SIMPLEX METHOD TOOLBOX V1.3
USER'S MANUAL

December 27, 2017

JIANG Shijie
EMAIL: shijie.jiang@hotmail.com

Department of Civil & Environmental Engineering
National University of Singapore

Contents

INTRODUCTION

The Simplex Method Toolbox V1.3 (LP_SMT) is a C++-based and cross-platform program, which can run on Microsoft Windows, Linux, and macOS, for solving Linear Programming (LP) problems.

This program uses the **Two-phase Simplex Method** as its main algorithm and is capable of handling a variety of LP problems, including MAX / MIN objective function, equality and inequality constraints, and nonnegative / nonpositive / unrestricted variables. Both manually inputting and automatically importing can be used for defining the LP problem flexibly. Additionally, this toolbox can deal with several particular cases in the simplex method, such as unbounded solution, infeasible solution, as well as zero-level artificial variables at the end of Phase I.

The document layout is organized as follows:

- **Introduction** – a concise description of the toolbox
- **Getting Started** – how to compile this program and run the sample problem
- **Algorithm Description** – a brief introduction and flow charts of the main function and subroutines involved
- **Examples** – some examples of using this toolbox
- **Source Code** – an extract from the annotated source code
- **References** – the references which help to build this toolbox
- **License** – the license information

GETTING STARTED

Prerequisites

As this program is built by C++, you need a C++ compiler installed on your machine to compile the source code file to an executable binary file. There are an enormous number of development environments for C++ programmers, some of which are freely available and some of which are commercial products.
I highly recommend you use GCC, which is known as The GNU Compiler Collection, as your free C++ compiler. GCC is one of the most mature and compatible compilers, and it includes front ends for C, C++, Objective-C, Fortran, Ada, and Go, as well as libraries for these languages (libstdc++,...).

Fedora,Red Hat, CentOS, Scientific Linux

GCC was originally written as the compiler for the GNU/Linux operating system. If you are using *Fedora, Red Hat, CentOS*, or *Scientific Linux*, you can use the following **yum** command to install GNU C++ compiler:

```
$ sudo yum groupinstall 'Development Tools'
```

Debian,Ubuntu Linux

If you are using *Debian* or *Ubuntu Linux*, type the following **apt-get** command to install GNU C++ compiler:

```
$ sudo apt-get update
$ sudo apt-get install gcc
```

macOS

If you are a *macOS* user, you should have Xcode (in Mac App Store) installed, and then run the following command from your **Terminal**:

```
$ xcode-select --install
```

Microsoft Windows

However, if you are a *Microsoft Windows* user, you should install pre-compiled versions of GCC. There are multiple ports of GCC to Windows, but **MinGW** (Minimalist GNU for Windows) seems to get the most traction. More information about installing MinGW please refer to <http://www.mingw.org/>. This video (https://www.youtube.com/watch?v=DHekr3EtDOA) may help you to install MinGW (GCC) in Windows.

Compiling

I assume everything went well in the previous step and you have successfully installed GCC on your machine. To compile the Simplex Method Toolbox V1.3 is very easy, just run the following command in your **Terminal** or **Command Prompt**:

```
$ g++ -std=c++11 LP_SMT.cpp -o LP_SMT
```

Make sure *LP_SMT.cpp* is located in your current path. This command will use the installed GNU C++ Compiler (**g++**) with an ISO C++ standard ratified in 2011 (**-std=c++11**) to compile the source code file (**LP_SMT.cpp**) into an executable binary file (**LP_SMT**) under the same path.

Running the Sample Problem

Now an executable binary file named (**LP_SMT**) is generated, you can execute this program directly. Some information will be shown in the **Terminal** or **Command Prompt**:

```
**********************************************************************
**                  SIMPLEX METHOD TOOLBOX V1.3                     **
**  THIS IS DEVELOPED BY SHIJIE JIANG (SHIJIE.JIANG@HOTMAIL.COM)    **
**              COPYRIGHT (C) 2017  ALL RIGHTS RESERVED             **
**         BUILT BY C++ FOR CE6001 PROJECT ON 15/OCT/2017           **
**                                                                  **
** *** Features (updated on 21/OCT/2017):                          **
** 1) Support MAX / MIN objective function, Equality and Inequality **
**    constraints, and Nonnegative / Nonpositive / Unrestricted     **
**    variables;                                                    **
** 2) Utilize Two-phase method;                                     **
** 3) Support both manually inputting or automatically importing;   **
** *** NEW Features (updated on 11/NOV/2017):                       **
** 1) Avoid artificial variables being basic (at ZERO level) at the **
**    end of Phase I.                                               **
** 2) Handle both infeasible solution and unbounded solution.       **
**                                                                  **
** More information please refer to User's Manual.                  **
**********************************************************************
Press Enter to Continue...
```

After pressing Enter, you should choose a method to define the LP problem. There are two options, one is importing from an ASCII file, and another one is inputting manually step by step.

```
****** Please choose a method to define the LP problem:
****** 1 -- Import from an ASCII file
****** 2 -- Input manually step by step
```

In this test, we just import the LP problem from an ASCII file named **test.txt** under the same path. This ASCII file corresponds to:

$$\begin{array}{lll} \text{Maximize} & z = 2x_1 + 3x_2 & \\ \text{s.t.} & 2x_1 + x_2 & \leq 4 \\ & x_1 + 2x_2 & \leq 5 \\ & x_1, x_2 & \geq 0 \end{array}$$

This problem is the same as **Example 3.2-1** in [1]. The format requirement of such ASCII file can be seen in **Examples Section**. We enter **1** and then input the file name **test.txt** just as follows:

```
 >>>>  1
######################################################################
######    STEP 1: IMPORT THE LP PROBLEM FROM AN ASCII FILE   ######
######################################################################
##S1## Please enter the file name with full path and extension :
>>>>  test.txt
```

Based on what you inputted or imported, the following information can help you to check your defined LP problem. If any modification is needed, you can enter **N** to redefine the problem. Otherwise, enter **Y** to confirm and proceed.

```
##S1## Your LP problem imported is equavlant to:
MAX Z = (2)X1 + (3)X2
s.t.
(2)X1 + (1)X2 <= 4
(1)X1 + (2)X2 <= 5
##S1## Your NONNEGATIVE variable(s) include:
X1 X2
Your NONPOSITIVE variable(s) include:
NONE
Your UNRESTRICTED variable(s) include:
NONE
##S1## Confirm? (Y/N)   y
##S1## Your LP problem is confirmed.
```

After importing the problem successfully, Simplex Method Toolbox V1.3 starts solving this LP problem from transforming the problem into a canonical form by adding some **SLACK**

or **SURPLUS** variables, as well as some **ARTIFICIAL** variables for two-phase method, and making all the right-hand side nonnegative.

```
######################################################################
######            STEP 2: PREPROCESS THE LP PROBLEM           ######
######################################################################
##S2## The row #1 is a (<=)-inequality, and it has been transformed
to an equation by adding a SLACK variable.
##S2## The row #2 is a (<=)-inequality, and it has been transformed
to an equation by adding a SLACK variable.
##S2## The canonical form of your LP problem is:
MAX Z = (2)X1 + (3)X2 + (0)X3 + (0)X4 + (0)X5 + (0)X6
s.t.
(2)X1 + (1)X2 + (1)X3 + (0)X4 + (1)X5 + (0)X6 = 4
(1)X1 + (2)X2 + (0)X3 + (1)X4 + (0)X5 + (1)X6 = 5
X1, X2, X3, X4, X5, X6 >= 0
```

Subsequently, Simplex Method Toolbox V1.3 will solve the canonical form of the original LP problem by two-phase method through several iterations.

```
######################################################################
######            STEP 3: TWO-PHASE METHOD: PHASE I           ######
######################################################################
ITERATION #1:
BV       X1        X2        X3        X4        X5        X6        CV
----------------------------------------------------------------------------
X5        2         1         1         0         1         0         4
X6        1         2         0         1         0         1         5
-Z       -3        -3        -1        -1         0         0        -9

ITERATION #2 :  (X5 Leaves and X1 Enters)
BV       X1        X2        X3        X4        X5        X6        CV
----------------------------------------------------------------------------
X1        1       0.5       0.5         0       0.5         0         2
X6        0       1.5      -0.5         1      -0.5         1         3
-Z        0      -1.5       0.5        -1       1.5         0        -3

ITERATION #3 :  (X6 Leaves and X2 Enters)
BV       X1        X2        X3        X4        X5        X6        CV
----------------------------------------------------------------------------
X1        1         0  0.666667 -0.333333  0.666667 -0.333333         1
X2        0         1 -0.333333  0.666667 -0.333333  0.666667         2
-Z        0         0 2.77556e-017 -5.55112e-017         1         1         0
```

```
ITERATION #4 :  (X2 Leaves and X4 Enters)
BV       X1        X2        X3        X4        X5        X6        CV
--------------------------------------------------------------------------
X1        1       0.5       0.5         0       0.5         0         2
X4        0       1.5      -0.5         1      -0.5         1         3
-Z        0 8.32667e-017         0         0         1         1 1.66533e-016

##S3## The artificial variables have completed their mission, now move to Phase II.

######################################################################
######            STEP 4: TWO-PHASE METHOD: PHASE II          ######
######################################################################
ITERATION #1:
BV       X1        X2        X3        X4        CV
----------------------------------------------------------
X1        1       0.5       0.5         0         2
X4        0       1.5      -0.5         1         3
-Z        0        -2         1         0        -4

ITERATION #2 :  (X4 Leaves and X2 Enters)
BV       X1        X2        X3        X4        CV
----------------------------------------------------------
X1        1         0  0.666667 -0.333333         1
X2        0         1 -0.333333  0.666667         2
-Z        0         0  0.333333   1.33333        -8

****** The optimal variable(s) of your original problem are:
| X1 1
| X2 2
****** The optimum of your original problem is : 8
```

Finally, as can be seen in the **Terminal**, the optimum of this problem is 8, and optimal variables are $x_1 = 1$ and $x_2 = 2$. It can be verified by the solution of **Example 3.2-1** in [1].

ALGORITHM DESCRIPTION

The Simplex Method Toolbox V1.3 is a C++-based program which contains a main function and a number of subroutines / functions.
This section will briefly introduce the main function and each subroutine.

main()

The flow chart of the *main()* function is shown in Figure 1. It is obvious that the LP problem can be defined by either *ImportProblem()* or *DataInput()*, before being transformed to a canonical form and solved through two-phase method. The program will end if no feasible solution is found, the solution is unbounded, or the problem has been solved successfully.

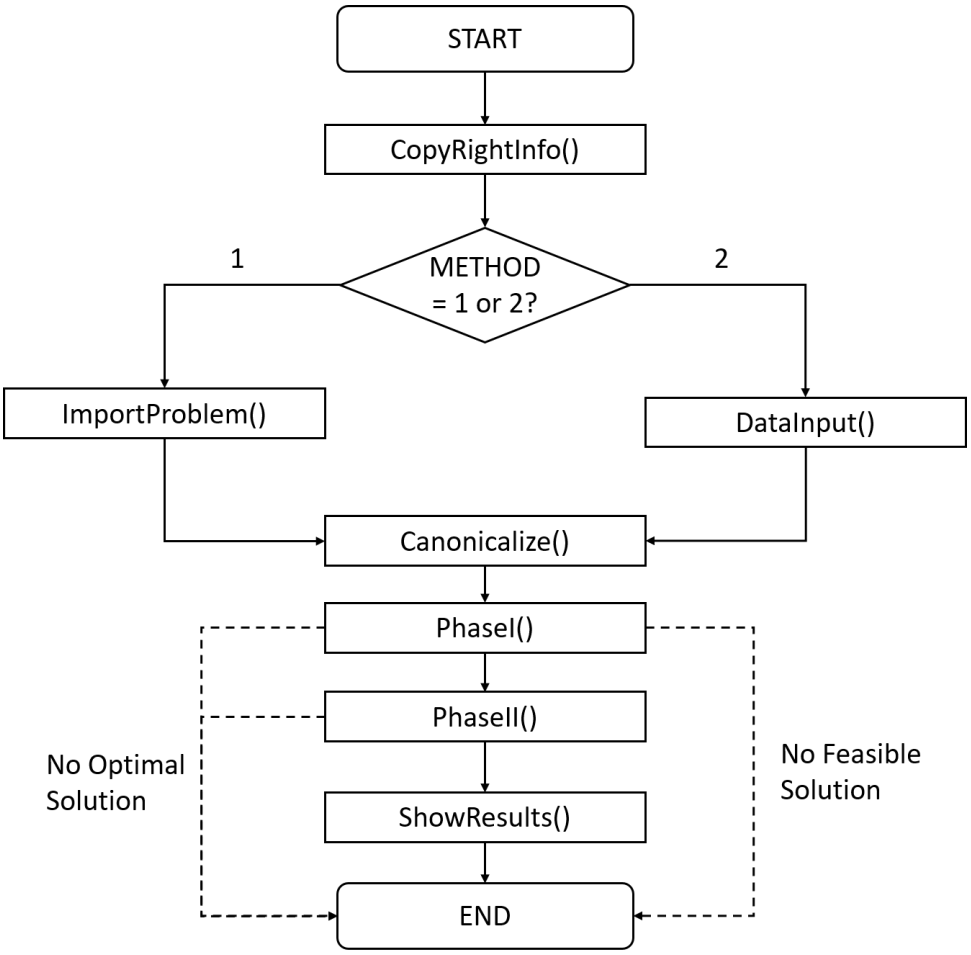

Figure 1: Flow chart of the *main()* function

CopyRightInfo()

The *CopyRightInfo()* function is used for showing the copyright information and new features of this toolbox.

ImportProblem()

The *ImportProblem()* function is used for defining a LP problem by importing a strictly formatted ASCII file. The flow chart of this function is shown in Figure 2.

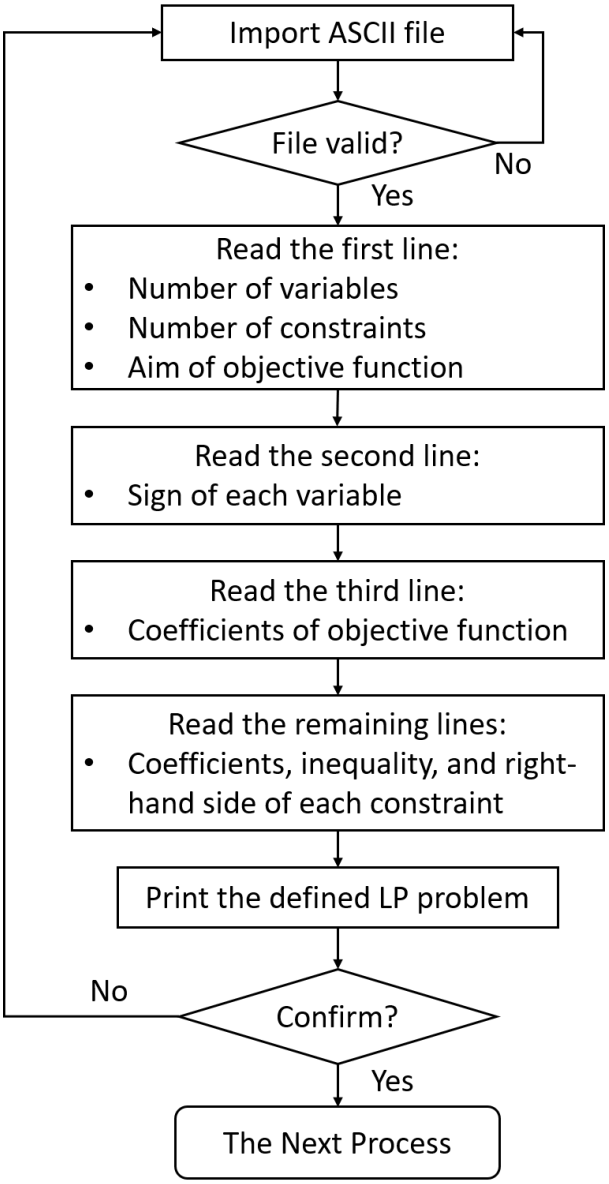

Figure 2: Flow chart of the *ImportProblem()* function

DataInput()

The *DataInput()* function is alternatively used for defining a LP problem by inputting with a step-by-step guide. The flow chart of this function is shown in Figure 3.

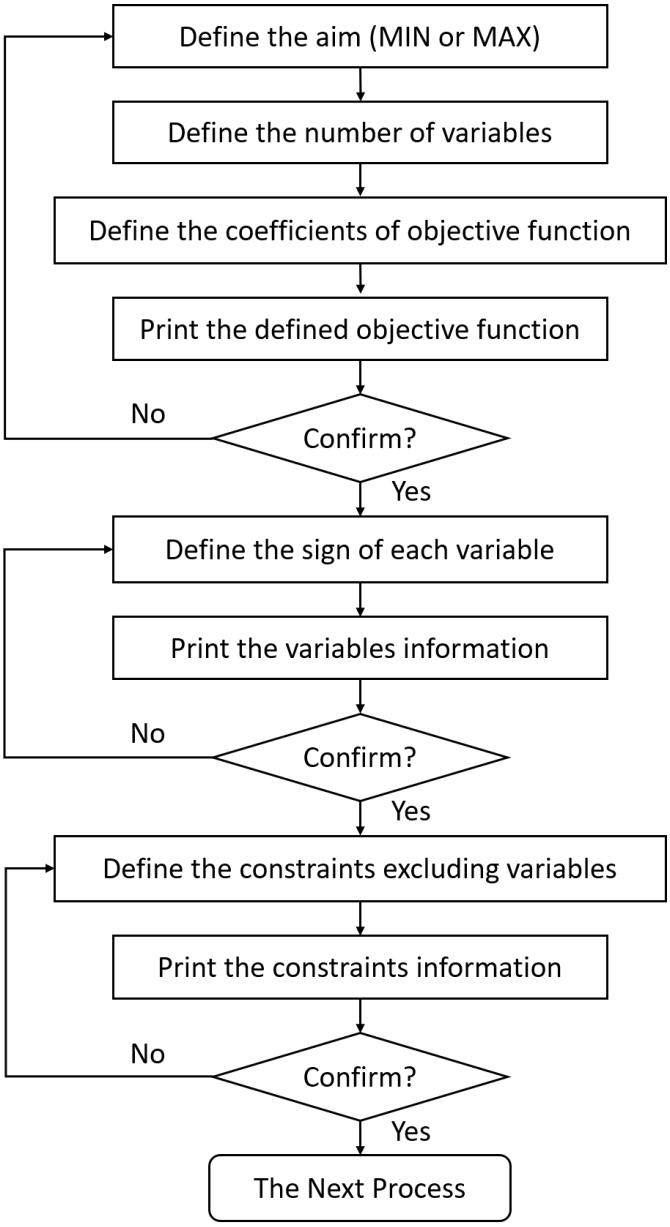

Figure 3: Flow chart of the *DataInput()* function

Canonicalize()

The *Canonicalize()* function is used for making the LP problem in a canonical form. The processes include:

1. making all the variables nonnegative (dealing with nonpositive and unrestricted variables);
2. transforming all the inequations into equations with nonnegative right-hand side;
3. adding one artificial variable for each constraint directly to transform the standard form into a canonical form.

The flow chart of this function is shown in Figure 4.

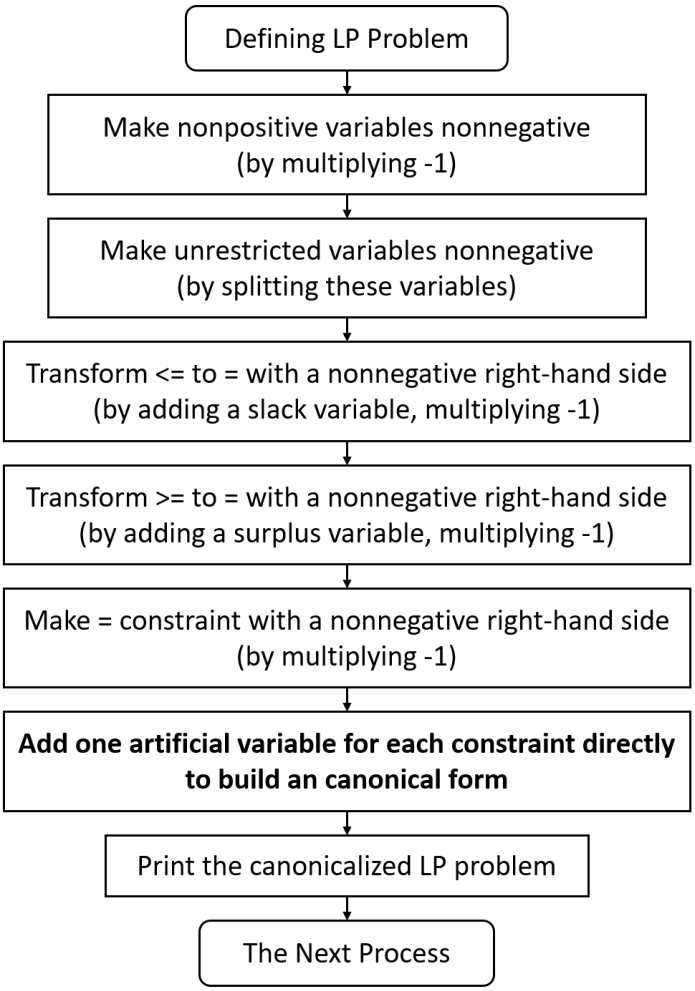

Figure 4: Flow chart of the *Canonicalize()* function

PhaseI()

The *PhaseI()* function is used for searching the initial BFS for the original LP problem by minimizing the sum of artificial variables to zero. The flow chart of this function is shown in Figure 5. In this function, *InitialBFS()* and *Simplex()*, which will be introduced later, are used for solving the Phase I problem. When the optimality condition is fulfilled, a further check of whether the optimum equal to zero will be made. If the minimum value of the sum is not zero, the LP problem has no feasible solution and exit; otherwise, proceed to next step, in which whether the basic variables including artificial variables are checked. If one or more artificial variables are basic at the end of Phase I, they should be replaced by nonbasic variables with a nonzero coefficient.

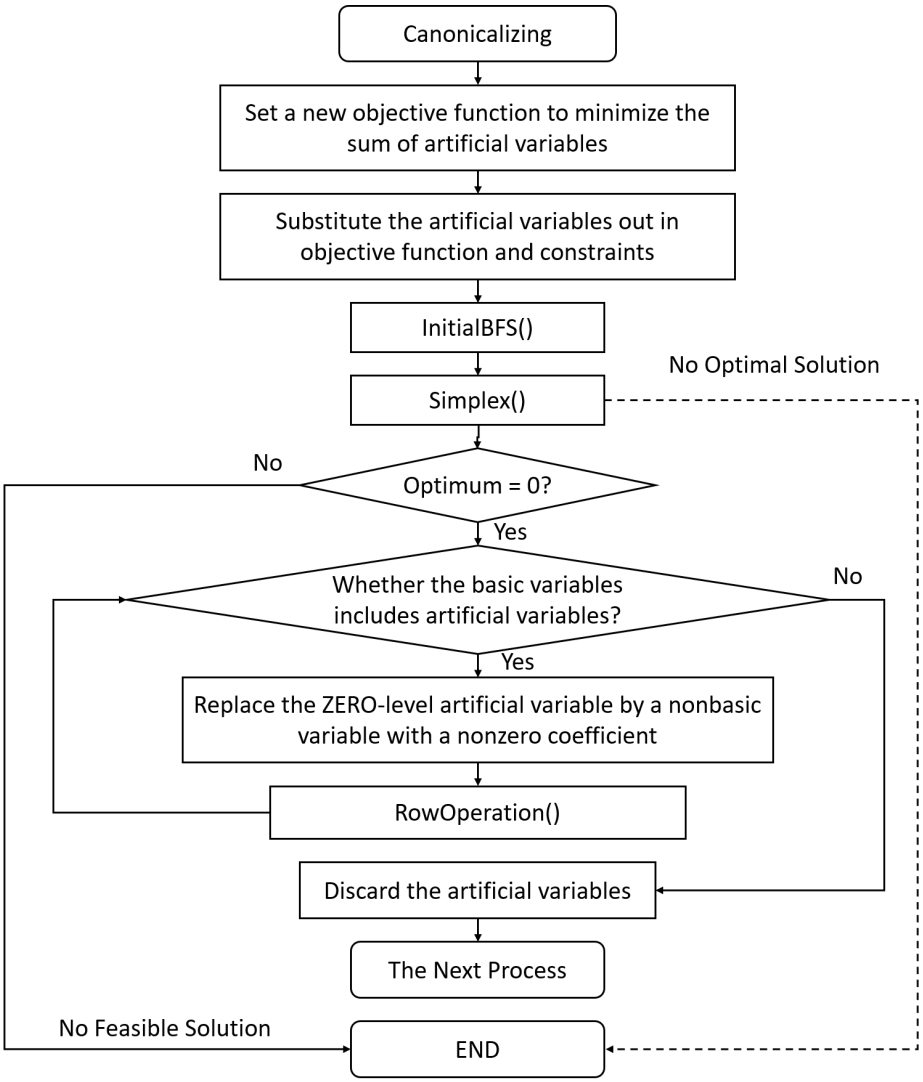

Figure 5: Flow chart of the *PhaseI()* function

PhaseII()

The *PhaseI()* function is used for solving the original LP problem by using the feasible solution from Phase I as a starting basic feasible solution. The flow chart of this function is shown in Figure 6.

This function first restores the original objective function from the sum of artificial variables, and then substitutes out the basic variables with nonzero coefficients in the z-row to eliminate the inconsistency. *Simplex()* will be called to solve the original LP problem. If the optimality condition is fulfilled, then move to the next process; if the objective value is unbounded, then exit.

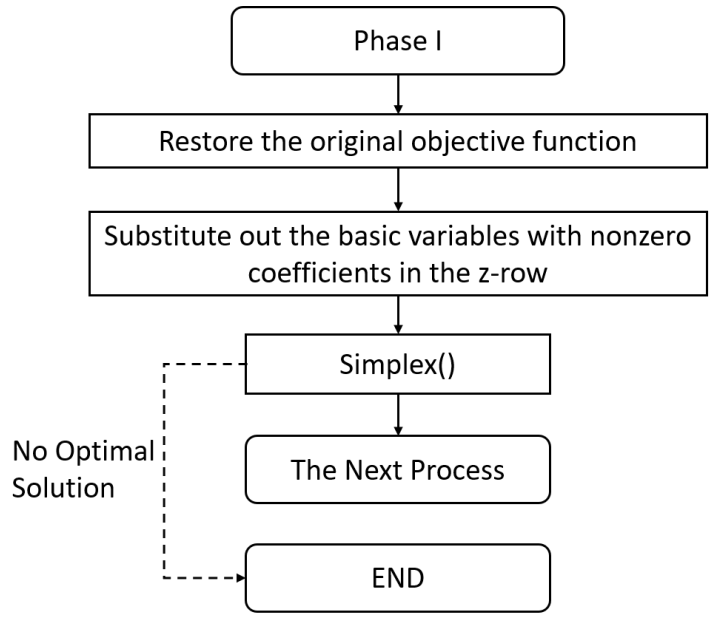

Figure 6: Flow chart of the *PhaseII()* function

InitialBFS()

The *InitialBFS()* function is used for searching the initial BFS. In this program, since *Canonicalize()* has added one artificial variable for each constraint, the initial BFS is the last NC (NC is the number of constraints) variables directly.

Simplex()

The *Simplex()* function is the core of this program, which is used for iteratively solving the LP problem. The flow chart of this function is shown in Figure 7.
This function first call RowOperation() and Optimal() to process the initial BFS, and then check whether the result is optimal. If the optimality condition is fulfilled (flag = 1), the routine will end and move to the next process; if the optimality condition is not fulfilled yet (flag = 0), then find another pivot and BFS to iteratively solve the problem until the result is optimal; if the objective value is unbounded (flag = -1), then exit.

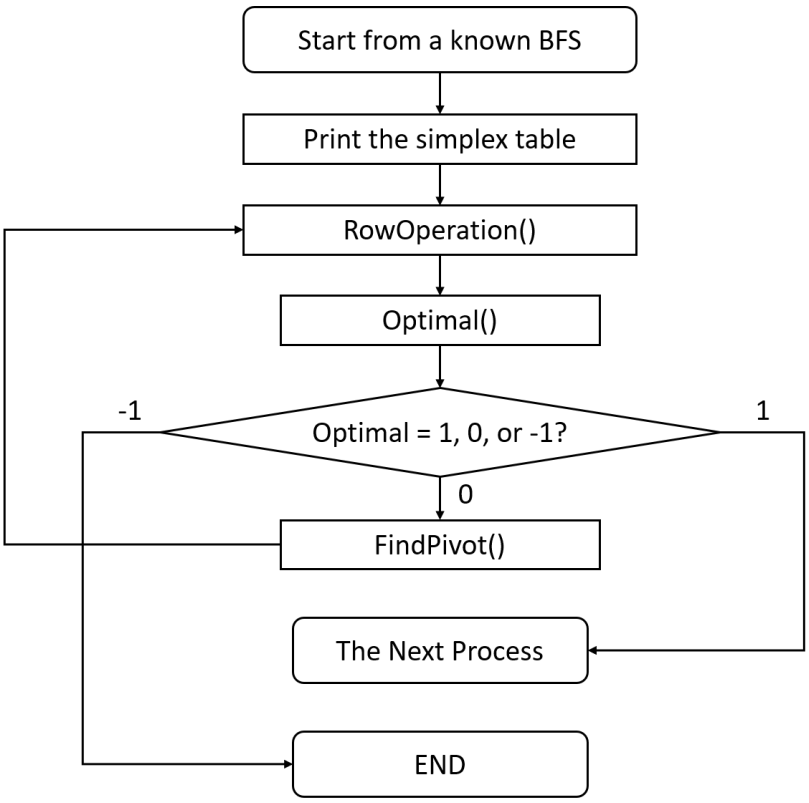

Figure 7: Flow chart of the *Simplex()* function

RowOperation()

The *RowOperation()* function is used for Gauss-Jordan row operation. It will make the current pivot to one, while other elements in the pivot column zeros. The flow chart of this function is shown in Figure 8.

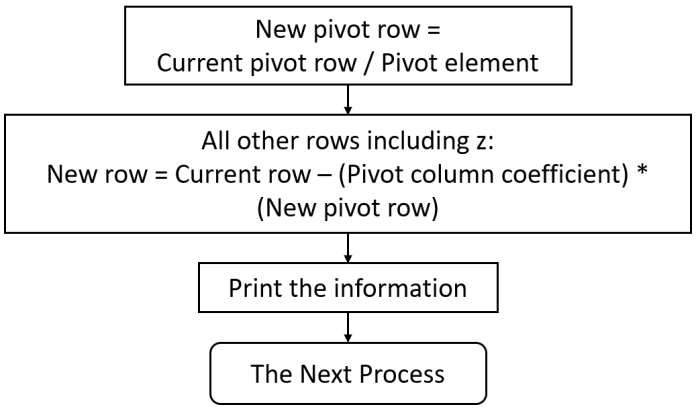

Figure 8: Flow chart of the *RowOperation()* function

Optimal()

The *Optimal()* function is used for checking whether the optimality condition is fulfilled by examining the z-row coefficients, and checking whether the objective value is unbounded. The flow chart of this function is shown in Figure 9.

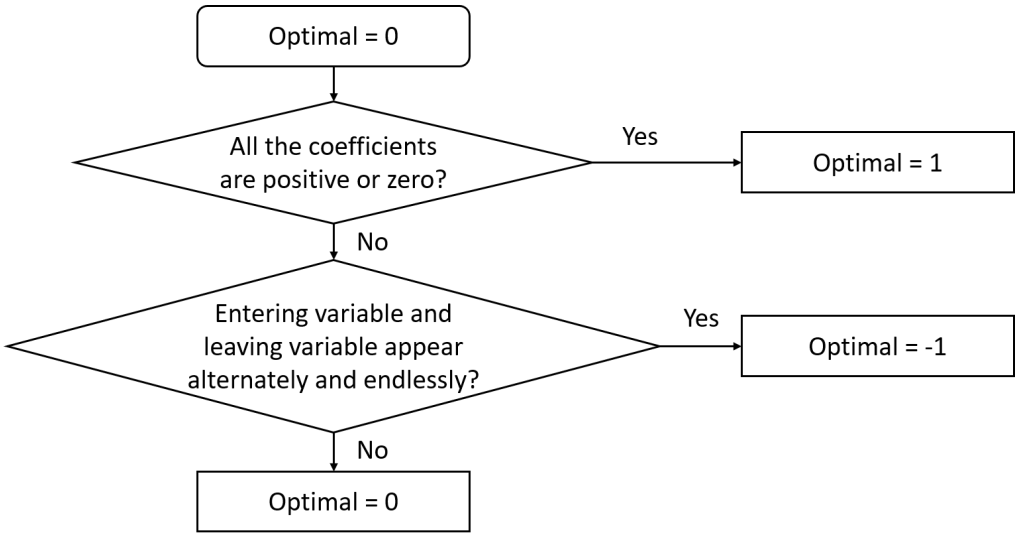

Figure 9: Flow chart of the *Optimal()* function

FindPivot()

The *FindPivot()* function is used for finding the entering variable by the optimality condition and the leaving variable by feasibility condition. The flow chart of this function is shown in Figure 10.

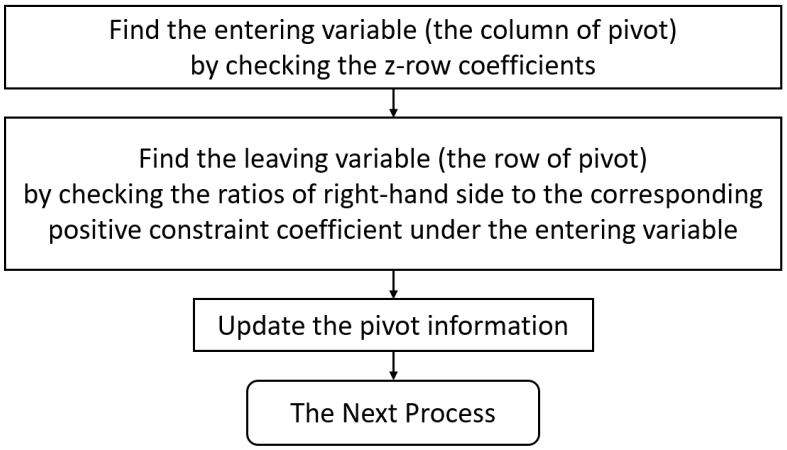

Figure 10: Flow chart of the *FindPivot()* function

ShowResults()

The *ShowResults()* function is used for showing the final solution of the original LP problem. As the original nonnegative variables and unrestricted variables are preprocessed at the beginning, this routine will restore these variables.

EXAMPLES

As stated before, this toolbox supports defining LP problems by both manually inputting or automatically importing, and supports a variety of LP problems, including MAX / MIN objective function, equality and inequality constraints, and nonnegative / nonpositive / unrestricted variables.
The following subsections will illustrate more applications of the Simplex Method Toolbox V1.3.

ASCII File

Importing from an ASCII file is a convenient and flexible way to define a LP problem. The ASCII file should follow a strict format

```
<int: NV>  <int: NC>  <int: AIM>
<int: VS[1]>  <int: VS[2]> ...  <int: VS[NV]>
<float: CO[1]>  <float: CO[2]> ...  <float: CO[NV]>
<float: C[1][1]>  ...  <float: C[1][NV]>  <int: CS[1]>  <float: b[1]>
<float: C[2][1]>  ...  <float: C[2][NV]>  <int: CS[2]>  <float: b[2]>
...
<float: C[NC][1]>  ...  <float: C[NC][NV]>  <int: CS[NC]>  <float: b[NC]>
```

- `int` and `float` are the data types representing integers and numbers with floating decimal points, respectively;
- `NV` is the number of decision variables;
- `NC` is the number of constraints excluding variable constraints;
- `AIM` is an indicator of whether the objective function is MAX (1) or MIN (2);
- `VS[1]`, `VS[2]`,..., `VS[NV]` are the indicators of the signs of the `NV` variables respectively (1: nonnegative, 2: nonpositive, 3: unrestricted);
- `CO[1]`, `CO[2]`,..., `CO[NV]` are the coefficients of the `NV` variables of the objective function respectively;
- `C[i][j]` ($i = 1,2,...,NC$, $j = 1,2,...,NV$) is the coefficient of the jth variable for the ith constraint;
- `CS[1]`, `CS[2]`,..., `CS[NC]` are the indicators of inequality of the `NC` constraints respectively ((1: $\leq$, 2: $\geq$, 3: $=$);
- `b[1]`, `b[2]`,..., `b[NC]` are the right-hand sides of the `NC` constraints respectively.

Thus, the corresponding ASCII file (*test.txt*) for the sample LP problem solved in **Getting Started**

$$\begin{array}{lll} \text{Maximize} & z = 2x_1 + 3x_2 & \\ \text{s.t.} & 2x_1 + x_2 & \leq 4 \\ & x_1 + 2x_2 & \leq 5 \\ & x_1, x_2 & \geq 0 \end{array}$$

should be

```
2 2 1
1 1
2 3
2 1 1 4
1 2 1 5
```

Manually Inputting

An alternative method for defining the LP problem in the Toolbox is inputting step by step. It is tedious but easy to understand for beginners. Here is an example to define the LP problem manually (**Example 2.2-1** in [1]).

$$\begin{array}{lll} \text{Maximize} & z = 5x_1 + 4x_2 & \\ \text{s.t.} & 6x_1 + 4x_2 & \leq 24 \\ & x_1 + 2x_2 & \leq 6 \\ & -x_1 + x_2 & \leq 1 \\ & x_2 & \leq 2 \\ & x_1, x_2 & \geq 0 \end{array}$$

At the beginning of the program, we choose **2** for inputting manually step by step. Firstly, the objective function is defined.

```
######################################################################
######          STEP 1.1: DEFINE THE OBJECTIVE FUNCTION        ######
######################################################################
#S1.1# Objective function is MAX(1) or MIN(2) ?
#S1.1# Please input 1 or 2:  1
#S1.1# Number of variables ? (please input an INTEGRE)  2
#S1.1# Please input the coefficient of each variable:
>>>>   X1? 5
>>>>   X2? 4
#S1.1# Your objective function inputted is equivalent to:
MAX Z = (5)X1 + (4)X2
#S1.1# Confirm? (Y/N)
```

After confirming the objective function, we need define the sign constraint of each variable.

```
######################################################################
######           STEP 1.2: DEFINE THE VARIABLE SIGN           ######
######################################################################
#S1.2# Please indicate whether the constraint of each variable is:
#S1.2# NONNEGATIVE(1), NONPOSITIVE(2), or UNRESTRICTED(3).
>>>>   X1? 1
>>>>   X2? 1
#S1.2# Your nonnegative variable(s) include:
X1 X2
#S1.2# Your nonpositive variable(s) include:
NONE
#S1.2# Your unrestricted variable(s) include:
NONE
#S1.2# Confirm? (Y/N)
```

After confirming the variable signs, we need define the constraints excluding above sign constraints.

```
######################################################################
###### STEP 1.3: DEFINE THE CONSTRAINTS (EXCLUDING VARIABLES)  ######
######################################################################
#S1.3# Number of constraints EXCLUDING variable constraints defined
in STEP 1.2 ? (please input an INTEGRE:) 4
#S1.3# Coefficient of each variable in constraint NO.1:
>>>>   X1? 6
>>>>   X2? 4
#S1.3# Inequality or equality (1:<= | 2:>= | 3:=) in constraint NO.1
       (please input 1/2/3):
>>>>   1
#S1.3# Right hand side in constraint NO.1:
>>>>   24
#S1.3# Coefficient of each variable in constraint NO.2:
>>>>   X1? 1
>>>>   X2? 2
#S1.3# Inequality or equality (1:<= | 2:>= | 3:=) in constraint NO.2
       (please input 1/2/3):
>>>>   1
#S1.3# Right hand side in constraint NO.2:
>>>>   6
#S1.3# Coefficient of each variable in constraint NO.3:
>>>>   X1? -1
>>>>   X2? 1
```

```
#S1.3# Inequality or equality (1:<= | 2:>= | 3:=) in constraint NO.3
       (please input 1/2/3):
>>>>   1
#S1.3# Right hand side in constraint NO.3:
>>>>   1
#S1.3# Coefficient of each variable in constraint NO.4:
>>>>   X1? 0
>>>>   X2? 1
#S1.3# Inequality or equality (1:<= | 2:>= | 3:=) in constraint NO.4
       (please input 1/2/3):
>>>>   1
#S1.3# Right hand side in constraint NO.4:
>>>>   2
#S1.3# Your constraint(s) excluding variable constraint(s) are:
(6)X1 + (4)X2 <= 24
(1)X1 + (2)X2 <= 6
(-1)X1 + (1)X2 <= 1
(0)X1 + (1)X2 <= 2
#S1.3# Confirm? (Y/N)
```

The remaining solving steps are the same as the sample problem in **Getting Started**, in which two-phase methods are employed after transforming the problem into a canonical form. The final results (verified) are

```
****** The optimal variable(s) of your original problem are:
| X1 3
| X2 1.5
****** The optimum of your original problem is : 21
```

Minimization Problem

The above said LP problem are all to find a maximum, yet this toolbox is also capable of dealing with minimization problems.
Example 2.4-3 in [1] defines a Multi Period Production-Inventory Model:

$$\begin{aligned}
\text{Minimize} \quad & z = 50x_1 + 45x_2 + 55x_3 + 48x_4 + 52x_5 + 50x_6 + 8(x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12}) \\
\text{s.t.} \quad & x_1 - x_7 = 100 \\
& x_7 + x_2 - x_8 = 250 \\
& x_8 + x_3 - x_9 = 190 \\
& x_9 + x_4 - x_{10} = 140 \\
& x_{10} + x_5 - x_{11} = 220 \\
& x_{11} + x_6 = 110 \\
& x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12} \geq 0
\end{aligned}$$

The corresponding ASCII input file (*min.txt*) is

```
12 6 2
1 1 1 1 1 1 1 1 1 1 1 1
50 45 55 48 52 50 8 8 8 8 8 8
1 0 0 0 0 0 -1 0 0 0 0 0 3 100
0 1 0 0 0 0 1 -1 0 0 0 0 3 250
0 0 1 0 0 0 0 1 -1 0 0 0 3 190
0 0 0 1 0 0 0 0 1 -1 0 0 3 140
0 0 0 0 1 0 0 0 0 1 -1 0 3 220
0 0 0 0 0 1 0 0 0 0 1 0 3 110
```

The final results(verified) are

```
****** The optimal variable(s) of your original problem are:
| X1 100
| X2 440
| X3 0
| X4 140
| X5 220
| X6 110
| X7 0
| X8 190
| X9 0
| X10 0
| X11 0
| X12 0
****** The optimum of your original problem is : 49980
```

Inequality Constraints

Unlike some similar LP solvers which need a strict canonical form as input, this toolbox can solve LP problems with both equality and inequality constraints flexibly.
Example 3.4-1 in [1] defines such a LP program:

$$\begin{array}{lll} \text{Minimize} & z = 4x_1 + x_2 & \\ \text{s.t.} & 3x_1 + x_2 & = 3 \\ & 4x_1 + 3x_2 & \geq 6 \\ & x_1 + 2x_2 & \leq 4 \\ & x_1, x_2 & \geq 0 \end{array}$$

The corresponding ASCII input file (*cons.txt*) is

```
2 3 2
1 1
4 1
3 1 3 3
4 3 2 6
1 2 1 4
```

The final results (verified) are

```
****** The optimal variable(s) of your original problem are:
| X1 0.4
| X2 1.8
****** The optimum of your original problem is : 3.4
```

Nonpositive and Unrestricted Variables

Likewise, this toolbox is also able to solve LP problems with nonnegative, nonpositive and unrestricted decision variables.
Example 4.1-3 in [1] defines such a LP program:

$$\begin{array}{lll} \text{Minimize} & z = 5x_1 + 3x_2 + 8x_3 & \\ \text{s.t.} & x_1 - x_2 + 4x_3 & = 5 \\ & 2x_1 + 5x_2 + 7x_3 & \geq 6 \\ & x_1 & \text{unrestricted} \\ & x_2 & \leq 0 \\ & x_3 & \geq 0 \end{array}$$

The corresponding ASCII input file (*var.txt*) is

```
3 2 2
3 2 1
5 3 8
1 -1 4 3 5
2 5 7 2 6
```

The final results (verified) are

```
****** The optimal variable(s) of your original problem are:
| X1 -11
| X2 0
| X3 4
****** The optimum of your original problem is : -23
```

Zero-level Artificial Variables

The removal of the artificial variables and their columns at the end of Phase I can take place only when they are nonbasic. If one or more artificial variables are basic (at zero level) at the end of Phase I, then their removal requires some additional steps (See **Section 3.4.2** in [1]). The toolbox can address this problem by replacing such artificial variables with some nonbasic variables with a nonzero coefficient as stated before. The logic behind this is that is the feasibility of the remaining basic variables will not be affected when a zero artificial variable is made nonbasic regardless of whether the pivot element is positive or negative[1].
Porblem Set 3.4B (6) in [1] defines such a LP program:

$$\begin{array}{lrcl} \text{Maximize} & z = 3x_1 + 2x_2 + 3x_3 & & \\ \text{s.t.} & 2x_1 + x_2 + 2x_3 & = & 2 \\ & x_1 + 3x_2 + x_3 & = & 6 \\ & 3x_1 + 4x_2 + 2x_3 & = & 8 \\ & x_1, x_2, x_3 & \geq & 0 \end{array}$$

The corresponding ASCII input file (*zeroR.txt*) is

```
3 3 1
1 1 1
3 2 3
2 1 1 3 2
1 3 1 3 6
3 4 2 3 8
```

Generally, the Phase I will terminate with two zero artificial variables in the basic solution. However, this program can handle this at the end of Phase I.

```
######################################################################
######             STEP 3: TWO-PHASE METHOD: PHASE I          ######
######################################################################
ITERATION #1:
BV       X1        X2        X3        X4        X5        X6        CV
----------------------------------------------------------------------------
X4        2         1         1         1         0         0         2
X5        1         3         1         0         1         0         6
X6        3         4         2         0         0         1         8
-Z       -6        -8        -4         0         0         0       -16

ITERATION #2 :  (X4 Leaves and X2 Enters)
BV       X1        X2        X3        X4        X5        X6        CV
----------------------------------------------------------------------------
X2        2         1         1         1         0         0         2
X5       -5         0        -2        -3         1         0         0
X6       -5         0        -2        -4         0         1         0
-Z       10         0         4         8         0         0         0
##S3## As artificial variables X5 is at ZERO level,
##S3## it should be replaced by a nonbasic variable with a nonzero coefficient.

ITERATION #3 :  (X5 Leaves and X4 Enters)
BV       X1        X2        X3        X4        X5        X6        CV
----------------------------------------------------------------------------
X2  2.55556         1   1.22222         0  0.333333         0         2
X4-0.555556         0 -0.222222         1 -0.333333        -0        -0
X6 -7.22222         0  -2.88889         0  -1.33333         1         0
-Z  14.4444         0   5.77778         0   2.66667         0         0

##S3## As artificial variables X6 is at ZERO level,
##S3## it should be replaced by a nonbasic variable with a nonzero coefficient.

ITERATION #4 :  (X6 Leaves and X3 Enters)
BV       X1        X2        X3        X4        X5        X6        CV
----------------------------------------------------------------------------
X2     -0.5         1         0         0   1.96296  -1.22222         2
X4-5.55112e-017         0         0         1  -0.62963  0.222222         0
X3      2.5        -0         1         0  -1.33333         1         0
-Z4.44089e-016         0         0         0   10.3704  -5.77778         0


##S3## The artificial variables have completed their mission, now move to Phase II.
```

It can be seen that the zero artificial variables **X5** and **X6** are replaced by **X4** and **X3** at the end of Phase I respectively.
The final results (verified) are

```
****** The optimal variable(s) of your original problem are:
| X1 0
| X2 2
| X3 0
****** The optimum of your original problem is : 4
```

Unbounded Solution

In some LP models, the solution space is unbounded in at least one variable, meaning that variables may be increased indefinitely without violating any of the constraints. The associated objective function may also be unbounded in this case[1].
If the entering variable and leaving variable appear alternately and endlessly, then we can say the LP problem has an unbounded solution[1].
Example 3.5-3 in [1] defines such a LP program with unbounded solution space:

$$\begin{array}{lll} \text{Maximize} & z = 2x_1 + x_2 & \\ \text{s.t.} & x_1 - x_2 & \leq 10 \\ & 2x_1 & \leq 40 \\ & x_1, x_2 & \geq 0 \end{array}$$

The corresponding ASCII input file (*ub.txt*) is

```
2 2 1
1 1
2 1
1 -1 1 10
2 0 1 40
```

The results of the second phase are

```
######################################################################
######             STEP 4: TWO-PHASE METHOD: PHASE II          ######
######################################################################
ITERATION #1:
BV       X1        X2        X3        X4        CV
----------------------------------------------------
X1        1         0         0       0.5        20
X2        0         1        -1       0.5        10
```

```
-Z        0         0        -1       1.5       -50

ITERATION #2 :  (X2 Leaves and X3 Enters)
BV       X1        X2        X3        X4        CV
-------------------------------------------------------
X1        1         0         0       0.5        20
X3       -0        -1         1      -0.5       -10
-Z        0        -1         0         1       -40

ITERATION #3 :  (X3 Leaves and X2 Enters)
BV       X1        X2        X3        X4        CV
-------------------------------------------------------
X1        1         0         0       0.5        20
X2        0         1        -1       0.5        10
-Z        0         0        -1       1.5       -50

************************************************************************
****** OPPS! Unbounded Objective Value!
```

We can find that at the end of Phase II, **X2** and **X3** enter and leave alternatively, thus the program identifies this LP problem is unbounded.

No Feasible Solution

The feasible solution does not always exist according to the constraints defined. If at least one artificial variable is positive in the optimum iteration, then the LP has no feasible solution. **Example 3.5-4** in [1] defines such a LP program with infeasible solution space:

$$\begin{array}{lrcl} \text{Maximize} & z = 3x_1 + 2x_2 & & \\ \text{s.t.} & 2x_1 + x_2 & \leq & 2 \\ & 3x_1 + 4x_2 & \geq & 12 \\ & x_1, x_2 & \geq & 0 \end{array}$$

The corresponding ASCII input file (*infeasible.txt*) is

```
2 2 1
1 1
3 2
2 1 1 2
3 4 2 12
```

The results of the first phase are

```
######################################################################
######             STEP 3: TWO-PHASE METHOD: PHASE I          ######
######################################################################
ITERATION #1:
BV       X1        X2        X3        X4        X5        X6        CV
----------------------------------------------------------------------------
X5        2         1         1         0         1         0         2
X6        3         4         0        -1         0         1        12
-Z       -5        -5        -1         1         0         0       -14

ITERATION #2 :  (X5 Leaves and X1 Enters)
BV       X1        X2        X3        X4        X5        X6        CV
----------------------------------------------------------------------------
X1        1       0.5       0.5         0       0.5         0         1
X6        0       2.5      -1.5        -1      -1.5         1         9
-Z        0      -2.5       1.5         1       2.5         0        -9

ITERATION #3 :  (X1 Leaves and X2 Enters)
BV       X1        X2        X3        X4        X5        X6        CV
----------------------------------------------------------------------------
X2        2         1         1         0         1         0         2
X6       -5         0        -4        -1        -4         1         4
-Z        5         0         4         1         5         0        -4

##S3## OOPS! No Feasible Solution!
```

It can be seen that the at the final iteration, the optimality condition is fulfilled, while the optimum is still greater than zero. Thus, this problem has no feasible solution.

REFERENCES

[1] Hamdy A. Taha. *Operations Research: An Introduction (9th Edition).* Pearson, 2010.

LICENSE