FACTA UNIVERSITATIS (NIŠ)
Ser. Math. Inform. Vol. 33, No 2 (2018), 337–360
https://doi.org/10.22190/FUMI1802337M

A BRANCH-AND-BOUND ALGORITHM FOR A PSEUDO-BOOLEAN OPTIMIZATION PROBLEM WITH BLACK-BOX FUNCTIONS *

Igor S. Masich and Lev A. Kazakovtsev

Abstract. We consider a conditional pseudo-Boolean optimization problem with both the objective function and all constraint functions given algorithmically (black-box functions) and defined on $\{0,1\}^n$ only. We suppose that these functions have certain properties, for example, unimodality and monotonicity. To solve problems of this type, we propose an optimization algorithm based on finding boundary points of the feasible region and the branch-and-bound method. The developed algorithm is aimed at the reception of an exact solution of an optimization problem. In addition, this algorithm can be used as an improvement of approximate algorithms such as the greedy heuristic and the random search algorithms for finding boundary points. Even after a small number of iterations (branchings), a significant improvement of the found feasible solution is achieved.
Keywords: Pseudo-Boolean optimization problem, branch-and-bound method, Constrained pseudo-Boolean optimization problem.

1. Introduction

In the optimization model construction, many problems are naturally formalized as pseudo-Boolean optimization problems. A typical formulation of a pseudo-Boolean optimization problem is as follows. Let $X = (x_1, \dots, x_n)$ be a set of n independent binary variables and $f(X)$ be a real-valued function to be optimized: $f : S \to \mathbb{R}$, where $S \subset \{0,1\}^n$ is a subregion of Boolean variables space defined by a given system of constraints imposed on the values of variables X.

If $S = \{0,1\}^n$, that is, no constraints are imposed on the choice of variables $x_1, \dots, x_n$ then such a problem is called an unconstrained pseudo-Boolean optimization problem. For its solution, in [2], exact algorithms based on the detection of the optimized function behavioural features in binary variables space are worked

Received March 06, 2018; accepted June 11, 2018
2010 *Mathematics Subject Classification.* Primary 90C57; Secondary 90C27, 90C09
*Results were obtained in the framework of the state task 2.5527.2017/8.9 of the Ministry of Education and Science of the Russian Federation

out. These features were used to construct and justify effective exact algorithms. In particular, an exact optimization algorithm demanding $n+1$ calculations of a function was developed for a strictly monotone pseudo-Boolean function.

A special feature of these algorithms is that they do not require algebraic definition of an objective function. It may be so called a black-box function. We can calculate the value of the function in points $\{0,1\}^n$ only. The issue of constructing algorithms for solving pseudo-Boolean optimization problems with black-box functions are considered in this paper.

The most "primitive“ way to find the exact solution of a pseudo-Boolean optimization problem is to search all possible combinations of values of binary variables. The number of such combinations is equal to 2^n. For a lot of real problems it is unacceptable. To decrease the number of calculations, unpromising combinations of values of variables (such combinations form the subregions of the original space of binary variables) should not be considered. But to reveal them it is necessary to know the properties of the function, that is, the behaviour of the function on the points (combinations of variables). Such approaches as the dynamic programming method [6] and the branch-and-bound method [17] are based on the exclusion of sets of unpromising alternatives.

Many practical problems of choice are formalized as pseudo-Boolean optimization problems with constraints on choosing a combination of the variables; in this case in the behaviour of the objective function and constraints there are peculiarities which allow one to construct acceptable algorithms to find the exact solution. It is the problem of construction of such algorithms for a widespread class of problems that is considered in this paper.

In this paper, we propose an algorithm for solving problems of conditional pseudo-Boolean optimization based on the branch-and-bound scheme and using properties of functions of the optimization model for estimating upper bounds and eliminating unpromising solutions. The branch-and-bound method was originally developed to solve integer linear programming problems [17]. Then, based on this scheme, algorithms were developed to solve special classes of problems, such as nonlinear programming problems [1, 15, 25, 26], the traveling salesman problem [11, 21, 22], facility location [10, 20, 24], network design [8, 14, 16].

In addition, various modifications of the original branch-and-bound algorithm have been developed, combining the branch-and-bound principles with other techniques, such as cutting planes [11, 19, 21, 26], column generation [5, 9, 12], genetic and evolutionary algorithms [7, 13, 23].

2. Problem statement and basic notions

2.1. Constrained pseudo-Boolean optimization problem

Let us consider the problem of the following form:

$$C(X) \to \max_{X \in B_2^n}, \tag{2.1}$$

$$A_j(X) \leqslant H_j, j = 1, \ldots, m, \tag{2.2}$$

where $B_2^n = \{0,1\}^n$ is a space of binary variables, $C(X)$ and $A_j(X)$ are pseudo-Boolean functions (real-valued functions of binary variables) which are generally defined implicitly (algorithmically).

To describe the proximity of the vectors (points in space B_2^n), we shall apply the notion of neighborhood [2]. Two points $X_1, X_2 \in B_2^n$ are *k-neighboring* if they differ in values of k coordinates. Let the set of all points k-neighbouring to point X be called the level of some point X and denoted as $O_k(X)$. The level $O_1(X)$ can be presented as the neighborhood of point X.

Point $X^* \in B_2^n$ is called the *local minimum* of the pseudo-Boolean function f, if $f(X^*) < f(X)$ for all $X \in O_1(X^*)$. The notion of a local maximum is introduced similarly. If a function has the only point of a local minimum (maximum), it is often called *unimodal*.

In many works devoted to pseudo-Boolean functions special classes of functions with definite properties are considered. In this paper we shall consider a class of monotone functions which are rather often met in practical problems.

The unimodal function f is called *monotone* on B_2^n if for each $X^k \in O_k(X^*)$ $(k = 1, \ldots, n)$ the following condition is met: $f(X^{k-1}) \leqslant f(X^k)$ for all $X^{k-1} \in O_{k-1}(X^*) \cap O_1(X^k)$, where X^* is a local minimum of the function. That is, a function is a monotone one if it does not decrease while moving away from the point of minimum. If a sign of inequality is strict, then the function is strictly monotone.

It is easy to show that if the function is strictly monotone then the only points of local minimum and maximum differ in the value of n coordinates.

Let us take two points $Y, Z \in B_2^n$ the values of some coordinates in which coincide: $y_i = z_i, i \in A \subset \{1, \ldots, n\}$; $y_j \neq z_j, j \notin A$. Let a set of all points X the values of whose variables with $i \in A$ index are fixed and equal to $x_i = y_i = z_i$ and the values of all the rest variables can take any values, be called a *subcube* $K(Y, Z)$ (Figure 2.1). In [2] subcube $K(Y, Z)$ is introduced as the union of the shortest paths from Y to Z.

2.2. Properties of a set of feasible solutions

Let us introduce some notions for points placed in a binary space in a particular way [4].

- Point $Y \in A$ is a *boundary point* of set A if $\exists X \in O_1(Y)$, such that $X \notin A$.
- Point $Y \in O_i(X^0) \cap A$ is called a *limiting point* of set A with reference point $X^0 \in A$ if $X \notin A$ for any $X \in O_1(Y) \cap O_{i+1}(X^0)$ (Figure 2.2).

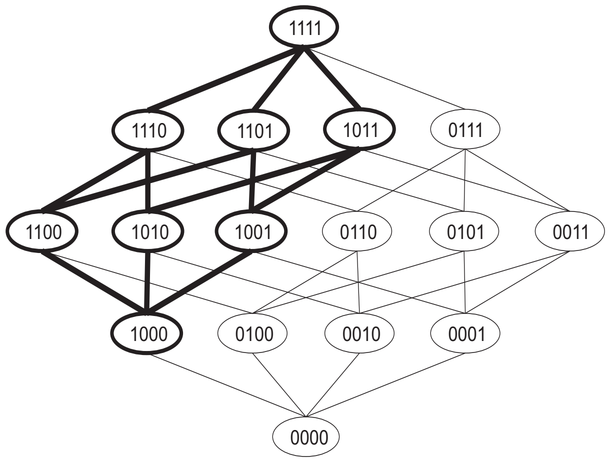

FIG. 2.1: An example of a subcube in the binary space

- Let the constraint which determines the subregion of Boolean variables space be called *active* if the optimal solution of the constrained optimization problem does not coincide with the optimal solution of a corresponding optimization problem without regard to the constraint. In other words, a constraint is active if the optimal solution of an unconstrained problem is unfeasible for a problem with the constraint.

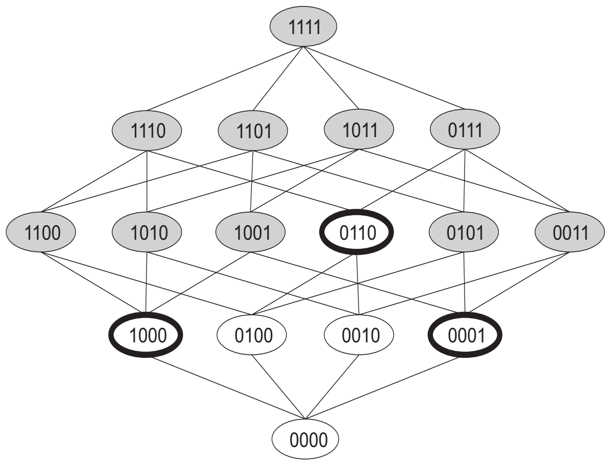

FIG. 2.2: An example of limiting points

One of the properties of a feasible set of solutions looks like this:

Let us consider the problem (2.1)-(2.2). If the objective function is a monotone unimodal function and the constraint is active, then the optimal solution of the problem will be the point belonging to the subset of limiting points of the set S of feasible solutions with reference point X^0 in which the objective function possesses the minimum value:

$$C(X^0) = \min_{X \in B_2^n} C(X).$$

Also it is not difficult to show that if the constraint function (2.2) is an unimodal pseudo-Boolean function then the set of feasible solutions S of the problem is a connected set.

3. Class of monotone pseudo-Boolean functions

Let us consider a class of problems of the following form

$$C(X) \to \max_{X \in B_2^n},$$
$$A(X) \leqslant H,$$

where the objective function $C(X)$ and the function $A(X)$ determining the system of constraints belong to the class of monotone pseudo-Boolean functions.

Let us note some properties of classes of unimodal and monotone pseudo-Boolean functions which form the considered class of problems. In optimization algorithms construction it is necessary to take these properties into account.

First of all, let us consider a following property that will be used later on. On the basis of the definitions of a subcube and monotonicity of a pseudo-Boolean function it can be argued that if a function f increases steadily from $X^0 \in B_2^n$ then for any point $Y \in B_2^n$ is fulfilled:

a) $f(X) \leqslant f(Y)$ for all $X \in K(X^0, Y)$;

b) $f(X) \geqslant f(Y)$ for all $X \in K(Y, X^1)$, where $X^1 = (1 - x_1^0, \ldots, 1 - x_n^0) \in O_n(X^0)$.

3.1. Properties of constraint functions

Unimodal constraint function

Let us consider a constraint function $A(X)$ which has the unique minimum in the point $X^0 \in B_2^n$. Let us denote $X^1 \equiv X \in O_n(X^0)$.

As it was noted above, a set of feasible points in this case is a connected set.

The main property which follows from the definitions introduced above is:

If the function $A(X)$ is a unimodal one (it has the unique local minimum in the point X^0) and on a level $O_k(X^0)$ all points are unfeasible or limiting, then on a level $O_l(X^0)$ where $l > k$ there are no feasible points. It can be illustrated with the following picture (Figure 3.1).

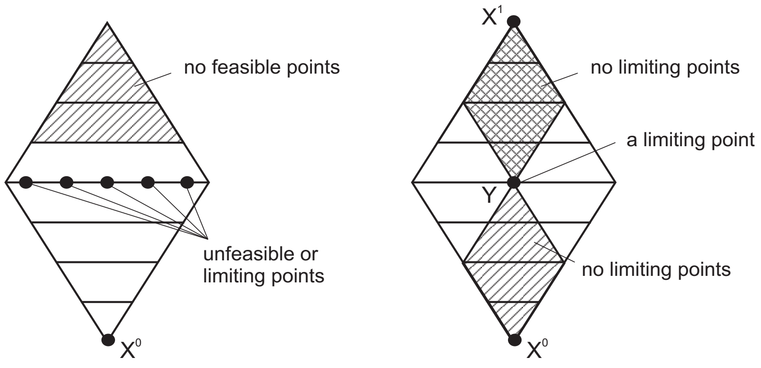

FIG. 3.1: Case of an unimodal constraint function (left) and case of a monotone constraint function (right)

2. Monotone constraint function

Let us consider a constraint function which increases steadily from the point $X^0 \in B_2^n$. On the basis of notions of a subcube and monotonicity we can deduce the following properties:

a) If the function $A(X)$ is monotone and a point $Y \in B_2^n$ is feasible (satisfies the constraint $A(Y) \leqslant H$) then any point $X \in K(X^0, Y)$ is also feasible.

b) If the function $A(X)$ is monotone and a point $Y \in B_2^n$ is unfeasible (doesn't satisfy the constraint $A(Y) \leqslant H$) then any point $X \in K(Y, X^1)$ is also unfeasible.

Generalizing these properties and the notion of a limiting point one can conclude that if the function $A(X)$ is monotone and a point $Y \in B_2^n$ is limiting then any point $X \in K(X^0, Y) \setminus Y$ is not limiting, and any point $X \in K(Y, X^1) \setminus Y$ is not limiting either (that is, while looking for all the other limiting points the subcubes $K(X^0, Y)$ and $K(Y, X^1)$ can be excluded from consideration.

3.2. Properties of objective functions

1. Unimodal objective function

If f is an unimodal function on B_2^n with the local minimum point X^0 then

$$\min_{X_j^k \in O_k(X^0)} f(X_j^k) \leqslant \min_{X_j^{k+1} \in O_{k+1}(X^0)} f(X_j^{k+1}).$$

This implies that if the function $C(X)$ is unimodal (it has the unique local maximum in the point X^1) and the solution giving the maximum value of the function $C(X)$ on a level $O_k(X^1)$ is feasible then on a level $O_l(X^1)$ where $l > k$ there is no the optimal solution (Figure 3.2).

2. Monotone objective function

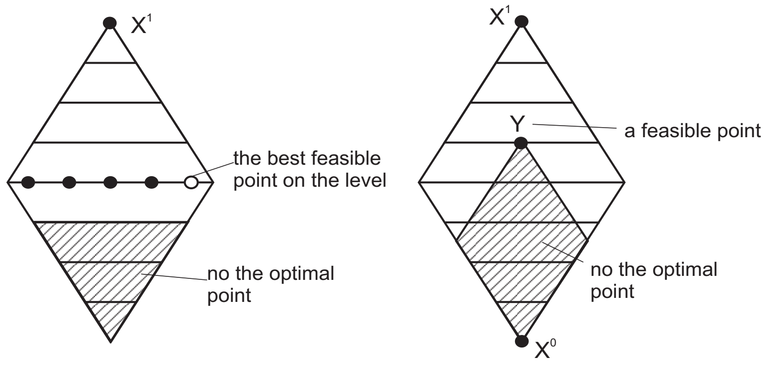

Fig. 3.2: Case of an unimodal objective function (left) and case of a monotone objective function (right)

If the objective function $C(X)$ increases steadily from the point $X^0 \in B_2^n$ then the optimal solution belongs to the subset of limiting points. From the property considered at the beginning of this unit we have the following.

If the function $C(X)$ is monotone and the solution $Y \in B_2^n$ is feasible (satisfies the constraint $A(Y) \leqslant H$) then in the subcube $K(X^0, Y)$ there is no the optimal solution.

4. A scheme of the branch-and-bounds method for a problem with black-box functions

The basis of the branch-and-bounds method is the idea of sequential partition of a set of feasible solutions into subsets. At each step of the method the elements of partition are checked to find out whether the given subset contains an optimal solution. The check is carried out by means of calculating the upper bound for an objective function on a given subset. If the upper bound is not better than the *record* – the best of the found solutions – then the subset can be discarded. A checked subset can be also discarded if the best solution was found in it. If the value of the objective function on a found solution is better than the record then the record is changed. On finishing the algorithm work the record is the result of its work.

If one manages to discard all elements of partition then the record is the optimal solution of the problem. Otherwise the most promising subset (for example, with the greatest value of the upper bound) is chosen from those which were discarded, and it is partitioned. New subsets are checked again, and so on.

It is obvious that the use of specific structural peculiarities of the problem allows one to construct a workable branch and bound algorithm.

Let us consider the application of the scheme for the solution of optimization problem in which all variables are binary, and the objective function and the constraint are unimodal and monotone.

The most widely used variant of application of the branch-and-bounds scheme for the solution of pseudo-Boolean optimization problems is the following. The problem of continuous optimization which is relaxation of the original problem is being solved (for example, with a simplex algorithm). As a result we have solution X^*, which will not be binary in general. Then the problem is divided into two subproblems and two mutually exclusive constraints exhausting all possibilities are added. For example, let component x'_i in X^* be not binary. Then constraints $x'_i = 0$ and $x'_i = 1$ appear in corresponding subproblems. Further branching occurs similarly.

Such an approach is suitable for problems in which the objective function and the constraints are defined explicitly (in the form of algebraic expressions). But the problem under consideration consists of functions defined algorithmically (black-box functions), that is, it is impossible to calculate the value of the function in the point which is not binary. Therefore there appeared the necessity to investigate other variants of application of the scheme.

Here we shall consider the question of application of the branch-and-bounds scheme for optimization problems in which the objective function and the constraints are defined algorithmically. Namely, for the problem (2.1)-(2.2), in which the functions $C(X)$ and $A(X)$ increase monotonically from the point $X^0 = (x^0_1, \dots, x^0_n)$.

The simplest algorithm of the branch-and-bounds method based on the properties of the considered class of problems will look like this. In the first stage of branching set B^n_2 is partitioned into two equicardinal subsets: $S^0_1 = \{X \in B^n_2 : x_1 = 0\}$ and $S^1_1 = \{X \in B^n_2 : x_1 = 1\}$ (let's call it branching of the first order). Each of these subsets is a subcube of dimension $n-1$, the cardinality of the subsets is 2^{n-1}. Partition of elements of the space B^n_2 for $n = 4$ into two subcubes is shown on Figure 4.1. In the next stages of branching each of subcubes is partitioned into two subcubes and so on. For example, S^0_1 is partitioned into $S^{00}_2 = \{X \in B^n_2 : x_1 = 0, x_2 = 0\}$ and $S^{01}_2 = \{X \in B^n_2 : x_1 = 0, x_2 = 1\}$ (Figure 4.2). So, after branching of the k-th order there appear subcubes consisting of 2^{n-k} elements (vectors).

In the subset got after branching of the k-th order k coordinates are fixed for any binary vector from this subset. Let the vector in which variable coordinates are equal to corresponding coordinates of initial vector X^0 be called *"lower" point* $\underline{X}$ and the vector in which all variable coordinates are opposite to corresponding coordinates of X^0 be called *"upper" point* $\overline{X}$:

$$\overline{X} = (x_1, \dots, x_k, 1 - x^0_{k+1}, 1 - x^0_{k+2}, \dots, 1 - x^0_n),$$
$$\underline{X} = (x_1, \dots, x_k, x^0_{k+1}, x^0_{k+2}, \dots, x^0_n).$$

The objective function and the constraint function increase monotonically on B^n_2 with chosen initial point $X^0 = (x^0_1, \dots, x^0_n)$ from where it follows that in the

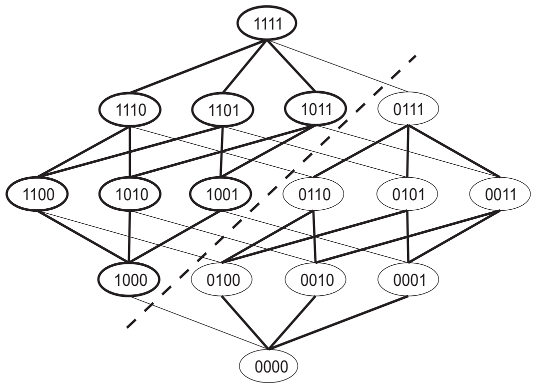

FIG. 4.1: Partition B_2^4 into two subcubes

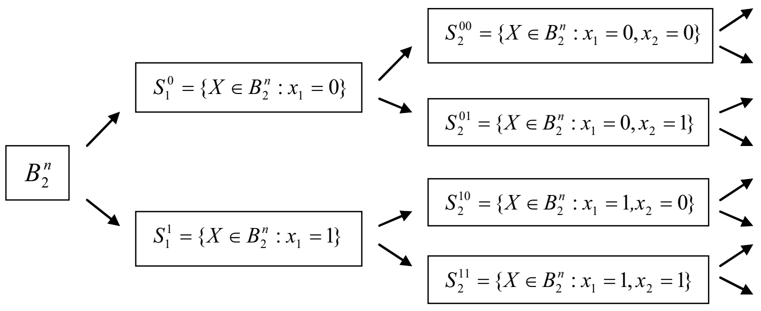
B_2^n
$S_1^0 = \{X \in B_2^n : x_1 = 0\}$
$S_1^1 = \{X \in B_2^n : x_1 = 1\}$
$S_2^{00} = \{X \in B_2^n : x_1 = 0, x_2 = 0\}$
$S_2^{01} = \{X \in B_2^n : x_1 = 0, x_2 = 1\}$
$S_2^{10} = \{X \in B_2^n : x_1 = 1, x_2 = 0\}$
$S_2^{11} = \{X \in B_2^n : x_1 = 1, x_2 = 1\}$

FIG. 4.2: The scheme of branching

"upper" point of the subcube they take the greatest value, and in the "lower" point the smallest one.

The subset (subcube) is excluded from consideration in three cases:

1. In point $\underline{X}$ the constraint is not performed; in this case all solutions in the subset are unfeasible.

2. In point $\overline{X}$ the constraint is performed; then this solution is the best one in the subset, and it is compared with the record.

3. In point $\overline{X}$ the constraint is not performed, but the objective function in it takes the value which is smaller than the record.

Otherwise further branching of this subset takes place.

At the first stage the value of the objective function in any feasible point of the space B_2^n can be taken as the record. If in the considering subset $K(\underline{X}, \overline{X})$ the point $\overline{X}$ is feasible and the value of the criterial function in it is greater than the record then the record is changed.

It is easy to show that the received solution will be exact. The constraint $A(X) \leqslant H$ partitions the set B_2^n into two subsets one of which satisfies the constraint and the other does not. From the condition of monotonicity of functions C (X) and A (X) it follows that the solution of the problem will be the point belonging to the subset of limiting points.

In case 1 in subcube $K(\underline{X}, \overline{X})$ there are no limiting points. In case 2 only point $\overline{X}$ in the subcube can be limiting. In case 3 there are limiting points in the subcube, but they are worse than those found before. So the scheme provides exact solution of the problem.

The considered approach allows one to easily calculate the lower bound which is equal to the value of the objective function in the upper point of the subcube.

Though the approach described above offers considerable reduction of the number of points to be searched in the process of finding the optimal solution nevertheless this process is labour intensive as it may require a great deal of branching.

The next part of the paper describes the optimization algorithm combining the schemes of the branch-and-bounds method and the rule of subcubes truncation considered in the previous part.

5. The optimization algorithm

Let us consider the problem (2.1)-(2.2) in which functions $C(X)$ and $A(X)$ monotonically increase from point $X^0 = (x_1^0, \dots, x_n^0)$. Let's denote $X^1 = O_n(X^0)$, $X^1 = (x_1^1, \dots, x_n^1)$. All set of points of the space B_2^n can be presented as the subcube $K(X^0, X^1)$.

5.1. Branching

Let us suppose that some limiting point $X' \in B_2^n$ is found. Then subcubes $K(X', X^0)$ and $K(X', X^1)$ can be excluded from further consideration.

Let us introduce an auxiliary variable

$$z_i = \begin{cases} x_i, & \text{if } x_i^0 = 0, \\ \bar{x}_i, & \text{if } x_i^0 = 1. \end{cases}$$

Then subcube $K(X', X^0)$ can be represented as a set of points for which the following boolean expression is true:

$$T^0 = \bigwedge_{i:x_i'=x_i^0} \bar{z}_i.$$

And subcube $K(X', X^1)$ can be described as follows

$$T^1 = \bigwedge_{i:x_i'=x_i^1} z_i.$$

For the sake of convenience let's denote a set of indexes for which $x'_i = x^1_i$ is fulfilled as $A(X') = \{i_1, \ldots, i_k\}$ and a set of indexes for which $x'_i = x^0_i$ is fulfilled as $B(X') = \{i_1, \ldots, i_{n-k}\}$. It is obvious that $|A(X')| = k$ and $|A(X')| = n - k$ where k is the number of the level on which point $K(X') \in O_k(X^0)$ is located. Then we may write:

$$T^0 = \bigwedge_{i \in B(X')} \bar{z}_i, \quad T^1 = \bigwedge_{i \in A(X')} z_i.$$

Let us partition subcube $K(X^0, X^1)$ into two parts:

$$\begin{tikzcd}
 & K(X^0, X^1) \arrow[dl] \arrow[dr] & \\
(T^0 = 1) \vee (T^1 = 1) & & (T^0 = 0) \wedge (T^1 = 0)
\end{tikzcd}$$

The left part, as it was stated above, is excluded from further consideration. The right part $(T^0 = 0) \wedge (T^1 = 0)$ can be represented as a set of subcubes.

Let us consider the condition $(T^1 = 0)$. It is fulfilled if $z_i = 0$ for at least one $i \in A(X')$. If $|A(X')| > 1$ then a set of points fulfilling the condition $(T^1 = 0)$ can be represented only as a number of subcubes, but not as one subcube. The most evident way to partition this set of points into k subcubes is to fix alternately the value of variable $z_i = 0$ for $i \in A(X')$. In this case we receive k subcubes of dimension $n - 1$. The disadvantage of this method is that the received subcubes substantially intersect each other.

To avoid this we shall use the following approach. We shall get the first subcube K^1_1 having fixed one variable $z_{i_1} = 0$. For the second K^1_2 we shall fix two variables : $z_{i_1} = 1$ and $z_{i_2} = 0$. For the third K^1_3 - three variables: $z_{i_1} = 1$, $z_{i_2} = 1$ and $z_{i_3} = 0$. And so on. For the k-th subcube K^1_k: $z_{i_s} = 1, s = 1, \ldots, k - 1$, $z_{i_k} = 0$.

As a result we get k subcubes of different dimensions. Such an approach guarantees that the received subcubes don't intersect.

The same procedure is offered for condition $(T^0 = 0)$. The corresponding set of points should be partitioned into $(n - k)$ subcubes by fixing variables $j \in B(X')$. For the first subcube K^0_1 we shall fix one variable $z_{j_1} = 1$. For the second subcube K^0_2 we shall fix two variables: $z_{j_1} = 0$ and $z_{j_2} = 1$. For the third K^0_3 - three variables: $z_{j_1} = 0$, $z_{j_2} = 0$ and $z_{j_3} = 1$. For $(n - k)$-th subcube K^0_{n-k}: $z_{j_s} = 0, s = 1, \ldots, n - k - 1$, $z_{j_{n-k}} = 0$.

As a result we get two sets of subcubes: $K^1_1, \ldots, K^1_k$ and $K^0_1, \ldots, K^0_{n-k}$. A set of points fulfilling the condition $(T^0 = 0) \wedge (T^1 = 0)$ corresponds to the union of all possible intersections of pairs of subcubes taken from these two sets:

$$\bigcup_{\substack{i \in A(X') \\ j \in B(X')}} (K^1_i \cap K^0_j).$$

So, having found in subcube $K(X^0, X^1)$ some limiting point $X' \in O_k(X^0)$ we partition this subcube into two parts one of which is discarded (subcubes $K(X', X^0)$

and $K(X', X^1)$), and from the other part $k \cdot (n-k)$ new branches are formed. Each of these branches is a subcube which can be subjected to the same procedure of branching as described above.

5.2. Upper bound

Let us denote the upper and the lower points of some subcube as $\overline{X}$ and $\underline{X}$ respectively. $z_i^{\underline{X}} = 0$ is fulfilled in point $\underline{X}$ for all free (unfixed) variables, and $z_i^{\overline{X}} = 1$ is fulfilled in point $\overline{X}$ for all free variables. For fixed variables naturally $z_i^{\underline{X}} = z_i^{\overline{X}}$.

Subcube $K(\underline{X}, \overline{X})$ can contain an optimal solution only if the following conditions are fulfilled:

1. There are feasible solutions in the subcube.
2. The upper bound of the corresponding branch is above the best found solution.

As the constraint function $A(X)$ increases monotonically from point X^0, then within subcube $K(\underline{X}, \overline{X})$ function $A(X)$ increases monotonically from point $\underline{X}$ possessing its minimum value in this point. Therefore if point $\underline{X}$ is unfeasible then all points of this subcube are unfeasible. The objective function $C(X)$ within the subcube also increases monotonically from point $\underline{X}$ possessing its maximum value in point $\overline{X}$. Point $\overline{X}$ itself can be unfeasible, but value $C(\overline{X})$ can be used as the upper bound of the branch corresponding to the subcube.

Also, if point $\overline{X}$ is feasible then all other points of this subcube are a fortiori not better; besides, in this case there are no limiting points in the subcube with the possible exception of $\overline{X}$.

So, subcube $K(\underline{X}, \overline{X})$ is excluded from further search if at least one of the following conditions is fulfilled:

- Point $\underline{X}$ is unfeasible.
- Point $\overline{X}$ is feasible.
- The value of upper bound $C(\overline{X})$ does not exceed the already found best feasible value of the objective function.

Such a check including calculation of the upper bound requires the scanning of only two points of the subcube.

5.3. Search for limiting points

To carry out branching in a way described above it is necessary to find some limiting point belonging to the considered subcube $K(\underline{X}, \overline{X})$. This solution should not necessarily be the best one in the given subcube. However, a good solution can exceed the record (the best already found feasible solution) and it also can increase the chances to discard new branches got in the process of further branching (if their upper bound will turn out to be lower).

The simplest stochastic algorithm for search of limiting points is as follows. The search begins from $\underline{X}$. At each step the algorithm chooses a feasible neighbouring point on the following level moving along the way of increasing of the objective function to the bound of a feasible area. In case of necessity the procedure is repeated several times and the best point is chosen from the found limiting points.

Algorithm "Random search"

1. Suppose $l = 1$.
2. Suppose $X_1 = \underline{X}$, $i = 1$.
3. Randomly choose a point $X_{i+1} \in O_1(X_i) \cap O_i(\underline{X}) \cap \{X \in K(\underline{X}, \overline{X}) : A(X) \leqslant H\}$, $i = i + 1$. If there are no such points go to step 4, otherwise the cycle is repeated.
4. $Y_l = X_i$. If $l < L$ then $l = l + 1$ and go to step 2.
5. Define X^* from the condition

$$C(X^*) = \max_{l=1,\dots,L} C(Y_l).$$

Defined number L is a number of limiting points which is planned to find. As $card\{O_1(X_k) \cap O_{k+1}(\underline{X})\} = n_K - k$, where $X_k \in O_k(X)$, n_K is the dimension of subcube $K(\underline{X}, \overline{X})$ (the number of free variables) then from current search point X_k the algorithm looks through not more than $n_K - k$ following points. So the computational complexity of the algorithm can be calculated as follows

$$T \leqslant L \cdot \sum_{i=0}^{n-1} (n_K - i) = L \cdot \frac{n_K(n_K + 1)}{2}.$$

A regular algorithm using greedy heuristics is an alternative to random search of limiting points.

Greedy algorithms are natural heuristics in which at each step the most effective at the given moment decision is made without considering what happens at the following steps of search.

For the problem being considered a greedy algorithm can have the following form.

Algorithm "Greedy"

1. Suppose $X_1 = \underline{X}$, $i = 1$.
2. Calculate $C(X_j)$ and $A(X_j)$ for $X_j \in O_1(X) \cap O_i(\underline{X})$, $j = 1, \dots, n_K - i + 1$.

3. If there is no X_j for which $A(X_j) \leqslant H$, then $X^* = X$ is the solution of the problem.

4. From those X_j for which $A(X_j) \leqslant H$ find $X = \arg\max_{X_j} \lambda(X_j)$.

5. $i = i + 1$, go to step 2.

Here $\lambda(X_j) = C(X_j)/A(X_j)$ or $\lambda(X_j) = C(X_j)$.

In more detail these and other algorithms of search of limiting points have been considered in [3].

5.4. The algorithmic scheme

The procedures described above are the main elements of which an algorithm of search of an optimal solution consists. Now we shall consider the algorithm itself.

The first step is the choice of a branch for branching. In the first cycle there is only one branch which corresponds to the binary space of n dimension. In the following cycles, when there are several open branches, the branch with the maximum upper-bound estimate is chosen. If there are no open branches then the algorithm finishes its work.

At the second step the search for an approximate solution representing some limiting point in the corresponding subcube in a chosen branch is carried out. If the value of the objective function in this point is better than the record (the best found solution) then the change of the record occurs. The search of an limiting point can be carried out with the help of, for example, a random-search algorithm or a greedy algorithm described above.

At the third step the procedure of branching of a chosen branch according to the found limiting point is performed. The check of received branches is carried out. If there are feasible solutions in a branch and the upper bound is greater than the record then this branch is added to the list of open branches.

After completing some number of such cycles one should interrupt in order to sort the branches by the value of the upper bound and close the branches the upper bound of which is less than the record.

The search is stopped if there are no open branches left. In this case it can be argued that the exact solution of the problem (global constrained maximum) is found.

While solving the problems of great dimensions it can be inaccessible due to excessively large amount of search time. The achievement of a number of formed branches or a number of branching of some defined value can serve as a stopping criterion.

The algorithm is shown schematically in Figure 5.1.

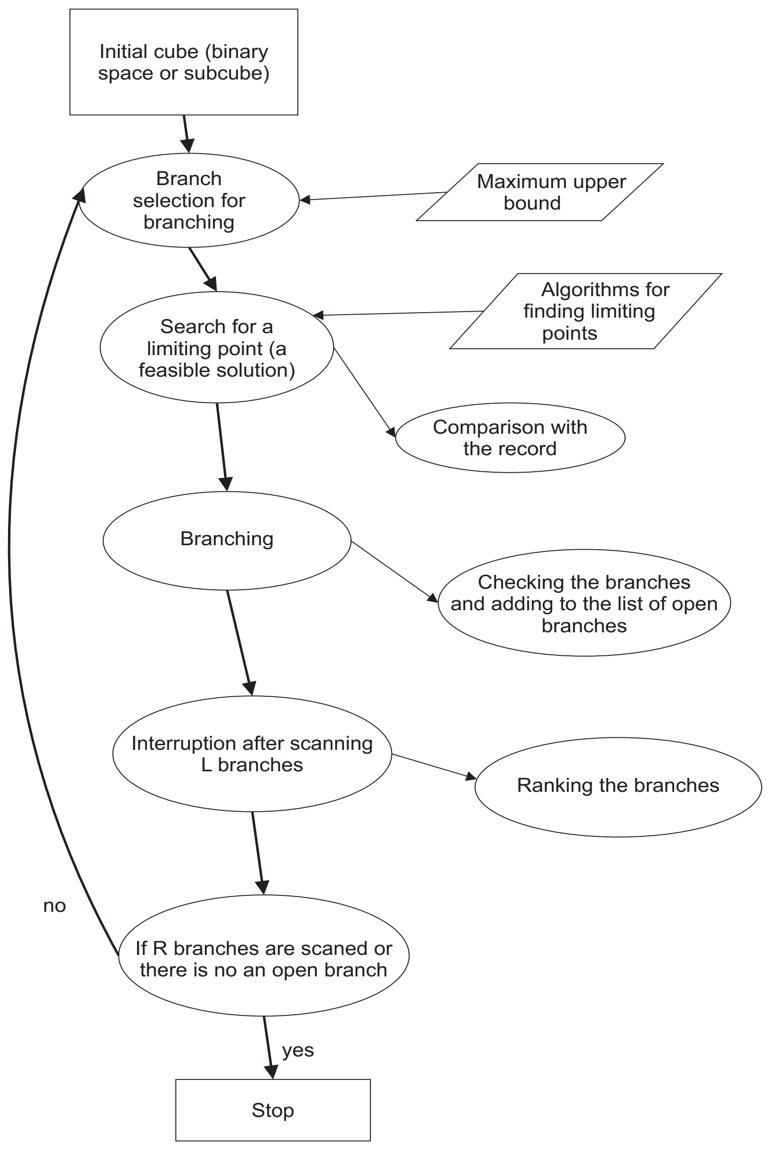
Initial cube (binary space or subcube)
Branch selection for branching
Maximum upper bound
Algorithms for finding limiting points
Search for a limiting point (a feasible solution)
Comparison with the record
Branching
Checking the branches and adding to the list of open branches
Interruption after scanning L branches
Ranking the branches
no
If R branches are scaned or there is no an open branch
yes
Stop

FIG. 5.1: The algorithmic scheme

6. Experimental investigation

This paper gives the results of the experimental investigation of the described algorithm work on the constrained pseudo-Boolean optimization problems generated randomly. Objective functions and constraints have the following form:

$$C(X) = \sum_{i=1}^{n} c_1^i x_i + \sum_{i=1}^{n-1} c_2^i x_i x_{i+1} + \sum_{i=1}^{n-2} c_3^i x_i x_{i+1} x_{i+2} \to \max,$$

$$A(X) = \sum_{i=1}^{n} a_1^i x_i + \sum_{i=1}^{n-1} a_2^i x_i x_{i+1} + \sum_{i=1}^{n-2} a_3^i x_i x_{i+1} x_{i+2} \leqslant b,$$

$$X \in \{0,1\}^n,$$

where coefficients $c_1^i, c_2^i, c_3^i, a_1^i, a_2^i, a_3^i$ are random numbers taken from the range $[0, 20]$; $b = A(X^r)$, where $X^r = (x_1^r, \ldots, x_n^r)$ is a randomly chosen point: $x_i^r = 1$ with probability $1/4$ and $x_i^r = 0$ with probability $3/4$ (this bias is made due to some real-world problems, in this case the set of feasible points is less than the set of infeasible points). As all coefficients are non-negative numbers then functions $C(X)$ and $A(X)$ are monotone ones with the minimum in point $(0, \ldots, 0)$ and unconstrained maximum in point $(1, \ldots, 1)$.

Efficiency of the algorithm will be characterized by the search time and achieved value of the objective function (if the maximum of the objective function is defined inexact or there is not proof that the solution is exact). By the search time (or the time complexity) we will mean the number of computing values of the objective function (and/or the constraint function) that the algorithm has made (the number of points that the algorithm has scanned).

At first let us investigate how fast the optimization algorithm finds an exact solution. For this purpose series of tests were conducted on the problems of small dimension: $n = 10, 15, 20$. 500 tasks were solved for each dimension value. A simple algorithm "random search" with repetition number $L = 1$ was used to find boundary points.

The distribution of values of time complexity and the number of branches that occur as a result of complete solution of the problem are shown in the graphs (Figure 6.1 and Figure 6.2). In the experiments it is guaranteed that the exact solution of the problem has been found (i.e. there are not open branches remained). For comparison, time complexity of the exhaustive search for the dimension $n = 10$ is $2^{10} = 1024$, for $n = 15$ complexity is $2^{15} = 32768$, for $n = 20$ complexity is $2^{20} = 1048576$.

Further the graphs (Figure 6.3 and Figure 6.4) present the distribution of the time complexity and the number of branches when the exact solution has been found, but absence of a better solution has not been guaranteed yet (i.e.there are open branches remained).

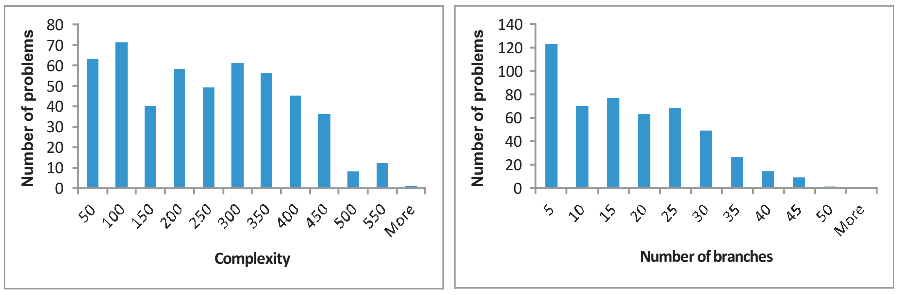

FIG. 6.1: Time complexity and number of branches for $n = 10$ (the exact solution is justified)

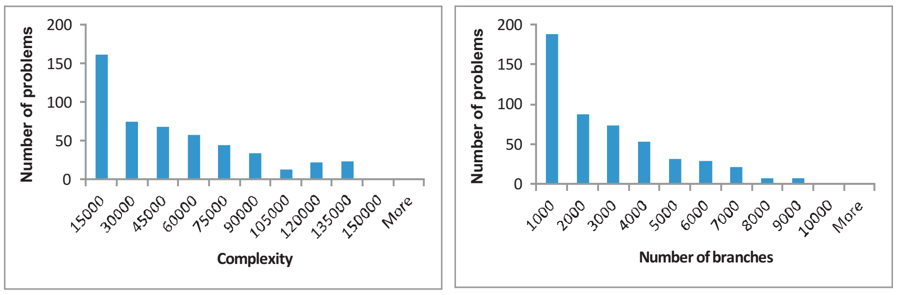

FIG. 6.2: Time complexity and number of branches for $n = 20$ (the exact solution is justified)

As can be seen from the graphs, the exact solution is found usually well before completion of the full search.

It should be noted that the results of the solutions of problems generated in such a way differ greatly from problem to problem therefore it doesn't make sense to give the mean values of efficiency indexes. Instead the results of solution of separate problems are given here what in this case is more demonstrative.

During the search the number of open branches is being changed significantly. The first few cycles it is increased rapidly and towards the end of the search it is gradually reduced, approaching zero. Absence of open branches upon completion of the search means that the solution is exact, that is there is no a better feasible solution under the given conditions.

The pictures (Figure 6.5, Figure 6.6 and Figure 6.7) show examples for the dynamics of change in the number of open branches and the value of the record in the process of search. The number of branching is shown on X-axis, the amount of open branches on the left of Y-axis, the value of the record on the right of Y-axis.

Also here was examined the question of how the proposed optimization algo-

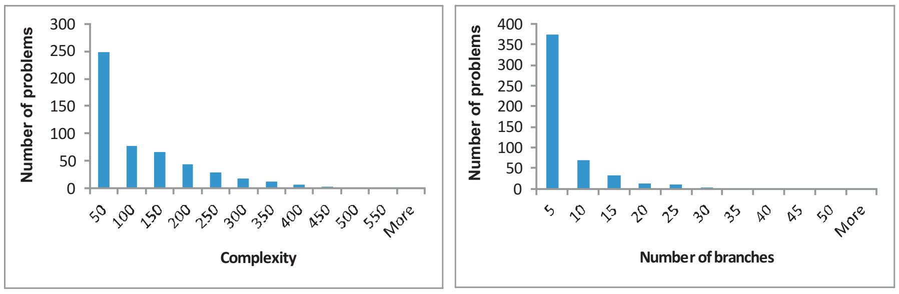

FIG. 6.3: Time complexity and number of branches for $n = 10$ (the exact solution is not justified)

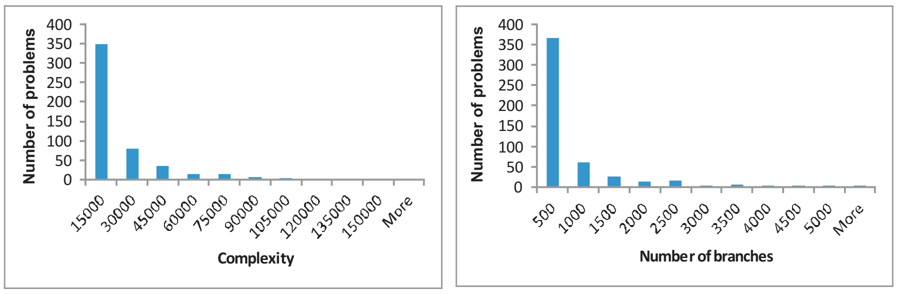

FIG. 6.4: Time complexity and number of branches for $n = 20$ (the exact solution is not justified)

rithm can improve the solution obtained individually by the algorithm of search for boundary points (the greedy heuristic or random search of boundary points).

To find the first approximate solution the greedy optimization algorithm described above was used. The obtained solution was used as a branching point in accordance with the procedure described above for branching the optimization space. To find solutions in the formed branches also the greedy algorithm was used. Values of the best solutions found during the search by the branch and bound algorithm presented in the tables.

Number of branching	Found solution	Time complexity
Greedy algorithm	47	34
1	47	82
2	56	141
4	58	202
7	73	281

Table 6.1: The greedy algorithm and the branch and bound algorithm, $n = 10$

Number of branching	Found solution	Time complexity
Greedy algorithm	31	90
1	52	216
2	53	341
4	66	492
8	67	862
19	68	1703
32	74	2645
88	78	5403
153	79	8840

Table 6.2: The greedy algorithm and the branch and bound algorithm, $n = 20$

Number of branching	Found solution	Time complexity
Greedy algorithm	74	165
1	74	407
2	76	707
5	89	1529
12	106	2850
34	108	6161
139	109	19937
183	111	24662
541	118	54243
589	119	58145
1515	123	126962

Table 6.3: The greedy algorithm and the branch and bound algorithm, $n = 30$

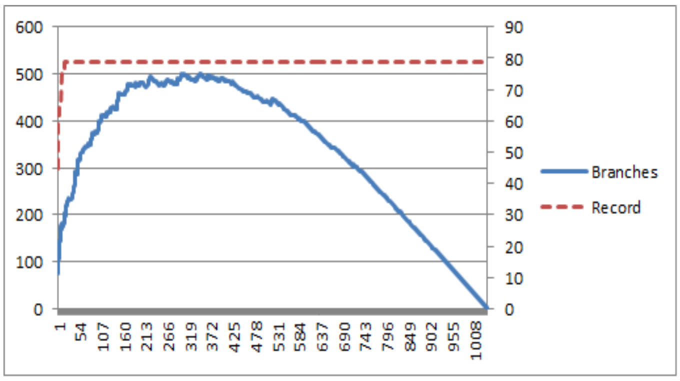

FIG. 6.5: Amount of open branches and value of the record for $n = 20$

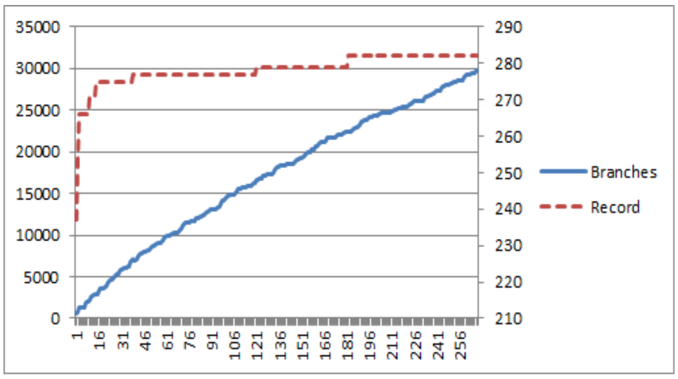

FIG. 6.6: Amount of open branches and value of the record for $n = 50$

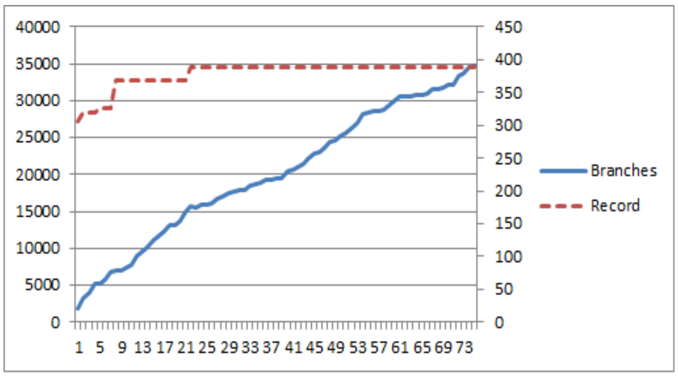

FIG. 6.7: Amount of open branches and value of the record for $n = 100$

Number of branching	Found solution	Time complexity
Greedy algorithm	342	1810
1	342	4874
3	371	12318
8	374	26684
35	388	88616
67	401	158873
70	406	163781

Table 6.4: The greedy algorithm and the branch and bound algorithm, $n = 100$

Number of branching	Found solution	Time complexity
Greedy algorithm	708	7380
1	743	20153
2	784	22181
4	798	35944
7	800	70062
11	803	112484
12	840	114011

Table 6.5: The greedy algorithm and the branch and bound algorithm, $n = 200$

7. Conclusions

The main peculiarity of the considered class of problems is that an objective function and constraint functions are supposed to be defined implicitly, that is, calculations of the functions in points are possible, but their algebraic notation is not known. On the one hand, such problems are often met in practice, for example, when it is necessary to turn to a data array to calculate a function. On the other hand, even for problems for which algebraic notation of functions is possible, these functions can be considered as algorithmically defined, which significantly simplifies the work with an available optimization model.

Such a class of models restricts the number of optimization algorithms available for application. Of course it is always possible to apply the local search algorithm or the algorithms of genetic type, but they do not guarantee finding of the exact solution, and one cannot say how close the found solution is to the optimal one.

At the same time in many practical problems objective functions and constraints have the same properties, such as unimodality and monotonicity. And these properties are not taken into account in application of universal algorithms.

The approach presented in this paper is aimed at the reception of an exact solution of an optimization problem. The realized way of branching divides a branch which represents a subcube of binary variables space into a great number of branches a significant part of which is at once subjected to exclusion. It offers quick reduction of the area in which an optimal solution can be found.

The developed algorithm can be also applied for the problems of high dimensionality. For all that, of course, it will not be proved that the found solution is an optimal one, if there are still unconsidered open branches. In this case such an algorithm can be considered as improvement of approximate algorithms of boundary points search, such as a greedy algorithm and random search of boundary points. Such improvement even on a small number of iterations (branchings) offers significant improvement of the found feasible solution.

From now on it is planned to investigate the work of the algorithm on practical problems: for example, on the problem of capacity planning, the problem of searching for rules in data in logical algorithms of classification. It is interesting to compare this algorithm with popular search algorithms such as local search with multi-start and algorithms of genetic type.

REFERENCES

1. S. Ahmed, M. Tawarmalani, N. V. Sahinidis: *A finite branch-and-bound algorithm for two-stage stochastic integer programs.* Mathematical Programming. **100(2)** (2004), 355–377.

2. A. N. Antamoshkin: *Regular Opimization of Pseudo-Boolean Functions.* Krasnoyarsk University Press, Krasnoyarsk, 1989 (in Russian).

3. A. N. Antamoshkin, I. S. Masich: *Heuristic search algorithms for monotone pseudo-boolean function conditional optimization.* Engineering and automation problems. **5(1)** (2006), 55–61.
4. A. Antamoshkin, I. Masich: (2007) *Pseudo-Boolean Optimization in Case of an Unconnected Feasible Set.* In: Models and Algorithms for Global Optimization (A. Törn, J. Žilinskas, eds.), Optimization and Its Applications, vol. 4. Springer, Boston, MA, 2007.
5. C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, P. H. Vance: *Branch-and-Price: Column Generation for Solving Huge Integer Programs.* Operations Research. **46(3)** (1998), 316–329.
6. R. E. Bellman: *Dynamic Programming.* Princeton University Press, Princeton, NJ, 1957.
7. C. Cotta, J. Troya: *Embedding Branch and Bound within Evolutionary Algorithms.* Applied Intelligence. **18** (2003), 137–153.
8. F. B. Cruz, G. R. Mateus, J. M. Smith: *A Branch-and-Bound Algorithm to Solve a Multi-level Network Optimization Problem.* Journal of Mathematical Modelling and Algorithms. **2(1)** (2003), 37–56.
9. J. Desrosiers, M. E. Lubbecke: *Branch-Price-and-Cut Algorithms.* Wiley Encyclopedia of Operations Research and Management Science, 2010.
10. M. A. Efroymson, T. L. Ray: *A branch and bound algorithm for plant location.* Operations Research. **14** (1996), 361–368.
11. M. Fischetti, A. Lodi, P. Toth: *Solving Real-World ATSP Instances by Branch-and-Cut.* Lecture Notes In Computer Science (2003), 64–77.
12. D. Feillet: *A tutorial on column generation and branch-and-price for vehicle routing problems.* 4OR. **8(4)**, 407–424.
13. J. E. Gallardo, C. Cotta and A. J. Fernandez: *A hybrid model of evolutionary algorithms and branch-and-bound for combinatorial optimization problems.* 2005 IEEE Congress on Evolutionary Computation, 2005, vol. 3, pp. 2248–2254.
14. O. Günlük: *A branch-and-cut algorithm for capacitated network design problems.* Mathematical Programming. **86(1)** (1999), 17–39.
15. E. R. Hansen: *Global optimization using interval analysis.* Marcel Dekker New York, 2004.
16. K. Holmberg, D. Yuan: *A Lagrangean Heuristic Based Branch-and-bound Approach for the Capacitated Network Design Problem.* Operations Research. **48(3)** (2000), 461–481.
17. A. H. Land, A. G. Doig: *An automatic method of solving discrete programming problems.* Econometrica **28** (1960), 397–520.
18. E. L. Lawler and D. E. Wood: *Branch and bounds methods: A survey.* Operations Research. **4(4)** (1966), 669–719.
19. J. E. Mitchell: *Branch and Cut.* Wiley Encyclopedia of Operations Research and Management Science, 2011.
20. R. M. Nauss: *An Improved Algorithm for the Capacitated Facility Location Problem.* The Journal of the Operational Research Society. **29(12)** (1978), 1195–1201.
21. M. Padberg, G. Rinaldi: *A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems.* SIAM Review. **33(1)** (1991), 60–100.

22. N. Pascheuer, M. Jünger, G. Reinelt. *A Branch and Cut Algorithm for the Asymmetric Traveling Salesman Problem with Precedence Constraints.* Computational Optimization and Applications. **17(1)** (2000), 61–84.
23. C. Pessan, J.-L. Bouquard, E. Néron: *Genetic Branch-and-Bound or Exact Genetic Algorithm?* In: Artificial Evolution (N. Monmarché, EG. Talbi, P. Collet, M. Schoenauer, E. Lutton, eds.), EA 2007. Lecture Notes in Computer Science, vol. 4926. Springer, Berlin, Heidelberg, 2008.
24. E. L. F. Senne, L. A. N. Lorena, M. A. Pereira: *A branch-and-price approach to p-median location problems.* Computers and Operations Research. **32(6)** (2005), 1655–1664.
25. M. Tawarmalani, N. V. Sahinidis: *Global optimization of mixed-integer nonlinear programs: A theoretical and computational study.* Mathematical Programming. **99(3)** (2004), 563–591.
26. D. Vandenbussche, G. L. Nemhauser: *A branch-and-cut algorithm for nonconvex quadratic programs with box constraints.* Mathematical Programming. **102(3)** (2005), 559–575.

Igor S. Masich
Siberian State University of Science and Technology
Department of Systems Analysis and Operations Research
prosp. Krasnoyarskiy Rabochiy, 31
660014 Krasnoyarsk, Russia
is.masich@gmail.com

Lev A. Kazakovtsev
Siberian State University of Science and Technology
Department of Systems Analysis and Operations Research
prosp. Krasnoyarskiy Rabochiy, 31
660014 Krasnoyarsk, Russia
levk@bk.ru