Solving Integer and Mixed Integer Linear Problems with ABS Method

József Abaffy

John von Neumann Faculty of Informatics, Óbuda University
Bécsi út 96/b, 1034 Budapest, Hungary
abaffy.jozsef@nik.uni-obuda.hu

Szabina Fodor

Department of Computer Science, Corvinus University of Budapest
Fővám tér 13-15, 1093 Budapest, Hungary
szabina.fodor@uni-corvinus.hu

Abstract: Solving mixed integer linear programming (MILP) problems is a difficult task due to the parallel use of both integer and non-integer values. One of the most widely used solution is to solve the problem in the real space and they apply additional iteration steps (so-called cutting-plane algorithms or Gomory's cuts) to narrow down the solution to the optimal integer solution. The ABS class of algorithms is a generalized class of algorithms which, with appropriate selection of parameters, is suitable for the solution of both integer and non-integer linear problems. Here we provide for the first time a complete ABS-based algorithm for MILP problems by adaptation of the ABS approach to Gomory's cutting-plane algorithm. We also provide a numerical example demonstrating the working principle of our algorithm.

Keywords: linear programming; ABS methods; mixed integer problem; cutting-plane methods; Gomory's cuts

1 Introduction

Mixed Integer Linear Programming (MILP) problems are linear programming problems in which some but not all elements of the solution vector are integer. They can be formulated in the following general form:

$$\min\{c^T x \ : \ Ax \geq b, x \geq 0, \ x_j \in Z \ \forall j \in I\}, \tag{1}$$

where no assumption related to the structure of the matrix A is made but the set **I** of the **x** variables need to be integer. MILP problems arise during everyday life whenever continuous and discrete parameters need to be optimized, ranging from basic business decisions through traffic control to guiding unmanned aerial vehicles. Despite their major importance, there is no perfect solution for MILP problems. In particular, because of the integer nature of some elements of the solution vector, general MILP algorithms are nondeterministic polynomial time (NP) hard algorithms which are not very effective in practice. Therefore, various approaches have been developed to solve MILP problems in polynomial or quasi-polynomial time.

One of the possible approaches to overcome the NP-hard nature of MILP problems is to first solve the same problem in the real space, i. e. without any constraints on whether elements of the solution vector need to be integer or not. Such modified problems are called the LP relaxation of the original problem and can be described in the following general form:

$$\min\{c^T x \ : \ Ax \geq b, \ x \geq 0\}, \tag{2}$$

The advantage of this approach is that (2) can be solved by general linear programming (LP) applications in polynomial time which are much more effective in practice. However, those solutions contain both integer and non-integer solution values which need to be separated (so-called separation problem). Since the condition of integer nature has to be met, the optimum solution has to be identified in a second step where the optimum solution vector is narrowed down to values that meet the integer requirement. This is performed by establishing a new condition that is only satisfied if the solution matrix meets the relevant integer requirement. With other words, solutions that do not satisfy the integer requirements are "cut out" of the resulting solution matrix. Therefore, such algorithms called "cutting-plane algorithms", "cuts" or, according to its first description, "Gomory's cuts". During a cutting-plane algorithm, several iteration steps are used to refine the solution matrix in order to find the optimum integer solution (a solution to the separation problem). Cutting planes are inequalities that solve the separation problem and. Such cutting planes serve to tighten the so-called LP relaxation resulting in better approximation of the convex hull of the original MILP problem. All current commercial MILP problem solving algorithms apply Gomory's cutting-plane algorithm to find the optimal integer solution.

Historically, Gomory first described an algorithm that finds the optimal solution in finite iteration steps for Integer Linear Programming (IP) in 1958 [1]. Such an algorithm solves the separation problem when $\mathbf{x}^*$ is an optimal basis of the LP relaxation. In 1960, Gomory introduced the Gomory Mixed Integer (GMI) cuts to deal with the mixed-integer case [2]. However, he never emphasized the practical use of this method, since the cutting plane algorithms converge very slowly to the optimum solution and the resulting large number of cuts results in very large LPs

with corresponding numerical difficulties. However, a major improvement came from Balas et al. [3] who re-analyzed the original Gomory mixed-integer cuts [2] and overruled the common belief that these cuts had no practical importance. In fact, a series of improvements eventually led the same group to show that Gomory 's cuts are fundamental tools for the solution of 0-1 MILP problems [4]. However, given the major importance of MILP problems, additional approaches to solving such problems or performing Gomory's cuts are still actively needed.

The ABS class of algorithms are generalized algorithms which, with appropriate selection of parameters, can be used to solve diverse mathematical problems. They were initially developed by Abaffy, Broyden and Spedicato [5-7] to solve linear systems of equations over the real space. The class was later generalized (so-called scaled ABS class) and applied also to the solution of various additional linear and nonlinear problems [8]. The ABS algorithm was applied to mathematical optimization problems such as LP problems via a certain subclass of ABS (called implicit LX) by reformulating the simplex method [9, 10]. However, those studies did not address the problem of finding an initial basis and an initial feasible solution.

It is theoretically possible that ABS-based algorithms may also be able to provide suitable solutions for MILP problems. If so, then the algorithm could take advantage of various unique features of the ABS class. For example, ABS algorithms have n inherent capability of finding cutting planes that are linearly dependent, which is a major obstacle in the algorithm presented by [4]. However, no ABS-based algorithms have yet been reported that are capable of solving an entire MILP problem.

In this paper we present a new method for solving mixed-integer problems by applying the ABS approach to Gomory's cutting plane algorithm. Since no ABS-based methods to finding the initial basis and initial feasible solution for the simplex method have yet been described, we first present an ABS-based solution for finding those parameters. In parallel, we construct the projection matrix H of the ABS class. Together with the above mentioned LX method for the ABS-based reformulation of the simplex method, these results now allow the ABS-based solution of LP problems. Next we describe a new method by applying the ABS class to Gomory's cutting plane methods. Those components are placed into a frame allowing the solution of MILP problems. Finally, we provide a numerical example to illustrate the working principles of our algorithm.

2 ABS Algorithm for Solving LP Problems

Let us consider the following modified system

$$\begin{aligned} &\min\{e^T u\} \\ &Ax + Iu = b \\ &x, u \geq 0 \end{aligned} \tag{3}$$

where **e** is the vector of all ones, and **b≥0**, (if not, then we can multiply the constraints by **-1** to achieve this) and the **u** are artificial (slack) variables. Define $\tilde{x} = \left[x^T \; u^T\right]^T$ and $\tilde{A} = [A\, I]$ so that the constraints of the modified can be written as $\tilde{A}\tilde{x} = b,\ \tilde{x} \geq 0$.

Let **B** be the indices corresponding to the artificial variables. Then **B** is a basis, since the corresponding columns of $\tilde{A}$ are I, the identity, and thus linearly independent. The corresponding basic feasible solution is $x = 0$, $u = b$. We use this to initialize the necessary parameters (i.e. the projection matrix) for the ABS-based simplex algorithm.

Algorithm 1: ***Finding an initial feasible solution***

(A1) Let $\tilde{x}_1 \in R^n$ be arbitrary, $\tilde{x}_1 = 0$, **i=1**, and $H_1 = I$, where $I \in R^{n,n}$ unit matrix.

(B1) Calculate the following vectors

$$s_i = H_i \tilde{a}_i, \tag{4}$$

$$p_i = \tilde{a}_i^T \tilde{x}_i - b_i. \tag{5}$$

If $s_i \neq 0$, then go to C1.

If $s_i = 0$ and $p_i = 0$ then $\tilde{x}_{i+1} = \tilde{x}_i$, $H_{i+1} = H_i$ go to F1. (The *ith* equation linearly depends on the previous ones.)

(C1) Compute the search vector

$p_i = H_i^T e_{m+i}$, where e_{m+i} is the **m+ith** unit vector.

(D1) Update the approximation of the solution by

$$\tilde{x}_{i+1} = \tilde{x}_i - \alpha_i p_i \text{, where } \alpha_i = \frac{r_i}{\tilde{a}_i^T p_i}.$$

(E1) Update the H_i matrix by

$$H_{i+1} = H_i - \frac{s_i p_i^T}{p_i^T \tilde{a}_i}$$

(F1) If **i=m**, then STOP. $\tilde{x}_{m+1}$ is a solution of the system.

If **i≠m**, then increment the index **i** by one and go to B1.

Remark 2.1 The algorithm is well-defined as the conditions $e_{m+i}^T s_i \neq 0$ and the $\tilde{a}_i^T p_i \neq 0$ are trivially true.

Remark 2.2 The original ABS algorithm contains a case $s_i = 0$ and $r_i \neq 0$ in step B1, which means the incompatibility of the system of equations. This never happens in our case as our system has the obvious solution $x^T = [\mathbf{0},...,\mathbf{0},\mathbf{b}^{\mathbf{T}}]$.

Remark 2.3 The H_i projection matrices, generated by Algorithm 1, are Hermitian and they have the following special structure

$$\begin{bmatrix} \mathbf{0} & & \mathbf{...} & & & \mathbf{0} \\ \mathbf{.} & & & & & \mathbf{.} \\ \mathbf{0} & & \mathbf{...} & & & \mathbf{0} \\ * & \mathbf{...} & * & \mathbf{1} & \mathbf{...} & \mathbf{1} \\ \mathbf{.} & & \mathbf{.} & \mathbf{.} & & \mathbf{.} \\ * & \mathbf{...} & * & \mathbf{0} & \mathbf{..} & \mathbf{1} \end{bmatrix},$$

where * indicates possible non-zero elements. Furthermore, the indexes of the zero rows are the basis elements, and the indexes of the non-zero rows are the non-basis ones. [10]

Remark 2.4 In general, finding an initial basis for the standard problem is as difficult as finding an optimal solution for the original problem. Please refer to Abaffy et al. [11] for finding an initial feasible solution, where the initial bases and the **H** projection matrix are parallely calculated saving a number of operations.

Let's use the following notation. The indexes of the non-identically zero rows of the $\mathbf{H_i}$ matrix is $\mathbf{B_i}$ and the indexes of the zero rows of $\mathbf{H_i}$ is $\mathbf{N_i}$.

The simplex algorithm performs successive iteration steps (pivot operations) to gradually improve the feasible intermediate solution.

Once the pivot column has been selected, the choice of pivot row is largely determined by the requirement that the resulting solution is feasible. This means using the ABS terminology that we need to minimize the expression $\frac{x^T e_k}{e_k^T H_i^T e_{N^*}} \mid k \in B_i$ such that $e_k^T H_i^T e_{N^*} > 0$ [10].

The ABS formulation of the simplex method is defined by the following procedure.

Algorithm 2: ***ABS based simplex method (Finding the optimal solution in real space)***

(A2) Let $x_1 \in R^n$ be a feasible solution of problem (2). H_1 is the projection matrix for this feasible solution, and **i=1**.

(B2) Compute the search vector $\mathbf{p_i}$ by

$$p_i = H_i^T e_{N^*}, \quad \text{where} \quad e_{N^*} \text{ is unit vector, where } c^T H_i^T e_{N^*} = \min\{c^T H_i^T e_j \mid j \in N_i\}.$$

(C2) Update the solution

$$x_{i+1} = x_i - \alpha_i p_i \quad , \quad \text{where}$$

$$\alpha_i = \frac{-x_{B^*}}{e_{B^*}^T H_i^T e_{N^*}} = \min\{\frac{-x_k}{e_k^T H_i^T e_{N^*}} \mid k \in B_i \text{ and } e_k^T H_i^T e_{N^*} > 0\}.$$

(D2) Update the projection matrix

$$H_{i+1} = H_i - \frac{(H_i e_{B^*} - e_{B^*}) e_{N^*}^T H_i}{e_{N^*}^T H e_{B^*}}$$

(E2) If $c^T H_{i+1}^T > 0$ then STOP. $\mathbf{x_{i+1}}$ is the optimal solution.

If $c^T H_{i+1}^T$ vector has negative element, then the $\mathbf{x_{i+1}}$ solution is not optimal, $\mathbf{H_i=H_{i+1}}$, $\mathbf{x_i=x_{i+1}}$and go to step B2.

Remark 2.5 Computing the $\mathbf{p_i}$ vector means that we determine the entering variable into the basis in step B5 and updating the solution with the selected $\mathbf{e_{B^*}}$ means that we select the leaving variable from the basis.

Remark 2.6 The selection of the entering variable in step B2 is taken as the column with least relative cost. In the ABS approach it corresponds to the minimization of the expression $c^T H_i^T e_j$. However, the minimization can be changed to maximization or other selection strategy.

Remark 2.7 Residual cost vector is $r = H_i c$ in every iteration step [10].

There are two conceptually different approaches to solving LP problems in real space. The simplex algorithm first finds a basic solution that is feasible (i. e. the solution is nonnegative) and the following iteration steps refine this solution towards the optimum solution while maintaining feasibility of the intermediate solutions throughout the entire procedure. In contrast, the dual simplex algorithm first finds a basic solution that is primal infeasible (there are certain negative values) but dual feasible and the following iteration steps are similarly feasible in the dual but not in the primal case except for the last iteration step in which the final solution will be both primal and dual feasible. With other words, the simplex algorithm performs the entire iteration procedure in the primal feasible space whereas the dual simplex algorithm does so in the dual feasible (but primal infeasible) space and only the final step will ensure primal feasibility. Nevertheless, both algorithms are able to find the same final (optimal) solution.

An important feature of the dual simplex algorithm is that it is most suitable to solve problems where a dual feasible solution can easily be found, or when additional conditions (change of parameters, additional constraints) are set after having obtained an initial fasible solution for the original problem.

As mentioned above, Algorithm 2 finds a feasible optimum solution for problem (3) in the real space using the principles of the simplex algorithm. In the following section we re-formulate the ABS algorithm to also perform the dual simplex method in the real space (Algorithm 3). This algorithm will then be used to re-optimize the intermediate solution following the introduction of a new integer condition in Algorithm 4.

Algorithm 3: ***ABS based dual simplex method (Re-optimizing with dual simplex method)***

(A3) Let $x_1 \in R^n$ a dual feasible solution of the problem (2), H_1 is the projection matrix for this dual feasible solution, and **i=1**.

(B3) (*Selection of the leaving variable.*) Find an index (**N**$^{*)}$ with a negative right-hand-side constant. If more than one value is negative then select

$$N^* = \min\{x_j \mid x_j < 0 \;\; j \in B_i\}.$$

(C3) (*Determining the entering variable.*) Let $K_i = (I\text{-}H_i)$, where $I \in R^{n,n}$ unit matrix. Find the index B^* where

$B^* = \min\{\frac{-c^T H_i^T e_k}{e_k^T K_i^T e_{N^*}} \mid k \in N_i \; and \; e_k^T K_i^T e_{N^*} < 0\}$, where e_{N^*} and e_k are unit vector.

(D3) (*Change the basis.*)

Update the solution

$x_{i+1} = x_i - \alpha_i p_i$, where $\alpha_i = \frac{-x_{B^*}}{e_{B^*}^T H_i^T e_{N^*}}$ and $p_i = H_i^T e_{N^*}$

Update the projection matrix

$$H_{i+1} = H_i - \frac{(H_i e_{B^*} - e_{B^*}) e_{N^*}^T H_i}{e_{N^*}^T H e_{B^*}}$$

(E3) (Feasibility test)

If all entries, $\mathbf{x_{i+1}}>0$ are nonnegative the solution is primal feasible, so STOP $\mathbf{x_{i+1}}$ is the optimal solution.

If $\mathbf{x_{i+1}}$ vector has negative element, then the $\mathbf{x_{i+1}}$ solution is not optimal, $\mathbf{H_i=H_{i+1}}$, $\mathbf{x_i=x_{i+1}}$and go to step B3.

Remark 2.7 In step B3, there are several strategies for choosing the index of the leaving variable ($\mathbf{N}^*$). We select the most negative one, but selecting the first negative element has also been proposed.

3 ABS Algorithm for Solving Integer and Mixed Integer Problems

Here we will present our ABS-based algorithm to solve integer and mixed integer LP problems. Our basic idea is to apply Gomory cutting plane methods to add a linear constraint to exclude any non-integer optimal solutions. Let an MILP problem be formulated as in (2). The method proceeds by first dropping the requirement that certain $\mathbf{x_i}$ be integer and solving the associated linear programming problem. If the solution found does not satisfy to the integer condition, then we add constraints (cuts) to the already solved LP. While such constraints can make the primal solution infeasible, they do not affect feasibility of the dual solution. We can therefore simply add the constraint and continue running the dual LP algorithm from the current solution until the primal solution again becomes feasible. The process is repeated until an integer solution is found. Cut or condition generation is a crucial step in the method. Many different strategies are known to construct the condition [3]. We implemented the pure Gomory cut to illustrate the running principle of our algorithm.

$$s_1 + \sum_{j \notin B} (\lfloor \bar{a}_{i,j} \rfloor - \bar{a}_{i,j}) x_j = \lfloor \bar{b}_i \rfloor - \bar{b}_i ,$$

where $\mathbf{s_1 \geq 0}$ is a new slack variable, and $\bar{a}_{i,j}$ denotes entry of the optimal tableau in the **ith** row and the **jth** column.

Algorithm 4 ***Solving Integer and Mixed Integer LP.***

(A4) Initialization:

Rephrase the mixed or integer LP that we drop the integrity restriction for the variable.

(B4) Find an initial feasible solution for the new problem. (use Algorithm-1)

(C4) Solve the LP relaxation LP_0 problem (use Algorithm-2)

If the relaxation does not have optimal solution, then STOP. Denote $\mathbf{x}^*$ is an optimal vertex.

(D4) If $\mathbf{x}^*$ is integer then STOP, otherwise

Choose the first (i.e., highest) row **ith** where the optimal solution ($\mathbf{x}^*$) is not integral. (Note that this includes the **zeroth** row.)

Add the cut to the bottom of the optimal tableau, and add n+1 to the basis B.

(E4) Use the dual simplex algorithm starting with the previous optimal tableau extended by the Gomory cut to find the lexicographically largest feasible solution of the relaxation (use Algorithm-3)

(F4) Set i:= i+ 1. Go to D4

Remark 3.1 A cut is never based on a previous cut, so $i \neq n+1$ in step D4.

Remark 3.2 The current algorithm adds just one new line to the system in every step, and the new cut uses previous ones. Cuts can also be rewritten.

Remark 3.3 The Gomory Cutting Plane Algorithm terminates in a finite number of steps. The proof strongly utilizes the fact that we choose the first row for the new cut in step D4 [1, 2].

Remark 3.4 Note that the form the implicit LX algorithm follows he special structure of the projection matrix. Therefore the number of the non-zero rows remains fix that is it does not increase with the new cuttings.

4 Numerical Results

To illustrate the numerical feature of our algorithm, we implemented it in MATLAB R2010a on a personal computer running Microsoft Windows 7. In all cases our algorithm found the optimal solution.

Below we show an example to illustrate how our algorithm finds the solution. Consider the following integer problem [12] to

$$z = (7x_1 + 4x_2 + 3x_3 + 2x_4) \rightarrow \max$$

subject to
$$\begin{aligned} 2x_1 + x_2 - x_3 &\le 10 \\ x_1 + x_2 + 2x_3 + x_4 &\le 12 \\ x_1 + 3x_3 + 2x_4 &\le 14,\ x_1, x_2, x_{3,} x_4 \in Z^+ \end{aligned} \tag{6}$$

Define the following LP problem

$$\min_x c^T x$$

subject to $Ax \le b$, $x \ge 0$, where

$$A = \begin{bmatrix} \mathbf{2} & \mathbf{1} & \mathbf{-1} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \mathbf{2} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{3} & \mathbf{2} \end{bmatrix},\ b = \begin{bmatrix} \mathbf{10} \\ \mathbf{12} \\ \mathbf{14} \end{bmatrix},\ \mathbf{c^T} = \begin{bmatrix} \mathbf{-7} & \mathbf{-4} & \mathbf{-3} & \mathbf{-2} \end{bmatrix}$$

Introduce the $\mathbf{u_1}$, $\mathbf{u_2}$, $\mathbf{u_3}$ non-negative slack variable to obtain the following standard LP problem

$$\tilde{A} = \begin{bmatrix} \mathbf{2} & \mathbf{1} & \mathbf{-1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \mathbf{2} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \mathbf{3} & \mathbf{2} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix},\ \tilde{x} = \begin{bmatrix} \mathbf{x_1} \\ \mathbf{x_2} \\ \mathbf{x_3} \\ \mathbf{x_4} \\ \mathbf{u_1} \\ \mathbf{u_2} \\ \mathbf{u_3} \end{bmatrix},$$

$$d^T = \begin{bmatrix} \mathbf{-7} & \mathbf{-4} & \mathbf{-3} & \mathbf{-2} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

We apply our Algorithm-1(***Finding an initial feasible solution - Phase 1)*** in three steps using $\mathbf{e_5}$, $\mathbf{e_6}$, $\mathbf{e_7}$ unit vectors respectively. We obtain the following projection matrix

$$H_3^T = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{-2} & \mathbf{-1} & \mathbf{-1} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{-1} & \mathbf{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{-2} & \mathbf{-3} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{-1} & \mathbf{-2} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix},$$

and an initial feasible solution

$$x_3 = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{10} \\ \mathbf{12} \\ \mathbf{14} \end{bmatrix}.$$

We can notice that the $\mathbf{H_3}$ matrix is Hermitian and the indexes of basis $\mathbf{B_3}$**={5, 6, 7}**. Note that the number of the non-zero rows is the number of the elements of the bases. As the cost vector $c^T H_3 = \begin{bmatrix} \mathbf{-7} & \mathbf{-4} & \mathbf{-3} & \mathbf{-2} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$ has negative elements, our feasible solution is not optimal.

We apply the ABS based simplex algorithm (***Finding the optimal solution***). After four steps we obtain the optimal solution

$$x_7 = \begin{bmatrix} \mathbf{6} \\ \mathbf{\frac{2}{3}} \\ \mathbf{\frac{8}{3}} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix},$$

and our projection matrix is

$$H_7 = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\frac{1}{2} & \frac{3}{18} & \frac{3}{18} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \frac{1}{2} & -\frac{21}{18} & -\frac{3}{18} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ -\frac{1}{2} & \frac{5}{6} & -\frac{3}{18} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

The structure of the $\mathbf{H_7}$ matrix clearly shows that the indexes of the basis are $\mathbf{B_7=\{1, 2, 3\}}$. As the obtained solution is not integer we need to introduce a new slack variable s_1 and add a new equation (constraint)

$$-\frac{1}{2}x_4 - \frac{15}{18}x_5 - \frac{1}{6}x_6 - \frac{1}{6}x_7 + s_1 = -\frac{2}{3} \qquad (\mathbf{c_1})$$

The basic feature of the ABS methods is that by adding a new equation to our system the algorithm finds a solution lying at the intersection of the linear varieties of the solutions of the original and the new equation within one step. We add a new line for our projection matrix

$$H_{\tilde{7}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\frac{1}{2} & \frac{3}{18} & \frac{3}{18} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \frac{1}{2} & -\frac{21}{18} & -\frac{3}{18} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ -\frac{1}{2} & \frac{5}{6} & -\frac{3}{18} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix},$$

and we solve the new equations.

We obtain that our solution (x_8) is infeasible.

$$x_8 = \begin{bmatrix} 6 \\ \frac{2}{3} \\ \frac{8}{3} \\ 0 \\ 0 \\ 0 \\ 0 \\ -\frac{2}{3} \end{bmatrix}.$$

We need to use our Algorithm-3 to move to a feasible solution. The projection matrix for our dual simplex method is

$$H_8 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & 1 & 0 & 0 & 0 & 0 \\ -\frac{1}{2} & \frac{3}{18} & \frac{3}{18} & 0 & 1 & 0 & 0 & 0 \\ \frac{1}{2} & -\frac{21}{18} & -\frac{3}{18} & 0 & 0 & 1 & 0 & 0 \\ -\frac{1}{2} & \frac{5}{6} & -\frac{3}{18} & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

After two steps we obtain the optimal solution for the modified system.

$$x_{c_1} = \begin{bmatrix} \frac{16}{3} \\ \frac{4}{3} \\ 2 \\ \frac{2}{3} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

and our projection matrix is

$$H_{c_1} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{\frac{1}{3}} & \mathbf{-\frac{2}{3}} & \mathbf{1} & \mathbf{-\frac{4}{3}} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{\frac{2}{3}} & \mathbf{-\frac{4}{3}} & \mathbf{0} & \mathbf{-\frac{1}{3}} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{-\frac{1}{3}} & \mathbf{\frac{2}{3}} & \mathbf{0} & \mathbf{-\frac{1}{3}} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{-1} & \mathbf{1} & \mathbf{-1} & \mathbf{2} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}.$$

A new fractional solution has been found, we need to generate a new constraint, which is valid for the integer solution, but not for our current solution. The new cutting plane is

$$-\frac{2}{3}x_5 - \frac{1}{3}x_6 - \frac{1}{3}x_7 + s_2 = -\frac{1}{3} \qquad (\mathbf{c_2})$$

After solving the new equations, and using Algorithm-3 for re-optimizing the solution, we obtain a new solution.

$$x_{c_2} = \begin{bmatrix} \mathbf{\frac{11}{2}} \\ \mathbf{1} \\ \mathbf{\frac{5}{2}} \\ \mathbf{\frac{1}{2}} \\ \mathbf{\frac{1}{2}} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix},$$

and projection matrix is

$$H_{c_2} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \frac{\mathbf{1}}{\mathbf{2}} & -\mathbf{1} & -\frac{\mathbf{1}}{\mathbf{2}} & \frac{\mathbf{1}}{\mathbf{2}} & -\frac{\mathbf{1}}{\mathbf{2}} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\frac{\mathbf{1}}{\mathbf{2}} & \mathbf{1} & -\frac{\mathbf{1}}{\mathbf{2}} & \frac{\mathbf{1}}{\mathbf{2}} & -\frac{\mathbf{1}}{\mathbf{2}} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & -\mathbf{1} & \mathbf{2} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \frac{\mathbf{1}}{\mathbf{2}} & -\mathbf{1} & \frac{\mathbf{3}}{\mathbf{2}} & -\frac{\mathbf{5}}{\mathbf{2}} & \frac{\mathbf{5}}{\mathbf{2}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

The found solution is not integer, therefore we add the constraint

$$-\frac{1}{2}x_6 - \frac{1}{2}x_7 - \frac{1}{2}x_9 + s_3 = -\frac{1}{2} \qquad (\mathbf{c_3})$$

After solving the new equations, and using Algorithm-3, we obtain the solution

$$x_{C_3} = \begin{bmatrix} \mathbf{5} \\ \mathbf{2} \\ \mathbf{2} \\ \mathbf{1} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{1} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix},$$

and projection matrix is

$$H_{c_3} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & -\mathbf{2} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & -\mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\mathbf{1} & \mathbf{1} & -\mathbf{1} & \mathbf{2} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & -\mathbf{2} & \mathbf{2} & -\mathbf{3} & \mathbf{2} & \mathbf{0} & -\mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ -\mathbf{1} & \mathbf{2} & -\mathbf{1} & \mathbf{1} & -\mathbf{1} & \mathbf{0} & \mathbf{2} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}.$$

Our optimal solution is integer, therefore we found the solution for our problem (6) because the x_{C_3} is primal feasible and every components is integer.

Discussion and Conclusions

In this paper we showed that the Gomory's original cutting-plane approach can be embedded in the ABS class of algorithms. Furthermore, we implemented our new algorithm in MATLAB and an example was given to demonstrate the correctness of our method.

Though Zou and Xia described an ABS-based algorithm for solving integer LP problems [13], the published method worked only on a special case when the **A** matrix is unimodular, i.e. the determinant of **A** is 1. The constraints on the **A** matrix and the inability of that algorithm to deal with mixed integer problems strongly limited the spectrum of problems that the algorithm was able to solve.

A cucial step of our algorithm is the generation of Gomory's cuts. In the current version of our algorithm, we used Gomory's original cuts defined in the LP optimal tableau. Since a number of additional cutting strategies have also been published, we also intend to extend our algorithm t those other types of cuts. We are planning to compare them and analyze the numerical feature of them, emphasizing the possibilities of the parallelization as the ABS algorithms are suitable for parallelization.

We should mention that every cut adds a new slack variable to the system, which means that the number of columns of the projection matrix increases by one in every steps (the number of rows remains). Some results were published to avoid this problem [14] and we are planning to implement them, too.

A number of papers were published showing that ABS-based algorithms are suitable for solving integer LP and Diophantine linear systems of equations too

[15-19]. Therefore some further work should be considered including investigations of the implementation of the pure integer algorithm using Gomory's cuts [14, 20].

Acknowledgement

We thank Ferenc Forgó for helpful discussions and encouraging notes.

References

[1] R. E. Gomory, "Outline of an Algorithm for Integer Solutions to Linear Programs," Bulletin of the American Mathematical Society, Vol. 64, pp. 275-278, 1958

[2] R. Gomory, "An Algorithm for the Mixed Integer Problem," DTIC Document 1960

[3] E. Balas, S. Ceria, G. Cornuéjols, and N. Natraj, "Gomory Cuts Revisited," Operations Research Letters, Vol. 19, pp. 1-9, 1996

[4] A. Zanette, M. Fischetti, and E. Balas, "Can Pure Cutting Plane Algorithms Work?," in Integer Programming and Combinatorial Optimization, ed: Springer, 2008, pp. 416-434

[5] J. Abaffy, "A lineáris egyenletrendszerek általános megoldásának egy direkt módszerosztálya. ," Alkalmazott Matematikai Lapok, Vol. 5, 1979

[6] J. Abaffy, C. Broyden, and E. Spedicato, "A Class of Direct Methods for Linear-Systems," Numerische Mathematik, Vol. 45, pp. 361-376, 1984

[7] J. Abaffy and E. Spedicato, ABS Projection Algorithms: Mathematical Techniques for Linear and Nonlinear Equations. Chichester, West Sussex, England: Ellis Horwood, 1989

[8] J. Abaffy and E. Spedicato, "A Class of Scaled Direct Methods for Linear-Systems," Annals of the Institute of Statistical Mathematics, Vol. 42, pp. 187-201, Mar 1990

[9] E. Spedicato, Z. Xia, and L. Zhang, "The Implicit LX Method of the ABS Class," Optimization Methods and Software, Vol. 8, pp. 99-110, 1997

[10] E. Spedicato, Z. Q. Xia, and L. W. Zhang, "ABS Algorithms for Linear Equations and Optimization," Journal of Computational and Applied Mathematics, Vol. 124, pp. 155-170, Dec 1 2000

[11] J. Abaffy, L. X-J, and X. Z-Q, "A Modified Non-Simplex Active Set Method for the Standard LP Problem," Pure Mathematics and Applications, Vol. 23, pp. 1-11, 2012

[12] L. Gáspár, "Operációkutatás II.," ed. Budapest: Tankönyvkiadó, 1977

[13] M.-F. Zou and Z.-Q. Xia, "ABS Algorithms for Diophantine Linear Equations and Integer LP Problems," Journal of Applied Mathematics and Computing, Vol. 17, pp. 93-107, 2005

[14] B. Vizvári, Operációkutatási modellek: Typotex Kft, 2009

[15] H. Esmaeili, N. Mahdavi-Amiri, and E. Spedicato, "ABS Solution of a Class of Linear Integer Inequalities and Integer LP Problems," Optimization Methods & Software, Vol. 16, pp. 179-192, 2001

[16] H. Esmaeili, N. Mahdavi-Amiri, and E. Spedicato, "A Class of ABS Algorithms for Diophantine Linear Systems," Numerische Mathematik, Vol. 90, pp. 101-115, Nov 2001

[17] S. Fodor, "Symmetric and Non-Symmetric ABS Methods for Solving Diophantine Systems of Equations," Annals of Operations Research, Vol. 103, pp. 291-314, 2001

[18] S. Fodor, "ABS Class of Methods for Diophantine Systems of Equations Part I.," Quaderni del Dipartimento di Matematica, Statistica, Informatica ed Applicazioni, Universita degli Studi di Bergamo, Vol. Report No. 2000/15, pp. 1-19, 2000

[19] S. Fodor, "ABS Class of Methods for Diophantine Systems of Equations Part II.," Quaderni del Dipartimento di Matematica, Statistica, Informatica ed Applicazioni, Universita degli Studi di Bergamo, Vol. Report No. 2000/16, pp. 1-16, 2000

[20] F. Forgó, Nonconvex programming: Akadémiai Kiadó Budapest, 1988