PRIMAL-DUAL INTERIOR POINT METHOD FOR LINEAR PROGRAMMING

KELLER VANDEBOGERT AND CHARLES LANNING

1. Introduction

Interior point methods are, put simply, a technique of optimization where, given a problem with both equality and inequality constraints, reduces the problem to a sequence of equality constrained problems. One of these methods include *barrier methods*, in which a barrier function is considered. Intuitively, barrier methods convert a constrained problem to an unconstrained problem by selecting a specially constructed function $B(x)$ which has the property that B vanishes at the constrained optimal value, and $B(x) \to \infty$ as x tends to the boundary of the feasible region. A sequence of equalities is then considered, in which the limiting value converges to the optimal value of the constrained value.

In this paper we consider an approach known as the *Primal-Dual Method* for Linear Programming problems. There is a Primal-Dual method for nonlinear problems, but we shall only cover the case for linear problems here.

2. Lagrangians and Dual Problems

The heart of optimization lies in problems of the following form:

Date: September 3, 2017.
Key words and phrases. Linear Programming, Interior Point Methods.

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t} \quad & g_i(x) \leq 0 \\ & h_j(x) = 0 \end{aligned} \tag{2.1}$$

We refer to each g_i as an inequality constraint, and each h_j as an equality constraint. We then form the *Lagrangian Function* for the problem (2.1).

$$L(x, \mu, \lambda) = f(x) - \sum_{i=1}^{m} \mu_i g_i(x) - \sum_{j=1}^{l} \lambda_j h_j(x) \tag{2.2}$$

The constants μ_i and λ_j are referred to as our Lagrange multipliers. In general, the Lagrangian is extremely important to the theory of optimization. Another question can be asked, in which we consider minimizing the Lagrangian with respect to our variable x. This curiosity will lead us in the direction of formulating the *Dual Problem*. Define:

$$q(\mu, \lambda) = \inf_x L(x, \mu, \lambda) \tag{2.3}$$

We require that the infimum exists and is finite. We can refer to q as the dual function, and it is easy to see that this provides a lower bound on the solution of (2.1). Now, in order for the direction of our inequality constraints to remain the same, it is essential that all $\mu_i \geq 0$. This is essentially a constraint, and with this, we can introduce the *Dual Problem*:

$$\begin{aligned} \max \quad & q(\mu, \lambda) \\ \text{s.t} \quad & \mu_i \geq 0 \end{aligned} \tag{2.4}$$

It is natural to consider the difference $f(x)-q(\mu,\lambda)$, which is referred to as the *duality gap*. As $x \to x^*$, where x^* denotes the solution of (2.1), it can be shown that the duality gap will tend to 0. We are now in a position to derive the Primal-Dual method.

3. Derivation of Primal-Dual Interior Point Method for Linear case

Consider a linear programming problem of the following form:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t} \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{3.1}$$

Where $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, and $A \in \mathbb{R}^{m\times n}$. We intend to construct the dual problem for this problem, known as the primal problem. We thus form the Lagrangian function:

$$L(x,y,s) = c^T x - y^T(Ax-b) + s^T x \tag{3.2}$$

Where y is the vector of our Lagrange multipliers corresponding to the equality constraints, and $s \geq 0$ is the vector of inequality constraint multipliers. We now intend to rewrite (3.2) in order to deduce some properties of our dual problem. We have:

$$\begin{aligned} L(x,y,s) &= c^T x - y^T(Ax-b) - s^T x \\ &= c^T x - y^T Ax + y^T b + s^T x \\ &= y^T b + (-A^T y + c - s)^T x \end{aligned}$$

Where the last step uses the fact that $y^TA = (A^Ty)^T$. Here is where we use properties of linear functions and the assumption of our dual function in the previous section. If we examine $\inf_x L(x, y, s)$, we see that we are looking for $\inf_x(-A^Ty + c - s)^Tx$. However, we required that the dual function be bounded below. It is obvious that a linear function will be bounded below if and only if its slope is identically 0. Thus, we see that $-A^Ty + c - s = 0$. We have thus derived the corresponding dual problem:

$$\begin{aligned} \max \quad & b^Ty \\ \text{s.t} \quad & A^Ty + s = c \\ & s \geq 0 \end{aligned} \tag{3.3}$$

We can now consider a barrier reformulation of our linear problem, which uses the fact that the solution set for (3.1) is precisely the same as the following:

$$\begin{aligned} \min \quad & c^Tx - \mu \sum_{i=1}^{n} \ln(x_i) \\ \text{s.t} \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{3.4}$$

This is easy to see purely by the properties of barrier reformulations. We see that as x tends to the boundary of our feasibly region, our barrier function will tend to infinity, effectively forcing our problem to remain constrained. If we consider the Lagrangian function for the above barrier reformulation, we easily deduce the following system:

$$\begin{aligned} &\bigtriangledown_x L(x,y) = c - \mu X^{-1}e - A^T y = 0 \\ &\bigtriangledown_y L(x,y) = b - Ax = 0 \\ &x > 0 \end{aligned} \tag{3.5}$$

Where we let X denote the square matrix whose ith diagonal entry is just the ith entry of x. This is where we can introduce a "slack" vector $s = \mu X^{-1}e$, where e is an nx1 vector of 1's, and reformulate (3.5) in the following form:

$$\begin{aligned} &A^T y + s = c \\ &Ax = b, \quad x > 0 \\ &Xs = \mu e \end{aligned} \tag{3.6}$$

We can then immediately formulate the dual barrier problem as so:

$$\begin{aligned} \max \quad & b^T y - \mu \sum_{i=1}^{n} \ln(s_i) \\ \text{s.t} \quad & A^T y = s \\ & s \geq 0 \end{aligned} \tag{3.7}$$

Which yields the following system almost immediately after considering the Lagrangian:

$$\begin{aligned} &A^T y + s = c, \quad s > 0 \\ &Ax = b, \\ &Xs = \mu e \end{aligned} \tag{3.8}$$

But then we immediately combine the conditions above from (3.4) and (3.8) and derive the following conditions, which are essential to constructing the Primal-Dual interior point method algorithm.

$$\begin{aligned} &A^T y + s = c, \quad s > 0 \\ &Ax = b, \quad x > 0 \\ &Xs = \mu e \end{aligned} \tag{3.9}$$

We now consider the system (3.9), and intend to apply Newton's method for multidimensional optimization. We assume the reader is familiar with this method, and consider the following vector function $F_\gamma(x, y, s)$ which we have perturbed by a small arbitrary parameter γ:

$$F_\gamma(x, y, s) = \begin{bmatrix} Ax - b \\ A^T y + s - c \\ Xs - \gamma\mu e \end{bmatrix} \tag{3.10}$$

We now consider the gradient of our function. Each column represents the matrix derivative with respect to vectors x, y, and s, respectively. We have:

$$\bigtriangledown F_\gamma(x, y, s) = \begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ S & 0 & X \end{bmatrix} \tag{3.11}$$

Of course the search directions for Newton's method are determined by solving the system $\bigtriangledown F_\gamma(x, y, s)D = -F_\gamma(x, y, s)$, where D is our direction method. We then have:

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ S & 0 & X \end{bmatrix} \begin{bmatrix} d_x \\ d_y \\ d_s \end{bmatrix} = - \begin{bmatrix} Ax - b \\ A^T y + s - c \\ Xs - \gamma\mu e \end{bmatrix} \tag{3.12}$$

Now assume we are given initial iterates (x^0, y^0, s^0) which satisfy x^0, $s^0 > 0$. Then we can define the initial dual residuals as so:

$$\begin{aligned} r_P^0 &= b - Ax^0 \\ r_D^0 &= c - A^T y^0 - s^0 \end{aligned} \tag{3.13}$$

This then quickly generalizes to the following iteration scheme, where (x^k, y^k, s^k) is the kth iterate:

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ S^k & 0 & X^k \end{bmatrix} \begin{bmatrix} d_x \\ d_y \\ d_s \end{bmatrix} = \begin{bmatrix} r_P^k \\ r_D^k \\ -X^k s + \gamma \mu_k e \end{bmatrix} \tag{3.14}$$

Of course we then require finding a step size for our next iterate. There is no perfect step size, but the general method is to follow a central path that converges to a solution. This is the biggest advantage of interior point methods, which do not require explicit calculation of the central path. Instead, we form a *horn neighborhood*, which is named because of the shape of the neighborhood as it follows the central path. See [1] for an explicit illustration. The horn neighborhood improves calculation efficiency and speed because our iterates need only stay within some distance of the central path, which tends to 0 as the method converges. Intuitively, this will of course converge to the optimal point, but with minimal computing power compared to explicit calculations of the central path. See [1] for additional theoretical considerations on this derivation.

4. The Primal-Dual Interior Point Algorithm: Simplified Version for Linear Programming

There is a simplified algorithm for this method [1] which has the following explicit form. This can be found by means of backward substitution in the system (3.12).

(1) Initialization step: Choose β, $\gamma \in (0,1)$ and let $\epsilon > 0$ be your tolerance. Choose $x^0 = s^0 = e$ and $y^0 = 0$

(2) Set k=0

(3) We now begin the iteration scheme: Set:

$$r_P^k = b - Ax^k$$

$$r_D^k = c - A^T y^k - s^k$$

$$\mu_k = \frac{(x^k)^T s^k}{n}$$

(4) Check our termination: If $||r_P^k|| < \epsilon$, $||r_D^k|| < \epsilon$, and $(x^k)^T s^k < \epsilon$, then we stop.

(5) Compute our directions using the following:

$$Md_y = r \tag{4.1}$$

where

$$M = A(S^k)^{-1} X^k A^T$$

$$r = b + A(S^k)^{-1}(X^k r_D^k - \gamma \mu_k e)$$

We then find our other directions with the following:

$$d_s = r_D^k - A^T d_y \tag{4.2}$$

$$d_x = -x^k + (S^k)^{-1}(\gamma \mu_k e - X^k d_s) \tag{4.3}$$

(6) We then calculate step size using the following:

$$\alpha_P^{max} = \min\left\{ -\frac{x_i}{(d_x)_i} : (d_x)_i < 0, i \in I \right\} \tag{4.4}$$

$$\alpha_D^{max} \min\left\{ -\frac{s_i}{(d_s)_i} : (d_s)_i < 0, i \in I \right\} \tag{4.5}$$

We then choose our value for α^{max} to be the minimum of (4.4) and (4.5), and the value α_k is then chosen as:

$$\alpha_k = \min\{1, \theta\alpha^{max}$$

Where $\theta \in (0, 1)$. In practice we will often choose θ to be close to 1. In our code we chose $\theta = .95$.

(7) Update our iterates:

$$x^{k+1} = x^k + \alpha_k d_x$$

$$y^{k+1} = y^k + \alpha_k d_y$$

$$s^{k+1} = s^k + \alpha_k d_s$$

(8) Repeat the algorithm, with $k = k + 1$.

In practice, the above algorithm is computationally efficient and fairly open to adjustments. Indeed, changes to our starting points may yield different paths to convergence, and changes to the constants may affect the amount of iterations required for desired precision.

5. Numerical Tests and Implementation

We tested our code on various problems and obtained excellent results. On most problems, between 10 and 20 iterations are needed when our ϵ is 10^{-8}, which will be our standard tolerance, as explained later. In these tests, we will encounter both minimum and maximum problems. Out code is designed to minimize a given objective function, but if we would maximize, then the algorithm simply multiplies all coefficients in the objective function by -1. Let's look at the following example.

$$\begin{aligned} \max \quad & 2x_1 + 2x_2 \\ \text{s.t.} \quad & x_1 + x_2 \leq 3 \\ & x_1, x_2 \geq 0 \end{aligned} \tag{5.1}$$

This problem was solved just 8 iterations, with the optimal value found to be 6 at $(x_1, x_2) = (1.5, 1.5)$. In fact, there are infinitely many optimal solutions lying on the line $x_1 + x_2 = 3$, however our method will always return the analytic center in these cases. Our solution can be seen in Figure 1.

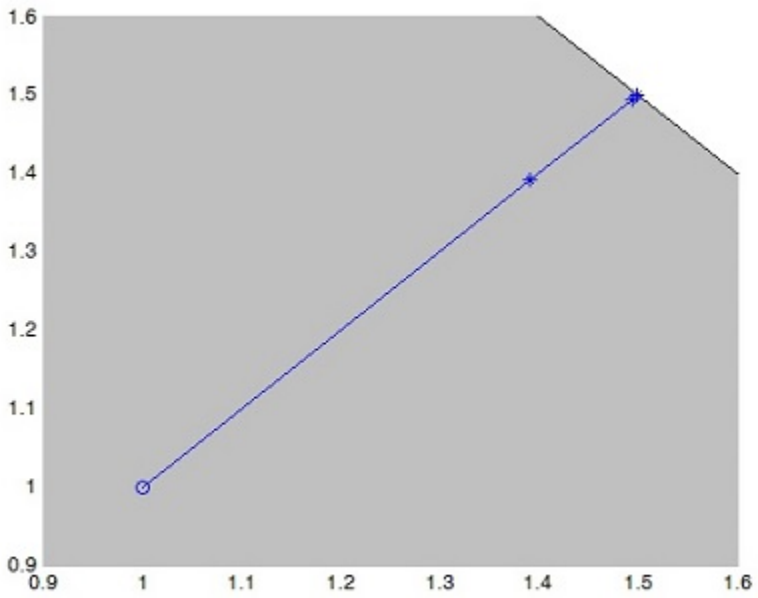

FIGURE 1. Problem 5.1.

For a more interesting example, let's look at the following.

$$\begin{aligned} \max \quad & 3x_1 + 5x_2 \\ \text{s.t.} & x_1 \leq 3 \\ & 2x_2 \leq 12 \\ & 3x_1 + 2x_2 \leq 18 \\ & x_1, x_2 \geq 0 \end{aligned} \tag{5.2}$$

Our algorithm solved this one in 11 iterations, with optimal value 36 found at $(x_1, x_2) = (2, 6)$. If we change our initial point from $(1, 1, 1, 1, 1)$ to others, the algorithm still reaches the solution in about 11 iterations (occasionally 10 or 12). These can be seen in Figure 2.

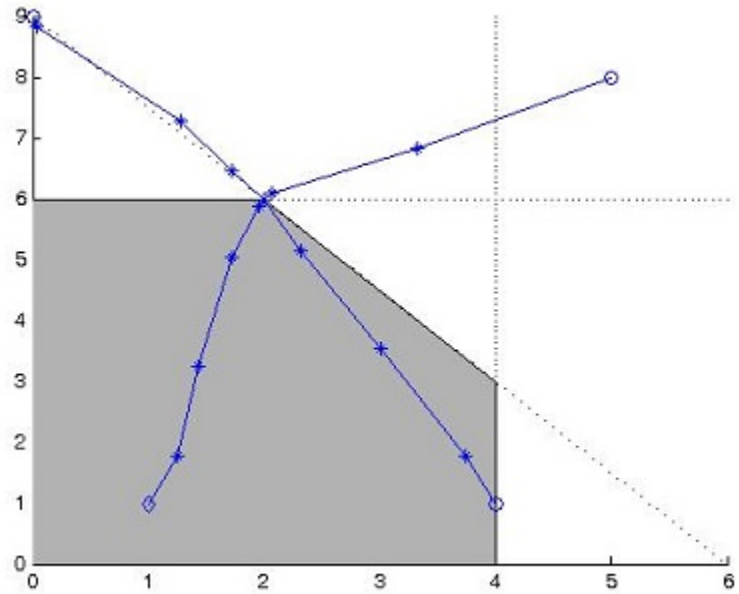

FIGURE 2. Problem 5.2 using various initial points.

If we let ϵ be 10^{-32}, then we obtain a solution that is within $9.8900443 \times 10^{-10}$ of the solution of 35.999999999010996 obtained when ϵ is the

usual 10^{-8}, and our algorithm takes 42 iterations. Thus, an increase in ϵ is probably not worth the machine time unless accuracy is imperative.

Our algorithm does not just handle $\leq$ constraints, though. When the constraint is $=$, our algorithm will not add a slack, and when it is $\geq$, our algorithm will add a slack and set its coefficient in that constraint to -1, instead of 1. For example, consider the following.

$$\begin{aligned} \min \quad & 2x_1 + 3x_2 \\ \text{s.t.} \quad & 0.5x_1 + 0.25x_2 \leq 4 \\ & x_1 + 3x_2 \geq 20 \\ & x_1 + x_2 = 10 \\ & x_1, x_2 \geq 0 \end{aligned} \tag{5.3}$$

In this problem, we have all three types of constraints. Our code finds an optimal solution in just 15 iterations. See Figure 3.
Note that in this example, our set of feasible points is not a region, and is instead a line segment, denoted by the thicker constraint line.

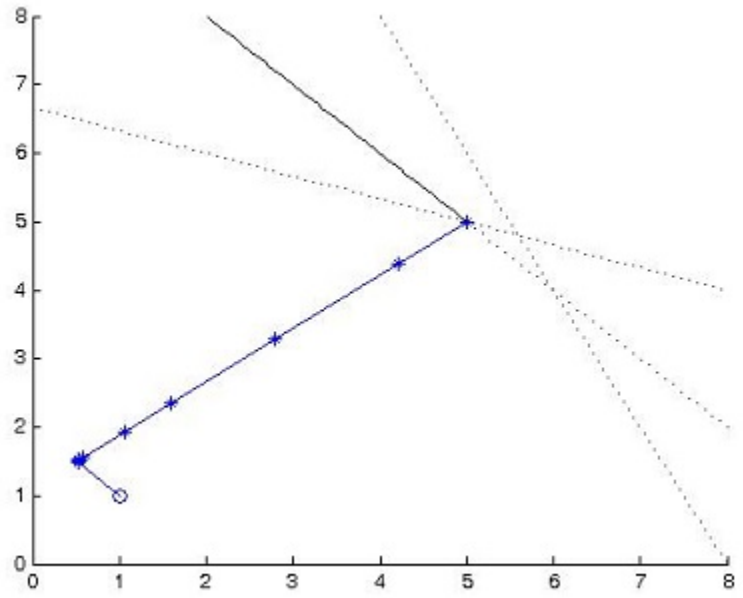

FIGURE 3. Problem 5.3.

6. More Numerical Tests

We have chosen additional problems with various numbers of variables, and with different constraint types. Our tolerance in each is 10^{-8}. Problems 5.1, 5.2, and 5.3 will be included as well.

(1)

$$\begin{aligned} \max \quad & 2x_1 + 2x_2 \\ \text{s.t.} \quad & x_1 + x_2 \leq 3 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Iterations:

8

Optimal Solution:

x1 = 1.499999999576468.

x2 = 1.499999999576468.

Optimal Value:

5.999999998305872.

(2)

$$\begin{aligned} \max \quad & 3x_1 + 5x_2 \\ \text{s.t.} \quad & x_1 \leq 3 \\ & 2x_2 \leq 12 \\ & 3x_1 + 2x_2 \leq 18 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Iterations:

11

Optimal Solution:

x1 = 1.999999999944728.

x2 = 5.999999999834196.

Optimal Value:

35.99999999900516.

(3)

$$\begin{aligned} \min \quad & 2x_1 + 3x_2 \\ \text{s.t.} \quad & 0.5x_1 + 0.25x_2 \leq 4 \\ & x_1 + 3x_2 \geq 20 \\ & x_1 + x_2 = 10 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Iterations:

15

Optimal Solution:

5.000000000000392

4.999999999999693

Optimal Value

25.000000000760810

(4)

$$\begin{aligned} \max \quad & 2x_1 + 7x_2 + 6x_3 + 4x_4 \\ \text{s.t.} \quad & x_1 + x_2 + 0.83x_3 + 0.5x_4 \leq 65 \\ & 1.2x_1 + x_2 + x_3 + 1.2x_4 \leq 96 \\ & 0.5x_1 + 0.7x_2 + 1.2x_3 + 0.4x_4 \leq 80 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

Iterations:

15

Optimal Solution:

x1 = 2.249787143151893e-10.

x2 = 5.16007534179303.

x3 = 53.20150657500549.

x4 = 31.36534840108348.

Optimal Value:

480.7909604473681.

(5)

$$\begin{aligned} \max \quad & 2x_1 - x_2 + 2x_3 \\ \text{s.t.} \quad & 2x_1 + x_2 \leq 10 \\ & x_1 + 2x_2 - 2x_3 \leq 20 \\ & x_2 + 2x_3 \leq 5 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

Iterations:

15

Optimal Solution:

x1 = 4.999999998963977.

x2 = 5.180121984762661e-10.

x3 = 2.499999998963975.

Optimal Value:

14.99999999533789.

(6)

$$\begin{aligned} \min \quad & -2x_1 - 3x_2 - 4x_3 \\ \text{s.t.} \quad & 3x_1 + 2x_2 + x_3 = 10 \\ & 2x_1 + 5x_2 + 3x_3 = 15 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

Iterations:

10

Optimal Solution:

x1 = 2.142857142835484.

x2 = 1.516095219269412e-10.

x3 = 3.571428571190327.

Optimal Value:

-18.5714285708871.

Note that the number of iterations was never any more that 15, and our accuracy was always within 10^{-9} of the actual solution. The iterations seem to be higher when working with more variables (in example 4, we use four variables and three slacks) and when there are $\geq$ constraints.

7. Conclusion

In conclusion, we see that interior point methods are computationally efficient even with dealing with relatively large constraints. The theory

behind them is based on the concept of Primal-Dual barrier reformulations and uses a modified Newton's method to stay sufficiently close to the central path of convergence, as opposed to explicit calculation of this central path, which is the reason this method is so successful. Our numerical tests all converged successfully, and the code used was extremely flexible in dealing with equality-inequality constraints.

References

[1] Lesaja, Goran. 2009. "Introducing Interior-Point Methods for Introductory Operations Research Courses and/or Linear Programming Courses." Open Operational Research Journal, 3: 1-12. doi: 10.2174/1874243200903010001