Binary Cut-and-Branch Method for Solving Linear Programming Problems with Boolean Variables

Yu. A. Mezentsev

Novosibirsk State Technical University, Russia
e-mail: mesyan@yandex.ru

Abstract. A numerical method is proposed for solving linear programming problems with Boolean variables. The method is based on an iterative application of a cutting-plane procedure that takes into account, as fully as possible, the properties of the problems being solved. Heuristic procedures are applied for the synthesis of cutting-planes as an intermediate step substantiating the construction of a solution tree.

Keywords: integer programming, binary cuts, efficient algorithm, solution tree.

1 Introduction

Theoretical and applied studies in the field of discrete optimization (DO) do not lose their relevance over the last fifty years. Although there are numerous and vigorously developing avenues of research within the field, a major issue associated with the NP-hardness of the majority of practically relevant DO problems has not yet been solved. An analysis of the current challenges in this area and ways of addressing them can be found, e.g., in a review paper by Leont'ev [1]. The recent trends in the development of DO methods are well described in the proceedings of the 16th Baikal International School–Seminar on optimization methods and their applications [2]. A systematic modern view of problem formulations and optimization algorithms can be found in well-known books by Western authors [3–6]. Special mention should be made of the book written by Korte and Vygen [3], which is a fundamental work reflecting the state and main directions in the development of DO methods as of the early 2000s, except for random search methods. Tobias Achterberg's doctoral thesis [7] may serve as a guide to DO algorithms that are widely used in software packages developed in North America and the EU. He considered the classical integer optimization methodologies, in particular cutting-plane methods (several types of cuts associated with specific types of problems: covers, knapsacks, cliques, Gomory cuts, Chvátal–Gomory cuts, etc.), branch and bound, branch and cut, and a series of heuristic algorithms including several techniques of constraint programming. In addition, he presented the basic techniques used to prescreen options and reduce dimensions, which were known as of 2007. All the DO methods and algorithms listed in the work have

In: A. Kononov et al. (eds.): DOOR 2016, Vladivostok, Russia, published at http://ceur-ws.org

been implemented in modern software optimization systems, such as IBM ILOG CPLEX optimization studio [8].

The vast majority of algorithms that are used for solving DO problems, in particular those of integer linear programming, can be classified into three groups [1,7,9-14]: (i) exact; (ii) approximate, and (iii) heuristic. According to their construction principles, the algorithms of the first and second groups may, in turn, be conventionally divided into (i) exhaustive search algorithms, (ii) dynamic programming, (iii) matroid optimization and greedy algorithms, and (iv) linearization. Many of the algorithms use hybrid schemes combining several principles.

The modern metaheuristics used in solving DO problems are described, e.g., in [10–12]. Usually, this term refers to a rather heterogeneous group of computational techniques, such as constraint programming (CP) [3, 7] and various modifications of random search methods in combination with local descent, including evolutionary (genetic) algorithms, ant colony algorithms, and annealing simulations (sometimes in combination with neural network algorithms). Most of the currently used techniques that belong to these groups can be found in [10, 12].

However, to date, no efficient algorithms have been proposed for exact solution of general DO problems, including linear programming problems with Boolean variables (LPPs with BVs).

There is a long and successful history of specialized algorithms for solving special DO problems that are compact reducible or directly belong to the class of mixed programming problems with Boolean variables. These include, e.g., location problems, network problems, trajectory problems, and various subclasses of scheduling theory problems (with fixed, nonfixed, and uncertain routes), cutting and packing problems, clique problems, knapsack problems, cover problems, and many other problems with relevant practical applications [15]. At present, the most intensive efforts are focused on these areas of the DO theory and applications. As a rule, there are attempts to design approximate or asymptotically accurate efficient algorithms. There has been substantial progress in the construction of special algorithms for solving the above problems [2, 3, 7].

Today, however, there are no known efficient algorithms for finding an accurate solution of the general DO problem, including the general linear programming problem with Boolean variables.

This work is also focused mainly on the development of methods for solving mixed linear programming problems with Boolean variables and is a direct development of the questions discussed in [13, 14]. The object of research is a problem of the form

$$\gamma(x) = c^T x \to \max, \tag{1}$$

$$Ax \le b, \ \overline{0} \le x \le \overline{1}, \tag{2}$$

$$x \in I_2^n, \tag{3}$$

which is a linear programming problem with Boolean variables. Conditions (3) specify that the solution belongs to one of the vertices of a unit hypercube of dimension n;

$c, x, \bar{0}, \bar{1}$; are vectors of the same dimension ($\bar{0}$ is a zero vector and $\bar{1}$ is a vector of ones).

This approach does not lead to a loss of generality since any integer programming problem can be compactly reduced to an LPP with BVs [13].

In [13], an original numerical method was presented for solving this kind of problems, the main idea of which is a consistent building of a system of binary cuts (BCs) for the relaxed problem (1)–(2) so that the basis matrix of the complemented inequality system (2) would eventually become totally unimodular, which guarantees the fulfillment of conditions (3) in achieving the objective (1).

This method uses a heuristic BC synthesis procedure, which does not guarantee the building of correct cuts at each step and, therefore, uses implicit cyclic enumeration for the right-hand sides of the cuts. Numerical experiments showed that, along with advantages, the method has a number of drawbacks, the main of which is that the optimum points of problem (1)–(3) may be cut off.

This deficiency was addressed by applying a number of new rules of BC synthesis and supplementing the method by a branching procedure [14]. The aim of this paper is to discuss the results obtained in [13, 14].

2 Main Principles of the Binary Cut-and-Branch Method

Suppose x^0 is the solution of the relaxed problem (1)–(2), $[\cdot]$ is the integer part of the number, and $\beta_0 = \hat{\alpha}^T x^0$, where $\hat{\alpha}_j \in \{-1, 0, 1\}$, $j = \overline{1,n}$. Then any inequality of the form

$$\hat{\alpha}^T x \le \hat{\beta}_0 , \ \hat{\alpha}_j \in \{-1, 0, 1\} , \ j = \overline{1,n} , \ \hat{\beta}_0 = [\beta_0], \tag{4}$$

is called a *binary cut* for problem (1)–(2). Strictly speaking, such a cut is not exactly binary because the vector $\hat{\alpha}$ may contain components (–1) other than zeroes and ones. However, since problem (1)–(3) can always be normalized in such a way that the coefficients of the objective function (1) would be nonnegative: $c_j \ge 0, \ j = \overline{1,n}$, it can be assumed for $\hat{\alpha}_j$ in (4) that $\hat{\alpha}_j \in \{0,1\}$, $j = \overline{1,n}$ [13]. Thus, (4) transforms into inequalities also known as *covers* (they are usually applied in solving the covering problem [3]).

The thus reduced problem (1)–(3) can be written as:

$$\gamma(\tilde{x}) = \tilde{c}^T \tilde{x} + C \to \max , \tag{5}$$

$$\tilde{A}\tilde{x} \le \tilde{b} , \ \bar{0} \le \tilde{x} \le \bar{1} , \tag{6}$$

$$\tilde{x} \in I_2^n , \tag{7}$$

In this case the coefficients in the left-hand part of any *i*-th cut (4) have the property $\hat{\alpha}_j^i \in \{0,1\}$, $j = \overline{1,n}$ [13]. Let $\tilde{x}^0$ be the solution of problem (5)–(6), then a BC with respect to system (6) is

$$\hat{\alpha}^T \tilde{x} \le \hat{\beta}_0\,,\ \hat{\alpha}_j \in \{0,\ 1\}\,,\ j = \overline{1,n}\,,\ \hat{\beta}_0 = [\beta_0]\,,\ \beta_0 = \hat{\alpha}^T \tilde{x}^0\,, \tag{8}$$

and the complementary BC system for the system of constraints (6) is

$$\mathrm{A}^D \tilde{x} \le \beta\,, \tag{9}$$

where $\mathrm{A}^D = \left\| \hat{\alpha}_j^i \right\|_{m^D \times n}$, $\hat{\alpha}_j^i \in \{0,1\}$, $j = \overline{1,n}$ is the coefficient matrix of the binary system, and the vector composed of the right-hand parts of the cuts β is defined in accordance with (8).

If A^D has the property of total unimodularity and all cuts (9) are valid (A^D is a part of the basis matrix of system (6),(9)), then at $m^D \ge n$, A^D includes the basis matrix and the optimal solution of the LP problem (5),(6),(9) is the optimal solution of the LPP with BVs (5)–(7).

Like in Gomory algorithms, any active constraints, i.e., inequalities that form the basis system (6), (9) given an optimal solution of the current problem and inequalities ensuing from the basis inequality system, can act as generating constraints. If $\tilde{x}^0$ is the solution of the relaxed problem (5)–(6), then

$$\varsigma^T \tilde{x} \le \varphi_0\,,\ \varphi_0 = \varsigma^T \tilde{x}^0\,, \tag{10}$$

Expression (10) can be a generating inequality only if $\varsigma_j = \sum_{i \in I^B} \lambda_i \tilde{a}_{ij}$, $\lambda_i \ge 0$,

where $\tilde{a}_{ij}, i \in I^B$ are the coefficients of the basis part $\tilde{A}$ and λ_i are the nonnegative weights of the basis constraints. In particular, if λ_i are the dual estimates of constraints (6), then $\varsigma_j = \tilde{c}_j$, $j = \overline{1,n}$. The generating inequalities are also promising in the case when λ_i are the reciprocals of any norm of the constraint vectors (e.g., Euclidean norm).

However, in the general case, the notion of a valid BC [13] is not equivalent to that of a valid Gomory cut. Now we define the conditions that must be met by a valid BC.

Consider an auxiliary problem:

$$\beta(z) = \hat{\alpha}^T z \to \max\,, \tag{11}$$

$$\tilde{A}z \le \tilde{b}\,,\ \bar{0} \le z \le \bar{1}\,, \tag{12}$$

$$\hat{\beta} = \left[\beta(z^0) \right], \tag{13}$$

where z^0 is the optimal solution of (11)–(12) and $\hat{\beta}$ is the right-hand part of (8).

There are three possible outcomes.

1. $z^0 = \tilde{x}^0$. In this case, (8) is called a strong cut [14].
2. $z^0 \neq \tilde{x}^0$ and $\left[\hat{\alpha}^T \tilde{x}^0 \right] = \left[\beta(z^0) \right]$. Then (8) is a slack valid cut.
3. $z^0 \neq x^0$ and $\left[\hat{\alpha}^T \tilde{x}^0 \right] < \left[\beta(z^0) \right]$. In these circumstances there is a positive integer residue $\mu = \left[\hat{\beta}(z^0) \right] - \left[\hat{\alpha}^T \tilde{x}^0 \right] > 0$, and the inequality $\hat{\alpha}^T \tilde{x} \le \left[\hat{\alpha}^T \tilde{x}^0 \right]$ is an invalid cut. Then $\hat{\alpha}^T \tilde{x} \le \hat{\beta}$ with the right-hand part of (13) does not make sense since it would not be active if added to problem (5)–(6).

It can be shown that the so-called strong BCs are a special case of *Gomory cuts* [14]. They do not always exist for this class of problems. This fact is illustrated by the following example.

Example 1.

$$\gamma(x) = 3x_1 + 2x_2 \to \max$$
$$2x_1 + x_2 \le 2.3,$$
$$2.5x_1 + 3x_2 \le 3.325,$$
$$x_1 - x_2 \le 0.8, \quad x_1, x_2 \in I_2^2.$$

In the relaxed problem, the latter condition is replaced by $0 \le x_j \le 1, \quad j = 1,2$. Its optimal solution is $(x^0)^T = (0.85, 0.6)$ and $\gamma_{\max} = 3.75$.

The objective function of problem (11)–(12) is $\beta(z) = z_1 + z_1 \to \max$.

The solution of the auxiliary problem $(z^0)^T = (0.53, 1)$ is not x^0; however, $x_1 + x_2 \le 1$ is a valid cut.

In Fig. 1 the boundaries of constraints (6) are shown in solid lines; the level lines of the objective function are given in dashed lines; and the boundaries of the generating inequity and slack BC are shown in dash-dot lines. In this example there is no strong BC. In contrast to strong BCs, slack ones always exist.

The proof of the existence theorem is based on the convexity property of polyhedron (6) and the fact that there is always inequality (4), which cuts off any set of adjacent vertices in a unit hypercube from (6) ($\overline{0} \le \tilde{x} \le \overline{1}$).

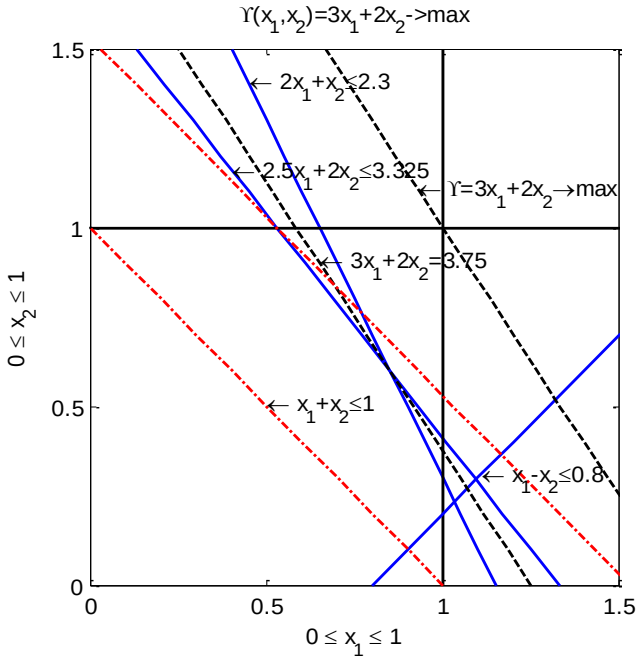

Fig. 1. Example of a slack valid cut**.**

There are no Gomory cuts corresponding to a slack BC. Theoretically, any such cut can only be built using a finite sequence of Gomory cuts. From this perspective, the slack BCs are not exactly slack.

3 Binary Cut Formation Algorithms

It has not yet been possible to suggest a polynomial complexity algorithm for the synthesis of reliably valid BCs. However, a number of heuristic rules and procedures (of varying difficulty) can be proposed for solving the synthesis problem in a sense of good binary cuts.

A good cut is, naturally, any valid BC or a cut with a minimum integer residual μ.

1) Rule of nonzero coordinates for the previous (current) relaxation.

The rule is used to calculate the coefficients in the left-hand part of the BC:

$$\hat{\alpha}_j = \begin{cases} 1, & \text{if} \quad \tilde{x}_j^0 > 0 \\ 0, & \text{if} \quad \tilde{x}_j^0 = 0 \end{cases} \tag{14}$$

The right-hand part is calculated according to (8).

2) *Rule of the nearest cut to the generating inequality.* This rule is reduced to solving the following problem:

find the coefficients $\hat{\alpha}_j \in \{0,1\}$, $j = \overline{1,n}$ that maximize the relation

$$k(\hat{\alpha}) = \frac{\vec{\varsigma}^T \hat{\alpha}}{|\varsigma|_2 |\hat{\alpha}|_2} \tag{15}$$

At first glance, this problem is not simpler than the original LPP with BVs (5)–(7) since the number of "rational" options alone for the desired cut is greater than the total number of combinations of variables in (5)–(7). However, in reality there is no need for an exhaustive search through solution options. There is a complexity algorithm $O(n \log n)$ [13] for finding an exact solution, which is based on sorting [14].

The two heuristics in Example 1 make it possible to construct a valid cut. However, there are numerous counterexamples wherein the application of (14) and (15) does not lead to the synthesis of valid BCs [13].

The underlying ideas of (14) and (15) can be developed by using second-level heuristics. In addition to (11)–(13), an alternative indication can be used to identify a valid BC. Consider a problem:

$$\gamma(\tilde{x}) = \tilde{c}^T \tilde{x} + C \to \max \tag{16}$$

$$\tilde{A}\tilde{x} \le \tilde{b}\,,\ \overline{0} \le \tilde{x} \le \overline{1}\,, \tag{17}$$

$$\hat{\alpha}^T \tilde{x} \ge \beta(\tilde{x}^0) + 1\,, \tag{18}$$

where $\beta(\tilde{x}^0) = \left\lceil \hat{\alpha}^T \tilde{x}^0 \right\rceil$ and $\tilde{x}^0$ is the solution of (5)–(6).

If problem (16)–(18) has a solution, then the cut $\hat{\alpha}^T \tilde{x} \le \beta(\tilde{x}^0)$ is invalid. Conversely, if (16)–(18) has no solution due to the inconsistency of system (17)–(18), then $\hat{\alpha}^T \tilde{x} \le \beta(\tilde{x}^0)$ is a valid cut.

In Example 1, it is easy to see that the additional constraint $x_1 + x_2 \ge 2$ renders the inequality system inconsistent and $x_1 + x_2 \le 1$ is a valid cut.

3) *Selection in a set of estimates for the variables of the current relaxation.*

Let λ_j^0 be the estimates for the variables $\tilde{x}_j$ in (5)–(6), which are defined as optimal dual estimates for the variables $\tilde{x}_j \le 1$, $j = \overline{1,n}$. υ_j^0 are the estimates for the constraints in the dual of (5)–(6). It follows from the complementary slackness conditions that $\lambda_j^0 = 0$ if $\tilde{x}_j^0 < 1$ and $\lambda_j^0 \ge 0$ if $\tilde{x}_j^0 = 1$. Note that the case $\lambda_j^0 = 0$ and

$\tilde{x}_j^0 = 1$ may only take place if there are alternative optima in the current relaxation. Let the fines for raising $\tilde{x}_j$ to one ($\bar{h}_j$) and reducing it to zero ($\underline{h}_j$) be used as estimates for the basis variables $\tilde{x}_j$ in the case $0 < \tilde{x}_j^0 < 1$

If the coefficients of the expansion row of the *l*th basis variable in the optimal table are denoted by $\tilde{a}_{lj}^0$ and the corresponding coefficient of the column containing the right-hand parts by $\tilde{x}_l^0 (< 1)$, then an accurate estimate of the fine for reducing $\tilde{x}_l$ to zero can be obtained by adding to the transformed problem a constraint $\sum_{j=1}^{n} -\tilde{a}_{lj}^0 \xi_j \le -\tilde{x}_l^0$, where ξ_j are the nonbasis variables in the optimal table. An accurate estimate of the fine for raising $\tilde{x}_l$ to one is obtained by adding a constraint $\sum_{j=1}^{n} \tilde{a}_{lj}^0 \xi_j \le \tilde{x}_l^0 - 1$ to the transformed problem. Calculating these two estimates is rather time-consuming because of the need to solve additional LP problems. Therefore, it makes sense to confine the analysis to the minimum estimates [12], which can be defined as follows:

$$\underline{h}_l = -\max_{<0} \left\{ \frac{\tilde{c}_j^0}{-\tilde{a}_{lj}^0} \right\} \cdot \tilde{x}_l^0, \quad \bar{h}_l = -\max_{<0} \left\{ \frac{\tilde{c}_j^0}{\tilde{a}_{lj}^0} \right\} \cdot (1 - \tilde{x}_l^0).$$

In the general case, both fines are calculated according to the relations:

$$\underline{h}_l = \begin{cases} \lambda_l^0, & \text{if } x_l^0 = 1, \\ -\max\limits_{<0} \left\{ \dfrac{\tilde{c}_j^0}{-\tilde{a}_{lj}^0} \right\} \cdot \tilde{x}_l^0, & \text{if } 0 < x_l^0 < 1, \\ 0, & \text{if } x_l^0 = 0. \end{cases} \quad \bar{h}_l = \begin{cases} 0, & \text{if } x_l^0 = 1, \\ -\max\limits_{<0} \left\{ \dfrac{\tilde{c}_j^0}{\tilde{a}_{lj}^0} \right\} \cdot (1 - \tilde{x}_l^0), & \text{if } 0 < x_l^0 < 1, \\ \upsilon_l^0, & \text{if } x_l^0 = 0. \end{cases}$$

Then the priority of the coordinate $\tilde{x}_l$ in a cut can be determined, e.g., as shown below:

$$h_l = \begin{cases} \lambda_l^0, & \text{if } x_l^0 = 1, \\ -\max\limits_{<0} \left\{ \dfrac{\tilde{c}_j^0}{-\tilde{a}_{lj}^0} \right\} \cdot \tilde{x}_l^0 + \max\limits_{<0} \left\{ \dfrac{\tilde{c}_j^0}{\tilde{a}_{lj}^0} \right\} \cdot (1 - \tilde{x}_l^0), & \text{if } 0 < x_l^0 < 1, \\ -\upsilon_l^0, & \text{if } x_l^0 = 0. \end{cases} \tag{19}$$

Relation (19) can be used to estimate all the coordinates and introduce, according to $h_j, j=\overline{1,n}$, a nonstrict-order relationship over the entire set of $\tilde{x}_j, j=\overline{1,n}$. This makes it possible to build and estimate a variety of alternative cuts (8) rather than only one cut, as in (14) and (15).

According to (19), the entire set of variables can be split into three disjoint subsets, which are sorted descendingly: $h_j, j=\overline{1,n}$. Let n_1, n_2, and n_3 be the number of elements in each subset, with all $\tilde{x}_j, j=\overline{1,n_1}$ belonging to the first subset, $\tilde{x}_j$, $j=\overline{n_1+1,n_1+n_2}$ belonging to the second subset, and $\tilde{x}_j$, $j=\overline{n_1+n_2+1,n}$ belonging to the third subset.

Consider a totality $\hat{\mathrm{A}}$ of n_2 vectors of dimension n

$$\hat{\alpha}^1=(1,1,...,1,0,0,...,0) \text{ (contains } n_1+1 \text{ original ones)},$$

$$\hat{\alpha}^2=(1,1,...,1,1,0,...,0) \text{ (contains } n_1+2 \text{ original ones)},$$

$$\ldots \qquad \ldots$$

$$\hat{\alpha}^{n_2}=(1,1,...,1,1,1,...,0) \text{ (contains } n_1+n_2 \text{ original ones)}.$$

For each vector there is a nonincreasing estimate $h_j, j=\overline{1,n_2}$.

Thus, to find the coefficients of the best cut (8) in the totality $\hat{\mathrm{A}}$, it is sufficient to consistently synthesize from one to n_2 BCs until either the conditions intrinsic in regular cuts are met or there are no more elements left in $\hat{\mathrm{A}}$.

4) *Selection in a set of the nearest cuts.*

Relation (15) allows an even simpler search and estimation of a set of alternative cuts. To this end, it is sufficient to arrange the components of the vector ς in descending order (the new vector is denoted by $\vec{\varsigma}$) and consider a totality $\hat{\mathrm{A}}$ of n vectors of dimension n

$$\hat{\alpha}^1=(1,0,...,0),$$

$$\hat{\alpha}^2=(1,1,0,...,0),$$

$$\ldots \qquad \ldots$$

$$\hat{\alpha}^j=(1,1,...,1,0,0,...,0) \text{ (contains } j \text{ original ones)},$$

$$\ldots \qquad \ldots$$

$$\hat{\alpha}^n=(1,1,...,1).$$

Each $\hat{\alpha}^j$ is set in correspondence with $k(\hat{\alpha}^j)$ from (15). Then, the discrete function $k(\hat{\alpha}^j)$ uniquely defines the priority of each of the (alternative) cuts with the coefficients $\hat{\alpha}^j$. Here $k(\hat{\alpha}^j)=\dfrac{\vec{\varsigma}^T\hat{\alpha}^j}{|\varsigma|_2\left|\hat{\alpha}^j\right|_2}$, $j=\overline{1,n}$. This function has a strict maximum. Therefore, to find the coefficients of the best set in the totality $\hat{\mathrm{A}}$, there is no

need to analyze all $\hat{\alpha}^j$, $j=\overline{1,n}$. It is enough to compare cuts (8) at the maximum point $k(\hat{\alpha}^j)$ with those built at the nearest points (vectors) $\hat{\alpha}^j$ to the right and left.

4 Cut-and-Branch Algorithms

A simplest algorithm is based on heuristics (14) and (15). Therefore, it is not guaranteed that the cuts being constructed are valid. This circumstance calls for the use of a branching scheme for the solutions of a sequence of relaxed LP problems (5)–(6).

Algorithm A1

1. Suppose that we have obtained the solution of the original relaxed problem (5)–(6): $\tilde{x}^0$ and $\gamma(\tilde{x}^0)$. If $\tilde{x}^0$ is an integer, the algorithm stops. Otherwise, it goes to step 2.

2. At step t we select the vertex with the maximum estimate $\gamma(\tilde{x}^r)$ for probing. If the list of vertices is empty, the problem does not have an integer solution. The algorithm stops. If the vertex with the maximum estimate $\gamma(\tilde{x}^r)$ contains an integer solution $\tilde{x}^r$, it is the optimal one. The algorithm stops. Otherwise:

3. We create two new candidates for each of which the current matrix A^D is supplemented, according to procedures (14) or (15), by the only cut (8) or (18), respectively.

4. We use the principle applied in the auxiliary problem (16)–(18) and solve a pair of alternative subproblems with the cuts $\hat{\alpha}^{(t+1)T}\tilde{x} \le \beta(\tilde{x}^r)$ and $\hat{\alpha}^{(t+1)T}\tilde{x} \ge \beta(\tilde{x}^r)+1$.

5. Their solutions $\tilde{x}^{t+1}$ and $\tilde{x}^{t+2}$ and estimates $\gamma(\tilde{x}^{t+1})$ and $\gamma(\tilde{x}^{t+2})$ are saved by adding them to the list of the tree vertices. If any of the candidates has no solution, it is withdrawn from the list of the vertices.

6. We increase the step number ($t := t+1$) and go to step 2.

If we use the more complex heuristic rules of BC synthesis, which are presented above, the algorithm should be correspondingly modified. This concerns mostly steps 3 and 4, which require the consideration not of a single pair but a set of pairs of alternative cuts. Statistical data on the efficiency of applying the heuristic rules of BC synthesis in algorithm A1 are given in Table 1.

Table 1. Efficiency of applying the heuristic rules of BC synthesis

Rule	Percentage of regular cuts in the total number		
	minimum	maximum	average
(1) by nonzero coordinates for the previous relaxation	17	42	31

(2) selection of the nearest cut	21	63	39
(3) selection in a set of estimates of variables	34	100	57*
(4) selection in a set of the nearest cuts	43	100	65*

* requires experimental verification

Consider an example of the application of algorithm *A1*.

Example 2 [13]. Find $x^T = (x_1, x_1, x_3), x_j \in I_7^3, j = \overline{1,3}$ under the conditions

$$2x_1 + x_2 - x_3 \le 4,$$
$$4x_1 - 3x_2 \le 2,$$
$$-3x_1 + 2x_2 + 3x_3 \le 12,$$
$$\gamma = x_1 - 3x_2 + 3x_3 \to \max.$$

Let us reduce this problem to an LLP with BVs with the normalization γ. Suppose that

$$x_j = \sum_{l=1}^{L} \beta_l x'_{jl},\ \beta_l = (1,2,4),\ \tilde{x}_{jl} = \begin{cases} x'_{jl}, & j = \overline{1,3},\ \ l = 1,3 \\ 1 - x'_{jl}, & j = \overline{1,3},\ \ l = 2 \end{cases}.$$

The transformed problem has the form:

find $\tilde{x} = (\tilde{x}_{11}, \tilde{x}_{12}, ..., \tilde{x}_{33}), \tilde{x}_{jl} \in I_2^9, j = \overline{1,3}, l = \overline{1,3}$ under the conditions:

$$2\tilde{x}_{11} + 4\tilde{x}_{12} + 8\tilde{x}_{13} - \tilde{x}_{21} - 2\tilde{x}_{22} - 4\tilde{x}_{23} - \tilde{x}_{31} - 2\tilde{x}_{32} - 4\tilde{x}_{33} \le -3$$
$$4\tilde{x}_{11} + 8\tilde{x}_{12} + 16\tilde{x}_{13} + 3\tilde{x}_{21} + 6\tilde{x}_{22} + 12\tilde{x}_{23} \le 23$$
$$\gamma = \tilde{x}_{11} + 2\tilde{x}_{12} + 4\tilde{x}_{13} + 3\tilde{x}_{21} + 6\tilde{x}_{22} + 12\tilde{x}_{23} + 3\tilde{x}_{31} + 6\tilde{x}_{32} + 12\tilde{x}_{33} - 21 \to \max.$$

Algorithm *A1* with the use of the third heuristics generates the following BC system (the cuts are given in the order in which they are formed):

$$y_1 \to 0\tilde{x}_{11} + 0\tilde{x}_{12} + 1\tilde{x}_{13} + 0\tilde{x}_{21} + 0\tilde{x}_{22} + 0\tilde{x}_{23} + 0\tilde{x}_{31} + 0\tilde{x}_{32} + 0\tilde{x}_{33} \le 0$$
$$y_2 \to 0\tilde{x}_{11} + 1\tilde{x}_{12} + 0\tilde{x}_{13} + 1\tilde{x}_{21} + 1\tilde{x}_{22} + 1\tilde{x}_{23} + 0\tilde{x}_{31} + 0\tilde{x}_{32} + 0\tilde{x}_{33} \le 3$$
$$y_3 \to 1\tilde{x}_{11} + 0\tilde{x}_{12} + 0\tilde{x}_{13} + 1\tilde{x}_{21} + 1\tilde{x}_{22} + 1\tilde{x}_{23} + 0\tilde{x}_{31} + 0\tilde{x}_{32} + 0\tilde{x}_{33} \le 3$$
$$y_4 \to 0\tilde{x}_{11} + 0\tilde{x}_{12} + 0\tilde{x}_{13} + 0\tilde{x}_{21} + 1\tilde{x}_{22} + 1\tilde{x}_{23} + 0\tilde{x}_{31} + 1\tilde{x}_{32} + 1\tilde{x}_{33} \le 3$$
$$y_5 \to 0\tilde{x}_{11} + 0\tilde{x}_{12} + 0\tilde{x}_{13} + 0\tilde{x}_{21} + 1\tilde{x}_{22} + 1\tilde{x}_{23} + 1\tilde{x}_{31} + 0\tilde{x}_{32} + 1\tilde{x}_{33} \le 3$$
$$y_6 \to 0\tilde{x}_{11} + 0\tilde{x}_{12} + 0\tilde{x}_{13} + 0\tilde{x}_{21} + 0\tilde{x}_{22} + 1\tilde{x}_{23} + 0\tilde{x}_{31} + 1\tilde{x}_{32} + 1\tilde{x}_{33} \le 2$$
$$y_7 \to 1\tilde{x}_{11} + 1\tilde{x}_{12} + 0\tilde{x}_{13} + 1\tilde{x}_{21} + 1\tilde{x}_{22} + 1\tilde{x}_{23} + 1\tilde{x}_{31} + 1\tilde{x}_{32} + 1\tilde{x}_{33} \le 4$$

All the cuts except for the last one are regular. Figure 2 shows the solution tree for this example. The numbers near the vertices are the values of the objective functions in the optimal solutions of the subproblems.

As a result, we obtain the solutions of the transformed and original problems

$$\tilde{x}^{*T} = (0, 0, 0, 1, 1, 1, 0, 0, 1),\ \gamma(\tilde{x}^*) = 12.$$

$$x'^{*T} = (0, 0, 0, 0, 0, 0, 0, 0, 1),\ x^{*T} = (0,\ 0,\ 4),\ \text{and}\ \gamma(x^*) = 12.$$

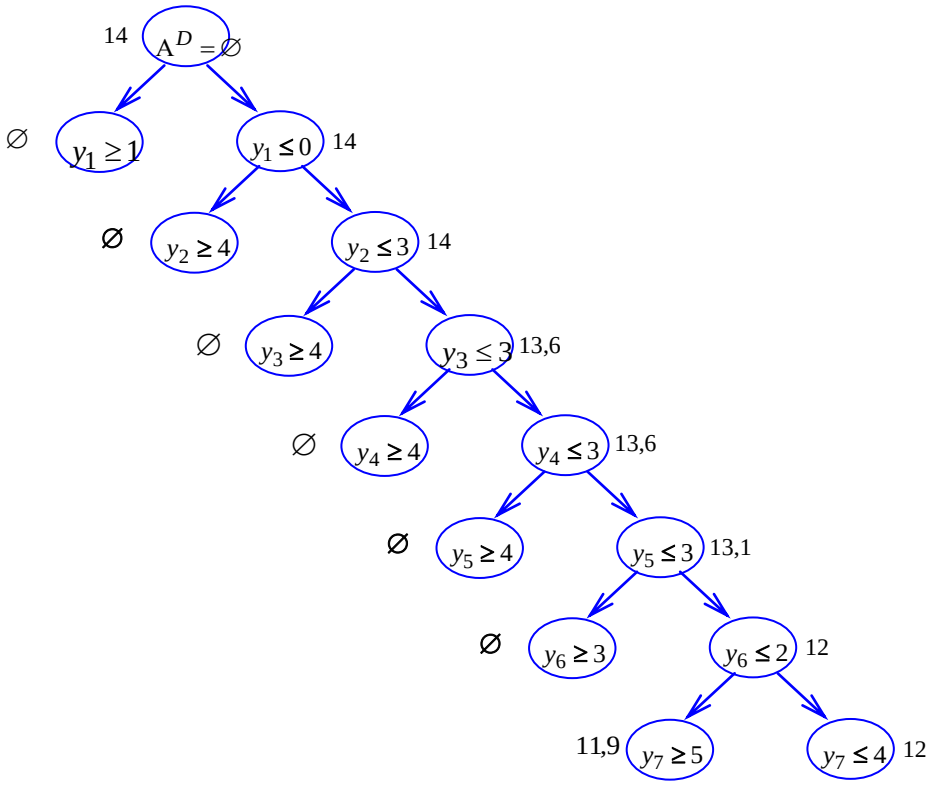

Fig. 2. Tree of solutions and cuts.

5 Comparative Performance Evaluations

There are no a priori lower bounds for the performance of the algorithms used for solving NP-hard LPPs with BVs. Nevertheless, we will try to compare the performance of algorithm A1 with other search algorithms on the basis of upper bounds.

The upper bound for the complexity of solving an LPP with BVs, which is the same for all exhaustive search algorithms, is: $N = 2^n$, where N is the number of relaxed subproblems which need to be solved to reliably determine the accurate optimum and n is the number of Boolean variables in the problem. Such a complexity estimate can be provided, e.g., by exhaustive search and dynamic programming methods. In BB, N corresponds to the maximum number of end vertices of the solution tree. Since the length of any branch of the tree from the initial vertex to the end one is $O(n)$, then the upper bound for the complexity of BB is higher than for exhaustive search: $N_{bb} = O(n)2^n$.

Let N_{bc} be the upper bound for the complexity of *A1*. As shown in [14], the upper bound for the number of binary cuts necessary to find the optimum of an LPP with

BVs under the condition that all the cuts are valid is $O(n^2)$. The proportion of valid BCs in their total number is a measure of efficiency of a heuristics, which is used in algorithm A1. We denote this proportion by δ, $0 \le \delta \le 1$. Then $N_{bc} = O(n^2)2^{(1-\delta)n}$ is the upper bound for the complexity of algorithm *A1*. Let us now return to Table 1, from which we derive an a posteriori estimate for the guaranteed number of valid BCs, which is 30% of the total number of BCs. Thus, we have an efficiency estimate for the first heuristics $\delta = 0.3$, and the overall estimate is, hence, $N_{bc} = O(n^2)2^{0.7n}$.

The third and fourth heuristics allow for an increase in the efficiency of the cuts. The results obtained may also affect the lower bounds and potentially suggest that the efficiency of A1 is many times greater than that of all the known integer optimization algorithms. For example, let us compare N_{bb} and N_{bc} : $N_{bb} / N_{bc} = \dfrac{O(n)2^n}{O(n^2)2^{0.7n}} = \dfrac{2^{0.3n}}{O(n)}$. If we assume that $n = 1000$, the complexity N_{bc} is 86 orders of magnitude lower than N_{bb}.

We also note that if δ tends to unity, *A1* becomes an efficient integer optimization algorithm.

6 Conclusions

Experimental evidence was obtained for the reliability of the binary cut-and-branch method. The applicability of the method for solving mixed integer optimization problems has been successfully tested in experiments. The simplest versions of the BC synthesis procedure were tested. An understanding was reached as to the development prospects of the proposed approach to solving computationally hard optimization problems. As regards the possibility of creating an efficient exact numerical method based on this approach, it should be noted that the mere existence of a solution tree is not identical to the fundamental impossibility of solving an LPP with BVs in polynomial time. If the current number of tree vertices at any iteration depends polynomially on the dimension and the length of each branch (the number of intermediate vertices) to any end vertex is also a polynomial of the dimension of the original problem, the proposed method is effective. While the second condition is confirmed experimentally, meeting the first one requires an increase in the efficiency of BC synthesis procedures.

References

1. Leont'ev V.K. Discrete optimization, Zh. Vychisl. Mat. Mat. Fiz., 2007, vol. 47, N 2, pp. 338–352.
2. Proceedings of the 16th Baikal International School-Seminar of optimization methods and their applications, Irkutsk: ISEM SO RAN, 2014.

3. Korte B., Vygen J. Combinatorial optimization. Theory and algorithms. Springer, 2002. 572 p.
4. Du D., Pardalos P. (eds.) Handbook of combinatorial optimization. Supplement vol. A, Kluwer academic publishers, 1999. 649 p.
5. Du D., Pardalos P. (eds.) Handbook of combinatorial optimization. Supplement vol. B. Springer, 2005, 403 p.
6. Schrijver A. Theory of linear and integer programming. Wiley, 1999, 483 p.
7. Achterberg T. Constraint integer programming, Genehmigte Dissertation doktor der Naturwissenschaften. Technischen Universitat Berlin, 2007, 418 p.
8. IBM ILOG CPLEX V12.5 User's Manual for CPLEX. IBM Corporation, 2012 – 952 p.
9. Zaslavskii A.A., Lebedev S.S. Nodal-vector method in integer programming [in Russian], Preprint # WP/2000/94, Moscow: TsEMI RAN, 2000, 81 p.
10. Glover F., Kochenberger G.A. eds. Handbook of metaheuristics. Kluwer academic publishers, 2003 – 560 p.
11. Vasil'ev I.L., Klimentova K.B. Branch-and-cut method for a facility location problem with clients' preference orderings, Diskret. Analiz. Issled. Operatsyi, 2009, vol. 16, N 2, pp. 21–41.
12. Rutkovskaya D., Pilin'skii M., Rutkovskii L., Neural networkd, genetic algorithms, and fuzzy systems [in Russian], Noscow: Goryachaya Liniya – Telekom, 2006, 452 p.
13. Mezentsev Yu.A., Efficient algorithm of integer programming, Nauch. Vest. Novosib. Gos. Tekh. Univ., 2009, N 2(35), pp. 91–114.
14. Mezentsev Yu.A., Binary cut-and-branch method in binary programming, Dokl. Akad. Nauk. Vyssh. Shkoly RF, Novosibirsk: Novosib. Gos. Tekh. Univ., 2011, N 1(16), pp. 12-25.
15. Mezentsev Yu.A. Effective numerical methods for solution of discrete optimization problems management [in Russia] Serija "Monografii NGTU", Novosibirsk: NSTU, 2015, 275 p. ISBN 978-5-7782-2689-0