Pesquisa Operacional (2022) 42: e255132 p.1-19
doi: 10.1590/0101-7438.2022.042.00255132
© 2022 Brazilian Operations Research Society
Printed version ISSN 0101-7438 / Online version ISSN 1678-5142
www.scielo.br/pope
ARTICLES

A NEW STRATEGY TO SOLVE LINEAR INTEGER PROBLEMS WITH SIMPLEX DIRECTIONS

Bruno Luís Hönigmann Cereser[1*], Aurelio Ribeiro Leite de Oliveira[2] and Antonio Carlos Moretti[3]

Received August 9, 2021 / Accepted January 9, 2022

ABSTRACT. In this paper, we propose a new heuristic strategy to solve linear integer mathematical problems. The strategy begins by finding the optimal solution of the continuous associated problem and the simplex directions from that optimal solution. A total of $n-m$ problems are generated and solved by the strategy, where m is the number of constraints and n is the number of variables of the problem. The best solution to those problems is the solution to the original problem. Several instance problems were randomly generated to validate the strategy, and our strategy finds good solutions in 80% of the instances. We also tested the proposed strategy with instances of the MIPLIB 2017. We compare a total of 39 instances and on average our strategy performed 47% fewer iterations than the solver, in 34 of 39 instances, the strategy found good solutions.

Keywords: mixed integer linear programming, algorithms, heuristic.

1 INTRODUCTION

Many practical problems can be formulated using integer programming. An Integer Linear Program (ILP) can be written as (1).

$$\begin{array}{rl} \max & z = c^t x \\ subject\ to & \begin{cases} Ax \leq b \\ x \geq 0 \text{ and integer.} \end{cases} \end{array} \tag{1}$$

*Corresponding author

[1]IMECC, Unicamp, Campinas, São Paulo, Brazil. Email: brunolhcereser@gmail.com https://orcid.org/0000-0003-0753-9386

[2]IMECC, Unicamp, Campinas, São Paulo, Brazil. Email: moretti@ime.unicamp.br https://orcid.org/0000-0002-6471-4710

[3]IMECC, Unicamp, Campinas, São Paulo, Brazil. Email: aurelio@ime.unicamp.br https://orcid.org/0000-0001-5604-1002

Where A is a $m \times n$ matrix, b is a $m \times 1$ vector, c is a $n \times 1$ vector and x is a $n \times 1$ vector of variables. If some, but not all, variables of the problem are continuous we have a Mixed Integer Linear Program (MILP), it can be written as (2).

$$\begin{array}{rl} \max & z = c^t x + d^t y \\ subject\ to & \begin{cases} Ax + Gy \leq b \\ y \geq 0 \\ x \geq 0 \text{ and integer.} \end{cases} \end{array} \tag{2}$$

Where G is a $m \times p$ matrix, d is a $p \times 1$ vector and y is a $p \times 1$ vector of variables. It could be challenging to solve general MILP or ILP problems efficiently, therefore, several strategies and methods are proposed in the literature to solve those problems. In this paper, we propose a new strategy to solve integer linear problems. Given an ILP and its LP relaxation solution, the heuristic solves one or more external MILP that aims at finding an integer solution closer to an edge of the LP relaxation solution. The strategy starts from an LP relaxation optimal solution x^* and the associated simplex directions d_i, $i = 1, 2, ..., m-n$. Using x^* and d_i, the method defines a MILP that aims to find an integer solution closer to an edge, and since we have $m-n$ simplex directions, we can define $m-n$ MILP problems, therefore, the strategy performs at most $n-m$ iterations. Each iteration solves a MILP that is much easier than the original, and some of those can be infeasible. At the end of the strategy, we will have at most $n-m$ integer solutions. The solution with the best objective function value is selected. In most tests, the MILP problems take fewer iterations to be solved and generate a good integer solution to the original problem. The difference between our strategy and the strategies found in the literature is mainly in the use of the simplex directions for determining integers solutions.

2 BIBLIOGRAPHY REVIEW

The conventional exact strategies to solve ILP and MILP problems are the branch and bound, cutting plane and its variations, those methods are found in Nemhauser & Wolsey (1988) and Wolsey (2007). In problems with many constraints and variables, apart from special structured or well-solved problems, such as the problems described in Wolsey (2007), it needs several iterations to find a solution. To overcome this, we have non-exact methods, such as heuristics, metaheuristics, and matheuristics methods, the one did not guarantee the optimum solution, but they need significantly fewer iterations to converge. Table 1 was mainly elaborated with information from Genova & Guliashki (2011) and Fischetti & Lodi (2010) and, summarizes some of those strategies and methods.

The feasibility pump, described in Fischetti et al. (2005) and Berthold et al. (2019), is a heuristic strategy to efficiently solve mixed-integer problems. The one with any feasible solution of the set $P = \{Ax \leq b\}$, and defines its rounding. Each iteration searches for a point $x^* \in P$ that is as close as possible to the current $\bar{x}$ by solving the problem $\min \delta(x, \bar{x}) : x \in P$. Assuming $\delta(x, \bar{x})$ is chosen properly, they claim that linear problems can be easily solved. When $\delta(x^*, \bar{x}) = 0$, then x^* is a feasible MIP solution and the feasibility pump ends. Otherwise, $\bar{x}$ is replaced by the rounding

Table 1 – Strategies and Methods.

Strategy	**Summary**
Rounding methods	Fractional components of the LP solution are rounded.
Diving methods	Also known as Relax-and-Fix. Sequentially fix some integer-valued variables that assume a fractional value in the solution of the current LP relaxation. The sequence can be viewed as "diving" on a branch-and-bound tree.
Pivoting Methods	It is based on the observation that a feasible pure 0–1 ILP solution is just a basic solution of the LP relaxation where all 0–1 variables are nonbasic. See Eckstein & Nediak (2007) for an example.
Line Search Methods	Starts from an LP relaxation optimal solution and draws a line toward a second point and move in that line in a discretized way. An example of a line search method can be seen in Yuan & Wei (2009).
Feasibility Pump	It is based on the observation that a feasible MILP solution is a point that is close with its rounding. See Fischetti et al. (2005) and Berthold et al. (2019) for an example of the Feasibility Pump.
Local Branching	Given x, a feasible reference solution of a MILP, Local Branching aims at finding an improved solution that is in the neighborhood of x. See Fischetti & Lodi (2003) for an example of Local Branching.
Relaxation Induced Neighborhood Search (RINS)	Uses the solutions of simplified MILP and LP problems to find a heuristic solution to the original problem. See Danna et al. (2005) for an example of RINS.
Evolutionary Algorithms	Evolutionary Algorithms are metaheuristics. In general, they have a Population, Combination, Mutation, and Selection of solutions. See Rothberg (2007) for an example of Evolutionary Algorithms to solve MILP problems.

of x^*, and the step is repeated until $\delta(x^*, \bar{x}) = 0$. In Achterberg & Berthold (2007) the feasibility pump was improved, and they can obtain better solutions for MILP instances.

According to Danna et al. (2005) the RINS uses solutions of simplified external MILPs and LPs to find the solution of the original MILP problem. At specified nodes of the branch-and-bound tree, the current LP relaxation solution x^* and the incumbent $\bar{x}$ are compared and all integer variables that agree in value are fixed and removed from the MILP. In this way, the MILP is simplified, and it is also reduced in size.

The work of Eisenbrand & Weismantel (2019) describes the Proximity search as an alternative to Large-Neighborhood Search and RINS heuristics, which are aimed at improving a given feasible solution. The one modifies the objective function to make the search easier. More specifically, proximity search replaces the objective function with a *proximity* one, to enhancing the heuristic behavior of the MIP black-box solver. The one starts with a feasible solution $\bar{x}$, and adds an explicit cutoff constraint $f(x) \leq f(\bar{x}) - \theta$ to the MIP, where θ is a given cutoff tolerance.

In Richard et al. (2003), the authors present a simplex-based algorithm for mixed-integer problems. The algorithm uses a hybrid cutting plane, based on a strong cutting plane and Gomory-type. The cuts are obtained from the relaxation of the simplex tableau and used on the original problem. The authors show that the cuts can be computed in polynomial time and can be embedded in a finitely convergent algorithm.

Some Lagrangian relaxation strategies, such as Fisher (2004), can be used to improve the bounds of the branch and bound method faster. Metaheuristics methods like particle swarm (see Laskari et al. (2002)), Simulated Annealing, Genetic Algorithm (see Arenales et al. (2007)) are usually used to solve complex and large MILP instances.

Some strategies are developed for specific problems, such as Park & Kim (2005) where the authors present proposes a method to solve the scheduling problem of a port berth and quay cranes and Biagio et al. (2012) which proposes a heuristic to solve capacitor allocation problems in electric energy radial distribution networks. New applications of preexisting strategies can also be found, like Ceschia et al. (2017) which proposes a Simulated Annealing application for the discrete single-machine, multi-item lot-sizing scheduling problem and obtained good results.

3 DETERMINATION OF INTEGER SOLUTIONS WITH SIMPLEX DIRECTIONS

Given an integer problem, such as (1) and x^* the LP relaxation solution, an integer solution of the problem will be generated with the simplex directions associated with x^*. That is, we start with an LP relaxation solution of the ILP and search for integer solutions on the adjacent edges of x^*. If the edges do not contain any integer solutions, we search for the feasible integer solution closer to an edge.

3.1 Simplex Direction

According to Bazaraa et al. (2011), the simplex direction d^* of an LP is defined as the direction that starts at a basic solution x_B and walks on an edge of the polytope generated by the constraints. This generates the line $w^* = x_B + \lambda d^*$, Figure 1 shows the simplex direction and the line $w^* = x_B + \lambda d^*$.

Figure 1 shows the x_B on the intersection of lines 1 and 2 and the direction d^* which is the simplex direction that leaves x_B and generates the line segment $w^* = x_B + \lambda d^*$ where $\lambda \geq 0$. Definition 1 formalizes the simplex direction concept.

Definition 1. (Simplex Direction) *The simplex direction d^* is defined by the direction which starts at a basic solution x_B and walks on the edge of the polytope generated by the constraints. Mathematically $d^* = \begin{bmatrix} B^{-1}a_{N_k} \\ -e_k \end{bmatrix}$ where N_k is the index of the non-basic variable which is entering on the basis, a_{N_k} is the N_k-th column of the A matrix.*

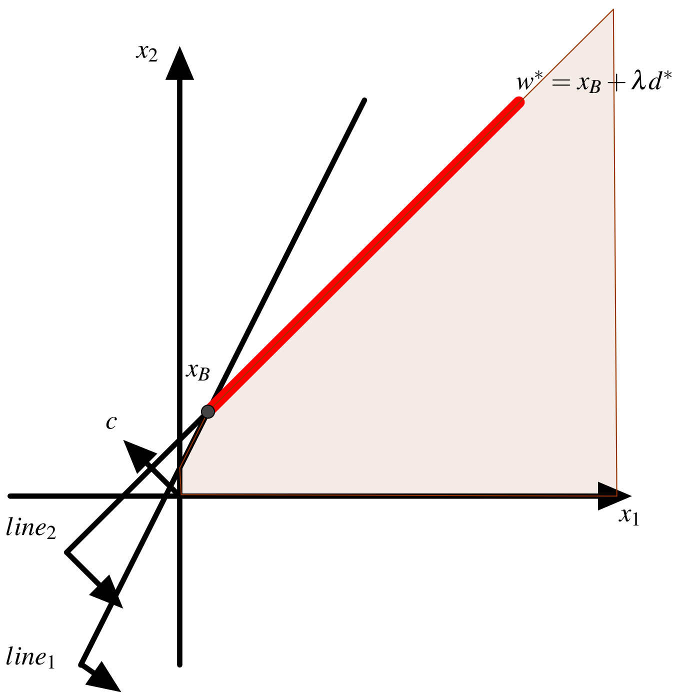

Figure 1 – Simplex Direction.

3.2 Generalization of the Strategy

Given an ILP problem, such as Problem (1), the first step is to define and solve the LP relaxation, given by Problem (3).

$$\begin{array}{rl} \max & z = c^t x \\ subject\ to & \left\{ \begin{array}{l} Ax \leq b \\ x \geq 0. \end{array} \right. \end{array} \tag{3}$$

With the LP relaxation solution x^* of the Problem (3), we calculate the simplex directions in x^*. There are $n-m$ simplex directions in x^* and those directions are decent if it is a maximization problem or ascent if it is a minimization problem. To obtain those simplex directions, we solve the linear systems given by (4).

$$y_k = B^{-1} a_{N_k} \qquad k = 1,2,\ldots,n-m. \tag{4}$$

Where N is the non-basic matrix and N_k is the k-th column of N. The simplex direction d_k, where $k = 1,2,\ldots,n-m$, is given by (5).

$$d_k = \begin{bmatrix} y_k \\ -e_k \end{bmatrix}. \tag{5}$$

The next step is to find integer solutions in the lines $x^* + \lambda d_k$. To do that we need to select a k in $1, 2, ..., n-m$, and solve the ILP given by (6).

$$\begin{array}{rl} \max & z = c^t x \\ subject\ to & \begin{cases} Ax \leq b \\ x = x^* - \lambda d_k \\ x \geq 0 \text{ and integer.} \end{cases} \end{array} \tag{6}$$

Where $k = 1, 2, ..., n-m$. After solving the problems, we will have at most $n-m$ integer solutions. We select the solution with the best object function value.

This strategy is for an ILP problem. For a MILP problem, the continuous variables should not be included in the step of searching for an integer solution on the adjacent edges. There are cases when the problem has no integer solution on the adjacent edges of the continuous solution. In those cases, we need to find the integer solution closer to an edge.

If the edges do not contain at least one good integer solution, the next step is to find the integer solutions inside the feasible region that are closer to an edge. To find these solutions, we solve one or more of the $n-m$ problems given by (7).

$$\begin{array}{rl} \max & z = c^t x - \beta^t p \\ \text{subject to} & \begin{cases} Ay \leq b \\ x = x^* - d_k \lambda \\ y = x - p \\ x, p \geq 0 \\ y \geq 0 \text{ and integer.} \end{cases} \end{array} \tag{7}$$

Where $k = 1, 2, ..., n-m$, β is the penalization value to ensure a solution close to the edge, it must be proportional to c, p is the distance between the integer solution and the edge, x is the solution in the edge and y is an integer solution.

By solving the mathematical models, we obtain at most $n-m$ integer solutions. The solution with the best object function value is chosen. Algorithms 1 and 2 summarize the strategy of this paper.

Algorithms 1 and 2 solve $n-m$ integer problems, which could take many iterations and not be a good strategy. In some problems, such as the MIPLIB problems, a selection criterion can be applied instead of testing every simplex direction. In Section 4 we present a selection criterion for the MIPLIB problems.

Algorithm 1 Algorithm to find integer solutions on the edges

```
1: Given an ILP as the Problem 1.
2: Find the LP relaxation solution $x^*$ of Problem 3;
3: Define the matrix with slack variables $Aext = [A_{m\times n}\ I_{m\times m}]$;
4: Find the ordered basis index $index_B$ and the basic columns of $Aext$, use them to find the basis
   matrix $B$ of the solution $x^*$;
5: Find the ordered non-basis index $index_{NB}$ and the non-basis matrix $NB$;
6: Initiate the values of $x$ and $f$ as the solution and objective function values;
7: while k $\leq$ n-m do
8:     Solve the linear system $By = NB_k$;
9:     Define $d$ according to equation (5);
10:    Solve the Problem (6), find the solution value $\hat{x}$ and the objective function value $f_{\hat{x}}$;
11:    if $f < f_{\hat{x}}$ then
12:        $f = f_{\hat{x}}$ and $x = \hat{x}$;
13:    end if
14:    $k = k+1$
15: end while
```

Algorithm 2 Algorithm to find integer solutions closer to the edges of the problem

```
1: Given an integer linear problem as the Problem 1.
2: Find the LP relaxation solution $x^*$ of Problem 3;
3: Define the matrix with slack variables $Aext = [A_{m\times n}\ I_{m\times m}]$;
4: Find the ordered basis index $index_B$ and the basic columns of $Aext$, use them to find the basis
   matrix $B$ of the solution $x^*$;
5: Find the ordered non-basis index $index_{NB}$ and the non-basis matrix $NB$;
6: Initiate the values of $x$ and $f$ as the solution and objective function values;
7: while k $\leq$ n-m do
8:     Solve the linear system $By = NB_k$;
9:     Define $d$ according to equation (5);
10:    Define the value of the penalization $\beta$;
11:    Solve the Problem (7), find the solution value $\hat{x}$ and the objective function value $f_{\hat{x}}$;
12:    if $f < f_{\hat{x}}$ then
13:        $f = f_{\hat{x}}$ and $x = \hat{x}$;
14:    end if
15:    $k = k+1$
16: end while
```

3.3 Example Problem

Consider, as an example, the Problem (8) as our ILP.

$$\begin{array}{ll} \max & z = 8x_1 + 5x_2 \\ s.t. & \begin{cases} 1x_1 + 1x_2 \leq 6 \\ 9x_1 + 5x_2 \leq 45 \\ x_1, x_2 \geq 0 \text{ and integer} \end{cases} \end{array} \tag{8}$$

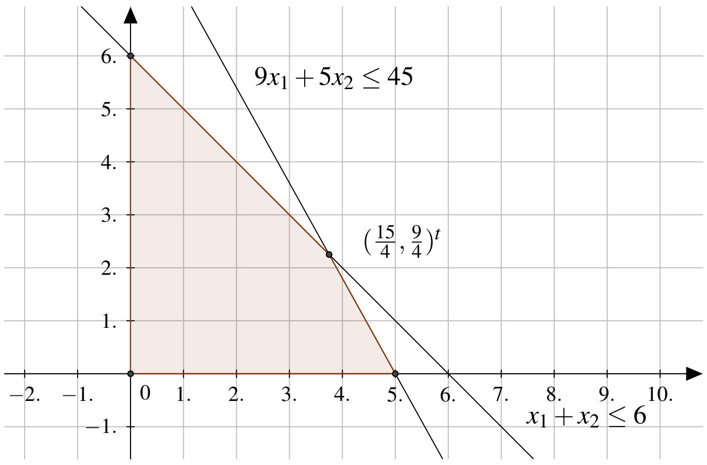

Figure 2 – Constrains of Problem (8).

Figure 2 shows the geometric interpretation of Problem (8), in the figure we can see the LP relaxation solution $x^* = (\frac{15}{4}, \frac{9}{4})^t$ and the adjacent edges. Consider x^* as the initial solution and the simplex directions d_1 and d_2, given by the vectors (9a) and (9b), respectively.

$$d_1 = \begin{pmatrix} \frac{5}{4} \\ \\ -\frac{9}{4} \end{pmatrix} \tag{9a}$$

$$d_2 = \begin{pmatrix} -\frac{1}{4} \\ \\ \frac{1}{4} \end{pmatrix} \tag{9b}$$

Starting with the solution x^* and the simplex directions d_1 and d_2, we search for integer solutions in the line segments generated by the optimum point and the simplex directions. These line segments must be limited by the problem constraints. Starting from x^* we search for integer solutions, if exists, in the edges defined by $x^* + \lambda d_1$ and $x^* + \lambda d_2$, which are given by the equations $(10a)$ and $(10b)$, respectively.

$$R_1 \begin{cases} x_1 = \frac{15}{4} + \frac{5}{4}\lambda \\ x_2 = \frac{9}{4} - \frac{9}{4}\lambda \end{cases} \tag{10a}$$

$$R_2 \begin{cases} x_1 = \frac{15}{4} - \frac{1}{4}\lambda \\ x_2 = \frac{9}{4} - \frac{1}{4}\lambda \end{cases} \tag{10b}$$

The objective is to obtain an integer solution that satisfies the equation (10a) or (10b), the constraints, and maximizes the problem. The problem of finding a solution in the line (10a) which satisfies the constraints of Problem (8) is given by Problem (11).

$$\begin{array}{ll} \max & z = 8x_1 + 5x_2 \\ \text{subject to} & \begin{cases} 1x_1 + 1x_2 \leq 6 \\ 9x_1 + 5x_2 \leq 45 \\ x_1 + 0x_2 - \frac{5}{4}\lambda = \frac{15}{4} \\ 0x_1 + x_2 + \frac{9}{4}\lambda = \frac{9}{4} \\ x_1, x_2 \geq 0 \text{ and integer.} \end{cases} \end{array} \tag{11}$$

Problem (11) is an ILP with a feasible region given by one edge of the original problem. The feasible region has only a fraction of the number of integer solutions of the problem and, may not have integer solutions. With some mathematical manipulations, the Problem (11) can be written as the Problem (12).

$$\begin{array}{ll} \max & z = 8x_1 + 5x_2 \\ \text{subject to} & \begin{cases} x_1 = \frac{15}{4} + \frac{5}{4}\lambda \\ x_2 = \frac{9}{4} - \frac{9}{4}\lambda \\ 0 \leq \lambda \leq 1 \\ x_1, x_2 \geq 0 \text{ and integer.} \end{cases} \end{array} \tag{12}$$

The solution of Problem (12) is an integer solution that tends to be closer to the LP relaxation solution that is in line (10a), given that the direction d_1 is descending on a maximization problem. Solving Problem (12), we find the solution $(5,0)^t$ with the objective function value of 40.

We also need to obtain the best integer solution in line (10b). Similarly, the problem of finding the integer solution in line (10b) and satisfies the constraints of Problem (8) is given by (13).

$$\begin{array}{ll} \max & z = 8x_1 + 5x_2 \\ \text{subject to} & \begin{cases} 1x_1 + 1x_2 \leq 6 \\ 9x_1 + 5x_2 \leq 45 \\ x_1 + 0x_2 + \frac{1}{4}\lambda = \frac{15}{4} \\ 0x_1 + x_2 - \frac{1}{4}\lambda = \frac{9}{4} \\ x_1, x_2 \geq 0 \text{ and integer.} \end{cases} \end{array} \tag{13}$$

Problem (13) is an ILP with line (10b) as the feasible region. With some mathematical manipulations, Problem (13) can be written as Problem (14).

$$\begin{array}{ll} max & z = 8x_1 + 5x_2 \\ \text{subject to} & \begin{cases} x_1 = \frac{15}{4} - \frac{1}{4}\lambda \\ x_2 = \frac{9}{4} + \frac{1}{4}\lambda \\ 0 \leq \lambda \leq 15 \\ x_1, x_2 \geq 0 \text{ and integer.} \end{cases} \end{array} \tag{14}$$

Solving Problem (14) we find the solution $(3,3)^t$ with the objective function value of 39.

Comparing the solutions of Problems (12) and (14) we have the best integer solution on the adjacent edges as $(5,0)^t$ with the objective function value of 40.

3.4 Example Problem 2

The problem may not have a feasible integer solution on the adjacent edges. In that case, we need to find integer solutions inside the feasible region. As an example, Problem (15) has no integer solution in the adjacent edges.

$$\begin{array}{ll} \max & z = 8x_1 + 5x_2 \\ s.t. & \begin{cases} 1x_1 + 1x_2 \leq 6.5 \\ 9x_1 + 5x_2 \leq 44 \\ x_1, x_2 \geq 0 \text{ and integer.} \end{cases} \end{array} \tag{15}$$

Figure (3) shows the feasible region of Problem (15). The LP relaxation solution of Problem (15) is given by $x^* = (2.875, 3.625)^t$ and neither of the adjacent edges has integer solutions.

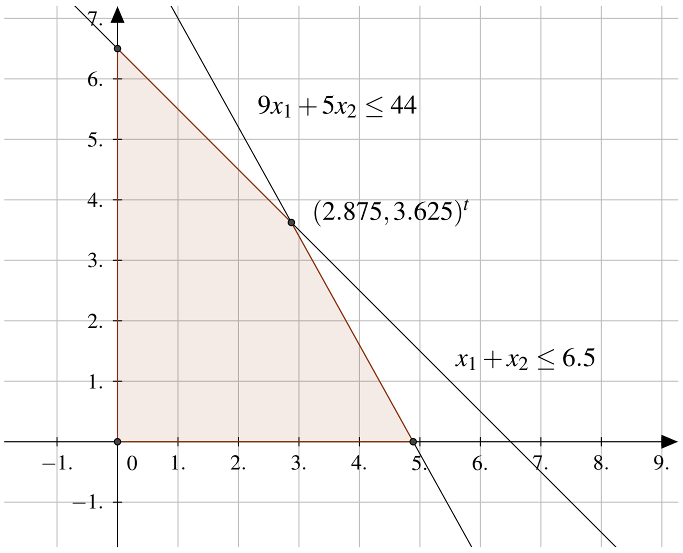

Figure 3 – Feasible Region of Problem (15)

Problem (16) can be used to find the integer solution that tends to be closer to the adjacent edge d_1.

$$\begin{array}{ll} \max & z = 8y_1 + 5y_2 - \beta_1 p_1 - \beta_2 p_2 \\ \text{subject to} & \begin{cases} y_1 + y_2 \leq 6.5 \\ 9y_1 + 5y_2 \leq 44 \\ x_1 = 2.875 + \frac{5}{4}\lambda \\ x_2 = 3.625 - \frac{9}{4}\lambda \\ y_1 = x_1 - p_1 \\ y_2 = x_2 - p_2 \\ 0 \leq \lambda \leq 1.6 \\ x_1, x_2, p_1, p_2 \geq 0 \\ y_1, y_2 \geq 0 \text{ and integer.} \end{cases} \end{array} \tag{16}$$

Where $x = (x_1, x_2)^t$ is a solution in the edge $x = x^* + \lambda d_1$ with d_1 given by the equation (9a), p_1 is the distance between x_1 and y_1, p_2 is the distance between x_2 and y_2, given $p = (p_1, p_2)^t$ we have $y = x - p$ as the integer solution closer to the edge defined by the line $x = x^* + \lambda d_1$. We recommend that the value of $\beta = (\beta_1, \beta_2)$ be proportional to the value of c.

Solving Problem (16), we have the solutions $x = (3.2, 3)^t$ and $y = (3, 3)^t$. The integer solution is given by y and has the objective function value of 39. Figure 4 shows the solution.

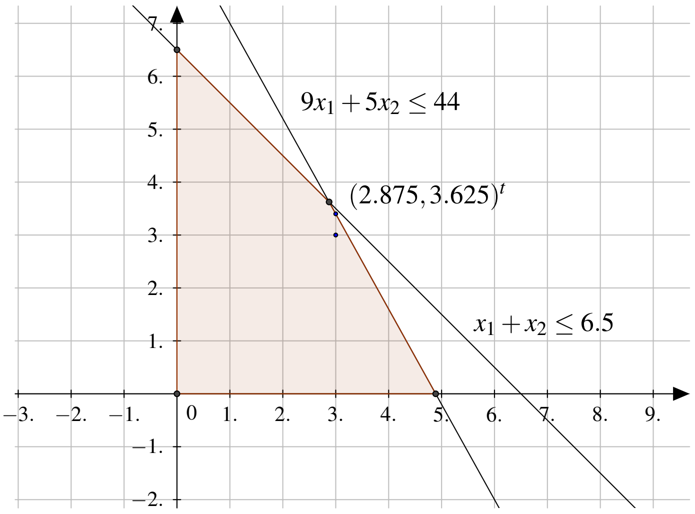

Figure 4 – Graphical Representation of Problem (16)

To find the solution that tends to be closer to the other adjacent edge, we use Problem (17).

$$\begin{array}{ll} \max & z = 8y_1 + 5y_2 - \beta_1 p_1 - \beta_2 p_2 \\ \text{subject to} & \begin{cases} y_1 + y_2 \leq 6.5 \\ 9y_1 + 5y_2 \leq 44 \\ x_1 = 2.875 - \frac{1}{4}\lambda \\ x_2 = 3.625 + \frac{1}{4}\lambda \\ y_1 = x_1 - p_1 \\ y_2 = x_2 - p_2 \\ 0 \leq \lambda \leq 11.5 \\ x_1, x_2, p_1, p_2 \geq 0 \\ y_1, y_2 \geq 0 \text{ and integer.} \end{cases} \end{array} \tag{17}$$

Where $x = (x_1, x_2)^t$ is a solution in the edge $x = x^* + \lambda d_2$ with d_2 given by equation (9b), p_1 is the distance between x_1 and y_1, p_2 is the distance between x_2 and y_2, given $p = (p_1, p_2)^t$ we have $y = x - p$ as the integer solution closer to the edge define by the line $x = x^* + \lambda d_2$.

Solving Problem (17), we find $x = (2.5, 4)^t$ and $y = (2, 4)^t$. The integer solution is given by y and has the objective function value of 36. Figure 5 shows the solution.

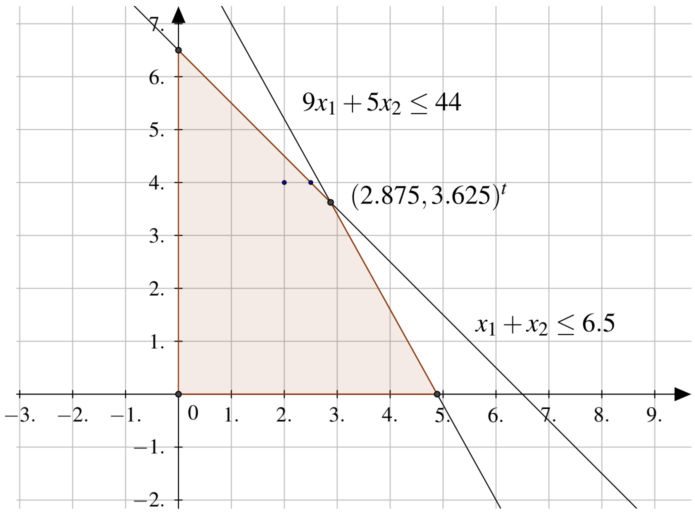

Figure 5 – Graphical Representation of the Problem (17).

To find the best integer solution we compare the solutions of Problems (16) and (17), therefore we have $y = (3, 3)^t$ with the objective function value of 39.

4 RESULTS

To test the strategy, we initially made a thousand instances. Those instances are randomly generated, considering A a matrix $m \times n$, where $1 \leq m \leq 200$ and $200 \leq n \leq 500$. The values are generated by a normal distribution. Each value of the vector c is an integer between 0 and n, each value of the matrix A is an integer between 0 and mn and each value of the vector b is an integer between 1 and $30mn$. We implemented the instances in the solver and compare the results with our strategy, the solver used was the IBM Cplex.

Algorithm 1 finds a solution in 34 instances. For comparison purposes, given z^* the optimal value, we consider solutions with $z \geq 0.7z^*$ as good solutions, therefore, Algorithm 1 finds good solutions in 16 instances. Besides the low convergence rate, every instance problem took one iteration to find the solution or determine infeasibility. Figure 6 shows the solution quality.

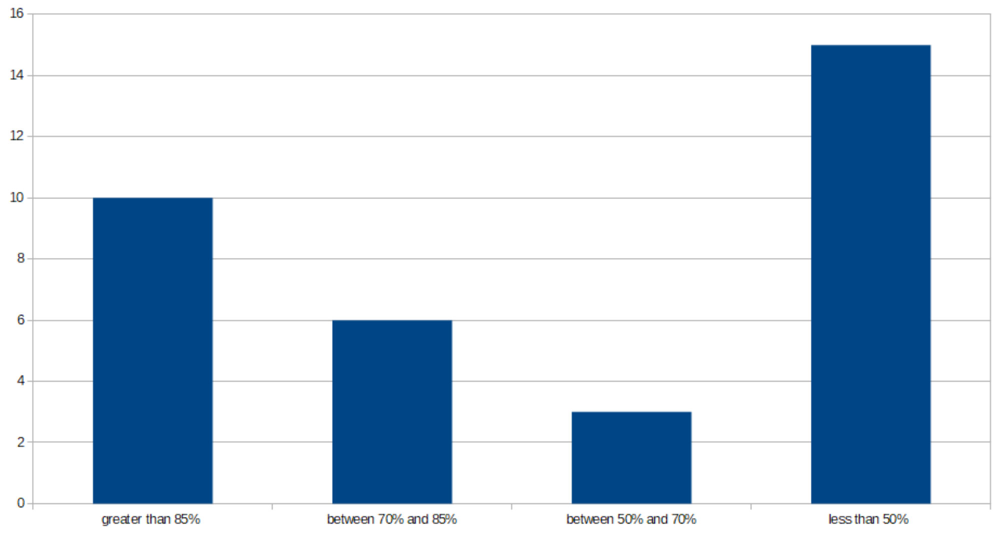

Figure 6 – Quality of the solutions of Algorithm 1.

Algorithm 2 found a solution in all instances, of which 783 are good solutions. The algorithm found a trivial solution in 93 instances. The algorithm took an average of 334 iterations to find the solution while the solver took 1019 iterations, which represents a reduction of 67%. Each MILP took an average of 1.8 iterations to converge, due to the number of MILP problems solved in each instance, the number of iterations reaches 334. It was observed that the instances with the largest number of variables generated the largest reductions in the number of iterations. Table 3 shows some of those instances.

Figure 7 shows the solution quality obtained by Algorithm 2 and Table 2 details the quality of the solutions found by Algorithm 2.

In Table 3 the column Instance shows the number of the instance, m shows the number of constraints, n shows the number of variables, Algorithm 2 Solution shows the objective value obtained by Algorithm 2, Algorithm 2 Iterations shows the total number of iterations performed by the Cplex to solve the $n-m$ MILP problems of Algorithm 2, Cplex Solution shows the objective value obtained by Cplex, Cplex Iterations shows the number of iterations performed by Cplex, Solution Quality shows the quality of the algorithm solution relative to the Cplex solution, for

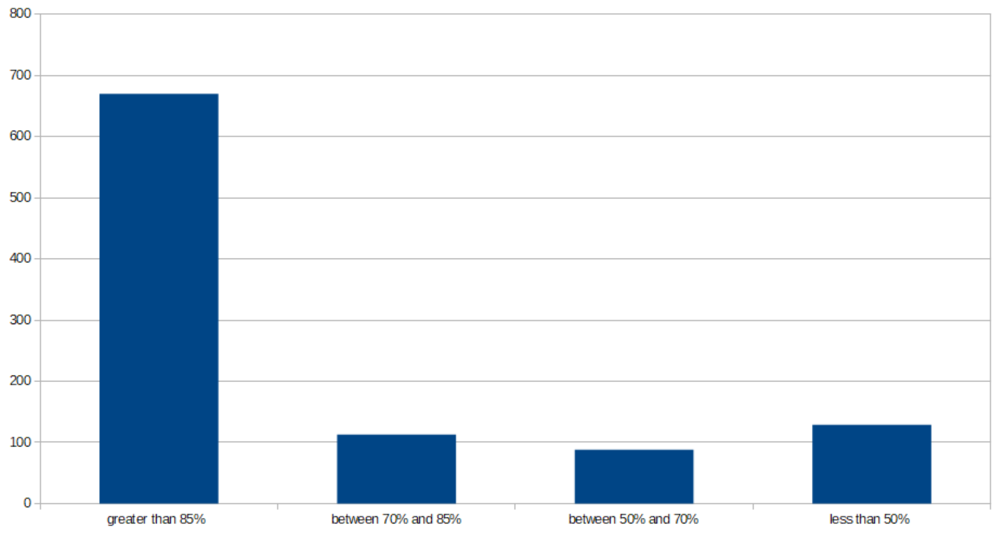

Figure 7 – Quality of the solutions of Algorithm 2.

Table 2 – Quality of the solutions.

Range	**Number of solutions**
Greater than 85%	670
Between 70% and 85%	113
Between 50% and 70%	88
Less than 50%	129

example, instance 149 has a Cplex solution of 1954 and an Algorithm 2 solution of 1163, which represents 60% of the Cplex solution. The column Relative Number of Iterations is the number of iteration performed by Algorithm 2 relative to the number performed by Cplex, for example, Algorithm 2 performed less than 1% of the number of iterations performed by Cplex.

Table 3 – Results of instances.

Instance	m	n	Algorithm 2 Solution	Algorithm 2 Iterations	Cplex Solution	Cplex Iterations	Solution Quality	Relative Number of Iterations
149	136	399	1163	295	1954	35490	60%	<1%
307	17	269	8522	1098	8992	41718	95%	3%
498	90	349	2621	274	3156	42464	83%	<1%
570	45	468	6720	3316	7298	488955	92%	<1%
601	80	409	3099	195	3635	25133	85%	<1%
766	20	398	7737	3411	8086	40257	96%	8%
895	27	381	4220	3664	4748	69798	89%	5%

In most of the instances of Table 3, Algorithm 2 found a good solution. The variation of the solution quality tends to increase according to the value of m. The increase of the value of n, which represents the number of variables, does not show an increasing tendency on the variation of the solution quality. Figure 8 shows the average solution quality by the number of constraints, and Figure 9 shows the average solution quality by the number of variables.

We also tested our strategy on MIPLIB 2017 instances with integer or binary variables, we selected instances that can be solved by solvers such as Cplex or Gurobi. Instead of testing every edge in Algorithm 2, we first try the edge $argmax(c_B - NB' * y)$. If the problem generated by

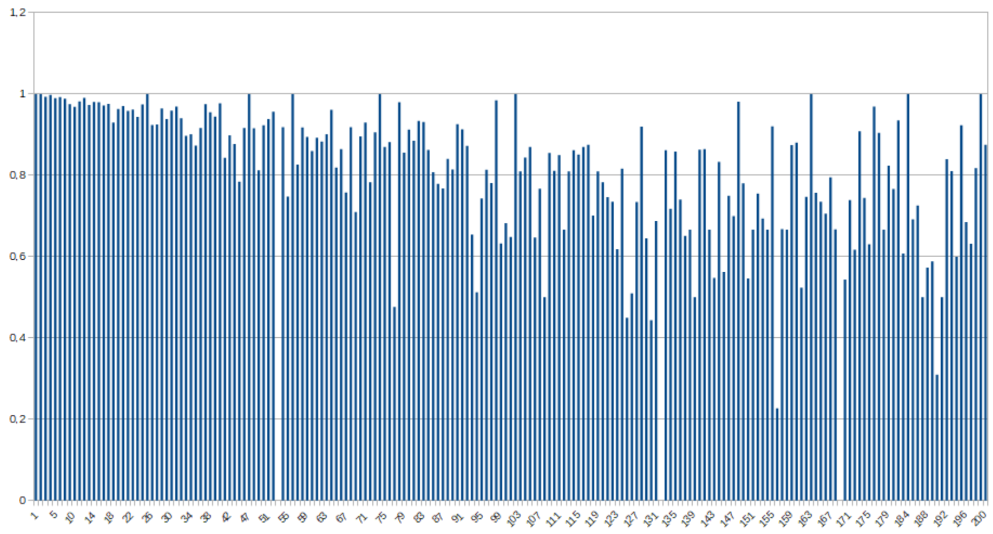

Figure 8 – Average solution quality by the number of constraints.

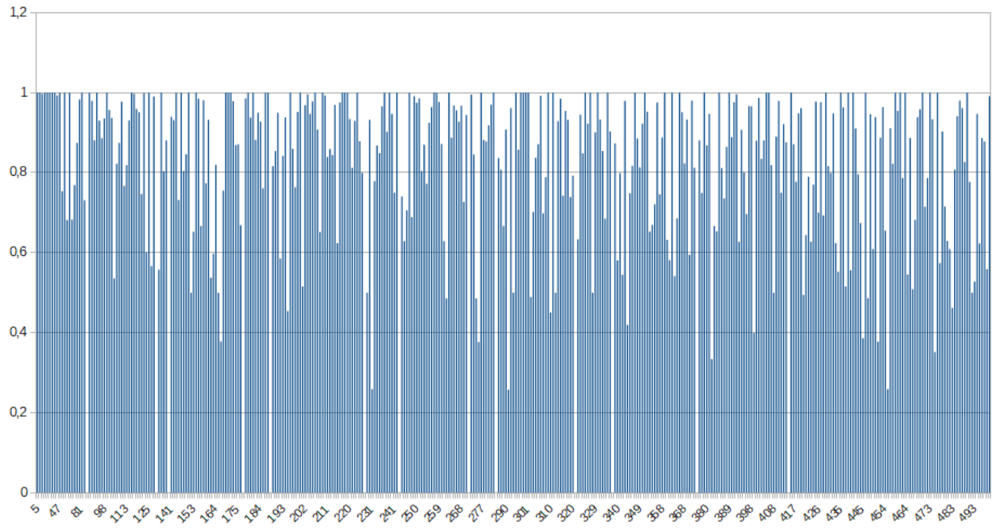

Figure 9 – Average solution quality by the number of variables.

that edge is infeasible, then we try other edges, in order, until a solution is found. Table 4 brings results of MIPLIB 2017 instances, those instances can be found in MIPLIB (2018).

In Table 4 the column Instance shows the name of the instance, Algorithm 2 Solution shows the objective value obtained by Algorithm 2, Algorithm 2 Iterations shows the total number of iterations performed by the Cplex to solve the MILP problems generated by Algorithm 2, Solver Solution shows the objective value obtained by the solver, Solver Iterations shows the number of iterations performed by the solver. The solver used in the tests was the Cplex, except *Chromaticindex32-8*, *Neos-555343*, *neos18*, and *acc-tight5*. In those instances, the Gurobi solver was used.

In all 39 instances of Table 4, the solver performed a total of 13280128 iterations, and Algorithm 2 performed 7125257 iterations, therefore, Algorithm 2 perform 53% of the iterations performed by Cplex. In 28 instances, Algorithm 2 performs less or equal iterations than the solver. In 34 instances, Algorithm 2 obtains a good solution, with a gap lower than 18%, of which 22 instances, the solution of the algorithm is the same solution found by the solver, with a gap lower than 10^-8.

The Cplex had issues solving the MILP generated by Algorithm 2 in the instances *Chromaticindex32-8*, *Neos-555343*, *neos18*, and *acc-tight5* and did not return any solution. Those instances were solved by the Gurobi, the process was stoped due to numerical issues and the results are in Table 4, those solutions have a gap of up to 81%. Both Cplex and Gurobi solvers had issues solving the MILP generated by Algorithm 2 in the instances *mine-166-5*, *mine-90-10*, *qap10*, and *cod105* and did not return any solution.

Different optimal solutions were found by the Cplex and Gurobi solvers for the instances *App2-2* and *Neos-555424*. The Cplex solver found the objective values of 212042.5 and 1331800, respectively, and the Gurobi solver found the objective values of 212047.5 and 1286800, respectively.

5 CONCLUSIONS

Algorithm 1 found a solution in 34 of 1000 randomly generated instances, which suggests that few problems have an integer solution in an adjacent edge of the continuous solution. Besides the results, the algorithm has a low cost and was able to find a solution or determine the infeasibility in one iteration.

Algorithm 2 found a solution for all generated instances, of which 783 are good solutions. On average, the algorithm performed 33% of the iterations performed by the solver. On larger instances, like the ones in Table 3, the number of iterations was between 1% to 5% of the number of iterations taken by Cplex and obtains solutions with objective function values close to the optimum value.

In the MIPLIB instances, Algorithm 2 also performs well, with an average save of 47% on iterations. In 34 of 39 instances, the algorithm found solutions close or equal to the optimal

Table 4 – Results of MIPLIB Instances.

Instance	m	n	Algorithm 2 Solution	Algorithm 2 Iterations	Solver Solution	Solver Iterations
sp98ir	1531	1680	268448539,2	8202	219676790,4	135277
haprp	1048	1828	3673280,7	646	3673280,7	659
enlight_hard	100	200	37	2284	37	2284
App2-2	335	1226	213270,5	2	212042,5	1
App2-1	1038	3283	35109,6	1	19294,3	24
mod010	146	2655	6548	692	6548	907
RococoC10-001000	1293	3117	11460	940551	11460	1156239
f2gap201600	20	1600	76605	1322347	76453	211
f2gap401600	40	1600	82307	742	82307	421
f2gap801600	80	1600	86679	1474	86679	0
Eil33-2	32	4516	934	84096	934	109587
gt2	29	188	21166	124	21166	192
p0201	133	201	7815	1324	7615	545
supportcase14	234	304	288	150	288	157
supportcase16	130	319	288	223	288	223
Chromaticindex32-8	2111	2304	4	17891	4	17891
Neos-1516309	489	4500	36724	173	35954	251
Neos-1599274	1237	4500	32863,6	220	32075,6	186
Neos-831188	2185	4612	2,6	3650003	2,6	1184819
neos-3592146-hawea	995	4870	79435711,5	1205	15465800,7	9531675
Neos-555424	2676	3815	4808400	7760	1331800	25076
neos-3381206-awhea	479	2375	453	76493	453	60701
Neos-555343	3326	3815	5428400	29755	1512800	62177
Neos-555001	3474	3855	4725000	7946	1210625	761
neos-2624317-amur	342	524	3,5	107034	3,5	123595
neos18	11402	3312	16	68053	16	70885
manna81	6480	3321	-13164	3153	-13164	3153
mzzv11	9499	10240	-18980	4639	-21718	15466
enlight8	64	128	27	1996	27	2024
f2gap40400	40	400	20772	312	20772	313
pw-myciel4	8164	1059	10	498133	10	378876
acc-tight5	3052	1339	0	15304	0	0
opm2-z6-s1	15533	1350	-5107	3179	-6202	64407
acc-tight2	2520	1620	0	0	0	0
acc-tight4	3285	1620	0	0	0	0
opm2-z7-s8	31798	2023	-8929	3817	-11242	135078
p2756	755	2756	3640	475	3124	528
harp2	112	2993	-73896921	264854	-73897012	195290
disktom	399	10000	-5000	4	-5000	4
TOTAL				7125257		13280128

solution in fewer iterations. Instances like *sp98ir* and *opm2-z7-s8* Algorithm 2 needs very few iterations to find a good solution, compared with the number of iteration taken by the solver. In 28 of 39 MIPLIB instances, the algorithm performs fewer iterations than the solver.

The usage of the strategy is recommended in instances where conventional methods require many iterations to find a solution. The solution obtained by the strategy can be used as an initial solution for the original problem. With some modifications, the method can be used to solve MILP problems with equality or inequality constraints.

References

ACHTERBERG T & BERTHOLD T. 2007. Improving the feasibility pump. *Discrete Optimization*, **4**(1): 77–86.

ARENALES MN, ARMENTANO VA, MORABITO NETO R & HIDEKI YH. 2007. *Pesquisa operacional*. Elsevier.

BAZARAA MS, JARVIS JJ & SHERALI HD. 2011. *Linear programming and network flows*. John Wiley & Sons.

BERTHOLD T, LODI A & SALVAGNIN D. 2019. Ten years of feasibility pump, and counting. *EURO Journal on Computational Optimization*, **7**(1): 1–14.

BIAGIO MA, COELHO MA & FRANCO PEC. 2012. Heuristic for solving capacitor allocation problems in electric energy radial distribution networks. *Pesquisa Operacional*, **32**(1): 121–138.

CESCHIA S, DI GASPERO L & SCHAERF A. 2017. Solving discrete lot-sizing and scheduling by simulated annealing and mixed integer programming. *Computers & Industrial Engineering*, **114**: 235–243.

DANNA E, ROTHBERG E & LE PAPE C. 2005. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming*, **102**(1): 71–90.

ECKSTEIN J & NEDIAK M. 2007. Pivot, cut, and dive: a heuristic for 0-1 mixed integer programming. *Journal of Heuristics*, **13**(5): 471–503.

EISENBRAND F & WEISMANTEL R. 2019. Proximity results and faster algorithms for integer programming using the Steinitz lemma. *ACM Transactions on Algorithms (TALG)*, **16**(1): 1–14.

FISCHETTI M, GLOVER F & LODI A. 2005. The feasibility pump. *Mathematical Programming*, **104**(1): 91–104.

FISCHETTI M & LODI A. 2003. Local branching. *Mathematical programming*, **98**(1): 23–47.

FISCHETTI M & LODI A. 2010. Heuristics in mixed integer programming. *Wiley Encyclopedia of Operations Research and Management Science*, .

FISHER ML. 2004. The Lagrangian relaxation method for solving integer programming problems. *Management science*, **50**(12_supplement): 1861–1871.

GENOVA K & GULIASHKI V. 2011. Linear integer programming methods and approaches–a survey. *Journal of Cybernetics and Information Technologies*, **11**(1).

LASKARI EC, PARSOPOULOS KE & VRAHATIS MN. 2002. Particle swarm optimization for integer programming. In: *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, vol. 2. pp. 1582–1587. IEEE.

MIPLIB. 2018. MIPLIB 2017. Available at: http://miplib.zib.de.

NEMHAUSER GL & WOLSEY LA. 1988. Integer programming and combinatorial optimization. *Wiley, Chichester. GL Nemhauser, MWP Savelsbergh, GS Sigismondi (1992). Constraint Classification for Mixed Integer Programming Formulations. COAL Bulletin*, **20**: 8–12.

PARK YM & KIM KH. 2005. A scheduling method for berth and quay cranes. In: *Container terminals and automated transport systems*. pp. 159–181. Springer.

RICHARD JPP, DE FARIAS IR & NEMHAUSER GL. 2003. A simplex-based algorithm for 0-1 mixed integer programming. In: *Combinatorial Optimization—Eureka, You Shrink!*. pp. 158–170. Springer.

ROTHBERG E. 2007. An evolutionary algorithm for polishing mixed integer programming solutions. *INFORMS Journal on Computing*, **19**(4): 534–541.

WOLSEY LA. 2007. *Mixed integer programming*. Wiley Online Library. 1–10 pp.

YUAN G & WEI Z. 2009. New line search methods for unconstrained optimization. *Journal of the Korean Statistical Society*, **38**(1): 29–39.

How to cite

CERESER BLH, OLIVEIRA ARL & MORETTI AC. 2022. A new strategy to solve linear integer problems with simplex directions. *Pesquisa Operacional*, **42**: e255132. doi: 10.1590/0101-7438.2022.042.00255132.