Mathematical Programming	**Lecture 25**
OR 630 Fall 2006	**November 21, 2006**
Scribe: Yuemeng Sun & Brian Levine	

Interior-Point Methods

Review of the last class:
In the last class, we considered the Primal problem and its dual:

$$\begin{array}{rl} \min & c^T x \\ s.t. & Ax = b \\ & x \geq 0 \end{array} \qquad \begin{array}{rl} \max & b^T y \\ s.t. & A^T y + s = c \\ & s \geq 0 \end{array}$$

Define:

$$\begin{aligned} \mathcal{F}(P) &= \{x \in R^n : Ax = b,\, x \geq 0\} \\ \mathcal{F}^\circ(P) &= \{x \in R^n : Ax = b,\, x > 0\} \\ \mathcal{F}(D) &= \{(y,s) \in R^m \times R^n : A^T y + s = c,\, s \geq 0\} \\ \mathcal{F}^\circ(D) &= \{(y,s) \in R^m \times R^n : A^T y + s = c,\, s > 0\} \end{aligned}$$

Considering a barrier function: $F(x) = -\ln(x) = -\sum(\ln(x_j))$
Instead of optimizing the $c^T x$, we consider the penalty function $\theta_\mu(x) = c^T x + \mu F(x)$ for each $\mu > 0$ and optimize the $\theta_\mu(x)$ within $F^\circ(P)$ without worrying about the boundary.

Also we have the key theorem:

Theorem 1 *a) A necessary and sufficient condition for $\theta_\mu(x)$ to have a minimizer over $F^\circ(P)$ is that both $F^\circ(P)$ and $F^\circ(D)$ are non-empty.*
b) If these conditions hold,a necessary and sufficient condition for x in $F^\circ(P)$ to be a minimizer of $\theta_\mu(x)$ is that: There exists (y,s) in $F^\circ(D)$

$$\begin{array}{rlll} s.t. & A^T y + s = c & s > 0 & \\ & Ax = b & x > 0 & (*) \\ & XSe = \mu e & & \end{array}$$

where $X = Diag(x)$,$S = Diag(s)$,$e = (1, ...1)^T$.

Something to mention are:

- From this theorem we know that $\theta_\mu(x)$ is minimized of $F^\circ(P)$ along the "central path" which satisfies(*).

- What about if we considering the dual objective funtion max $b^T y - \mu F(s)$ over the interior of dual $F^\circ(D)$?
  Answer: The optimal maximizer is again the solutions to the system (*).

- Suppose (x, y, s) satisfy(*) for some small μ. How close to optimal is this solution?
  From the weak duality theorem, we know that for any pair of feasible primal and dual solutions, the difference between their respective objective function values is an upper bound on the additive error for each (since the common optimal value must lie in the interval between them). We can compute that $c^T x - b^T y$
  $= (A^T y + s)^T x - (Ax)^T y$
  $= y^T Ax + s^T x - x^T A^T y$
  $= y^T Ax + s^T x - x^T A^T y$
  $= y^T Ax - x^T A^T y + s^T x$
  $= s^T x$
  $= n\mu$

  From this we can conclude that the barrier parameter μ is just a rescale error parameter.

Introduction to Interior point methods:

1. Affine scaling methods:

   This method was originally due to Dikin ('67) and rediscovered several times after Karmarkar. In this method, in each iteration we take a step along the steepest descent direction (normalized to ensure that we stay in the interior of the feasible region). There are convergence results known, for a number of different variant, but it is not thought to be a polynomial-time algorithm.

2. Potential Reduction Method. Karmarkar introduced potential function reduction methods. The key methodological principle is that rather than measuring progress in terms of the true objective function (or a parameterized extension of it), there is an auxilliary function that is shown to be "not too big" to start, and whenever it gets sufficiently small, then the current solution is near-optimal. Then the key thing is to prove that the potential function decreases sufficiently in each iteration. For the ellipsoid method, one can view the the potnetial function as being the volume of the ellipsoid. (This approach is common in theoretical analyses of the running time of algorithms; it is an often used-tool in the so-called amortized analysis of algorithms.)

   - Primal potential function: This is the setting in which Karmarkar originally proposed this approach to LP.
     $\phi_q(x, \gamma) = q \ln(c^T x - \gamma) + F(x)$
     where $q \geq n$ and $\gamma \leq \gamma^*$, where γ^* is the value of dual.

Karmarkar showed we could find a step Δx for which ϕ decreases by a constant. This implies that the resulting algorithm requires $O(n \ln \frac{1}{\epsilon})$ iterations, where ϵ is the resulting error.

- Primal-Dual Potential Function:
  $\phi_q(x, y, s) = q(ln(x^T s)) + F(s) + F(x)$
  where $q = n + \sqrt{n}$
  Todd and Ye showed one can decrease this potential function by a constant at every iteration. As a result, this leads to a bound of $O(\sqrt{n} \ln \frac{1}{\epsilon})$ iterations to obtain a solution with error ϵ.

Path-Following Methods

Suppose that we have a solution (x, y, s) that approximately solves the system (*), where $x \in F^o(P), (y, s) \in F^o(D)$ and $\frac{1}{\mu} XSe \approx e$, where $\mu = \frac{x^T s}{n}$. We want to move from (x, y, s) to another solution that is of the same form (nearly satisfying system (*), but where the barrier parameter is $\sigma\mu$, where $0 < \sigma < 1$. Note that $\sigma = 0$ means we are trying to find the optimal solution, whereas $\sigma = 1$ means we are trying to get a better approximation for the current barrier value, so typically one might choose $\sigma \approx 0.9$.

Then our new point is $(x + \Delta x, y + \Delta y, s + \Delta s)$. This point must satisfy the following system:

$$\begin{array}{rl} A^T(y + \Delta y) + (s + \Delta s) & = c \\ A(x + \Delta x) & = b \end{array}$$

Since $Ax = b \Longrightarrow A\Delta x = 0$ and since $A^T y + s = c \Longrightarrow A^T \Delta y + \Delta s = 0$.

Think about what the third condition of system (*) means. We are multiplying two diagonal matrices together, which yields something of the form: $(x_j + \Delta x_j)(s_j + \Delta s_j) = \sigma\mu$ for each $j = 1, \ldots, n$. When these terms are multiplied out, we get a second-order term $\Delta x_j \Delta s_j$ which we drop in order to get a linear system (with the hope that this is a low-order term, and the resulting system is a sufficiently good approximation). The approximation becomes $x_j s_j + x_j \Delta s_j + \Delta x_j s_j = \sigma\mu$. Thus the perturbation satisfies:

$$\begin{array}{rrrl} & A^T \Delta y & +\Delta s & = 0 \\ A\Delta x & & & = 0 \\ S\Delta x & & +X\Delta s & = \sigma\mu e - XSe \end{array}$$

We need to solve this system of equations in $(2m + n)$ variables and $(2m + n)$ constraints. We are not going to do this by a direct method (such as Gaussian elimination), but rather by an iterative method. Such iterative linear equation solvers have the advantage that they make use of a good starting solution (or old good solution for the previous μ) and converge in a few iterations to an approximate solution (which will be sufficient for our purposes). Note that even if we solved the system exactly, we would not be on the central path (since we have dropped the second-order term).

Note that the system needed to be solved exhibits the same sparsity patterns as the original input; this is important in solving large systems. We can set up our algorithm as follows:

1. We can express Δs in terms of Δy: $\Delta s = -A^T \Delta y$.
2. We can express Δx in terms of Δs, and hence Δy: $\Delta x = \sigma \mu s^{-1} - x - XS^{-1}\Delta s$ which becomes $\Delta x = \sigma \mu s^{-1} - x + XS^{-1}A^T \Delta y$, where s^{-1} is the vector composed of $1/s_i$, $i = 1...n$.
3. Substitute this into the second set of equations: $A\Delta x = A(\sigma \mu s^{-1} - x + XS^{-1}A^T \Delta y) = 0$. Recall that $Ax = b$ and we get $AXS^{-1}A^T \Delta y = b - \sigma \mu s^{-1}$. Note that $AXS^{-1}A^T$ is a symmetric $m \times m$ positive-definite matrix. This system is also very much similar to what we used to compute the affine-scaling direction $\bar{d} : (AX^2A^T)y = AX^2c$, then $\bar{d} = -X^2c - XA^T y$. Note that we can view XS^{-1} as giving a different affine tranformation of the space (but this is, of course, not the only difference). It is significant, in that we need to solve structured systems, that are structured exactly the same as the ones developed previously, and so the same numerical linear algebraic tools can be used.

What we need now is an iterative method for solving this system. Specifically, we want an algorithm that takes as few steps as possible to get us near to the exact solution. We solve for Δy, and then substitute to find Δs and Δx.

How does this relate to a theorem about running time that can be proved? When solving the LP using a path-following method, we want to stay in some neighborhood of the central path. One way to do so is by ensuring $\| \frac{1}{\mu} XS - e \|_2 \leq \beta$. We let $\sigma = 1 - \frac{\theta}{\sqrt{n}}$ for some $0 < \theta < 1$. Note that this is a much change than is done in practice, but it allows us to stay in a small neighborhood of the central path and we can prove that it takes $O(\sqrt{n} \ln \frac{1}{\epsilon})$ iterations to obtain a solution with error ϵ.

Conclusions

There is a tradeoff between the parameters we choose for proofs and what is actually done in practice. Sometimes algorithms are good even though we can't prove their running times. It is useful to think about when to use interior-point methods by considering the question "When does simplex do badly?". The answer is that simplex performs horribly when there is both primal and dual degeneracy. Primal and dual degneracy is not a problem for Interior Point Methods (IPM), which are not sensitive to degeneracy. To solve problems in practice, note that IPM makes great strides initially, so we can use this in combination with simplex to obtain a true optimal solution. These are so-called "crossover methods". As an example, say we do 10 iterations of IPM and then round to get a basic feasible solution. Then we can use the simplex method to finish the optimization. Today's CPLEX, and any state of the art code offer options that include primal and dual simplex, interior point methods (possibly primal, dual, and primal-dual), as well as crossover methods.