A TUTORIAL APPLET FOR SIMPLEX METHOD

Pablo Zafra and Henry Cleary
Kean University
Department of Mathematics
1000 Morris Avenue, Union, NJ 07083
pzafra@kean.edu and clearyh@verizon.net

Abstract

We will demonstrate an interactive Java program called *Simplify* that can be used by anyone interested in learning and exploring the simplex method for solving linear programming problems. It is valuable as a study tool to help reinforce material and prepare for exams and assignments on the manual calculation of simplex tableaux.

I. Introduction

We developed an interactive Java computer program called *Simplify*, which is available as a web applet or as a stand alone application. Simplify is intended for students and anyone interested in learning about the simplex method for solving linear programming (LP) problems. LP problems are usually covered in undergraduate mathematics courses such as finite math, mathematical modeling, linear algebra and operations research.

In order to demonstrate the simplex algorithm, Simplify provides dynamic and colorful graphical displays for a wide of array linear programming example problems for both maximization and minimization objective functions. These include special LP cases involving unbounded problems, degeneracy, cycling and alternate optimal solutions.

The program provides full optimization (automatic) mode if the goal is simply to obtain the optimal solution without seeing all the individual steps. However, it is most useful as a study tool when single iteration (manual) mode is invoked, as it displays the resulting simplex tableaus for all iterations that take place. In addition, it allows users to modify the entering column in each step for further exploration, and to compare with the algorithm's original selection of entering variables.

Simplify can therefore be used as a valuable study aid to help reinforce material and prepare for exams and assignments on the manual calculation of simplex tableaux. It also serves as a validation tool to verify students' work on simplex tableaus, and by extension it also lessens their frustration when arithmetic errors are introduced since they can easily compare their own work with Simplify's results. As a teaching tool, Simplify helps in speeding up the discussion of simplex method by avoiding tedious calculations, that would otherwise have to be done by hand.

II. Linear Programming Problem Model

A linear programming (LP) problem consists of a linear objective function in n variables (called decision variables) that we wish to find the maximum (or minimum) value subject to a set of m linear inequalities involving these variables as constraints. Symbolically, a typical LP problem looks like as follows:

$$\begin{aligned}
&\text{Max} \quad z = c_1x_1 + c_2x_2 + \ldots + c_nx_n \\
&\textit{subject to} \\
&a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leq b_1 \\
&a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \leq b_2 \\
&\vdots \\
&a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \leq b_m \\
&\text{all } x_i\text{'s} \geq 0
\end{aligned}$$

Figure 1. LP Problem Model

It is further assumed that the right hand side constants are all non-negative. Generally, the inequalities can be a mix of greater than, less than or equality. However, Simplify currently requires the inequalities as shown in figure 1. We hope to incorporate other types of inequalities in a future version of Simplify.

III. The Simplex Method

The Simplex method was developed and introduced in 1947 by George Dantzig, known as the "Father of Operations Research", as a method of solving linear programming problems. In order to apply the algorithm, one needs to convert first the LP model (figure 1) in a so-called standard form of LP. In the standard form, the constraints consists of equations and the objective function is written in the form $z - c_1x_1 - c_2x_2 - \ldots - c_nx_n = 0$. In order to have equations in our model, we need to add a separate variable, known as slack variable, to the left of each inequality. Figure 2 shows an example of an LP problem given in original form and its standard form:

$$\begin{aligned}
&\text{Max} \quad z = 60x_1 + 28x_2 + 16x_3 \\
&\textit{subject to} \\
&(1)\ 8x_1 + 6x_2 + 2x_3 \leq 42 \\
&(2)\ 2x_1 + 4x_2 + 5x_3 \leq 10 \\
&(3)\ 2x_1 + 3x_2 + 0.5x_3 \leq 8 \\
&(4)\ x_2 \leq 5 \\
&x_1, x_2, x_3 \geq 0
\end{aligned}$$

Figure 2a. LP in original form

$$\begin{aligned}
&\text{Max} \quad z - 60x_1 - 28x_2 - 16x_3 = 0 \\
&\textit{subject to} \\
&(1)\ 8x_1 + 6x_2 + 2x_3 + s_1 = 42 \\
&(2)\ 2x_1 + 4x_2 + 5x_3 + s_2 = 10 \\
&(3)\ 2x_1 + 3x_2 + 0.5x_3 + s_3 = 8 \\
&(4)\ x_2 + s_4 = 5 \\
&x_1, x_2, x_3, s_1, s_2, s_3, s_4 \geq 0
\end{aligned}$$

Figure 2b. LP in standard form

The standard form of the LP is then used to create the initial simplex tableau as shown in figure 3 for our example. Note that in figure 3, the objective function is designated as row 0. The basic feasible solution (bfs) can be seen in the column rhs. The simplex method then proceeds by applying rules to:

- determine if the current bfs is optimal
- and if not, determine entering basic variable and leaving basic variable

Row	z	x1	x2	x3	s1	s2	s3	s4	rhs (basic)	Test ratios
0	1	-60	-28	-16	0	0	0	0	0	
1	0	8	6	2	1	0	0	0	s1=42	42/8=5.25
2	0	2	4	3	0	1	0	0	s2=10	10/2=5
3	0	2	3	0.5	0	0	1	0	s3=8	8/2=4
4	0	0	1	0	0	0	0	1	s4=4	NA

Figure 3. Initial simplex tableau

- then use Gaussian elimination to find new bfs with better objective function, and repeat steps until optimal

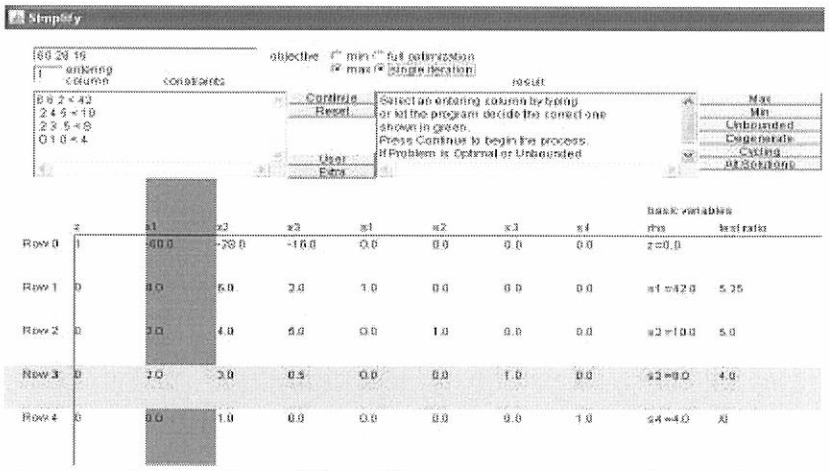
Details of simplex method can be found in any standard Operations Research book.

In figure 3, the grayed-out column x1 and row 3 indicate the entering basic variable x1 and the leaving basic variable s3 (the slack variable corresponding to row3). Variable x1 is chosen to enter because of its most negative coefficient (-60) in row 0, and it will replace variable s3 because row 3 has the least test ratio (rhs over column x1).

Gaussian elimination would then be applied as the next step using the highlighted entry 2 (intersection of column x1 and row 3) as pivot element. This is the step that students find intimidating if not tedious, and the cause of many frustrations due to simple arithmetic errors. Simplify can help these students by assisting in executing the steps of simplex method, and students can then either compare or reproduce results generated by Simplify.

IV. Simplify: a Java program

Simplify is an interactive learning tool written in Java that implements the Simplex method. It is available as a web-based applet for Java-enabled browser or a stand-alone application that can be downloaded to a pc. Both versions have the same user interface as shown in figure 4, which shows the initial simplex tableau of our example in figure 3.



60 28 16

1 entering column

objective ( ) min [✓ max] ( ) full optimization [✓ single iteration]

constraints

```
8 6 2 < 42
2 4 3 < 10
2 3 .5 < 8
0 1 0 < 4
```

result

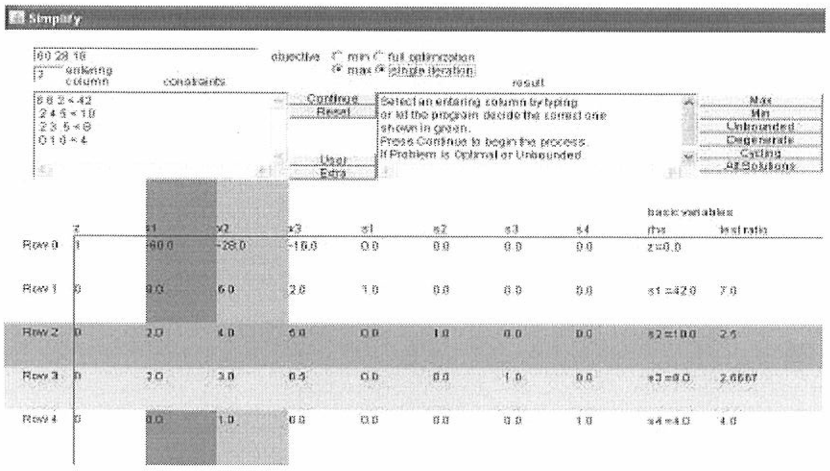
```
Select an entering column by typing
or let the program decide the correct one
shown in green.
Press Continue to begin the process.
If Problem is Optimal or Unbounded
```

	z	x1	x2	x3	s1	s2	s3	s4	basic variables rhs	test ratio
Row 0	1	-60.0	-28.0	-16.0	0.0	0.0	0.0	0.0	z=0.0	
Row 1	0	8.0	6.0	2.0	1.0	0.0	0.0	0.0	s1=42.0	5.25
Row 2	0	2.0	4.0	3.0	0.0	1.0	0.0	0.0	s2=10.0	5.0
Row 3	0	2.0	3.0	0.5	0.0	0.0	1.0	0.0	s3=8.0	4.0
Row 4	0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	s4=4.0	[illegible]

Figure 4: Screen shot of Simplify. Note that the result box displays useful information and messages regarding the problem at hand.

To demonstrate the Simplex method, Simplify has preloaded LP example problems, including special cases of unboundedness, alternative optimal solution, cycling and degeneracy. Each sample problem can be invoked by simply pressing the appropriate button from the array located on the upper right side of the screen. A user supplied LP problem can be entered by pressing the reset button, and following the input format as illustrated by any of the sample problems. Note that only the coefficients of the decision variables are needed for input, and any decision variable that that is "missing" in a constraint has to be noted with a coefficient of 0 as input; this will maintain the one to one correspondence between the number of coefficients entered in the objective function box and the constraints box. Currently, Simplify accepts only less than or equal constraints with a positive right hand side, and fractional coefficients are also not acceptable as input. We hope to address these limitations in future version of Simplify.

Simplify solves a max or min problem in either full optimization (automatic) or single iteration (manual) mode. In full optimization mode, Simplify takes a tableau and when users press "Continue", only the final optimal tableau will be displayed. Single iteration mode allows an iteration to take place each time "Continue" is pressed. There is also a notable feature of Simplify which allows users to modify the selection of entering columns in the single iteration mode. First the program picks out an entering column (shown in green) and a leaving row (shown in yellow) using the standard simplex operations. If a user presses "Continue," the indicated iteration will occur. Users can choose to ignore the computer's pivot choice and pick their own entering column by typing it into the "entering column" text field. The new chosen column then appears in orange. The ratio test is performed and the new leaving row is selected and shown in blue. For users unable to get immediate display changes for a different entering value, they may press "User" to remedy the problem. Once the new column and row are displayed, pivot operations will be performed upon pressing "Continue". Figure 5 displays the result when column 2 was chosen as entering column in our example.



60 28 16

2 entering column

constraints

```
8 6 2 < 42
2 4 5 < 10
2 3 .5 < 8
0 1 0 < 4
```

result

```
Select an entering column by typing
or let the program decide the correct one
shown in green.
Press Continue to begin the process.
If Problem is Optimal or Unbounded
```

	z	x1	x2	x3	s1	s2	s3	s4	basic variables rhs	test ratio
Row 0	1	-60.0	-28.0	-16.0	0.0	0.0	0.0	0.0	z=0.0	
Row 1	0	8.0	6.0	2.0	1.0	0.0	0.0	0.0	s1 =42.0	7.0
Row 2	0	2.0	4.0	5.0	0.0	1.0	0.0	0.0	s2 =10.0	2.5
Row 3	0	2.0	3.0	0.5	0.0	0.0	1.0	0.0	s3 =8.0	2.6667
Row 4	0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	s4 =4.0	4.0

Figure 5. User-selected entering column

V. Special LP Cases

For alternate solutions, the program will look at the zero row of an optimal solution. If all but one x variable columns are zero in row 0, then the program will search the columns for a column with no pivot (that is having more than one nonzero values; this will be the new entering column). The program then performs the ratio test and picks a leaving row. It notifies the user to press the "Extra" button. When the user does so, row operations are performed to make the entering column one in the leaving row and zero elsewhere. Simplify then displays a new optimal tableau in the display area.

In Simplify, unbounded problems are detected when we attempt to select an entering column where there are all non-positive entries below row 0. The column indicating unboundedness is highlighted in the graphical display. Users are then informed to reset and try another problem.

Using Simplify in single iteration mode, cycling can be observed and remedied depending on what the user knows. In full optimization mode, there really is no cycling handling capability. Instead, when the count of simplex iterations becomes unreasonably large as though an infinite loop is taking place, all calculation is stopped. Simplify then informs the user that the large number of iterations probably indicates cycling.

VI. Conclusion

Simplify is designed primarily as a tutorial and teaching aid for learning Simplex method using small sizes of LP problems. As such, it is not our intention to compete with commercial software which can handle large scale problems.

We hope that Simplify will help in students' understanding of the inner workings of the simplex method through self-exploration, and that classroom teaching will be enhanced by focusing more on the various concepts related to simplex method rather than the tedious arithmetic of matrix row operations. For example, by simply selecting different entering column in Simplify, cycling can be avoided for some LP. One can also investigate the effect on the total number of iterations needed to achieve the optimal solution when entering column is modified at each iteration, which is easily done with Simplify but very time consuming when done by hand.

Interested users may explore Simplify by accessing the applet at:

http://lobo.kean.edu/simplify

We welcome any feedback and suggestions.

VII. References

Gass, Saul I. Linear Programming: Methods and Applications, 5th edition. New York: McGraw-Hill, 1985.

Winston, Wayne. Operations Research. Belmont, California: Brooks Cole – Thomson Learning, 2004.