Communications on Applied Nonlinear Analysis
ISSN: 1074-133X
Vol 32 No. 3 (2025)

Comparison Between Three Algorithms to Study Their Overall Convergence

Farouk Benoumelaz

TSEGC Department, University Hadj Lakhdar, Batna, Algeria,
farouk.benoumelaz@univ-batna.dz

Article History:

Received: *29-05-2025*

Revised: *14-09-2025*

Accepted: *13-10-2025*

Abstract:

In this research paper, I presented a new algorithm for solving integer linear programming problems based on previous methods for solving such problems, including the boundary method and Gommari's truncation algorithm. The two known ones. The new algorithm relies on a coupling process between the two aforementioned methods. The reasons that led to the connection between the branch and node method and the cutting planes method are to overcome some of the disadvantages of the two methods, especially in the case of large repetitions and the large time spent on the solution, and to obtain results that are superior to the results of each of the two methods. It can be said that the new algorithm was characterized by good features and excluded and eliminated many of the bad qualities.

Keywords: Operations research, Integer programming, Gomori algorithm, Algorithms.

1. Introduction

Algorithms are valuable to us because they form the basis of much of the technology we use in our daily lives, from mobile apps to search engines. Powerful innovations in various industries enhance our capabilities (for example, artificial intelligence assistants or medical diagnostics...). That's why we work to improve them.

Linear programming is one of the most widespread optimization techniques in operational research. This is mainly due to the ease of modeling, the efficiency of the algorithms developed, and the existence of many software programs on the market. In addition, the widespread use of microcomputing has made linear programming accessible to everyone.[1]

The importance of linear programming is attributed mainly to its multiple applications, as well as its contribution to generating techniques for finding optimal solutions.

Linear programming is a decision making tool, particularly in quantitative decisions in the business world, in industrial engineering companies and, to a lesser extent, in certain activities in the life sciences and social sciences.

The main objective of linear programming is to determine the optimal allocation of scarce resources between competing activities or products. Economic situations often require that a function be optimized under several constraints in the form of inequalities.

Mathematical programming can be defined as a mathematical technique for solving management problems, particularly those where the manager must determine, given different possibilities, the

optimal use of the company's resources to achieve a specific objective, such as maximizing pro.ts or minimizing costs.[2]

In most cases, the company problems that can be treated by mathematical programming involve a number of resources, such as labor, raw materials, capital, space, etc. that are available in limited quantities and that we want to distribute in an optimal way between a number of manufacturing processes.[3]

To solve problems will be divided into two main steps:

1. Modeling the problem in the form of equations or inequalities that will allow us to properly identify and structure the constraints that the variables of the model must respect; in addition, we must de.ne the contribution of each variable to achieving the objective pursued by the company, which will result in a function to be optimized.

2. Determining the mathematical optimum using certain techniques specific to mathematical programming. Subsequently, we will examine the stability of the optimal solution in the face of certain variations in the problem data.

Manipulating the model using mathematical programming techniques will provide an objective and effective method for arriving at an optimal strategy when several possibilities must be considered. But the fact remains that mathematical programming, although it allows decisions to be prepared, cannot replace the practical sense, experience, leadership, and risk appetite of the business leader.

2. Objectives

The aim of this research is to present a new algorithm for solving an integer linear programming problems, which is a combination of two classical methods, the cutting planes method and the branching and knotting method. This algorithm is added to the methods and algorithms for solving an integer linear programming problems to provide new solutions and possibilities for those following this topic, showing them the importance of working and thinking about creating other new algorithms that shorten the effort and time to obtain the optimal solution to integer linear programming problems.

3. Methods

3.1 Branch and Bound method-node algorithm

We have the following integer programming problem

$$\begin{cases} \text{Max z} = c^t x \\ \quad st \\ \text{sc Ax} = \text{B; x} \geq 0 \text{.....1} \\ \forall x_{j;j} \in I; \text{ integer} \\ \quad . \end{cases}$$

A matrix with dimension m × s, B matrix with dimension m ×1; C matrix with dimension s × 1; X matrix with dimension s × 1:

Step 1: Initial solution

We use the simplex method to solve problem (1)

Then we neglect the restrictions related to the integer variables,if all values

$\forall x_{j;j} \in I$

are Integer, we move to the step2

Step 2:

We choose from the variables

$\forall x_{j;j} \in I$ the variable whose value is integer at that node, and this value has the largest fractional part. The variables chosen x_J must be basic, otherwise their value will be zero.[6][7]

If the basic variable chosen from the last table in solving the linear programming problem is the variable whose index is i;and its incorrect value is x_{B_i}which we can write as follows:

$x_{B_i}=\left[\left[x_{B_i}\right]\right]+f_{i'}$

where: $0< f_{i'}< 1$

Since the value of the variable x_J is integer, it satis.es one of the inequalities

$x_J \leq \left[\left[x_{B_i}\right]\right]$.....2

$x_J \geq \left[\left[x_{B_i}\right]\right]+1$.....3

Step 3:

We work on the two problems identified in the second step.

The .

first problem is to add the second constraint,the second problem is to add the third constraint,to solve them, we use the simplex method.

Step 4:(choose the fi.nal node)

All nodes obtained in step 3 may be .nal nodes, due to one of the following two things:

1-The problem at that node has no possible solution, so go to step 5

2-The values of the variables $x_{j;j} \in I$ Iare all integer, so of the objective function at that node with the best value.

I arrived at it, if the value of the objective function at the new node is better,

then change the value of the old node to this node and go to

Step5.

Step 5: Choosing the node:

1. If exactly one knot in step 4 is .nished, use the non-terminated knot and go to step 2.

2- If all the nodes in step 4 are in.nite, choose the node that is most guar-

anteed to reach the goal, which is the node at which the value of the objective function is the largest possible. Go to step 2.

The main negative characteristic of this algorithm lies in solving a complete linear programming problem at each node,the procedure, especially for large problems, will take a long time as well as a greater number of iterations.[8][9][10]

3.2 Cutting Planes Method

Cutting Planes Method or Gomary Method is a unique method and is the second convergent method for solving integer programming problems, which was published in 1958 by Gomary , the algorithm was named after him Ralph.E.Gomory.

The basic idea of this method is to add a (conclusive) constraint to the problem again and again until the ideal solution is reached. This restriction has two fundamental characteristics:

First, the incorrect optimal solution to the linear programming problem does not satisfy this constraint.

Second, all possible correct solutions to the original problem will satisfy this new constraint.[3]

Cutting Planes Method algorithm:

We have the following integer programming problem.

$$\begin{cases} \text{Max z} = c^t x \\ \quad st \\ \text{sc Ax} = \text{B; x} \geq 0 \text{.....4} \\ \forall x_{j;j} \in I; \text{ integer} \end{cases}$$

A matrix with dimension m × s, B matrix with dimension m ×1; C matrix with dimension s × 1;X matrix with dimension s × 1:

Step 1: Initial solution

We use the simplex method to solve problem (2)

Then we neglect the restrictions related to the integer variables,if all values

$\forall x_{j;j} \in I$

are Integer, we move to the step2

Step 2:

Choose the line from the last table in the solution to the linear programming problem in which the basic variable x_{B_i} is not an integer value (bi use the line in which the value of that variable has the largest fractional part, because this may help reduce the number of iterations and the time it takes to converge) and from it generate or create a cutting planes constraint.

Step 3

Let the chosen line be the line with index i and its equation is:

$$x_{B_i} + \sum_{J=1}^{n} a_{ij\cdot}\, x_j = b_{ij}; j \in I$$

$$x_{B_i} + \sum_{J=1}^{n} (\,[a_{ij}] + f_{ij\cdot})x_i = [b_i] + f_i$$

$$x_{B_i} + \sum_{J=1}^{n} ([a_{ij}]x_i - [b_i])x_i = f_i - \sum_{J=1}^{n} f_{ij}\, x_i \leq 0$$

New condition is

$$f_i - \sum_{J=1}^{n} f_{ij}\, x_i + \mu = 0 \quad \ldots 5$$

such as

$f_{ij} = a_{ij\cdot} - [a_{ij}]$is the fractional part of $a_{ij\cdot} 0 \leq f_{ij} \leq 1$

$f_i = b_i - [b_i]$ is the fractional part of $b_i 0 \leq f_i \leq 1$

μ new auxiliary variable is integer

Step 4

Add constraint

Add the constraint 5 to the last table of the solution to the simplex algorithm and solve the solution as a linear programming problem. [11]

Here we discuss the following cases

$\forall x_{j;j} \in I$ If the values are integer and possible, the problem is end. Otherwise, go to the next step2,

3.3 New algorithm for solving integer linear programming problems

The proposed method is a new technique that is very successful in solving a wide range of integer programming problems Guarantee of reaching the integer ideal solution. We will explain below how this technique is performed, noting that the new cutting and branching algorithm is used
It can be developed and expanded, and it is a promising algorithm that can be computer programmed, and it will be of greater importance with the exploitation of speed Current computers.

The new technique and branching algorithm is a combination of the cutting planes method and the branching and knotting method, and it works like its predecessors, it involves solving a successive series of linear programming problems to obtain the solution to the integert programming problem.
The method of cutting planes that was explained in the previous paragraph 5 does not seem to be a strong method, as it seeks to obtaining the integer ideal solution has a slow convergence, and may not give the optimal solution or guarantee obtaining it, as in the branching method
And the nodes that are faster and more guaranteed to reach the optimal solution, so we tried to make the level cutting method better from the previous, this was done by making a synthesis and linking it with the method of branch ing and contracting, and we called this new synthesis the method of cutting and branching, noting here that this new structure was built through the two algorithms described in paragraphs 1.6. and the following two.6
We have the following integer programming problem.

$$\begin{cases} \text{Max z} = c^t x \\ \quad st \\ \text{sc Ax} = \text{B; x} \geq 0 \text{.....6} \\ \forall x_{j;j} \in I; \text{ integer} \\ \quad . \end{cases}$$

A matrix with dimension m × s, B matrix with dimension m ×1; C matrix with dimension s × 1;X matrix with dimension s × 1:

Step 1: Initial solution

Start by solving the problem given in 6 as a linear programming problem using the Simplex algorithm, ignoring the constraints. intrgera on the decision variables $x_{j;j} \in I$
If all variables $\forall x_{j;j} \in I$ are integer values, stop, otherwise go to step2.
Step 2:We choose the aconstraint
We use the simplex method to solve problem (6)
Then we neglect the restrictions related to the integer variables,if all values
$\forall x_{j;j} \in I$ are Integer, we move to the step 6.
Choose the line from the last table in the solution to the linear programming problem in which the basic variable x_{B_i} is not integer value (b_i use the line in which the value of that variable has the largest fractional part, because this
may help reduce the number of iterations and the time it takes to converge) and from it generate or create a cutting planes constraint.
Step 3: Generate the cutting planes constraint:

$$x_{B_i} + \sum_{J=1}^{n} a_{ij\cdot}\, x_j = b_{ij}; j \in I$$

$$x_{B_i} + \sum_{J=1}^{n} (\left[a_{ij}\right] + f_{ij\cdot}) x_i = [b_i] + f_i$$

$$x_{B_i} + \sum_{J=1}^{n} (\left[a_{ij}\right] x_i - [b_i]) x_i = f_i - \sum_{J=1}^{n} f_{ij}\, x_i \leq 0$$

New condition is

$$f_i - \sum_{J=1}^{n} f_{ij}\, x_i + \mu = 0 \quad \ldots 7$$

such as

$f_{ij} = a_{ij\cdot} - [a_{ij}]$is the fractional part of $a_{ij\cdot} 0 \le f_{ij} \le 1$

$f_i = b_i - [b_i]$ is the fractional part of $b_i 0 \le f_i \le 1$

μ new auxiliary variable is integer

Step 4 Add constraint

Add the constraint 7 to the last table of the solution to the simplex algorithm and solve the solution as a linear programming problem. Here we discuss the following cases

1-Variables$\forall x_{j;j} \in I$ If the values are integer and possible, the problem is

end. Otherwise, go to the next step2.

2-Some variables $\forall x_{j;j} \in I$ If the values are not integer ,go to step 2. Otherwise, go to the next step 3.
3-One of the variables $\forall x_{j;j} \in I$ I has an integer value, then start using the branch and node method to branch ,the values are not integer variable, go to step5.
Step 5: Choose the branching variable
Choose the variable with the not integer value from the variables $\forall x_{j;j} \in I$, which will be used to form the branching constraints, which It has the largest fractional part.he variable x_i that is chosen must be a basic variable, otherwise its value will be zero. Assume that the variable is let the basic variable i from the last table to solve the simplex algorithm be its value x_{B_i}
We will write
$x_{B_i} = [x_{B_i}] + f_i$
where

$$0 < f_i < 1$$

$x_J$satisfies one of the 8 and 9 inequalities.

$x_J \le \left[[x_{B_i}]\right]$.....8

$x_J \ge \left[[x_{B_i}]\right]$+1.....9

Step 6:New nodes

Create two new integer programming problems with the constraints in step 5 the first problem consists of adding the constraint, 8, and the second problem consists of adding the constraint,9. Solve each of the following: ,these two

problems are linear programming problems using the Simplex algorithm

Step 7:Node test

Each of the nodes obtained in step 6 is a finished node in one of the following two cases:

The first case: The problem represented by this node has no possible solution.

The second case: All the variables $\forall x_{j;j} \in I$ are integer and this is the optimal solution to the problem 6. But if those,the knot is not finished, so go to the next step5.

4.Results

The general problem of linear programming is to .nd the values of variables that maximize or minimize the value of the objective function according to constraints representing the amount of available resources. We solve the integer

linear programming problem first, ignoring the conditions for the variables to take integer values, that is, we solve the problem as a linear programming problem in which we are asked to find the optimal true solution. After that, we use the algorithms of the integer linear programming problem to move from

NOI number of reiteration ; CPU Time

Problems	Cutting planes		Branch and Bound		New Algorithm	
	NOI	CPU	NOI	CPU	NOI	CPU
$\begin{cases} Max\ z = 6x_1 + 7x_2 \\ st \\ x_1 + 2x_2 \leq 8 \\ x_1 - x_2 \leq 4 \\ x_1; x_2 \geq 0\ integer \end{cases}$	**2.**	**1.07**	**9.00**	**0.72**	**3**	**0.82**
$\begin{cases} Max\ z = 2x_1 + 1x_2 - x_3 \\ st \\ 1x_1 + 2x_2 - x_3 \leq 5 \\ -x_1 + 2x_2 + 3x_3 \leq -2 \\ x_1; x_2; x_3 \geq 0\ integer \end{cases}$	**1**	**0.66**	**2**	**0.32**	**1**	**0.48**
$\begin{cases} Max\ z = x_1 + 1x_2 - x_3 \\ st \\ x_1 + 2x_2 - x_3 \leq 3 \\ -x_1 + 2x_2 + 3x_3 \leq 2 \\ x_1; x_2; x_3 \geq 0\ integer \end{cases}$	**2**	**0/86**	**9**	**0.72**	**1**	**0.48**
$\begin{cases} Max\ z = 4x_1 + 3x_2 \\ st \\ 2x_1 + x_2 \leq 11 \\ -x_1 + x_2 \leq 6 \\ x_1; x_2 \geq 0\ integer \end{cases}$	**4**	**1.04**	**5**	**0.39**	**1**	**0.48**
$\begin{cases} Max\ z = 3x_1 + x_2 \\ st \\ x_1 + 2x_2 \leq 8 \\ 3x_1 - 4x_2 \leq 12 \\ x_1; x_2 \geq 0\ integer \end{cases}$	**5**	**1.6**	**7**	**0.32**	**3**	**0.54**

$\begin{cases} Max\ z = 5x_1 + 8x_2 \\ st \\ x_1 + x_2 \le 6 \\ 5x_1 + 94x_2 \le 49 \\ x_1; x_2 \ge 0\ integer \end{cases}$	**3**	**0.88**	**8**	**0.32**	**6**	**0.76**
$\begin{cases} Max\ z = 120x_1 + 80x_2 \\ st \\ 2x_1 + x_2 \le 6 \\ 7x_1 + 8x_2 \le 28 \\ x_1; x_2 \ge 0\ integer \end{cases}$	**4**	**0.91**	**7**	**0.34**	**3**	**0.63**
$\begin{cases} Max\ z = 3x_1 + x_2 \\ st \\ 4x_1 + 3x_2 \le 18 \\ 3x_1 - 4x_2 \le 6 \\ x_1; x_2 \ge 0\ integer \end{cases}$	**1**	**0.33**	**2**	**0.12**	**5**	**0.67**
$\begin{cases} Max\ z = 3x_1 + x_2 \\ st \\ 2x_1 + 3x_2 \le 8 \\ 2x_1 - 4x_2 \le 3 \\ x_1; x_2 \ge 0\ integer \end{cases}$	**3**	**0.64**	**4**	**0.25**	**1**	**0.18**
$\begin{cases} Max\ z = -2x_1 + 1x_2 - x_3 \\ st \\ 1x_1 + x_2 - 2x_3 \le 5 \\ -2x_1 - 2x_2 + x_3 \le 2 \\ x_1; x_2; x_3 \ge 0\ integer \end{cases}$	**3**	**0.63**	**4**	**0.23**	**3**	**0.42**
$\begin{cases} Max\ z = 2x_1 - 1x_2 + x_3 \\ st \\ 1x_1 + 2x_2 - x_3 \le 5 \\ -2x_1 - 2x_2 + 3x_3 \le -2 \\ x_1; x_2; x_3 \ge 0\ integer \end{cases}$	**3**	**0.62**	**4**	**0.24**	**3**	**0.62**
Total	**30**	**6.78**	**53**	**3.81**	**36**	**6.47**

4.1.Interpretation of results

We show the percentage of performance improvement in the new algorithm compared to the classical methods. Classic cutting planes, branching and nodes
Total
Branch and Bound
Number of iterations
58.88%
Solution time
147%Number of iterations
Cutting planes
Number of iterations
101.19%

Solution time
83.33%Number of iterations
New Algorithm
Number of iterations
99.99%
Solution time
99.98%Number of iterations
The efficiency of the calculation and the precision of the results.
We show the percentage of performance improvement in the new algorithm compared to the classical methods. Classic cutting planes, branching and nodes
Efficiency ratio
Branch and Bound
07.18%
Cutting planes
22.6%
New Algorithm
17.99%
We notice that the first and second features of the new algorithm are achieved from the characteristics of the new cut and branch method.

4. Discussion

We conclude from this research that two methods can be combined to obtain a third method with better results or similar to the results of other algorithms.

The Cutting planes method and the branch-and-bound method can be combined more than once. We also note that the Cutting planes method was reconnected, meaning we obtained a branch a branching, resulting in two solutions: a solution outside the solution region and an integer optimal solution to the problem. This new triple combination is an excellent combination, as it takes on the good qualities of the new method and the Cutting planes method. This opens up a broad and new field in integer linear programming problems.

From the three previous properties, the new cut-and-knot method is guaranteed to reach the optimal solution because it ends with the branching-andknotting method.

From the three properties mentioned in the paragraph above, the efficiency of the calculation and the precision of the results, which indicate that the new cut-and-knot method is guaranteed to reach the optimal solution because it ends with the branching-and-knotting method, it is guaranteed to reach the optimal solution. Therefore, we recommend using this method and exploring other combinations of known algorithms in integer linear programming to develop new methods, taking advantage of the high speeds of current computers.

We also recommend using the new algorithm to solve the integer linear programming problem, as it is a very successful technique in solving a wide range of integer programming problems and it provides a guarantee of reaching the optimal solution.

Since the steps are precise enough to specify exactly what to do at each step, the precise order of operations in the algorithm is well organized and concrete.

All steps in the algorithm are feasible (also known as efficiently computable). I recommend using the algorithm.

References

[1] Cornuéjols, Gérard (2007). Revival of the Gomory Cuts in the 1990s. Annals of Operations Research, Vol. 149 (2007), pp. 63.66.

[2] Gilmore, Paul C; Gomory, Ralph E (1963). "A linear programming approach to the cutting stock problem-Part II". Operations Research. 11 (6):863.888

[3] Cornuéjols, Gérard, "Revival of the Gomory Cuts in the 1990's" (2005). Tepper School of Business. Paper71 http://repository.cmu.edu/tepper/71.

[4] Cornuéjols, Gérard (2008). Valid Inequalities for Mixed Integer Linear Programs. Mathematical Programming Ser. B, (2008) 112:3.44.

[5] Dantzig, G. B. 1963, Linear Programming and Extensions, Princeton University Press, Princeton, NJ.

[6] A H. Land and A. G. Doig (1960). "An automatic method of solving discrete programming problems". Econometrica. 28 (3): 497.520.

[7] Narendra, Patrenahalli M.; Fukunaga, K. (1977). "A branch and bound algorithm for feature subset selection" (PDF). IEEE Transactions on Computers. C-26 (9): 917.922.

[8] Nowozin, Sebastian; Lampert, Christoph H. (2011). "Structured Learning and Prediction in Computer Vision". Foundations and Trends in Computer Graphics and Vision. 6 (3.4): 185.365.

[9] D.R. Morrison, E.C. Sewell, S.H. Jacobson, An application of the branch, bound, and remember algorithm to a newsimple assembly line balancing dataset, European J. Oper. Res. (2013) in press.

[10] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. Bixby, E. Danna, G. Gamrath, A. Gleixner, S. Heinz,A. Lodi, H. Mittelmann, T. Ralphs, D. Salvagnin, D. Ste¤y, K. Wolter, MIPLIB 2010: Mixed integer programming library version 5, Math. Program. Comput. 3 (2011) 103.163

[11] E.C. Sewell, S.H. Jacobson, A branch, bound, and remember algorithm for the simple assembly line balancing problem,INFORMS J. Comput. 24(2012) 433.442