Interior Point Methods and Linear Programming

Robert Robere
University of Toronto

December 13, 2012

Abstract

The linear programming problem is usually solved through the use of one of two algorithms: either simplex, or an algorithm in the family of interior point methods. In this article two representative members of the family of interior point methods are introduced and studied. We discuss the design of these interior point methods on a high level, and compare them to both the simplex algorithm and the original algorithms in nonlinear constrained optimization which led to their genesis.

1 Introduction

Linear programs are ubiquitous in many areas of applied science today. The primary reason for this is their flexibility: linear programs frame problems in optimization as a system of linear inequalities. This template is general enough to express many different problems in engineering, operations research, economics, and even more abstract mathematical areas such as combinatorics. From the time of Dantzig's original example of finding the best assignment of 70 people to 70 jobs [Dan47], the interest in linear programs have exploded.

Owing to their vast applicability, there has been much interest in finding efficient algorithms which find the best solutions to linear programs. The very first algorithm used for linear programming was the simplex algorithm, also by Dantzig [Dan47]. In the 60 years of research since the introduction of simplex, this algorithm has been carefully optimized to perform extremely well in practice. However, a problem arose in the 1970s: it turns out that we *cannot* guarantee that simplex will work well on all possible linear programs [KM70]. This problem led to the introduction of the *interior point methods* for solving linear programs, which is the focus of this paper. The interior point methods are a family of algorithms solving linear programs which come along with an efficient performance guarantee.

In this article, we introduce and discuss two of these interior point methods. The aim is to give an overview: by the end of the paper, it is our wish that reader will have a firm, intuitive grasp of how interior point methods work and enough pointers to the literature to be able to further his or her knowledge.

The rest of this paper is organized as follows. In the next section we formalize the linear programming problem, and recall some notions from constrained optimization that

will help understand (at least on an intuitive level) how and why the interior point methods converge. In Section 3 we introduce and discuss the simplex algorithm, so the reader will have an idea of how the major competitor to the interior point methods differs in form. In Section 4 we describe two interior point algorithms — the first of which is equivalent to the original interior point method for linear programming by Karmarkar [Kar84], and the second of which underlies the efficient algorithms used for solving large scale linear programs in industry today. We then close with a discussion and a pointer to further literature.

2 Linear Programming Problem

In this section we introduce the problem at the center of this study: the *linear programming problem* (LPP). We will be brief in our description of the problem — the theory behind linear programs is vast, and it is all to easy to be caught up in the details. To that end, we will focus on aspects of the LPP that will be important in describing the simplex method and the interior point methods we consider.

2.1 Definitions

Definitions can be found in nearly any optimization textbook, and we will follow [NW06, Chapter 13]. A linear program is a constrained optimization problem in which the objective function and each of the constraints are linear. Therefore the problem can always be written in the following form. Let $c = (c_1, c_2, \ldots, c_n)^T \in \mathbb{R}^n$, $b = (b_1, b_2, \ldots, b_m)^T \in \mathbb{R}^m$ be real vectors of constants, and let $A = (a_{ij})$ be an $m \times n$ real matrix. Finally, let $x = (x_1, x_2, \ldots, x_n)$ be a vector of real variables. Then the linear programming problem can be written as

$$\min_{x \in \mathbb{R}^n} c_1x_1 + c_2x_2 + \cdots + c_nx_n,$$

subject to:

$$a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \leq b_i, 1 \leq i \leq m_1$$
$$a_{j1}x_1 + a_{j2}x_2 + \ldots + a_{in}x_n = b_j, m_1 < j \leq m_2$$

where $m = m_1 + m_2$. Note that this template covers all types of linear constraints, since any constraint containing a "$\geq$" can be converted to a constraint using "$\leq$" by a multiplication of -1.

For ease of manipulation, linear programs are often put into the so-called *standard form*. Let A be an $m \times n$ real valued matrix, $c \in \mathbb{R}^n$, and $b \in \mathbb{R}^m$. Then we can write the above problem like so:

$$\min c^T x \text{ subject to: } Ax = b, x \geq 0. \tag{1}$$

A *solution* to the linear programming problem is a vector $x^* \in \mathbb{R}^n$ such that $c^T x^*$ is minimized and the constraints $Ax^* = b, x^* \geq 0$ are each satisfied. Throughout this paper, we will assume that A has full row rank, for otherwise we can just pre-process to remove any dependent rows.

We can put any problem into this form using two simple transformations. First, transform *inequality* constraints of the form

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \leq b$$

into *equality* constraints by adding slack variables. That is, we add a variable u an a new constraint like so:

$$\begin{aligned} a_1x_1 + a_2x_2 + \ldots a_nx_n + u &= b \\ u &\geq 0. \end{aligned}$$

If any of the variables in x are constrained, we can transform them identically by adding a slack variable and then substituting the new sum back into the other equations in the program. Now, if any of the x variables are unconstrained, we just replace them by a *sum* of unconstrained variables like so:

$$\begin{aligned} x^+ - x^- &= x \\ x^+ &\geq 0 \\ x^- &\geq 0, \end{aligned}$$

and then appropriately modify our objective function and other constraints. We will assume throughout that $m < n$, since if $m > n$ then either A contains redundant rows or is an infeasible system of equations, or if $m = n$ then the system $Ax = b$ has a unique solution.

Note that the *feasible region* of points satisfying constraints in the LPP is defined by the intersection of m half-planes, and so it must be a convex subset of $\mathbb{R}^n$ (more specifically, a convex polytope, a generalization of a polygon to higher dimensions). It is also easy to see that the optimum for the problem must lie on the boundary of this feasible region. For consider $f(x) = c^Tx$, our objective function which we are trying to minimize. Then $\nabla f(x) = c$, a constant vector. To minimize $f(x)$ we should move the direction of $-\nabla f(x) = -c$, and so the furthest feasible point in the $-c$ direction would be lying on the boundary of our polytope (in fact, on a vertex of the polytope). Many algorithms take advantage this fact (notably, the simplex algorithm).

2.2 The KKT Conditions and Duality Theory

Now, we will briefly sketch the first order conditions used for ensuring optimality of a solution to the LPP, and also some elements of duality theory which will make the later algorithms much easier to understand.

Suppose we're at a feasible point in an instance of the LPP — how can we determine if that point is an optimal solution? The *Karush-Kuhn-Tucker* conditions can be used to determine just that. More generally, given an objective function $f : \mathbb{R}^n \to \mathbb{R}$ and a set of smooth constraint functions $\{c_i(x) \geq 0 \vee c_i(x) = 0\}_{i=1}^m$, where $c_i : \mathbb{R}^n \to \mathbb{R}$, one can define the Lagrangian

$$\mathcal{L}(x, \lambda) = f(x) - \sum_{i=1}^{m} \lambda_i c_i(x),$$

where $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_m)^T$ is a collection of Lagrange multipliers (see e.g. [NW06, Section 12.3]). Recall that in unconstrained optimization on a function f, we can characterize necessary and sufficient conditions for a point x^* to be a minimum by considering the gradient and Hessian of f. In constrained optimization the Lagrangian is used in a similar way by including information about the constraints. Key to the new characterizations are the Lagrange multipliers in λ, which are used to describe how the gradients of the constraint functions behave at minima of the objective function f.

The Karush-Kuhn-Tucker conditions are used to formalize this idea (see Theorem 12.1 and the beginning of Section 13.1 in [NW06]). We present them as a theorem:

Theorem 1. *Consider the problem of minimizing a smooth objective function $f : \mathbb{R}^n \to \mathbb{R}$ with respect to a collection of constraints $\{c_i(x) \geq 0 \vee c_i(x) = 0\}_{i=1}^m$, where $c_i : \mathbb{R}^n \to \mathbb{R}$ is also a smooth function. Let $\mathcal{L}$ be the Lagrangian associated with this problem. Suppose x^* is a solution of this problem, and that the subset of constraints satisfying $\{c_i(x^*) = 0\}$ has the property that $\{\nabla c_i(x^*)\}$ is linearly independent. Then there is a vector of Lagrange multipliers λ^* for which the following holds:*

1. $\nabla_x \mathcal{L}(x^*, \lambda^*) = 0$
2. *All of the constraints are satisfied at x^*.*
3. $\lambda_i^* \geq 0$
4. *For all $1 \leq i \leq m$, $\lambda_i c_i(x^*) = 0$.*

Let A, b, c be an instance of the LPP as defined above. The Lagrangian associated with this instance is

$$\mathcal{L}(x, \lambda, s) = c^T x - \lambda^T (Ax - b) - s^T x,$$

where $\lambda, s \in \mathbb{R}^N$ are vectors of Lagrange multipliers. In the case of the LPP, the KKT conditions say that a point x is an optimum only if there exist $\lambda, s \in \mathbb{R}^n$ such that the following conditions hold:

$$A^T \lambda + s = c \tag{2a}$$
$$Ax = b \tag{2b}$$
$$x \geq 0 \tag{2c}$$
$$s \geq 0 \tag{2d}$$
$$x_i s_i = 0, 1 \leq i \leq n. \tag{2e}$$

Clearly the reverse direction must also be true. To see this, let (x^*, λ^*, s^*) satisfy the KKT conditions and let x' be any other feasible point. Then

$$c^T x^* = (A^T \lambda^* + s^*)^T x^* = (Ax^*)^T \lambda^* = b^T \lambda^*,$$

and

$$c^T x' = (A^T \lambda^* + s^*)^T x' = (\lambda^*)^T A x' + (x')^T s^* \geq b^T \lambda^* = c^T x^*.$$

We conclude that since x^* satisfies the KKT conditions, it's objective value must be lower than any other points objective value.

We draw attention to the last equation above, which is known as the *complementarity condition*. Later, we will see that precisely one of the differences between the simplex algorithm and the interior point algorithms is their treatment of this condition.

We now define the *dual* to a linear program (we call the original program the *primal*). Basically, given a linear program defined by A, b, c, we can define another linear program like so:

$$\max b^T \lambda \text{ subject to: } A^T \lambda + s = c, s \geq 0. \tag{3}$$

Two things to notice in the above problem. First, the vectors b, c have switched responsibilities — b is responsible for defining the linear objective function and c is used to define the polytope. Second, our use of λ and s is deliberate — by appealing to the KKT conditions one can show that the optimal Lagrange multipliers λ in the primal problem are the optimal variables in the dual problem and vice versa (for a proof of this fact, see [NW06, Theorem 13.1]). In this way the dual problem gives a lot of information about the primal problem, and this extra information is important in the development of linear programming theory.

Now, consider the KKT conditions for the dual problem. The astute reader will notice that they are identical to the KKT conditions to the primal problem. As mentioned we have the following theorem ([NW06, Theorem 13.1]):

Theorem 2. *If either the primal (1) or the dual (3) has a solution, then so does the other, and the objective values are equal. If either problem is unbounded, then the other problem is infeasible (it's feasible set is empty). If a solution exists, then the optimal solution to the dual problem are the Lagrange multipliers to the primal problem and vice versa.*

Given a linear program and it's dual, we can therefore refer to triples of variables (x, λ, s), in which x is the variable of the primal and λ, s are the Lagrange multipliers, or (λ, s) are the variables in the dual and x is the Lagrange multiplier. Utilizing this information is important in the development of the *primal-dual* interior point methods (see Section 4.2).

2.3 Newton Search Directions

We will require one more ingredient to make our description of interior point methods complete: the *Newton search direction*. Definitions and further discussion can be found in [NW06, Chapter 11]. First suppose $f : \mathbb{R}^n \to \mathbb{R}^m$ is a continuously differentiable vector valued function, and write $f(x) = (f_1(x), f_2(x), \cdots, f_m(x))^T$. The *Jacobian* of $f(x)$, denoted $J(x)$, is the $m \times n$ matrix of first derivatives of f:

$$J(x) = (\partial f_i(x)/\partial x_j)_{i,j}, 1 \leq i \leq m, 1 \leq j \leq n.$$

Now suppose we were trying to solve the equation $f(x) = 0$. We can form a method similar to Newton's method for *unconstrained* optimization, using the Jacobian of the matrix in

place of the Hessian. That is, we perform a line search where the *Newton direction* p_N is computed as the solution to the system

$$J(x)p_N = -f(x).$$

For the convergence of this method, we have the following theorem (see [NW06, Theorem 11.2])

Theorem 3. *Suppose that f is continuously differentiable in a convex open set $\mathcal{D} \subset R^n$. Let $x^* \in \mathcal{D}$ be a nondegenerate solution of $f(x) = 0$ (i.e. a solution for which $J(x^*)$ is nonsingular), and let $\{x_k\}$ be the sequence of iterates generated by repeatedly finding and stepping along the Newton direction p_N from an initial point x_0. When $x_k \in \mathcal{D}$ is sufficiently close to x^*, we have*

$$||x_{k+1} - x^*|| \leq o(||x_k - x^*||),$$

so the sequence converges Q-superlinearly to x^*. When f is Lipschitz continuously differentiable near x^*, we have for all x_k sufficiently close to x^* that

$$||x_{k+1} - x^*|| \leq O(||x_k - x^*||^2),$$

so the sequence converges Q-quadratically to x^*.

As we will see, the Newton search direction plays a key role in the definition of the interior point methods.

3 Simplex

No description of interior point methods would be complete without mentioning the simplex algorithm. Introduced by George Dantzig in 1947 [Dan47], simplex is a method for solving linear programs which is based on this principle:

Theorem 4. *Let A, b, c be an instance of the LPP, defining a convex polytope in $\mathbb{R}^n$. Then there exists an optimal solution to this program at one of the vertices of the polytope.*

The simplex algorithm works roughly as follows. We begin with a feasible point at one of the vertices of the polytope. Then we "walk" along the edges of the polytope from vertex to vertex, in such a way that the value of the objective function monotonically decreases at each step. When we reach a point in which the objective value can decrease no more, we are finished. Each step along an edge of the polytope is determined by a "pivoting" operation, the definition of which is key to the performance of the resulting algorithm. If the pivoting operation is not defined carefully then a problem known as "cycling" can occur, where the algorithm gets trapped walking around on the same cycle of vertices without decreasing the value of the objective function.

The simplex algorithm has excellent performance in practice: in fact, it tends to run in time *linear* in the number of constraints of the problem. However, there is a simple

LP known as the "twisted cube" [KM70] for which simplex can run in *exponential* time in the worst case. This troubled researchers at the time of it's appearance — the simplex algorithm had been around for 20 years at that point, and this behaviour tended not to appear in practice. Despite the pathological nature of this input, the complexity of the LPP was thrown into doubt: was there actually a worst-case polynomial time algorithm for this problem? At the time, researchers were searching for a pivoting rule that was strong enough to guarantee polynomial time performance in the worst case, but this is still an open problem. (For an interesting explanation of the excellent performance of simplex in practice, see [ST01]). It turns out that the answer lied beyond simplex, and interior point methods for linear programming were born.

4 Interior Point Methods

Our above question about the complexity of the LPP was answered by Khachiyan in 1979. He demonstrated a worst-case polynomial time algorithm for linear programming dubbed the ellipsoid method [Kha79] in which the algorithm moved across the *interior* of the feasible region, and not along the boundary like simplex. Unfortunately the worst case running time for the ellipsoid method was high: $O(n^6L^2)$, where n is the number of variables in the problem and L is the number of the bits in the input. Moreover, this method tended to approach the worst-case complexity on nearly all inputs, and so the simplex algorithm remained dominant in practice. This algorithm was only partially satisfying to researchers: was there a worst-case polynomial time algorithm for linear programming which had a performance that rivalled the performance of simplex on day-to-day problems?

This question was answered by Karmarkar in 1984. He produced a polynomial-time algorithm — soon called the projective algorithm — for linear programming that ran in much better time: $O(n^{3.5}L^2)$ [Kar84]. Moreover, in practice, the algorithm actually seemed to be reasonably efficient. This algorithm also worked by moving across the interior of the feasible region, but was far more complicated than the ellipsoid method. Luckily, within a year, Gill et. al. [GMS$^+$86] had shown that the path traced by the projective algorithm was the same as the path traced by a simpler algorithm in the family of *barrier methods*.

In this section, we first introduce this simple interior point method — the *primal Newton barrier method*. This method is equivalent to Karmarkar's original projective algorithm, but it captures the form of modern interior point methods well. Using this basic primal method we will then show how to introduce information from the dual problem, yielding a family of interior point methods known as the *primal-dual interior point methods*. We will close with a discussion of the two methods, and how they stand up to the simplex algorithm in practice.

4.1 The Primal Newton Barrier Method

As stated above, Karmarkar's original algorithm is actually equivalent to a simple algorithm in the family of barrier methods. These algorithms were often used in the 60s for general

nonlinear constrained programming, although by the time of Karmarkar they had generally fallen out of use. Indeed, the paper by Gill et. al. revitalized interest in the barrier methods for both linear and nonlinear programming, and gave rise to the form that most interior point methods are presented in today. We will straddle the presentations of this method in [Wri05] and [GMS+86], giving pause for further explanation where necessary.

Central to the idea of a barrier method is the *barrier function*. Consider an instance of the LPP in standard form:

$$\min c^T x \text{ subject to: } Ax = b, x \geq 0,$$

where c, A, b are defined as usual. We associate a barrier function $B(x, \mu)$ (called the *logarithmic* barrier function) defined as

$$B(x, \mu) = c^T x - \mu \sum_{i=1}^{n} \ln x_j,$$

for real $\mu > 0$. We then change our focus — instead of considering the original LPP, we consider the problem of minimizing the above barrier function subject to $Ax = b$:

$$\min_{x \in \mathbb{R}^n} B(x, \mu) \text{ subject to: } Ax = b. \tag{4}$$

(Due to the existence of the new logarithmic term, we can drop the $x \geq 0$ constraint.) Associated with this new "barrier problem" is a Lagrangian defined by

$$\mathcal{L}(x, \lambda, s) = c^T x - \mu \sum_{i=1}^{n} \ln x_j - \lambda^T (Ax - b),$$

where $\lambda \in \mathbb{R}^m$ is the Lagrange multiplier for the $Ax = b$ constraint. Let $g(x) = \nabla_x \mathcal{L}(x, \mu) = c - A^T \lambda - \mu X^{-1} e$ where $X = diag(x_1, x_2, \ldots, x_n)$ and $e = (1, 1, \ldots, 1)^T$. Writing the KKT conditions for this problem using Theorem 1, at an optimal solution x there exists a vector λ such that:

$$\begin{aligned} A^T \lambda + \mu X^{-1} e &= c \\ Ax &= b \end{aligned}$$

Setting $s = \mu X^{-1} e$, we recover (nearly) the original KKT conditions for the linear programming problem. Substituting in this new definition of s yields

$$A^T \lambda + s = c \tag{5a}$$

$$Ax = b \tag{5b}$$

$$XSe = \mu e, \tag{5c}$$

where $S = diag(s_1, s_2, \ldots, s_n)$. Observe that these conditions for the optimality of a solution to this new barrier subproblem is identical to the conditions for the original LPP, except the

complementarity condition is relaxed by the barrier constant μ. In effect, this shows that as we drive μ to 0, the optimal solution to this barrier subproblem is the optimal solution to the original linear problem.

By considering the barrier function we can convert our original problem with linear constraints into a problem with nonlinear constraints. As discussed above, when $\mu \to 0$, then a minimum to the barrier function will be a minimum to our original LPP. We will solve this *barrier subproblem* iteratively, alternating between taking a step in the direction that results in a sufficient decrease in the *current* barrier problem $B(x, \mu)$, and then decreasing the value of μ for the next iteration. If we appropriately minimize and choose a good sequence $\{u_k\}$ of decreasing iterates for μ, we will approach the optimal solution x^* for the original problem in the limit.

We will choose our steps for solving the barrier subproblem by a generalization of the common line search used in unconstrained optimization — the *feasible point descent method* (see, e.g., [GMW81]). Briefly, recall the line search paradigm used in unconstrained optimization: we are given a point $x_k \in \mathbb{R}^n$, and we are trying to minimize a function $f : \mathbb{R}^n \to \mathbb{R}$. We find a search direction p and a parameter α such that $x_{k+1} = x_k + \alpha p$ satisfies $f(x_{k+1}) < f(x_k)$. The feasible point descent method is similar except now we enforce the property that at every step, the iterate x_k satisfies the constraints $Ax_k = b$. The particular search direction we choose will be the Newton direction p_B associated with $B(x, \mu)$, which we will derive an expression for next.

First, notice that the gradient and Hessian of $B(x, \mu)$ can be expressed rather simply:

$$\begin{aligned} g &= \nabla_x B(x, \mu) = c - \mu X^{-1} e \\ H &= \nabla_x^2 B(x, \mu) = \mu X^{-2}. \end{aligned}$$

Similar to the Newton direction in unconstrained minimization, given $x \in \mathbb{R}^n$ we can choose a *constrained* Newton step p_B that satisfies

$$\min_p \frac{1}{2} p^T H p + g^T p \text{ subject to } A(x + p) = b, \tag{6}$$

that is, a normal Newton step that also satisfies our linear constraints. Ignoring the constraints this would clearly be a descent direction, since $H = \mu X^{-2}$ is symmetric positive definite if $x, \mu \geq 0$. By considering the KKT conditions for this "Newton problem", we can gather together all the constraints in one matrix equation like so:

$$\begin{pmatrix} H & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} -p_B \\ \lambda \end{pmatrix} = \begin{pmatrix} c - \mu X^{-1} e \\ 0 \end{pmatrix}. \tag{7}$$

Focusing on the first line in the above equation, we get

$$-H p_B + A^T \lambda = c - \mu X^{-1} e. \tag{8}$$

Multiplying by AX^2 and rearranging gives

$$AX^2 A^T \lambda = AX^2 c - \mu X e + AX^2 H p_B.$$

Notice that in (6), we need to satisfy $A(x+p) = b$. By assumption $Ax = b$, and so it follows that if p_B is a solution to the problem then $Ap_B = 0$. Combining this fact with the above equation, and that $H = \mu X^{-2}$, the equation simplifies to

$$AX^2A^T\lambda = AX^2c - \mu Xe.$$

The matrix X is positive definite since $x \geq 0$, and since A has full rank the matrix AX^2A^T must have a unique solution λ^*. We can therefore use (8) to express

$$p_B = x + \frac{1}{\mu}X^2(A^T\lambda^* - c). \tag{9}$$

Recall that we need to decrease the barrier parameter μ over the course of the algorithm, so that in the limit the sequence of iterates $\{\mu_k\}$ satisfies

$$\lim_{k\to\infty} \mu_k = 0.$$

We will handle this in the simplest way possible in the algorithm: we will choose a parameter $\rho \in (0,1)$ and set $\mu_{k+1} = \rho\mu_k$, where the first μ_k is chosen to be sufficiently large.

Now we can write the primal barrier Newton method as a complete algorithm. The

Algorithm 1 Interior Point Method 1: Primal Newton Barrier Method

Choose $\rho \in (0,1)$ and $\mu_0 > 0$ sufficiently large.
Choose a point x_0 so that $Ax_0 = b$ and $x \geq 0$ are both satisfied.
$k \leftarrow 0$
for $k = 0, 1, 2, \ldots$ **do**
  $\mu_k = \rho\mu_{k-1}$
  Compute the constrained Newton direction p_B using (9)
  Solve: $\min_\alpha B(x_k, \mu_k)$, where $x_k = x_{k-1} + \alpha p_B$, subject to $Ax_k = b$
  $x_k \leftarrow x_{k-1} + \alpha p_B$
  $k \leftarrow k + 1$
end for

motivation for calling this an "interior point method" should now be clear — unlike simplex, Algorithm 1 actually begins in the interior of the feasible region of the LPP, and travels on a path towards the the boundary, converging at the optimum. The path that the iterates $\{x_k\}$ trace throughout the operation of the algorithm is referred to as the *central path*, and the study of this path is central to the convergence analysis of this algorithm. For a proof of convergence of the algorithm, we will refer the interested reader to [GMS+86]. We close by remarking that, since the path traced by Algorithm 1 is the same as the path traced by Karmarkar's algorithm, it follows that Algorithm 1 converges in time $O(n^{3.5}L^2)$, where n is the number of variables in the input and L is the number of bits in the input.

4.2 Primal-Dual Interior Point Methods

In this section, we will introduce a generalization of the primal Newton barrier method above that utilizes the information given by the dual problem to an instance to the LPP. These primal-dual algorithms sit at the very core of the most efficient linear program solvers today, converging to an ϵ-accurate solution with a worst case bound of $O(\sqrt{m+n}\ln 1/\epsilon)$ on the number of iterations of the method [Ren88]. In fact, according to the general theory, using a barrier function of $\ln x$ implies that root term in the above expression is the very best that we can do [Gon12, Nes04].

First, recall the definition of the linear programming problem and the dual problem:

$$\begin{aligned} \textbf{Primal:} &\quad \min c^T x \text{ subject to: } Ax = b, x \geq 0 \\ \textbf{Dual:} &\quad \max b^T \lambda \text{ subject to: } A^T\lambda + s = c, s \geq 0 \end{aligned}$$

The solutions of these problems (x, λ, s) are characterized by the KKT conditions (2), which we will write again here:

$$\begin{aligned} A^T\lambda + s &= c \\ Ax &= b \\ x &\geq 0 \\ s &\geq 0 \\ x_i s_i &= 0, 1 \leq i \leq n. \end{aligned}$$

The primal-dual interior-point methods differ from the strict primal methods in one key point: the Newton direction is computed for (λ, s) as well as for x. To compute this Newton direction, we will appeal to the theory discussed in Section 2.3.

First, we collect together the KKT constraints for the dual problem into a single mapping:

$$F(x, \lambda, s) = \begin{pmatrix} A^T\lambda + s - c \\ Ax - b \\ XSe \end{pmatrix} = 0 \tag{10}$$

The Jacobian of F can be written as the following matrix:

$$J(x, \lambda, s) = \begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix},$$

where each column (taken from left to right) correspond to taking the partial derivative with respect to x, λ and s, respectively. We can write our new Newton direction (x_B, λ_B, s_B) as the solution to the following system of equations.

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} x_B \\ \lambda_B \\ s_B \end{pmatrix} = \begin{pmatrix} -(A^T\lambda + s - c) \\ -(Ax - b) \\ -XSe \end{pmatrix}.$$

This is good, but where does the barrier come in? Recall the KKT conditions (5) associated with the barrier function. They are identical to the KKT conditions for the regular linear programming problem, except for the relaxed complementarity constraint $XSe = \mu e$. We will in fact solve the Newton equations while including information about the barrier parameter using this relaxed constraint:

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} x_B \\ \lambda_B \\ s_B \end{pmatrix} = \begin{pmatrix} -(A^T\lambda + s - c) \\ -(Ax - b) \\ -(XSe - \mu e) \end{pmatrix}. \tag{11}$$

So, the step update in Algorithm 1 just becomes

$$(x_{k+1}, \lambda_{k+1}, s_{k+1}) = (x_k, \lambda_k, s_k) + \alpha(x_B, \lambda_B, s_B),$$

where (x_B, λ_B, s_B) is our new Newton direction. Using this we can restate the primal Newton algorithm using the dual information. Algorithm 2 forms the basis for many LP solvers today

Algorithm 2 Interior Point Method 2: Primal-Dual Barrier Method

Choose $\rho \in (0, 1)$ and $\mu_0 > 0$ sufficiently large.
Choose a point (x_0, λ_0, s_0) so that x_0 is feasible for the primal problem and (λ_0, s_0) is feasible for the dual.
for $k = 0, 1, \ldots$ **do**
  $\mu_k = \rho\mu_{k-1}$
  Compute the primal-dual Newton direction (x_B, λ_B, s_B) using (11)
  Find α, the solution to: $\min_\alpha B(x_k, \mu_k)$, where $x_k = x_{k-1} + \alpha p_B$, subject to $Ax_k = b$
  $x_k \leftarrow x_{k-1} + \alpha p_B$
end for

due to it's efficiency. For a proof of convergence, we refer to either [NW06, Chapter 14] or for a more general treatment [Gon12].

The tradeoff between the interior point methods and simplex can now be intuitively quantified: both of the above interior point methods require a polynomially-bounded number of iterations, but each individual iteration can be quite expensive: calculating the Newton direction and the optimal steplength α are nontrivial operations. On the other hand, each iteration of simplex is relatively easy to compute, but it can require exponentially many iterations in the worst case.

4.3 Discussion

The presentation of the primal and primal-dual methods above admittedly left some important questions unanswered. In this section we will discuss some of these questions and problems, and how they are typically answered in practice.

First is the question of convergence: what kind of measure can be used to determine when to stop the algorithm? In both algorithms, some piece of the KKT conditions are

often used. In the primal Newton barrier method the residual r of the first KKT condition can be used, that is, at iteration k we set

$$r = x^T(c - A^T\lambda) + \mu e,$$

and check if both $||r||$ and μ are sufficiently small.

In the primal-dual interior point method the information from the dual problem allows us to use the relaxed complementarity gap condition to measure progress. Recall the equation $XSe = \mu$ from the KKT conditions for the barrier subproblem, and suppose (x, λ, s) is primal and dual feasible. Then from the definition of the primal and dual problems

$$s^T x = (c - A^T\lambda)^T x = c^T x - \lambda^T A x = c^T x - \lambda^T b,$$

the difference between the primal and dual objective values at this point. By Theorem 2, x is a solution for the primal if and only if (λ, s) is a solution for the dual, and if we're at a solution the objective values are equal. It follows that $s^T x = n\mu$ can be used to exactly measure the progress of the algorithm, and so it is sufficient to halt when μ is small.

Another important question is: how do we find a good feasible starting point for the barrier problem? Unfortunately, this is not easy to answer. For the primal method, Gill et. al. suggests solving another linear program using a simple barrier method [GMS+86]. The primal-dual method is more difficult, as satisfying the feasibility of a point (x, λ, s) with respect to both the primal and dual barrier problems is not straightforward. In practice, a good heuristic technique is described in [NW06, Section 14.2], and some modern strategies (for quadratic programming, but the techniques can be applied without loss of generality here) are presented in [DDSdS10].

We also note that the primal and primal-dual methods that we discuss strictly maintain constraint feasibility in all the steps we take. In practice, it is often more efficient to relax some of these feasibility constraints (and so the system of equations (11) will not be homogeneous). In this case, however, strong damping of the Newton step may be required [Gon12].

5 Conclusion

As discussed above, the interior point methods for linear programming are fundamentally developed from the application of nonlinear optimization methods to problems with a linear objective function. Just the simple addition of the barrier parameter μ allows us enough control over the convergence of the nonlinear optimization problem to a solution of the linear optimization problem.

In this paper we have only touched on the design and implementation of interior point methods. For further reading, an excellent reference that we have used throughout is [NW06]. The book by Nesterov [Nes04] also has some good material. For a book discussing the finer points of primal-dual interior point methods, Wright [Wri97] is indispensible. It is our hope that the reader will be able to use this article and these references to help deepen their knowledge of the linear programming problem, and even constrained optimization, as a whole.

References

[Dan47] G.B. Dantzig. Maximization of a linear function of variables subject to linear inequalities. *Activity Analysis of Production and Allocation*, pages 339 – 347, 1947.

[DDSdS10] Marco DApuzzo, Valentina De Simone, and Daniela di Serafino. Starting-point strategies for an infeasible potential reduction method. *Optimization Letters*, 4:131–146, 2010. 10.1007/s11590-009-0150-9.

[GMS+86] Philip E. Gill, Walter Murray, Michael A. Saunders, J. A. Tomlin, and Margaret H. Wright. On projected newton barrier methods for linear programming and an equivalence to karmarkar's projective method. *Mathematical Programming*, 36(2):183–209, November 1986.

[GMW81] Phillip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization*. Academic Press, London, 1981.

[Gon12] Jacek Gondzio. Interior point methods 25 years later. *European Journal of Operational Research*, 218(3):587–601, 2012.

[Kar84] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–396, 1984.

[Kha79] Leonid Khachiyan. A polynomial time algorithm in linear programming. *Doklady Akademiia Nauk SSSR*, 244:1093 – 1096, 1979.

[KM70] Victor Klee and George J. Minty. How good is the simplex algorithm? Technical report, Washington University Department of Mathematics, 1970.

[Nes04] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer, Boston, 2004.

[NW06] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 2006.

[Ren88] James Renegar. A polynomial-time algorithm, based on newton's method, for linear programming. *Mathematical Programming*, 40:59–93, 1988. 10.1007/BF01580724.

[ST01] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time. In Jeffrey Scott Vitter, Paul G. Spirakis, and Mihalis Yannakakis, editors, *STOC*, pages 296–305. ACM, 2001.

[Wri97] S.J. Wright. *Primal-Dual Interior Point Methods*. SIAM, Philadelphia, 1997.

[Wri05] Margaret H. Wright. The interior-point revolution in optimization: history, recent developments, and lasting consequences. *Bull. Amer. Math. Soc. (N.S*, 42:39,56, 2005.