MS-E2121 Homework 3

Tuomas Porkamaa

February 28, 2022

1 Implementing the revised simplex method

Our running example of revised simplex is initialized as follows:

$$
\begin{aligned}
A &= \begin{pmatrix} 2 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 2 & 3 & 0 & 0 & 1 \end{pmatrix} \\
b &= \begin{pmatrix} 5 & 1 & 12 \end{pmatrix}^T \\
c &= \begin{pmatrix} -5 & -1 & 0 & 0 & 0 \end{pmatrix}^T \\
I_B &= \{1, 2, 4\} \\
B &= \begin{pmatrix} 2 & 1 & 0 \\ 0 & 1 & -1 \\ 2 & 3 & 0 \end{pmatrix} \\
B^{-1} &= \begin{pmatrix} 0.75 & 0 & -0.25 \\ -0.5 & 0 & -0.5 \\ -0.5 & -1 & 0.5 \end{pmatrix} \\
x &= \begin{pmatrix} 0.75 & 3.5 & 0.0 & 2.5 & 0.0 \end{pmatrix}^T \\
x_B &= \begin{pmatrix} 0.75 & 3.5 & 2.5 \end{pmatrix}^T
\end{aligned}
\tag{1}
$$

1.1 `iteration_condition`

This function returns *false* if all reduced costs are nonnegative, i.e., the current solution is optimal and the algorithm should terminate, *true* otherwise. For example, the reduced cost vector [0.0, 6.5, 2.5] would return *false*, while [-3.25, 0.0, -0.75] would return *true*.

1.2 `calculate_cr`

This function calculates and returns the reduced costs that are later used to check if the simplex method is converged, or is the problem unbounded, see 1.1. The nonbasic variable that becomes basic is also decided based on reduced

costs. Function implementation calculates reduced costs by

$$\bar{c} = c - c_B^T B^{-1} A$$

For example scenario (1), we obtain $\bar{c} = (0.0, 0.0, -3.25, 0.0, -0.75)^T$

1.3 `select_entering_variable`

This function implements two rules, Dantzig's and Bland's, that decide which one of the nonbasic variables becomes basic. Function returns the index j of the corresponding variable. In Dantzig's rule we select the nonbasic variable that have the most negative reduced cost.

$$j = \text{argmin} \left\{ \bar{c}_j \right\}_j$$

In Bland's rule we select the nonbasic variable with smallest index $j \in I_N$ and negative reduced cost.

$$j = \min_{\bar{c}_j < 0} I_N$$

For $\bar{c} = (0.0, 0.0, -3.25, 0.0, -0.75)^T$, both rules would would select $j = 3$.

1.4 `calculate_reverse_basic_directions`

This function returns the opposite of the basic feasible direction d_B

$$u = -d_B = B^{-1} A_j$$

where j is obtained from (1.3). This vector is later used to move x to the adjacent solution. It is also used when computating the step size to reach the adjacent solution and when selecting the variable leaving the basis.

For $j = 3$, we obtain $u = B^{-1} A_3 = (-0.75, 0.5, 0.5)^T$

1.5 `calculate_step_length`

This function returns an optimal step size $\bar{\theta}$ such that one can move furthest along direction $d = [d_B, d_N] = [-u, d_N]$ while remaining feasibility $x + \bar{\theta} d \geq 0$. $\bar{\theta}$ is calculated as

$$\bar{\theta} = \min_{i \in I_B, u_i > 0} \left\{ \frac{x_{B(i)}}{u_i} \right\}$$

For $u = (-0.75, 0.5, 0.5)^T$, we obtain $\bar{\theta} = \min \left\{ \frac{3.5}{0.5}, \frac{2.5}{0.5} \right\} = 5$.

1.6 `select_leaving_variable`

This function returns the index $l \in I_B$ of the basic variable that will become nonbasic.

$$l = \underset{i \in I_B, u_i > 0}{\arg\min} \left\{ \frac{x_{B(i)}}{u_i} \right\}$$

In the case of ties, i.e. if there exists l_1 and l_2 such that $\bar{\theta}_{l_1} = \bar{\theta}_{l_2}$, we use Bland's rule and select l_1.

For $u = (-0.75, 0.5, 0.5)^T$, we obtain $l = \arg\min \left\{ \cdot, \frac{3.5}{0.5}, \frac{2.5}{0.5} \right\} = 3$. This means that variable x_4 ($I_B[3] = 4$) will become nonbasic. The "$\cdot$" correspond to negative component of the direction that is ignored.

1.7 `update_x!`

This function updates x to move to the adjacent solution, because current solution is not optimal. For the selected nonbasic variable (1.3) to become basic we set it to equal to the optimal step size (1.5)

$$x_j = \bar{\theta}$$

The variables related to the current basis are updated by moving along the basic feasible direction (1.4) by the amount of optimal step size (1.5).

$$x_B = x_B - \bar{\theta} u$$

For $j = 3$ and $\bar{\theta} = 5$, we set $x_3 = 5$.
For $u = (-0.75, 0.5, 0.5)^T$, we obtain $x_B = (4.5, 1.0, 0.0)^T$.
This makes $x = (4.5, 1.0, 5.0, 0.0, 0.0)^T$.

1.8 `update_B_ind!`

This function updates the basis to respond the modifications done in 1.7. Because one variable enters (1.3, index j) and one variable leaves (1.6, index l) the basis, we update basis indices I_B by replacing l with j

For $l = 3$ and $j = 3$ we set $I_B[3] = 3$. The new basis index set is then $I_B = \{1, 2, 3\}$.

1.9 `update_inv_B!`

This function updates the B^{-1} to $\bar{B}^{-1}$ correspond to the newly formed basis. Because revised simplex keeps track of B^{-1} continuously for computational reasons (it is needed to compute both $\bar{c}$ and u), we have to update it such that it is available at the beginning of the next iteration. $\bar{B}^{-1}$ is obtained by performing elementary row operations on B^{-1}, i.e.

1. $\forall i \neq l : B^{-1}[i,:] = B^{-1}[i,:] - \frac{u_i}{u_l} * B^{-1}[l,:]$
2. $B^{-1}[l,:] = u_l^{-1} B^{-1}[l,:]$

After ERO, new basis inverse matrix equals to

$$\bar{B}^{-1} = \begin{pmatrix} 0 & -1.5 & 0.5 \\ 0 & 1 & 0 \\ -1 & -2 & 1 \end{pmatrix}$$

After 1.9 new iteration begins, and we notice that $\bar{c} = (0, 0, 0, 6.5, 2.5)^T$ which indicates that revised simplex terminates with an optimal solution $x = (4.5, 1, 5, 0, 0)^T$.

2 Implementing the two-phase simplex method

The LP problem in hand was formulated as

$$\begin{aligned} \text{(P): max. } & 5x_1 + x_2 \\ \text{s.t.} & \\ & 2x_1 + x_2 \geq 5 \\ & x_2 \geq 1 \\ & 2x_1 + 3x_2 \leq 12 \\ & x_1, x_2 \geq 0 \end{aligned} \tag{2}$$

which can be formulated to standard form as

$$\begin{aligned} \text{(P): min. } & -5x_1 - x_2 \\ \text{s.t.} & \\ & 2x_1 + x_2 - x_3 = 5 \\ & x_2 - x_4 = 1 \\ & 2x_1 + 3x_2 + x_5 = 12 \\ & x_1, ..., x_5 \geq 0 \end{aligned} \tag{3}$$

If we make x_3, x_4, x_5 the first basic variables, this would yield infeasible basic solution. Hece we use two phase simplex to solve this problem, where the first phase finds suitable BFS for the problem. First phase solves the following auxilary problem

$$\begin{aligned} \text{(AUX): min. } & x_6 + x_7 + x_8 \\ \text{s.t.} & \\ & 2x_1 + x_2 - x_3 + x_6 = 5 \\ & x_2 - x_4 + x_7 = 1 \\ & 2x_1 + 3x_2 + x_5 + x_8 = 12 \\ & x, y \geq 0 \end{aligned} \tag{4}$$

We can construct the parameter setting for AUX by modifying the original setting in P. Parameter b stays the same and c considers only artificial variables.

For constraints, we set $A_{AUX} = [A_P|I]$ that holds for every LP. For the exercise problem, following AUX parameterizations are obtained:

$$A_{AUX} = \begin{pmatrix} 2 & 1 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 1 & 0 \\ 2 & 3 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

$$b = \begin{pmatrix} 5 & 1 & 12 \end{pmatrix}^T$$

$$c = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}^T$$

In notebook, the AUX-parameters are set in *two_phase* function in lines 13-15. The artificial variable x_8 and column $A_8 \in A_{AUX}$ are not necessarily needed because slack variable x_5 is positive, but for simplicity in implementation I decided to add artificial variable for each constraint.

Minimizing AUX yields an optimal solution $\hat{x}_{AUX}$ and corresponding basis B_{AUX}. If $\hat{x}_{AUX,6}, \hat{x}_{AUX,7}, \hat{x}_{AUX,8} \neq 0$, there is no BFS for P indicating infeasibility. If this is not the case and B_{AUX} does not contain columns associated with artificial variables, we can use B_{AUX} as an initial basis for P. If optimal AUX basis have artificial variable associated columns, an additional preprocessing step is needed but this functionality is not implemented.

Finally we can solve P with original A, b and c where we use B_{AUX} as an initial basis.

3 Comparing variable selection rules

The optimal solutions (x-values and objective values) for the given problems are given in Table 1.

Table 1:

Problem	x	$c^T x$
a)	(4, 4, 4)	136
b)	(0, 0, 12, 8)	20
c)	(1.67, 6.67, 0, 0)	-11.67

The number of iterations required to find an optimal solution in each of the given problems is given in Table 2.

Table 2:

Problem	k_{bland}	$k_{dantzig}$
a)	3	3
b)	7	5
c)	3	4

Results of Table 2 indicate that for a) both impelemtations terminated in equal iterations, in b) the Dantzig's rule implementation terminated faster and in c) the Bland's rule implementation terminated faster.