THE RAY-METHOD: THEORETICAL BACKGROUND AND COMPUTATIONAL RESULTS

Erik Bajalinov
University College of Nyíregyháza, Institute of Mathematics and Informatics
H-4400 Nyíregyháza, Sóstói út 31/b, Hungary
E-mail: Bajalinov@NyF.Hu

Anett Rácz
Debrecen University, Faculty of Informatics
4028 Debrecen, Kassai u. 26, Hungary
E-mail: Racz.Anett@Inf.UniDeb.Hu

Abstract

In our talk we present an algorithm for determining initial bound for the Branch and Bound (B&B) method. The idea of the algorithm is based on the use of the "ray" introduced in the "ray-method" developed for solving integer programming problems [13], [14]. Instead of solving a common integer programming problem we use the main idea of the ray-method to find an integer feasible solution of integer linear programming (ILP) problem along the ray as close to optimal solution of relaxation problem, as possible. Objective value obtained in this way may be used as an initial bound for B&B method. The algorithm has been implemented in the frame of educational package WinGULF [3] for linear and linear-fractional programming and has been tested on various ILP problems. Then inspired by the results obtained we implemented the method using the so-called callable library of CPLEX package by IBM. Computational experiments with the algorithm proposed show that such preprocessing procedure in many cases results an integer feasible solution very close to the solution of relaxation problem. Initial bound for branch and bound method determined in this way often can significantly decrease the size of the binary tree to be searched and in this manner can improve performance of the B&B method.

Key words: *Branch and bound, Integer programming*

1. INTRODUCTION

The Branch and Bound (B&B) algorithm, first proposed by Land and Doig [16], is a well known and efficient algorithm for solving Integer Programming (IP) or Mixed Integer Programming (MIP) problems, it is used by all commercial solvers.

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

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

Numerous efforts have been made in past decades to investigate general properties and behavior of B&B method, e.g. [5], [8], [9], [15], [17], [18], [19], [20], to improve its computational efficiency, e.g. [6], [10], [11], [12], to maximize its performance in different computational environments, see for example [7], [21], etc.

Generally speaking, while branching variable and searching strategy determine the size of 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 objective function cannot be better than the current bound. Our aim was to construct such a general algorithm which could provide a bound before we start the tree-building.

Below we describe a procedure which can define a feasible integer solution aspiring to the best possible objective value using minimal computational effort and time.

2. RAY METHOD

Let us consider the following integer linear programming problem (ILP):

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

st.

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

$$x_j \ge 0,\ \text{integer},\ j=1,2,\ldots,n \tag{3}$$

Here and in what follows we assume that the corresponding relaxation problem of (1)-(3) is solvable (i.e. has a non-empty feasible set *S* 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 vector

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

is its optimal solution.

These two assumptions play a very important role in the new algorithm we proposed since points x^{min} and x^{max} are used to determine the ray for indicating the direction of the search. Moreover we assume that

$$x^{min} \neq x^{max}$$

2.1. Main steps

Using notation given above, the new algorithm proposed can be described in the following way.

1. Initial point. Let us denote point x^{min} by x^0, and then let us enter vector $l = (l_1, l_2, \dots, l_n)$, where

$$l_j = x_j^{max} - x_j^{min}, \quad j = 1, 2, \dots, n.$$

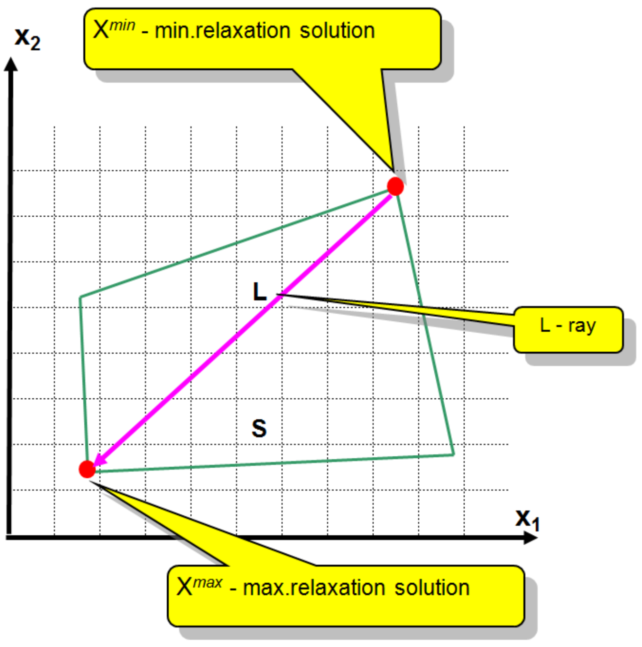
X^min - min.relaxation solution
x2
L
L - ray
S
x1
X^max - max.relaxation solution

Figure 1: The ray L.

2. Ray. Define the *ray* in the following way:

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

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

3. Constructing set $O(x^0)$. Let J^0 be a set of indexes of integer components of vector x^0, i.e.

$$J^0 = \{j \in J \mid x_j^0 = [x_j^0]\}, \text{ where } J = \{1, 2, \ldots, n\}.$$

Define set $O(x^0)$ as the set of such points *x* which satisfy the following constraints:

$$\begin{cases} [x_j^0] \le x_j \le [x_j^0]+1, & \text{if } j \notin J^0 \\ [x_j^0] \le x_j \le [x_j^0]+1, & \text{if } j \in J^0 \text{ and } l_j > 0, \\ [x_j^0]-1 \le x_j \le [x_j^0], & \text{if } j \in J^0 \text{ and } l_j < 0, \\ x_j = [x_j^0], & \text{if } j \in J^0 \text{ and } l_j = 0, \end{cases} \tag{4}$$

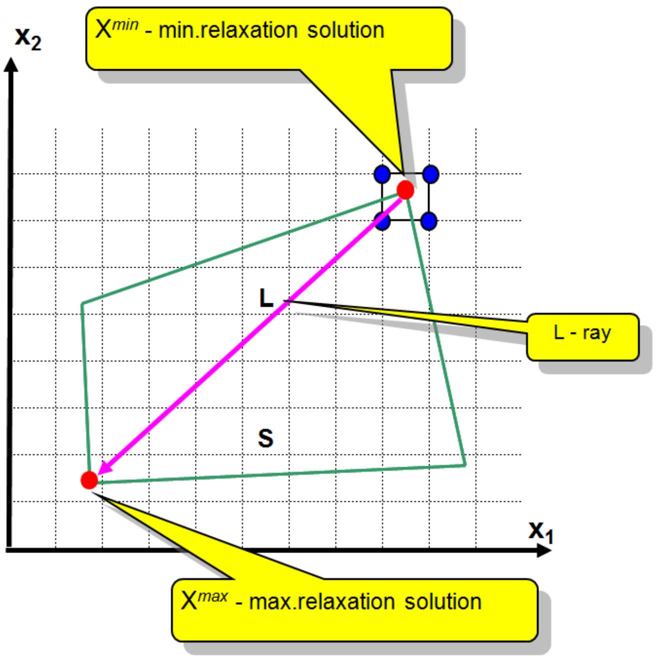
X^{min} - min.relaxation solution
x_2
L
L - ray
S
x_1
X^{max} - max.relaxation solution

Figure 2: The first unit-cube.

Before starting the iterations we have to define point $x' := x^0$ and calculate the first *perforation* point

$$P_{act} = (p_1, p_2, \ldots, p_n)$$

solving the following optimization problem:

$$\lambda \to \max \tag{5}$$

st.

$$p_j = x_j^0 + \lambda l_j \quad j = 1, 2, \ldots, n, \tag{6}$$

$$\begin{cases} [x_j^0] \le p_j \le [x_j^0]+1, & \text{if } j \notin J^0, \\ [x_j^0] \le p_j \le [x_j^0]+1, & \text{if } j \in J^0 \text{ and } l_j > 0, \\ [x_j^0]-1 \le p_j \le [x_j^0], & \text{if } j \in J^0 \text{ and } l_j < 0, \\ p_j = [x_j^0], & \text{if } j \in J^0 \text{ and } l_j = 0. \end{cases} \tag{7}$$

4. 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

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

where

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

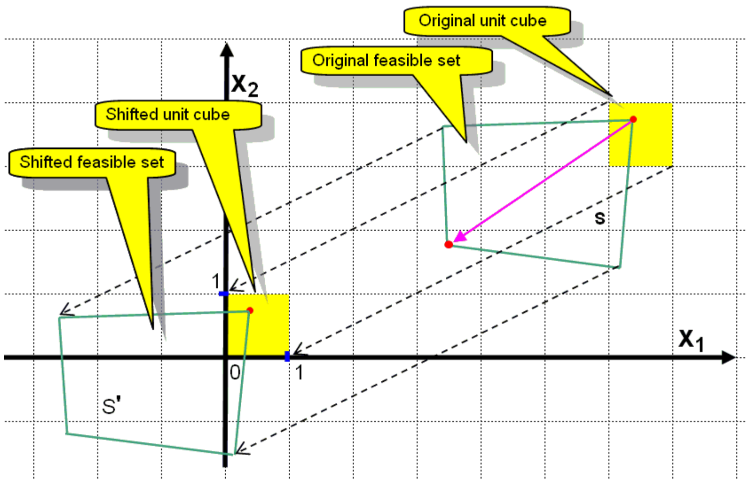

Figure 3: The shifted problem.

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

$$f'(y) = \sum_{j=1}^{n} c_j y_j + c_0 \to \max \tag{8}$$

st.

$$\sum_{j=1}^{n} a_{ij} y_j \le b'_i, \ i = 1, 2, \ldots, m, \tag{9}$$

$$y_j = 0/1, \quad j = 1, 2, \ldots, n, \tag{10}$$

where

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

using **Balas'** additive algorithm (implicit enumeration) [4]. If problem (8)-(10) 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,

5. 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:

$$\lambda \to \max \tag{11}$$

st.

$$x_j = x_j^0 + \lambda l_j \quad j = 1, 2, \ldots, n, \tag{12}$$

$$\begin{cases} [p_j] & \le & x_j & \le & [p_j]+1, & j \notin J', \\ [p_j] & \le & x_j & \le & [p_j]+1, & j \in J' \text{ and } l_j > 0, \\ [p_j]-1 & \le & x_j & \le & [p_j], & j \in J' \text{ and } l_j < 0, \\ & & x_j & = & [p_j], & j \in J' \text{ and } l_j = 0, \end{cases} \tag{13}$$

where $J' = \{ j \in J \mid p_j = [p_j] \}$. Here constraints (12) and (13) 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 (11)-(13) and λ' be the maximal value of objective function (11).

Let us define middle point *x'* of the line segment $[P_{act}; P_{next}]$:

$$x'_j = \frac{p_j + p'_j}{2} \quad j = 1, 2, \ldots, n. \tag{14}$$

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

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

$$\begin{cases} [x'_j] & \le & x_j & \le & [x'_j]+1, & \text{if } j \notin J', \\ [x'_j] & \le & x_j & \le & [x'_j]+1, & \text{if } j \in J' \text{ and } l_j > 0, \\ [x'_j]-1 & \le & x_j & \le & [x'_j], & \text{if } j \in J' \text{ and } l_j < 0, \\ & & x_j & = & [x'_j], & \text{if } j \in J' \text{ and } l_j = 0, \end{cases} \tag{15}$$

where $J' = \{ j \in J \mid x'_j = [x'_j] \}$. 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 (1)-(3). It means the method fails.

3. NUMERIC 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 [3]. 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:

$$f(x) = 20x_1 + 21x_2 + 18x_3 \rightarrow \min$$

st.

$$\begin{array}{rrrcr} 9x_1 + & 1.5x_2 + & 7x_3 & \le & 1350, \\ 5.5x_1 + & 1x_2 + & 9x_3 & \ge & 1250, \\ -4.5x_1 - & 10x_2 + & 2.5x_3 & \le & -1050, \end{array}$$

$$x_1, x_2, x_3 - \text{ integer.}$$

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

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

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

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

So the first shifted problem will be as follows:

$$\begin{array}{lccccccc} f'(y) & = & 20y_1 + & 21y_2 + & 18y_3 & + & 4921 & \to \max \\ \text{st.} & & & & & & & \\ & & 9y_1 + & 1.5y_2 + & 7y_3 & \le & 3.5\,, & \\ & & 5.5y_1 + & 1y_2 + & 9y_3 & \ge & 3.5\,, & \\ & & -4.5y_1 - & 10y_2 + & 2.5y_3 & \le & -7.5\,, & \\ & & & y_1, y_2, y_3 \text{ - } 0/1 & & & & \end{array}$$

Since this 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 (12)-(13)):

$$\lambda \to \max$$

st.

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

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

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.

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

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

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:

$$\begin{array}{ccccccc} f'(y) & = & 20y_1 + & 21y_2 + & 18y_3 & + & 4942 & \rightarrow \max \\ \text{st.} & & & & & & \\ & & 9y_1 + & 1.5y_2 + & 7y_3 & \le & 2, \\ & & 5.5y_1 + & 1y_2 + & 9y_3 & \ge & 2.5, \\ & & -4.5y_1 - & 10y_2 + & 2.5y_3 & \le & -2.5, \\ & & & & y_1, y_2, y_3 & \text{- } 0/1. & \end{array}$$

This problem has no feasible solution.

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

$$\lambda \rightarrow \max$$

st.

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

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

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:

$$\begin{array}{lcccccl} f'(y) & = & 20y_1 + & 21y_2 + & 18y_3 & + & 4922 \rightarrow \max \\ \text{st.} \\ & & 9y_1 + & 1.5y_2 + & 7y_3 & \le & 11, \\ & & 5.5y_1 + & 1y_2 + & 9y_3 & \ge & 8, \\ & & -4.5y_1 - & 10y_2 + & 2.5y_3 & \le & -2, \\ & & & y_1, y_2, y_3 - 0/1. \end{array}$$

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 the table with results obtained from WinGULF[1] 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".

Table 1:

Branching	Left → Right		Right → Left		
variable	Wo.I.B.	W.I.B.	Wo.I.B.	W.I.B.	
Minimal index	291	37	37	33	
Maximal index	31	23	243	23	90.5%
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	

In this numeric example the average reduction of the number of nodes to be processed in the binary tree is equal to 32.4%, while the best result is 90.5%.

4. MIPLIB TESTS

Encouraged and inspired by this results we performed further numeric experiments and tests using the professional solver **CPLEX** (i.e. IBM ILOG CPLEX Optimizer) and official test files from MIPLIB collection (see for instance http://miplib.zib.de/).

Therefore A. Rácz implemented the MIP adaptation of the method proposed using callable library of package CPLEX.

[1] Web site: http://zeus.nyf.hu/~bajalinov

The steps of the test code developed are as follows:

1. Reading **.mps** or **.lp** input file with the test problem.
2. CPLEX solves the problem without our bound.
3. Ray method calculates an initial bound.
4. CPLEX solves the problem using the bound defined in the previous step.

The following table shows the results obtained. Most of the files can be found in MIPLIB library (contributor A. Atamtürk, [1], [2]).

Table 2: Test results with MIPLIB problems

Name	Rows × Cols	Int	Nz	Time spent (sec) without bound	with bound	getting bound	Improvement
n4-3	1236 × 3596	174	14036	361,14	288,459	0,254	20,1%
n12-3	2430 × 6120	216	24048	69,689	42,633	0,568	38,0%
n5-3	1062 × 2550	150	9900	46,335	35,434	0,143	23,2%
n7-3	2336 × 5626	174	22156	24,586	21,502	0,31	11,3%
pr9	2220 × 7350	42	22176	108,52	86,464	11,618	9,6%
pr12	2313 × 5868	36	17712	2,604	1,604	0,4	23,0%
						Average	**20,9%**

This table shows the corresponding information about the test problems in the first five column (name, number of rows and columns, number of integer variables and nonzero coefficients). The other columns present the test results. In column "Time without bound" you can find the time in sec which was needed for IBM CPLEX for solving the IP problem without our bound. In the column "Time to define bound" we see the time elapsed while method was calculating the initial bound. Finally, column "Time with bound" shows CPLEX's reduced run time while solving the same IP problem but using initial bound calculated by our ray-method. As we can see an overall about 20,9% average improvement on run time can be observed in case of these test files. So spending a very little time for calculating an initial bound using our method often leads to significant improvement in performance of branch and bound procedure.

5. CONCLUSIONS

Summarizing results presented above, we have to emphasize that when choosing a ray it is very important to find such a ray that is located in the most inner part of feasible set *S*, since it gives us a hope that the ray selected in such a way will result a unit-cube with at least one feasible vertex quite quickly.

Originally the idea of using some sort of ray when solving IP problems was proposed by V.R. Khachaturov[2] et al. [13], [14]. Instead of search an optimal integer solution we use the idea of the ray to find a feasible integer solution as soon as possible. Our experiments with our package WINGULF have shown that such type of procedure often results such a good initial bound that dramatically reduces the size of the binary tree to be searched. All these results inspired us for experiments with callable library of CPLEX. The experiments with CPLEX have shown that the main idea of the method proposed may be useful, so we intend to continue numeric experiments and check the method with using package GUROBI[3].

REFERENCES

1 Atamtürk, A. (2002), "On Capacitated Network Design Cut-Set Polyhedra" *Mathematical Programming,* Vol. 92, pp. 425-437.

2 Atamtürk, A., Rajan D. (2002), "On Splittable and Unsplittable Capacitated Network Design Arc-Set Polyhedra" *Mathematical Programming,* Vol. 92, pp. 315-333.

3 Bajalinov, E. (2003), *Linear-fractional programming: theory, methods, software and applications,* Kluwer.

4 Balas, E. (1965), "An additive algorithm for solving linear programs with zero-one variables" *Operations Research,* Vol. 13, pp. 517-46.

5 Berliner, H. (1979), "The B tree search algorithm: A best-first proof procedure" *Artificial Intell.,* Vol. 12, no. 1, pp. 23-40.

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

7 Gendron, B., Crainic, T.G. (1994), "Parallel Branch-and-Bound Algorithms: Survey and Synthesis" *Opererations Research,* Vol. 42 issue 6, pp. 1042-1066.

8 Gupta, O. K., Ravindran, V. (1985), "Branch and Bound Experiments in Convex Nonlinear Integer Programming" *Management Science,* Vol. 31, no. 12, 1533-1546.

9 Hawkins, D. M. (2006), *Branch-and-Bound method,* Encyclopedia of Statistical Sciences, John Wiley and Sons.

10 Ibaraki, T. (1976), "Computational efficiency of approximate branch-and-bound algorithms" *Math. Oper. Res.*, Vol. 1, no. 3, pp. 287-298.

11 Ibaraki, T. (1976), "Theoretical comparisons of search strategies in branch-and-bound algorithms" *Int. J. Computer and Information Sciences*, Vol. 5, no. 4, pp. 315-343.

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

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

[2] Russian Academy Of Science, Computer Center, Moscow
[3] Gurobi Optimization

14 Khachaturov, V.R. (2000), *Combinatorial methods and algorithms for solving large-scale discrete optimization problems,* Nauka, Moscow.

15 Kumar, V., Kanal, L. N. (1983), "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.

16 Land, A. H., Doig, A. G. (1960), "An Automatic Method for Solving Discrete Programming Problems" *Econometrica,* Vol. 28, pp. 497-520.

17 Lawler, E. L., Wood, D. E. (1966), "Branch-And-Bound Methods: A Survey" *Operations Research*, Vol. 14, no. 4, pp. 699-719.

18 Mitten, L. (1970), "Branch and bound methods: General formulation and properties" *Operation Research*, Vol. 18, pp. 24-34.

19 Smith, D.R. (1984), "Random trees and the analysis of branch and bound procedures" *Journal of the Association for computing machinery*, Vol.31, no.1, pp. 163-188.

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

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