Project 2

Qianlang Chen, Kevin Song, and Jamie Kelemen

April 2020

1 Linear Programming Problem

We have chosen **Linear Programming** as our project to do.

First of all, we would like to introduce the concept of **Systems of Linear Inequalities** where we are trying to find a solution of the inequality. And the problem of inequalities and constraints is called **Linear Programming Problem**. For example,

$$3x - 2y < 6$$

$$x + y \geq 6$$

And $(1,1)$ is a solution to as it qualifies both inequalities with two variables x and y. And the solution must occur within the set of feasible solutions. In real world, there are what is called **optimization** where we are trying to find the maximum or minimum within the feasible solutions.

2 Simplex Method

Simplex method is a systemic procedure in linear programming to solve a optimization problem. This method is particularly convenient to solve problems involving more than two variables or problems with a large number of constraints (Larson, 2016). Given the complexity of the real world problems, simplex method is a convenient way to determine the optimal value within the feasible region.

The core idea behind the simplex method is to perform elementary row operations on a matrix called **simplex tableau** until we are able to find the **basic variables** and the **nonbasic variables**. Furthermore, we make sure that the entries are non-negative.

And then, we improve the current solution by **pivoting**: bring a new basic variable into the current solution called **entering variable** to replace one of the current basic variables called **departing variable**.

The rules of choosing the entering and departing variables are: 1. The **entering variable** is the smallest entry in the bottom row of the tableau. 2. The **departing variable** is the smallest non negative ratio of b_i/a_{ij} in the column determined by the entering variable.

3 Problem 1

1. Solve the following linear programming problem using the simplex method:

The objective of this linear programming problem is try to find the maximized value, which is to try to find the linear programming problem in standard form.

Maximize:

$$z = 20x_1 + 10x_2 + 15x_3 \tag{1}$$

$$3x_1 + 2x_2 + 5x_3 \leq 55 \tag{2}$$

$$2x_1 + x_2 + x_3 \leq 26 \tag{3}$$

$$\tag{4}$$

subject to

$$3x_1 + 2x_2 + 5x_3 \leq 55 \tag{5}$$

$$2x_1 + x_2 + x_3 \leq 26 \tag{6}$$

$$x_1 + x_2 + 3x_3 \leq 30 \tag{7}$$

$$5x_1 + 2x_2 + 4x_3 \leq 57 \tag{8}$$

$$x_1, x_2, x_3 \geq 0 \tag{9}$$

Below is the MatLab programming code to compute the solution for this linear programming problem using the simplex method.

Initialize the objective function (the function to minimize). Since the simplex method always minimizes the system, we take the opposite values for the coefficients for the algorithm, then we'll take the opposite of the result for the maximized value.

```matlab
z = [-20  -10  -15];

% Initiaize the less-than constraints.
A = [
  3  2  5;
  2  1  1;
  1  1  3;
  5  2  4;
];
b = [55  26  30  57];

% Initialize the equality constraints (none for this problem).
Aeq = [];
beq = [];

% Initialize variable bounds.
lb = [0  0  0];
ub = [];

% Run the simplex algorithm to solve the model.
[X, Z] = linprog(z, A, b, Aeq, beq, lb, ub);

% Take the opposite of the result to get the maximized value.
Z = -Z;

% Display the results.
disp("x1 = " + X(1));
disp("x2 = " + X(2));
disp("x3 = " + X(3));
disp("z  = " + Z);
```

And here are the generated solution:

```
x1 = 1.8
x2 = 20.8
x3 = 1.6
z  = 268
```

4 Problem 2

2. Seven patients require blood transfusions. We assume four blood types: A, AB, B, and O. Type AB is called the universal recipient; type O is called the universal donor. The blood supply and patient data is as follows:

Type	Supply	Cost per pint
A	7 pints	$1
AB	6 pints	$2
B	4 pints	$4
O	5 pints	$5

Patient	Blood Type	Need(pints)
1	A	2
2	AB	3
3	B	1
4	O	2
5	A	3
6	B	2
7	AB	1

Solution Explanation:

Again, we can solve problem 2 with linear programming using the simplex method similar to problem 1. Let $p1_a$ be the amount of type A blood patient 1 should receive, and $p2_{ab}$ be the amount of type AB blood patient 2 should receive, and so on. All of these variables must be positive.

Each patient can only get the blood types they're compatible with:

Patient	Original Type	Compatible Types	Corresponding Variables
1	A	A,O	p1$_a, p1_o$
2	AB	A,AB,B,O	p2$_a, p2_{ab}, p2_b, p2_o$
3	B	B,O	p3$_b, p3_o$
4	O	O	p4$_o$
5	A	A,O	p5$_a, p5_o$
6	B	B,O	p6$_b, p6_o$
7	AB	A,AB,B,O	p7$_a, p7_{ab}, p7_b, p7_o$

The total cost we're minimizing is the sum of the amount of each type of blood each patient gets, multiplied by its unit price:

$$cost = 1 \times p1_a + 1 \times p2_a + 1 \times p5_a + 1 \times p7_a + 2 \times p2_{ab} + 2 \times p7_{ab} + 4 \times p2_b$$

$$+4 \times p3_b + 4 \times p6_b + 4 \times p7_b + 5 \times p1_o + 5 \times p2_o + 5 \times p3_o + 5 \times p4_o + 5 \times p5_o$$

$$+5 \times p6_o + 5 \times p7_o$$

Each patient must get the exact amount they need (subjective to):

$$p1_a + p1_o = 2 \tag{1}$$

$$p2_a + p2_{ab} + p2_b + p2_o = 3 \tag{2}$$

$$p3_b + p3_o = 1 \tag{3}$$

$$p4_o = 2 \tag{4}$$

$$p5_a + p5_o = 3 \tag{5}$$

$$p6_b + p6_o = 2 \tag{6}$$

$$p7_a + p7_{ab} + p7_b + p7_o = 1 \tag{7}$$

Finally, the use of each blood type cannot exceed its supply:

$$p1_a + p2_a + p5_a + p7_a \leq 7 \tag{1}$$

$$p2_{ab} + p7_{ab} \leq 6 \tag{2}$$

$$p2_b + p3_b + p6_b + p7_b \leq 4 \tag{3}$$

$$p1_o + p2_o + p3_o + p4_o + p5_o + p6_o + p7_o \leq 5 \tag{4}$$

Math Model:

$$Minimize\,Cost = 1 \times p1_a + 1 \times p2_a + 1 \times p5_a + 1 \times p7_a + 2 \tag{1}$$

$$\times p2_{ab} + 2 \times p7_{ab} + 4 \times p2_b + 4 \times p3_b + 4 \tag{2}$$

$$\times p6_b + 4 \times p7_b + 5 \times p1_o + 5 \times p2_o \tag{3}$$

$$+5 \times p3o + 5 \times p4_o + 5 \times p5_o + 5 \times p6_o + 5 \times p7_o \tag{4}$$

Subjective to

$$
\begin{aligned}
p1_a + p2_a + p5_a + p7_a &\leq 7 & p7_a &\geq 0 \\
p2_{ab} + p7_{ab} &\leq 6 & p2_{ab} &\geq 0 \\
p2_b + p3_b + p6_b + p7_b &\leq 4 & p7_{ab} &\geq 0 \\
p1_o + p2_o + p3_o + p4_o + p5_o + p6_o + p7_o &\leq 5 & p2_b &\geq 0 \\
p1_a + p1_o &= 2 & p3_b &\geq 0 \\
p2_a + p2_{ab} + p2_b + p2_o &= 3 & p6_b &\geq 0 \\
p3_b + p3_o &= 1 & p7_b &\geq 0 \\
p4_o &= 2 & p1_o &\geq 0 \\
p5_a + p5_o &= 3 & p2_o &\geq 0 \\
p6_b + p6_o &= 2 & p3_o &\geq 0 \\
p7_a + p7_{ab} + p7_b + p7_o &= 1 & p4_o &\geq 0 \\
p1_a &\geq 0 & p5_o &\geq 0 \\
p2_a &\geq 0 & p6_o &\geq 0 \\
p5_a &\geq 0 & p7_o &\geq 0
\end{aligned}
$$

Below is the MatLab programming code to compute the solution for this linear programming problem using the simplex method.

```
% Initialize the objective function (the function to minimize).
cost = [1  1  1  1  2  2  4  4  4  4  5  5  5  5  5  5  5];

% Initialize the less-than constraints (none for this problem).

% 1  2  5  7  2  7  2  3  6  7  1  2  3  4  5  6  7 <- patient
% a  a  a  a  ab ab b  b  b  b  o  o  o  o  o  o  o <- type
A = [
  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0;
  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0;
  0  0  0  0  0  0  1  1  1  1  0  0  0  0  0  0  0;
  0  0  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1;
];
b = [7  6  4  5];

% Initialize the equality constraints.
Aeq =[
  1  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0;
  0  1  0  0  1  0  1  0  0  0  0  1  0  0  0  0  0;
  0  0  0  0  0  0  0  1  0  0  0  0  1  0  0  0  0;
  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0;
  0  0  1  0  0  0  0  0  0  0  0  0  0  0  1  0  0;
  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  1  0;
  0  0  0  1  0  1  0  0  0  1  0  0  0  0  0  0  1;
];
beq = [2  3  1  2  3  2  1];

% Initialize variable bounds.
lb = [0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0];
ub = [];

% Run the simplex algorithm to solve the model.
[Pints, Cost] = linprog(cost, A, b, Aeq, beq, lb, ub);

```

```
% Display the results.
disp("Patient 1 gets type A:  " + Pints(1));
disp("Patient 2 gets type A:  " + Pints(2));
disp("Patient 5 gets type A:  " + Pints(3));
disp("Patient 7 gets type A:  " + Pints(4));
disp("Patient 2 gets type AB: " + Pints(5));
disp("Patient 7 gets type AB: " + Pints(6));
disp("Patient 2 gets type B:  " + Pints(7));
disp("Patient 3 gets type B:  " + Pints(8));
disp("Patient 6 gets type B:  " + Pints(9));
disp("Patient 7 gets type B:  " + Pints(10));
disp("Patient 1 gets type O:  " + Pints(11));
disp("Patient 2 gets type O:  " + Pints(12));
disp("Patient 3 gets type O:  " + Pints(13));
disp("Patient 4 gets type O:  " + Pints(14));
disp("Patient 5 gets type O:  " + Pints(15));
disp("Patient 6 gets type O:  " + Pints(16));
disp("Patient 7 gets type O:  " + Pints(17));
disp("Total cost:             " + Cost);
```

And here are the displayed results:

```
Patient 1 gets type A:  2
Patient 2 gets type A:  1
Patient 5 gets type A:  3
Patient 7 gets type A:  1
Patient 2 gets type AB: 2
Patient 7 gets type AB: 0
Patient 2 gets type B:  0
Patient 3 gets type B:  1
Patient 6 gets type B:  2
Patient 7 gets type B:  0
Patient 1 gets type O:  0
Patient 2 gets type O:  0
Patient 3 gets type O:  0
Patient 4 gets type O:  2
Patient 5 gets type O:  0
Patient 6 gets type O:  0
Patient 7 gets type O:  0
Total cost:             33
```

5 Problem 3

Create and solve one linear programming problem on your own.

Problem text: A company makes two products, X and Y, using two machines, A and B. Producing each unit of product X requires 15 minutes processing time on machine A followed by 10 minutes on machine B. Producing each unit of Y requires 5 minutes on machine A followed by 10 minutes on machine B. The available processing time is 72 hours on machine A and 60 hours on machine B.

There are 24 units of X and 84 units of Y currently in stock. The demand for the coming week is forecast to be 96 units of product X and 120 units of product Y. The company would like to maximize the combined total of the units X and Y in stock at the end of the week. Using a linear program, find out how much of each product the company should make for the current week.

Solution: Let x be the number of X units to produce for the week and y be the number of Y units to produce for the week.

The units of products that'd remain in stock at the end of the week would be the units currently in stock, plus the units to produce, minus the units to be sold:

$$total = (24 + x - 96) + (84 + y - 120) = x + y - 108 \tag{1}$$

$$\tag{2}$$

The company must have enough product made for the demands:

$$x \geq 96 - 24 \longrightarrow x \geq 72y \geq 120 - 84 \longrightarrow y \geq 36 \tag{1}$$

The total time used by the machines cannot exceed their available times:

$$15 \times x + 10 \times y \leq 72 \times 60 \longrightarrow 15 \times x + 10 \times y \leq 4,320 \tag{1}$$

$$5 \times x + 10 \times y \leq 60 \times 60 \longrightarrow 5 \times x + 10 \times y \leq 3,600 \tag{2}$$

Math Model:

Maximize

$$total = x + y - 108 \tag{1}$$

Subjective to

$$15 \times x + 10 \times y \leq 4320 \tag{1}$$

$$5 \times x + 10 \times y \leq 3600 \tag{2}$$

$$x \geq 72 \tag{3}$$

$$y \geq 36 \tag{4}$$

Below is the MatLab programming code to compute the solution for this linear programming problem using the simplex method.

```
% Initialize the objective function (the function to minimize).
%
% Since the simplex method always minimizes the system, we take the
% opposite values for the coefficients for the algorithm, then we'll take
% the opposite of the result for the maximized value.
total = [-1  -1];

% Initialize the integer constraint so we only produce integer units of
% either product.
intcon = [1  2];

% Initialize the less-than constraints.
A = [
  15  10;
  5  10;
];
b = [4320  3600];

% Initialize the equality constraints (none for this problem).
Aeq = [];
beq = [];

% Initialize variable bounds.
lb = [72  36];
ub = [];

% Run the simplex algorithm to solve the model.
[Prod, Total] = intlinprog(total, intcon, A, b, Aeq, beq, lb, ub);

% Take the opposite of the result to get the maximized value and subtract back
    50
% like in the original maximizing equation.
Total = -Total - 108;

```

```
% Display the results.
disp("Units of product X to produce: " + Prod(1));
disp("Units of product Y to produce: " + Prod(2));
disp("Units remain in stock after the week: " + Total);
```

And here are the displayed results:

```
Units of product X to produce: 72
Units of product Y to produce: 324
Units remain in stock after the week: 288
```

References

[1] Stephen P. Bradley and Arnoldo C. Hax and Thomas L. Magnanti. *Applied Mathematical Programming, Chapter 2* . URL:`http://web.mit.edu/15.053/www/AMP-Chapter-02.pdf` Addison-Wesley Publishing Company, 1977

[2] Ron Larson and Bruce H. Edwards *Elementary Linear Algebra*. URL:`https://college.cengage.com/mathematics/larson/elementary_linear/4e/shared/downloads/c09s3.pdf` Cengage Learning, 2016

[3] J E Beasley *OR-Notes*. URL:`http://people.brunel.ac.uk/~mastjjb/jeb/or/morelp.html`