Optimization **Summer 2010**

Lecture 4 — April 21

Lecturer: Julián Mestre

4.1 Assumption

Today we will introduce the simplex algorithm. The main idea behind the simplex algorithm is very simple and can be distilled into a single sentence: Starting from a basic feasible solution, find an adjacent basic feasible solution with better cost until it is not possible to improve the current solution. However, the devil is in the detail, and there are many details to work out to fully develop the algorithm. The aim of this lecture is to introduce the main ideas of the simplex algorithm without overwhelming you with too many details. This means that we will have to make a number of simplifying assumptions. For the time being, we will assume that

1. The linear program is in standard form.

$$\begin{aligned} \text{minimize} \quad & \mathbf{c} \cdot \mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned}$$

   where $A \in \mathcal{R}^{m \times n}$ has m linearly independent rows.

2. Every feasible basis B of A is non-degenerate; that is, $\mathbf{x_B} = \mathbf{A_B^{-1}b} > \mathbf{0}$.

3. We are provided with an initial basic feasible solution.

4. The feasible region of the LP is bounded.

In subsequent lectures we will see how to remove each of these assumptions.

4.2 Developing the Simplex algorithm

Here is our starting point:

Algorithm 1 SIMPLEX-TAKE-1$(\mathbf{A}, \mathbf{b}, \mathbf{c})$

```
1.  B = some feasible basis of A
2.  // we denote the indices of B by {b_1, ..., b_m}
3.  repeat
4.    for j ∈ {1, ..., n} \ B and b_i ∈ B do
5.      D = B + j − b_i
6.      x_B = A_B^{-1} b
7.      if D is a basis of A then
8.        y_D = A_D^{-1} b
9.        if y_D ≥ 0 and c_D · y_D < c_B · x_B then
10.         B = D
11. until basis B hasn't changed in this iteration
12. return B
```

The first refinement of this basic algorithm is to be more careful when choosing the index $b_i \in B$ that is supposed to leave the basis in each iteration.

Suppose we are at the basis B and the algorithm decides to move to the basis D; that is $D = B + j - b_i$. Let $\mathbf{x}$ and $\mathbf{y}$ be the basic feasible solutions associated with the bases. Define $\mathbf{d}$ and $\theta > 0$ such that $\mathbf{y} = \mathbf{x} + \theta\mathbf{d}$. Notice that $d_j > 0$, so we can normalize the vector so that $d_j = 1$. (Why is it okay to assume that $d_j > 0$?) Also, notice that $d_k = 0$ for all $k \notin B + j$. Furthermore, since $\mathbf{x}$ and $\mathbf{y}$ are feasible, we have $\mathbf{Ad} = \mathbf{0}$. Putting everything together we get

$$\mathbf{d_B} = -\mathbf{A_B^{-1}A_j}.$$

We can imagine the algorithm is finding the largest θ such that $y \geq 0$. The means

$$\theta = \min_{b_i \in B : d_{b_i} < 0} \left| \frac{x_i}{d_{b_i}} \right|.$$

This expression is always strictly positive by our assumption that the instance in non-degenerate. Furthermore, by the same assumption, there is a unique index b_i that attains the minimum, which is the index that will leave the basis.

Finally, the change in cost in going from B to D is just

$$\mathbf{c_D} \cdot \mathbf{y_D} - \mathbf{c_B} \cdot \mathbf{x_B} = \theta(c_j + \mathbf{d_B} \cdot \mathbf{c_B}).$$

So as long the expression within parentheses is negative, we are guaranteed to improve the cost of the basic feasible solution induced by the basis. In particular, this means that we will never consider the same basis twice.

Algorithm 2 SIMPLEX-TAKE-2$(\mathbf{A}, \mathbf{b}, \mathbf{c})$

1. B = some feasible basis of A
2. // we denote the indices of B by $\{b_1, \ldots, b_m\}$
3. **repeat**
4. **for** $j \in \{1, \ldots, n\} \setminus B$ **do**
5. $\mathbf{d_B} = -\mathbf{A_B^{-1}A_j}$
6. **if** $c_j + \mathbf{d_B} \cdot \mathbf{c_B} < 0$ **then**
7. b_i = index $b_i \in B : d_{b_i} < 0$ minimizing $\left| \frac{x_i}{d_{b_i}} \right|$
8. $B = B + j - b_i$
9. **until** basis B hasn't changed in this iteration
10. **return** B

The expression in Line 6 of the Algorithm 2 is the called the *reduced cost* of j. Notice that the reduced cost of a basic variable is always 0. Therefore, we can write the above pseudo-code in a slightly more compact form.

Algorithm 3 SIMPLEX-TAKE-3$(\mathbf{A}, \mathbf{b}, \mathbf{c})$

1. B = some feasible basis of A
2. // we denote the indices of B by $\{b_1, \ldots, b_m\}$
3. **repeat**
4. $\quad \bar{\mathbf{c}}' = \mathbf{c}' - \mathbf{c}'_{\mathbf{B}}\mathbf{A}_{\mathbf{B}}^{-\mathbf{1}}\mathbf{A}$
5. $\quad$ **if** $\exists j : \bar{c}_j < 0$ **then**
6. $\quad\quad \mathbf{u} = \mathbf{A}_{\mathbf{B}}^{-\mathbf{1}}\mathbf{A}_{\mathbf{j}}$
7. $\quad\quad$ **if** $\mathbf{u} \leq \mathbf{0}$ **then**
8. $\quad\quad\quad$ **return** "unbounded"
9. $\quad\quad b_i$ = index in B such that $u_i > 0$ minimizing $\frac{x_{b_i}}{u_i}$.
10. $\quad\quad B = B + j - b_i$
11. **until** basis B hasn't changed in this iteration
12. **return** B

Theorem 4.1. *Let B be a basis such that $\mathbf{A}_{\mathbf{B}}^{-\mathbf{1}}\mathbf{b} \geq \mathbf{0}$ and $\bar{\mathbf{c}}' = \mathbf{c}' - \mathbf{c}'_{\mathbf{B}}\mathbf{A}_{\mathbf{B}}^{-\mathbf{1}}\mathbf{A} \geq \mathbf{0}$. Then the basic feasible solution induced by B is optimal.*

Proof: Let $\mathbf{x}$ be the basic feasible solution defined by B and let $\mathbf{y}$ be any other feasible solution. Consider the direction $\mathbf{d} = \mathbf{x} - \mathbf{y}$. From the feasibility of $\mathbf{x}$ and $\mathbf{y}$ it follows that

$$\mathbf{0} = \mathbf{Ad} = \mathbf{A_B d_B} + \sum_{k \notin B} \mathbf{A_k} d_k \quad \text{and thus} \quad \mathbf{d_B} = -\sum_{k \notin B} \mathbf{A}_{\mathbf{B}}^{-\mathbf{1}}\mathbf{A_k} d_k.$$

Now the different in cost between $\mathbf{y}$ and $\mathbf{x}$ is

$$\mathbf{c} \cdot \mathbf{d} = \mathbf{c}'_{\mathbf{B}}\mathbf{d_B} + \sum_{k \notin B} d_k c_k = \sum_{k \notin B} d_k (c_k - \mathbf{A}_{\mathbf{B}}^{-\mathbf{1}}\mathbf{A_k}) = \sum_{k \notin B} d_k \bar{c}_k.$$

Notice that for all $k \notin B$ we have $x_k = 0$ and $y_k \geq 0$ and therefore $d_k \geq 0$. By our assumption, all reduced costs are non-negative. Thus, $\mathbf{c} \cdot \mathbf{y} - \mathbf{c} \cdot \mathbf{x} = \mathbf{c} \cdot \mathbf{d} \geq 0$. Since this holds for all feasible $\mathbf{y}$, it follows that $\mathbf{x}$ is indeed optimal. □

Corollary 4.2. *Given that assumptions 1 through 4 hold. Algorithm 3 returns an optimal feasible solution after a finite number of steps.*