

## FACULTY OF ENGINEERING

# GROUP SEMESTER ASSIGNMENT

REII 313 – Object-Oriented programming

Submitted to: Mr. R Luies  
Date: 22/06/2020  
Author: J.G. Roos 29870089

## Table of Contents

### Contents

1. Introduction .....	3
2.1 Background.....	3
2.2.1 Simplex Method .....	3
2.2.2 Branch and Bound .....	4
2.2.3 Methodology .....	4
3.1 GUI Interface .....	5
3.1.1 The Main Window .....	5
3.1.1.1 Number of Variables and Constraints.....	6
3.1.1.2 Enter Objective Function- Button .....	6
3.1.1.3 Enter Constraint Button.....	7
3.1.1.4 Answers Button.....	8
3.1.1.5 Minimize and Maximization Buttons .....	8
3.1.1.6 Display Button.....	8
3.2 Limitations .....	9
3.3 UML.....	9
3.4 Test .....	10
3.4.1 Maximization Test 1 .....	10
3.4.1.1 Input.....	11
3.4.1.2 Desired Result and Obtained Result.....	12
4. Conclusion .....	14
5. References.....	15

### List of Figures

Figure 1: Components of simplex.....	3
Figure 2: Tree .....	4
Figure 3: Agile Development.....	5
Figure 4: Linear Problem Information (Main Window).....	6
Figure 5: Objective Function Window.....	7
Figure 6: Constraint Window .....	7
Figure 7: Answer of constraint Window .....	8

Figure 8: Maximization Assumption example .....	8
Figure 9: Minimization Assumption Example.....	8
Figure 10: UML (in theory) .....	9
Figure 11: Practical UML.....	10
Figure 12: Example 1 Maximization .....	10
Figure 13: Example 1 Variables and Constraints.....	11
Figure 14: Input order of objection function .....	11
Figure 15: Order of input of constraints .....	12
Figure 16: Order of input answers .....	12
Figure 17: Desired Result .....	12
Figure 18: Result for Example one .....	13

# 1. Introduction

As described by the assignment information given by our lecturer an individual or a group of two students was expected to write an application that could solve any given Linear Programming problem. This report gives feedback on the application software developed. The software is intended to use the Simplex method and branch and bound calculations. However, more will be discussed in this report. The application aims to make calculations of linear problems easier, addressing the Linear Problem (Simplex) and Integer Problem (Branch and Bound).

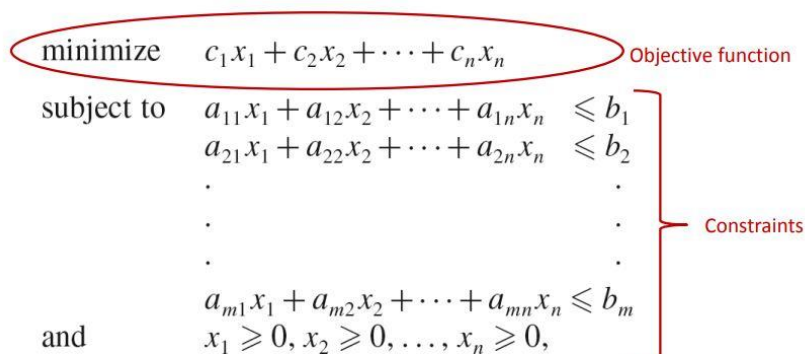
## 2.1 Background

We live in a world filled with finite resources, like time and money, and the goal is not to waste the resources given to use but use it in the best possible way, getting the most out of every resource. "Optimization is a way of living" [1], Linear programming is one of the most simplistic ways of optimizing. "Linear Programming Problems are concerned with the efficient use or allocation of limited resources to meet desired objectives." [2]

The aim was to use two linear programming techniques in our solver, to explain Linear programming in more detail we will be using these two techniques.

### 2.2.1 Simplex Method

The Simplex Method can be described as an iterative method that improves the solution after each "run". The process stops when it is not possible to improve the solution anymore. The goal of the function is to maximize or minimize the objective function based on the user's desire.



The diagram illustrates the components of a Simplex Method problem. It shows a minimization objective function, a set of linear constraints, and non-negativity conditions. The objective function is circled in red and labeled 'Objective function'. The constraints are grouped by a red bracket and labeled 'Constraints'. The non-negativity conditions are listed below the constraints.

$$\begin{array}{ll} \text{minimize} & c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ \text{subject to} & \begin{array}{l} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leq b_2 \\ \cdot \\ \cdot \\ \cdot \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq b_m \end{array} \\ \text{and} & x_1 \geq 0, x_2 \geq 0, \dots, x_n \geq 0, \end{array}$$

$c_1, c_2, \dots, c_n$  are the cost coefficients

$x_1, x_2, \dots, x_n$  are the decision variables (the unknowns to be determined)

$a_{11}, a_{12}, \dots, a_{mn}$  are the constraint coefficients

$b_1, b_2, \dots, b_m$  are the constraint requirements (right-hand-side values)

**FIGURE 1: COMPONENTS OF SIMPLEX**

As you will notice in Figure 1 above the simplex method is designed to work any number of variables subjected to any number of constraints. The simplex makes use of slack variables and Matrices do get to the optimal solution.

## 2.2.2 Branch and Bound

The branch and bound is typically used to solve Integer programming problems, however, it is not limited to only solving these types of problems. Therefore, it could be described as a solution approach that can be applied to several different types of problems. It works on the principle divide and conquer, the solutions are broken down into smaller subsets and are evaluated until the optimal solution given Certain parameters are found.

To best illustrate the branch and bound method in terms of calculations and programming will be to use Figure 2. Figure 2 illustrates a tree data structure. The first “circle” you will see represents the Root node, the two nodes underneath will be the children nodes, making the root node also a parent node. The nodes contain information of the variables, upper-bound (UB) and lower-bound(LB). How to find this info:

- The UB is found by calculating the best solutions Linearly (in terms of our project using Simplex) for the number of variables,  $x_1, x_2, \dots, x_n$ , given all the restrictions and information, and applying this info to the “Cost function” to find the maximum possible solution.
- LB is found rounding down the values of  $x_1, x_2, \dots, x_n$ , found with the Linear programming calculation, to the nearest possible integer and calculating the “Cost Function” accordingly.

The process thereafter is to see which variable is furthest from the previous or next possible integer and use the rounded up and rounded down integers to add a new parameter. (For example  $x_1 = 5.6$  and  $x_2 = 1.3$ , we will round  $x_1$  up and down to find the parameters of our new nodes). The idea is to compare branches coming from the parent node and see which delivers the higher value inside the UB and LB of the root node. This is done until the highest possible solution is found with only integers.

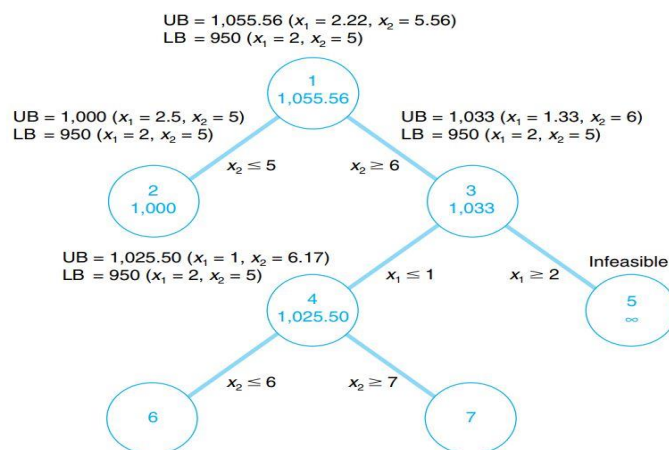
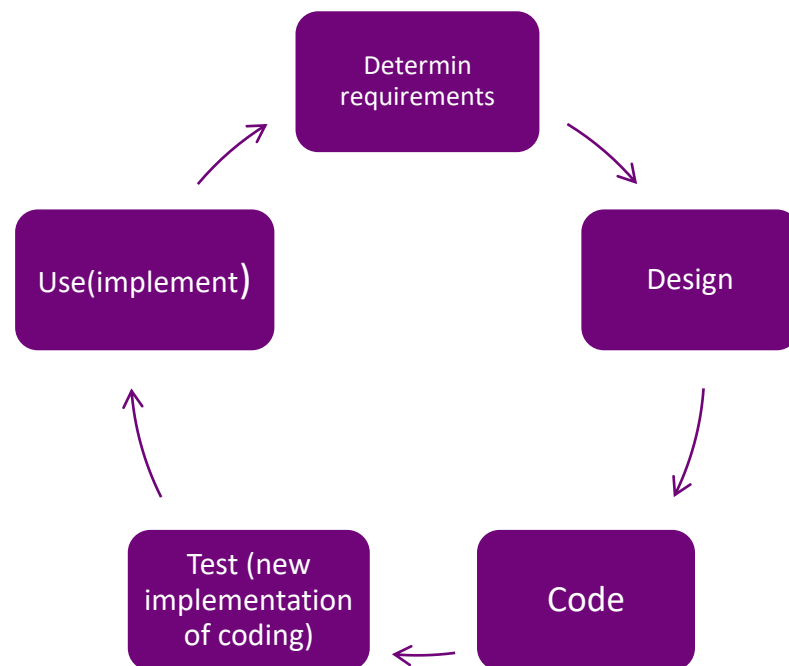


FIGURE 2: TREE

## 2.2.3 Methodology

The research surrounding the project included, how to develop a GUI in Qt, the theory and calculations of each method required and, finally the work covered in REll 313. The data was collected from textbooks, slides, research papers and questions frequently asked concerning the subjects on interactive sites. Most of the research collected beforehand was gathered through concerns of limitations I thought I might have during the development, however, like

our lecturer mentioned at the start of the year: “You will only understand the theory when you can apply the work in programming”. Out of this project I’ve learned new skills. What follows is a graphical representation of the method followed.



**FIGURE 3: AGILE DEVELOPMENT**

The Methodology described in Figure 3 was used as an overall method and strategy, but it was also used as a step by step method in developing the process. The strategy was to move forward slowly and focus on one thing at a time, broken up there were three main Goals:

- Developing a good interactive graphical user interface.
- Getting the desired input from the GUI.
- Implementing the Maximization and Minimization of the simplex.

## 3.1 GUI Interface

The GUI or graphic user interface is a window where the user can interact with the program. The GUI is set up in such a way to get information from the user. This is information regarding the Linear problem.

### 3.1.1 The Main Window

When running the application, the program will open up to the first window, the “MainWindow”, (see Figure 4 below):

**FIGURE 4: LINEAR PROBLEM INFORMATION (MAIN WINDOW).**

### 3.1.1.1 Number of Variables and Constraints

You will notice that the application prompts the user to enter two elements, either making use of the spin boxes or manually typing each desired element. The user then enters the number of variables in the problem, followed by the number of constraints (the number of equations excluding the objection function). After the user has entered the desired element they would proceed to press enter. By pressing the enter button below the Variable and Constraint prompt, the program not only receives the number of variables and constraints but this button also enables all the other buttons that were previously disabled (meaning all the other buttons in the Main Window).

### 3.1.1.2 Enter Objective Function- Button

When pressing this button, it will open the objective function window (see Figure 5).

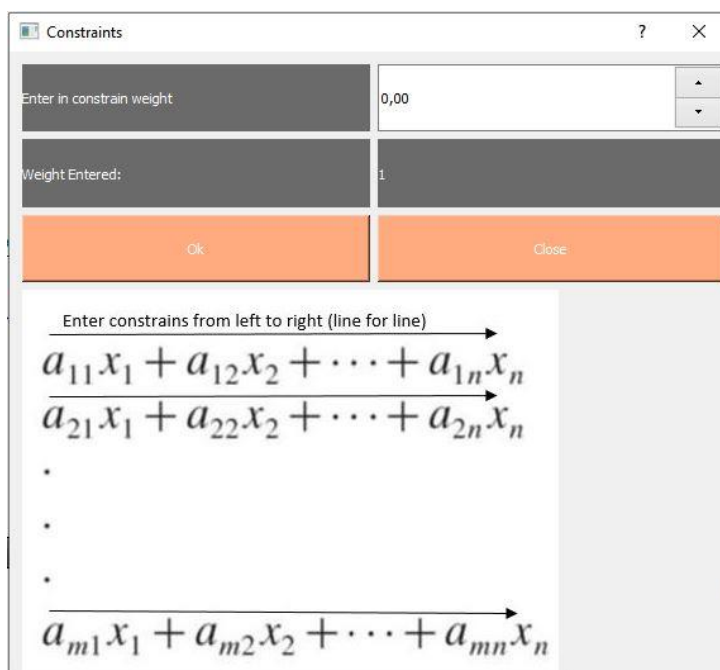


**FIGURE 5: OBJECTIVE FUNCTION WINDOW**

This window allows the user to enter in the different weights of the objective function corresponding with their related variable. Each weight is entered when the “Ok button is pressed. When the user is done they should press either the close button or the X button in the top right corner. The user can insert more weights than the number of variables entered however, the program will only work with the number of weights equal to the number of variables inserted in the main window.

### 3.1.1.3 Enter Constraint Button

By pressing this button the user opens a new window and is prompted to enter the weight of each variable in the constraints. The order in which the user should enter each constraint is specified by an image, the image indicated that weight should be entered row for row, the procedures regarding the buttons of the Objective Function window is also applicable in this window (see 3.1.1.2 Enter Objective Function- Button).

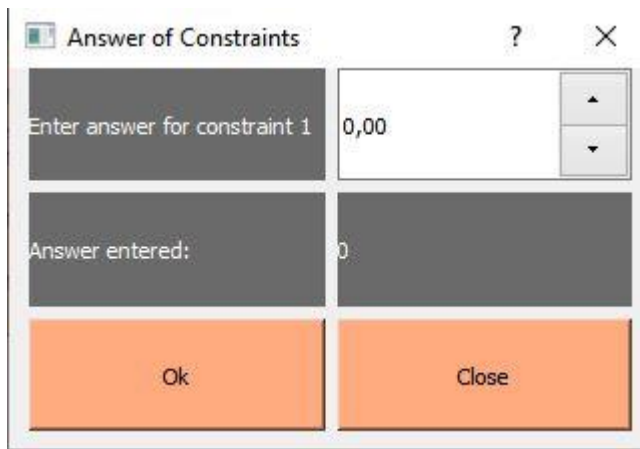


**FIGURE 6: CONSTRAINT WINDOW**



### 3.1.1.4 Answers Button

The answer button opens a window that will allow the user to enter a constant to which each constraint is subjected to, or in other terms the answer of each equation.



**FIGURE 7: ANSWER OF CONSTRAINT WINDOW**

The “Ok” and “Close” buttons and other procedures are the same as the other “sub-windows” (see 3.1.1.2 Enter Objective Function- Button).

### 3.1.1.5 Minimize and Maximization Buttons

These functions allow the user to choose whether they would like to find the optimal maximum or minimum solution. Meaning when the user enters Maximize the program assumes that all the equations look like Figure 8 below:

$$\begin{array}{rclcl} -x & + & y & \leq & 11 \\ x & + & y & \leq & 27 \\ 2x & + & 5y & \leq & 90 \end{array}$$

**FIGURE 8: MAXIMIZATION ASSUMPTION EXAMPLE**

When the user presses the Minimize button the program assumes all the constraints are bigger than the constant, as illustrated in the example below:

$$\begin{array}{rclcl} 60x_1 & + & 60x_2 & \geq & 300 \\ 12x_1 & + & 6x_2 & \geq & 36 \\ 10x_1 & + & 30x_2 & \geq & 90 \end{array}$$

**FIGURE 9: MINIMIZATION ASSUMPTION EXAMPLE**

### 3.1.1.6 Display Button

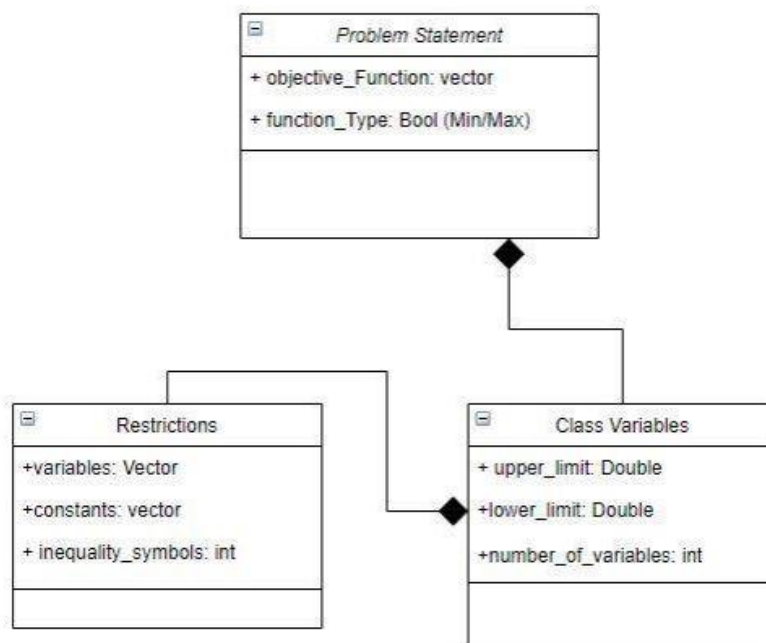
This button displays what the user requested from the GUI in the display box beneath the button. Meaning the minimization or maximization of the linear problem, whichever was

requested. If the user pressed the Minimization button and the display is will show Maximization of the problem as zero, and vice versa. However, when the user presses bot Minimization and Maximization GUI will show results for both requests.

## 3.2 Limitations

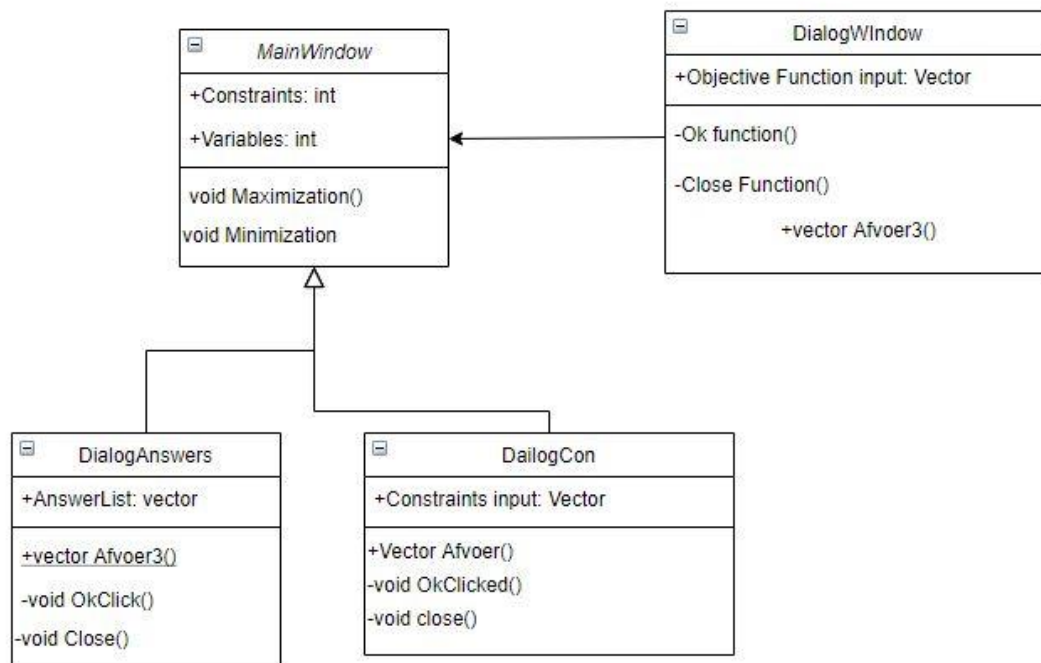
The program will not correctly execute a problem with different conditions, meaning the application needs all the constraints to be either greater than or less than.

## 3.3 UML



**FIGURE 10: UML (IN THEORY)**

Figure 10 shows the UML that I believed the program will look like, after looking deeper into a solver system the UML might look a bit more intense in response to the actual code and functions that needs to be used.



**FIGURE 11: PRACTICAL UML**

The UML seen in Figure 11 is the UML based on the actual program. The program differs from the original design because the original design was drawn up with the lack of knowledge of the requirements of the project. However, we all know that the final product usually differs from the original design.

### 3.4 Test

In this section, we will be looking at different tests for each request of the application. The first test will show each step of the problem inserted and thereafter include only examples, desired results, and obtained outputs.

#### 3.4.1 Maximization Test 1

$$\max 20x_1 + 14x_2$$

$$-2x_1 + x_2 \leq 300$$

$$x_1 + 4.3x_2 \leq 500$$

$$6x_1 + 3x_2 \leq 800$$

$$x_i \geq 0, \quad i = 1, 2$$

**FIGURE 12: EXAMPLE 1 MAXIMIZATION**

### 3.4.1.1 Input

#### Variables and Constraints:

From the problem you will note there are three constraints and two variables, the input in the main window should look like this (see Figure 13):

Enter number of constraints:	3
Number of constraints:	3
Enter number of Variables	2
Number of variables:	2
Enter	

**FIGURE 13: EXAMPLE 1 VARIABLES AND CONSTRAINTS**

Objective Function	
Weight of x1:	20,00
Value set:	0
Ok	Close

Objective Function	
Weight of x2:	14,00
Value set:	20,00
Ok	Close

**FIGURE 14: INPUT ORDER OF OBJECTION FUNCTION**

The figure shows a sequence of six dialog boxes for entering constraint weights. Each dialog box contains two input fields: 'Enter in constrain weight' and 'Weight Entered'. The 'Weight Entered' field is pre-filled with a value, and the 'Enter in constrain weight' field is for manual input. The sequence is as follows:

- Dialog 1: 'Enter in constrain weight' is 2,00; 'Weight Entered' is 1.
- Dialog 2: 'Enter in constrain weight' is 1,00; 'Weight Entered' is 1,00.
- Dialog 3: 'Enter in constrain weight' is 1,00; 'Weight Entered' is -2,00.
- Dialog 4: 'Enter in constrain weight' is 4,30; 'Weight Entered' is 1,00.
- Dialog 5: 'Enter in constrain weight' is 6,00; 'Weight Entered' is 1,00.
- Dialog 6: 'Enter in constrain weight' is 3,00; 'Weight Entered' is 1,00.

**FIGURE 15: ORDER OF INPUT OF CONSTRAINTS**

The figure shows a sequence of three dialog boxes for entering answers for constraints. Each dialog box contains two input fields: 'Enter answer for constraint' and 'Answer entered'. The 'Answer entered' field is pre-filled with a value, and the 'Enter answer for constraint' field is for manual input. The sequence is as follows:

- Dialog 1: 'Enter answer for constraint 1' is 300,00; 'Answer entered' is 0.
- Dialog 2: 'Enter answer for constraint 2' is 500,00; 'Answer entered' is 300,00.
- Dialog 3: 'Enter answer for constraint 3' is 800,00; 'Answer entered' is 500,00.

**FIGURE 16: ORDER OF INPUT ANSWERS**

### 3.4.1.2 Desired Result and Obtained Result

Figure 17 below shows the results for the three iterations needed and the result obtained when calculating the Linear Problem.

Objective function value	Maximum?
0	No
2666.67	No
3052.63	Yes

**FIGURE 17: DESIRED RESULT**

The screenshot shows a window titled "Simplex Method". It contains the following elements:

- Input fields: "Enter number of constraints:" with value 3, "Number of constraints:" with value 3, "Enter number of Variables" with value 2, and "Number of variables:" with value 2.
- Buttons: "Enter", "Enter Objective Function", "Enter Constraints", "Answers", "Maximize", "Minimize", and "Display".
- Output area: "The maximum solution is: 3052.63" and "The minimum solution is: 0".

**FIGURE 18: RESULT FOR EXAMPLE ONE**

### 3.4.2 MORE TESTS

Problem	Desired Result	Result obtained through the program:
Max. $Z = 4x_1 + 6x_2$ $-x + y \leq 11$ $x + y \leq 27$ $2x + 5y \leq 90$	$Z = 132$	<div>Display</div> <div>The maximum solution is: 132</div> <div>The minimum solution is: 0</div>
Min. $Z = 6x + 8y$ $40x + 10y \geq 2400$ $10x + 15y \geq 2100$ $5x + 15y \geq 1500$	$Z = 1140$	<div>Display</div> <div>The maximum solution is: 0</div> <div>The minimum solution is: 1140</div>

Min. $Z = 3x + 9y$  $2x + y \geq 8$ $1x + 2y \geq 8$	$Z = 24$	<div data-bbox="691 194 1393 230" data-label="Section-Header"> Display </div> <div data-bbox="691 230 1393 338" data-label="Text"> The maximum solution is: 0 </div> <div data-bbox="691 338 1393 448" data-label="Text"> The minimum solution is: 24 </div>
---	----------	--

## 4. Conclusion

A solver is a very good application to have, especially when it comes to Industrial Engineers where optimization plays such a large part. Linear problems could get very difficult to do on paper, not impossible, but errors could easily be made. Mistakes are humane, therefore taking the human aspect out of calculations ensures you'll get always get the right answer that will be easy to test.

## 5. References

- [1] S. R. Gass SI, Linear Programming Methods of application, New York: Dover Publications, Inc., 2003.
- [2] Anonymous, "Analytics Vidhya," 26 February 2017. [Online]. Available: <https://www.analyticsvidhya.com/blog/2017/02/introductory-guide-on-linear-programming-explained-in-simple-english/>.