INDR 262

Introduction to

Operation Research

Revised Simplex Method in Java

Berk Kaan Kuguoglu

Spring 2016

Background

The revised simplex method is built upon the matrix form of the simplex method, though it varies from the generic simplex method with the way that it iterates to solve the given Linear Programming problem. The most fundamental difference between these two simplex methods is that the revised simplex method only updates **B** $^{-1}$ from the values got from the previous iteration, whereas the original simplex method inverts the **B** after each iteration and storing all the data in the matrix. However, they both solve the linear programming methods in a similar manner, but with different computation and space complexities. As the revised simplex method yields less computation and space complexity for the linear programming problem, in terms of computational efficiency, it may be seen superior to the generic simplex method.

To describe this procedure formally, let x_k entering basic variable, a_{ik} coefficient of x_k in current Eq. (i), for i 1, 2, . . . , m (identified in step 2 of an iteration),

r number of equation containing the leaving basic variable.

Recall that the new set of equations [excluding Eq. (0)] can be obtained from the preceding set by subtracting a_{ik}/a_{rk} times Eq. (r) from Eq. (i), for all i 1, 2, . . . , m except i r, and then dividing Eq. (r) by a . Therefore, the element in row i and column j of **B** $^{-1}$ is

$$(\mathbf{B}_{\text{new}}^{-1})_{ij} = \begin{cases} (\mathbf{B}_{\text{old}}^{-1})_{ij} - \dfrac{a'_{ik}}{a'_{rk}}(\mathbf{B}_{\text{old}}^{-1})_{rj} & \text{if } i \neq r, \\ \dfrac{1}{a'_{rk}}(\mathbf{B}_{\text{old}}^{-1})_{rj} & \text{if } i = r. \end{cases}$$

These formulas are expressed in matrix notation as

$$\mathbf{B}_{\text{new}}^{-1} = \mathbf{E}\mathbf{B}_{\text{old}}^{-1},$$

where matrix **E** is an identity matrix except that its *r*th column is replaced by the vector

$$\boldsymbol{\eta} = \begin{bmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_m \end{bmatrix}, \qquad \text{where} \qquad \eta_i = \begin{cases} -\dfrac{a'_{ik}}{a'_{rk}} & \text{if } i \neq r, \\ \dfrac{1}{a'_{rk}} & \text{if } i = r. \end{cases}$$

Thus, **E** [$\mathbf{U}_1$, $\mathbf{U}_2$, . . . , $\mathbf{U}_{r\,1}$, , $\mathbf{U}_{r\,1}$, . . . , $\mathbf{U}_m$], where the *m* elements of each of the $\mathbf{U}_i$ column vectors are 0 except for a 1 in the *i*th position. ([1])

Implementation

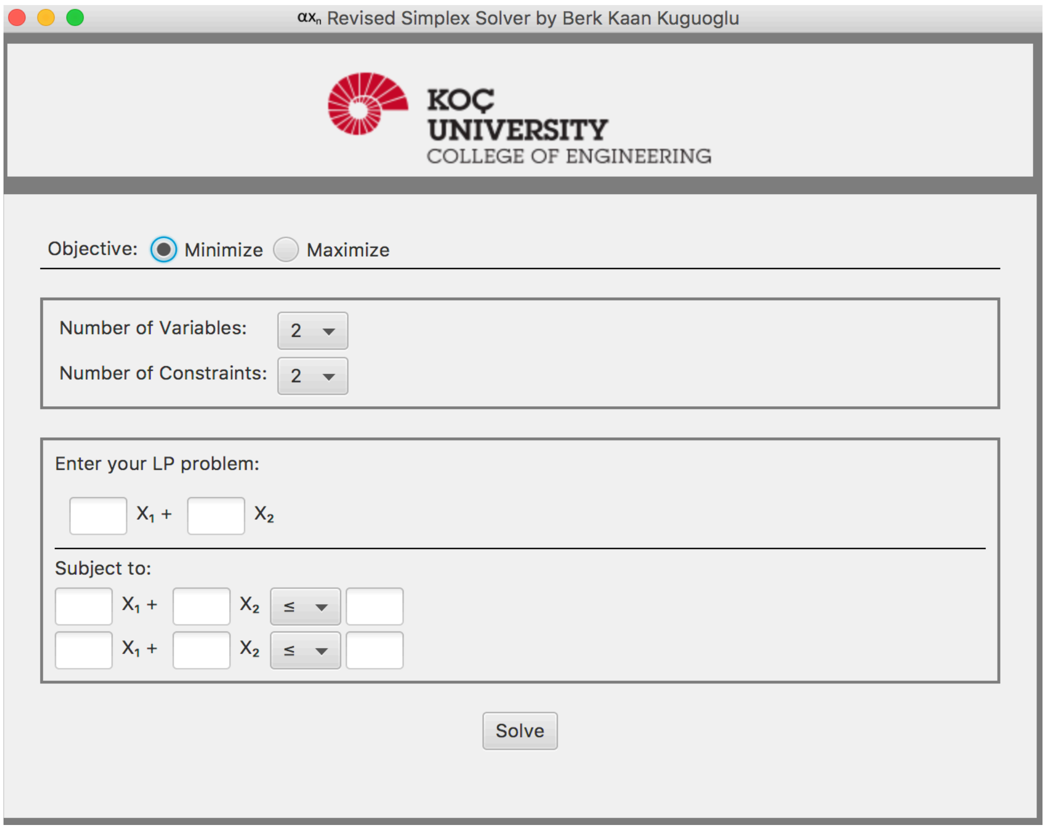
Java is the programming language that is chosen for the implementation since it offers wide range of libraries for mathematical functions and, more importantly, an application which is compatible to various platforms. Speaking of Java libraries available, the GUI of the project is written by using JavaFX libraries and their functionalities offered with related packages. The scope of the project is comparatively small, nevertheless it was one of the main goals to create a user-friendly application that mostly satisfies main UI principles, namely structure, simplicity, visibility, feedback, tolerance, and reusability principles([2]).

The application has two main frames, namely *Data Entry Screen* and *Revised Simplex Screen (see Appendix)*, that operates seamlessly integrated to each other. Firstly, user needs to define a set of variables and constraints, and also corresponding coefficients in the constraints matrix and in the row 0. Then, it can be accessed to the next screen, *Revised Simplex Screen*, by clicking the button called *Solve*. The next frame provides user a compact and very informative view of the revised simplex method by showing each component and step of the method explicitly on the screen. Whereas a user can choose to iterate step by step, it is also possible to do a full iteration in which all the steps taken in one iteration are done at once. Similar to these two functionalities, quitting the program is made possible by adding a *Quit* button on the right bottom panel, which simply holds all the buttons on the screen.

Lastly, all the methods and classes are defined and documented by using Java Doc comments, so any developer or engineer can easily grasp the basic notions of the implementation. Moreover, the source code of the project can be found and also downloaded from my GitHub account, called ‘*bkaankuguoglu’*. There are multiple number of existing projects and algorithms that gave me a solid basis for the implementation. The project hereby acknowledges and references all the work has been used by means of both algorithms and software design. Although not all of them are mentioned in this script, the resources that help the project grow in terms of design and implementation, namely source codes and algorithms in the textbooks, are referred in the *References* section.

Conclusion

In conclusion, despite the algorithm for the revised simplex method may seem quite trivial and intuitive for those who have a comprehensive knowledge on the generic simplex method and linear programming methods, the implementation of the project based on an object-oriented language, Java, has been a quite challenge throughout the creation process. Speaking of difficulties encountered during the implementation phase, the design and the creation of the GUI for the application stands out as the most challenging part in the project, nonetheless the final artifact proves all the effort and time spent are paid off. All the sources referred and used in the project are listed in the Reference section and their credits were given. Hopefully, my project will help students who’re passionately looking for a simplex solver desktop application for their studies, or, in general, individuals who are willing to investigate on the revised simplex method.



Objective: [✓ Minimize] Maximize

Number of Variables: 2

Number of Constraints: 2

Enter your LP problem:

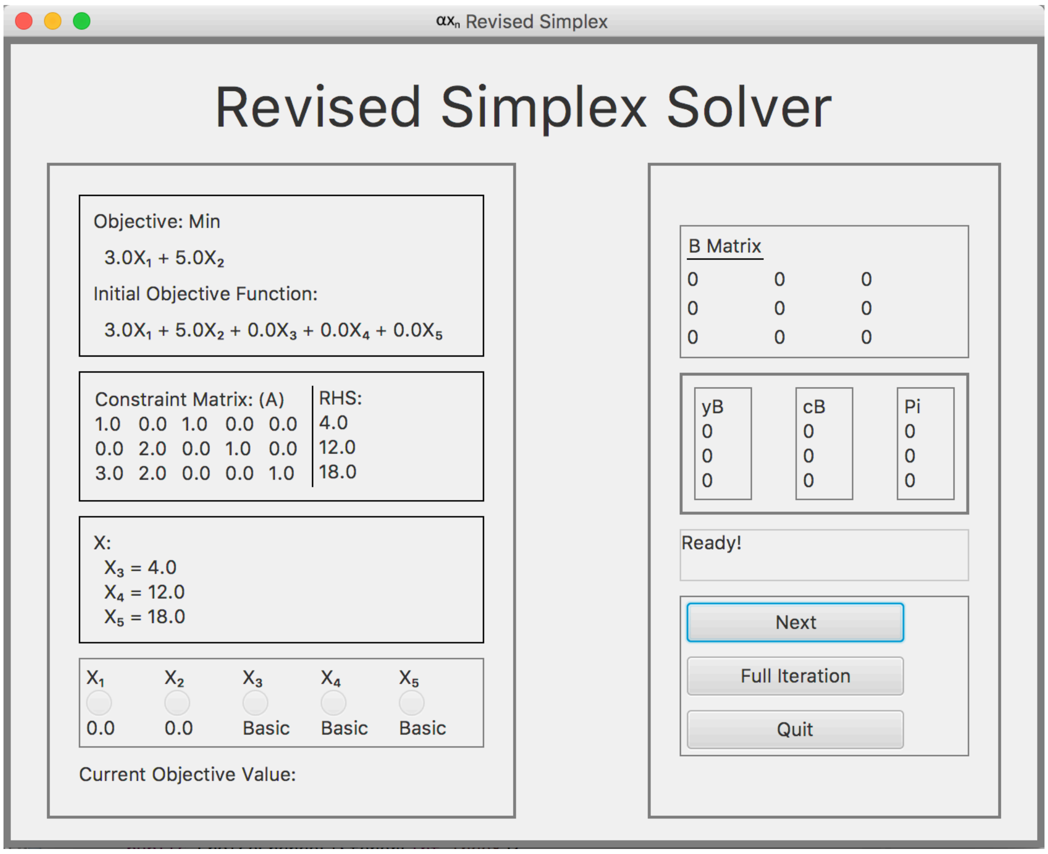
X_1 + X_2

Subject to:

X_1 + X_2 ≤

X_1 + X_2 ≤

(Figure 1: Data Entry Screen)



Revised Simplex Solver

Objective: Min

$3.0X_1 + 5.0X_2$

Initial Objective Function:

$3.0X_1 + 5.0X_2 + 0.0X_3 + 0.0X_4 + 0.0X_5$

```
Constraint Matrix: (A)      | RHS:
1.0  0.0  1.0  0.0  0.0     | 4.0
0.0  2.0  0.0  1.0  0.0     | 12.0
3.0  2.0  0.0  0.0  1.0     | 18.0
```

X:

X_3 = 4.0

X_4 = 12.0

X_5 = 18.0

X_1	X_2	X_3	X_4	X_5
0.0	0.0	Basic	Basic	Basic

Current Objective Value:

B Matrix

```
0   0   0
0   0   0
0   0   0
```

yB	cB	Pi
0	0	0
0	0	0
0	0	0

Ready!

(Figure 2: Revised Simplex)

References

Hillier, Frederick S., and Gerald J. Lieberman. *Introduction to Operation Research*. New York McGraw-Hill, 1995. Print. ([1])

Raskin, Jef. *The Humane Interface: New Directions for Designing Interactive Systems*. Reading, MA: Addison-Wesley, 2000. Print. ([2])

"Simplex Method." *NEOS*. Web. 27 May 2016. ([3])

Lau, H. T. *A Java Library of Graph Algorithms and Optimization*. Boca Raton: Chapman & Hall/CRC, 2007. Print. ([4])

Passos, Waldemar Dos. *Numerical Methods, Algorithms, and Tools in C#*. Boca Raton, FL: CRC, 2010. Print. ([5])