Branch and Bound Implementation on Solving Integer Linear Programming Problems

Renaldi Arlin 13519114
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519114@std.stei.itb.ac.id

***Abstract*—Linear Programming or Linear Optimization problem is one of the prominent mathematical problems. The problem objective is to find an optimal solution of a linear function subject to a set of inequalities. This problem could be divided into two variations by the solution type value: Real and Integer. While the Real Linear Programming problem solution may be seen many times in academic studies, the latter variation may not. Integer Linear Programming Problem could be solved by using Branch and Bound algorithm strategy, a strategy that solves a problem by creating a state space tree and eliminating nodes every time it is not subject to the boundary function.**

Keywords—Branch and Bound, Integer, Linear Programming, Linear Optimization, Real

I. Introduction

Linear Programming problems are one of the most common mathematical problems in academic studies. The problem objective is to find the optimal (maximum or minimum value) solution of a linear function of several variables restricted by conditions that the variables are non-negative and satisfy a set of linear inequalities. Linear Programming could be applied to real life situations such as market calculations to get the maximum profit.

Linear programming or linear optimization was found in 1939 by a Soviet mathematician and economist, Leonid Kantrovich. It is a method he found, in World War II, to plan the cash outflow and inflow in order to reduce costs of the army and increase losses of the enemy's fund.

Branch and Bound is an algorithm strategy that solves a problem by creating a state space tree and inserting nodes following the rule of Breadth First Search with the boundary function to remove a node. It inserts a node until there is no more that can be created when this happens the remaining node is the solution.

Branch and Bound was first proposed by Alisa Land and Alison Doig during the research at the London School of Economics in 1960 for discrete programming. The name of the strategy itself first appeared in the work of Little *et al*, on the traveling salesman problem.

II. Basic Theory

For better understanding of this paper, it is important to review the basic theory of both linear programming and branch and bound algorithm strategy.

A. Linear Programming Problem

In this subchapter, there will be explanations regarding the definition, variations, examples. and generic solution.

1) Definition

Linear Programming or Linear Optimization problem is a problem that has an objective to obtain the optimal solution of a linear function (objective function) of several variables subject to constraints that the variables are non-negative and satisfy a set of linear inequalities.

2) Variations

a) Real Number Linear Programming Problem

Just as the name supposed, this linear programming problem has a solution for each variable as a real number. (e.g. $x_1 = 3.5$; $x_2 = 4$, etc.)

b) Integer Linear Programming Problem

This problem variation requires a solution such that all variables are an integer. (e.g. $x_1 = 3$; $x_2 = 4$, etc.). This variation could not be solved by using a generic solution explained later.

3) Examples

In this chapter, it is going to be explained as an example of a linear programming problem with a real number solution.

Let an equation be $5x_1 + 6x_2 = z$, *so that the value of z could be maximum. The solution must subject to the following inequalities:*

$$x_1 + x_2 \le 5$$

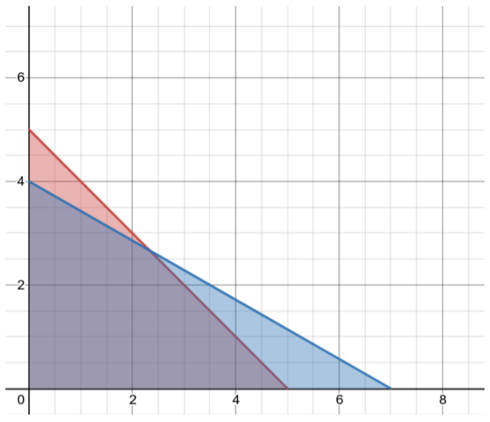
$$4x_1 + 7x_2 \le 28$$

$$x_1, x_2 \ge 0$$

This is one of the examples of a linear programming problem, the solution is to be explained in the following sub-subchapter.

4) Generic Solution

To solve the problem before, the generic solution is by using a graphical method. Visualize the area created by the inequalities given.



The x_1 is represented by x-axis and x_2 by y-axis.

Find the intersection of each line, from the visualization, it could be seen there are a total of 3 intersection points. We could find the intersections by seeing the cartesius diagram or by eliminating the inequalities.

intersection 1: (0, 5) by seeing the diagram or by setting x = 0

intersection 2: (7, 0) by seeing the diagram or by setting y = 0

intersection 3: (7/3, 8/3)

By using the elimination method,

$(x_1 + x_2 = 5) * -4 = \quad -4x_1 - 4x_2 = -20$

$4x_1 + 7x_2 = 28 \qquad 4x_1 + 7x_2 = 28 \ +$

$$3x_2 = 8$$

$$x_2 = 8/3$$

$$x_1 + 8/3 = 5$$

$$x_1 = 7/3$$

Insert all the possible optimal solutions to the objective function.

$$5 * (7/3) + 6 * (8/3) = 27.67$$

This is the maximum value of z, thus (7/3, 8/3) is the solution for this particular problem.

B. Branch and Bound

In this subchapter, there will be explanations regarding the definition, variations, and example of each variation.

1) Definition

Branch and Bound is algorithm strategy for discrete and combinatorial optimization problems such as mathematical optimization. This algorithm consists of a systematic list of candidate solutions named state space tree. Nodes of the tree are created following the rule of Breadth First Search in general case, if a node doesn't subject to the boundary function set, expansion of the node would be ceased and continued to another node until all of the candidate solutions are created. The remaining nodes in this phase are the possible solutions.

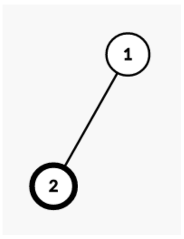
2) Variations

a) FIFO

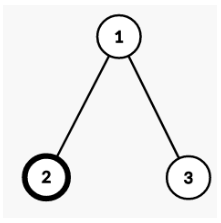
FIFO or First In First Out Branch and Bound is a variation that uses a Queue data structure for the next nodes that would be expanded. The process would terminate if the queue is already empty. Observe the following example:

Node expanded: 1
Adjacent Nodes of 1: {2, 3}
Queue: {2, 3}

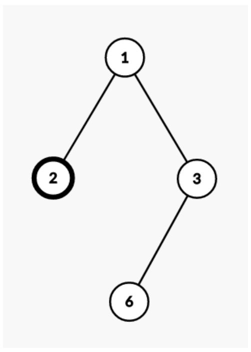
Node expanded: 2
Adjacent Nodes of 2: {4, 5}
Queue: {3}
Let the node 2 doesn't subject to the Boundary function



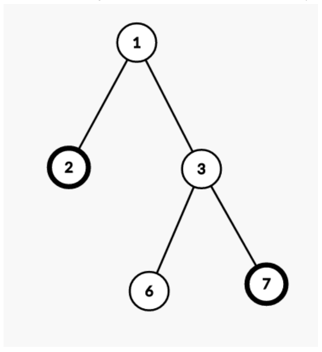
Node expanded: 3
Adjacent Nodes of 3: {6, 7}
Queue: {6, 7}



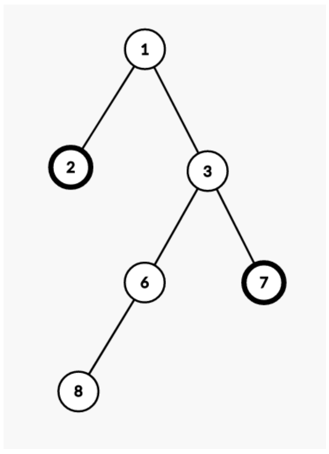
Node expanded: 6
Adjacent Nodes of 6: {8, 9}
Queue: {7, 8, 9}



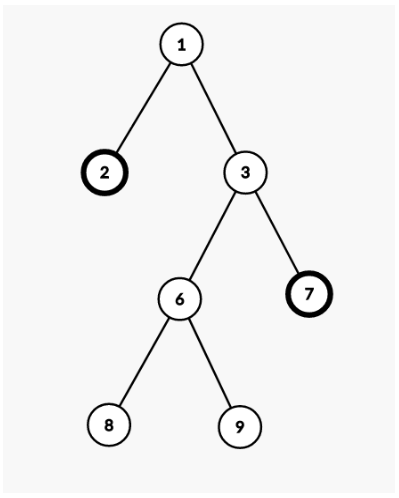
Node expanded: 7
Adjacent Nodes of 7: {10, 11}
Queue: {8, 9}
Let the node 7 doesn't subject to the Boundary function



Node expanded: 8
Adjacent Nodes of 8: {}
Queue: {9}



Node expanded: 9
Adjacent Nodes of 9: {}
Queue: {}



Therefore, the nodes 8 and 9 are the possible solutions or path 1-3-6-8 and 1-3-6-9 are the possible solutions.

b) LIFO

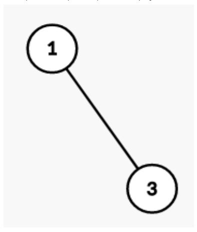
LIFO or Last In First Out Branch and Bound is a variation that uses a Stack data structure for the next nodes that would be expanded. The process would terminate if the stack is already empty. However, the pushing process of the stack still follows the rule of BFS, to put it simply the stack data structure only acts as a reversed queue. The example would be entirely the same as the FIFO example except the queue content would always be reversed.

c) Least Cost

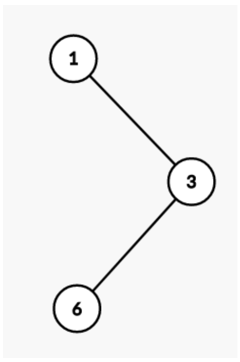
Least Cost Branch and Bound is a variation that creates a cost function for each node and takes the least cost node as the next expansion. This variation could be assumed to use Priority Queue as its data structure. The process would terminate if the Priority Queue is empty. Observe the following example.

Node expanded: 1
Cost of node 1: 5
Adjacent Nodes of 1: {2, 3}
Priority Queue: {(3, 1), (2, 3)}

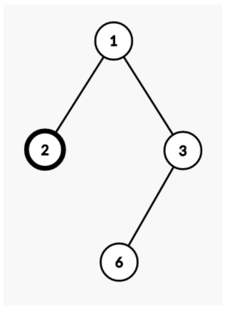
Node expanded: 3
Cost of node 3: 1
Adjacent Nodes of 3: {6, 7}
Priority Queue: {(6, 2), (2, 3), (7, 4)}



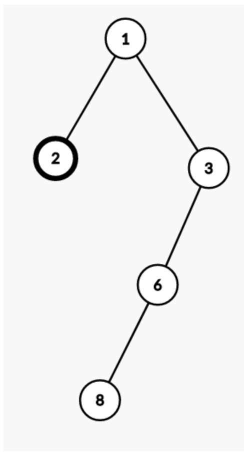
Node expanded: 6
Cost of node 6: 2
Adjacent Nodes of 6: {8, 9}
Priority Queue: {(2, 3), (8, 3), (7, 4), (9, 3)}



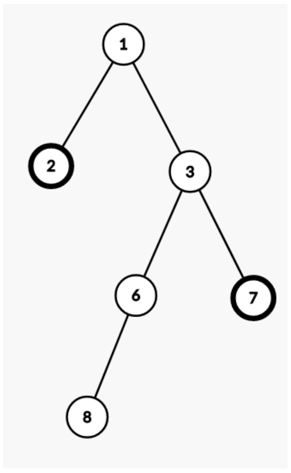
Node expanded: 2
Cost of node 2: 3
Adjacent Nodes of 2: {4, 5}
Priority Queue: {(8, 3), (7, 4), (9, 3)}
Let the node 2 doesn't subject to the Boundary function



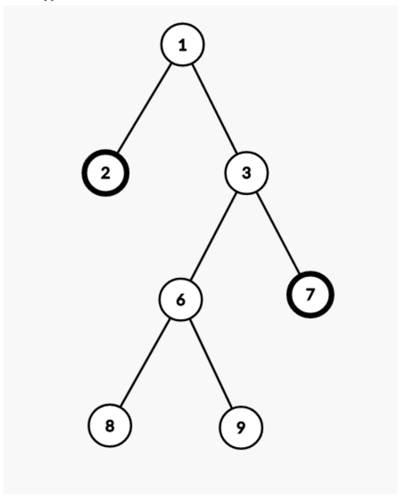
Node expanded: 8
Cost of node 8: 3
Adjacent Nodes of 8: {}
Priority Queue: {(7, 4), (9, 3)}



Node expanded: 7
Cost of node 7: 4
Adjacent Nodes of 7: {}
Priority Queue: {(9, 3)}
Let the node 7 doesn't subject to the Boundary function



Node expanded: 9
Cost of node 9: 3
Adjacent Nodes of 9: {}
Priority Queue: {}



III. IMPLEMENTATION

In this chapter, it is to be explained about the branch and bound implementation on solving integer linear programming problems. The problem used in this chapter is the same as the previous.

Let an objective equation be $5x_1 + 6x_2 = z$, *so that the value of z could be maximum. The solution must subject to the following inequalities:*

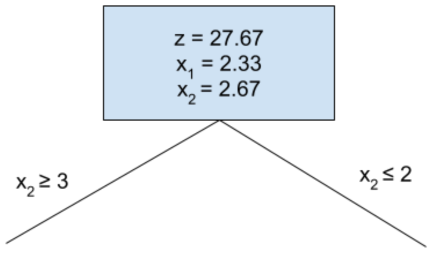
$$x_1 + x_2 \leq 5$$

$$4x_1 + 7x_2 \leq 28$$

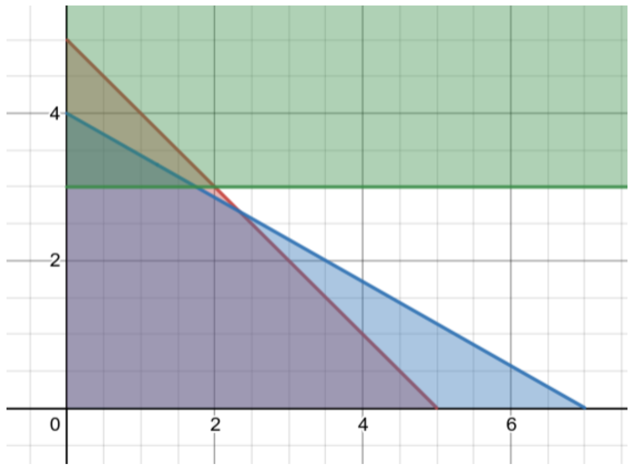
$$x_1, x_2 \geq 0$$

First step is to solve the problem itself using generic elimination or graphical methods. In the chapter II.A.4, the solution was $x_1 = 7/3$, $x_2 = 8/3$, $z = 27.67$. The solution is not an integer, therefore set this as the expansion node. The adjacent node is determined by creating a new inequality based on the highest decimal point in the variables, in this case it is 2.67. Set the inequalities to $x_2 \leq 2$ and $x_2 \geq 3$ (the nearest integer value to it) for each branch. The boundary function for this method is when the inequalites do not have any intersection area.

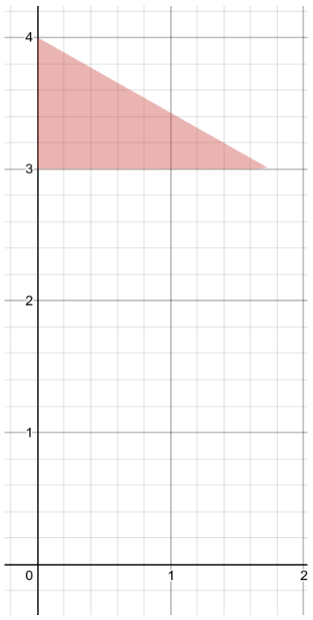
First Iteration:



Next, we follow the branch according to FIFO, so we insert the new node with an additional inequality of $x_2 \geq 3$. Solve the new problem using the elimination or graphical method.



Or see the shade that only shows the intersection for better understanding.



By seeing the graphic, we could see there are 3 intersections. (0, 3), (0, 4), and the last intersection could be find by using elimination method.

$x_2 = 3$

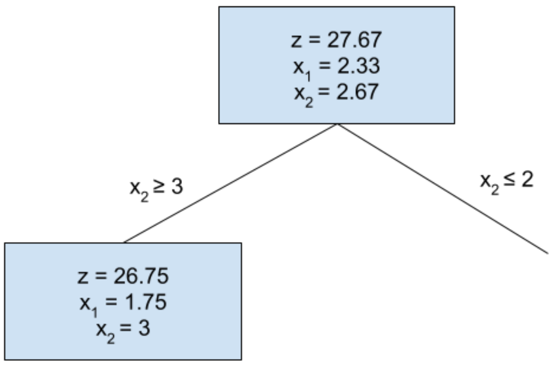
$4x_1 + 7x_2 = 28$

$4x_1 = 7$

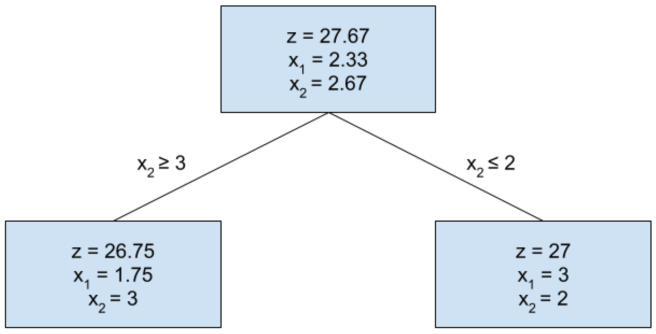
$x_1 = 1.75$

$z = 5 * 1.75 + 6 * 3 = 26.75$

This means the node is not yet a solution and needed to be expanded later on. Therefore the second iteration would look like:

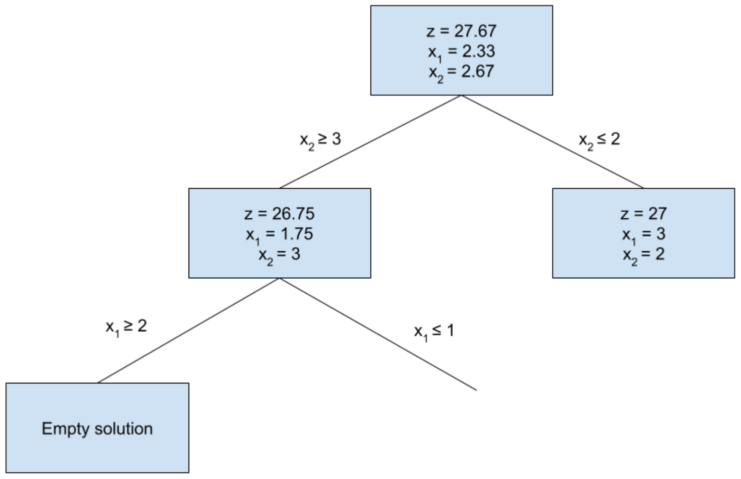


The third iteration involves the inequality $x_2 \leq 2$ and by doing the same method for getting the solution, the results are as following:

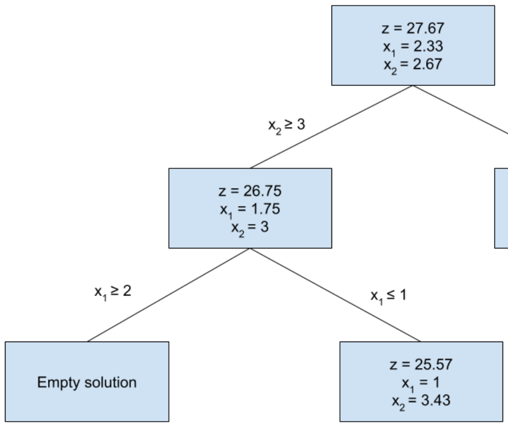


Because the third iteration results in an integer solution we set z = 27 as the lower bound and end the node branching.

The fourth iteration is branched from the ($x_1 = 1.75$, $x_2 = 3$, $z = 26.75$) because x_2 does not have any decimal point we take x_1 as the base number and set the branching inequality as $x_1 \geq 2$ and $x_1 \leq 1$. Then, solve the problem firstly adding the inequality $x_1 \geq 2$. However, this problem results in an empty solution therefore, it ends the node branching.

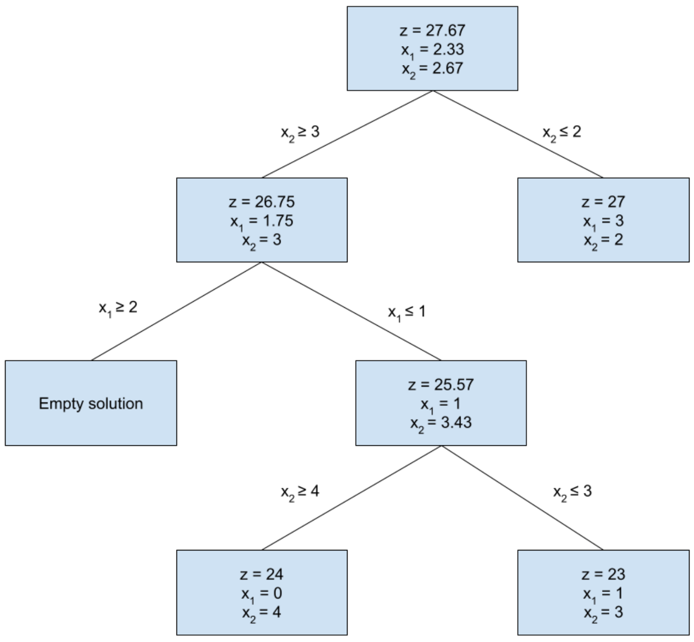


The fifth iteration would follow the additional $x_1 \leq 1$ and results the following:



Because the solution is not an integer, it requires another branching which is $x_2 \leq 3$ and $x_2 \geq 4$.

The sixth and seventh iteration:



This results in integer solutions for both iterations, therefore it ceases the branching and compares it to the lower bound, since the lower bound is z = 27, both solutions have smaller z value therefore we do not change the lower bound. Because branching is not possible any further, we take the lower bound node as the solution. In conclusion, the integer solution to this problem is $x_1 = 3$, $x_2 = 2$, $z = 27$.

IV. Conclusion

By using Branch and Bound algorithm strategy we could solve an Integer Linear Programming problem by following this step:

1. Solve the problem in the current expanded node using Graphical or Elimination method
2. If the solution results in an Integer Solution, set z value as the new Lower Bound if it is lower than the previous Lower Bound value. (If the Lower Bound value was not set before, assume it was the infinite)
3. If the solution results in an Empty Solution, terminate the node branching, else create a node branching following this rules:
    a. Determine the base number for branching inequalities, it should be the variable with the higher decimal point (*Let it called as X*)
    b. Create two branches with inequalities:

    $X \leq floor(X)$ and $X \geq floor(X) + 1$
4. Create new nodes following the FIFO rule until no more branches could be created in the state space tree.
5. If there is no more node that could be created, take the node that has the value of the Lower Bound as the solution.

Video Link at Youtube

https://youtu.be/0mN8OIipatM

Acknowledgment

Firstly, the author would like to thank God for His grace and guidance. The author would also like to thank Prof. Ir. Dwi Hendratmo Widyantoro, M.Sc., Ph.D. as the lecturer for IF2211. The author would also like to thank all the colleagues and friends who supported the author. Lastly, the author apologizes if there are mistakes in this paper.

References

[1] Kantorovich, L. V. (1940). "Об одном эффективном методе решения некоторых классов экстремальных проблем" [A new method of solving some classes of extremal problems]. Doklady Akad Sci SSSR.

[2] A. H. Land and A. G. Doig (1960). "An automatic method of solving discrete programming problems". Econometrica. 28 (3). pp. 497–520

[3] Mirzaei,Shokoufe."https://www.youtube.com/watch?v=upcsrgqdeNQ".accessed at 10th May 2021

[4] Munir,Rinaldi."http://informatika.stei.itb.ac.id/~rinaldi.munir/". accessed at 11th May 2021

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Jakarta, 11 Mei 2021

[illegible]

Renaldi Arlin 13519114