User Guide: the Simplex Method in Matlab

The Simplex method is an algorithm designed to solve linear optimization problems, which are referred to as linear programs (LPs). The purpose of this guide is to explain how to use and understand the provided Matlab code to solve any LP that is written in canonical form, which will be explained using the following example problem (problem 7 from homework 5).

$$\begin{aligned} \text{maximize} \quad & 6x_1 + 5x_2 \\ \text{subject to} \quad & 5x_1 + 2x_2 \leq 34, \\ & x_1 - x_2 \geq -3, \\ & x_1, x_2 \geq 0. \end{aligned} \tag{1}$$

Canonical form is an alternative way of writing an LP that makes use of only equality constraints. Any equality constraints in the original LP are unchanged, however, if a problem contains a less than or equal to constraint, we introduce a slack variable, s_i, to represent all possible solutions that satisfy only the "less than" condition. For the problem above, we would write its first constraint as follows:

$$\begin{aligned} 5x_1 + 2x_2 + s_1 &= 34, \\ s_1 &\geq 0. \end{aligned}$$

Notice that we have added s_1 in this constraint. This is because, to ensure equality, a positive value, s_1 in this case, is added to all solutions that satisfied the "less than" portion of the constraint in (1) above. Similarly, we do the opposite for greater than or equal to constraints, as in the second constraint of (1) above:

$$\begin{aligned} x_1 - x_2 - s_2 &= -3, \\ s_2 &\geq 0. \end{aligned}$$

Therefore, our constraints now looks like this:

$$\begin{aligned} \text{subject to} \quad & 5x_1 + 2x_2 + s_1 = 34, \\ & x_1 - x_2 - s_2 = -3, \\ & x_1,\ x_2,\ s_1,\ s_2 \geq 0. \end{aligned}$$

Using these rules, we can now begin to formulate our inputs for the Simplex algorithm in Matlab. The algorithm requires three inputs: the constraint matrix, A, the right hand side coefficients, d, and the objective vector, c. Using the problem above, we define A, d, and c as follows:

$$A = \begin{bmatrix} 5 & 2 & 1 & 0 \\ 1 & -1 & 0 & -1 \end{bmatrix}, d = \begin{bmatrix} 34 \\ -3 \end{bmatrix}, c = \begin{bmatrix} 6 \\ 5 \\ 0 \\ 0 \end{bmatrix}.$$

We have zeros for the final two entries of our c vector because the values of s_1 and s_2 do not affect our objective function.

To execute our Matlab package, first, ensure that all 9 files from the folder titled "SimplexMethod-master" are in your active matlab directory. To add these files to your active directory, select the file titled "simplexMethod.m" and press the green play button with three blue dots under it. In the window that pops up, select "add to path." This registers our program into your Matlab session so we can solve the LP we set up above. Once this is done, we can begin to input our LP. To do so, we need Matlab to have these matrices and vectors already stored as A, d, and c before we can execute our program. To do so, enter the following syntax into your command window/script window, pressing enter after each line to register each command into Matlab:

```
A = [5 2 1 0; 1 -1 0 -1]
d = [34;-3]
c = [6;5;0;0]
```

Once these have been successfully stored in Matlab, we can now execute our algorithm using the following command sequence:

```
[optimum, solution] = simplexMethod(c,A,d);
optimum
solution
```

Your command window in Matlab should look like this:

```
>> [optimum, solution] = simplexMethod(c,A,d);

>> optimum

optimum =

59

>> solution

solution =

4.0000
7.0000
0
0
```

If it does not, and an error message does not appear, check that everything has been typed properly into Matlab. The first output is the value of our objective function at the optimum, while the second output tells us the coordinates of the optimal solution in terms of our variables x_1,x_2, s_1, and s_2. This is our final answer for this problem, but there are several types of problems our package can handle.

It is important to note that, while our package is designed to let you know if the LP is unbounded or infeasible, it does assume that the origin is a basic feasible solution (i.e. corner point) to the LP. If an LP without the origin as a basic feasible solution is entered, our algorithm will throw an error message. Otherwise, this method will find the optimal solution for any LP in canonical form.

Lastly, the following is the list of names of the members of our group, followed by our primary and secondary roles for this project.

Chris Andersen: primary role: Technical Writing, secondary role: Documentation.
Jack Monaghan: primary role: Testing and User Interface, secondary role: Technical Writing.
Brad Olson: primary role: Backend Implementation, secondary role: Testing and User Interface.
Derek Albosta: primary role: Documentation, secondary role: Backend Implementation.