Acta Cybernetica **19** (2009) 135–146.

Determining Initial Bound by "Ray-method" in Branch and Bound Procedure

Anett Rácz*

Abstract

In this paper we present an algorithm for determining initial bound for the Branch and Bound (B&B) method. The idea of this algorithm is based on the use of "ray" as introduced in the "ray-method" developed for solving integer linear programming problems [11], [12]. Instead of solving an integer programming problem we use the main idea of the ray-method to find an integer feasible solution of an integer linear programming problem along the ray as close to an optimal solution of the relaxation problem as possible. The objective value obtained in this manner may be used as an initial bound for the B&B method. It is well known that getting a "good bound" as soon as possible can often significantly increase the performance of the B&B method.

Keywords: integer programming, branch and bound, ray-method, initial bound

1 Introduction

It is well known that the performance of the B&B method mainly depends on the following three main factors:

- the rule used to choose the "branching" variable,
- the strategy used for generating binary search tree and
- the value of the initial bound.

Generally speaking, while the branching variable and strategy determine the size of the binary tree to be generated, getting a "good" bound as soon as possible can dramatically reduce the size of the tree to be considered, since the bound is used to prune those parts of the tree where the value of the objective function cannot be better than the bound.

Numerous efforts have been made in the past decades to investigate the general properties and behavior of the B&B method, e.g. [3, 6, 7, 13, 14, 16, 17, 18], to improve its computational efficiency, e.g. [4, 8, 9, 10, 15], to maximize its performance in different computational environments, see for example [5, 19], etc.[1]

*Department of Applied Mathematics and Probability Theory, Faculty of Informatics, University of Debrecen, 4032 Debrecen Egyetem tér 1., Hungary, E-mail: anett.racz@inf.unideb.hu

[1]The list of the relevant literature is so long, that in the framework of the current paper there is not enough space even to begin to cover all of the relevant items.

DOI: 10.14232/actacyb.19.1.2009.9

However, there is still some research being done in order to develop alternative algorithms for problems involving discrete variables. One of such investigations was performed in the Computer Center of the Russian Academy of Sciences [11, 12]. Schematically, the method developed by Khachaturov et al. is based on the search of "better" feasible integer solutions in some special set along some direction called a "ray". In this paper we present a new procedure for determining an initial bound for the branch and bound method which uses the basic ideas of the ray-method described in [11, 12].

The paper is organized as follows. In Section 2 we briefly overview the original ray-method – its mathematical background, its general scheme, different ways to define the ray, and implementation issues. In Section 3 we explain the main idea of the algorithm proposed: here we define the ray, then describe the main steps of the procedure, and finally, using a small illustrative numerical example, we show how this algorithm may be utilized. Section 4 summarizes my conclusions and my plans for future research.

2 The original *"Ray-method"*

Originally, the method was developed for a non-linear integer programming problem of the following form:

$$f(x) \longrightarrow \min \tag{1}$$

st.

$$x \in S \subset R^n, \tag{2}$$

$$x = (x_1, x_2, \ldots, x_n), \quad x_j - \text{ integer}, \ j = 1, 2, \ldots, n, \tag{3}$$

where feasible set

$$S = \{x \in R^n \mid g_i(x) \leq b_i, \ i = 1, 2, \ldots, m\}$$

is a convex, bounded and non-empty set, objective function $f(x)$ is non-linear differentiable $\forall x \in S$ and bounded from below on S, functions $g_i(x), \ i = 1, 2, \ldots, m$, are non-linear and differentiable.

2.1 The mathematical background

Definition 1. *Let be given a feasible integer point $x^0 \in R^n$. We will say that integer point x' belongs to neighborhood set $O(x^0)$, i.e. $x' \in O(x^0)$ if values*

$$\left|x_1^0 - x_1'\right|, \quad \left|x_2^0 - x_2'\right|, \quad \ldots, \quad \left|x_n^0 - x_n'\right|$$

are relative primes.

Note that $O(x^0)$ has the following obvious properties.

Property 1. $x^0 \notin O(x^0)$.

Property 2. *If point $x' \in O(x^0)$, then on the straight line open segment (x^0, x') there is no integer point, i.e. there is no integer point x such that*

$$x \;=\; x^0 + \lambda(x' - x^0), \quad 0 < \lambda < 1\,.$$

Let $S(x^0)$ denote the subset of feasible set S, where $f(x)$ is strictly less than $f(x^0)$, i.e.

$$S(x^0) \;=\; \{x \in S \mid f(x) < f(x^0)\}.$$

Using this notation we can formulate the following optimization criteria.

Theorem 1. *Feasible integer point x^0 is an optimal solution for problem (1)-(3) if and only if*

$$O(x^0) \;\cap\; S(x^0) \;=\; \emptyset \tag{4}$$

Proof: see [11], [12].

2.2 The general scheme

Let x^0 be an integer feasible solution for problem (1)-(3). In accordance with the main idea of the method we have to find such an integer point $x' \in O(x^0)$ that $x' \in O(x^0) \cap S(x^0)$. If $O(x^0) \cap S(x^0) = \emptyset$, then x^0 is an optimal solution for problem (1)-(3). The problem is solved.

Otherwise, i.e. if $O(x^0) \cap S(x^0) \neq \emptyset$, then we solve the following one-variable minimization problem:

$$f(\lambda) \;=\; f(x^0 + \lambda(x' - x^0)) \longrightarrow \min \tag{5}$$

$$x^0 + \lambda(x' - x^0) \in S \tag{6}$$

$$\lambda \geq 0. \tag{7}$$

Since both points x^0 and x' belong to the convex bounded non-empty feasible set S, constraint (6) defines the segment of straight line $x^0 + \lambda(x' - x^0)$, which belongs to S. So constraints (6) and (7) determine a non-empty bounded subset of S. Since original objective function $f(x)$ is continuous $\forall x \in S$ and bounded from below, it means that function $f(\lambda)$ is continuous and bounded too on any subset of S. The latter means that problem (5)-(7) is solvable and may be solved by any suitable numerical method. Let λ_{min} be its optimal solution and $[\lambda_{min}]$ denote its integer part. Concerning value $[\lambda_{min}] + 1$ it may occur that

$$x^0 + ([\lambda_{min}] + 1)(x' - x^0) \notin S, \tag{8}$$

or

$$f(x^0 + ([\lambda_{min}] + 1)(x' - x^0)) \geq f(x^0 + [\lambda_{min}](x' - x^0))\,. \tag{9}$$

Now we construct the following point:

$$x'' \;=\; \begin{cases} x^0 + [\lambda_{min}](x' - x^0), & \text{if (8) or (9) takes place,} \\ x^0 + ([\lambda_{min}] + 1)(x' - x^0), & \text{otherwise.} \end{cases}$$

In other words, first, solving problem (5)-(7) we search the minimal value of function (5) on the ray beginning from point x^0 and passing through point x'. Then on this ray we have to find an integer feasible point x'' for which $f(x'')$ is most close to value $f(\lambda_{min})$. In the next step we denote point x'' by x^0 and repeat the process. The new set $S(x^0)$ differs from the previous one by only one constraint $f(x) <= f(x^0) = f(x'')$.

Since the number of integer points in feasible set S is bounded, the process will terminate in a finite number of iterations.

2.3 Different rules to define the ray

Let point x^0 be an integer feasible solution for problem (1)-(3), i.e. $x^0 \in S$. We will say that $L = \{x \in R^n \,|\, x = x^0 + \lambda l,\, \lambda \geq 0\}$, where $l = (l_1, l_2, \ldots, l_n) \in R^n$, is a ray, if there is $\lambda' > 0$ such that $f(x^0 + \lambda' l) < f(x^0)$. Using this notation, we describe here the following three ways by [12] for constructing a ray.

Procedure 1: Let x^* be the optimal solution of the relaxation problem (i.e. the problem without the integrality constraints) (1)-(2). Define ray L as

$$L \;=\; \{x \in R^n \,|\, x = x^0 + \lambda\,(x^* - x^0),\; \lambda \geq 0\}. \tag{10}$$

Procedure 2: Calculate the gradients $\nabla g_i(x)$ at the point x^0 for all $g_i(x),\; i = 1, 2, \ldots, m,$ functions, and introduce rays $L_i,\; i = 1, 2, \ldots, m,$ in the following way:

$$L_i \;=\; x^0 + \lambda\,\nabla g_i(x^0), \quad \lambda \geq 0,$$

if exists such $\lambda' > 0$, that $f(x^0 + \lambda'\,\nabla g_i(x^0)) < f(x^0)$. And

$$L_i \;=\; x^0 - \lambda\,\nabla g_i(x^0), \quad \lambda \geq 0,$$

otherwise.

Procedure 3: Choose the following formula for the ray.

$$L \;=\; x^0 - \lambda\,\nabla f(x^0), \quad \lambda \geq 0. \tag{11}$$

2.4 Implementation issues

In contrast to the transparency of the theoretical background for the method there are serious difficulties with its implementation and computational efficiency. The main and most serious of them is checking optimality criteria for a given feasible integer point x^0 since set $O(x^0)$ may contain a huge number of integer points. Note that, as was mentioned above, these integer points were selected on the basis of the usage of relative prime numbers, so determining these integer points may be a very hard and computationally very expensive problem. This is why the developers of the method for numerical experiments used different approximate variants of the method and tested it using problems of relatively small size (up to 100 constraints $\times$ 120 variables).

3 The new method proposed

As was mentioned above, the ray-method was originally developed as a method for solving non-linear integer programming problems. Using the main ideas of the method, below we propose a new algorithm which may be used for determining the initial bound in the branch and bound method when solving integer linear programming problems.

3.1 Preliminaries

Consider the following pure integer linear programming minimization problem:

$$f(x) = \sum_{j=1}^{n} c_j x_j \longrightarrow \min \tag{12}$$

st.

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1, 2, \ldots, m, \tag{13}$$

$$x_j \geq 0, \quad \text{integer}, \quad j = 1, 2, \ldots, n. \tag{14}$$

Here and in what follows we assume that relaxation problem (12)-(14) is solvable (i.e. has a non-empty feasible set and objective function $f(x)$ over the feasible set has a finite lower bound) and vector

$$x^{min} = (x_1^{min}, x_2^{min}, \ldots, x_n^{min})$$

is its relaxation non-integer solution. Furthermore, we suppose that the corresponding maximization relaxation problem is solvable too, and

$$x^{max} = (x_1^{max}, x_2^{max}, \ldots, x_n^{max})$$

is its optimal solution. These two assumptions play a very important role in our algorithm since we use points x^{min} and x^{max} to determine the ray for indicating the direction of the search. Moreover we assume that $x^{min} \neq x^{max}$.

3.2 Main steps

Using the given notation, the algorithm proposed may be described in the following steps.

0. Initial point: Let us denote point x^{min} by x^0, and $l = (l_1, l_2, \ldots, l_n)$, where $l_j = x_j^{max} - x_j^{min}$, $j = 1 \ldots n$.

1. Ray: Define the ray in the following way:

$$L = x^0 + \lambda(x^{max} - x^0), \quad 0 \leq \lambda \leq 1 \, .$$

Note that since feasible set S is convex, it means that all points of L are elements of set S.

2. Constructing set $O(x^0)$: Let J^0 be a set of indexes of integer components of x^0, i.e. $J^0 = \{j \in J |\ x_j^0 = [x_j^0]\}$, where $J = \{1, 2, \ldots, n\}$. We define set $O(x^0)$ as the set of such points x which satisfy the following constraints:

$$\left.\begin{array}{lcll} [x_j^0] & \leq x_j \leq & [x_j^0]+1, & \text{if } j \notin J^0\ , \\ [x_j^0] & \leq x_j \leq & [x_j^0]+1, & \text{if } j \in J^0 \text{ and } l_j > 0\ , \\ [x_j^0]-1 & \leq x_j \leq & [x_j^0], & \text{if } j \in J^0 \text{ and } l_j < 0\ , \\ & x_j = & [x_j^0], & \text{if } j \in J^0 \text{ and } \ l_j = 0\ . \end{array}\right\} \tag{15}$$

Generally speaking $O(x^0)$ is the unit-cube containing the point x^0. If $J^0 = \emptyset$, then the dimension of this unit-cube is n.

Before starting the iterations let us define the point $x' := x^0$ and calculate the first perforation point $P_{act} = (p_1, p_2, \ldots, p_n)$ solving the following optimization problem:

$$\lambda \longrightarrow \max \tag{16}$$

st.

$$p_j = x_j^0 + \lambda l_j \quad j = 1, 2, \ldots, n, \tag{17}$$

$$\left.\begin{array}{lcll} [x_j^0] & \leq p_j \leq & [x_j^0]+1, & j \notin J^0\ , \\ [x_j^0] & \leq p_j \leq & [x_j^0]+1, & j \in J^0 \text{ and } l_j > 0\ , \\ [x_j^0]-1 & \leq p_j \leq & [x_j^0], & j \in J^0 \text{ and } l_j < 0\ , \\ & p_j = & [x_j^0], & j \in J^0 \text{ and } l_j = 0\ . \end{array}\right\} \tag{18}$$

3. Shifting: Now we enter new variables $y_j = x_j - [x'_j],\ j = 1, 2, \ldots, n,$ and construct new feasible set S' in the following way

$$S' : \qquad \sum_{j=1}^{n} a_{ij} y_j \leq b'_i,\ \ i = 1, 2, \ldots, m\ ,$$

where

$$b'_i \ = \ b_i - \sum_{j=1}^{n} a_{ij} [x'_j],\ \ i = 1, 2, \ldots, m\ .$$

Obviously, set S' is the intersection of the current set $O(x')$ and feasible set S shifted to point 0. Then we solve the following 0-1 LP problem

$$f'(y) = \sum_{j=1}^{n} c_j y_j + c_0 \to \max \tag{19}$$

st.

$$\sum_{j=1}^{n} a_{ij} y_j \leq b'_i,\ \ i = 1, 2, \ldots, m, \tag{20}$$

$$y_j = 0/1, \quad j = 1, 2, \ldots, n, \tag{21}$$

where

$$c_0 = \sum_{j=1}^{n} c_j [x'_j] \ ,$$

using **Balas'** additive algorithm (implicit enumeration) [2]. If problem (19)-(21) has 0-1 optimal solution y^*, then we determine vector $x^* = (x_1^*, x_2^*, \ldots, x_n^*)$, where

$$x_j^* = y_j^* + [x'_j], \ \ j = 1, 2, \ldots, n \ ,$$

and use value $f(x^*) = f'(y^*)$ as an initial bound for the branch and bound method. **Stop**.
Otherwise,

4. Perforation point: We determine point P_{next} where the ray "perforates" the hull of the next unit-cube along the ray solving the following optimization problem:

$$\lambda \longrightarrow \max \tag{22}$$

st.

$$x_j = x_j^0 + \lambda l_j \quad j = 1, 2, \ldots, n \ , \tag{23}$$

$$\left.\begin{array}{lcll} [p_j] & \leq x_j \leq & [p_j] + 1, & j \notin J' \ , \\ [p_j] & \leq x_j \leq & [p_j] + 1, & j \in J' \text{ and } l_j > 0 \ , \\ [p_j] - 1 & \leq x_j \leq & [p_j], & j \in J' \text{ and } l_j < 0 \ , \\ & x_j = & [p_j], & j \in J' \text{ and } l_j = 0 \ , \end{array}\right\} \tag{24}$$

where $J' = \{j \in J |\ p_j = [p_j]\}$. Here constraints (23) and (24) provide $P_{next} \in L$ and $P_{next} \in O(P_{act})$, correspondingly. Obviously, this problem is solvable, i.e. has a non-empty feasible set and its objective function is bounded from above. Let $P_{next} = (p'_1, p'_2, \ldots, p'_n)$ solve problem (22)-(24) and λ' be the maximal value of objective function (22).

Let us define middle point x' of the section $[P_{act};\ P_{next}]$:

$$x'_j = \frac{p_j + p'_j}{2} \ \ j = 1, 2, \ldots, n \ . \tag{25}$$

Furthermore, we do not need point P_{act} any more, so we overwrite it with the value of P_{next}, i.e. $P_{act} := P_{next}$.

5. Next unit-cube: Having point x' we can determine the next unit-cube along the ray using the following rule:

$$\left.\begin{array}{lcll} [x'_j] & \leq x_j \leq & [x'_j] + 1, & \text{if } j \notin J' \ , \\ [x'_j] & \leq x_j \leq & [x'_j] + 1, & \text{if } j \in J' \text{ and } l_j > 0 \ , \\ [x'_j] - 1 & \leq x_j \leq & [x'_j], & \text{if } j \in J' \text{ and } l_j < 0 \ , \\ & x_j = & [x'_j], & \text{if } j \in J' \text{ and } \ l_j = 0 \ , \end{array}\right\} \tag{26}$$

where $J' = \{j \in J |\ x'_j = [x'_j]\}$. Go to step 3.

Since the number of unit-cubes "perforated" by the ray is finite, the process will terminate in a finite number of iterations. It may occur that when determining next perforation point x' we obtain $\lambda' > 1$. It means that unit-cubes constructed along the ray do not contain any integer feasible point for original problem (12)-(14). It means the method fails.

3.3 An illustrative numerical example

Here the main steps of the method proposed using a small numerical example are illustrated. The method proposed was partially implemented in the frame of the educational linear and linear-fractional package WinGULF [1]. The package has numerous options for the B&B method - we can choose the direction of the search (first left node and then right one or vice versa), different rules for selecting a branching variable (for example, "fractional part most close to 0.5", "smallest fractional part", "biggest fractional part", "smallest value", "biggest value", etc.), user defined initial bound, etc. When testing the method proposed, most of the options built in were used.

Consider the following numerical example:

$$\begin{array}{rcrrrcr} f(x) & = & 20x_1+ & 21x_2+ & 18x_3 & \longrightarrow & \min \\ \text{st.} & & & & & & \\ & & 9x_1+ & 1.5x_2+ & 7x_3 & \leq & 1350\,, \\ & & 5.5x_1+ & 1x_2+ & 9x_3 & \geq & 1250\,, \\ & & -4.5x_1- & 10x_2+ & 2.5x_3 & \leq & -1050\,, \\ & & x_1,\ x_2,\ x_3 & - & \text{integer.} & & \end{array}$$

Solving both (minimization and maximization) relaxation problems we obtain the following:

$$\begin{array}{rcrrr} x^{min} & = & (\ 65.042, & 97.799, & 88.274)\,, \\ x^{max} & = & (\ 0.000, & 523.076, & 80.769)\,, \\ L & = & (\ -65.042, & 425.277, & -7.504)\,. \end{array}$$

Let us denote x^{min} with x^0 and construct set $O(x^0)$, i.e. the following unit-cube:

$$O(x^0): \begin{cases} 65 \leq x_1 \leq 66\,, \\ 97 \leq x_2 \leq 98\,, \\ 88 \leq x_3 \leq 89\,. \end{cases}$$

So we can construct the first shifted problem:

$$\begin{array}{rcrrrcrl} f'(y) & = & 20y_1+ & 21y_2+ & 18y_3 & + & 4921 & \longrightarrow \max \\ \text{st.} & & & & & & & \\ & & 9y_1+ & 1.5y_2+ & 7y_3 & \leq & 3.5\,, & \\ & & 5.5y_1+ & 1y_2+ & 9y_3 & \geq & 3.5\,, & \\ & & -4.5y_1- & 10y_2+ & 2.5y_3 & \leq & -7.5\,, & \end{array}$$

$$y_1,\ y_2,\ y_3\ -\ 0/1$$

and try to solve it. Since the problem is infeasible, we have to determine the next unit-cube along the ray. In order to obtain the next unit-cube first we solve the following problem (see (23)-(24)):

$$\lambda \to \max$$

st.

$$\left.\begin{array}{l} x_1 = 65.042 + \lambda(-65.042)\,, \\ x_2 = 97.799 + \lambda(425.277)\,, \\ x_3 = 88.274 + \lambda(-7.504)\,, \end{array}\right\}$$

$$\left.\begin{array}{rcl} 65 \leq & x_1\,, & \\ & x_2 & \leq 98\,, \\ 88 \leq & x_3 & \end{array}\right\}$$

and obtain the first perforation point P_1:

$$\lambda' \ = \ 0.00047, \quad P_1 \ = \ (65.011,\ 98,\ 88.270)\,.$$

To find the next perforation point P_2 we have to solve the following problem:

$$\lambda \to \max$$

st.

$$\left.\begin{array}{l} x_1 = 65.042 + \lambda(-65.042)\,, \\ x_2 = 97.799 + \lambda(425.277)\,, \\ x_3 = 88.274 + \lambda(-7.504)\,, \end{array}\right\}$$

$$\left.\begin{array}{rcl} 65 \leq & x_1\,, & \\ & x_2 & \leq 99\,, \\ 88 \leq & x_3\,, & \end{array}\right\}$$

so we obtain

$$\lambda'' \ = \ 0.00064, \quad P_2 \ = \ (65,\ 98.07,\ 88.26)\,.$$

Using these points P_1 and P_2 we find the middle point

$$x' = (65.006,\ 98.03,\ 88.26)\ .$$

This point allows us to construct the next shifted problem:

$$\begin{array}{rcrrrcrl} f'(y) & = & 20y_1+ & 21y_2+ & 18y_3 & + & 4942 & \longrightarrow \max \\ \text{st.} & & & & & & & \\ & & 9y_1+ & 1.5y_2+ & 7y_3 & \leq & 2\ , & \\ & & 5.5y_1+ & 1y_2+ & 9y_3 & \geq & 2.5\ , & \\ & & -4.5y_1- & 10y_2+ & 2.5y_3 & \leq & -2.5\ , & \end{array}$$

$$y_1,\ y_2,\ y_3\ -\ 0/1\ .$$

This problem has no feasible solution.

Proceeding to the next perforation point P_3, we obtain the following optimization problem to solve

$$\lambda \to \max$$

st.

$$\left.\begin{array}{l} x_1 = 65.042 + \lambda(-65.042)\ , \\ x_2 = 97.799 + \lambda(425.277)\ , \\ x_3 = 88.274 + \lambda(-7.504)\ , \end{array}\right\}$$

$$\left.\begin{array}{rrl} 64 \leq & x_1\ , & \\ & x_2 & \leq 99\ , \\ 88 \leq & x_3\ . & \end{array}\right\}$$

We obtain $\lambda = 0.0028$ and the next perforation point $P_3 = (64.85,\ 99,\ 88.25)$. Therefore the next middle point is $x' = (64.93,\ 98.53,\ 88.26)$ and the shifted problem is as follows:

$$\begin{array}{rcrrrcrl} f'(y) & = & 20y_1+ & 21y_2+ & 18y_3 & + & 4922 & \longrightarrow \max \\ \text{st.} & & & & & & & \\ & & 9y_1+ & 1.5y_2+ & 7y_3 & \leq & 11\ , & \\ & & 5.5y_1+ & 1y_2+ & 9y_3 & \geq & 8\ , & \\ & & -4.5y_1- & 10y_2+ & 2.5y_3 & \leq & -2\ , & \end{array}$$

$$y_1,\ y_2,\ y_3\ -\ 0/1\ .$$

The optimal solution of this problem is $y^* = (0,\ 1,\ 1)$ and $f'(y^*) = 4961$. This value may be used as an initial bound. The corresponding integer point is $x^* = (64,\ 99,\ 89)$.

Note that the initial bound obtained (4961) is very close to the optimal value (after solving the problem we obtain 4959). Below is presented a table with results

obtained from WinGULF after running B&B method on this numerical example using different strategies (from left to right and vice versa) and different branching rules. *"Wo.I.B."* means without an initial bound and *"W.I.B."* means with an initial bound.

Branching variable	Left → Right		Right → Left	
	Wo.I.B.	W.I.B.	Wo.I.B.	W.I.B.
Minimal index	291	37	37	33
Maximal index	31	23	243	23
Max. value	33	25	245	25
Min. value	31	25	25	25
Max. fract. part	105	41	37	37
Min. fract. part	31	23	23	23
Most close to 0.5	31	25	25	25

4 Further work and ideas to improve efficiency

Some simple manual tests were performed. These preliminary tests show that it could be worth making more efforts in the topic and in performing computerized numerical tests with different LP problems including MIPLIB and other test collections. It is clear that test results depend very hard on the search strategy and branching rules used when running B&B. This is why it would be worth developing a special software application, which could be used when comparing the efficiency of the ray-method in different cases. Moreover, we would like to improve the ray-method too. One of the possible directions for further research may be an investigation connected with extending the main ideas of the method to the case of mixed integer programming problems. Another open question is ray-selection in the case when the maximization problem is unbounded.

References

[1] Bajalinov, E., *"Linear-fractional programming: theory, methods, software and applications"*, Kluwer, 2003.

[2] Balas, E., *"An additive algorithm for solving linear programs with zero-one variables"*, Operations Research, vol. 13, pp. 517-46, 1965.

[3] Berliner, H., *"The B tree search algorithm: A best-first proof procedure"*, Artificial Intell., vol. 12, no. 1, pp. 23-40, 1979.

[4] Borchers, B., Mitchell, J.E., *"An improved branch and bound algorithm for mixed integer nonlinear programs"*, Computers and Operations Research, vol. 21, issue 4, pp. 359-367, 1994.

[5] Gendron, B., Crainic, T.G., *"Parallel Branch-and-Bound Algorithms: Survey and Synthesis."*, Opererations Research, vol. 42 issue 6, pp. 1042-1066, 1994.

[6] Gupta, O. K., Ravindran, V., *"Branch and Bound Experiments in Convex Nonlinear Integer Programming"*, Management Science, vol. 31, no. 12, 1533-1546, 1985.

[7] Hawkins, D. M., *"Branch-and-Bound method"*, Encyclopedia of Statistical Sciences, John Wiley and Sons, 2006.

[8] Ibaraki, T., *"Computational efficiency of approximate branch-and-bound algorithms"*, Math. Oper. Res., vol. 1, no. 3, pp. 287-298, 1976.

[9] Ibaraki, T., *"Theoretical comparisons of search strategies in branch-and-bound algorithms"*, Int. J. Computer and Information Sciences, vol. 5, no. 4, pp. 315-343, 1976.

[10] Ibaraki, T., *"The Power of Dominance Relations in Branch-and-Bound Algorithms"*, Journal of the ACM (JACM), vol. 24, issue 2, pp. 264 - 279, 1977.

[11] Khachaturov, V.R., Mirzoyan, N.A. *"Solving problems of integer programming with ray-method."* Notes on applied mathematics, Computer Center of Soviet Academy of Science, 1987.

[12] Khachaturov, V.R., *"Combinatorial methods and algorithms for solving large-scale discrete optimization problems"*, Moscow, Nauka, 2000.

[13] Kumar, V., Kanal, L. N., *"A general branch and bound formulation for understanding and synthesizing and/or tree search procedures"*, Artificial Intell., vol. 21, no. 1-2, pp. 179-198, 1983.

[14] Lawler, E. L., Wood, D. E., *"Branch-And-Bound Methods: A Survey"*, Operations Research, vol. 14, no. 4, pp. 699-719, 1966.

[15] Linderoth, T. Savelsbergh, M. W. P., *"A computational study of branch and bound search strategies for mixed integer programming"*, INFORMS J. Computing, vol. 11, no. 2, 173-187, 1999.

[16] Mitten, L., *"Branch and bound methods: General formulation and properties"*, Operation Research, vol. 18, pp. 24-34, 1970.

[17] Smith, D.R., *"Random trees and the analysis of branch and bound procedures"*, Journal of the Association for computing machinery, vol.31, no.1, pp.163-188, 1984.

[18] Yu, C.-F., Wah, B.W., *"Stochastic modeling of branch-and-bound algorithms with best-first search"*, IEEE Transactions on Software Engineering, vol. SE-11, no. 9, pp. 922-934, 1985.

[19] Yu, C.-F., Wah, B.W., *"Efficient Branch-and-Bound Algorithms on a Two-Level Memory System"*, IEEE Transactions on Software Engineering, vol. 14, no. 9, pp. 1342-1356, 1988.