STUDIA Z AUTOMATYKI I INFORMATYKI
VOL. 41 – 2016

Michał Ignaczak, Dariusz Horla*

PERFORMANCE EVALUATION OF BASIC OPTIMIZATION METHODS FOR POLYNOMIAL BINARY PROBLEMS

Keywords: Binary polynomial problems, optimization, computational complexity

1. INTRODUCTION

Linear programming is applied in many decision-making problems [1]. There are, however, situations where linear formalism cannot be used. In addition, for problems that involve nonlinear functions, binary programming task is usually more difficult to solve and requires greater computational costs [6, 7].

The paper focuses on binary programming problems involving special case of nonlinear functions of polynomial type. The majority of problems presented in the paper can be described in a general form

$$\begin{aligned} \max_{\underline{x}} \quad & f(\underline{x}) \\ \text{s.t.} \quad & \gamma_j(\underline{x}) \leqslant \beta_j \quad (j = 1, \ldots, m)\,, \\ & \underline{x} \in \{0, 1\}^n\,, \end{aligned}$$

where f denotes the aim function, $\underline{x} \in \mathscr{R}^n$ is the vector of decision variables, $\underline{\gamma} : \mathscr{R}^n \to \mathscr{R}^m$ is a function that defines, possibly nonlinear, constraints.

Representation of an optimization problem using variables with binary values usually leads to heuristic approaches to solving it, what streamlines the solution. In the paper, the following algorithms are presented and evaluated:

- basic binary algorithm for unconstrained polynomial problems,
- branch and bound method,
- branch and bound method with a penalty function,
- cutting-planes method,
- binary knapsack problem with a quadratic aim function,
- branch and bound methods based on Lagrange relaxation.

In a less general case, for specific problems with binary variables, the aim function takes the form

$$f(\underline{x}) = \sum_{i=1}^{n} c_i x_i + \sum_{j \in \mathscr{N}} \left(a_j \prod_{i \in \mathscr{S}_j} x_i \right)\,, \tag{1}$$

*Poznan University of Technology, Institute of Control and Information Engineering, Department of Control and Robotics, Piotrowo 3a Str., 60-965 Poznan, e-mail: Dariusz.Horla@put.poznan.pl

where $\underline{c} \in \mathscr{R}^n$ forms the linear part of the aim function, $\mathscr{N} = \{1, \ldots, n\}$, a_j is a constant coefficient, and $\mathscr{S}_j$ defines a set of decision variables.

The main engineering problem is to formulate the aim function, and a subsequent task is to find appropriate algorithm to find its optimal value. The basic advantage of binary problems is the set of two values only that the decision variables can take on, what enables one to decrease the computational burden, e.g. positive powers of decision variables do not change their values or when calculating gradients its specific values are not important, since the problem is not continuous and thus only signs in gradient elements matter.

Aim functions used in the paper can be divided into specific forms:

- basic algorithm, branch and bound method, cutting-planes method, see (1),
- quadratic knapsack problem, and branch and bound method with Lagrange relaxation

$$f(\underline{x}) = \sum_{i=1}^{n} a_{ii} x_i + \sum_{1 \leqslant i < j \leqslant n} a_{ij} x_i x_j \,. \tag{2}$$

2. Basic algorithm for unconstrained binary polynomial problems

The solver dedicated to this problem is easy to implement, but its use is limited. The first problem is that partial derivatives of the aim function must be enumerated for every decision variable and whenever new aim functions are constructed, what leads to complexity increase [2, 3].

In this algorithm, and for every variable, a new subproblem is constructed that virtually replaces the problem from the previous iteration. This approach is mainly based on observing signs of partial derivatives to generate subproblems as functions of $\Phi_k(x_1, \ldots, x_k)$. The advantage of this approach is its versatility, and the fact that every subproblem can be easily solved.

Let the following aim function be given:

$$f(\underline{x}) = \sum_{j} a_j \left(\prod_{i} x_i \right) , \tag{3}$$

and $\Delta_i = \frac{\partial f_i}{\partial x_i} = f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) - f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)$.

A local minimum of f is composed of the following variables:

$$x_i = \begin{cases} 1 \text{ if } \Delta_i(\underline{x}) > 0 \\ 0 \text{ if } \Delta_i(\underline{x}) \leqslant 0 \end{cases} .$$

Let us now define the residual

$$\theta_i(\underline{x}) = f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) - x_i \Delta_i(\underline{x}) \,,$$

to express the aim function as

$$f(\underline{x}) = x_i \Delta(\underline{x}) + \theta_i(\underline{x}) \,.$$

The aim function of n variables can now be presented as

$$f_n(\underline{x}) = x_n g_n(x_1, \ldots, x_n) + h_n(x_1, \ldots, x_n)\,,$$

where:

$$\begin{aligned} g_n(\cdot) &= \frac{\partial f_n}{\partial x_n}\,, \\ h_n(\cdot) &= f_n(x_1, \ldots, x_{n-1}, 0)\,. \end{aligned}$$

Now, the following function can be created

$$\Phi_k(\cdot) = \begin{cases} 1 \text{ if } g_n(x_1, \ldots, x_{n-1}) > 1 \\ 0 \quad \text{otherwise} \end{cases}$$

and a new aim function can be created

$$f_{k-1}(x_1, \ldots, x_{n-1}) = \Phi_k(x_1, \ldots, x_{n-1}) g_n(x_1, \ldots, x_{n-1}) + h_n(x_1, \ldots, x_{n-1})\,.$$

As can be seen, $(k-1)$th iteration is connected to a single variable, which is eliminated from the aim function. The algorithm terminates when all variables are updated on the basis of appropriate derivatives. According to the above outline, the algorithm terminates when the aim function of a single variable is obtained, i.e. some $f_j(x_1)$, and

$$\begin{aligned} x_1^* = 1 \text{ if } f_1(1) > f_1(0)\,, \\ x_1^* = 0 \text{ if } f_1(0) \geqslant f_1(1)\,. \end{aligned}$$

The remaining elements of the optimal solution are $x_{i+1}^* = \Phi_k(x_1^*, \ldots, x_i^*)$.

The algorithm can be summarized as follows:

0) Initialization step

   Introduce:

$$\begin{aligned} f_n(\underline{x}) &= f(\underline{x})\,, \\ k &= n\,. \end{aligned}$$

1) Compute the following functions:

$$\begin{aligned} g_k(x_1, \ldots, x_{k-1}) &= \frac{\partial f_k}{\partial x_k}\,, \\ h_k(x_1, \ldots, x_{k-1}) &= f_k(x_1, \ldots, x_{k-1}, 0)\,, \\ \Phi_k(x_1, \ldots, x_{k-1}) &= \begin{cases} 1 \text{ if } g_k(x_1, \ldots, x_{k-1}) > 0 \\ 0 \quad \text{otherwise} \end{cases} \end{aligned}$$

2) Using g_k and h_k compute

$$f_{k-1}(x_1, \ldots, x_{k-1}) = \Phi_k(x_1, \ldots, x_{k-1}) g_k(x_1, \ldots, x_{k-1}) + h_k(x_1, \ldots, x_{k-1})\,.$$

3) For $k > 2$ put $k := k - 1$ and proceed to Step 1, otherwise set

$$x_1^* = 1 \text{ iff } f_1(1) > f_1(0)\,,$$
$$x_1^* = 0 \text{ iff } f_1(0) \geqslant f_1(1)\,,$$

and for $i = 1, \ldots, n-1$, $k = i + 1$ compute $x_{i+1}^* = \Phi_k(x_1^*, \ldots, x_i^*)$, to find the optimal solution.

As an example, let us consider the following problem:

$$\max_{\underline{x}} \quad 4x_1x_2x_3 - x_1x_2 - x_1x_3 - x_2x_3\,,$$
$$\underline{x} \in \{0,\, 1\}^3\,.$$

The notation presented below is adopted:

$$f_3(\underline{x}) = 4x_1x_2x_3 - x_1x_2 - x_1x_3 - x_2x_3\,,$$
$$k = 3\,.$$

Iteration 1

1) $(k = 3)$

$$g_3(x_1,\, x_2) = \frac{\partial f_3}{\partial x_3} = 4x_1x_2 - x_1 - x_2\,,$$
$$h_3(x_1,\, x_2) = f_3(x_1,\, x_2,\, 0) = -x_1x_2\,,$$
$$\Phi_3(x_1, x_2) = \begin{cases} 1 \text{ if } g_3(x_1,\, x_2) > 0 \\ 0 \quad \text{otherwise} \end{cases}\,.$$

Possible values for g_3:

x_1	x_2	g_3
0	0	0
0	1	-1
1	0	-2
1	1	1

thus $\Phi_3(x_1,\, x_2) = x_1x_2$.

2)

$$f_2(x_1,\, x_2) = \Phi_3(x_1,\, x_2)g_3(x_1,\, x_2) + h_3(x_1,\, x_2) = x_1x_2(4x_1x_2 - x_1 - x_2) - x_1x_2 =$$
$$= 4x_1^2x_2^2 - x_1^2x_2 - x_2^2x_1 = 4x_1x_2 - x_1x_2 - x_1x_2 - x_1x_2 = x_1x_2\,.$$

(since for binary-valued variables $x_i^2 = x_i$).

Iteration 2

1) $(k = 2)$

$$\begin{aligned} g_2(x_1) &= \frac{\partial f_2}{\partial x_2} = x_1\,, \\ h_2(x_1) &= f_2(x_1,\,0) = 0\,, \\ \Phi_2(x_1) &= \begin{cases} 1 \text{ if } g_2(x_1) > 0 \\ 0 \quad \text{otherwise} \end{cases} . \end{aligned}$$

Possible values for g_2:

x_1	g_2
0	0
1	1

,

thus $\Phi_2(x_1) = x_1$.

2)

$$f_1(x_1) = \Phi_2(x_1) g_2(x_1) + h_2(x_1) = x_1^2 = x_1\,,$$

3)

$$f_1(1) = 1 > f_1(0) = 0\,,$$

thus $\underline{x}^* = [1,\,1,\,1]^T$, $f(\underline{x}^*) = 1$.

The summary of all possible solutions is given below

x_1	x_2	x_3	$f(\underline{x})$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	-1
1	0	0	0
1	0	1	-1
1	1	0	-1
1	1	1	1

.

3. BRANCH AND BOUND METHOD

The problem to be solved by this method can be stated as follows

$$\begin{aligned} \max_{\underline{x}} \quad & f(\underline{x}) = \sum_{i=1}^{n} c_i x_i + \sum_{j \in \mathscr{N}} \left(a_j \prod_{i \in \mathscr{S}_j} x_i \right) \\ \text{s.t.} \quad & \sum_{i=1}^{n} d_{ji} x_i \leqslant \beta_j \quad (j = 1,\,\ldots,\,m)\,, \\ & \underline{x} \in \{0,\,1\}^n\,. \end{aligned}$$

At every stage, the problem is repeatedly divided into two subproblems, thus a tree of partial solutions is obtained. The maximum number of branches R is calculated using n and is given by

$$R = \sum_{r=0}^{n} 2^r \, .$$

At every branching point, the two subproblems have a single variable eliminated in comparison with the prior problem, to choose the better solution with smaller costs-to-go. The algorithm starts from the solution with all variables put to zero, apart from the currently processed variable. Introduced equality constraints are taken into account by observing consecutive combinations that form branches and are eliminated as infeasible on the basis of prior solutions.

At every branching point, the linear programming (LP) problem must be solved, i.e. the original problem with additional linear constraints added between iterations, what is a disadvantage of the presented algorithm. When the solution of the current LP problem is binary, the algorithm terminates.

Implementation of the penalty function is based on obtaining costs-to-go between the branches. The penalty functions values are defined by formulas given below, where c_k is the value of the aim function in the current branch, and c'_k in the other branch:

$$\begin{aligned} p_j^0 &= \sum_{\kappa} \max_{\mathscr{S}_\kappa \in \{j\}} (0,\, c_\kappa) \, , \\ p_j^1 &= \min_{\mathscr{S}_\kappa \in \{j\}} (0,\, c_\kappa) + \sum_{c_\kappa,\, c'_\kappa} \min_{\mathscr{S}_\kappa \in \{j\}} (|c_\kappa|,\, |c'_\kappa|) \, , \\ & \quad j \in \{1, \ldots, n\} \, . \end{aligned}$$

Eventually, these formulas reduce to obtaining the cost

$$v_i = \min_{\mathscr{S}_\kappa \in \{j\}} (0,\, c_\kappa) + \sum_{c_\kappa,\, c'_\kappa} \max_{\mathscr{S}_\kappa \in \{j\}} (|c_\kappa|,\, |c'_\kappa|) \, ,$$

where k is the iteration number $k \in \{1, \ldots, n\}$.

In this algorithm, combinations of binary variables which aim function has already been calculated for are blocked. If, for example, the constraints $x_3 = 1$ and $x_4 = 1$ are satisfied by the vector $[0,\, 0,\, 1,\, 1]^T$, all the aim functions with $x_3 = 1$ and $x_4 = 1$ are blocked.

The algorithm can be summarized as follows [4, 2]:

0) Initialization step

   Find the initial solution $\underline{x}$ of the problem (4) using, e.g. Frank-Wolfe or simplex method. If this solution is binary, terminate the algorithm. Otherwise, calculate the aim function for this solution. Put $\underline{x}_{\mathrm{opt}} = \underline{x}$, $f_{\mathrm{opt}} = f(\underline{x})$. Start Step 1 from the first decision variable.

1) If all the decision variables have been checked, stop the algorithm. Otherwise, verify whether the optimal binary solution of the obtained subproblem $\underline{x}^*$ with $f^* = f(\underline{x}^*)$ improves the aim function f_{opt}. If so, set $\underline{x}_{\mathrm{opt}} = \underline{x}^*$. Prior to obtaining solution to the LP problem check if the current combination of the decision variables is blocked.

Terminate the algorithm if the solution is binary and block the current combination so that this branch would not be considered anymore in further combinations. Check the remaining constraints.

2) Update the aim function for the current partitioning of variables by substituting a decision variable with either $x_i = 0$ or $x_i = 1$.
3) Update the constraints by adding a single equality constraint to the existing set of constraints for currently processed variables, either $x_i = 0$ or $x_i = 1$.
4) (only for the algorithm with the penalty function)

   For every variable with unfixed value in the subproblem calculate p_j^0 and p_j^1, $j \in \{1, \ldots, n\}$, and choose the constraint with the smallest penalty function value.
5) If all possible decision variables are considered the final solution is obtained. If no feasible solution has been found, the problem is infeasible. If both possible values for the i–th variable have been considered, put $i := i + 1$ and proceed do Step 1.

As an example, let us consider the following problem:

$$\begin{aligned} \max_{\underline{x}} \quad & 9x_1 + 5x_1x_2 + 6x_3 + 4x_4\,, \\ \text{s.t.} \quad & 6x_1 + 3x_2 + 5x_3 + 2x_4 \leqslant 10\,, \\ & x_3 + x_4 \leqslant 1\,, \\ & -x_1 + x_3 \leqslant 0\,, \\ & -x_2 + x_4 \leqslant 0\,, \\ & \underline{x} \in \{0,\, 1\}^4\,, \end{aligned}$$

with the initial solution passed to Franke-Wolfe algorithm as $\underline{x}^{(0)} = [0,\, 0,\, 0,\, 0]^T$.

0) Maximum number of possible branching points $R = 30$. Using Frank-Wolfe method the solution to LP problem is found

$$\begin{aligned} \underline{x}^* &= [0.8333,\, 1,\, 0,\, 1]^T\,, \\ f(\underline{x})^* &= 13.5\,, \end{aligned}$$

what forms the first branching point with $\underline{x}_{\text{opt}} = [0.8333,\, 1,\, 0,\, 1]^T$, $f_{\text{opt}} = 13.5$.

Iteration 1

1) The optimal solution is not binary.
2) Branching with respect to x_1: assuming that $x_1 = 0$.

   There is no block for the current equality constraint $x_1 = 0$.
3) The constraints are updated according to branching point (possible values $x_1 = 0$ or $x_1 = 1$); setting $x_1 = 0$ (equality constraint $x_1 = 0$ added to LP program solved with Franke-Wolfe algorithm).

Iteration 2

1) Solution to LP problem is $\underline{x}^* = [0,\, 0,\, 0,\, 0]^T$ and the aim function $f(\underline{x}^*) = 0$. This solution is an integer solution and x_1 is the branching variable, thus $x_1 = 0$ must be included to blocked combinations. Setting $\underline{x}_{\mathrm{opt}} = [0,\, 0,\, 0,\, 0]^T$, $f_{\mathrm{opt}} = 0$.
2) Branching with respect to x_1: assuming that $x_1 = 1$.

   There is no block for the current equality constraint $x_1 = 1$.
3) The constraints are updated according to branching point; setting $x_1 = 1$ (equality constraint $x_1 = 1$ added to LP program solved with Franke-Wolfe algorithm, replacing of the previously added constraint $x_1 = 0$).

Iteration 3

1) Solution to LP problem is $\underline{x}^* = [1,\, 0.8,\, 0,\, 0.8]^T$ and the aim function $f(\underline{x}^*) = -13.8$ (since 13.8 is not a natural number, this combination is not blocked).
2) Branching with respect to x_2: assuming that $x_2 = 0$.

   Equality constraints have the form $x_1 = 1$, $x_2 = 0$, and these variables are fixed (blocked), thus this branch is no longer considered. Due to the history of previous iterations, combinations $[x_1,\, x_2]$ of the form:

$$[0,\, 0]\,,$$
$$[0,\, 1]$$

   are blocked and no longer considered.
3) The following equality constraints are generated: $x_1 = 1$, $x_2 = 0$.

Iteration 4

1) Solution to LP problem is $\underline{x}^* = [1,\, 0,\, 0,\, 0]^T$ and aim functions $f(\underline{x}^*) = -9$. Since 9 is a natural number, the current solution is blocked ($x_1 = 1$).
2) Branching with respect to x_2: assuming that $x_2 = 1$.

   Equality constraints have the form $x_1 = 1$, $x_2 = 1$, and this combination is not blocked.
3) The following equality constraints are generated: $x_1 = 1$, $x_2 = 1$.

Iteration 5

1) Solution to LP problem is $\underline{x}^* = [1,\, 1,\, 0,\, 0]^T$ and aim functions $f(\underline{x}^*) = -14$. Since 14 is a natural number, the current solution is blocked ($x_1 = 1$, $x_2 = 1$).
2) Branching with respect to x_3: assuming that $x_3 = 0$.

   Equality constraints have the form $x_1 = 1$, $x_2 = 1$, $x_3 = 0$ and this combination is blocked, thus this branch is no longer considered.

Due to the history of previous iterations, combinations $[x_1,\, x_2,\, x_3]$ of the form:

$$[0,\, 0,\, 0]\,,$$
$$[0,\, 0,\, 1]\,,$$
$$[0,\, 1,\, 0]\,,$$
$$[0,\, 1,\, 1]\,,$$
$$[1,\, 0,\, 0]\,,$$
$$[1,\, 0,\, 1]\,,$$
$$[1,\, 1,\, 0]\,,$$
$$[1,\, 1,\, 1]$$

are blocked and no longer considered.

Iteration 6

2) Branching with respect to x_4: assuming that $x_4 = 0$.

   Equality constraints have the form $x_1 = 0$, $x_2 = 0$, $x_3 = 0$, and these variables are blocked, similarly to all the remaining combinations.

The optimal solution becomes $\underline{x}^* = [1,\, 1,\, 0,\, 0]^T$, $f(\underline{x}^*) = 14$.
As an example, let us consider the same problem again, this time using penalty function.

0) Maximum number of possible branching points $R = 30$. Using Frank-Wolfe method the solution to LP problem is found

$$\underline{x}^* \,=\, [0.8333,\, 1,\, 0,\, 1]^T\,,$$
$$f(\underline{x})^* \,=\, 13.5\,,$$

   what forms the first branching point with $\underline{x}_{\rm opt} = [0.8333,\, 1,\, 0,\, 1]^T$, $f_{\rm opt} = 13.5$.

Iteration 1

1) The optimal solution is not binary.
2) Branching with respect to x_1: assuming that $x_1 = 0$.

   There is no block for the current equality constraint $x_1 = 0$.
3) The constraints are updated according to branching point (possible values $x_1 = 0$ or $x_1 = 1$); setting $x_1 = 0$ (equality constraint $x_1 = 0$ added to LP program solved with Franke-Wolfe algorithm).

Iteration 2

1) Solution to LP problem is $\underline{x}^* = [0,\, 1,\, 0,\, 1]^T$ and aim functions $f(\underline{x}^*) = 6$. This solution is an integer solution and x_1 is the branching variable, thus $x_1 = 0$ must be included to blocked combinations. Setting $\underline{x}_{\rm opt} = [0,\, 1,\, 0,\, 1]^T$, $f_{\rm opt} = 6$.
2) Branching with respect to x_1: assuming that $x_1 = 1$.

   There is no block for the current equality constraint $x_1 = 1$.
3) The constraints are updated according to branching point; setting $x_1 = 1$ (equality constraint $x_1 = 1$ added to LP program solved with Franke-Wolfe algorithm, replacing of the previously added constraint $x_1 = 0$).

4) Cost-to-go for the current branch $13.5 - 6 = 7.5$.

Iteration 3

1) Solution to LP problem is $\underline{x}^* = [1,\, 0.8,\, 0,\, 0.8]^T$ and aim functions $f(\underline{x}^*) = -13.8$ (since 13.8 is not a natural number, this combination is not blocked).
2) Branching with respect to x_2: assuming that $x_2 = 0$.

   Equality constraints have the form $x_1 = 1$, $x_2 = 0$, and these variables are fixed (blocked), thus this branch is no longer considered. Due to the history of previous iterations, combinations $[x_1,\, x_2]$ of the form:

$$[0,\, 0]\,,$$
$$[0,\, 1]$$

   are blocked and no longer considered.
3) The following equality constraints are generated: $x_1 = 1$, $x_2 = 0$.
4) Cost-to-go for the current branch $13.5 - 13.8 = -0.3$. Minimum cost $\min(7.5,\, -0.3) = -0.3$.

Iteration 4

1) Solution to LP problem is $\underline{x}^* = [1,\, 0,\, 0,\, 0]^T$ and aim functions $f(\underline{x}^*) = -9$. Since 9 is a natural number, the current solution is blocked ($x_1 = 1$).
2) Branching with respect to x_2: assuming that $x_2 = 1$.

   Equality constraints have the form $x_1 = 1$, $x_2 = 1$, and this combination is not blocked.
3) The following equality constraints are generated: $x_1 = 1$, $x_2 = 1$.
4) Cost-to-go for the current branch $13.8 - 9 = 4.8$.

Iteration 5

1) Solution to LP problem is $\underline{x}^* = [1,\, 1,\, 0,\, 0]^T$ and aim functions $f(\underline{x}^*) = -14$. Since 14 is a natural number, the current solution is blocked ($x_1 = 1$, $x_2 = 1$).
2) Branching with respect to x_3: assuming that $x_3 = 0$.

   Equality constraints have the form $x_1 = 1$, $x_2 = 1$, $x_3 = 0$ and this combination is blocked, thus this branch is no longer considered.

   Due to the history of previous iterations, combinations $[x_1,\, x_2,\, x_3]$ of the form:

$$[0,\, 0,\, 0]\,,$$
$$[0,\, 0,\, 1]\,,$$
$$[0,\, 1,\, 0]\,,$$
$$[0,\, 1,\, 1]\,,$$
$$[1,\, 0,\, 0]\,,$$
$$[1,\, 0,\, 1]\,,$$
$$[1,\, 1,\, 0]\,,$$
$$[1,\, 1,\, 1]$$

   are blocked and no longer considered.

3) Cost-to-go for the current branch $13.8 - 14 = -0.2$. Minimum cost $\min(4.8,\ -0.2) = -0.2$.

Iteration 6

2) Branching with respect to x_4: assuming that $x_4 = 0$.

Equality constraints have the form $x_1 = 0$, $x_2 = 0$, $x_3 = 0$, and these variables are blocked, similarly to all the remaining combinations.

The optimal solution becomes $\underline{x}^* = [1,\ 1,\ 0,\ 0]^T$, $f(\underline{x}^*) = 14$.

4. CUTTING PLANES METHOD

Let us consider a binary problem with a linear aim function and nonlinear constraints of the form [4]

$$\begin{aligned} \max_{\underline{x}} \quad & f(\underline{x}) = \sum_{i=1}^{n} c_i x_i \\ \text{s.t.} \quad & \gamma_i = \sum_{\kappa=1}^{p_i} a_{i\kappa} \left(\prod_{j \in \mathscr{S}_{i\kappa}} x_j \right) \leqslant \beta_i \quad (i = 1,\ \ldots,\ m)\,, \\ & \underline{x} \in \{0,\ 1\}^n\ , \end{aligned}$$

where $\mathscr{S}_{i\kappa} \subset \{1,\ \ldots,\ n\}$. The general idea of the cutting planes method is to replace the nonlinear constraints with linear functions without introducing neither additional variables nor new constraints, leading to generation of a generalised covering relaxation (GCR) from nonlinear constraints. The resulting problem should be solvable using standard LP algorithms for binary programs.

Let us consider a constraint of the form

$$\sum_{j \in \mathscr{N}} \left(a_j \prod_{i \in \mathscr{S}_j} x_i \right) \leqslant \beta\,,$$

where $\mathscr{N}$ and $\mathscr{S}_j$ are non-empty sets of indexes and $\cup_{j \in \mathscr{N}} \mathscr{S}_j = \{1,\ \ldots,\ n\}$. Let us define:

$$\begin{aligned} \mathscr{N}^+ &= \{j \in \mathscr{N} : a_j \geqslant 0\}\ , \\ \mathscr{N}^- &= \{j \in \mathscr{N} : a_j < 0\}\ , \end{aligned}$$

and assume that $\sum_{j \in \mathscr{N}^+} a_j > b$, since $\gamma(\underline{x}) \leqslant \beta$ is satisfied for every combination.

Let us define

$$\begin{aligned} \gamma^+(\underline{x}) &= \sum_{j \in \mathscr{N}^+} \left(a_j \prod_{i \in \mathscr{S}_j} x_i \right)\ , \\ \gamma^-(\underline{x}) &= \sum_{j \in \mathscr{N}^-} \left(a_j \prod_{i \in \mathscr{S}_j} x_i \right)\ . \end{aligned}$$

The set $\mathscr{M} \subseteq \mathscr{N}$ is a cover of the inequality of

$$\sum_{j \in \mathscr{M}} |a_j| > b = \sum_{j \in \mathscr{N}^-} a_j \, .$$

A simple conclusion is that $\mathscr{N}$ is a relaxation of the given constraint function as long as it holds that $\sum_{j \in \mathscr{N}^+} a_j > 0$. A cover $\mathscr{M}$ is minimal if none of its subsets are covers.

The algorithm can be summarized as follows:

0) Put $k = 0$. Generate a set of relaxed constraints and generate initial GCR problem for a single constraint that is violated and label the problem as GCR$_k$.
1) Solve GCR$_k$ problem with a binary LP solver. Let $\underline{x}^{(k)}$ be the optimal solution of GCR$_k$. If this solution satisfies constraints of the original problem, terminate the algorithm and consider $\underline{x}^{(k)}$ to be the optimal solution to the binary polynomial programming problem.
2) For every function defining constraints of the form $\gamma_i(\underline{x}) \leqslant \beta_i$ that is violated for the decision variables $\underline{x}^{(k)}$ generate a generalised covering inequality and add it to GCR$_k$ problem, and, next, using LP solver find the optimal solution of the resulting linear programming task. If this solution is feasible in the original problem, terminate the algorithm, otherwise put $k := k + 1$ and go to Step 1.

Suppose that constraints are in the form

$$\gamma(\underline{x}) = \sum_{j \in \mathscr{N}} a_i \prod_{j \in \mathscr{S}_i} x_j \leqslant \beta \, ,$$

for nonempty $\mathscr{N}$ with nonempty sets $\mathscr{S}_i$ with $\cap_{j \in \mathscr{N}} S_j = \{1, \ldots, n\}$.

Now, let us define the following sets of indexes:

$$\begin{aligned}
\mathscr{G}^1(\underline{x}) &= \left\{ j \in \mathscr{N}^+ : \prod_{i \in \mathscr{S}_j} x_i = 1 \right\} , \\
\mathscr{G}^0(\underline{x}) &= \left\{ j \in \mathscr{N}^- : \prod_{i \in \mathscr{S}_j} x_i = 0 \right\} , \\
\mathscr{G}(\underline{x}) &= \mathscr{G}^0(\underline{x}) \cup \mathscr{G}^1(\underline{x}) \, .
\end{aligned}$$

and then from

$$\sum_{i \in \mathscr{G}(\underline{x})} |a_i| + \sum_{j \in \mathscr{N}^-} a_j > b$$

it turns out that $\mathscr{G}(\underline{x})$ is a cover for the presented constraint.

Now, let $\mathscr{M} \subseteq \mathscr{G}(\underline{x})$ be any cover for the presented constraint. For $i \in \mathscr{G}^0(\underline{x})$ let $\phi(i)$ be a mapping of indexes for which $x_{\phi(i)} = 0$, and

$$\begin{aligned}
\mathscr{G}_{\mathscr{M}} &= \cup_{i \in (\mathscr{M} \cap \mathscr{G}^1(\underline{x}))} \mathscr{S}_i \, , \\
\mathscr{G}_{\phi} &= \left\{ j = \phi(i) : i \in (\mathscr{M} \cap \mathscr{G}^0(\underline{x}) \right\} .
\end{aligned}$$

Then the generalized covering inequality becomes

$$\sum_{i\in\mathscr{G}_{\mathscr{M}}}(1-x_i)+\sum_{j\in\mathscr{G}_{\phi}}x_j\geqslant 1\,.$$

As an example, let us consider the following problem:

$$\begin{aligned}\max_{\underline{x}}\quad & -3x_1-5x_2-6x_3-x_6-x_5-5x_6\,,\\ \text{s.t.}\quad & 7x_1+2x_2-x_5x_6+4x_3+6x_6\leqslant 14\,,\\ & 4x_1x_3+8x_2x_3+3x_1x_2x_3-2x_3x_4-3x_5x_6+7x_6\leqslant 11\,,\\ & 3x_1x_2+4x_3-x_4x_5\leqslant 15\,,\\ & 5x_1x_3+4x_2x_6+2x_3-4x_1x_4-x_4x_5x_6\leqslant 14\,,\\ & \underline{x}\in\{0,\,1\}^6\,.\end{aligned}$$

Initial combination has the form $\underline{x}=[1,\,1,\,1,\,1,\,1,\,1]^T$.

Iteration 1

1) Generation of relaxed constraints

The constraint $7x_1+2x_2-x_5x_6+4x_3+6x_6\leqslant 14$ is violated by $\underline{x}=[1,\,1,\,1,\,1,\,1,\,1]^T$. The GCR_0 is generated:

$$\begin{aligned}\mathscr{N}&=\{1,\,2,\,3,\,4,\,5\}\,,\\ \mathscr{N}^+&=\{1,\,2,\,4,\,5\}\,,\\ \mathscr{N}^-&=\{3\}\,,\\ a_{\mathscr{N}}&=\{7,\,2,\,4,\,6\}\,,\\ a_{\mathscr{N}^+}&=\{7,\,2,\,4,\,6\}\,,\\ a_{\mathscr{N}^-}&=\emptyset\,.\end{aligned}$$

The current point violates the constraint.

Now, taking indexes from $\mathscr{N}^+$ the set $\mathscr{G}^1$ is updated:

$$\begin{aligned}&7x_1\to 7>0 \quad \text{index 1 added to } \mathscr{G}^1\,,\\ &2x_2\to 2>0 \quad \text{index 2 added to } \mathscr{G}^1\,,\\ &4x_3\to 4>0 \quad \text{index 4 added to } \mathscr{G}^1\,,\\ &6x_6\to 6>0 \quad \text{index 5 added to } \mathscr{G}^1\,,\end{aligned}$$

finally, $\mathscr{G}^1=\{1,\,2,\,4,\,5\}$.

Taking indexes from $\mathscr{N}^-$ the set $\mathscr{G}^0$ is updated:

$$-x_5x_6\to -1\neq 0 \quad \text{index 3 is not added to } \mathscr{G}^0\,,$$

finally, $\mathscr{G}^0=\emptyset$.

After calculations, $\mathscr{G}=\mathscr{G}^1\cup\mathscr{G}^0$, $\mathscr{G}_{\mathscr{M}}=\mathscr{G}^1$.

Now, the GCR_0 takes the form

$$(1-x_1)+(1-x_2)+(1-x_3)+(1-x_6)\geqslant 1\,.$$

The constraint $4x_1x_3+8x_2x_3+3x_1x_2x_3-2x_3x_4-3x_5x_6+7x_6\leqslant 11$ is also violated by $\underline{x}=[1,\,1,\,1,\,1,\,1,\,1]^T$. The GCR_1 is generated:

$$\begin{aligned}\mathscr{N} &= \{1,\,2,\,3,\,4,\,5,\,6\}\,,\\ \mathscr{N}^+ &= \{1,\,2,\,3,\,6\}\,,\\ \mathscr{N}^- &= \{4,\,5\}\,,\\ a_{\mathscr{N}} &= \{4,\,8,\,3,\,2,\,3,\,7\}\,,\\ a_{\mathscr{N}^+} &= \{4,\,8,\,3,\,7\}\,,\\ a_{\mathscr{N}^-} &= \{2,\,3\}\,.\end{aligned}$$

As in the latter case, the generated combination does not satisfy constraints, and $\mathscr{G}^1=\{1,\,2,\,3,\,6\}$, $\mathscr{G}^0=\emptyset$.

After calculations, $\mathscr{G}=\mathscr{G}^1\cup\mathscr{G}^0$, $\mathscr{G}_{\mathscr{M}}=\mathscr{G}^1$.

Now, the GCR_1 takes the form

$$(1-x_1)+(1-x_2)+(1-x_3)+(1-x_6)\geqslant 1\,.$$

The remaining two constraints are satisfied by the proposed point.

2) The problem with four constraints replaced by two new constraints is solved, and as the result $\underline{x}^*=[0,\,1,\,1,\,1,\,1,\,1]^T$, $f(\underline{x}^*)=6$, what satisfies original constraints

$$\begin{aligned}7\cdot 0+2\cdot 0-1\cdot 1+4\cdot 0+6\cdot 1=5\leqslant 14\,,\\ 4\cdot 0\cdot 0+8\cdot 0\cdot 0+3\cdot 0\cdot 0\cdot 0-2\cdot 0\cdot 0-3\cdot 1\cdot 1+7\cdot 1=4\leqslant 11\,,\\ 3\cdot 0\cdot 0+4\cdot 0-0\cdot 1=0\leqslant 15\,,\\ 5\cdot 0\cdot 0+4\cdot 0\cdot 1+2\cdot 0-4\cdot 0\cdot 0-0\cdot 1\cdot 1=0\leqslant 14\,.\end{aligned}$$

The optimal solution has the form:

$$\begin{aligned}\underline{x}^* &= [0,\,1,\,1,\,1,\,1,\,1]^T\,,\\ f(\underline{x}^*) &= 6\,,\end{aligned}$$

and is achieved in a single iteration.

5. Quadratic binary knapsack problem

In this case, the problem takes the form ($q_{ij}=q_{ji}$ for $i>j$) [4, 5]:

$$\begin{aligned}\max_{\underline{x}}\quad & f(\underline{x})=\sum_{i=1}^{n}\left(q_{ii}+\frac{1}{2}\sum_{j\neq i}q_{ij}x_j\right)x_j\\ \text{s.t.}\quad & \sum_{i=1}^{n}a_ix_i\leqslant b\,,\\ & \underline{x}\in\{0,\,1\}^n\,.\end{aligned}$$

A unique feature of the algorithm solving this problem is that greater attention is paid to a_i coefficients than to the price vector. The first combination of variables to be considered is this for which the constraint function has the maximum value, and in consecutive iterations other combinations are considered. As a result, decision variables are at maximum levels.

The algorithm can be summarized as follows:

1) Calculate

$$c_i = q_{ii} + \frac{1}{2}\sum_{j \neq i} q_{ij} \,,$$

and $p_i = \frac{c_i}{a_i}$ $(i = 1, \ldots, n)$. Set $\mathscr{K}_1 = \emptyset$ and $\mathscr{K}_0 = \{1, \ldots, n\}$ as the set of indexes of all decision variables, $\mathscr{I} = \mathscr{K}_0$ and $s = b$.

2) Compute $\kappa = \arg\max_{i \in \mathscr{I}} p_i$. If $\sum_{i \in \mathscr{K}_1 \cup \{\kappa\}} a_i > b$ set $\mathscr{I} := \mathscr{I} \setminus \{\kappa\}$. If $\mathscr{I} = \emptyset$, go to Step 4. If $\sum_{i \in \mathscr{K}_1 \cup \{\kappa\}} a_i \leqslant b$ set $\mathscr{K}_1 := \mathscr{K}_1 \cup \{\kappa\}$ and $\mathscr{K}_0 := \mathscr{K}_0 \setminus \{\kappa\}$, $s := s - a_\kappa$. Proceed to Step 3.

3) If $s < \min_{i \in \mathscr{K}_0} a_i$ go to Step 4. Otherwise, put

$$p_i := p_i - \frac{q_{\kappa i}}{2a_i}$$

for $i \in \mathscr{K}_0$. Set $\mathscr{I} = \mathscr{K}_0$ and return to Step 2.

4) For indexes $i \in \mathscr{K}_0$ set $\underline{x}_i = 0$, and for $j \in \mathscr{K}_1$ set $\underline{x}_j = 1$. Stop the algorithm.

As an example, let us consider the following problem:

$$\begin{aligned} \max_{\underline{x}} \quad & (x_1 + 4x_2 + x_3 + 2x_4 + 6x_5 + 5x_6 + 6x_1x_2 + 4x_1x_3 + 10x_1x_4 + \\ & + x_2x_3 + 5x_2x_4 + 4x_3x_4 + x_4x_5 + x_6x_3 + x_7x_4) \,, \\ \text{s.t.} \quad & 7x_1 + 5x_2 + 4x_3 + 2x_4 + 3x_5 + 4x_6 + 6x_7 \leqslant 16 \\ & \underline{x} \in \{0, 1\}^7 \,. \end{aligned}$$

1) In order to start the algorithm, the following must be calculated:

$$
\begin{aligned}
c_1 &= 1 + 0.5\,(6 + 4 + 10) = 11\,,\\
c_2 &= 4 + 0.5\,(6 + 1 + 5) = 10\,,\\
c_3 &= 1 + 0.5\,(4 + 1 + 4 + 1) = 6\,,\\
c_4 &= 2 + 0.5\,(10 + 5 + 4 + 1 + 1) = 12.5\,,\\
c_5 &= 6 + 0.5 = 6.5\,,\\
c_6 &= 5 + 0.5 = 5.5\,,\\
c_7 &= 0.5\,,\\
\underline{c} &= [11,\ 10,\ 6,\ 12.5,\ 6.5,\ 5.5,\ 0.5]^T\,,\\
p_1 &= \frac{11}{7}\,,\\
p_2 &= \frac{10}{5}\,,\\
p_3 &= \frac{6}{4}\,,\\
p_4 &= \frac{12.5}{2}\,,\\
p_5 &= \frac{6.5}{3}\,,\\
p_6 &= \frac{5.5}{4}\,,\\
p_7 &= \frac{0.5}{6}\,,\\
\underline{p} &= [1.5714,\ 2,\ 1.5,\ 6.25,\ 2.1667,\ 1.375,\ 0.0833]^T\,,\\
s &= 16\,,\\
\mathscr{K} &= \emptyset\,,\\
\mathscr{K} &= (1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7\}\ ,\\
\mathscr{I} &= \mathscr{K}_0\,.
\end{aligned}
$$

Iteration 1

2)

$$
\begin{aligned}
\kappa = \arg\max_{i \in \mathscr{I}} p_i &= 4\,,\\
\sum_{i \in \mathscr{K}_1 \cup \{4\}} a_i = a_4 = 2 \leqslant 16\,,&\\
\mathscr{K}_1 &= \{4\}\ ,\\
\mathscr{K}_0 &= \{1,\ 2,\ 3,\ 5,\ 6,\ 7\}\ ,\\
s &:= 16 - 2 = 14\,.
\end{aligned}
$$

3) $14 \not< \min_{i\in\mathscr{K}_0} a_i = 3$,

$$
\begin{aligned}
p_1 &:= p_1 - \frac{q_{41}}{2a_1} = 1.5714 - \frac{10}{14}\,,\\
p_2 &:= p_2 - \frac{q_{42}}{2a_2} = 2 - \frac{5}{10}\,,\\
p_3 &:= p_3 - \frac{q_{43}}{2a_3} = 1.5 - \frac{4}{8}\,,\\
p_5 &:= p_5 - \frac{q_{45}}{2a_5} = 2.1667 - \frac{1}{6}\,,\\
p_6 &:= p_6 - \frac{q_{46}}{2a_6} = 1.3750 - \frac{0}{8}\,,\\
p_7 &:= p_7 - \frac{q_{47}}{2a_7} = 0.0833 - \frac{1}{12}\,,\\
\underline{p} &= [0.85714,\, 1.5,\, 1,\, 2.1667,\, 1.37,\, 0.083333]^T\,,\\
\mathscr{I} &= \{1,\, 2,\, 3,\, 5,\, 6,\, 7\}\,.
\end{aligned}
$$

Iteration 2

2)

$$
\begin{aligned}
\kappa &= \arg\max_{i\in\mathscr{I}} p_i = 5\,,\\
\sum_{i\in\mathscr{K}_1\cup\{5\}} a_i &= a_4 + a_5 = 5 \leqslant 14\,,\\
\mathscr{K}_1 &= \{4,\, 5\}\,,\\
\mathscr{K}_0 &= \{1,\, 2,\, 3,\, 6,\, 7\}\,,\\
s &:= 14 - 3 = 11\,.
\end{aligned}
$$

3) $11 \not< \min_{i\in\mathscr{K}_0} a_i = 4$,

$$
\begin{aligned}
\underline{p} &= [0.8571,\, 1.5,\, 1,\, 1.375,\, 0.0833]^T\,,\\
\mathscr{I} &= \{1,\, 2,\, 3,\, 6,\, 7\}\,.
\end{aligned}
$$

Iteration 3

2)

$$
\begin{aligned}
\kappa &= \arg\max_{i\in\mathscr{I}} p_i = 2\,,\\
\sum_{i\in\mathscr{K}_1\cup\{2\}} a_i &= a_4 + a_5 + a_2 = 10 \leqslant 14\,,\\
\mathscr{K}_1 &= \{4,\, 5,\, 2\}\,,\\
\mathscr{K}_0 &= \{1,\, 3,\, 6,\, 7\}\,,\\
s &:= 11 - 5 = 6\,.
\end{aligned}
$$

3) $11 \not< \min_{i \in \mathscr{K}_0} a_i = 4$,

$$\begin{aligned} \underline{p} &= [0.4286,\, 1,\, 1.375,\, .0833]^T\,, \\ \mathscr{I} &= \{1,\, 3,\, 6,\, 7\}\,. \end{aligned}$$

Iteration 4

2)

$$\begin{aligned} \kappa &= \arg\max_{i \in \mathscr{I}} p_i = 6\,, \\ \sum_{i \in \mathscr{K}_1 \cup \{6\}} a_i &= a_4 + a_5 + a_2 + a_6 = 14 \leqslant 14\,, \\ \mathscr{K}_1 &= \{4,\, 5,\, 2,\, 6\}\,, \\ \mathscr{K}_0 &= \{1,\, 3,\, 7\}\,, \\ s &:= 6 - 4 = 2\,. \end{aligned}$$

3) $11 \not< \min_{i \in \mathscr{K}_0} a_i = 4$,

$$\begin{aligned} \underline{p} &= [0.4286,\, 1,\, 0.0833]^T\,, \\ \mathscr{I} &= \{1,\, 3,\, 7\}\,. \end{aligned}$$

Iteration 5

2)

$$\begin{aligned} \kappa &= \arg\max_{i \in \mathscr{I}} p_i = 7\,, \\ \sum_{i \in \mathscr{K}_1 \cup \{7\}} a_i &= a_4 + a_5 + a_2 + a_6 + a_7 = 20 > 14\,, \\ \mathscr{I} &= \{1,\, 3\}\,. \end{aligned}$$

Iteration 6

2)

$$\begin{aligned} \kappa &= \arg\max_{i \in \mathscr{I}} p_i = 1\,, \\ \sum_{i \in \mathscr{K}_1 \cup \{1\}} a_i &= a_4 + a_5 + a_2 + a_6 + a_7 + a_1 = 27 > 14\,, \\ \mathscr{I} &= \{3\}\,. \end{aligned}$$

Iteration 7

2)

$$\begin{aligned} \kappa &= \arg\max_{i\in\mathscr{I}} p_i = 3\,, \\ \sum_{i\in\mathscr{K}_1\cup\{3\}} a_i &= a_4 + a_5 + a_2 + a_6 + a_7 + a_1 + a_3 = 31 > 14\,, \\ \mathscr{I} &= \emptyset\,. \end{aligned}$$

Optimal solution found in 6 iterations:

$$\begin{aligned} \underline{x}^* &= [0,\,1,\,0,\,1,\,1,\,1,\,0]^T\,, \\ f(\underline{x}^*) &= 23\,. \end{aligned}$$

6. BRANCH AND BOUND METHOD BASED ON LAGRANGE RELAXATION

For this method, the main problem has the form as in the case of a Quadratic Knapsack Problem (QKP).

The main QKP problem is rewritten to the form [4]

$$L(\underline{x},\,\lambda) = f(\underline{x}) - \lambda(g(\underline{x}) - b)\,,$$

where $\lambda \geqslant 0$ is the Lagrange multiplier that is calculated in every iteration.

The dual Lagrange function is of the form

$$d(\lambda) = \max_{\lambda} \left(L(\underline{x},\,\lambda) : \underline{x} \in \{0,\,1\}^n\right)\,.$$

The Lagrange multiplier is defined by the formula

$$\lambda_k = \max_{\underline{x}} \left(\frac{f(\underline{x}) - f(\underline{x}^{(l-1)})}{g(\underline{x}) - g(\underline{x}^{(l-1)})} \right)\,,$$

where l is the index denoting a previous value of a vector of decision variables.

At every branching point it is necessary to obtain value of the Lagrange multiplier on the basis of the maximal value of the aim function and constraint function and current combination of decision variables. On this basis, a vector is generated, from which the multiplier is sought according to the maximum formula.

Branching process is based on the weight of the decision variables in the problem, from the constraint function. The only feature that changes during the algorithm is a method to select variables to branch. To do this Lagrange relaxation is used. Value s denotes the current constraint values.

Algorithm can be described as follows.

Main Step I

Create the set of indexes of decision variables $\mathscr{I} = \{1,\,\ldots,\,n\}$, and find a feasible solution using QKP algorithm. Set this solution as the current optimal solution. If it does not exist, terminate the algorithm.

Main Step II (selection of variables)

1) From the initial function obtain Lagrange multiplier h, and formulate the dual function

$$d(h) = \max_{\underline{x}} \left(Q(\underline{x}) - h(\underline{a}^T \underline{x} - b) \right) ,$$

where Q denotes the aim function, $\underline{a}$ is the constraint vector, and b is the right-hand side of the constraint.
2) set $\mathscr{J} = \emptyset$ and $j = 1$,
3) Include $\{j\}$ to $\mathscr{J}$ if $1 - x_j = 1$, otherwise include $\{-j\}$. Include $\{-\underline{\kappa}\}$ to $\mathscr{J}$ for every $\kappa \in \mathscr{I} / \mathscr{J}$ such that $a_\kappa > b - \sum_{i \in \mathscr{J}} a_i$ is satisfied. Next, calculate the solution to the subproblem with $x_i = 0$ whenever $-i \in \mathscr{J}$ or $x_j = 1$ whenever $j \in \mathscr{J}$. If the Lagrangean bound of the subproblem satisfies $d_j \leqslant f_{\text{opt}}$ (current optimal solution), change $\{j\}$ in set $\mathscr{J}$ to $\{-\underline{j}\}$ or change $\{-j\}$ in set $\mathscr{J}$ to $\{\underline{j}\}$ and remove all underlined indexes to its right from $\mathscr{J}$.
4) Unless $j < n$ set $j := j + 1$ and go to Step 3. Otherwise, proceed to Main Step 3.

Main Step III

1) Update the constraint for prior branching points

$$s = b - \sum_{j \in \mathscr{J}} a_j ,$$

and if $s < 0$, to go step 6.
2) Update $\mathscr{J}$ – for every $j \in \mathscr{I} / \mathscr{J}$ add $\{-\underline{j}\}$ to $\mathscr{J}$ if $a_j > s$.
3) Calculate Lagrange multiplier and create Lagrange dual function for its variables ($x_i = 0$ if $-i \in \mathscr{J}$ or $x_i = 1$ if $i \in \mathscr{J}$). If the solution of the Lagrange dual problem $D \leqslant f_{\text{opt}}$, proceed to Step 6.
4) Solve Lagrange dual problem by substituting the current combination, if $\underline{x}^*$ is feasible and improves the current optimal value of the function – set this combination as current optimal. If constraint becomes active proceed to Step 6.
5) For every index $j \in \mathscr{I} \setminus \mathscr{J}$ find the smallest cost-to-go $p_i = L(\underline{x}^*, \lambda^*) - L(\underline{y}^j, \lambda^*)$ with $L(\underline{x}, \lambda) = Q(\underline{x}) - \lambda(\underline{a}^T \underline{x} - b)$ and $y_i^j = x_i^*$ for $i \neq j$, $y_j^j = 1 - x_j^*$. Select j as the index of the smallest cost and include this index to $\mathscr{J}$ if $x_j^* = 0$ or negative index if $x_j^* = 1$. Go to Step 1.
6) Find moving from right to left the first j or $-j$ index that is not underlined. If there is no index to branch, the current solution is optimal. Otherwise, move all indexes to the right of the found index out from $\mathscr{J}$ and change $\{j\}$ in set $\mathscr{J}$ to $\{-\underline{j}\}$ or change $\{-j\}$ in set $\mathscr{J}$ to $\{\underline{j}\}$. Proceed to Step 1.

As an example, let us consider again the previous example, after [4], with $\underline{x}^{(0)} = [1,\ 0,\ 1,\ 1]^T$, $f(\underline{x}^{(0)}) = 22$ ($\underline{x}_{\text{opt}} = [1,\ 0,\ 1,\ 1]^T$, $f_{\text{opt}} = 22$.

Step II

1) Lagrange multiplier $h = 2.1111$, and

$$d(h) = 2.1111 \cdot 13 + x_1 + 4x_2 + x_3 + 2x_4 + 6x_1x_2 + 4x_1x_3 + 10x_1x_4 + x_2x_3 + 5x_2x_4 + \\ +4x_3x_4 - 2.1111(7x_1 + 5x_2 + 4x_3 + 2x_4) = 27.4444$$

with $\underline{x}^* = [0,\ 0,\ 0,\ 0]^T$.

2) Let $\mathscr{J} = \emptyset$. It is impossible to create constraint in dual problem for $d_1 = 25$ since f_{opt} is exceeded.

It is impossible to create constraint in dual problem for $d_2 = 25.7273$ since f_{opt} is exceeded.

It is impossible to create constraint in dual problem for $d_3 = 25.0833$ since f_{opt} is exceeded.

It is impossible to create constraint in dual problem for $d_4 = 26.75$ since f_{opt} is exceeded.

No variable can be fixed

Step III

1) Setting $s = 13$.

3) By solving the dual problem with $\mathscr{J} = \emptyset$ we get

$$\begin{aligned} h &= 2.1111\,, \\ d(h) &= 27.4444 > f_{\text{opt}}\,, \\ \underline{x}^* &= [0,\ 0,\ 0,\ 0]^T\,. \end{aligned}$$

5) Costs-to-go become:

$$\begin{aligned} \underline{p} &= [27.4444 - 13.6667,\ 27.4444 - 28.8889,\ 27.4444 - 20,\ 27.4444 - 25.2222]^T = \\ &= [13.7778,\ 6.5555,\ 7.4444,\ 2.2222]^T\,, \\ \min_j p_j &= 2.2222\,, \\ j &= 4\,, \\ &\quad \{4\} \text{ included to } \mathscr{J} \end{aligned}$$

1)

$$s = 13 - 2 = 11$$

3) By solving the dual problem with $\mathscr{J} = \{4\}$ we get

$$\begin{aligned} h &= 2.25\,, \\ d(h) &= 26.75 > f_{\text{opt}}\,, \\ \underline{x}^* &= [0,\ 0,\ 0,\ 1]^T\,. \end{aligned}$$

5) Costs-to-go become:

$$\begin{aligned} \underline{p} &= [4.75,\, 2.25,\, 4]^T\,, \\ \min_j p_j &= 2.25\,, \\ j &= 2\,, \\ & \{2\} \text{ included to } \mathscr{J} \end{aligned}$$

1)

$$s = 11 - 5 = 6$$

2) Since $a_1 = 7 > 6 = s$, $\mathscr{J} = \{4,\, 2,\, \underline{-1}\}$.

4) By solving the dual problem with $\mathscr{J} = \{4,\, 2,\, \underline{-1}\}$ we get

$$\begin{aligned} h &= 0\,, \\ d(h) &= 17 \leqslant f_{\text{opt}}\,, \\ \underline{x}^* &= [0,\, 1,\, 1,\, 1]^T\,. \end{aligned}$$

6) Backtracking, $\mathscr{J} = \{4,\, \underline{-2}\}$

2)

$$s = 13 - 2 = 11$$

4) By solving the dual problem with $\mathscr{J} = \{4,\, \underline{-2}\}$ we get

$$\begin{aligned} h &= 0\,, \\ d(h) &= 22 = f_{\text{opt}}\,, \\ \underline{x}^* &= [1,\, 0,\, 1,\, 1]^T\,. \end{aligned}$$

6) Backtracking, $\mathscr{J} = \{\underline{-4}\}$

2)

$$s = 13$$

4) By solving the dual problem with $\mathscr{J} = \{\underline{-4}\}$ we get

$$\begin{aligned} h &= 1.0625\,, \\ d(h) &= 13.9125 < f_{\text{opt}}\,, \\ \underline{x}^* &= [0,\, 0,\, 0,\, 0]^T\,. \end{aligned}$$

7) There are no non-underlined indexes in $\mathscr{J}$, stopping the algorithm.

The response to the problem becomes:

$$\begin{aligned} \underline{x}^* &= [1,\, 0,\, 1,\, 1]^T\,, \\ f(\underline{x}^*) &= 22\,. \end{aligned}$$

7. COMPARISON OF PERFORMANCE OF SELECTED ALGORITHMS

In order to test the presented algorithms general test must be conducted, enabling one to compare them in a unique fashion. The basic algorithm without constraints is the simplest, with a fixed number of iterations, related directly to the number of variables. The tests have been carried for $n = 1, \ldots, 20$ and $m = 1, \ldots, 20$, assuming that in the worst case the aim function consists of products of 8 variables of arbitrary index for a randomly generated problem with given dimensions. Coefficients of aim functions have taken on values from the span $[0,\ 10]$, left-hand side coefficients of linear constraints from $[0,\ 10]$, and right-hand side from $[0,\ 20]$.

Tab. 1. Performance evaluation of the branch and bound method with penalty function

$m \setminus n$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1.00	1.40	3.10	3.40	4.45	5.34	6.23	7.12	8.01	8.90	9.79	10.68	11.57	12.46	13.35	14.24	15.13	16.02	16.91	17.80
2	1.00	1.70	2.70	2.90	3.75	4.42	5.09	5.76	6.43	8.00	8.13	8.85	9.57	10.29	11.01	11.73	12.44	13.16	13.88	14.60
3	1.00	1.40	2.30	3.60	4.25	5.12	5.99	6.86	7.73	8.60	9.47	10.34	11.21	12.08	12.95	13.82	14.69	15.56	16.43	17.30
4	1.00	1.50	2.80	4.10	5.00	6.06	7.12	8.18	9.24	10.30	11.36	12.42	13.48	14.54	15.60	16.66	17.72	18.78	19.84	20.90
5	1.00	1.40	2.90	4.00	4.95	6.00	7.05	8.10	9.15	10.20	11.25	12.30	13.35	14.40	15.45	16.50	17.55	18.60	19.65	20.70
6	1.00	1.40	3.00	4.10	5.10	6.19	7.28	8.37	9.46	10.55	11.64	12.73	13.82	14.91	16.00	17.09	18.18	19.27	20.36	21.45
7	1.00	1.50	3.20	4.00	5.10	6.17	7.24	8.31	9.38	10.45	11.52	12.59	13.66	14.73	15.80	16.87	17.94	19.01	20.08	21.15
8	1.00	1.40	3.30	4.50	5.65	6.89	8.13	9.37	10.61	11.85	13.09	14.33	15.57	16.81	18.05	19.29	20.53	21.77	23.01	24.25
9	1.00	1.50	3.40	3.60	4.80	5.77	6.74	7.71	8.68	9.65	10.62	11.59	12.56	13.53	14.50	15.47	16.44	17.41	18.38	19.35
10	1.00	1.60	3.00	3.70	4.70	5.65	6.60	7.55	8.50	9.45	10.40	11.35	12.30	13.25	14.20	15.15	16.10	17.05	18.00	18.95
11	1.00	1.49	3.31	4.19	5.34	6.48	7.62	8.76	9.90	10.91	12.12	13.25	14.39	15.52	16.65	17.78	18.91	20.04	21.17	22.31
12	1.00	1.50	3.38	4.26	5.45	6.61	7.78	8.94	10.11	11.12	12.38	13.53	14.69	15.85	17.01	18.16	19.32	20.48	21.63	22.79
13	1.00	1.50	3.44	4.33	5.55	6.74	7.94	9.13	10.32	11.32	12.63	13.81	15.00	16.18	17.36	18.54	19.73	20.91	22.09	23.27
14	1.00	1.50	3.50	4.40	5.65	6.87	8.10	9.32	10.54	11.52	12.88	14.09	15.30	16.51	17.72	18.93	20.13	21.34	22.55	23.76
15	1.00	1.50	3.56	4.48	5.76	7.01	8.25	9.50	10.75	11.73	13.14	14.37	15.61	16.84	18.07	19.31	20.54	21.77	23.01	24.24
16	1.00	1.51	3.63	4.55	5.86	7.14	8.41	9.69	10.96	11.93	13.39	14.65	15.91	17.17	18.43	19.69	20.95	22.21	23.47	24.73
17	1.00	1.51	3.69	4.62	5.96	7.27	8.57	9.87	11.18	12.13	13.65	14.93	16.22	17.50	18.79	20.07	21.36	22.64	23.92	25.21
18	1.00	1.51	3.75	4.69	6.07	7.40	8.73	10.06	11.39	12.34	13.90	15.21	16.52	17.83	19.14	20.45	21.76	23.07	24.38	25.69
19	1.00	1.51	3.81	4.76	6.17	7.53	8.89	10.25	11.61	12.54	14.15	15.49	16.83	18.16	19.50	20.83	22.17	23.51	24.84	26.18
20	1.00	1.52	3.88	4.84	6.27	7.66	9.05	10.43	11.82	12.74	14.41	15.77	17.13	18.49	19.85	21.22	22.58	23.94	25.30	26.66

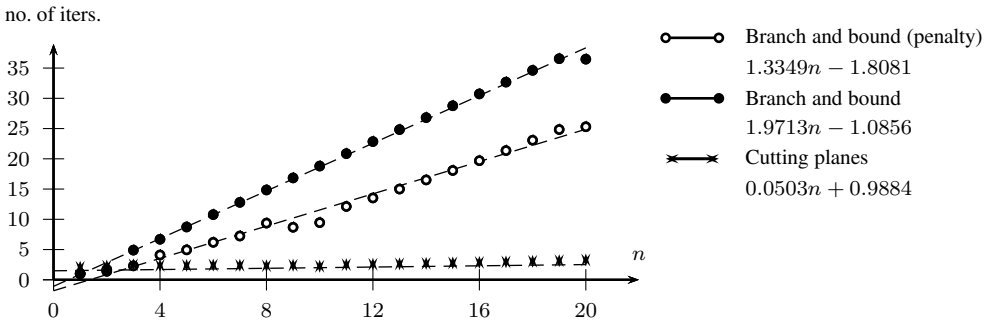

Fig. 1. Performance evaluation of selected methods with $\frac{m}{n} = 1$

Tab. 2. Performance evaluation of the branch and bound method without penalty function

$m \setminus n$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1.00	1.40	5.40	6.70	8.90	11.01	13.12	15.23	17.34	19.45	21.56	23.67	25.78	27.89	30.00	32.11	34.22	36.33	38.44	40.55
2	1.00	1.70	5.00	6.70	8.70	10.74	12.78	14.82	16.86	18.90	20.94	22.98	25.02	27.06	29.10	31.14	33.18	35.22	37.26	39.30
3	1.00	1.40	4.90	6.80	8.75	10.84	12.93	15.02	17.11	19.20	21.29	23.38	25.47	27.56	29.65	31.74	33.83	35.92	38.01	40.10
4	1.00	1.50	5.20	6.70	8.80	10.88	12.96	15.04	17.12	19.20	21.28	23.36	25.44	27.52	29.60	31.68	33.76	35.84	37.92	40.00
5	1.00	1.40	4.95	6.75	8.73	10.81	12.89	14.97	17.05	19.13	21.21	23.29	25.37	27.45	29.53	31.61	33.69	35.77	37.85	39.93
6	1.00	1.40	4.88	6.76	8.70	10.78	12.85	14.93	17.00	19.08	21.16	23.23	25.31	27.38	29.46	31.54	33.61	35.69	37.76	39.84
7	1.00	1.50	4.81	6.77	8.68	10.74	12.80	14.86	16.92	18.99	21.05	23.11	25.17	27.23	29.30	31.36	33.42	35.48	37.54	39.61
8	1.00	1.40	4.74	6.78	8.65	10.72	12.79	14.85	16.92	18.99	21.06	23.13	25.19	27.26	29.33	31.40	33.47	35.53	37.60	39.67
9	1.00	1.50	4.67	6.79	8.63	10.68	12.73	14.79	16.84	18.90	20.95	23.00	25.06	27.11	29.17	31.22	33.27	35.33	37.38	39.44
10	1.00	1.60	4.60	6.80	8.60	10.64	12.68	14.72	16.76	18.80	20.84	22.88	24.92	26.96	29.00	31.04	33.08	35.12	37.16	39.20
11	1.00	1.49	4.53	6.81	8.58	10.62	12.67	14.72	16.76	18.81	20.86	22.90	24.95	27.00	29.04	31.09	33.14	35.18	37.23	39.28
12	1.00	1.50	4.46	6.82	8.55	10.59	12.63	14.68	16.72	18.76	20.80	22.85	24.89	26.93	28.97	31.02	33.06	35.10	37.14	39.19
13	1.00	1.50	4.39	6.83	8.53	10.56	12.60	14.64	16.68	18.72	20.75	22.79	24.83	26.87	28.91	30.95	32.98	35.02	37.06	39.10
14	1.00	1.50	4.32	6.84	8.50	10.53	12.57	14.60	16.64	18.67	20.70	22.74	24.77	26.81	28.84	30.87	32.91	34.94	36.98	39.01
15	1.00	1.50	4.25	6.85	8.48	10.50	12.53	14.56	16.59	18.62	20.65	22.68	24.71	26.74	28.77	30.80	32.83	34.86	36.89	38.92
16	1.00	1.51	4.18	6.86	8.45	10.48	12.50	14.53	16.55	18.58	20.60	22.63	24.65	26.68	28.70	30.73	32.76	34.78	36.81	38.83
17	1.00	1.51	4.11	6.87	8.43	10.45	12.47	14.49	16.51	18.53	20.55	22.57	24.59	26.62	28.64	30.66	32.68	34.70	36.72	38.74
18	1.00	1.51	4.04	6.88	8.40	10.42	12.43	14.45	16.47	18.48	20.50	22.52	24.54	26.55	28.57	30.59	32.60	34.62	36.64	38.65
19	1.00	1.51	3.97	6.89	8.38	10.39	12.40	14.41	16.43	18.44	20.45	22.46	24.48	26.49	28.50	30.52	32.53	34.54	36.55	38.57
20	1.00	1.52	3.90	6.90	8.35	10.36	12.37	14.38	16.38	18.39	20.40	22.41	24.42	26.43	28.43	30.44	32.45	34.46	36.47	38.48

Tab. 3. Performance evaluation of the cutting-planes method

$m \setminus n$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1.00	1.10	1.20	1.10	1.10	1.10	1.09	1.08	1.07	1.06	1.08	1.08	1.08	1.07	1.07	1.07	1.07	1.07	1.07	1.06
2	1.00	1.20	1.30	1.20	1.20	1.20	1.19	1.18	1.17	1.16	1.20	1.20	1.21	1.21	1.21	1.22	1.22	1.23	1.23	1.23
3	1.00	1.10	1.40	1.10	1.10	1.10	1.07	1.04	1.01	0.98	1.00	0.98	0.97	0.95	0.93	0.92	0.90	0.89	0.87	0.85
4	1.00	1.30	1.50	1.30	1.30	1.30	1.28	1.26	1.24	1.22	1.28	1.28	1.28	1.29	1.29	1.29	1.29	1.29	1.29	1.30
5	1.00	1.30	1.30	1.30	1.30	1.30	1.30	1.30	1.30	1.30	1.36	1.38	1.39	1.41	1.43	1.44	1.46	1.47	1.49	1.51
6	1.00	1.35	1.20	1.35	1.35	1.35	1.37	1.38	1.40	1.41	1.48	1.51	1.54	1.57	1.60	1.63	1.66	1.69	1.72	1.75
7	1.00	1.30	1.10	1.30	1.30	1.30	1.32	1.34	1.36	1.38	1.44	1.47	1.50	1.53	1.56	1.59	1.63	1.66	1.69	1.72
8	1.00	1.20	1.00	1.20	1.20	1.20	1.22	1.24	1.26	1.28	1.32	1.35	1.37	1.40	1.42	1.45	1.47	1.50	1.52	1.55
9	1.00	1.30	1.20	1.30	1.30	1.30	1.31	1.32	1.33	1.34	1.40	1.42	1.45	1.47	1.49	1.52	1.54	1.57	1.59	1.61
10	1.00	1.20	1.30	1.20	1.20	1.20	1.19	1.18	1.17	1.16	1.20	1.20	1.21	1.21	1.21	1.22	1.22	1.23	1.23	1.23
11	1.00	1.31	1.15	1.31	1.31	1.31	1.32	1.34	1.35	1.37	1.43	1.46	1.49	1.52	1.55	1.57	1.60	1.63	1.66	1.69
12	1.00	1.32	1.13	1.32	1.32	1.32	1.34	1.36	1.38	1.40	1.46	1.49	1.52	1.55	1.59	1.62	1.65	1.68	1.71	1.74
13	1.00	1.33	1.11	1.33	1.33	1.33	1.36	1.38	1.40	1.42	1.49	1.52	1.56	1.59	1.63	1.66	1.70	1.73	1.76	1.80
14	1.00	1.35	1.09	1.35	1.35	1.35	1.37	1.40	1.42	1.45	1.52	1.55	1.59	1.63	1.67	1.70	1.74	1.78	1.82	1.85
15	1.00	1.36	1.07	1.36	1.36	1.36	1.39	1.42	1.44	1.47	1.55	1.59	1.63	1.67	1.71	1.75	1.79	1.83	1.87	1.91
16	1.00	1.37	1.05	1.37	1.37	1.37	1.40	1.44	1.47	1.50	1.57	1.62	1.66	1.70	1.75	1.79	1.83	1.88	1.92	1.97
17	1.00	1.38	1.03	1.38	1.38	1.38	1.42	1.46	1.49	1.53	1.60	1.65	1.70	1.74	1.79	1.83	1.88	1.93	1.97	2.02
18	1.00	1.40	1.02	1.40	1.40	1.40	1.44	1.47	1.51	1.55	1.63	1.68	1.73	1.78	1.83	1.88	1.93	1.98	2.03	2.08
19	1.00	1.41	1.00	1.41	1.41	1.41	1.45	1.49	1.54	1.58	1.66	1.71	1.76	1.82	1.87	1.92	1.97	2.03	2.08	2.13
20	1.00	1.42	0.98	1.42	1.42	1.42	1.47	1.51	1.56	1.60	1.69	1.74	1.80	1.85	1.91	1.97	2.02	2.08	2.13	2.19

Tab. 4. Comparison of performance of selected methods with: a) $\frac{m}{n} = 1$, b) $\frac{m}{n} = 2$, c) $\frac{m}{n} = 0.5$

a)

n	m	BBP	BB	CP
1	1	1.00	1.00	1.00
2	2	1.40	1.70	1.20
3	3	2.30	4.90	1.40
4	4	4.10	6.70	1.30
5	5	4.95	8.73	1.30
6	6	6.19	10.78	1.35
7	7	7.24	12.80	1.32
8	8	9.37	14.85	1.24
9	9	8.68	16.84	1.33
10	10	9.45	18.80	1.16
11	11	12.12	20.86	1.43
12	12	13.53	22.85	1.49
13	13	15.00	24.83	1.56
14	14	16.51	26.81	1.63
15	15	18.07	28.77	1.71
16	16	19.69	30.73	1.79
17	17	21.36	32.68	1.88
18	18	23.07	34.62	1.98
19	19	24.84	36.55	2.08
20	20	25.30	36.47	2.19

b)

n	m	$n+m$	BBP	BB	CP
1	2	3	1.00	1.00	1.00
2	4	6	1.50	1.50	1.30
3	6	9	3.00	4.88	1.35
4	8	12	4.50	6.78	1.20
5	10	15	4.70	8.60	1.23
6	12	18	6.61	10.59	1.32
7	14	21	8.10	12.57	1.37
8	16	24	9.69	14.53	1.44
9	18	27	11.39	16.47	1.51
10	20	30	12.74	18.39	1.60

c)

n	m	$n+m$	BBP	BB	CP
2	1	3	1.40	1.40	1.10
4	2	6	2.90	6.70	1.20
6	3	9	5.12	10.84	1.10
8	4	12	8.18	15.04	1.26
10	5	15	10.20	19.13	1.30
12	6	18	12.73	23.23	1.51
14	7	21	14.73	27.23	1.53
16	8	24	19.29	31.40	1.45
18	9	27	17.41	35.33	1.57
20	10	30	22.31	39.20	1.51

Tab. 5. Comparison of performance of selected methods for the problem with a single constraint

n	QKP	LR
1	1.0	1.0
2	1.8	2.8
3	2.9	4.5
4	4.2	6.1
5	4.9	7.0
6	5.8	8.0
7	6.9	11.0
8	8.2	11.2
9	9.1	13.0
10	11.1	13.6
20	19.6	29.0
30	26.0	43.0
40	40.1	59.0
50	50.5	57.0

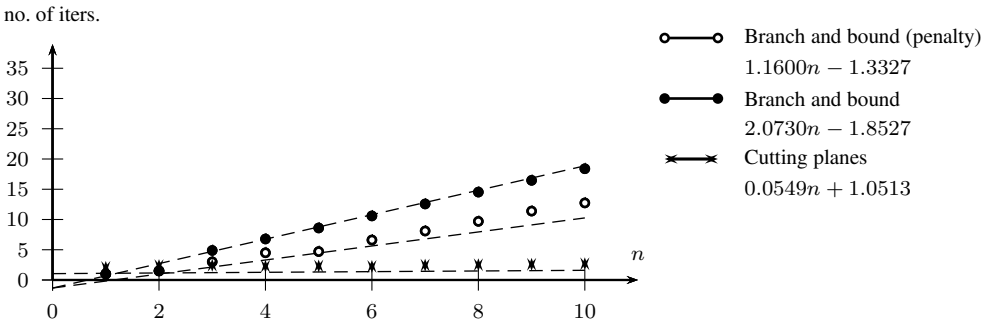

Fig. 2. Performance evaluation of selected methods with $\frac{m}{n} = 2$

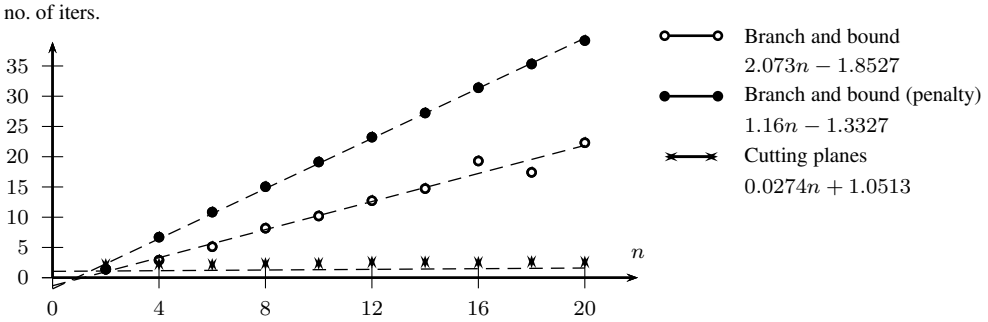

Fig. 3. Performance evaluation of selected methods with $\frac{m}{n} = 0.5$

Tables 1–5 show mean numbers of iterations vs. m and n for 10 problems solved for the given dimension of the program. The graphical interplay between problem dimensions has been presented in Figures 1–4.

As can be seen, introduction of the penalty function to branch and bound algorithm reduces mean burden expressed by the number of iterations, by approximately 25%. The cutting planes method requires less iterations to solve given problems, nevertheless, it is to be borne in mind that the time consumed to terminate is greater (LP problems solved along the way).

Figures suggest that the presented algorithms have linear computational complexity, with specific L_1 regression line equations presented in respectful Figures.

The cutting planes method is most appealing when solving problems, independently of the ratio $m : n$, the branch and bound method with penalty function has the intermediate performance, and the worst method turns out to be the basic branch and bound algorithm.

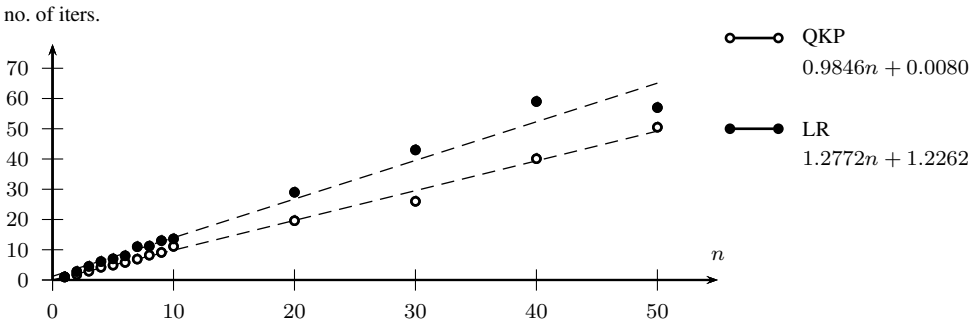

Fig. 4. Performance evaluation of selected methods with a single constraint

8. SUMMARY

The paper aimed at comparing performance of different 0-1 polynomial programming solvers taking number of constraints and number of decision variables into account. In the case of every algorithm, a sample problem has been explicitly solved, with a general outline of the algorithm presented, to enable the reader to fully understand the course of solution.

As a result of the presented comparison, the most-appealing method has been chosen, namely cutting planes method for problems with multiple constraints, and, in the case of single-constraint programs, the method dedicated for QKP problem, applied even for problems with large number of decision variables, reaching $n = 50$.

It is to be stressed that any more general comparison is impossible, since virtually all algorithms solve problems with different structures.

REFERENCES

[1] J. Arora. *Introduction to Optimum Design.* Elsevier Academic Press, 2nd edition, 2004.

[2] D. Bertsimas, D. Iancu, and D. Katz. A general purpose local search algorithm for binary optimization (manuscript). 2008.

[3] D. Granot and D. Granot. Generalized covering relaxation for 0-1 programs. *Operations Research*, 28(6):1442–1450, 1980.

[4] D. Li and S. Xiaoling. *Nonlinear Integer Programming*. USA, Springer, 2006.

[5] D. Pisinger, A. Rasmussen, and R. Sandvik. Solution of large-sized quadratic knapsack problems through aggressive reduction. Technical Report 2004/11, University of Copenhagen, 2004.

[6] M. Syslo, N. Deo, and J. Kowalik. *Algorytmy optymalizacji dyskretnej z programami w jezyku Pascal*. Wydawnictwo Naukowe PWN, Warszawa, 1995.

[7] P. Venkataraman. *Applied Optimization with Matlab Programming*. Wiley, 2009.

ABSTRACT

The paper considers performance issues of a class of iterative minimization methods of binary programs with polynomial functions. Problem structures that assure superior performance of a specific method have been stipulated with appropriate conclusions drawn.

OCENA EFEKTYWNOŚCI METOD OPTYMALIZACJI DLA ZADAŃ PROGRAMOWANIA 0-1 Z FUNKCJAMI WIELOMIANOWYMI

Ignaczak Michał, Dariusz Horla

W artykule poruszono zagadnienie szybkości działania metod optymalizacji dla zadań z wielomianową funckją celu i 0-1 zmiennymi decyzyjnymi. Wskazano przypadki, dla których konkretna metoda działa szybciej niż pozostałe oraz wyciągnięto wnioski odnośnie takiego stanu rzeczy.

Received: 2016-10-05
Revised: 2016-12-17
Accepted: 2017-01-10