24MAC175 Group 1 Coursework Report

Matthew Lakin, James Death, Nathan Moore, Saad Tahir

November - December 2024

Contents

1 Algorithm Explanation

The question we recieved to solve isn't able to be solved using the Simplex Method as it is. The Simplex Method is used to solve linear programming problems in the canonical form. So the first step is to clean the question for the Simplex method.

1.1 Converting to Canonical Form

The question we were asked to solve was as follows:

Minimise $z = 7x_1 + 11x_3 - 10x_4 - x_5 + 26x_6$, subject to

$$x_1 - x_2 + x_3 + x_5 + x_6 = 76 \tag{1.1}$$
$$x_2 - x_3 + x_4 + 3x_6 \leq 18 \tag{1.2}$$
$$x_1 + x_2 - 3x_3 + x_4 + x_5 \leq 12 \tag{1.3}$$
$$x_1 + x_2 + x_6 \geq 50 \tag{1.4}$$

with all variables non-negative: $x_1, x_2, x_3, x_4, x_5, x_6 \geq 0$.

The first step in solving this is to put it into canonical form. Canonical form is where the question is in the form:

$$\text{Minimise } z = \underline{c}^T \cdot \underline{x} \tag{1.5}$$
$$A\underline{x} = \underline{b} \tag{1.6}$$
$$\underline{x} \geq 0 \tag{1.7}$$

For this, we add slack and surplus variables to (1.1), (1.2), and (1.3) to remove the inequality in favour of an equality:

Minimise $z = 7x_1 + 11x_3 - 10x_4 - x_5 + 26x_6$, subject to

$$x_1 - x_2 + x_3 + x_5 + x_6 = 76 \tag{1.8}$$
$$x_2 - x_3 + x_4 + 3x_6 + S_1 = 18 \tag{1.9}$$
$$x_1 + x_2 - 3x_3 + x_4 + x_5 + S_2 = 12 \tag{1.10}$$
$$x_1 + x_2 + x_6 - S_3 = 50 \tag{1.11}$$

with all variables non-negative: $x_1, x_2, x_3, x_4, x_5, x_6, S_1, S_2, S_3 \geq 0$.

However (1.11) violates the non-negativity constraint, so to fix that we need to introduce artificial variables to the LPP. In order to make a basic feasible solution we need to add artificial variables to both (1.8) and (1.11). We will also amend the objective function with a large M value. This will be used in calculations so the artificial variables tend to 0 as iterations progress. So now the problem becomes:

Minimise $z = 7x_1 + 11x_3 - 10x_4 - x_5 + 26x_6 + M(R_1 + R_2)$, subject to

$$x_1 - x_2 + x_3 + x_5 + x_6 + R_1 = 76 \tag{1.12}$$
$$x_2 - x_3 + x_4 + 3x_6 + S_1 = 18 \tag{1.13}$$
$$x_1 + x_2 - 3x_3 + x_4 + x_5 + S_2 = 12 \tag{1.14}$$
$$x_1 + x_2 + x_6 - S_3 + R_2 = 50 \tag{1.15}$$

with all variables non-negative: $x_1, x_2, x_3, x_4, x_5, x_6, S_1, S_2, S_3, R_1, R_2 \geq 0$.

The LPP is now in canonical form and is ready for simplex.

1.2 Revised Simplex Algorithm

The Revised Simplex Algorithm is a variant of the Simplex Algorithm that is used to solve linear programming problems. It uses inverse matrices to simplifiy the calculations for computers.

The question consists of m constraints and n variables. The matrix A is $m \times n$ and is defined and defined as follows:

$$A = \begin{bmatrix} | & | & & | \\ \underline{P_1} & \underline{P_2} & \cdots & \underline{P_n} \\ | & | & & | \end{bmatrix} \tag{1.16}$$

The matrix A is partitioned into two matrices, B and N, where B is a $m \times m$ matrix and N is a $m \times (n - m)$ matrix. The matrix B is the basis matrix and the matrix N is the non-basis matrix. The matrix B is initially the identity matrix due to choosing slack and artificial variables. The matrix N is the matrix of non-basic variables. The matrix A is then defined as:

$$A = \begin{bmatrix} B & N \end{bmatrix} \tag{1.17}$$

We then see if the current solution is valid. Since we chose the basis using the slack and artificial variables, we know that it is valid. The amount of basic solutions is equal to $\binom{n}{m}$. This grows exponentially so it is impractical to enumerate all possible solutions.

1.2.1 Optimality Test

The optimality test is used to determine if the current solution is optimal. The optimality test is defined as:

$$\underline{c_N} - \underline{c_B^T} B^{-1} N \tag{1.18}$$

where c_B is the cost vector of the basic variables, c_N is the cost vector of the non-basic variables, and B^{-1} is the inverse of the basis matrix. There are three possibilities for the optimality test:

- If all values in the optimality test positive, then any x_B will increase z so we're at the minimum.
- If all values are positive or zero, then the current solution is a non unique optima.
- If there is a negative value, then the current solution is not optimal. In this case, continue the algorithm.

We choose the most negative value as the entering variable. This is the variable that will enter the basis.

1.2.2 Feasibility Test

As we have found an entering variable, we find the associated column of N and call it a_E. We then find the ratios of the right hand side to the column of the entering variable. If all values are negative, then the problem is unbounded. If there is a positive value, then we choose the smallest value as the corrensponding leaving variable. This is the variable that will leave the basis.

Finally, we update the basis with the entering column from N, remove the leaving column from B and update the coefficients of the objective function. We then repeat until we reach the optimal solution.

2 Worked Example

On paper, we solved a slightly simpler version of the question we were given. We did this to verify the Simplex Method. The solution is below:

Minimise $Z = -6x_1 + 7x_2 + 4x_3$

Subject to:

$$
\begin{aligned}
2x_1 + 5x_2 - x_3 &\leq 18 \quad - \text{I}\\
x_1 - x_2 - 2x_3 &\leq -14 \quad - \text{II}\\
3x_1 + 2x_2 + 2x_3 &= 26 \quad - \text{III}\\
x_1, x_2, x_3 &\geq 0
\end{aligned}
$$

Above function is equivalent to maximising:

$$Z' = 6x_1 - 7x_2 - 4x_3$$

The constraints will remain the same, but we can multiply II by -1:

$$
\begin{aligned}
2x_1 + 5x_2 - x_3 &\leq 18 \quad - \text{I}\\
-x_1 + x_2 + 2x_3 &\geq 14 \quad - \text{II}\\
3x_1 + 2x_2 + 2x_3 &= 26 \quad - \text{III}\\
x_1, x_2, x_3 &\geq 0
\end{aligned}
$$

Standard Form

We introduce the artificial / slack variables to remove inequalities:

$$
\begin{aligned}
\text{I} &\rightarrow 2x_1 + 5x_2 - x_3 + s_1 = 18\\
\text{II} &\rightarrow -x_1 + x_2 + 2x_3 - s_2 + a_2 = 14\\
\text{III} &\rightarrow 3x_1 + 2x_2 + 2x_3 + a_3 = 26
\end{aligned}
$$

where: $x_1, x_2, x_3, s_1, s_2, a_2, a_3 \geq 0$

Next, we add a large artificial variable penalty M for a_2 and a_3, where M is a large positive number:

$$Z' = 6x_1 - 7x_2 - 4x_3 - Ma_2 - Ma_3$$

Initial Simplex Tableau

Pivot column due to 4M (largest positive value)

Basis	C_B	x_1 6	x_2 -7	x_3 -4	s_1 0	s_2 0	a_2 -M	a_3 -M	b	Ratio
s_1	0	2	5	-1	1	0	0	0	18	18/-1 (ignore since negative)
Pivot Row (smallest ratio) a_2	-M	-1	1	2	0	-1	1	0	14	14/2 = 7
a_3	-M	3	2	2	0	0	0	1	26	26/2 = 13
	z_j	-2M	-3M	-4M	0	M	-M	-M	-40M	
	$c_j - z_j$	6+2M	-7+3M	-4+4M	0	-M	0	0		

Since a_2 is an artificial variable and is the pivot row, we will remove it from the columns and replace it in the row with x_3

Iteration 1

Row 2_{old} / 2 = Row 2_{New}
Row 1_{old} + Row 2_{New} = Row 1_{New}
Row 3_{old} - Row 2_{old} = Row 3_{New}

Pivot column

Basis	C_B	x_1 6	x_2 -7	x_3 -4	s_1 0	s_2 0	a_3 -M	b	Ratio
s_1	0	1.5	5.5	0	1	-0.5	0	25	25/1.5 = 50/3
x_3	-4	-0.5	0.5	1	0	-0.5	0	7	(Ignore since negative)
a_3	-M	4 (Pivot element)	1	0	0	1	1	12	12/4 = 3
	Z_j	2-4M	2-M	-4	0	2-M	-M	-28-12M	
	$C_j - Z_j$	4+4M	-5+M	0	0	-2+M	0		

Since a_3 is an artificial variable and the pivot row, we will remove it from the columns and replace it in the row with x_1

Iteration 2

$Row\ 3_{old}/4 = Row\ 3_{New}$
$Row\ 1_{old} - \frac{3}{2} Row\ 3_{New} = Row\ 1_{New}$
$Row\ 2_{old} + \frac{1}{2} Row\ 3_{New} = Row\ 2_{New}$

Basis	C_B	x_1 6	x_2 -7	x_3 -4	s_1 0	s_2 0	b	Ratio
s_1	0	0	5.125	0	1	-0.875	20.5	
x_3	-4	0	0.625	1	0	-0.375	8.5	N/A
x_1	6	1	0.25	0	0	0.25	3	
	Z_j	6	-1	-4	0	3	-16	
$C_j - Z_j$		0	-6	0	0	-3	≤ 0	

Since no positive value in the net evaluation row, objective function cannot be optimised further.

Solution

$x_1 = 3$
$x_2 = 0$
$x_3 = 8.5$
$s_1 = 20.5$
$s_2 = 0$
$Z = 16 \implies$ Since we are ultimately looking to minimise Z, we can multiply the objective function by -1

3 Implementation

For this project, we used Python for access to the NumPy module and MatPlotLib for graphing. The code was written in a modular fashion, with each function performing a specific task.

3.1 Representation of the Problem

Before doing any simplex calculations on the question, we had to parse the question into a form that python could understand. For our representation, we used 5 data structures:

- Nature: This is a binary value that represents whether the problem is a maximisation or minimisation problem. If the problem is a maximisation problem, the value is -1, otherwise it is 1.
- A: This is a matrix that represents the coefficients of the constraints. Each row represents a constraint, and each column represents a variable.
- b: This is a vector that represents the right-hand side of the constraints. Each row represents a constraint.
- c: This is a vector that represents the coefficients of the objective function. Each column represents a variable.
- Signs: This is a vector that represents the signs of the constraints. Each row represents a constraint. The value -1 represents a less than or equal to constraint, 0 represents an equal to constraint, and 1 represents a greater than or equal to constraint.

As we have code to convert the LPP (Linear Programming Problem) to canonical form, we can accept any LPP in any form.

3.2 Main Method

The main method is the entry point for the program. It is where the question is inputted and the functions are called. When using this project, this is the only method that needs to be interacted with.

3.3 convertToCanonicalForm

The function convertToCanonicalForm in file bfs.py is used to convert the LPP to canonical form. The function takes the LPP in our parsing style as input and returns the LPP in canonical form. It also introduces artificial variables to the LPP to make a basic feasible solution along with amending the objective function with a large M value if required.
The function requires a valid LPP to work. We added a check to see if the LPP is valid before running the function.
It takes each constraint and does 1 of 3 things:

- If the constraint is an equality constraint, it adds an artificial variable to the constraint.
- If the constraint is a less than or equal to constraint, it adds a slack variable to the constraint.
- If the constraint is a greater than or equal to constraint, it adds a surplus and an artificial variable to the constraint.

3.4 Simplex Method

The function revisedSimplexMethod in file simplex.py is used to solve the LPP in canonical form. The function takes the LPP in canonical form as input and returns the optimal solution.
The function uses the exact same algorithm as described in the Algorithm Explanation section. It uses the optimality test to determine if the current solution is optimal. If it is not, it uses the feasibility test to find the entering and leaving variables. It then updates the basis and repeats until the solution is optimal.
It also checks for infeasibility and unboundedness. If the LPP is infeasible, it returns "Solution is Infeasible". If the LPP is unbounded, it returns "Unbounded solution".

3.5 Parsing the LPP

The function renderLPP in file render.py is used to render the LPP. It supports LPP, both canonical and non-canonical form. The function takes the LPP in our parsing style as input and returns a string representation of the LPP.

3.6 Graphing

The function graph in file variableGraphing.py is used to graph the variables. To change which constraint is being varied, you need to change the variable "bVaryNum". Then the function will make a graph of all feasible solutions which exist in the interval [1, 200].
If the interval of feasible solutions is smaller than 200, the interval will be reduced to the size of the feasible interval.

3.7 Using the Code

To use the code to answer the provided question, you must verify that variables in main.py are set correctly. The variables are as follows:

```
nature = 1
c = nature * np.array([7, 0, 11, -10, -1, 26])
A = np.array([[1, -1, 1, 0, 1, 1], [0, 1, -1, 1, 0, 3], [1, 1, -3, 1, 1, 0], [1, 1,
    0, 0, 0, 1]])
b = np.array([76, 18, 12, 50])
signs = np.array([0, -1, -1, 1])
```

Once you run main.py, you should see the multiple iterations with each entering and leaving variable and at the end the optimal solution. If the solution is infeasible or unbounded, it will print that instead:

```
Optimal Value:  200
Optimal Solution:  [49.2  0.8 27.6 44.8  0.   0.   0.   0.   0.   0.   0. ]
Status:  Optimal solution found
```

4 Special Cases

5 Verification

For verification, we entered the question:
Minimise $z = 7x_1 + 11x_3 - 10x_4 - x_5 + 26x_6$, subject to

$$x_1 - x_2 + x_3 + x_5 + x_6 = 76 \tag{5.1}$$
$$x_2 - x_3 + x_4 + 3x_6 \leq 18 \tag{5.2}$$
$$x_1 + x_2 - 3x_3 + x_4 + x_5 \leq 12 \tag{5.3}$$
$$x_1 + x_2 + x_6 \geq 50 \tag{5.4}$$

with all variables non-negative: $x_1, x_2, x_3, x_4, x_5, x_6 \geq 0$.
We then used the Python package SciPy to solve the question. The representation of the question in SciPy is as follows:

```
import numpy as np
from scipy.optimize import linprog

c = np.array([7, 0, 11, -10,-1,26])
Aup = np.array([[0, 1, -1, 1, 0, 3],
                [1, 1, -3, 1, 1, 0],
                [-1, -1, 0, 0, 0, -1]])
bup = np.array([18, 12, -50])
Aeq = np.array([[1, -1, 1, 0, 1, 1]])
beq = np.array([76])
```

Since SciPy's method doesn't support the greater than or equal to constraints, we had to alter (5.4) by multiplying both sides of the equation by -1 and flipping the inequality.

$$-x_1 - x_2 - x_6 \leq -50 \tag{5.5}$$

We then used the SciPy method linprog to solve the question. The results were as follows:

```
res = linprog(c,Aup,bup,Aeq,beq)
print(res)
```

```
message: Optimization terminated successfully. (HiGHS Status 7: Optimal)
success: True
 status: 0
    fun: 200.0
      x: [ 4.920e+01  8.000e-01  2.760e+01  4.480e+01  0.000e+00
           0.000e+00]
    nit: 5
  lower:  residual: [ 4.920e+01  8.000e-01  2.760e+01  4.480e+01
                      0.000e+00  0.000e+00]
         marginals: [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00
                      1.000e+00  4.500e+01]
  upper:  residual: [       inf        inf        inf        inf
                            inf        inf]
         marginals: [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00
                      0.000e+00  0.000e+00]
  eqlin:  residual: [ 0.000e+00]
         marginals: [-1.000e+00]
ineqlin:  residual: [ 0.000e+00  0.000e+00  0.000e+00]
         marginals: [-9.000e+00 -1.000e+00 -9.000e+00]
mip_node_count: 0
mip_dual_bound: 0.0
mip_gap: 0.0
```

The solution of the question corresponds with the "fun" value in the dictionary output. The value of the objective function is 200.0, which matches the value we found using the Revised Simplex Method. It also shows it took 5 iterations in the "nit" value which also aligns itself with our calculations. This verifies that our implementation is correct for this question.

6 Sensitivity Analysis

6.1 Behaviour of the Objective Function

6.2 Constraint 1

6.3 Constraint 2

6.4 Constraint 3

6.5 Constraint 4

7 Conclusion