Списание „Математика, компютърни науки и образование“ Том 5, кн. 1, 2022
Journal of Mathematics, Computer Science and Education Vol. 5, No. 1, 2022

DOI: 10.54664/YGJS5089

SimplexAlgo Software for Solving Problems in Linear Programming

Endrit Rushiti, Axhi Beqiri,
Egzona Iseni, Shpetim Rexhepi

Abstract: *The paper describes the development of a piece of software called "SimplexAlgo", which calculates and solves a simplex algorithm. We have created this software for educational purposes in linear programming, which is a technique for the optimization of a linear objective function, subject to linear equality and linear inequality constraints, and as such is widely used in many fields in operation research. SimplexAlgo has been developed with the C# programming language in Visual Studio desktop. To execute an exercise, the software is manually coded according to the methods and ways studied and learned. To achieve the desired result, various attributes are used, as well as network connections from where we get the steps of a function (step -step-step). The application has a special algorithm to solve an exercise and provide solutions.*

Comparisons have been made regarding the speed of execution of the algorithm of this software with the existing ones, where it has been established that the execution time is faster.

Keywords: *linear programming; standard form; simplex table; C# programming; Visual Studio.*

Introduction

The daily activity of businesses is full of different situations which must be managed in the most rational way. The most typical situation that a manager encounters in his/her work is that of using the resources he/she has in order to be good, so that the company he/she leads achieves the best results, maximizes profit, and minimizes cost. One way to facilitate the solution of these problems is the presentation by means of mathematical models and the use of quantitative methods for their solution.

A relevant issue of great importance for solving optimization problems is linear programming. Linear programming is a mathematical method for determining a way to achieve the best result (such as maximum profit or lowest cost) in a given mathematical model for some list of requirements presented as linear relationships. The earliest LP was first developed by Leonid Kantorovich in 1939 [8]. Linear programming was set up as a mathematical model, developed during World War II to plan expenditures and returns in order to reduce army costs and increase losses to the enemy. It was kept a secret until 1947. After the War, many industries found its use in their daily planning.

The applications of linear programming models include, but are not limited to: (1) the diet problem; (2) portfolio optimization; (3) crew scheduling; (4) manufacturing and transportation; (5) telecommunications; and (6) the Travelling Salesman Problem [1,5]. In 1963, Dantzig's *Programming and Linear Extensions* was published by Princeton University Press. Rich in insight and coverage of important topics, the book quickly became the "Bible" of linear programming [3, 5, 6].

Developments in computer science have given a very important place to the use of quantitative models and techniques of operational research, making them an indisputable tool of business management. A major proportion of all scientific computations on computers is devoted to the use of linear programming [7].

In general, the use of quantitative methods and algorithms for problem solving requires processing of extensive information and performance of a large volume of calculations. The very keeping of information and its availability at the right time and place cannot be understood without the help of computer systems. On the other hand, the time required for one-time problem-solving calculations can be so long that the solution is no longer valid.

The use of computers and computer programs has greatly facilitated the solution of problems, even complex ones. Some mathematical models are so complex that it is impossible to solve them by means of suitable optimization algorithms. In such cases, it is necessary to abandon the search for the correct optimal solution and to look for an approximately good solution. However, it only takes a moment to find the optimal solution by presenting the problem as a linear program and by applying the Simplex algorithm. For the purposes mentioned, we have developed a piece of software called “SimplexAlgo”.

One of the innovations brought by the use of this software is the shortening of time in solving the most complex problems that arise in the field of linear programming. During problem solving, it helps students compare problem solving with the classical way of doing so.

For students, the software ensures specification of the problem and rationalization in time. Many software that exist to solve problems from linear programming also have their own usefulness in the number of variables that are presented and in the time of interpretation of the results. Specifically, our software SimplexAlgo helps to improve these vulnerabilities and, compared to other existing software that have free access and use of the internet, we have proven to be faster in executing solutions.

2. Methodology

In this section, we will describe in detail the problematics in linear programming and the steps of developing SimplexAlgo, giving instructions of using this software and some exceptions in this regard.

2.1. Transforming Problems of Linear Programming into Standard Form

Transforming problems of linear programming into a standard one can be accomplished as follows: First, for each variable with a lower limit other than 0, a new variable is introduced that represents the difference between the variable and the link. The original variable can be eliminated by replacement.

Before completing the problem, the latter should be brought to the standard treatment as follows: The complete programming problem turns linearly into standard standards as an exercise. This shape calls from the system of equations and the smallest value is required for the function to work. If it does not work to target f, the larger value is required when deciding to set the target $f' = -f$ as function and to search for smaller values. If the system requires inequalities, they can be turned into equations by adding other unknowns.

2.2. Simplex Table

The first line of the simplex table defines the objective function, and the remaining lines specify the constraints. The zero in the first column represents the zero vector of the same dimension as the b vector (different authors use different conventions for accurate representation). If the columns of A can be rearranged to contain the identity matrix of order p (number of rows in A), then the table is said to be in canonical form. The variables corresponding to the columns of the identity matrix are called “base variables”, while the remaining variables are called “non-basic” or “free variables”. If the values of the non-basic variables are set to 0, then the values of the base variables are easily taken as notes in b, and this solution is a possible basic solution.

The algebraic interpretation here is that the coefficients of the linear equation represented by each row are either 0.1 or any other number. Each row will have one column with value 1, p-1 column with coefficient 0 and the remaining columns with some other coefficients (these other variables represent our non-basic variables).

By setting the values of the non-basic variables to zero, we ensure that, in each row, the value of the variable represented by 1 in its column is equal to the value of b in that row.

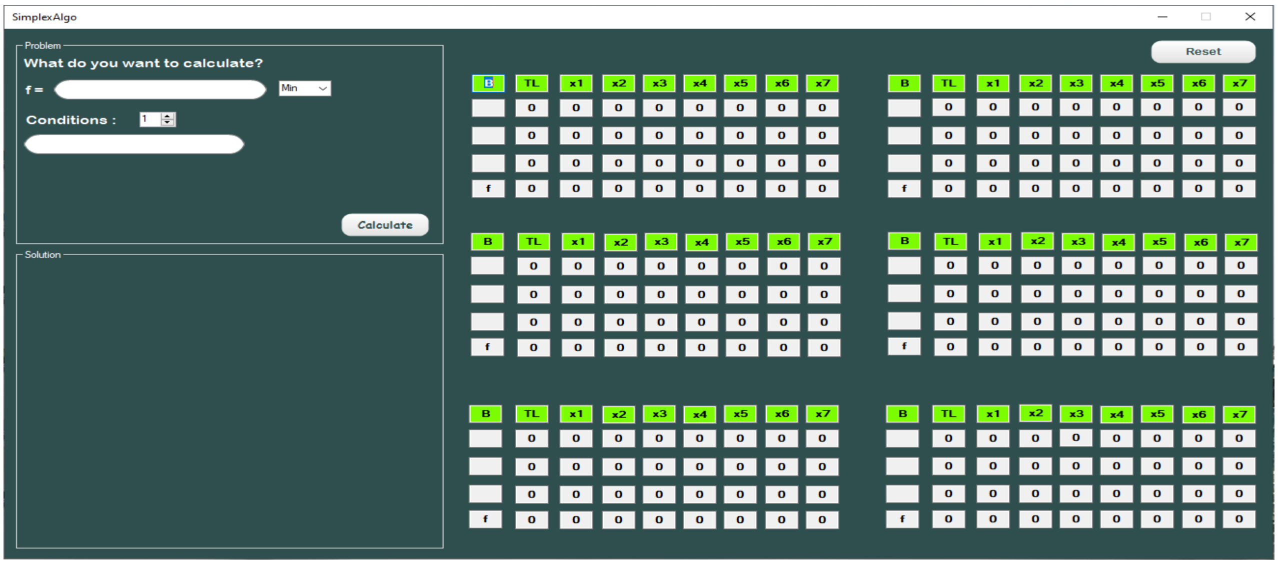
2.3. SimplexAlgo

We have developed this software for educational purposes, in mathematics in the field of linear programming, which is widely used, and in other scientific fields. We have called this software "SimplexAlgo", and it calculates and solves a simplex algorithm.

SimplexAlgo has been developed with the C# programming language in Visual Studio desktop. To execute a problem or an exercise, the software is manually coded according to the methods and ways studied and learned. To achieve the desired result, various attributes are used, as well as network connections from where we get the steps of a function (step -step-step). The application has a special algorithm to solve a problem and provide solutions.

• UI

The UI (User Interface) is very simple and easy to use. In order to help users operate easily in the app, we made the software only in one screen, meaning that every button and every number that you need to input or see is already there in one screen.



B	TL	x1	x2	x3	x4	x5	x6	x7
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
f	0	0	0	0	0	0	0	0

B	TL	x1	x2	x3	x4	x5	x6	x7
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
f	0	0	0	0	0	0	0	0

B	TL	x1	x2	x3	x4	x5	x6	x7
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
f	0	0	0	0	0	0	0	0

B	TL	x1	x2	x3	x4	x5	x6	x7
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
f	0	0	0	0	0	0	0	0

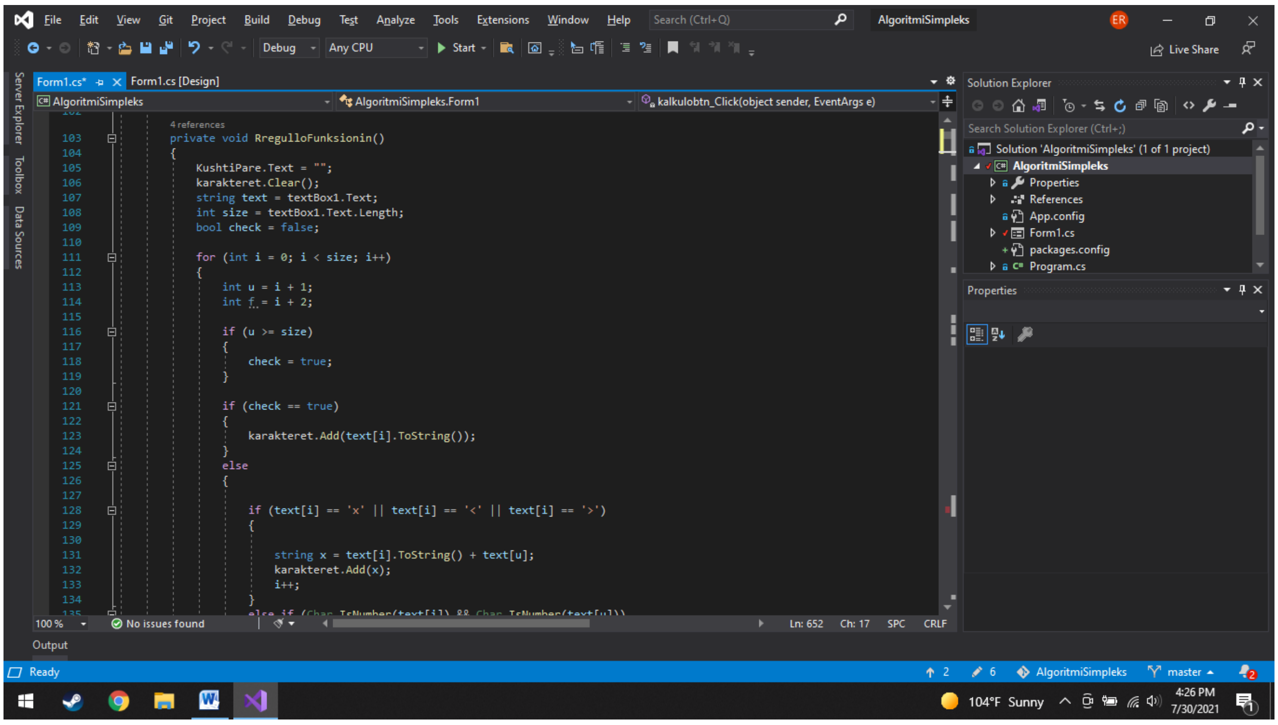
B	TL	x1	x2	x3	x4	x5	x6	x7
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
f	0	0	0	0	0	0	0	0

B	TL	x1	x2	x3	x4	x5	x6	x7
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0
f	0	0	0	0	0	0	0	0

• Code

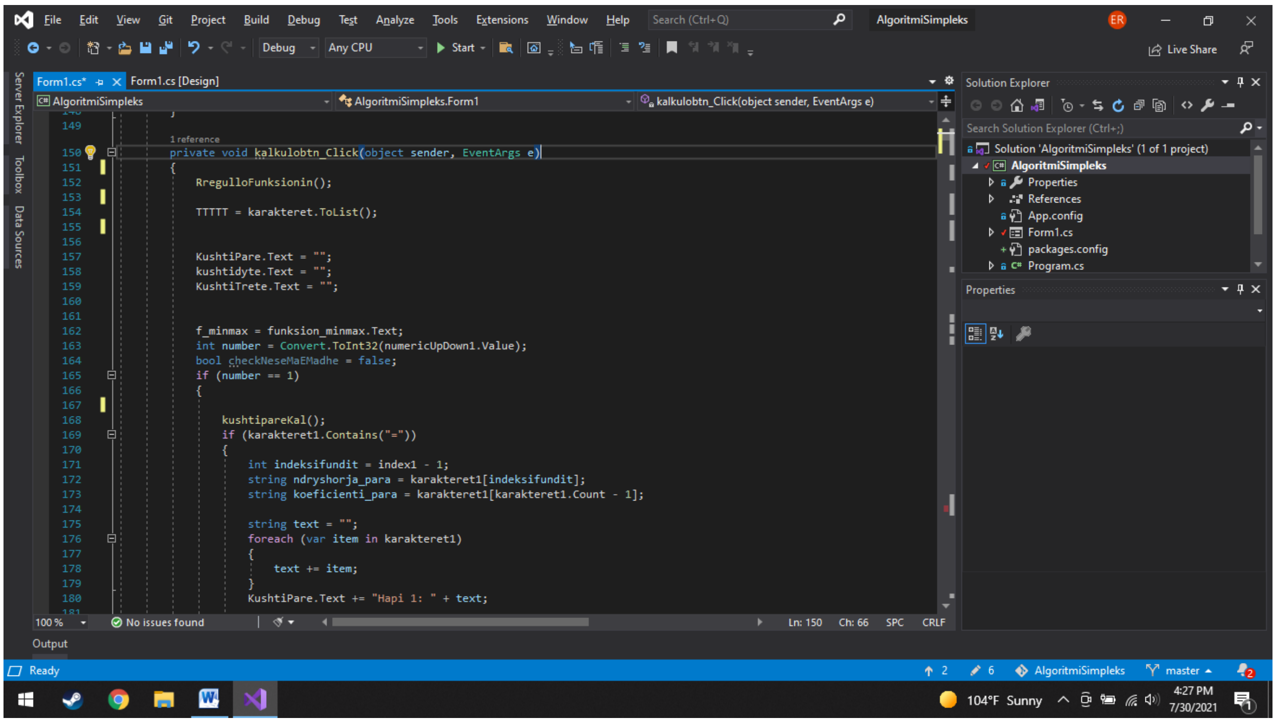
The programming language that we used to code this software is C# and we used Visual Studio as an IDE (Integrated Development Environment). We used C#, because we learned this programming language at university lessons and wanted to use in this software only what we learned at university, and also that C# is a programming language used to code desktop programs. The code is also a clean code, which is divided in some main functions like "Calcuate", "FindTheKey", "FillTheTables", etc.

The code has more than 12 500 rows, so it is impossible to describe them here, but we have provided some screenshots below.



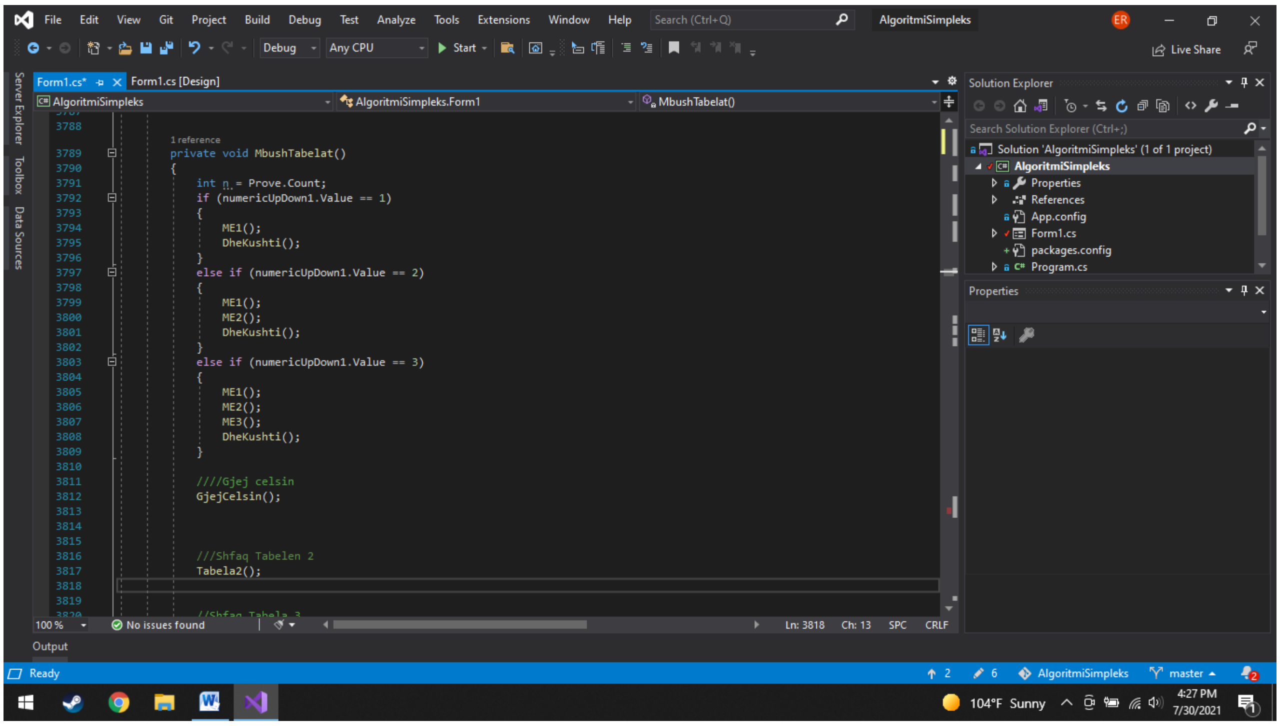
```
        private void RregulloFunksionin()
        {
            KushtiPare.Text = "";
            karakteret.Clear();
            string text = textBox1.Text;
            int size = textBox1.Text.Length;
            bool check = false;

            for (int i = 0; i < size; i++)
            {
                int u = i + 1;
                int f = i + 2;

                if (u >= size)
                {
                    check = true;
                }

                if (check == true)
                {
                    karakteret.Add(text[i].ToString());
                }
                else
                {

                    if (text[i] == 'x' || text[i] == '<' || text[i] == '>')
                    {

                        string x = text[i].ToString() + text[u];
                        karakteret.Add(x);
                        i++;
                    }
                    else if (Char.IsNumber(text[i]) && Char.IsNumber(text[u]))
```

Reformat the function



```
        private void kalkulobtn_Click(object sender, EventArgs e)
        {
            RregulloFunksionin();

            TTTTT = karakteret.ToList();


            KushtiPare.Text = "";
            kushtidyte.Text = "";
            KushtiTrete.Text = "";


            f_minmax = funksion_minmax.Text;
            int number = Convert.ToInt32(numericUpDown1.Value);
            bool checkNeseMaEMadhe = false;
            if (number == 1)
            {

                kushtipareKal();
                if (karakteret1.Contains("="))
                {
                    int indeksifundit = index1 - 1;
                    string ndryshorja_para = karakteret1[indeksifundit];
                    string koeficienti_para = karakteret1[karakteret1.Count - 1];

                    string text = "";
                    foreach (var item in karakteret1)
                    {
                        text += item;
                    }
                    KushtiPare.Text += "Hapi 1: " + text;
```

Calculate the function



```
        1 reference
        private void MbushTabelat()
        {
            int n = Prove.Count;
            if (numericUpDown1.Value == 1)
            {
                ME1();
                DheKushti();
            }
            else if (numericUpDown1.Value == 2)
            {
                ME1();
                ME2();
                DheKushti();
            }
            else if (numericUpDown1.Value == 3)
            {
                ME1();
                ME2();
                ME3();
                DheKushti();
            }

            ////Gjej celsin
            GjejCelsin();



            ///Shfaq Tabelen 2
            Tabela2();


            //Shfaq Tabela 3
```

Fill the tables

In the following steps, we give instructions for using the SimplexAlgo software:

1. Start the SimplexAlgo application;
2. In the *Problem* section, enter the function and fill in the conditions;
3. Once you have filled in the data in the *Problem* section, press the *Calculate* button and wait;
4. At the bottom, in *Solution*, you will see the solution of the exercises, as well as the calculations given step by step.
5. After the solution appears, the tables will be filled.
6. In the tables, the green cells are the ones that change, while the blue cells are the keys.
7. If you want to start a new exercise, you will have to press the *Reset* button and you will be able to start from the beginning.

In the following points, we present some exceptions in using SimplexAlgo:

1. To use the application, you must have internet access.
2. When you do not have a free term in the function, you will definitely need to add a 0 as a free term.
3. When we do not have a coefficient before a variable, we will have to add coefficient 1 (Example: 1x1).

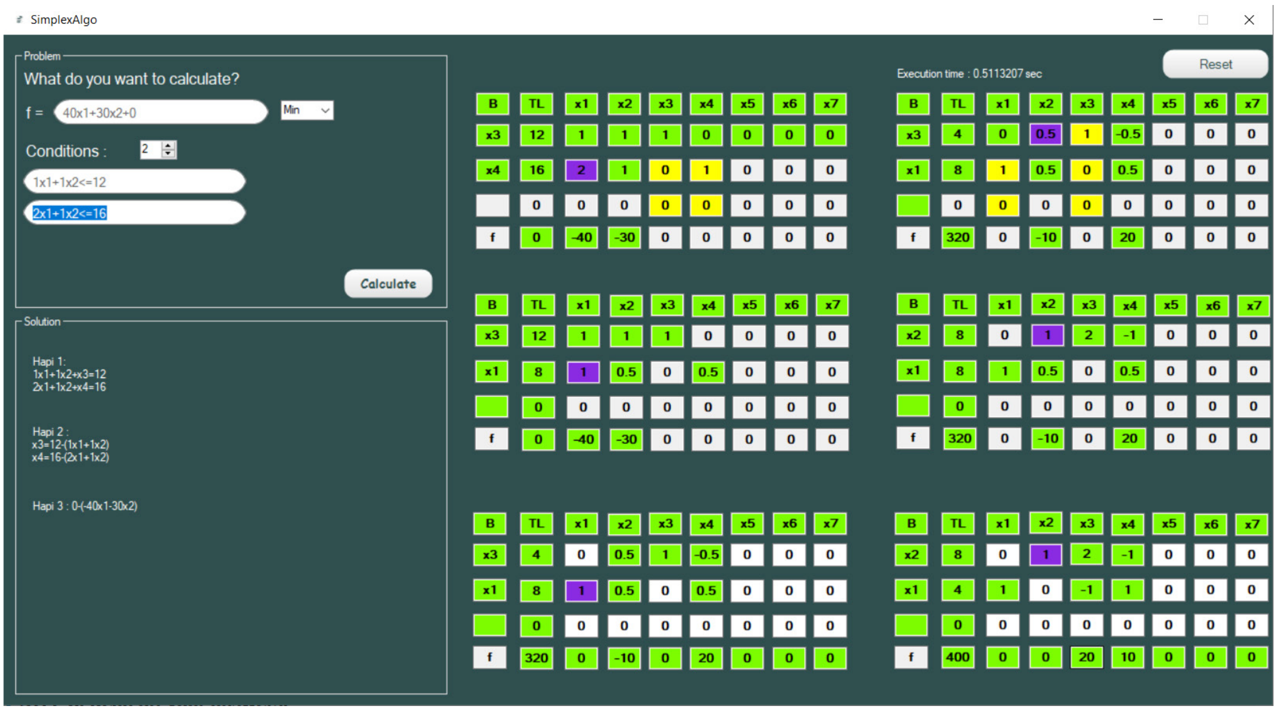
3. Results and Discussion

In this section, we will provide examples of executing exercises through the SimplexAlgo software, illustrated with tables. We also show how this software works and compare it with existing software with free internet access, such as Online Calculator: Simplex Method, clearly distinguishing the advantages that our software has both in the limitations of operation and in the time of execution.

3.1. Execution and Numerical Illustrations

Here we have examples of how SimplexAlgo is used and what it looks like when executing an exercise.

Example 1



SimplexAlgo

Problem

What do you want to calculate?

f = 40x1+30x2+0 Min

Conditions : 2

1x1+1x2<=12

2x1+1x2<=16

Calculate

Solution

Hapi 1:
1x1+1x2+x3=12
2x1+1x2+x4=16

Hapi 2 :
x3=12-(1x1+1x2)
x4=16-(2x1+1x2)

Hapi 3 : 0-(-40x1-30x2)

Execution time : 0.5113207 sec

Reset

B	TL	x1	x2	x3	x4	x5	x6	x7
x3	12	1	1	1	0	0	0	0
x4	16	2	1	0	1	0	0	0
	0	0	0	0	0	0	0	0
f	0	-40	-30	0	0	0	0	0

B	TL	x1	x2	x3	x4	x5	x6	x7
x3	4	0	0.5	1	-0.5	0	0	0
x1	8	1	0.5	0	0.5	0	0	0
	0	0	0	0	0	0	0	0
f	320	0	-10	0	20	0	0	0

B	TL	x1	x2	x3	x4	x5	x6	x7
x3	12	1	1	1	0	0	0	0
x1	8	1	0.5	0	0.5	0	0	0
	0	0	0	0	0	0	0	0
f	0	-40	-30	0	0	0	0	0

B	TL	x1	x2	x3	x4	x5	x6	x7
x2	8	0	1	2	-1	0	0	0
x1	8	1	0.5	0	0.5	0	0	0
	0	0	0	0	0	0	0	0
f	320	0	-10	0	20	0	0	0

B	TL	x1	x2	x3	x4	x5	x6	x7
x3	4	0	0.5	1	-0.5	0	0	0
x1	8	1	0.5	0	0.5	0	0	0
	0	0	0	0	0	0	0	0
f	320	0	-10	0	20	0	0	0

B	TL	x1	x2	x3	x4	x5	x6	x7
x2	8	0	1	2	-1	0	0	0
x1	4	1	0	-1	1	0	0	0
	0	0	0	0	0	0	0	0
f	400	0	0	20	10	0	0	0

Step 1:

The first function has the sign “<=”, and for this reason the software adds “x3”, equates it, and puts the sign “=”. The second function has the sign “< =”, and therefore the software adds the “x4” and equalizes the sign “=”.

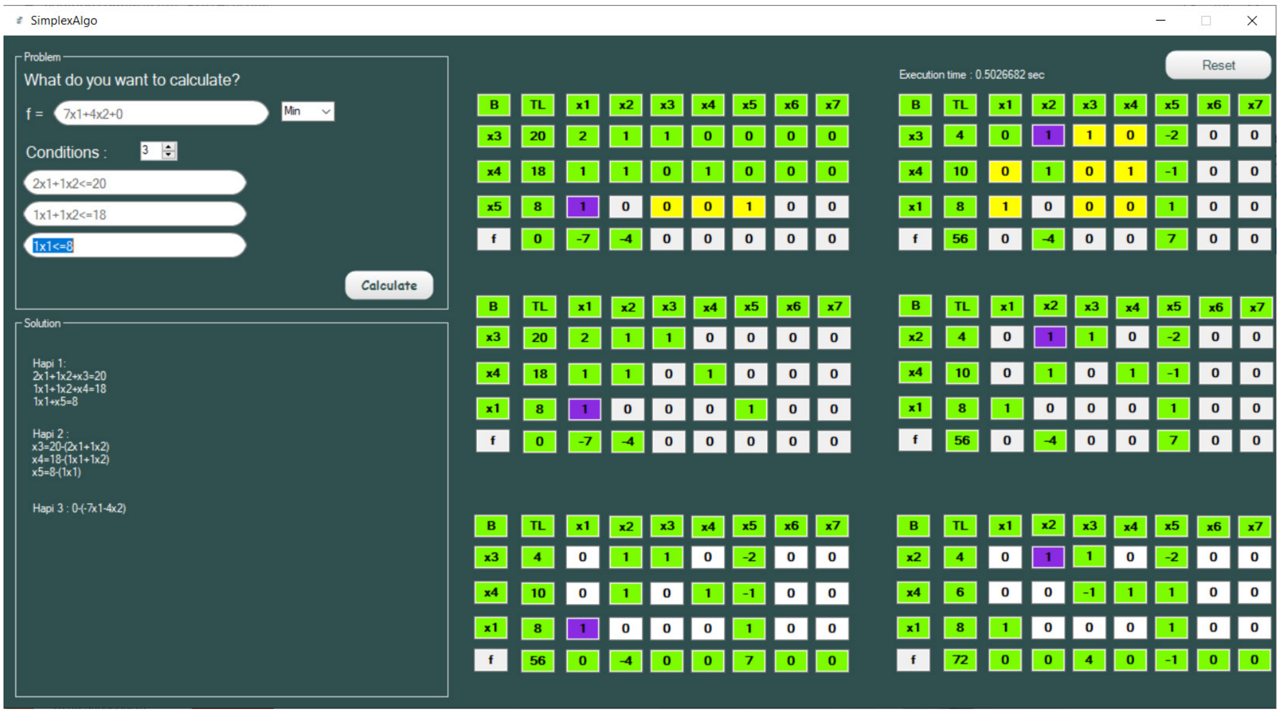
Step 2:

In the first condition, “x3” is taken out from the left side and the rest is taken out from the right side of the equation, and, after this change, we get the following view of the first condition: x3 = 12-(1x1 + 1x2). The same thing happens with the second condition, i.e. “x4” is drawn from the left side of the equation, while the rest comes out from the right side, and we get this view of the condition: x4 = 16- (2x1 + 1x2).

Step 3:

Check if “x3” or “x4” is in the function, then replace it with the value of “x3” or “x4” it received from Step 2; in this case, “x3” or “x4” is not part of the function, and Step 3 appears in this form by removing the free member from the last position to the first position: 0- (-40x1 - 30x2). In the upper right part next to the *Reset* button, we have *Execution time* which tells us how many seconds the exercise was executed for, and, as we can see, in Example 1 it was executed for 0.5113 sec.

Example 2



f = 7x1+4x2+0 Min

Conditions : 3

2x1+1x2<=20

1x1+1x2<=18

1x1<=8

Hapi 1:
2x1+1x2+x3=20
1x1+1x2+x4=18
1x1+x5=8

Hapi 2 :
x3=20-(2x1+1x2)
x4=18-(1x1+1x2)
x5=8-(1x1)

Hapi 3 : 0-(-7x1-4x2)

Execution time : 0.5026682 sec

B	TL	x1	x2	x3	x4	x5	x6	x7
x3	20	2	1	1	0	0	0	0
x4	18	1	1	0	1	0	0	0
x5	8	1	0	0	0	1	0	0
f	0	-7	-4	0	0	0	0	0

B	TL	x1	x2	x3	x4	x5	x6	x7
x3	4	0	1	1	0	-2	0	0
x4	10	0	1	0	1	-1	0	0
x1	8	1	0	0	0	1	0	0
f	56	0	-4	0	0	7	0	0

B	TL	x1	x2	x3	x4	x5	x6	x7
x3	20	2	1	1	0	0	0	0
x4	18	1	1	0	1	0	0	0
x1	8	1	0	0	0	1	0	0
f	0	-7	-4	0	0	0	0	0

B	TL	x1	x2	x3	x4	x5	x6	x7
x2	4	0	1	1	0	-2	0	0
x4	10	0	1	0	1	-1	0	0
x1	8	1	0	0	0	1	0	0
f	56	0	-4	0	0	7	0	0

B	TL	x1	x2	x3	x4	x5	x6	x7
x3	4	0	1	1	0	-2	0	0
x4	10	0	1	0	1	-1	0	0
x1	8	1	0	0	0	1	0	0
f	56	0	-4	0	0	7	0	0

B	TL	x1	x2	x3	x4	x5	x6	x7
x2	4	0	1	1	0	-2	0	0
x4	6	0	0	-1	1	1	0	0
x1	8	1	0	0	0	1	0	0
f	72	0	0	4	0	-1	0	0

Step 1:

The first function has the sign "<=", and for this reason the software adds "x3", equates it, and puts the sign "=". The second function has the sign "< =", and therefore the software adds "x4" and equalizes the sign "=". The third condition also contains the sign "<=", and for this reason the software adds "x5", equates it, and puts the sign "=".

Step 2:

In the first condition, "x3" is drawn from the left side of the equation, while the rest from the right side, and it looks like this: x3 = 20- (2x1 + 1x2).

In the second condition, "x4" is drawn from the left side of the equation, while the rest from the right side, and it looks like this: x4 = 18- (1x1+1x2). In the third condition, "x5" is drawn from the left side of the equation, while the rest from the right side, and it looks like this: x5 = 8- (1x1).

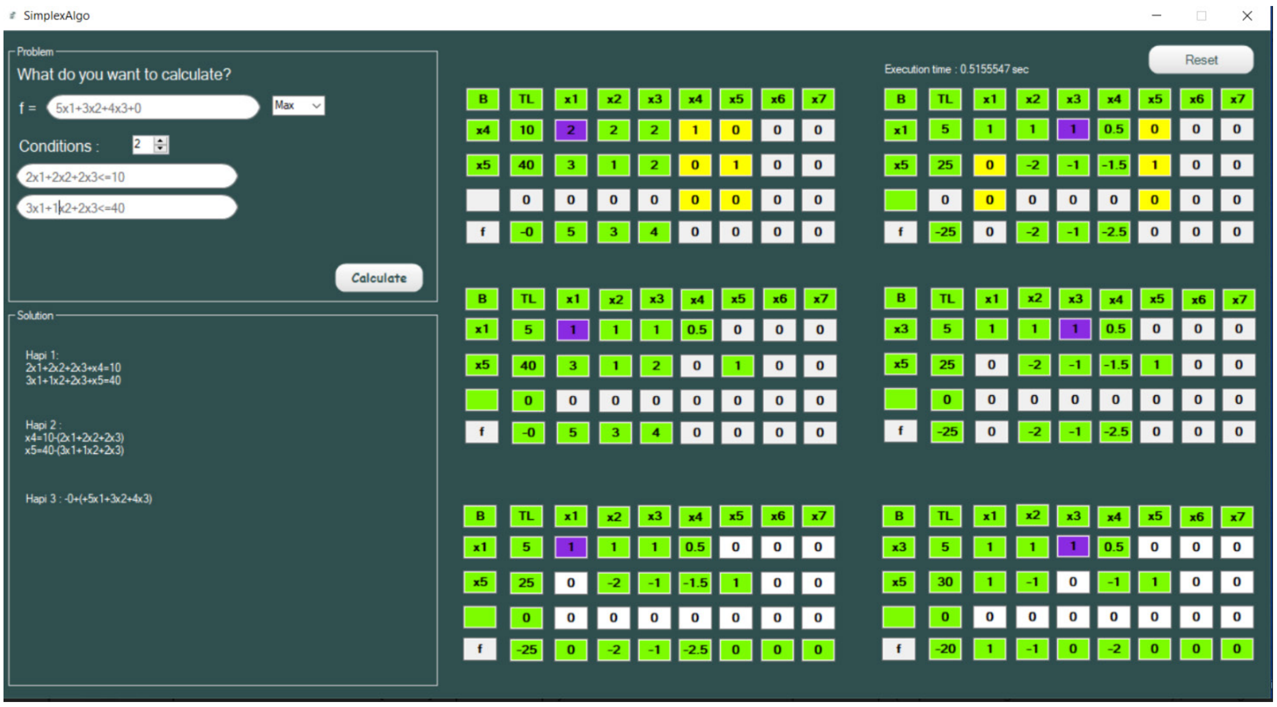
Step 3:

Check if "x3" or "x4" or "x5" is in the function, then replace it with the value of "x3" or "x4" or "x5" obtained from Step 2; in this case, "x3" or "x4" or "x5" is not part of the function, and Step 3 appears in this form by removing the free member from the last position to the first position: 0- (-7x1 - 4x2).

In the upper right part next to the *Reset* button, we have *Execution time* which tells us how many seconds the exercise was executed for, and, as we can see, in Example 2 it was executed in 0.5027 sec.

Next, we will take an example of unbounded functions.

Example 3



f = 5x1+3x2+4x3+0 Max

Conditions : 2

2x1+2x2+2x3<=10

3x1+1x2+2x3<=40

Hapi 1:
2x1+2x2+2x3+x4=10
3x1+1x2+2x3+x5=40

Hapi 2 :
x4=10-(2x1+2x2+2x3)
x5=40-(3x1+1x2+2x3)

Hapi 3 : -0+(+5x1+3x2+4x3)

Execution time : 0.5155547 sec

B	TL	x1	x2	x3	x4	x5	x6	x7
x4	10	2	2	2	1	0	0	0
x5	40	3	1	2	0	1	0	0
	0	0	0	0	0	0	0	0
f	-0	5	3	4	0	0	0	0

B	TL	x1	x2	x3	x4	x5	x6	x7
x1	5	1	1	1	0.5	0	0	0
x5	25	0	-2	-1	-1.5	1	0	0
	0	0	0	0	0	0	0	0
f	-25	0	-2	-1	-2.5	0	0	0

B	TL	x1	x2	x3	x4	x5	x6	x7
x1	5	1	1	1	0.5	0	0	0
x5	40	3	1	2	0	1	0	0
	0	0	0	0	0	0	0	0
f	-0	5	3	4	0	0	0	0

B	TL	x1	x2	x3	x4	x5	x6	x7
x3	5	1	1	1	0.5	0	0	0
x5	25	0	-2	-1	-1.5	1	0	0
	0	0	0	0	0	0	0	0
f	-25	0	-2	-1	-2.5	0	0	0

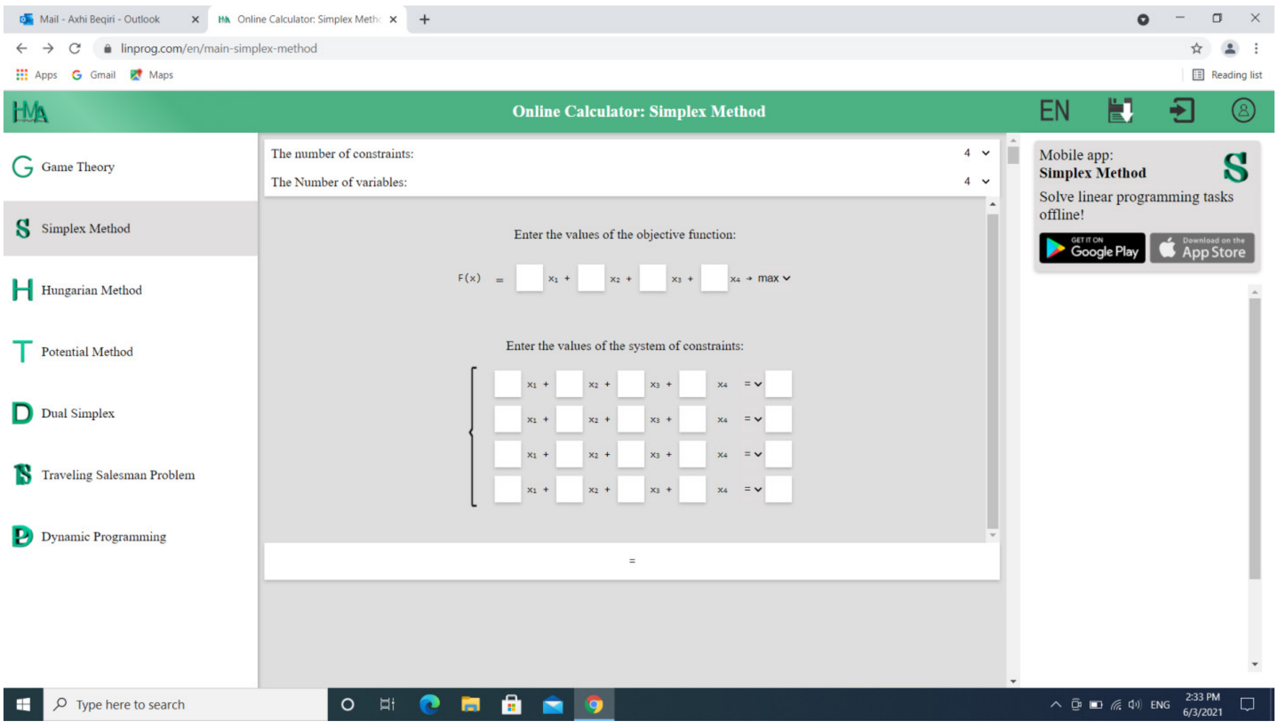
B	TL	x1	x2	x3	x4	x5	x6	x7
x1	5	1	1	1	0.5	0	0	0
x5	25	0	-2	-1	-1.5	1	0	0
	0	0	0	0	0	0	0	0
f	-25	0	-2	-1	-2.5	0	0	0

B	TL	x1	x2	x3	x4	x5	x6	x7
x3	5	1	1	1	0.5	0	0	0
x5	30	1	-1	0	-1	1	0	0
	0	0	0	0	0	0	0	0
f	-20	1	-1	0	-2	0	0	0

3.2. Discussion and Comparisons

There is a free access online software called “Online Calculator: Simplex Method”, which solves the same problem. The execution time of this software measured manually (by phone) is around 0.6 – 0.8 sec, i.e. we cannot get it 100% accurate. Since the software is on the network, the execution time also depends on the speed of the network to which we are connected.

Online Calculator: Simplex Method is slightly different from SimplexAlgo, because here we have limitations – we must have four unknowns (x1, x2, x3, x4) and four conditions for the software to answer us. On the other hand, in SimplexAlgo we do not have these limitations, and no matter how many conditions or unknowns we have, the software gives us results.


Mail - Axhi Beqiri - Outlook
Online Calculator: Simplex Meth
linprog.com/en/main-simplex-method
Apps
Gmail
Maps
Reading list
Online Calculator: Simplex Method
EN
Game Theory
Simplex Method
Hungarian Method
Potential Method
Dual Simplex
Traveling Salesman Problem
Dynamic Programming
Enter the values of the objective function:
Enter the values of the system of constraints:
Mobile app:
Simplex Method
Solve linear programming tasks offline!
Google Play
App Store
Type here to search
ENG
2:33 PM
6/3/2021

The number of constraints: 4

The Number of variables: 4

F(x) = x_1 + x_2 + x_3 + x_4 → max

Conclusion

From the illustrations with particular exercises from linear programming it is clear how Simplex-Algo works in practice, how easy it is to use, and how practical it is. Given the comparisons made with existing software, the one presented by us has some advantages in terms of limitations that were presented in operation, as well as in terms of execution time.

This software has also aroused the interest of the students we work with who compare the exercises in the class they choose in a classical way with the solution offered by the software, thus making the subject more attractive and more interesting.

REFERENCES

[1] **Aboelmagd, Y. M.** 2022. Corrigendum to "Linear Programming Applications in Construction Sites" [*Alexandria Engineering Journal*, vol. 57, no. 4, 2018, pp. 4177–4187]. // *Alexandria Engineering Journal*, vol. 61, no. 10, p. 7939. https://doi.org/10.1016/j.aej.2018.11.006.

[2] **Ahmed, S. et al.** 2021. "Sensitivity Analysis of Linear Programming in Decision Making Model." // *International Journal of Theoretical and Applied Mathematics*, vol. 7, no. 3: pp. 53–56. https://doi.org/10.11648/j.ijtam.20210703.12.

[3] **Albers, D. J., and Reid, C.** 1986. "An Interview with George B. Dantzig: The Father of Linear Programming." // *The College Mathematics Journal*, vol. 17, no. 4, pp. 292–314.
https://doi.org/10.1080/07468342.1986.1197297.

[4] **Ammar, E. E., and Emsimir, A. A.** 2020. A Mathematical Model for Solving Integer Linear Programming Problems. // *African Journal of Mathematics and Computer Science Research*, vol. 13, no. 1, pp. 39–50. https://doi.org/10.5897/AJMCSR2019.0804.

[5] **Dantzig, G. B.** 1982. "Reminiscences about the Origins of Linear Programming." // *Operations Research Letters*, vol. 1, no. 2, pp. 43–48. https://doi.org/10.1016/0167-6377(82)90043-8.

[6] **Dantzig, G. B.** 1987. "Origins of the Simplex Method." – In: Nash, S. G. (ed.). *A History of Scientific Computing*, pp. 141–151. https://doi.org/10.1145/87252.88081.

[7] **Nilu, T. Y., Ahmed, H., and Ahmed, S.** 2019. "Mathematical Modeling for Launch Vessel Operators at Internal Waterways Transportation in Bangladesh." // *GUB Journal of Science and Engineering*, vol. 6, no. 1, pp. 46–53. https://doi.org/10.3329/gubjse.v6i1.52050.

[8] **Stanimirović, I.** *Advances in Optimization and Linear Programming* (1st ed.). New York: Apple Academic Press, 27 Jan. 2022. https://doi.org/10.1201/9781003256052.

ABOUT THE AUTHORS

Endrit Rushiti – student in Informatics, Faculty of Computer Science, Mother Teresa University, Skopje, North Macedonia, e-mail: endrit.rushiti@students.unt.edu.mk

Axhi Beqiri – lecturer in Informatics, Faculty of Computer Science, Mother Teresa University, Skopje, North Macedonia, e-mail: axhi.beqiri@unt.edu.mk

Egzona Iseni – professor of Mathematics, Faculty of Computer Science, Mother Teresa University, Skopje, North Macedonia, e-mail: egzona.iseni@unt.edu.mk

Shpetim Rexhepi – professor of Mathematics, Faculty of Civil Engineering and Architecture, Mother Teresa University, Skopje, North Macedonia, e-mail: shpetim.rexhepi@unt.edu.mk