| **Optimization** | **Summer 2010** |
|---|---|

# Lecture 6 — April 28

*Lecturer: Julián Mestre*

## 6.1 Cycling

In this lecture we will remove the assumption that our instance is non-degenerate. The problem is that we cannot guarantee that the cost of the basic solution improves in each iteration since it may be the case that $\theta = 0$; recall that this fact was the basis of our argument that the algorithm terminates. Indeed, if we are not careful when choosing which non-basic variable to bring into the basis and which to remove, the algorithm can cycle for ever. Consider the following example[1] with 2 equations and 6 variables starting from the following tableau:

| | $-\mathbf{c}'\mathbf{x}$ | $\bar{c}_1$ | $\bar{c}_2$ | $\bar{c}_3$ | $\bar{c}_4$ | $\bar{c}_5$ | $\bar{c}_6$ |
|---|---|---|---|---|---|---|---|
| | 0 | -2.3 | -2.15 | 13.55 | 0.4 | 0 | 0 |
| $x_5$ | 0 | 0.4 | 0.2 | -1.4 | -0.2 | 1 | 0 |
| $x_6$ | 0 | -7.8 | -1.4 | 7.8 | 0.4 | 0 | 1 |

Suppose we decide to bring $x_1$ into the basis; then we need to remove $x_5$. The next tableau is

| | $-\mathbf{c}'\mathbf{x}$ | $\bar{c}_1$ | $\bar{c}_2$ | $\bar{c}_3$ | $\bar{c}_4$ | $\bar{c}_5$ | $\bar{c}_6$ |
|---|---|---|---|---|---|---|---|
| | 0 | 0 | -1 | 5.5 | -0.75 | 5.75 | 0 |
| $x_1$ | 0 | 1 | 0.5 | -3.5 | -0.5 | 2.5 | 0 |
| $x_6$ | 0 | 0 | 2.5 | -19.5 | -3.5 | 19.5 | 1 |

Suppose we decide to bring $x_2$ into the basis; then we can remove either index in the basis, suppose that we remove $x_6$. The next tableau is

[1] The example is taken from a paper by Hall and McKinnon, Math. Prog. B 100: 133-150 (2004)

| | $-\mathbf{c}'\mathbf{x}$ | $\bar{c}_1$ | $\bar{c}_2$ | $\bar{c}_3$ | $\bar{c}_4$ | $\bar{c}_5$ | $\bar{c}_6$ |
|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | -2.3 | -2.15 | 13.55 | 0.4 |
| $x_1$ | 0 | 1 | 0 | 0.4 | 0.2 | -1.4 | -0.2 |
| $x_2$ | 0 | 0 | 1 | -7.8 | -1.4 | 7.8 | 0.4 |

We have not cycled yet, but a quick look at the third tableau reveals that in fact can be obtained from the first by shifting two columnsto the right. Therefore, if we apply the same shift to our choices for the first two, we will end up with another table whose columns are shifted two positions further to the right. Repeating the operation one last time takes us back to the starting tableau and thus finishes the cycle.

## 6.2 Bland's anti-cycling rule

Luckily there are many anti-cycling rules to avoid the situation described above. Here we will cover Bland's rule, which happens to be very simple to state and implement. The rule basically says that whenever we have a choice between two or more indices, we should select the smallest one.

**Theorem 6.1.** *Suppose the simplex algorithm chooses the column to enter the basis by*

$$j = \min \{ j \, : \, \bar{c}_j < 0 \},$$

*and the row by*

$$i = \operatorname{argmin}_i \left\{ b_i \in B \, : \, u_i > 0 \text{ and } \frac{x_{b_i}}{u_i} \leq \frac{x_{b_k}}{u_k} \text{ for all } k \text{ such that } u_k > 0 \right\}.$$

*Then simplex terminates after a finite number of steps even in the presence of degeneracy.*

**Proof:** For the sake of contradiction assume that the algorithm cycles when following Bland's pivoting rule. Furthermore, assume we start from a basis in the cycle. This means that the cost of the solution never improves and that the basic feasible solution induced by the bases in the cycle is the same; in particular the zeroth column of the tableaux we cycle through never changes.

To facilitate the analysis, let us remove those rows and columns that never contain pivots in the cycle; that is, we remove those variables that are always either basic or non-basic in all bases. We have not lost anything, since if we run the algorithm in this reduced instance we would get the same cycle. If a variable $x_k$ is ever kicked out of the basis, we must have $x_k = 0$ for otherwise $\theta > 0$. The basic feasible solution induced in the reduced instance is the vector $\mathbf{0}$ and Bland's rule for selecting the basic variable $b_i$ to leave the basis reduces to picking the smallest $b_i$ such that $u_i > 0$.

Let $q$ be the largest index ever to have entered and left the basis. Let $T_1$ be the tableau the algorithm had when $q$ was brought into the basis and $T_2$ be the tableau the algorithm had when $q$ was removed and replaced by some index $p$. Let $B$ and $D$ be the bases associated with $T_1$ and $T_2$, and let $\bar{\mathbf{c}}$ and $\hat{\mathbf{c}}$ be the corresponding reduced cost vectors. The "direction of improvement" $\mathbf{d}$ that simplex uses in the iteration of $T_2$ is

$$d_j = \begin{cases} 1 & \text{if } j = p, \\ -u_i & \text{if } j = b_i, \\ 0 & \text{otherwise.} \end{cases}$$

where $\mathbf{u} = \mathbf{A_D^{-1} A_p}$. Recall that $\mathbf{Ad} = \mathbf{0}$. It follows that

$$\bar{\mathbf{c}}'\mathbf{d} = (\mathbf{c}' - \mathbf{c}'_\mathbf{B}\mathbf{A}_\mathbf{B}^{-1}\mathbf{A})\mathbf{d} = \mathbf{c}'\mathbf{d} = c_p - \mathbf{c}'_\mathbf{D}\mathbf{u} = c_p - \mathbf{c_D}\mathbf{A}_\mathbf{D}^{-1}\mathbf{A_p} = \hat{c}_p < 0$$

where the inequality follows from the fact that $p$ enters the basis in the iterations of $T_2$.

We will now show that $\bar{\mathbf{c}}'\mathbf{d} > 0$, thus reaching a contradiction. Because $q$ was chosen to leave the basis at $T_2$, we know that $d_q$ is the only negative coordinate of $\mathbf{d}$:

$$d_k \geq 0 \text{ for all } k < q \text{ and } d_q < 0.$$

Also, because $q$ was brought into the basis at $T_1$, we know $\bar{c}_q$ is the only negative coordinate of $\bar{\mathbf{c}}$:

$$\bar{c}_k \geq 0 \text{ for all } k < q \text{ and } \bar{c}_q < 0.$$

Putting these two facts together we reach the desired conclusion that $\bar{\mathbf{c}}'\mathbf{d} > 0$, which contradicts our previous derivation. Thus, our assumption that simplex can cycle under the Bland's rule must have been wrong. □