.

Algorithm 4.2 *Network-Based Rounding Heuristic*

(1) For each train t, randomly select an hour h for t to run. Set $z_{t,h}$ equal to one. For all other possible hours $\hat{h}$ for train t, set $z_{t,\hat{h}}$ equal to zero.

(2) Solve the resulting LP for x.

The results for these rounding heuristics are in table 4.3.2. The first column is the optimal objective function value of the linear programming relaxation of the problems. The second column contains the best integer feasible solutions returned from CPLEX 4.0, the third column contains the results from the lp-based rounding heuristic. The fourth contains the best integer feasible solution from the network based rounding heuristic. The table indicates that the Network-Based Rounding Heuristic provides better bounds on the optimal solution value than either CPLEX or the linear programming based rounding heuristic.

Name	LP SOLN	CPLEX	LP_RND	NET_RND
con33	630.69047619	715	705	711
con42	617.96071429	722	726	726
con44	605.14285714	707	706	695
con62	600.36904762	712	731	708
con63	591.57142857	704	727	708
con66	580.54761905	702	699	692
con82	586.84523810	717	704	716
con84	575.57467532	708	723	696
con88	568.58333333	703	722	683
con122	574.84523810	728	716	716
con123	569.18650794	713	701	708
con124	563.369048	696	714	696
con126	558.88095238	712	716	695
con1212	554.36309524	n/a	719	688
con242	564.91655349	773	718	717
con243	558.91321081	728	714	719
con244	553.11784297	713	724	695
con246	549.35912698	n/a	715	689
con248	546.39812091	n/a	n/a	691
con2412	n/a	n/a	n/a	689
con2424	n/a	n/a	n/a	688

Table 4.2 Results of rounding heuristics for Conrail problems where **n/a** indicates that a solution was not found within the time limit given.

Chapter 5

Factors Affecting Reoptimization

In theory, the global convergence of an infeasible primal-dual interior-point algorithm is guaranteed from starting points for which the non-negative variables are strictly greater than zero. The algorithm restricts the iterates to a subset of points for which the primal and dual infeasibilities are bounded by some multiple of the duality measure and the pairwise complementarity products are balanced. The algorithm ensures that the primal and dual infeasibilities decrease with each iteration and keeps the Newton-like search directions from causing the algorithm to stall near a boundary.

Although convergence is assured theoretically from any starting point, computational experience shows that a good starting point should be well-centered (i.e., the difference between the maximum and the minimum of the complementarity products should be small), and the ratio of infeasibility to duality measure should not be too large. A popular heuristic used to cold start the infeasible primal-dual algorithm attempts to find a point that satisfies these two conditions (see Mehrotra [47] and Wright [62]).

It is commonly held that although a point designed for use as a good warm start may likely be more well-centered and have a smaller ratio of infeasibility to duality measure than a cold-start point, a factor of two or three savings in iteration count and runtime is all that can be expected. This behavior is due to the difference between the optimal partition of the original problem (from whence the warm-start point came) and the optimal partition of the perturbed problem that we are attempting to warm start.

In the following sections, we address some of the issues thought to affect the convergence of the iteration sequence generated by interior-point methods during reoptimization.

5.1 Infeasibility

Suppose that we are given the following linear-programming relaxation as the problem defined at some node of the branch-and-bound tree for a 0/1 integer programming problem:

$$\begin{aligned} \min \quad & c^T x + c_k x_k \\ \text{st} \quad & Ax + a_k x_k = b \\ & x + s = e \\ & x_k + s_k = 1 \\ & x, x_k \geq 0 \\ & s \geq 0 \end{aligned} \tag{5.1}$$

where x, u, $c, s \in \mathbf{R}^n$, a_k and $b \in \mathbf{R}^m$, $A \in \mathbf{R}^{m \times n}$, x_k, s_k, and $c_k \in \mathbf{R}$, and e is the vector of all ones of length n.

The dual problem is defined as follows

$$\begin{aligned} \max \quad & b^T y - e^T w - w_k \\ \text{st} \quad & A^T y + z - w = c \\ & a_k^T y + z_k - w_k = c_k \\ & z, z_k \geq 0 \\ & w, w_k \geq 0 \end{aligned} \tag{5.2}$$

where $y \in \mathbf{R}^m$, $z, w \in \mathbf{R}^n$, and $z_k, w_k \in \mathbf{R}$.

Also, suppose variable x_k is fractional in a optimal solution to the parent node problems (5.1) and (5.2). Fixing this variable to one of its bounds (0 or 1) yields the following pair of problems for the child node

$$\begin{aligned} \min \quad & c^T x \\ \text{st} \quad & Ax = \tilde{b} \\ & x + s = e \\ & x \geq 0 \\ & s \geq 0 \end{aligned} \tag{5.3}$$

where $\tilde{b} = b$ if x_k is fixed at 0 and $\tilde{b} = b - a_k$ if x_k is fixed at 1, and

$$\begin{aligned} \max \quad & \tilde{b}^T y - e^T w \\ \text{st} \quad & A^T y + z - w = c \\ & z \geq 0 \\ & w \geq 0 \end{aligned} \tag{5.4}$$

Clearly, fixing x_k to either of its bounds introduces infeasibility in the primal problem (5.3) defined for the child node, but no infeasibility in the dual problem (5.4) defined for the child node. Therefore, it is reasonable to suppose that this primal infeasibility introduced each time a child node is created in a branch-and-bound tree has some bearing on the efficiency of a reoptimization in an interior-point algorithm that uses information from the optimization of the parent node problem. However, since the interior-point method that we are interested in is an infeasible interior-point

method, primal infeasibility alone does not prohibit us from using information from the parent optimization to reoptimize a child problem.

The infeasible primal-dual interior-point algorithm used in this work restricts iterates to a central path neighborhood that includes infeasible points. This neighborhood is defined by

$$\begin{aligned}\mathcal{N}_{-\infty}(\gamma,\beta) = \quad & \{(x,s,y,z,w) : \|(r_b,r_c)\| \leq [\|(r_b^0,r_c^0)\|/\mu^0]\beta\mu, \\ & (x,s,z,w) > 0, \min(Xz,Sw) \geq \gamma\mu\}\end{aligned} \tag{5.5}$$

where $\gamma \in (0,1)$ and $\beta \geq 1$, $\mu = (x^T z + s^T w)/2n$, $r_b = Ax - b$, and $r_c = A^T y + z - w - c$. The values of μ^0, r_b^0 and r_c^0 are evaluated at the starting point $(x^0, s^0, y^0, z^0, w^0)$. The first inequality in (5.5) is known as the Feasibility Priority Principle (see Zhang [63]). For all points in $\mathcal{N}_{-\infty}(\gamma,\beta)$ the infeasibility in the primal and dual constraints is uniformly bounded by some multiple of the duality measure μ. By forcing μ_k to zero monotonically and forcing all iterates to satisfy

$$\|(r_b^k, r_c^k)\| \leq [\|(r_b^0, r_c^0)\|/\mu^0]\beta\mu^k,$$

the infeasible primal-dual interior-point method ensures that $r_b^k \to 0$ and $r_c^k \to 0$ as $k \to \infty$. This inequality also ensures that infeasibility decreases at least as fast as complementarity.

Theoretical analysis (see Wright [62]) and computational experience tells us that, although we have considerable flexibility in choosing a starting point for the infeasible primal-dual interior-point method, the starting point should not be too infeasible. That is, the ratio of infeasibility to duality measure of $(x^0, s^0, y^0, z^0, w^0)$ should not be too large. Because of this, it is reasonable to expect that a warm-started optimization should experience a reduction in iteration count of one-third to one-half that of a cold-started optimization.

The next section will address the second inequality in the definition of $\mathcal{N}_{-\infty}(\gamma, \beta)$ (5.5).

5.2 Centrality and Boundary Behavior

In Chapter 1, we defined the central path as the the arc of strictly feasible points parameterized by μ such that the pairwise products $x_i z_i$ and $s_i w_i$ are identical for all $i \in \{1, ..., n\}$. Primal-dual interior-point algorithms keep iterates in a general vicinity of the central path. This is done to prevents the Newton-like search directions from being distorted by components of (x, s, z, w) that approach the boundary of the positive orthant prematurely. The second inequality in the definition of $\mathcal{N}_{-\infty}(\gamma, \beta)$,

$$\min(Xz, Sw) \geq \gamma\mu, \tag{5.6}$$

where $\gamma \in (0, 1)$ and $\beta \geq 1$, $\mu = (x^T z + s^T w)/2n$ addresses this concern. Iterates that satisfy (5.6) have pairwise products that decrease at a controlled rate. That is, no pairwise product can converge to zero too much faster than the others.

For Newton's Method, we know that a variable that is positive at optimality becoming zero at some iteration can have a devastating effect on the optimization. To see this, recall the KKT conditions for optimality of a feasible point

$$F(x, s, y, z, w) = \begin{pmatrix} Ax - b \\ x + s - u \\ A^T y + z - w - c \\ XZe \\ SWe \end{pmatrix} = 0$$

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

where $X = diag(x)$, $Z = diag(z)$, $S = diag(s)$, $W = diag(w)$ and e is the n-vector of all ones. Suppose that s_i is positive at optimality, and $s_i^k = 0$ and $w_i^k > 0$ at the

current iteration k. The Newton equation for $SW = e$ in the KKT conditions is

$$S\Delta w + W\Delta s = SWe. \tag{5.7}$$

Examining the i-th component of (5.7) at iteration k yields

$$s_i^k \Delta w_i^k + w_i^k \Delta s_i^k = s_i^k w_i^k = 0$$

which implies that $\Delta s_i^k = 0$. Thus, s_i^k will remain at zero for all subsequent iterations. For primal-dual interior-point methods this type of behavior translates into an iteration sequence that can be severely slowed because short steps must be taken when a component of a nonnegative variable is too close to the boundary.

5.3 Distance to Optimal

Along with being well-centered and not "too" infeasible, the distance a warm-start point is from an optimal solution is another factor that should effect reoptimization. Primal-dual interior-point methods are based on Newton's method, and for Newton's method, under standard assumptions, convergence is assured if the starting point is within a specific neighborhood of the optimal solution (see Dennis and Schnabel [15]). Given this, if a warm-starting solution is close enough to an optimal solution, then we would expect a savings in iteration count over starting from a cold solution.

We would also like to have some measure of how long it should take for the interior-point method to converge from a point that is a certain distance from the optimal solution set. A comparison can be made between the number of iterations required from each iterate in a cold-started optimization to optimality and the total number of iterations required for a warm-started optimization. If during a cold-started optimization an iterate is distance d from optimality, and the warm start used to reoptimize this problem is also distance d from optimality, it is reasonable to

expect that the numbers of iterations required for each of these scenarios should be comparable. Figure 5.3 illustrates the two distances with which we are concerned.

.

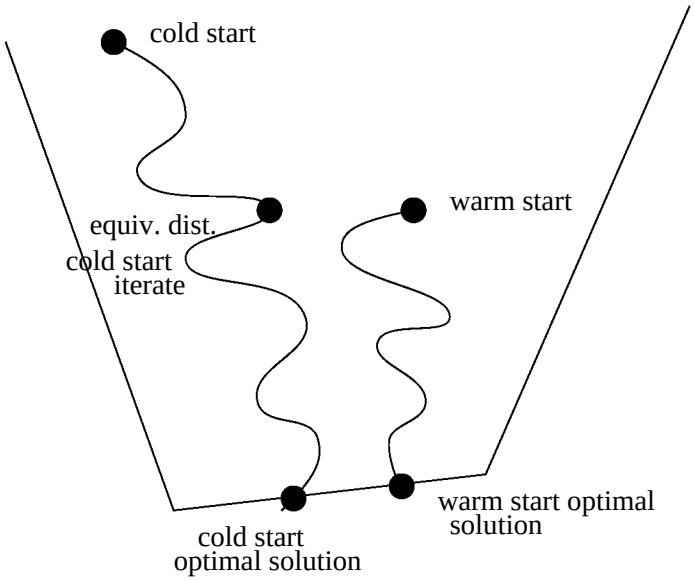

Figure 5.1 This figure demonstrates the computation of the expected iteration count.

5.4 Change in Optimal Partition

When the data associated with a linear program is changed, the new optimal solution set may have a different optimal partition than the original problem. Such is the case when a variable in the parent node problem in a branch-and-bound tree is modified to create the child node, and the solution to the parent node is used to reoptimize the child node. The indices that switch between $\mathcal{B}$ and $\mathcal{N}$ cause the central path to change. This change could mean that the interior-point iterates of the changed problem must follow a very dissimilar path from the iterates of the original problem. If the given warm start is far from the newly created central path, then the primal-dual

interior-point algorithm may take several iterations to adjust to this central path, since the algorithm must first balance the new pairwise complementarity products before proceeding to optimality.

We can demonstrate the effect of a change in optimal partition on Newton's Method with the following linear program.

$$\begin{aligned} \min \quad & 2x_1 + x_2 \\ \text{st} \quad & x_1 + x_2 \geq 2 \\ & x_1 - x_2 \leq 4 \\ & x_1 + x_2 \leq 8 \\ & x_1, x_2 \geq 0 \end{aligned} \tag{5.8}$$

and its dual

$$\begin{aligned} \min \quad & 2y_1 + 4y_2 + 8y_3 \\ \text{st} \quad & y_1 + y_2 + y_3 \leq 2 \\ & y_1 - y_2 + y_3 \leq 1 \\ & -y_1 \leq 0 \\ & y_2 \leq 0 \\ & y_3 \leq 0 \end{aligned} \tag{5.9}$$

The optimal solution to these two linear programs is

$$x^* = \begin{bmatrix} 0 \\ 2 \end{bmatrix} \quad \text{and} \quad y^* = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

We can apply damped Newton's Method to the KKT conditions of this problem starting from two pairs of points,

$$\hat{x} = \begin{bmatrix} 6 \\ 2 \end{bmatrix} \quad \text{and} \quad \hat{y} = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}$$

and

$$\bar{x} = \begin{bmatrix} 10^{-6} \\ 8 \end{bmatrix} \quad \text{and} \quad \bar{y} = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}.$$

These two points are the same distance from the optimal solutions to (5.8) and (5.9) and also have approximately the same primal and dual infeasibility (all primal and dual slacks are perturbed from 0 to 10^{-6}).

Table 5.1 demonstrates that although $(\hat{x}, \hat{y})$ and $(\bar{x}, \bar{y})$ have the same infeasibility (Column 5) and are the same distance from optimal (Column 4), the number of iterations required for convergence (Column 2) for $(\hat{x}, \hat{y})$ is nearly twice that for $(\bar{x}, \bar{y})$. The third column of Table 5.1, with **sw** standing for "switch," gives the number of indices i for which the primal variable x_i^* or s_i^* is nonzero but the corresponding component in the starting point is near zero plus the number of indices i for which the dual variables z_i^* or w_i^* is positive, but the corresponding component in the starting point is near zero. An examination of the iteration sequence reveals that this difference can be attributed to the algorithm's attempt to match more components of the starting point $(\hat{x}, \hat{y})$ to the optimal partition of (x^*, y^*).

We can infer from this example that, for primal-dual interior-point methods, if the optimal partition of the linear program defined at a parent node in a branch-and-bound tree is very different from the optimal partition of the linear program at the child node, then the reoptimization could be severely slowed.

Initial Point	iters	sw	$\|\|x^o - x^*\|\|$	$\|\|r_b^0\|\|/(\|\|b\|\| + 1)$
$(\hat{x}, \hat{y})$	18	4	2.00000000000175	1.732050807521381e-06
$(\bar{x}, \bar{y})$	10	2	1.99999900000125	1.732050807521381e-06

Table 5.1 Iteration counts and factor comparisons for damped Newton's method

Chapter 6

Numerical Experience

Our numerical tests were performed on a Sun Ultra 1 Model 140 with 192 megabytes of memory running Solaris 2.6. For our first run of the branch-and-bound algorithm, we solve each node of the search tree generated using an initial point derived from a cold-start heuristic. We then take the branch-and-bound search tree that was created from the first run and re-run the branch-and-bound algorithm using the same search tree. In this second run we warm start each node with one of the four methods outlined in Chapter 3. We allow the branch-and-bound algorithm for the cold-started search tree to run for no more than 500 seconds and generate no more than 500 nodes. Over the entire MIPLIB test set, approximately 8000 nodes were generated. Over the Capacitated Facility Location set, there were 80 nodes (these problems solved to optimality within the time frame given). Due to the size of the Conrail model, we only examined one problem: *con33*. For this problem we increased our solution time to 5000 seconds and generated approximately 20 nodes.

The linear programming solver we use at each node of the branch-and-bound tree is LIPSOL – Linear-programming Interior-Point Solver [65]. Like LIPSOL, our code is implemented under the MATLAB environment. MATLAB is a high level technical computing environment for numeric computation and visualization. Node selection in the branch-and-bound search tree is done by depth first search until an integer incumbent is found. Afterwards, node selection is done by *best bound*. *Best bound* is a node selection strategy in which the node with the smallest lower bound on the optimal solution is examined next. The branching variable is selected by *maximum integer infeasibility* which in our case means that we choose a binary variable whose

value in the optimal solution of the linear programming relaxation is closest to one-half.

Table 6.1 provides the number of feasible nodes in the branch and bound tree that were successfully reoptimized using each of the warm-start techniques. Column 2 in Table 6.1 gives the total number of nodes in the branch-and-bound tree for each method in column 1. The third column gives the number of feasible warm-started nodes that converged to an optimal solution. The number in the fourth column provides the number of feasible warm-started nodes that did not converge to an optimal solution. Additionally, the number in Column 4 includes those parent nodes that were unable to find and save a warm-start solution for its child nodes that satisfied the criteria of the method indicated in the table.

Method	Total	Converged	Not Converged
Total Relative Error	8192	6440	1752
Centered Iterate 1	8168	8078	90
Centered Iterate 2	8187	8088	99
Early Termination	8192	6836	1356

Table 6.1 Node counts for each method

The stopping criterion given in LIPSOL has not been extensively tested, and thus cannot be relied on exclusively for a determination of infeasibility and can in some cases incorrectly conclude that a problem is infeasible. In order to combat this problem, we modify the stopping criteria so that the heuristics used to determine infeasibility are disabled. We concluded that each node in Column 4 of Table 6.1 was not converging to optimal only if after 99 iterations the total relative error was greater than the stopping tolerance. We can see that the Centered Iterate Methods outperformed the other two methods in terms of reliably and correctly converging to optimality.

Table 6.2 gives some idea how the each method performed over the problem set. For each method, the ratio of the number of iterations required to converge from a warm-start initial point to the number of iterations required to converge from cold-start point is reported. Column 2 gives the number of nodes in the branch-and-bound search tree whose ratio (ws/cs) was less than 0.3. Columns 3 and 4 give the number of nodes whose warm-start to cold-start ratio is in [0.3, 0.6) and [0.6, 1) respectively. The last column reports the number of nodes for which the number of warm iterations is greater than the number of cold iterations. Recall that the total number of feasible nodes for which convergence is obtained for each method is different.

Method	ws/cs < 0.3	$0.3 \leq$ ws/cs < 0.6	$0.6 \leq$ ws/cs < 1	ws/cs ≥ 1
Total Relative Error	363	1326	1472	3279
Centered Iterate 1	148	1421	3365	3144
Centered Iterate2	143	1247	3533	3165
Early Termination	420	1291	1918	3207

Table 6.2 Node counts by ws/cs ratio for each method

Table 6.2 also shows that the number of nodes from the Centered Iterate Methods whose warm-start to cold-start iteration ratio was strictly less than 0.3 is lower than the other two methods. However, the total number of feasible nodes whose ratio is less than one is higher than the total number of feasible nodes whose ratio is less than one for the other two methods.

Method	% Converged	% Not Converged	% ws/cs < 1	% ws/cs ≥ 1
Total Relative Error	78.61	21.39	49.08	50.92
Centered Iterate 1	98.90	1.40	61.08	38.92
Centered Iterate2	98.79	1.27	60.87	39.13
Early Termination	83.45	16.55	53.09	46.91

Table 6.3 Performance results for each method
based on convergence and ws/cs ratio

These tables demonstrate that the centering of the parent iterate to be used to reoptimize the child node in the two Centered Iterate Methods is beneficial.

6.1 Factor Calculations

For each of the reoptimization methods referenced in the previous sections, there were a considerable number of problems that either took much longer to reoptimize from a warm start than from a cold start or were unable to converge to optimality within 99 iterations. To gain some insight into this behavior, during the optimization and reoptimization of the branch-and-bound search trees generated for the problems we compute the values of the factors discussed in Chapter 5 thought to affect the convergence of reoptimizations in the primal-dual interior-point framework.

Define $(x^0_{ws}, s^0_{ws}, y^0_{ws}, z^0_{ws}, w^0_{ws})$ as the warm-start point used to initialize a reoptimized child node in the branch-and-bound search tree. Also, define the warm-start optimal solution $(x^*_{ws}, s^*_{ws}, y^*_{ws}, z^*_{ws}, w^*_{ws})$ as the optimal solution to which the warm-started sequence $(x^k_{ws}, s^k_{ws}, y^k_{ws}, z^k_{ws}, w^k_{ws})$ converges. The duality measure μ^0_{ws} is defined as $((x^0_{ws})^T z^0_{ws} + (s^0_{ws})^T w^0_{ws})/2n$ where n is the number of primal variables.

The initial primal infeasibility of the warm-start point is computed in a standard manner as

$$\|Ax^0_{ws} - b\|_2/(\|b\|_2 + 1).$$

The ratio of infeasibility to duality measure of the warm-start point is computed as

$$\|(Ax^0_{ws} - b, A^T y^0_{ws} + z^0_{ws} - w^0_{ws})\|_2/\mu^0_{ws}.$$

Proximity to the central path is measured as

$$\left\| \frac{1}{\mu^0_{ws}} \begin{pmatrix} X^0_{ws} z^0_{ws} \\ S^0_{ws} w^0_{ws} \end{pmatrix} - e \right\|_2$$

The distance that a warm-start point is from the optimal warm-start solution is computed as

$$\|x^0_{ws} - x^*_{ws}\|_2.$$

The next set of tables (Tables 6.4, 6.5, 6.6, and 6.7) report the average values of the factors described in Chapter 5 for each method. The averages are reported within specific warm-start to cold-start iteration ratio ranges. The first column gives the range of the ratio of warm-start to cold-start iterations for the data on that row of the table as well as, in parentheses, the number of nodes involved in the computation. The average percentage of the number of indices that switched from $\mathcal{B}$ to $\mathcal{N}$ is reported in the second column. The third column gives the average Euclidean distance between the warm-start optimal solution and the warm-start point. The last two columns report the average relative primal infeasibility, and the average ratio of primal and dual infeasibility to duality measure, respectively where

$$r^0_b = Ax^0_{ws} - b$$

and

$$r^0_c = A^T y^0_{ws} + z^0_{ws} - w^0_{ws}.$$

(the subscript for the l_2 norms has been omitted for brevity.)

Two problems from the MIPLIB problem set were intentionally omitted from the calculations in Tables 6.4, 6.5, 6.6, and 6.7. Problems *harp2* and *enigma* were omitted because the initial primal infeasibilities of their warm-start points generated by each method were so much larger than that of the other problems tested.

We can see from Tables 6.4, 6.5, 6.6, and 6.7 that as the warm-start iterations to cold-start iterations ratio increases, the greater the change in optimal partition between the parent and child nodes. The same observation holds for the relative primal infeasibility, $||r^0_b||/(||b|| + 1)$.

A correlation between the distance that a warm-start solution is from optimal and the iteration counts ratio is observed for each method as well. Notice that the average distances that the warm-start initial points were from optimality are larger for the two Centered Iterate Methods. This behavior for methods that balance the complementary pairs before saving a warm-start initial solution is not surprising.

No relationship can be seen between the iteration counts ratio and the ratio of infeasibility to duality measure $||(Ax^o_{ws}-b, A^T y^o_{ws}+z^o_{ws}-w^o_{ws})||_2/\mu^o_{ws}$. Nor, interestingly, was there an obvious correlation between the proximity measure used

$$\left\| \frac{1}{\mu^0_{ws}} \begin{pmatrix} X^0_{ws} z^0_{ws} \\ S^0_{ws} w^0_{ws} \end{pmatrix} - e \right\|_2$$

and the warm-start to cold-start iterations ratio. Recent work by Gonzàlez-Lima, El-Bakry, and Tapia [28], indicates that this observation does not indicate that proximity to the central path can be ruled out as a factor. The authors study different centrality measures and show in the general primal-dual interior-point framework that their use makes a computational difference when approaching the central path.

	% sw/n	$\|\|x^o_{ws} - x^*_{ws}\|\|$	$\|\|r^0_b\|\|/(\|\|b\|\|+1)$	$\|\|(r^0_b, r^0_c)\|\|/\mu$
$ws/cs < 0.3$ (333)	0.1846	46.3494	0.0305	5.4864e+07
$0.3 \leq ws/cs < 0.6$ (1300)	1.6032	64.7467	0.0782	1.1241e+08
$0.6 \leq ws/cs < 1$ (1412)	3.6097	411.6424	0.1158	3.5316e+06
$ws/cs \geq 1$ (3241)	4.6833	1100.0468	0.1397	6.1835e+06

Table 6.4 Total Relative Error Method average factor calculations

6.2 Expected Iteration Count

Define $(x^0_{cs}, s^0_{cs}, y^0_{cs}, z^0_{cs}, w^0_{cs})$ as the cold-start point used to initialize a node in the branch-and-bound search tree during the first run. Also, define the cold-start opti-

	% sw/n	$\|\|x^o_{ws} - x^*_{ws}\|\|$	$\|\|r^0_b\|\|/(\|\|b\|\| + 1)$	$\|\|(r^0_b, r^0_c)\|\|/\mu$
$ws/cs < 0.3$ (112)	0.2462	573.1918	0.0493	1.8798e+06
$0.3 \leq ws/cs < 0.6$ (1310)	1.2691	1892.7406	0.0514	1.2190e+07
$0.6 \leq ws/cs < 1$ (3322)	2.4186	5716.0875	0.0878	1.4388e+07
$ws/cs \geq 1$ (3041)	4.7363	7198.7146	0.1556	2.6784e+06

Table 6.5 Centered Iterate Method 1 average factor calculations

	% sw/n	$\|\|x^o_{ws} - x^*_{ws}\|\|$	$\|\|r^0_b\|\|/(\|\|b\|\| + 1)$	$\|\|(r^0_b, r^0_c)\|\|/\mu$
$ws/cs < 0.3$ (104)	0.2283	499.3981	0.0489	1.9360e+06
$0.3 \leq ws/cs < 0.6$ (1156)	1.2481	2220.2404	0.0519	5.1285e+06
$0.6 \leq ws/cs < 1$ (3477)	2.3746	5646.6526	0.0875	1.5318e+07
$ws/cs \geq 1$ (3065)	4.7101	6962.9863	0.1572	3.4817e+06

Table 6.6 Centered Iterate Method 2 average factor calculations

	% sw/n	$\|\|x^o_{ws} - x^*_{ws}\|\|$	$\|\|r^0_b\|\|/(\|\|b\|\| + 1)$	$\|\|(r^0_b, r^0_c)\|\|/\mu$
$ws/cs < 0.3$ (364)	0.3001	60.5900	0.0331	6.4110e+07
$0.3 \leq ws/cs < 0.6$ (1247)	1.5576	302.3966	0.0730	1.0040e+08
$0.6 \leq ws/cs < 1$ (1897)	2.9029	2183.7677	0.1136	4.1616e+07
$ws/cs \geq 1$ (3131)	4.7681	3559.3676	0.1522	1.4268e+07

Table 6.7 Early Termination Method average factor calculations

mal solution $(x^*_{cs}, s^*_{cs}, y^*_{cs}, z^*_{cs}, w^*_{cs})$ as the optimal solution to which the cold-started sequence $(x^k_{cs}, s^k_{cs}, y^k_{cs}, z^k_{cs}, w^k_{cs})$ converges.

To find the expected number of iterations that a warm-started reoptimization should take we do the following. Let x^k_{cs} be the k-th iterate in the cold-started iteration sequence and x^*_{cs} be the solution that it converges to. Also, let x^o_{ws} be the warm-start point and x^*_{ws} be the solution that it converges to. If we define

$$d_k = ||x^k_{cs} - x^*_{cs}||_2$$

and

$$v = |(||x^*_{ws} - x^o_{ws}||_2 e - d)|$$

where e is the vector of all ones, then the expected number of iterations for the warm start x^o_{ws} to converge to x^*_{ws} is k such that v_k is the component of minimum value in v.

Tables 6.8, 6.9, 6.10, and 6.11 report the results of our experiment to determine the number of iterations that would be expected from a warm-start solution based upon the convergence of the iteration sequence from a cold-start solution. If a cold-start iterate that is distance d from optimality takes k iterations, then we hypothesize that a warm-start point that is the same distance d from optimality should take k iterations. Column 1 of Tables 6.8, 6.9, 6.10, and 6.11 indicate the warm-start to cold-start iterations range for the numbers in the next three columns. The second and third columns report the average number of cold iterations and the average number of warm iterations for each range in column one. The final column gives the average expected number of iterations.

As one might expect, the smaller the ratio of warm-start iterations to cold-start iterations, the closer the average number of warm-start iterations is to the expected number of warm-start to cold-start iterations. When the ratio is greater than 0.6 for

each method other than the Centered Iterate Methods the expected iterations count is not a good measure. However, for nodes whose warm-start iterations to cold-start iterations ratio is less than 1, the number of warm-start iterations when using the Centered Iterate Methods remains relatively close to expected.

	cold	warm	expected
$ws/cs < 0.3$	12.2528	2.4298	3.2893
$0.3 \leq ws/cs < 0.6$	10.5306	4.7255	4.1995
$0.6 \leq ws/cs < 1$	13.6787	10.6829	5.7156
$ws/cs \geq 1$	13.0805	22.7824	6.3883

Table 6.8 Total Relative Error Method expected iteration counts

	cold	warm	expected
$ws/cs < 0.3$	19.1339	4.7321	4.8393
$0.3 \leq ws/cs < 0.6$	15.9753	7.2271	6.2257
$0.6 \leq ws/cs < 1$	12.0627	8.9961	6.8537
$ws/cs \geq 1$	12.5363	23.7259	7.0006

Table 6.9 Centered Iterate Method 1 expected iteration counts

	cold	warm	expected
$ws/cs < 0.3$	19.2788	4.7981	4.7115
$0.3 \leq ws/cs < 0.6$	17.2148	7.8444	6.6881
$0.6 \leq ws/cs < 1$	12.0040	9.0400	6.8561
$ws/cs \geq 1$	12.3112	23.7726	6.8546

Table 6.10 Centered Iterate Method 2 expected iteration counts

Tables 6.12 and 6.13 demonstrate the expected iterations count measure on *con33* and *cap41*, respectively. Table 6.12 shows that the expected number of iterations

	cold	warm	expected
$ws/cs < 0.3$	13.1649	2.7382	3.5707
$0.3 \leq ws/cs < 0.6$	12.0693	5.4411	4.5056
$0.6 \leq ws/cs < 1$	13.3144	10.2457	6.1465
$ws/cs \geq 1$	12.9431	22.1764	6.7636

Table 6.11 Early Termination expected iteration counts

based on distance is a very good measure of how well the Total Relative Error Method performs on *con33*. Table 6.13 demonstrates that the Centered Iterate Method 2 outperforms the expected number of iterations based on distance for *cap41*. The centrality of the warm-start solution allows for larger step sizes than the iterate in the cold optimization sequence that is a comparable distance from optimality.

6.3 Branch and Bound: Children Are Different

When selecting a warm start for the child nodes of a particular problem, the warm start chosen may not be a point that is the best possible point for either child node. In fact, of the parent iterates the best warm-starting points for each child may be far apart. Table 6.14 demonstrates this for problem *cap41*. For each binary variable in *cap41* the up-branch (fixing variable to one) and the down-branch (fixing variable to zero) were warm started using each iterate in the parent optimization sequence. Table 6.14 contains the results.

Table 6.14 demonstrates the difference between initializing the optimization of children nodes with the best parent iterate for each versus initializing the optimization of the children nodes with the best iterate for both. The column **Var** is i such that x_i is fractional in the optimal solution of the parent node. The columns **Best Down** and **Best Up** provide information about the reoptimization that converges in the

Node	Cold Its.	Warm Its.	Expected Its.
1	26	9	9
2	26	10	11
3	26	10	9
4	26	10	8
5	26	11	11
6	26	12	12
7	25	13	12
8	25	14	13
9	25	15	13
10	26	16	16
11	27	17	18
12	27	18	18
13	27	19	19
14	27	20	19
15	27	21	21
16	27	21	20
17	26	22	22

Table 6.12 Expected iteration counts for *con33* using Total Relative Error Method.

Node	Cold Its.	Warm Its.	Expected Its.
1	75	15	37
2	75	15	31
3	68	14	36
4	68	14	25
5	66	14	25
6	61	13	27
7	60	13	20
8	59	13	26
9	59	13	23
10	68	15	27
11	58	13	26
12	57	13	24
13	57	13	24
14	57	13	22
15	61	14	29
16	61	14	28
17	56	13	26
18	60	14	19
19	64	15	35
20	55	13	28
21	53	13	25
22	53	13	26

Table 6.13 Expected iteration counts for *cap41* using Centered Iterate Method 2

fewest number of iterations for x_i fixed at 0 and x_i fixed at one, respectively. These columns also give the corresponding number of iterations when a parent iterate is used as the warm start for x_i fixed at zero or one, respectively, The column **P_Itt** under **Best Down** tells which iterate k of the parent optimization was used as the initial point for x_i fixed at zero, and **(Its0)** is the number of iterations required to reoptimize this child problem when it is initialized with the "**P_Itt**"th iterate. The columns under **Best Up** are defined likewise.

Var	Best Down		Best Up		Best Up + Down	
	P_Itt	(Its0)	P_Itt	(Its1)	P_Itt	(Its0, Its1)
1	1	(22)	50	(3)	41	(29, 12)
7	38	(19)	46	(7)	41	(19, 12)
8	1	(21)	46	(7)	1	(21, 22)
10	48	(11)	46	(7)	46	(13, 7)
12	1	(21)	46	(7)	44	(32, 9)
14	1	(22)	46	(7)	40	(23, 13)
15	47	(14)	46	(7)	46	(15, 7)
16	44	(17)	46	(7)	44	(17, 9)

Table 6.14 Iteration count comparisons for problem *cap41*

The minimum number of iterations required to reoptimize *cap41* when x_1 is fixed to zero is 22 using parent iterate 1 as the warm-start initial point, and when x_1 is fixed to one the minimum number of iterations is 3 using parent iterate 50 as the warm-start initial point. Yet, the parent iterate used as the warm-start initial point that attains the minimum total number of iterations for both the up-branch and the down-branch is iterate 41. Starting from iterate 41 requires 29 iterations for the down-branch and 12 iterations for the up-branch to converge to optimal. Notice that the number of iterations to solve the down-branch alone starting from iterate 41 is more than the sum of the minimum number of iterations for the up-branch and the minimum number of iterations for the down-branch.

Problem *cap41* demonstrates the difficulty that arises when an attempt is made to find a "good" starting point for two problems that for interior-point methods are vastly different. This difference can be attributed primarily to a change in the optimal partition that occurs when a variable is fixed to either zero or one. The optimal partition of the parent node does not change when the up branch is created, however 48 indices switch from $\mathcal{B}$ to $\mathcal{N}$ and vice versa when the down branch is created.

nt= 1648	$x_1 = 0$, cold iter= 60, Δ opt part= 48			$x_1 = 1$, cold iter= 58, Δ opt part= 0		
Parent Iterate	2	42	51	2	42	51
Iterations	22	29	83	21	12	3
$\|\|x^o_{ws} - x^*_{ws}\|\|$	1.2596e+04	6.1113e+03	4.5263e+03	1.2672e+04	5.7901e+03	3.1389e+01
$\|\|r^0_b\|\|/(\|\|b\|\| + 1)$	1.6557e+01	2.7452e-01	3.0082e-01	1.6554e+01	6.4194e-02	1.7757e-03
$\|\|(r^0_b, r^0_c)\|\|/\mu$	1.9238e+02	2.7615e+03	5.3343e+07	1.8786e-01	6.3119e+00	7.8633e+01

Table 6.15 Key factor comparisons for problem *cap41*

In addition to the difficulty of finding a warm start that "pleases two children" (the up-branch and the down-branch), reoptimizing with interior-point methods requires that one not find a warm-starting point that is so good for a child node that a good warm-start point cannot be found for the subsequent child nodes of this child node. For the Centered Iterate Method 1 this can happen when the warm start is close to optimal, but not close to the central path. Several iterations could be wasted readjusting to this new central path instead of moving the iteration sequence toward optimality.

We can see this happening if we examine the first four nodes in the branch-and-bound tree for problem *stein9*. Row 1 gives the number of the node. Row 2 of Table 6.3 gives the number of iterations required to converge from a warm start for the first four nodes. Rows 3 and 4 give the number of iterations required to converge to optimality when the nodes are warm started with a node from the parent found by

the Centered Iterate Method 1. However, Row 3 contains not only the number of iterations required to reoptimize the node using this warm start, it also contains the number of iterations required to reoptimize *and* find by the Centered Iterate Method 1 a warm-start point to be used for its subsequent children.

Node in B&B tree for problem *stein9*	1	2	3	4
# cold-start iterations	6	8	9	9
# warm-start iterations, find child warm start	7	9	32	25
# warm-start iterations, don't find child warm start	3	3	17	15

Table 6.16 Iteration counts for first four nodes in problem *stein9* using Centered Iterate Method 1

Table 6.3 demonstrates that if we do not look for a warm-start point for the child nodes of any of the first four nodes, we see that the problem can be solved in fewer iterations. For node 2, for example, the number of iterations to solve this node from a cold start is 8, and to reoptimize this node directly from a good warm start requires only three iterations. When a centered-iterate warm-start point must be saved for the next child extra iterations are required to find a point that satisfies the centrality criteria for the warm-start point.

6.4 Newton's Method: The Pull of the Boundary

Difficulties associated with reoptimizing child nodes in the branch-and-bound code can also be attributed to the effects of Newton's method that underlies the primal-dual interior-point method. As we demonstrated in the previous sections, a change in optimal partition can cause problems for the interior-point algorithm. Several nodes that were reoptimized in the branch-and-bound search tree suffered from the effect of taking short steps near a boundary.

It can be shown for the primal and dual programs (5.8) and (5.9) that the distance to optimality and primal feasibility are not necessarily the dominant factors for the difficulties in reoptimization. In fact, it can be shown that there exists a linear program such that for any $\epsilon \geq 0$, we can find a point that has distance ϵ to optimality, but Newton's Method does not converge.

Let the ϵ-ball around v^* be defined as

$$B(v^*, \epsilon) = \{v : ||v - v^*|| \leq \epsilon\},$$

then we have the following theorem:

Theorem 6.1 For any $\epsilon > 0$, there exists a linear program (LP),

$$\begin{aligned} \min \quad & c^T x \\ \text{st} \quad & Ax = \quad b \\ & x \geq 0, \end{aligned} \tag{6.1}$$

such that the following properties hold:

1. The LP has a nondegenerate optimal solution (x^*, y^*, z^*).
2. The LP has a feasible primal-dual solution $(x^o, y^o, z^o) \in B((x^*, y^*, z^*), \epsilon)$.
3. The sequence (x^k, y^k, z^k) generated by Newton's Method or damped Newton's Method starting from (x^o, y^o, z^o) does not converge to (x^*, y^*, z^*).

Proof. Consider the following linear program:

$$\min \quad 2x_1 + x_2 + x_3 \tag{6.2}$$

$$\begin{aligned} \text{st} \quad x_1 - x_2 + x_3 &= \frac{1}{2}\,\epsilon \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

and its dual

$$\begin{aligned} \min \quad & \tfrac{1}{2}\,\epsilon\, y \\ \text{st} \quad & y + z_1 = 2 \\ & -y + z_2 = 1 \\ & y + z_3 = 1 \\ & z_1, z_2, z_3 \geq 0 \end{aligned} \tag{6.3}$$

1. Clearly, the pair (6.2) and (6.3) have a unique primal-dual optimal solution (x^*, y^*, z^*) equal to $(0, 0, \frac{1}{2}\epsilon, 0, 2, 1, 1)$.

2. Consider the point (x^o, y^o, z^o) equal to $(\frac{3}{4}\epsilon, \frac{1}{4}\epsilon, 0, 0, 2, 1, 1)$. This point is feasible, and since

$$||(x^*, y^*, z^*) - (x^o, y^o, z^o)|| = \sqrt{(-\frac{3}{4}\epsilon)^2 + (-\frac{1}{4}\epsilon)^2 + (\frac{1}{2}\epsilon)^2} \; < \; \epsilon$$

   it is contained in $B((x^*, y^*, z^*), \epsilon)$

3. At each iteration of Newton's Method or Damped Newton's Method the following system is solved for the pair 6.2 and 6.3 at each iteration k:

$$F'(x^k, y^k, z^k)(\Delta x^k, \Delta y^k, \Delta z^k) = -F(x^k, y^k, z^k)$$

where

$$F(x,y,z) = \begin{pmatrix} x_1 - x_2 + x_3 - \frac{1}{2}\,\epsilon \\ y + z_1 - 2 \\ -y + z_2 - 1 \\ y + z_3 - 1 \\ x_1 z_1 \\ x_2 z_2 \\ x_3 z_3 \end{pmatrix}$$

If our initial point is (x^o, y^o, z^o), then at iteration $k = 0$ the linearized complementarity equation for x_3 and z_3

$$z_3^o \Delta x_3^o + x_3^o \Delta z_3^o = -x_3^o z_3^o$$

implies that $\Delta x_3^o = 0$. Thus, $x_3^1 = x_3^o + \alpha \Delta x_3^o = 0$ where $\alpha <= 1$ for damped Newton's Method and $\alpha = 1$ for Newton's Method. By induction, $x_3^k = 0$ for all iterations k of Newton's Method or damped Newton's Method, but $x_3^* = \frac{1}{2}\epsilon$ at optimality. Therefore, neither Newton's Method nor damped Newton's Method will converge from (x^o, y^o, z^o) to (x^*, y^*, z^*).

We know that under standard assumptions primal-dual interior-point methods converge from any starting point for which the nonnegative variables are strictly greater than zero. Theorem 6.1 shows that closeness to optimal is not enough to ensure convergence. As in the case of the linear programs in this theorem, we see that the "pull" of the boundary precludes convergence. An initial point can in fact be arbitrarily close to optimal, yet still Newton's Method is unable to converge to optimality. By continuity, we can argue that for initial points sufficiently close to the boundary, as may be the case for a warm-start initial point, the "pull" of the boundary can significantly slow convergence.

Table 6.4 demonstrates this phenomenon. The first column is the ratio of warm-start to cold-start iterations. The second column is the number of variables that switch from $\mathcal{B}$ to $\mathcal{N}$ when the child node is created. The third column is the number of primal variables. The fourth and fifth columns report the distance the primal x variable is from optimality and the relative primal infeasibility, respectively. The final column is the average step size taken during the reoptimization. For each of the eleven nodes in the table, the primal variable x is very close to optimal, the number of switches is relatively low, and the relative primal infeasibility is small. Yet, the average step size is small causing the warm-start to cold-start iterations ratio to be high.

ws/cs	sw	n	$\|\|x^o_{ws} - x^*_{ws}\|\|$	$\|\|r^0_b\|\|/(\|\|b\|\| + 1)$	avg(α_p, α_d)
1.9286	0	3290	0.9688	0.0838	(0.1948, 0.3024)
1.0526	8	644	0.8397	0.1534	(0.2919, 0.3470)
1.1429	4	644	0.5641	0.0788	(0.2216, 0.2920)
1.1739	6	644	0.6177	0.0785	(0.1426, 0.2376)
1.5000	6	644	0.9644	0.0861	(0.3087, 0.2499)
1.3125	6	644	0.6392	0.0654	(0.2195, 0.3197)
1.3750	4	644	0.8738	0.1337	(0.2380, 0.2930)
1.2857	4	644	0.9106	0.1709	(0.2882, 0.3542)
1.1818	4	103	0.9480	0.2879	(0.3096, 0.3169)
1.5455	4	103	0.9270	0.2085	(0.2546, 0.2587)
1.1667	3	535	0.9956	0.0900	(0.3169, 0.2856)

Table 6.17 These eleven nodes from the MIPLIB problem set demonstrate the effect of the "pull" of the boundary in spite of the primal solution being very close to optimal.

Chapter 7

Concluding Remarks

Of the methods implemented in this work, we have shown that the prior centering of the warm-start initial point suggested by Gondzio [23] provides for a more efficient and reliable reoptimization technique. However, none of the techniques presented were capable of keeping the ratio of warm-start iterations to cold-start iterations below one for all nodes.

We have established that a rapid reoptimization is not always possible in a branch-and-bound framework when using an interior-point method to solve the linear programs generated. We have shown that the "pull" of the boundary, the distance to optimality, and the initial primal infeasibility can negatively effect reoptimizations. We have also shown that the change in optimal partition from the parent node when a child node is created impacts the efficiency of the reoptimization.

This study shows that using an interior-pont method as the linear programming problem solver in the branch-and-bound framework does not have wide applicability for general 0/1 integer programming problems. However, for those problems where the linear programs generated in a branch-and-bound search tree solve faster for interior-point methods than simplex methods, the centering idea introduced by Gondzio [23] provides the best basis for an effective reoptimization technique. Continued research is needed in the direction of the present study in order to provide comprehensive guidelines for the most effective utilization of interior-point methods in a branch-and-bound algorithm.

Appendix A

Problem Statistics

Problem statistics for each problem set can be found in the following tables. Tables A.1 and A.2 correspond to the MIPLIB problem set, and Table A.3 corresponds to the Capacitated Facility Location problem set. The first column, **NAME**, contains the name of the model. The next two columns, **ROWS** and **COLS**, contain the number of rows (constraints), not including free rows, and the number of columns (variables) in the problem, respectively. The column **INT** specifies the number of variables that are restricted to integer values, and the **0/1** column specifies how many of these integer variables are binary. The column **CONT** specifies the number of variables that are continuous. The next two columns report the best-known integral solution and the optimal value with the integrality restrictions relaxed, respectively. The entries in the **INT SOLN** column indicate the optimal integer solution. The last column **LP SOLN** contains the solution to the linear programming relaxation for each problem.

For the Conrail problem statistics in Table A.4, the first column, **#TP**, and second column, **#ETP**, correspond to the value of the parameters $NumTP$ and $NumEnsTP$, respectively. The next two columns, **ROWS** and **COLS**, contain the number of rows (constraints), and the number of columns (variables) in the problem, respectively. The column **INT** specifies the number of variables that are restricted to integer values, and the **0/1** column specifies how many of these integer variables are binary. The column **CONT** contains the number of variables that are continuous.The next two columns report the best-known integral solution and the optimal value with the integrality restrictions relaxed, respectively. The entries in the **INT SOLN**

column indicate the best known integer solution. The last column **LP SOLN** contains the solution to the linear programming relaxation for each problem.

NAME	ROWS	COLS	INT	0/1	CONT	INT SOLN	LP SOLN
10teams	230	2025	1800	ALL	225	924	917
air01	23	771	771	ALL	0	6796	6743.0
air02	50	6774	6774	ALL	0	7810	7640.0
air03	124	10757	10757	ALL	0	340160	338864.25
air04	823	8904	8904	ALL	0	56137	55535.436
air05	426	7195	7195	ALL	0	26374	25877.609
cap6000	2176	6000	6000	ALL	0	-2451377	-2451537.325
cracpb1	143	572	572	ALL	0	22199	22199.0
danoint	664	521	56	ALL	465	65.67	62.637280418
dcmulti	290	548	75	ALL	473	188182	183975.5397
egout	98	141	55	ALL	86	568.101	149.589
enigma	21	100	100	ALL	0	0.0	0.0
fiber	363	1298	1254	ALL	44	405935.18000	156082.51759
fixnet6	478	878	378	ALL	500	3983	1200.88
harp2	112	2993	2993	ALL	0	-73899798.00	-74353341.502
khb05250	101	1350	24	ALL	1326	106940226	95919464.0
l152lav	97	1989	1989	ALL	0	4722	4656.36
lp4l	85	1086	1086	ALL	0	2967	2942.5
lseu	28	89	89	ALL	0	1120	834.68
markshare1	6	62	50	ALL	12	1	0
markshare2	7	74	60	ALL	14	1	0
mas74	13	151	150	ALL	1	11801.1857	10482.795280
mas76	12	151	150	ALL	1	40005.0541	38893.903641
mkc	3411	5325	5323	ALL	2	-553.75(not opt)	-611.85000000
mod008	6	319	319	ALL	0	307	290.93
mod010	146	2655	2655	ALL	0	6548	6532.08
mod011	4480	10958	96	ALL	10862	-54558535	-62121982.552
modglob	291	422	98	ALL	324	20740508	20430947.0
p0033	16	33	33	ALL	0	3089	2520.57
p0040	23	40	40	ALL	0	62027	61796.55
p0201	133	201	201	ALL	0	7615	6875.0
p0282	241	282	282	ALL	0	258411	176867.50
p0548	176	548	548	ALL	0	8691	315.29
p2756	755	2756	2756	ALL	0	3124	2688.75
pk1	45	86	55	ALL	31	11.0	0.0
pp08a	136	240	64	ALL	176	7350.0	2748.3452381
pp08aCUTS	246	240	64	ALL	176	7350.0	5480.6061563
qiu	1192	840	48	ALL	792	-132.873137	-931.638857

Table A.1 MIPLIB Problem Statistics

NAME	ROWS	COLS	INT	0/1	CONT	INT SOLN	LP SOLN
rentacar	6803	9557	55	ALL	9502	30356761	28806137.644
rgn	24	180	100	ALL	80	82.1999	48.7999
sample2	45	67	21	ALL	46	375	247.0
sentoy	30	60	60	ALL	0	-7772	-7839.278
set1ch	492	712	240	ALL	472	54537.75	32007.73
stein9	13	9	9	ALL	0	5	4.0
stein15	36	15	15	ALL	0	9	7.0
stein27	118	27	27	ALL	0	18	13.0
stein45	331	45	45	ALL	0	30	22.0
vpm1	234	378	168	ALL	210	20	15.4167
vpm2	234	378	168	ALL	210	13.75	9.8892645972

Table A.2 MIPLIB Problem Statistics, continued

NAME	ROWS	COLS	INT	0/1	CONT	INT SOLN	LP SOLN
cap41	66	816	16	ALL	800	4.3680668661e+09	4.3680342681e+09
cap42	66	816	16	ALL	800	4.3681418661e+09	4.3680875361e+09
cap43	66	816	16	ALL	800	4.3682168661e+09	4.3681408041e+09
cap44	66	816	16	ALL	800	4.3683293661e+09	4.3682207061e+09

Table A.3 Capacitated Facility Location Problem Statistics

NAME	#TP	#ETP	ROWS	COLS	INT	0/1	CONT	BEST INT	LP SOLN
con33	3	3	6118	17442	876	ALL	16566	705	630.69047619
con42	4	2	7607	17532	1168	ALL	16364	722	617.96071429
con44		4	7607	23048	1168	ALL	21880	695	605.14285714
con62	6	2	10585	23228	1752	ALL	21476	708	600.36904762
con63		3	10585	25986	1752	ALL	24234	704	591.57142857
con66		6	10585	34260	1752	ALL	32508	692	580.54761905
con82	8	2	13563	28924	2336	ALL	26588	704	586.84523810
con84		4	13563	34440	2336	ALL	32104	696	575.57467532
con88		8	13563	45472	2336	ALL	43136	683	568.58333333
con122	12	2	19519	40316	3504	ALL	36812	716	574.84523810
con123		3	19519	43074	3504	ALL	39570	701	569.18650794
con124		4	19519	45832	3504	ALL	42328	696	563.369048
con126		6	19519	51348	3504	ALL	47844	695	558.88095238
con1212		12	19519	67896	3504	ALL	64392	688	554.36309524
con242	24	2	37387	74492	7008	ALL	67484	717	564.91655349
con243		3	37387	77250	7008	ALL	70242	714	558.91321081
con244		4	37387	80008	7008	ALL	73000	695	553.11784297
con246		6	37387	85524	7008	ALL	78516	689	549.35912698
con248		8	37387	91040	7008	ALL	84032	691	546.39812091
con2412		12	37387	102072	7008	ALL	95064	689	544.74463167
con2424		24	37387	135168	7008	ALL	128160	688	*

Table A.4 Conrail Problem Statistics

Appendix B

Factor Calculations

The following tables provide for each reoptimization method the factor calculations for each problem set. The second column gives the average warm-start to cold-start iterations ratio for the problem named in column one. The **NODES** column reports the number of reopt nodes in branch and bound tree. The average of the number of indices that switched from B to N divided by the number of primal variables is in column four. The average distance between the warm-start optimal solution and the warm-start point, the average of the relative primal infeasiblity and the average of the ratio of primal and dual infeasibilty to duality measure over all of the nodes in the branch and bound tree are reported in the last three columns.

PROB	NODES	ws/cs	sw/n	$\|\|x^o_{ws} - x^*_{ws}\|\|$	$\|\|r^0_b\|\|/(\|\|b\|\|+1)$	$\|\|(r^0_b, r^0_c)\|\|/\mu$
10teams	64	2.4917	0.0065	1.1888e+00	0.0590	1.5544e+07
air01	6	0.6604	0.0022	1.2195e+00	0.2078	3.5706e+01
air02	16	1.1585	0.0008	1.1172e+00	0.1842	6.1501e+01
air04	2	0.4914	0.0038	1.6136e+00	0.0444	1.9576e+01
air05	7	1.0262	0.0038	1.7579e+00	0.0837	3.3579e+01
bm23	252	1.0612	0.0752	1.6516e+01	0.1538	5.2754e+02
cap6000	8	0.7764	0.0004	7.8619e+02	0.0017	7.2654e+04
cracpb1	74	0.3176	0.0056	4.2956e+00	0.1050	1.9454e+07
danoint	57	2.5796	0.0854	2.1801e+02	0.1404	3.1840e+05
dcmulti	203	2.1599	0.0257	3.5686e+02	0.1579	6.5662e+03
egout	176	1.2251	0.0402	3.4722e+01	0.1583	1.4917e+03
enigma	68	0.7098	0.0367	1.5529e+00	4685.1069	3.3954e+11
fiber	5	2.4314	0.0044	2.5931e+01	0.0396	1.4445e+05
fixnet6	20	1.4271	0.0113	1.4008e+02	0.1451	1.1790e+03
harp2	1	0.5479	0.0003	1.3500e+00	3504386.9479	4.8360e+10
khb05250	86	n/a	n/a	n/a	n/a	n/a
l152lav	13	1.0398	0.0032	1.7248e+00	0.0697	9.1315e+02
lp4l	113	0.8724	0.0066	1.8735e+00	0.0719	7.1732e+01
lseu	47	1.1480	0.0218	9.1596e+01	0.0872	1.0215e+05
markshare1	250	0.4737	0.0046	1.7850e+00	0.0295	2.9324e+08
markshare2	250	0.4620	0.0042	1.7479e+00	0.0248	3.3619e+08
mas74	250	1.0605	0.0194	1.6188e+03	0.0464	5.4351e+05
mas76	250	1.0020	0.0205	1.1684e+03	0.0504	8.3829e+05
mkc	13	0.4471	0.0012	4.9270e+00	0.2665	6.5415e+05
mod008	249	0.5869	0.0050	7.3061e-01	0.0985	4.3283e+03
mod010	41	0.7078	0.0025	2.3008e+00	0.0671	1.8452e+02
mod013	256	1.1248	0.0365	2.3673e+01	0.1337	7.6564e+02
modglob	165	1.8644	0.0169	1.6020e+04	0.3311	4.4249e+05
p0033	239	1.2255	0.0325	1.4990e+02	0.0889	4.0871e+06
p0040	79	1.1718	0.0336	4.0928e+02	0.2790	4.2823e+05
p0201	31	2.3387	0.0475	1.0065e+01	0.0804	6.2674e+02
p0282	91	2.4303	0.0140	7.9477e+01	0.0749	2.8531e+03
p0548	64	0.2920	0.0032	3.3450e+02	0.0022	2.5898e+05
p2756	42	0.1324	0.0000	6.5946e+00	0.0002	1.0404e+05
pipex	212	1.2491	0.0435	3.9213e+00	0.0741	1.5948e+01
pk1	250	0.9448	0.0691	7.3332e+00	0.0319	1.1178e+08
pp08a	226	1.5013	0.0306	2.3367e+02	0.1093	6.1736e+03
pp08aCUTS	239	1.5361	0.0288	3.0713e+02	0.1214	4.5186e+03
qiu	61	1.1934	0.0530	2.8287e+01	0.0807	1.9715e+02

Table B.1 Average of data for Total Relative Error Method on the MIPLIB problem set

PROB	NODES	ws/cs	sw/n	$\|\|x^o_{ws} - x^*_{ws}\|\|$	$\|\|r^0_b\|\|/(\|\|b\|\|+1)$	$\|\|(r^0_b, r^0_c)\|\|/\mu$
rentacar	1	1.2083	0.0031	5.1893e+02	0.0015	5.5989e+02
rgn	240	1.8703	0.0310	3.3339e+00	0.2807	7.0501e+05
sample2	236	1.3586	0.0552	3.5343e+01	0.2367	7.5296e+03
sentoy	278	1.1592	0.0366	1.7046e+03	0.0542	2.4139e+04
set1ch	71	1.3891	0.0067	4.3705e+02	0.0482	9.7852e+03
stein9	60	0.6977	0.1323	1.5716e+00	0.3634	4.5393e+03
stein15	292	0.9447	0.1025	2.4367e+00	0.2602	8.7593e+03
stein27	250	1.0240	0.0904	3.8747e+00	0.1645	6.0448e+04
stein45	126	1.1282	0.0599	5.4083e+00	0.1219	1.3017e+04
vpm1	250	0.8330	0.0095	1.2940e+02	0.0007	5.4250e+02
vpm2	161	1.6794	0.0067	7.4468e+01	0.0003	1.0203e+02

Table B.2 Average of data for Total Relative Error Method on the MIPLIB problem set, continued.

PROB	NODES	ws/cs	sw/n	$\|\|x^o_{ws} - x^*_{ws}\|\|$	$\|\|r^0_b\|\|/(\|\|b\|\|+1)$	$\|\|(r^0_b, r^0_c)\|\|/\mu$
10teams	31	0.5510	0.0029	7.3935e-01	0.0460	4.3667e+07
air01	4	0.9962	0.0029	1.2894e+00	0.1722	5.8632e+01
air02	16	1.2598	0.0008	1.0770e+00	0.1755	8.7668e+02
air04	2	0.4920	0.0038	1.6962e+00	0.0433	1.0755e+01
air05	7	1.0625	0.0038	1.5780e+00	0.0625	2.0109e+02
bm23	252	1.3920	0.0752	1.5342e+01	0.1353	1.0186e+04
cap6000	8	0.2824	0.0004	8.8109e+02	0.0099	1.8366e+03
cracpb1	74	0.3162	0.0056	4.5709e+00	0.1108	1.9786e+07
danoint	57	2.6967	0.0854	1.9720e+02	0.1436	1.0487e+06
dcmulti	242	0.9715	0.0255	8.2732e+02	0.1777	1.8957e+01
egout	181	1.2415	0.0401	3.7531e+01	0.1454	2.5323e+03
enigma	101	2.7916	0.0256	1.1349e+00	3073.8375	3.0467e+11
fiber	5	0.6543	0.0044	8.9750e+01	0.0302	6.0473e+00
fixnet6	20	1.1267	0.0113	4.8921e+02	0.1416	2.0101e+03
harp2	1	0.5479	0.0003	1.3500e+00	3504386.9479	4.8360e+10
khb05250	170	2.1720	0.0081	8.2796e+03	0.0950	1.6253e+01
l152lav	53	1.1706	0.0038	1.2840e+00	0.0465	6.4938e+05
lp4l	40	1.0339	0.0051	1.5069e+00	0.0479	1.2595e+03
lseu	239	1.1761	0.0171	1.5865e+02	0.1346	2.7914e+05
markshare1	249	0.5522	0.0045	1.2512e+00	0.0292	3.0128e+08
markshare2	250	0.5181	0.0042	1.7085e+00	0.0246	3.4286e+08
mas74	250	0.9309	0.0194	9.4630e+02	0.0274	8.5234e+05
mas76	250	0.7247	0.0205	7.9272e+02	0.0315	1.5800e+05
mkc	13	0.4263	0.0012	6.8646e+01	0.4412	6.1377e+02
mod008	249	0.6764	0.0050	4.4972e+00	0.0772	1.7767e+05
mod010	41	0.7189	0.0025	1.7842e+00	0.0313	7.0330e+03
mod013	248	0.9575	0.0365	2.3532e+01	0.1178	1.2133e+03
modglob	242	1.3306	0.0214	3.7025e+04	0.4787	4.8346e+02
p0033	231	1.1736	0.0328	1.2442e+02	0.0792	2.0097e+06
p0040	79	0.9054	0.0331	4.4876e+02	0.3313	1.3659e+03
p0201	163	0.7817	0.0206	1.3579e+01	0.0692	1.5015e+02
p0282	127	1.3293	0.0133	2.0316e+02	0.0813	4.0883e+01
p0548	64	0.3744	0.0032	3.2106e+03	0.0033	2.3553e+05
p2756	42	0.1429	0.0000	7.4609e+01	0.0002	1.0275e+05
pipex	138	1.2435	0.0478	3.8560e+00	0.0661	8.2566e+01
pk1	250	1.1779	0.0691	3.8146e+00	0.0251	4.2872e+08
pp08a	234	1.2708	0.0304	3.8477e+02	0.1022	5.7723e+03
pp08aCUTS	250	1.2964	0.0281	3.8945e+02	0.1225	1.3361e+03
qiu	61	0.8425	0.0530	3.0665e+01	0.0629	2.2739e+01

Table B.3 Average of data for Early Termination Method on the MIPLIB problem set.

PROB	NODES	ws/cs	sw/n	$\|\|x^o_{ws} - x^*_{ws}\|\|$	$\|\|r^0_b\|\|/(\|\|b\|\|+1)$	$\|\|(r^0_b, r^0_c)\|\|/\mu$
rentacar	17	3.3919	0.0177	3.2996e+04	0.1303	2.2011e+02
rgn	239	1.9973	0.0310	4.0682e+00	0.2780	1.2897e+06
sample2	238	1.4720	0.0552	3.6368e+01	0.2357	3.8459e+04
sentoy	278	0.9544	0.0366	1.3327e+03	0.0411	1.2885e+04
set1ch	124	0.7679	0.0059	2.5258e+04	0.0516	2.5165e+02
stein9	42	0.7863	0.1475	1.7229e+00	0.3446	5.8065e+02
stein15	259	1.0191	0.1016	2.4649e+00	0.2570	6.0786e+03
stein27	249	1.0575	0.0905	3.4888e+00	0.1517	2.9992e+04
stein45	126	1.3261	0.0599	4.5523e+00	0.1027	9.9933e+05
vpm1	213	1.1129	0.0096	1.2483e+02	0.0007	6.4226e+04
vpm2	22	2.2579	0.0087	1.8118e+02	0.0015	5.6283e+03

Table B.4 Average of data for Early Termination Method on the MIPLIB problem set, continued.

PROB	NODES	ws/cs	sw/n	$\|\|x^o_{ws} - x^*_{ws}\|\|$	$\|\|r^0_b\|\|/(\|\|b\|\|+1)$	$\|\|(r^0_b, r^0_c)\|\|/\mu$
10teams	80	0.7135	0.0068	1.4536e+00	0.0067	9.4873e-01
air01	6	1.4091	0.0022	1.8274e+00	0.1822	1.6882e-01
air02	15	0.8721	0.0008	1.1779e+00	0.1414	2.0928e+00
air04	2	1.2297	0.0038	6.3616e+00	0.0030	2.4501e-02
air05	7	0.8379	0.0038	4.2390e+00	0.0200	2.1892e-01
bm23	252	0.8258	0.0752	1.3533e+01	0.1273	2.4824e+02
cap6000	8	1.1356	0.0004	6.2195e+03	0.0001	1.5974e+00
cracpb1	179	0.8553	0.0119	6.4036e+00	0.1079	7.4645e+00
danoint	57	1.9476	0.0854	2.3845e+02	0.1060	9.5059e+02
dcmulti	249	1.4512	0.0253	1.0670e+03	0.1515	8.7004e-01
egout	186	1.5010	0.0394	5.4148e+01	0.1225	1.0049e+01
enigma	183	1.2949	0.0310	1.0579e+00	2579.3693	6.1610e+10
fiber	100	0.7652	0.0030	9.5140e+01	0.0384	1.8934e-01
fixnet6	156	1.5517	0.0029	2.7904e+02	0.0642	8.4696e+03
harp2	13	0.4986	0.0013	5.1144e+06	1104688.0458	1.3169e+03
khb05250	187	1.1349	0.0080	7.3645e+03	0.0411	3.0797e-02
l152lav	53	0.7534	0.0038	3.4801e+00	0.0041	5.7191e-01
lp4l	111	0.6256	0.0065	2.5191e+00	0.0137	2.8972e+00
lseu	244	0.9003	0.0172	1.2538e+02	0.1586	9.5632e+06
markshare1	250	0.6289	0.0046	1.7110e+00	0.0293	1.1185e+08
markshare2	250	0.6175	0.0042	1.6867e+00	0.0246	1.6437e+08
mas74	250	0.4709	0.0194	3.8599e+03	0.0264	8.0806e+05
mas76	250	0.4517	0.0205	5.5903e+03	0.0307	1.2636e+04
mkc	13	0.8424	0.0012	2.9736e+00	0.2493	9.4848e+02
mod008	249	0.4412	0.0050	1.0151e+00	0.0632	1.9374e+06
mod010	40	0.7346	0.0026	4.6372e+00	0.0051	2.7309e-01
mod013	262	1.5793	0.0363	2.7996e+01	0.1078	1.1513e+01
modglob	250	1.5075	0.0209	1.4649e+05	0.6094	3.9582e-01
p0033	250	0.9150	0.0324	1.7024e+02	0.0815	1.6159e+03
p0040	85	1.4322	0.0328	6.0912e+03	0.2675	2.9382e+00
p0201	250	0.8405	0.0150	9.1840e+00	0.0711	3.5881e+00
p0282	126	0.8296	0.0134	4.2018e+02	0.0533	4.2079e-01
p0548	153	0.5563	0.0037	1.8644e+03	0.0034	2.7996e+04
p2756	41	0.3750	0.0000	3.2609e+02	0.0002	9.4207e+01
pipex	249	1.3513	0.0434	4.4369e+00	0.0631	3.8136e-01
pk1	250	0.9583	0.0691	4.1874e+00	0.0249	1.9023e+04
pp08a	250	1.6319	0.0292	2.4894e+02	0.0926	2.3213e+01
pp08aCUTS	250	1.5566	0.0281	3.8061e+02	0.1030	3.5622e+01
qiu	42	1.3860	0.0604	2.2998e+01	0.0561	2.6998e+01

Table B.5 Average of data for Centered Iterate Method 1 on the MIPLIB problem set.

PROB	NODES	ws/cs	sw/n	$\|\|x^o_{ws} - x^*_{ws}\|\|$	$\|\|r^0_b\|\|/(\|\|b\|\|+1)$	$\|\|(r^0_b, r^0_c)\|\|/\mu$
rentacar	21	1.5806	0.0172	2.3575e+04	0.0981	2.4456e+01
rgn	250	1.4061	0.0302	3.1985e+00	0.2646	3.4951e+04
sample2	239	2.1419	0.0550	3.8958e+01	0.2264	1.8900e+01
sentoy	278	0.7639	0.0366	3.0414e+03	0.0370	1.5753e+02
set1ch	208	1.5772	0.0049	6.7567e+02	0.0394	1.1918e+05
stein9	60	1.7007	0.1323	1.5201e+00	0.3608	2.9045e+01
stein15	293	1.6619	0.1024	2.1368e+00	0.2512	5.9794e+02
stein27	250	1.3552	0.0904	3.0875e+00	0.1460	7.8387e+02
stein45	124	1.3639	0.0596	4.1251e+00	0.1005	3.4251e+02
vpm1	250	1.6433	0.0095	1.6562e+02	0.0003	2.1453e+00
vpm2	160	1.4449	0.0067	1.0433e+02	0.0003	8.1737e-01

Table B.6 Average of data for Centered Iterate Method 1 on the MIPLIB problem set, continued.

PROB	NODES	ws/cs	sw/n	$\|\|x^o_{ws} - x^*_{ws}\|\|$	$\|\|r^0_b\|\|/(\|\|b\|\|+1)$	$\|\|(r^0_b, r^0_c)\|\|/\mu$
10teams	83	0.7153	0.0066	1.4369e+00	0.0077	1.7484e+00
air01	6	1.3535	0.0022	1.8391e+00	0.1977	1.8543e-01
air02	16	1.4030	0.0008	1.1964e+00	0.1502	2.2232e+00
air04	2	1.0108	0.0065	8.0037e+01	0.0230	3.2308e+01
air05	7	0.8020	0.0038	4.2248e+00	0.0259	2.8344e-01
bm23	252	0.8629	0.0752	1.3527e+01	0.1274	2.4945e+02
cap6000	8	1.1612	0.0004	6.2187e+03	0.0001	1.5972e+00
cracpb1	181	0.9334	0.0118	6.3754e+00	0.1111	7.3098e+00
danoint	57	2.0026	0.0854	2.3840e+02	0.1060	9.5532e+02
dcmulti	249	1.4363	0.0253	1.0768e+03	0.1680	9.2897e-01
egout	186	1.5107	0.0394	5.4048e+01	0.1225	1.0033e+01
enigma	177	1.2982	0.0314	1.0707e+00	2659.7047	6.1830e+10
fiber	101	0.8853	0.0029	1.1999e+02	0.0425	1.6724e-01
fixnet6	157	1.5521	0.0029	2.7918e+02	0.0650	1.8010e+04
harp2	13	0.5050	0.0013	5.1136e+06	1104863.1141	1.3171e+03
khb05250	189	1.1229	0.0080	8.8522e+03	0.0834	6.1876e-02
l152lav	53	0.7725	0.0038	3.4826e+00	0.0043	5.9479e-01
lp4l	113	0.6465	0.0066	2.4987e+00	0.0137	3.0472e+00
lseu	244	0.9115	0.0172	1.2540e+02	0.1586	9.5616e+06
markshare1	250	0.6887	0.0046	1.7080e+00	0.0293	1.0645e+08
markshare2	250	0.6760	0.0042	1.6842e+00	0.0246	1.6153e+08
mas74	250	0.4781	0.0194	3.8798e+03	0.0264	8.0416e+05
mas76	250	0.4632	0.0205	5.6542e+03	0.0307	1.2407e+04
mkc	13	0.8572	0.0012	2.9729e+00	0.2493	9.4849e+02
mod008	249	0.4398	0.0050	1.0156e+00	0.0632	1.9374e+06
mod010	41	0.7371	0.0025	4.6376e+00	0.0054	2.8961e-01
mod013	262	1.5907	0.0363	2.8014e+01	0.1080	1.1511e+01
modglob	250	1.5240	0.0209	1.4647e+05	0.6094	3.9595e-01
p0033	250	0.9288	0.0324	1.7015e+02	0.0815	1.6159e+03
p0040	85	1.4318	0.0328	6.1046e+03	0.2676	2.9040e+00
p0201	250	0.8442	0.0150	9.2444e+00	0.0715	3.5316e+00
p0282	127	0.9354	0.0133	4.2222e+02	0.0538	4.2091e-01
p0548	155	0.5579	0.0037	1.8385e+03	0.0034	2.8034e+04
p2756	41	0.3750	0.0000	3.2609e+02	0.0002	9.4207e+01
pipex	249	1.3674	0.0434	4.4455e+00	0.0629	3.7596e-01
pk1	250	1.0114	0.0691	4.2040e+00	0.0249	1.8538e+04
pp08a	250	1.6394	0.0292	2.5021e+02	0.0927	2.3258e+01
pp08aCUTS	250	1.5730	0.0281	3.7943e+02	0.1025	3.5594e+01
qiu	61	1.5613	0.0530	1.9504e+01	0.0491	3.0249e+01

Table B.7 Average of data for Centered Iterate Method 2 on the MIPLIB problem set.

PROB	NODES	ws/cs	sw/n	$\|\|x^o_{ws} - x^*_{ws}\|\|$	$\|\|r^0_b\|\|/(\|\|b\|\|+1)$	$\|\|(r^0_b, r^0_c)\|\|/\mu$
rentacar	5	1.9512	0.0196	6.7513e+04	0.2181	5.1835e+01
rgn	250	1.4223	0.0302	3.1935e+00	0.2645	3.3626e+04
sample2	234	2.1293	0.0553	3.8754e+01	0.2273	1.8794e+01
sentoy	278	0.7769	0.0366	3.0564e+03	0.0370	1.5166e+02
set1ch	210	1.5754	0.0049	6.8011e+02	0.0391	1.1804e+05
stein9	60	1.7006	0.1323	1.5085e+00	0.3609	2.8272e+01
stein15	293	1.6706	0.1024	2.1423e+00	0.2511	5.9831e+02
stein27	250	1.3757	0.0904	3.0867e+00	0.1460	7.7265e+02
stein45	125	1.3750	0.0594	4.1426e+00	0.1012	3.4153e+02
vpm1	250	1.6462	0.0095	1.7055e+02	0.0003	2.1305e+00
vpm2	160	1.4733	0.0067	1.0188e+02	0.0002	7.8387e-01

Table B.8 Average of data for Centered Iterate Method 2 on the MIPLIB problem set, continued.

PROB	NODES	ws/cs	sw/n	$\|\|x^o_{ws} - x^*_{ws}\|\|$	$\|\|r^0_b\|\|/(\|\|b\|\|+1)$	$\|\|(r^0_b, r^0_c)\|\|/\mu$
cap41	17	0.7922	0.0080	3.6933e+03	0.1875	6.9124e+02
cap42	17	0.9150	0.0080	4.2175e+03	0.2167	6.0709e+02
cap43	17	0.8156	0.0080	4.1341e+03	0.2122	7.4458e+02
cap44	17	0.8780	0.0080	4.0431e+03	0.2060	9.4732e+02

Table B.9 Average of data for Total Relative Error Method on the Capacitated Facility Location problem set

PROB	NODES	ws/cs	sw/n	$\|\|x^o_{ws} - x^*_{ws}\|\|$	$\|\|r^0_b\|\|/(\|\|b\|\|+1)$	$\|\|(r^0_b, r^0_c)\|\|/\mu$
cap41	20	0.4116	0.0113	3.5896e+03	0.1433	1.4859e+01
cap42	19	0.3361	0.0118	3.7024e+03	0.1489	7.8840e-01
cap43	19	0.3900	0.0118	3.0650e+03	0.1424	6.0050e+00
cap44	20	0.3845	0.0113	3.4624e+03	0.1440	2.1140e+00

Table B.10 Average of data for Early Termination Method on the Capacitated Facility Location problem set

PROB	NODES	ws/cs	sw/n	$\|\|x^o_{ws} - x^*_{ws}\|\|$	$\|\|r^0_b\|\|/(\|\|b\|\|+1)$	$\|\|(r^0_b, r^0_c)\|\|/\mu$
cap41	20	0.3945	0.0113	6.4021e+03	0.1426	1.9126e-03
cap42	20	0.3869	0.0113	6.4024e+03	0.1426	1.9105e-03
cap43	20	0.3880	0.0113	6.4028e+03	0.1426	1.9084e-03
cap44	20	0.4013	0.0113	6.4033e+03	0.1426	1.9053e-03

Table B.11 Average of data for Centered Iterate Method 1 on the Capacitated Facility Location problem set

PROB	NODES	ws/cs	sw/n	$\|\|x^o_{ws} - x^*_{ws}\|\|$	$\|\|r^0_b\|\|/(\|\|b\|\|+1)$	$\|\|(r^0_b, r^0_c)\|\|/\mu$
cap41	20	0.4055	0.0113	6.4112e+03	0.1428	1.9103e-03
cap42	20	0.3952	0.0113	6.4114e+03	0.1428	1.9086e-03
cap43	20	0.3960	0.0113	6.4116e+03	0.1428	1.9069e-03
cap44	20	0.4256	0.0113	6.4053e+03	0.1426	1.9017e-03

Table B.12 Average of data for Centered Iterate Method 2 on the Capacitated Facility Location problem set

PROB	NODES	ws/cs	sw/n	$\|\|x^o_{ws} - x^*_{ws}\|\|$	$\|\|r^0_b\|\|/(\|\|b\|\|+1)$	$\|\|(r^0_b, r^0_c)\|\|/\mu$
con33	17	0.5778	0.0009	5.1540e+02	0.1530	6.1542e+02

Table B.13 Average of data for Total Relative Error Method on *con33*

PROB	NODES	ws/cs	sw/n	$\|\|x^o_{ws} - x^*_{ws}\|\|$	$\|\|r^0_b\|\|/(\|\|b\|\|+1)$	$\|\|(r^0_b, r^0_c)\|\|/\mu$
con33	17	0.8337	0.0009	2.8176e+01	0.1590	8.6409e+05

Table B.14 Average of data for Early Termination Method on *con33*

PROB	NODES	ws/cs	sw/n	$\|\|x^o_{ws} - x^*_{ws}\|\|$	$\|\|r^0_b\|\|/(\|\|b\|\|+1)$	$\|\|(r^0_b, r^0_c)\|\|/\mu$
con33	17	1.3312	0.0009	8.0927e+01	0.1058	4.5034e+02

Table B.15 Average of data for Centered Iterate Method 1 on *con33*

PROB	NODES	ws/cs	sw/n	$\|\|x^o_{ws} - x^*_{ws}\|\|$	$\|\|r^0_b\|\|/(\|\|b\|\|+1)$	$\|\|(r^0_b, r^0_c)\|\|/\mu$
con33	16	1.3254	0.0009	8.1500e+01	0.1061	4.5107e+02

Table B.16 Average of data for Centered Iterate Method 2 on *con33*

Appendix C

Expected Iterations Count Data

The following tables report the average expected iteration counts versus the average warm-start iterations for each problem set. The second and third columns report the average number of cold iterations and the average number of warm iterations, respectively, for each problem in column one. The final column gives the average expected number of iterations.

PROB	NODES	cold	warm	expected
10teams	64	10.8438	26.1562	5.6250
air01	6	11.5000	7.6667	5.1667
air02	16	17.2500	19.7500	9.7500
air04	2	28.5000	14.0000	13.5000
air05	7	26.7143	28.0000	13.2857
bm23	252	10.4921	11.1071	5.9722
cap6000	8	30.7500	23.8750	10.6250
cracpb1	74	10.7703	3.4459	4.0270
danoint	57	19.6316	49.1579	11.8070
dcmulti	203	16.9507	36.3054	7.0739
egout	176	10.2898	12.5284	4.7557
enigma	68	10.5441	7.5588	6.5294
fiber	5	14.8000	36.0000	6.8000
fixnet6	20	15.2000	21.7000	5.1000
harp2	1	73.0000	40.0000	26.0000
khb05250	86	n/a	n/a	n/a
l152lav	13	19.5385	20.5385	7.4615
lp4l	113	16.4425	14.2743	7.4602
lseu	47	12.7660	14.3617	5.7660
markshare1	250	5.0160	2.7120	2.6120
markshare2	250	5.0360	2.6960	2.6480
mas74	250	19.1040	20.1400	4.9760
mas76	250	21.1280	20.4440	4.7120
mkc	13	19.9231	9.0000	4.6154
mod008	249	19.9880	11.6426	5.8876
mod010	41	18.7317	13.2195	6.7073
mod013	256	10.9219	12.2305	5.2539
modglob	165	17.2424	32.1030	9.9212
p0033	239	11.6904	14.3013	6.7908
p0040	79	10.7722	12.4937	3.1139
p0201	31	13.0645	31.2258	5.3548
p0282	91	15.1099	36.5165	5.7473
p0548	64	22.2031	6.4844	3.4062
p2756	42	16.0000	2.1190	2.0714
pipex	212	8.5094	10.4009	5.5236
pk1	250	7.8920	8.3240	3.3960
pp08a	226	12.8186	19.1726	7.1549
pp08aCUTS	239	13.9456	21.3891	7.5481
qiu	61	22.3279	24.4098	14.8033

Table C.1 Total Relative Error Method expected iteration counts for the MIPLIB problem set

PROB	NODES	cold	warm	expected
rentacar	1	24.0000	29.0000	4.0000
rgn	240	9.8958	18.6417	5.1667
sample2	236	9.6864	12.9661	4.8729
sentoy	278	12.5899	14.4424	7.2626
set1ch	71	16.6901	23.1690	7.6197
stein9	60	8.4333	5.8167	4.2667
stein15	292	8.8973	8.4075	4.8425
stein27	250	10.9840	11.1680	5.4640
stein45	126	11.5397	13.1508	5.4048
vpm1	250	11.6160	9.6640	4.1680
vpm2	161	15.2484	25.5776	5.6273

Table C.2 Total Relative Error Method expected iteration counts for the MIPLIB problem set, continued

PROB	NODES	cold	warm	expected
10teams	31	10.9677	6.0323	3.8065
air01	4	11.7500	11.7500	5.5000
air02	16	17.2500	21.6250	9.3125
air04	2	28.5000	14.0000	13.5000
air05	7	26.7143	28.7143	12.8571
bm23	252	10.4921	14.6071	5.8571
cap6000	8	30.7500	8.6250	10.1250
cracpb1	74	10.7703	3.4324	4.0405
danoint	57	19.6316	52.2281	11.3333
dcmulti	242	16.7810	16.3017	9.6281
egout	181	10.2873	12.7127	4.8122
enigma	101	9.1485	24.6040	5.3663
fiber	5	14.8000	9.6000	7.0000
fixnet6	20	15.2000	17.1500	6.2500
harp2	1	73.0000	40.0000	26.0000
khb05250	170	19.2176	41.5882	12.4941
l152lav	53	19.7736	23.2453	7.2075
lp4l	40	16.0000	16.6250	6.1500
lseu	239	10.7782	12.4184	6.5021
markshare1	249	4.9880	3.0522	2.6225
markshare2	250	5.0360	2.9520	2.6640
mas74	250	19.1040	17.9120	4.4280
mas76	250	21.1280	14.8200	4.2000
mkc	13	19.9231	8.4615	7.5385
mod008	249	19.9880	13.3293	5.8594
mod010	41	18.7317	13.3171	5.8293
mod013	248	10.9718	10.4798	5.2056
modglob	242	17.5620	23.2810	11.2438
p0033	231	11.6883	13.7186	6.3766
p0040	79	10.7848	9.5443	3.8608
p0201	163	12.2638	9.7239	4.9325
p0282	127	15.0709	19.8504	6.7559
p0548	64	22.2031	8.3906	4.7344
p2756	42	16.0000	2.2857	2.2381
pipex	138	8.6159	10.5217	5.5942
pk1	250	7.8920	10.2240	3.2280
pp08a	234	12.8248	16.1752	7.1966
pp08aCUTS	250	13.8240	17.9560	7.6720
qiu	61	22.3279	17.6393	14.8033

Table C.3 Early Termination expected iteration counts for the MIPLIB problem set

PROB	NODES	cold	warm	expected
rentacar	17	28.5294	98.0000	17.1176
rgn	239	9.8996	19.8494	5.2092
sample2	238	9.6849	14.0966	4.9076
sentoy	278	12.5899	11.9173	6.7230
set1ch	124	16.2016	12.3065	7.8387
stein9	42	8.8333	6.7857	4.5714
stein15	259	8.9807	9.1969	5.0193
stein27	249	10.9920	11.4779	5.3052
stein45	126	11.5397	15.4286	5.1270
vpm1	213	11.5540	12.8263	3.9484
vpm2	22	14.3182	32.5909	7.6818

Table C.4 Early Termination expected iteration counts for the MIPLIB problem set, continued

PROB	NODES	cold	warm	expected
10teams	80	10.6625	7.6375	6.5500
air01	6	11.5000	16.0000	7.3333
air02	15	17.0000	14.8000	9.9333
air04	2	28.5000	35.0000	23.5000
air05	7	26.7143	22.4286	20.0000
bm23	252	10.4921	8.6468	5.7381
cap6000	8	30.7500	34.8750	14.7500
cracpb1	179	14.0391	11.4581	5.1564
danoint	57	19.6316	38.0000	13.2632
dcmulti	249	16.7590	24.2129	11.1044
egout	186	10.2849	15.3710	5.3602
enigma	183	9.0929	11.3607	5.2077
fiber	100	15.8200	11.8000	7.5200
fixnet6	156	14.5449	22.5769	5.0128
harp2	13	56.0000	27.5385	32.9231
khb05250	187	19.1765	21.6096	11.7380
l152lav	53	19.7736	14.9057	13.7358
lp4l	111	16.4685	10.2973	8.9099
lseu	244	10.7459	9.4918	5.9098
markshare1	250	5.0160	3.3200	2.6280
markshare2	250	5.0360	3.2960	2.6480
mas74	250	19.1040	8.9960	6.9280
mas76	250	21.1280	9.2800	7.5280
mkc	13	19.9231	16.7692	5.0000
mod008	249	19.9880	8.7229	6.5221
mod010	40	18.7500	13.7750	11.7500
mod013	262	10.9237	17.1870	5.6221
modglob	250	17.5640	26.3320	16.8600
p0033	250	11.6280	10.6680	7.4320
p0040	85	10.7059	15.3176	6.7529
p0201	250	11.6840	9.9080	4.6160
p0282	126	15.0714	12.5159	8.2540
p0548	153	19.8954	11.0850	6.0850
p2756	41	16.0000	6.0000	4.0000
pipex	249	8.4859	11.4056	5.8514
pk1	250	7.8920	7.7880	3.4560
pp08a	250	12.7520	20.7800	7.4200
pp08aCUTS	250	13.8240	21.4160	8.0600
qiu	42	24.3571	31.2857	15.5000

Table C.5 Centered Iterate Method 1 expected iteration counts for the MIPLIB problem set

PROB	NODES	cold	warm	expected
rentacar	21	29.6667	46.5714	14.7143
rgn	250	9.8480	14.0080	5.1840
sample2	239	9.6778	20.5439	5.1004
sentoy	278	12.5899	9.5755	8.8597
set1ch	208	15.1538	23.9760	7.3942
stein9	60	8.4333	13.1667	4.2167
stein15	293	8.8942	14.4642	4.6075
stein27	250	10.9840	14.7480	4.9200
stein45	124	11.5323	15.5323	4.9355
vpm1	250	11.6160	19.0840	4.5720
vpm2	160	15.2438	21.7437	7.0875

Table C.6 Centered Iterate Method 1 expected iteration counts for the MIPLIB problem set, continued

PROB	NODES	cold	warm	expected
10teams	83	10.8554	7.8072	6.5904
air01	6	11.5000	15.3333	7.5000
air02	16	17.2500	24.2500	10.4375
air04	2	22.5000	22.0000	11.5000
air05	7	26.7143	21.4286	19.8571
bm23	252	10.4921	9.0357	5.7579
cap6000	8	30.7500	35.6250	14.7500
cracpb1	181	14.0055	12.5580	5.1602
danoint	57	19.6316	39.1228	13.2456
dcmulti	249	16.7590	23.9357	11.1767
egout	186	10.2849	15.4677	5.3441
enigma	177	9.1525	11.4633	5.2486
fiber	101	15.8416	13.6337	7.9901
fixnet6	157	14.5478	22.5796	5.0191
harp2	13	56.0000	27.9231	32.9231
khb05250	189	19.1693	21.2593	12.8307
l152lav	53	19.7736	15.3019	13.7736
lp4l	113	16.4425	10.6195	8.8673
lseu	244	10.7459	9.6148	5.9139
markshare1	250	5.0160	3.6120	2.6320
markshare2	250	5.0360	3.5800	2.6520
mas74	250	19.1040	9.1360	6.9280
mas76	250	21.1280	9.5160	7.5200
mkc	13	19.9231	17.0769	5.0000
mod008	249	19.9880	8.6908	6.5181
mod010	41	18.7317	13.8049	11.7317
mod013	262	10.9237	17.3168	5.6221
modglob	250	17.5640	26.5720	16.8600
p0033	250	11.6280	10.8280	7.4320
p0040	85	10.7059	15.3176	6.7529
p0201	250	11.6840	9.9480	4.6360
p0282	127	15.0709	14.1811	8.2756
p0548	155	19.8581	11.0903	6.0645
p2756	41	16.0000	6.0000	4.0000
pipex	249	8.4859	11.5181	5.8956
pk1	250	7.8920	8.1920	3.4600
pp08a	250	12.7520	20.8720	7.4320
pp08aCUTS	250	13.8240	21.6320	8.0560
qiu	61	22.3279	32.9836	13.2787

Table C.7 Centered Iterate Method 2 expected iteration counts for the MIPLIB problem set

PROB	NODES	cold	warm	expected
rentacar	5	27.0000	54.0000	12.0000
rgn	250	9.8480	14.1640	5.1880
sample2	234	9.6838	20.4487	5.0940
sentoy	278	12.5899	9.7374	8.8669
set1ch	210	15.1714	23.9667	7.4190
stein9	60	8.4333	13.2333	4.2167
stein15	293	8.8942	14.5427	4.6177
stein27	250	10.9840	14.9800	4.9280
stein45	125	11.5520	15.6720	4.9440
vpm1	250	11.6160	19.1120	4.6360
vpm2	160	15.2438	22.2000	6.9750

Table C.8 Centered Iterate Method 2 expected iteration counts for the MIPLIB problem set, continued

PROB	NODES	cold	warm	expected
cap41	17	59.5882	47.1765	12.2941
cap42	17	60.9412	55.7059	16.4706
cap43	17	59.9412	49.0000	16.5294
cap44	17	58.0000	51.0588	15.4118

Table C.9 Total Relative Error Method expected iteration counts for the Capacitated Facility Location problem set

PROB	NODES	cold	warm	expected
cap41	20	60.3500	24.0500	23.5000
cap42	20	62.0500	24.0000	25.0500
cap43	20	60.6500	23.6500	25.6000
cap44	20	59.0500	23.6500	26.4000

Table C.10 Centered Iterate Method 1 expected iteration counts for the Capacitated Facility Location problem set

PROB	NODES	cold	warm	expected
cap41	20	60.3500	24.6500	23.6000
cap42	20	62.0500	24.4000	25.0500
cap43	20	60.6500	24.2000	25.6000
cap44	20	59.0500	25.0000	26.4000

Table C.11 Centered Iterate Method 2 expected iteration counts for the Capacitated Facility Location problem set

PROB	NODES	cold	warm	expected
cap41	20	60.3500	25.2500	11.4500
cap42	19	62.2105	21.2632	12.5789
cap43	19	60.6316	24.0000	10.7895
cap44	20	59.0500	22.7500	12.9500

Table C.12 Early Termination expected iteration counts for the Capacitated Facility Location problem set

PROB	NODES	cold	warm	expected
con33	17	26.1765	15.1765	14.7647

Table C.13 Total Relative Error Method expected iteration counts for *con33*

PROB	NODES	cold	warm	expected
con33	17	26.1765	34.8235	10.8824

Table C.14 Centered Iterate Method 1 expected iteration counts for *con33*

PROB	NODES	cold	warm	expected
con33	16	26.1875	34.6875	10.8750

Table C.15 Centered Iterate Method 2 expected iteration counts for *con33*

PROB	NODES	cold	warm	expected
con33	17	26.1765	21.6471	7.0588

Table C.16 Early Termination expected iteration counts for *con33*

Bibliography

[1] *Using the CPLEX Callable Library.* CPLEX Optimization, 1989-1995.

[2] E.D. Andersen, J. Gondzio, C. Mézáros, X. Xu. "Implementaton of a Interior Point Methods for Large Scale Linear Programming." Logilab Technical Report 1996.3, Logilab, HEC Geneva, Section of Management Studies, University of Geneva, 1996.

[3] A. A. Assad. "Models for Rail Transportation." *Transportation Research A*, 14A: 205-220.

[4] J.E. Beasley. "OR-Library: Distributing Test Problems by Electronic Mail." *Journal of the Operational Research Society'*, 41(11): 1069-1072, 1990.

[5] R.E. Bixby, S. Ceria, C.M. McZeal, and M.W.P Savelsbergh. "An Updated Mixed Integer Programming Library: MIPLIB 3.0." Technical Report TR 98-03, Department of Computational and Applied Mathematics, Rice University, 1998.

[6] B. Borchers. "Improved Branch and Bound Algorithms for Integer Programming." PhD Thesis, Department of Mathematical Sciences, Rensselaer Polytechnic Institute, 1992.

[7] B. Borchers and J.E. Mitchell. "Using an Interior Point Method in a Branch and Bound Algorithm for Integer Programming." R.P.I Technical Report No. 195, Department of Mathematical Sciences, Rensselaer Polytechnic Institute, 1992.

[8] V. Chvátal. *Linear Programming.* W. H. Freeman and Company, 1983.

[9] L. W. Clarke, C. A. Hane, E. L. Johnson, and G. L. Nemhauser. "Maintenance and Crew Considerations in Fleet Assignment." *Transportation science* 30(3): 249-260, 1996.

[10] Conrail Homepage. `http://www.conrail.com`

[11] G. B. Dantzig. *Linear Programming and Extensions'*. Princeton University Press, 1963.

[12] A. Duarte, R.J. Vanderbei, and B. Yang. "An Algorithmic and Numerical Comparison of Several Interior Point Methods." Technical Report SOR-94-05, Program in Statistics and Operations Research, Princeton University, Princeton, 1994.

[13] 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.

[14] A.S. El-Bakry, R.A. Tapia, and Y. Zhang. "The Logarithmic Tapia Indicators and the Identification of Subgroups of Variables in Interior Point Methods." Technical Report manuscript, Computational and Applied Mathematics Department, Rice University, 1991.

[15] J.E. Dennis, Jr. and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Society for Industrial and Applied Mathematics, Philadelphia, 1996.

[16] R. M. Freund. "Theoretical Efficiency of a Shifted-Barrier-Function Algorithm for Linear Programming." *Linear Algebra and Its Applications*, 152:19-41, 1991.

[17] R. M. Freund. "A Potential-Function Reduction Algorithm for Solving a Linear Program Directly from and Infeasible "Warm Start"." *Mathematical Programming*, 52(3): 441-466, 1991.

[18] D. M. Gay. "Electronic Mail Distribution of Linear Programming Test Problems." *Mathematical Programming Society COAL Newsletter*, 1985.

[19] J-L Goffin and J-P Vial. "Shallow, Deep, and Very Deep Cuts in the Analytic Center Cutting Plane Method." Logilab Technical Report 1996.1, Logilab, HEC Geneva, Section of Management Studies, University of Geneva, 1997.

[20] A. Goldman and A. Tucker. Theory of Linear Programming. In H.W. Kuhn and A.W. Tucker, eds., *Linear Inequalities and Related Systems,* pages 53-97. Princeton University Press, Princeton, NJ, 1956.

[21] J. Gondzio. "HOPDM (version 2.12)- A fast LP solver based on a primal-dual interior point method." *European Journal of Operational Research.* 85:221-225, 1995.

[22] J. Gondzio. "Multiple Centrality Corrections in a Primal-Dual Method for Linear Programming." Logilab Technical Report 1994.20, Logilab, HEC Geneva, Section of Management Studies, University of Geneva, 1995.

[23] J. Gondzio. "Warm Start of the Primal-Dual Method Applied in the Cutting Plane Scheme." Logilab Technical Report 96.3, Logilab, HEC Geneva, Section of Management Studies, University of Geneva, 1996.

[24] J. Gondzio and R. Sarkissian. "Column Generation with the Primal Dual Method." Logilab Technical Report 1995.34, Logilab, HEC Geneva, Section of Management Studies, University of Geneva, 1996.

[25] J. Gondzio and Tamás Terlaky. "A Computational View of Interior Point Methods." In J.E. Beasley (ed.) *Advances in Linear and Integer Programming.* Oxford University Press, Oxford, England, 1996.

[26] J. Gondzio and J-P Vial. "Warm Start and ϵ-subgradients in Cutting Plane Scheme for Block-Angular Linear Programs." Logilab Technical Report 1997.1, Logilab, HEC Geneva, Section of Management Studies, University of Geneva, 1997.

[27] M. D. Gonzàlez-Lima. "Effective Computation of the Analytic-Center of the Solution Set in Linear-Programming Using Primal-Dual Interior-Point Methods." Technical Report TR94-48, Department of Computational and Applied Mathematics, Rice University, 1994.

[28] M. D. Gonzàlez-Lima, A. S. El-Bakry, and R. A. Tapia. "The Computational Role of Proximity Measures in Computing Analytic Centers." Technical Report to appear, Department of Computational and Applied Mathematics, Rice University, 1999.

[29] Z. Gu, E. L. Johnson, G. L. Nemhauser, M.W.P. Savelsbergh. "Progress in Integer Programming: An Exposition." School of Industrial and Systems Engineering, Georgia Institute of Technology, 1997.

[30] Z. Gu, E. L. Johnson, G. L. Nemhauser, Y. Wang. "Some properties of the fleet assignment problem." *Operations Research Letters.*15(2): 59-71, 1994.

[31] C. A. Hane, C. Barnhart, E. L. Johnson, R. E. Marsten, G. L. Nemhauser, G. Sigismondi. "The fleet assignment problem: Solving a large-scale integer program." *Mathematical Programming* 70(2): 211-232, 1995.

[32] D. L. Jensen and R. A. Polyak. "Experience with Modified Barrier Function Methods for Linear Programming." Department of Operations Research, School of Information Technology and Engineering, George Mason University.

[33] M. Jünger, G. Reinelt and S. Thienel. "Practical Problem Solving with Cutting Plane Algorithms in Combinatorial Optimization." *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 20:111-152, 1995.

[34] N. Karmarkar. A New Polynomial-Time Algorithm for linear programming. *Combinatorica* 4: 373-395, 1984.

[35] M. Kojima, N. Megiddo, and S. Mizuno. "A primal-dual infeasible interior-point algorithm for linear programming." *Mathematical Programming* 61(3):263-280, 1993.

[36] M. Kojima, S. Mizuno, and A. Yoshise. "A primal-dual interior-point method for linear programming." *Progress in Mathematical Programming, Interior-Point and Related Methods,* N. Megiddo, editor. Springer-Verlag, New York, 1989.

[37] A.H. Land and A.G. Doig. "An Automatic Method for Solving Discrete Programming Problems." *Econometrica*, 28:497-520, 1960.

[38] E.K. Lee and J.E. Mitchell. "Computational Experience of an Interior Point Algorithm in a Parallel Branch-and-Cut Framework."

[39] C. E. Lemke. "The Dual Method of Solving the Linear Programming Problem." *Naval Research Logistics Quarterly 1.* 36-47, 1954.

[40] M. Luo. Conrail. Personal Communication.

[41] I.J. Lustig, R.E. Marsten, and D.F. Shanno. "Interior Point Methods for Linear Programming: Computational State of the Art." *ORSA Journal on Computing*, 6(1), 1994.

[42] I.J. Lustig, R.E. Marsten, and D.F. Shanno. "On Implementing Mehrotra's Predictor-Corrector Interior-Point Method for Linear Programming." *SIAM Journal of Optimization*, 3(3):435-449, 1992.

[43] I.J. Lustig, R.E. Marsten, and D.F. Shanno. "Computational Experience with a Globally Convergent Primal-Dual Algorithm for Linear Programming." *Mathematical Programming*, 66(1):123-135, 1994.

[44] I.J. Lustig, R.E. Marsten, and D.F. Shanno. "Starting and Restarting the Primal-Dual Interior Point Method." Technical Report SOR 90-14, Department of Civil Engineering and Operations Research, Princeton University, 1990.

[45] A. S. Manne. "DINAMICO, A Dynamic Multi-Section Multi-Skill Model." In L. M. Goreux and A. S. Manne (eds.), *Multi-Level Planning, Case Studies in Mexico*, North Holland Publishing Co., Amsterdam, 1973.

[46] A. S. Manne. "Economic Alternatives for Mexico, A Quantitative Analysis." In L. M. Goreux and A. S. Manne (eds.), *Multi-Level Planning, Case Studies in Mexico*, North Holland Publishing Co., Amsterdam, 1973.

[47] S. Mehrotra. "On the Implementation if a (Primal-Dual) Interior Point Method." Technical Report 90-03, Department of Industrial Engineering and Management Sciences, Northwestern University, 1990.

[48] M. Meketon. US Airways, Inc. (formerly of Conrail). Personal communication.

[49] J.E. Mitchell. Interior Point Algorithms for Integer Programming. In J.E. Beasley (ed.) *Advances in Linear and Integer Programming.* Oxford University Press, Oxford, England, 1996..

[50] J.E. Mitchell. "Fixing Variables and Generating Classical Cutting Planes When Using an Interior Point Branch and Cut Method to Solve Integer Programming Problems." R.P.I. Technical Report No. 216, Department of Mathematical Sciences, Rensselaer Polytechnic Institute, 1994.

[51] J.E. Mitchell. "Interior Point Methods for Combinatorial Optimization." R.P.I. Technical Report No. 217, Department of Mathematical Sciences, Rensselaer Polytechnic Institute, 1995.

[52] G.L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley & Sons, New York, NY,1988.

[53] R. Polyak. "Modified Barrier Functions (Theory and Methods)." *Mathematical Programming.* 54(2):177-222, 1992.

[54] R. Polyak. "The Nonlinear Rescaling Principle in Linear Programming." Technical Report RC 15030 (67093), Mathematical Sciences Department, IBM T.J. Watson Research Center, Yorktown Heights, NY, 10598, 1989.

[55] D. F. Shanno. "Computational Methods for Linear Programming." RUTCOR Research Report 19-93, Rutgers Center for Operations Research, Rutgers University, 1993.

[56] R. Subramanian, R. P. Scheff, J. D. Quillinan, S. D. Wiper, and R. E. Marsten. "Coldstart: Fleet Assignment at Delta Air Lines." *Interfaces* 24(1): 104-120, 1994.

[57] R.A. Tapia, Y. Zhang, M. Saltzman, and A. Weiser. "The Predictor-Corrector Interior-Point Method as a Composite Newton Method." Technical Repoint TR90-17, Department of Computational and Applied Mathematics, Rice University, Houston, Texas, 1990.

[58] K. T. Talluri. "Swapping Applications in a Daily Airline Fleet Assignment." *Transportation science* 30(3):237-248, 1996.

[59] R. J. Vanderbei. *Linear Programming Foundations and Extensions.* Kluwer Academic Publishers, Boston, MA, 1996.

[60] T. A. S. Vijayaraghavan, K. M. Anantharamaiah. "Fleet assignment strategies in urban transportation using express and partial services." *Transportation research. Part A, Policy and practice.* 29(2): 157-171, 1995.

[61] P.J. Williams. "Effective Finite Termination Procedures in Interior-Point Methods for Linear Programming." Technical Report TR 98-17, Department of Computational and Applied Mathematics, Rice University, 1998.

[62] S. Wright. *Primal Dual Interior Point Methods.* SIAM, Philadelphia, Pennsylvania, 1997.

[63] Y. Zhang. "ON the Convergence of a class of Infeasible Interior-Point Methds for the Horizontal Linear Complementarity Problems." *SIAM J. Optimization*, 4:208-227, 1994.

[64] Y. Zhang. " Solving Large-Scale Linear Programs by Interior Point Methods Under the MATLAB Environment." Technical Report TR96-01, Department of Mathematics and Statistics, University of Maryland Baltimore County, 1996.

[65] Y. Zhang. "User's Guide to LIPSOL, Linear-programming Interior Point Solvers v0.3." Department of Mathematics and Statistics, University of Maryland Baltimore County, 1995.

[66] Y. Zhang, R.A. Tapia, and J.E. Dennis. "On The Superlinear Convergence of Primal-Dual Interior Point Linear Programming Algorithms." *SIAM J. Optimization*, 2(2): 304-324, 1992.