

---

### Primal-Dual Interior Point Methods for Linear Programming

- Primal-Dual methods perform better than Primal methods or Dual methods. This will probably remain true.
- The current **most popular** algorithm is Predictor-Corrector. This may change.

---

### The basis of the Complexity Theory

- The **problem size** is the number of bits needed to store the problem on a machine. This is finite if all of the data is **rational**, so we will make this assumption, specifying each entry in  $A$ ,  $b$ , and  $c$  as the ratio of two integers. We'll suppose that it takes  $L$  bits to store these entries along with  $m$  and  $n$ .
- We note that  $m \leq n$ , so  $m$  never appears in the complexity bounds.
- Suppose we know the active constraints at the optimal solution  $x^*$ . Then  $x^*$  can be expressed as the solution to the linear system of equations defined by these constraints. Since the coefficient matrix and right-hand side are rational, so is  $x^*$ , and it has an exact representation in a number of bits bounded by a polynomial in the number of bits of data. In fact, the nonzero components are bounded below by  $\epsilon \equiv 2^{-L}$ .
- Thus we have motivation for allowing an algorithm to “round-off” to the closest rational number that is representable within our bit bound, and complexity proofs need to show that this does not hurt convergence.
- And we know that at some stage we can terminate the iteration and set all of the very small components of the solution vector to zero, using a linear system to solve for the exact values of the others.
- Each iteration will take time polynomial in  $L$ , so we just need to make sure that the number of iterations is bounded by a polynomial in  $L$ .

---

### The basic algorithm

Given  $(x^{(0)}, y^{(0)}, z^{(0)})$  satisfying

$$\begin{aligned} Ax^{(0)} &= b \\ A^T y^{(0)} + z^{(0)} &= c \\ x^{(0)} &> 0 \\ z^{(0)} &> 0 \end{aligned}$$

For  $k = 0, 1, \dots$ , until  $x^{(k)T} z^{(k)}$  small enough,

- Solve

$$\begin{bmatrix} Z^{(k)} & 0 & X^{(k)} \\ A & 0 & 0 \\ 0 & A^T & I \end{bmatrix} \begin{bmatrix} \Delta x^{(k)} \\ \Delta y^{(k)} \\ \Delta z^{(k)} \end{bmatrix} = \begin{bmatrix} -X^{(k)} z^{(k)} + \sigma_k \mu_k e \\ 0 \\ 0 \end{bmatrix}$$

where  $\sigma_k \in [0, 1]$  and  $\mu_k = x^{(k)T} z^{(k)} / n$ .

- Set

$$\begin{bmatrix} x^{(k+1)} \\ y^{(k+1)} \\ z^{(k+1)} \end{bmatrix} = \begin{bmatrix} x^{(k)} \\ y^{(k)} \\ z^{(k)} \end{bmatrix} + \alpha_k \begin{bmatrix} \Delta x^{(k)} \\ \Delta y^{(k)} \\ \Delta z^{(k)} \end{bmatrix}$$

choosing  $\alpha_k$  so that  $x^{(k+1)} > 0$  and  $z^{(k+1)} > 0$ .

**Note:** For nonzero  $\sigma$ , if we set  $\alpha_k = 1$ , then we will have

$$x_i^{(k+1)} z_i^{(k+1)} \approx \sigma \mu_k,$$

so we are targeting the particular point on the central path corresponding to the parameter  $\sigma \mu_k$ . If we set  $\sigma = 0$ , we are targeting the point corresponding to 0, i.e., the solution to the LP.

## Some Variations

### Potential Reduction Methods

- **Goal:** reduce a potential function, rather than follow the central path.
- **Measuring progress:** The value  $\phi$  should decrease sufficiently fast, where  $\phi$  is a potential function. One very useful one (Tanabe-Todd-Ye):

$$\phi_\rho(x, z) = \rho \log x^T z - \sum_{i=1}^n \log(x_i z_i)$$

where  $\rho > n$ .

- **Implementation:**

- Choose  $\sigma_k = n/\rho$ .

- Choose  $\alpha_k$  using a line search for the function  $\phi$  with an upper bound on  $\alpha$  equal to  $\alpha_{max}$  = the maximal step that hits the boundary.
- **Convergence result:** If  $\rho = n + \sqrt{n}$ , the bound on the number of iterations is  $O(\sqrt{n} \log(1/\epsilon))$ .
- **Practicalities:**
  - Choose a larger value like  $\rho = 10n$ .
  - Line search need not be exact: see if  $.99\alpha_{max}$  (or similar values) yield a prescribed constant decrease in  $\phi$ .

### Path Following Methods

- **Goal:** try to stay in a neighborhood of the central path, and thus avoid points that are too close to the boundary where  $x_i = 0$  or  $z_i = 0$ .
- **Measuring progress:** The value  $\mu$  should decrease, so that we move closer to a **KKT point**, one that satisfies the optimality conditions for the LP.
- **Classes of methods:**
  - **Short-step methods** choose  $\sigma$  close to 1 and are able to set  $\alpha_k = 1$  without straying far from the central path.
  - **Long-step methods** choose smaller values of  $\sigma$  and thus must choose an  $\alpha_k$  so that  $x_i^{(k+1)} z_i^{(k+1)} \geq \gamma \mu_k$ , where  $\gamma$  is chosen between 0 and 1. (A typical  $\gamma$  is  $10^{-3}$ .)
  - **Predictor-corrector methods** take
    - \* a **predictor step** with  $\sigma = 0$  and  $\alpha_k$  chosen to keep  $\|Xz - \mu e\|_2 \leq \theta \mu$  (typical  $\theta$  is 0.5),
    - \* followed by a **corrector step** with  $\sigma = 1$  and  $\alpha = 1$ .
 The predictor step is a “long step”, and the “short” corrector step pulls the iterate back toward the central path without significantly changing the  $\mu$  value achieved by the predictor.
- **Convergence analysis:** The short-step and predictor-corrector algorithms can be shown to terminate in  $O(\sqrt{n} \log 1/\epsilon)$  iterations, and  $\mu_k$  converges to zero superlinearly for the predictor-corrector algorithm. The bound on the long-step algorithm is  $O(n \log 1/\epsilon)$ , but it behaves well in practice.

### Dealing with infeasible initial points

- It is easy to choose an  $x > 0$  and a  $z > 0$ .
- It is hard to satisfy the equality constraints  $Ax = b$  and  $A^T y + z = c$ .

So infeasible IPMs replace the step equation by

$$\begin{bmatrix} Z^{(k)} & 0 & X^{(k)} \\ A & 0 & 0 \\ 0 & A^T & I \end{bmatrix} \begin{bmatrix} \Delta x^{(k)} \\ \Delta y^{(k)} \\ \Delta z^{(k)} \end{bmatrix} = \begin{bmatrix} -X^{(k)}z^{(k)} + \sigma_k \mu_k e \\ \mathbf{c} - \mathbf{A}^T \mathbf{y}^{(k)} - \mathbf{z}^{(k)} \\ \mathbf{b} - \mathbf{A} \mathbf{x}^{(k)} \end{bmatrix}$$

The **convergence analysis** is not as pretty:  $O(\mathbf{n}^2 \log 1/\epsilon)$  for a “medium-step” version, but the algorithms are much more convenient to use!

---

### Reference

Stephen J. Wright, *Primal-Dual Interior-Point Methods (for linear programming)*, SIAM, 1997.