Utz-Uwe Haus* · Matthias Köppe* · Robert Weismantel*,♯

A Primal All-Integer Algorithm Based on Irreducible Solutions

Revision: 1.90 Date: 2002/02/15 13:50:50

Abstract. This paper introduces an exact primal augmentation algorithm for solving general linear integer programs. The algorithm iteratively substitutes one column in a tableau by other columns that correspond to irreducible solutions of certain linear diophantine inequalities. We prove that various versions of our algorithm are finite. It is a major concern in this paper to show how the subproblem of replacing a column can be accomplished effectively. An implementation of the presented algorithms is given. Computational results for a number of hard 0/1 integer programs from the MIPLIB demonstrate the practical power of the method.

1. Introduction

Forty years have passed since Ralph Gomory suggested a series of primal and dual algorithms for tackling linear integer programs.

Dual type algorithms start solving a linear programming relaxation of the problem, typically with the dual simplex method. In the course of the algorithm one maintains as an invariant both primal and dual feasibility of the solution of the relaxation. While the optimal solution to the relaxation is not integral, one continues adding cutting planes to the problem formulation and reoptimizes. Within this framework, Gomory developed a method of systematically generating valid inequalities directly from a given simplex tableau, see [Gom58, Gom60]. A disadvantage of a dual-type method is, however, that intermediate stopping does not yield a primal feasible integral solution.

In contrast to dual methods, primal type algorithms always preserve integrality of the solution to the relaxation. Primal type algorithms may be further distinguished according to whether the algorithm preserves either dual feasibility or primal feasibility. Note that it is impossible to preserve integrality and both dual and primal feasibility. The only representative of the first family of primal type methods is Gomory's so-called "primal all integer algorithm" that appeared in [Gom63]. We are not aware of interesting computational experiments with this method, however.

A second alternative to design a primal type algorithm is to preserve integrality of the solution to the relaxation and simultaneously primal feasibility. In order to achieve

Otto-von-Guericke-Universität Magdeburg, Department of Mathematics/IMO, Universitätsplatz 2, 39106 Magdeburg, Germany. e-mail: {haus, mkoeppe, weismant}@imo.math.uni-magdeburg.de

Mathematics Subject Classification (1991): 90C10

* Supported by grants FKZ 0037KD0099 and FKZ 2495A/0028G of the Kultusministerium of Sachsen-Anhalt.

♯ Supported by a Gerhard-Hess-Preis and grant WE 1462 of the Deutsche Forschungsgemeinschaft, and by the European DONET program TMR ERB FMRX-CT98-0202.

this, one typically starts with a primal feasible simplex tableau. Iteratively, pivot steps are performed on entries equal to one, only. If such a pivot element cannot be detected by the standard dual simplex rule, a special cut row with this property is generated and then chosen for pivoting. When at any time the reduced cost coefficients of the tableau point into the "right direction", optimality is proved by the simplex criterion. The cuts one makes use of in this framework may be Gomory's rounding cuts as in [You68], or problem specific cuts such as facet-defining cuts for the TSP in [PH80].

Although there are integer programming instances where even finding a feasible solution is hard, in many problems one can apply heuristics to find feasible solutions easily. A primal-type method of the second type then makes "augmentation" steps, moving from one feasible solution to a better feasible solution, until optimality is reached. Many combinatorial optimization algorithms follow this scheme. To the best of our knowledge, a method of this type for Integer Programming has first been suggested by Ben-Israel and Charnes [BIC62]. They employed the fact that for any feasible solution of an integer program optimality can be proven by solving a subproblem arising from the non-basic columns of a primal feasible simplex tableau. Various specializations and variants were later given by Young [You68, You75] and Glover [Glo68]. For an overview on this subject we refer to [Hu69, GN72].

We propose in this paper a generic augmentation algorithm that uses a completely different idea in order to solve the augmentation subproblem. After turning a feasible solution into a basic feasible solution of an appropriate simplex tableau (section 2), we manipulate the set of non-basic columns until either optimality of the given solution is proved or an "applicable" improving direction, i.e., an augmentation vector, is detected. In each manipulation step, we substitute one column by new columns in a way that no feasible solutions is lost; this is what we call a "proper reformulation" of a tableau (section 3). The substitution is guided by the concept of irreducibility of lattice points. Section 4 introduces our algorithm, the Integral Basis Method. We prove that it is finite and illustrate the advantage of our algorithm on two examples when compared to a pure LP-based branch-and-bound procedure and a pure cutting-plane algorithm. In section 5 we present a variant of the algorithm that employs varying discrete relaxations of the integer program. In section 6 we turn to the study of a particular class of relaxations arising from a single tableau row. We discuss three variants that differ in their trade-off between the "strength" of the relaxation and the cost of computing the irreducible solutions.

In section 7 we describe our implementation of the Integral Basis Method for pure 0/1 integer programs. Our code makes use of a number of computationally relevant techniques that allow us to apply the method not only for verifying optimality, but also to improve suboptimal solutions by performing non-degenerate simplex pivot operations until optimality is reached. We present computational results for hard 0/1 integer programs from the benchmark library MIPLIB [BCMS98] in section 8.

It turns out that the Integral Basis Method works best when started from a solution close to optimality. For this reason, we use an augmentation heuristic applicable to arbitrary integer programs, which we present in section 9. It uses the Lovász lattice reduction algorithm to find short augmentation vectors in a certain lattice with regard to varying elliptical norms. Finding short lattice vectors is a prominent application of lattice basis reduction. Being able to solve this subproblem is at the core of Lenstra's algorithm for IP in fixed dimension [Len83]; it has furthermore been applied to guide a branching-on-hyperplanes algorithm [CRSS93]. Short lattice vectors are also important for problem reformulation, see [AHL00].

We are, however, not aware of an earlier application of lattice basis reduction to find short augmentation vectors. The combination of the augmentation heuristic with the Integral Basis Method yields an augmentation framework, which we successfully apply to solve MIPLIB problems "from scratch".

We remark that a short outline of our algorithm has already appeared in the paper [HKW01]. For a combinatorial variant of our method we refer to [GHK+01], where proper reformulations of stable set problems are determined by graph-theoretic means.

The work [BP75] of Balas and Padberg can be regarded as an early predecessor of our method. They exploit properties of the simplex tableau in the case of set partitioning problems, whereas our method works for general integer programs.

2. From Integer Programming to Basic Feasible Solutions

We assume that the reader is familiar with the basics of linear programming and the simplex algorithm. We consider linear integer programs in standard form,

$$\begin{aligned} \max\ & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.}\ & \mathbf{A}\mathbf{x} = \mathbf{b}, \\ & \mathbf{x} \geq \mathbf{0}, \\ & \mathbf{x} \in \mathbf{Z}^n \end{aligned} \tag{IP}$$

with integral data $\mathbf{c} \in \mathbf{Z}^n$, $\mathbf{A} \in \mathbf{Z}^{m\times n}$, and $\mathbf{b} \in \mathbf{Z}^m$. For the ease of presentation we assume that $\mathbf{A}$ has full row rank. In particular, $m \leq n$. Let

$$\mathscr{F} := \{\, \mathbf{x} \in \mathbf{Z}^n_+ : \mathbf{A}\mathbf{x} = \mathbf{b} \,\}$$

denote the feasible region of (IP).

It is our intention to design an algorithm that is based upon the assumption that a feasible solution is available. Iteratively the following problem is to be solved:

Problem 2.1 (The Augmentation Problem). Given a point $\mathbf{x} \in \mathscr{F}$, find a point $\mathbf{y} \in \mathscr{F}$ such that $\mathbf{c}^\top\mathbf{y} > \mathbf{c}^\top\mathbf{x}$, or provide a proof that $\mathbf{x}$ maximizes $\mathbf{c}^\top$ over $\mathscr{F}$.

A first step of our algorithm for solving the augmentation problem is to convert a feasible solution $\mathbf{x}^0 \in \mathscr{F}$ into a basic feasible solution of a simplex tableau. The reason is that the framework of simplex tableaux gives us a certificate for optimality, once the reduced cost are all pointing into the "right direction."

In order to convert a feasible solution into a basic feasible solution of a simplex tableau we proceed as follows. Let x_i be a variable which is supposed to attain a value x_i^0 in the basic feasible solution. We represent x_i as the sum of two non-negative variables: the first is again called x_i and has an upper bound x_i^0, the second is denoted by y_i (with no upper bound). This construction yields an equivalent formulation $\mathscr{F}'$ of (IP) in an extended variable space, which has a basic feasible solution corresponding to the feasible $\mathbf{x}^0 \in \mathscr{F}$ solution in the original space. The details are given in Table 1.

Table 1. Procedure to generate a basic feasible solution

1: **input** $\mathbf{x}^0$ feasible. W.l.o.g. $x_i^0 > 0$ for $i = 1, \dots, k$, $x_i^0 = 0$ for $i = k+1, \dots, n$.
2: **output** A modified system $\mathscr{F}'$ describing the feasible set in an extended space and a feasible solution $(\mathbf{x}^0, \mathbf{s}^0, \mathbf{t}^0, \bar{\mathbf{x}}^0, \mathbf{y}^0) \in \mathscr{F}'$.
3: Transform the linear integer program to be given in the form

$$
\begin{aligned}
\max\ & \bar{\mathbf{c}}^\top \mathbf{x} \\
\text{s.t.}\ & \mathbf{A}\mathbf{x} + \mathbf{I}\mathbf{s} = \mathbf{b}, \\
& -\mathbf{A}\mathbf{x} + \mathbf{I}\mathbf{t} = -\mathbf{b}, \\
& \mathbf{x} \geq \mathbf{0}, \quad \mathbf{s} \geq \mathbf{0}, \quad \mathbf{t} \geq \mathbf{0} \\
& \mathbf{x} \in \mathbf{Z}^n,\ \mathbf{s} \in \mathbf{Z}^m,\ \mathbf{t} \in \mathbf{Z}^m.
\end{aligned}
\tag{2.1}
$$

Set the slack variables $\mathbf{s}^0 = \mathbf{t}^0 = \mathbf{0}$.
4: **for all** $i \in \{1, \dots, k\}$ **do**
5: Introduce two new (non-negative) variables $\bar{x}_i, y_i \in \mathbf{Z}_+$, corresponding to columns with indices $n + 2m + i$ and $n + 2m + k + i$. Set $\mathbf{A}_{\cdot,n+2m+i} = \mathbf{0}$, $\mathbf{A}_{\cdot,n+2m+k+i} = \mathbf{A}_{\cdot,i}$ and $c_{n+2m+i} = 0$, $c_{n+2m+k+i} = c_i$.
6: Introduce the constraint

$$x_i + \bar{x}_i = x_i^0.$$

7: Define $\bar{\mathbf{x}}_i^0 = \mathbf{y}_i^0 = \mathbf{0}$ for $i = 1, \dots, n$.
8: Return the modified system $\mathscr{F}'$ describing the feasible set,

$$
\begin{aligned}
\max\ & \mathbf{c}^\top \mathbf{x} + (c_1, \dots, c_k)\mathbf{y} \\
\text{s.t.}\ & \mathbf{A}\mathbf{x} + \mathbf{I}\mathbf{s} + (\mathbf{A}_{\cdot,1}, \dots, \mathbf{A}_{\cdot,k})\mathbf{y} = \mathbf{b}, \\
& -\mathbf{A}\mathbf{x} + \mathbf{I}\mathbf{t} - (\mathbf{A}_{\cdot,1}, \dots, \mathbf{A}_{\cdot,k})\mathbf{y} = -\mathbf{b}, \\
& x_i + \bar{x}_i = x_i^0, \quad i = 1, \dots, k \\
& \mathbf{x} \in \mathbf{Z}_+^n,\ \mathbf{s} \in \mathbf{Z}_+^m,\ \mathbf{t} \in \mathbf{Z}_+^m,\ \bar{\mathbf{x}} \in \mathbf{Z}_+^k,\ \mathbf{y} \in \mathbf{Z}_+^k.
\end{aligned}
\tag{2.2}
$$

Return the feasible solution $(\mathbf{x}^0, \mathbf{s}^0, \mathbf{t}^0, \bar{\mathbf{x}}^0, \mathbf{y}^0)$.

Proposition 2.2. *The feasible solution* $(\mathbf{x}^0, \mathbf{s}^0, \mathbf{t}^0, \bar{\mathbf{x}}^0, \mathbf{y}^0) \in \mathscr{F}'$ *defines an integral feasible basic solution for* (2.2)*.*

Proof. The system (2.2) consists of $2m+k$ equations and $n+2m+2k$ variables. A set of column indices $B \subseteq \{1, \dots, n+2m+2k\}$ with $|B| = 2m+k$ is a basis if the submatrix $\tilde{\mathbf{A}}_B$ of the constraint matrix

$$
\begin{array}{cc}
 & \begin{array}{ccccc} \mathbf{x} & \mathbf{s} & \mathbf{t} & \bar{\mathbf{x}} & \mathbf{y} \end{array} \\
\tilde{\mathbf{A}} = & \begin{pmatrix}
\mathbf{A} & \mathbf{I}_m & & & \mathbf{A}_{1,\dots,k} \\
-\mathbf{A} & & \mathbf{I}_m & & -\mathbf{A}_{1,\dots,k} \\
\mathbf{I}_k & & & \mathbf{I}_k &
\end{pmatrix}
\end{array}
$$

is invertible. We select the slack variables $s_1, \dots, s_m$, $t_1, \dots, t_m$ and the regular variables $x_1, \dots, x_k$. The corresponding column indices constitute the set B. Obviously, B is a basis. Moreover, the matrix $\tilde{\mathbf{A}}_B^{-1}$ is integral. Since the right hand side vector $\tilde{\mathbf{b}} = (\mathbf{b}, -\mathbf{b}, \mathbf{x}^0)$ is integral, the associated solution $(\mathbf{x}^0, \mathbf{s}^0, \mathbf{t}^0, \bar{\mathbf{x}}^0, \mathbf{y}^0) = (\mathbf{z}_B, \mathbf{z}_N)$ with

$$\mathbf{z}_B = \tilde{\mathbf{A}}_B^{-1} \tilde{\mathbf{b}}, \quad \mathbf{z}_N = \mathbf{0}$$

is feasible and integral. □

Example 2.3. Consider the following system in three variables,

$$\begin{aligned} &3x_1 + 5x_2 + 7x_3 = 13,\\ &x_1 \in \mathbf{Z}_+,\ x_2 \in \mathbf{Z}_+,\ x_3 \in \mathbf{Z}_+. \end{aligned}$$

We start with the feasible solution $\mathbf{x}^0 = (1, 2, 0)$ and apply the procedure of Table 1. Splitting the equation into inequalities and introducing slack variables gives

$$\begin{aligned} 3x_1 + 5x_2 + 7x_3 + s \qquad &= \ \ 13,\\ -3x_1 - 5x_2 - 7x_3 \qquad + t &= -13,\\ x_1, x_2, x_3, s, t \in \mathbf{Z}_+. \end{aligned}$$

Then we introduce variables $\bar{x}_1$ and y_1,

$$\begin{array}{l} 3x_1 + 5x_2 + 7x_3 + s \qquad\qquad + 3y_1 = \ \ 13,\\ -3x_1 - 5x_2 - 7x_3 \quad + t \qquad - 3y_1 = -13,\\ x_1 \qquad\qquad\qquad\qquad + \bar{x}_1 \qquad\quad = \ \ 1,\\ x_1, x_2, x_3, s, t, \bar{x}_1, y_1 \in \mathbf{Z}_+. \end{array}$$

Next we introduce variables $\bar{x}_2$ and y_2 so that the system becomes

$$\begin{array}{l} 3x_1 + 5x_2 + 7x_3 + s \qquad\qquad\qquad + 3y_1 + 5y_2 = \ \ 13,\\ -3x_1 - 5x_2 - 7x_3 \quad + t \qquad\qquad - 3y_1 - 5y_2 = -13,\\ x_1 \qquad\qquad\qquad\qquad + \bar{x}_1 \qquad\qquad\qquad\quad = \ \ 1,\\ \quad x_2 \qquad\qquad\qquad\qquad + \bar{x}_2 \qquad\qquad\quad = \ \ 2,\\ x_1, x_2, x_3, s, t, \bar{x}_1, \bar{x}_2, y_1, y_2 \in \mathbf{Z}_+. \end{array}$$

This constitutes the system (2.2). Associated with this system is the basic feasible solution

$$x_1 = 1,\ x_2 = 2,\ s = 0,\ t = 0 \quad \text{and} \quad x_3 = \bar{x}_1 = \bar{x}_2 = y_1 = y_2 = 0,$$

which corresponds to the tableau in the extended space,

$$\begin{array}{l} x_1 \qquad\qquad + \bar{x}_1 \qquad\qquad\qquad\qquad\quad = 1,\\ \quad x_2 \qquad\qquad + \bar{x}_2 \qquad\qquad\qquad\quad = 2,\\ \qquad s \ + 7x_3 \qquad\qquad + 3y_1 + 5y_2 = 0,\\ \qquad\ t - 7x_3 \qquad\qquad - 3y_1 - 5y_2 = 0,\\ x_1, x_2, x_3, s, t, \bar{x}_1, \bar{x}_2, y_1, y_2 \in \mathbf{Z}_+. \end{array}$$

In practice one will try to avoid adding $\left|\operatorname{supp} \mathbf{x}^0\right|$ new rows and twice as many columns to the initial problem formulation to obtain a feasible simplex basis to start with. In fact, this is possible for all integer programs where it is easy to find an *infeasible* all-integer simplex basis. For instance, for every integer program given in inequality form,

$$\begin{aligned} \max\ & \mathbf{c}^\top \mathbf{z}\\ \text{s.t.}\ & \mathbf{A}\mathbf{z} \le \mathbf{b},\\ & \mathbf{z} \in \mathbf{Z}^n_+, \end{aligned} \tag{2.3}$$

with integral data $\mathbf{c} \in \mathbf{Z}^n$, $\mathbf{A} \in \mathbf{Z}^{m\times n}$ and $\mathbf{b} \in \mathbf{Z}^m$, there is an obvious all-integer simplex basis available: the slack variables $\mathbf{s}$ for the inequalities $\mathbf{A}\mathbf{z} \le \mathbf{b}$ can be selected as basic variables, all variables $\mathbf{z}$ as non-basic variables. This basis is all-integer but infeasible in general.

Starting from such a basis, we can resort to the method of Gomory [Gom63] and Young [You68] instead of the algorithm of Table 1 to construct a feasible, all-integer basis corresponding to a known feasible integer solution. The method introduces only two new variables and one new equation; see Table 2.

Table 2. Procedure to construct a feasible all-integer basis using a Gomory – Young vector

1: **input** An integer programming problem given as a tableau corresponding to a possibly infeasible basis

$$\max c_0 + \bar{\mathbf{c}}^\top \mathbf{x}_N \;:\; \mathbf{x} \in \mathscr{F} = \left\{ \mathbf{x} \in \mathbf{Z}_+^n \;:\; (\mathbf{I}, \bar{\mathbf{A}})(\mathbf{x}_B, \mathbf{x}_N)^\top = \bar{\mathbf{b}} \right\}$$

with $\bar{\mathbf{A}} \in \mathbf{Z}^{m \times n-m}$, $\bar{\mathbf{b}} \in \mathbf{Z}^m$, $\bar{\mathbf{c}} \in \mathbf{Z}^{n-m}$, and $c_0 \in \mathbf{Z}$. A feasible solution $\mathbf{x}' \in \mathscr{F}$.

2: **output** A reformulated problem with feasible region $\mathscr{F}'$ and basis B' whose basic feasible solution corresponds to the solution $\mathbf{x}'$.

3: Calculate the difference vector $\mathbf{t} = (\mathbf{t}_B, \mathbf{t}_N)$ between the basic solution $(\bar{\mathbf{b}}, \mathbf{0})$ and $\mathbf{x}'$, $\mathbf{t} = \mathbf{x}' - (\bar{\mathbf{b}}, \mathbf{0})$.

4: Add a new variable $\hat{y}$ to the tableau with reduced cost coefficient $\bar{\gamma} = \bar{\mathbf{c}}^\top \mathbf{t}_N$ and the integral matrix column $\bar{\mathbf{a}} = \bar{\mathbf{A}} t_N = -\mathbf{t}_B$. The problem now has the form

$$\max c_0 + \bar{\mathbf{c}}^\top \mathbf{x}_N + \gamma \hat{y} \;:\; (\mathbf{x}_B, \mathbf{x}_N, \hat{y}) \in \mathscr{F}_t$$

with

$$\mathscr{F}_t := \left\{ \begin{pmatrix} \mathbf{x}_B \\ \mathbf{x}_N \\ \hat{y} \end{pmatrix} \in \mathbf{Z}_+^{n+1} \;:\; (\mathbf{I}\ \bar{\mathbf{A}}\ \bar{\mathbf{a}}) \begin{pmatrix} \mathbf{x}_B \\ \mathbf{x}_N \\ \hat{y} \end{pmatrix} = \bar{\mathbf{b}} \right\}.$$

5: Add a row $\hat{y} + \hat{s} = 1$ with new non-negative basic slack variable $\hat{s}$.

6: Put $\hat{y}$ into the basis for the enlarged system, making $\hat{s}$ non-basic instead.

7: Return the enlarged problem

$$\max c_0 + \bar{\mathbf{c}}^\top \mathbf{x}_N + \gamma \hat{y} \;:\; (\mathbf{x}_B, \mathbf{x}_N, \hat{y}, \hat{s}) \in \mathscr{F}' \tag{2.4a}$$

with

$$\mathscr{F}' := \left\{ \mathbf{y} = \begin{pmatrix} \mathbf{x}_B \\ \mathbf{x}_N \\ \hat{y} \\ \hat{s} \end{pmatrix} \in \mathbf{Z}_+^{n+2} \;:\; \begin{pmatrix} \mathbf{I} & \bar{\mathbf{A}} & \bar{\mathbf{a}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 1 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{x}_B \\ \mathbf{x}_N \\ \hat{y} \\ \hat{s} \end{pmatrix} = \begin{pmatrix} \bar{\mathbf{b}} \\ 1 \end{pmatrix} \right\} \tag{2.4b}$$

and the basis $B' = B \cup \{n+1\}$ (where $n+1$ is the variable index of $\hat{y}$).

Lemma 2.4. *The algorithm of Table 2 returns a feasible all-integer basis B' for the transformed problem* (2.4)*.*

Proof. The tableau corresponding to the basis B' returned by the algorithm has the form

$$\begin{array}{lllll} \max & & \bar{\mathbf{c}}^\top \mathbf{x}_N & - \gamma \hat{s} & + \gamma + c_0 \\ \text{s.t.} & \mathbf{x}_B & + \bar{\mathbf{A}} \mathbf{x}_N & - (\bar{\mathbf{A}} \mathbf{x}'_N) \hat{s} & = \bar{\mathbf{b}} - \bar{\mathbf{A}} \mathbf{x}'_N \\ & & \hat{y} & + \hat{s} & = 1 \\ & \multicolumn{4}{l}{\mathbf{x} \in \mathbf{Z}_+^{n+m},\ \hat{y}, \hat{s} \in \mathbf{Z}_+,} \end{array} \tag{2.5}$$

which is clearly an integer tableau. Since $\mathbf{x}' \in \mathscr{F}$, we have $\bar{\mathbf{b}} - \bar{\mathbf{A}}\mathbf{x}'_N \geq \mathbf{0}$, so it is feasible as well. □

Lemma 2.5. *The algorithm of Table 2 reformulates the problem: Every feasible solution in $\mathscr{F}$ is represented in $\mathscr{F}'$, and vice versa. Optimality of a solution is preserved in both directions.*

Proof. Let the notation be as in the algorithm shown in Table 2. It is easy to see that $\mathbf{x} \in \mathscr{F}$ implies $(\mathbf{x}, 0, 0) \in \mathscr{F}'$. Now, for $\mathbf{y}' = (\mathbf{y}_B, \mathbf{y}_N, \hat{y}, \hat{s}) \in \mathscr{F}'$ we find $\mathbf{y} := (\mathbf{y}_B, \mathbf{y}_N, \hat{y}) \in \mathscr{F}_t$. For each $\mathbf{y} \in \mathscr{F}_t$ we have $\mathbf{x} := (\mathbf{y}_B, \mathbf{y}_N + \hat{y}\,\mathbf{t}_N) \in \mathscr{F}$, and the objective function values of $\mathbf{y}$ and $\mathbf{x}$ coincide. The latter is a consequence of the definition of $\bar{\gamma}$. To check the former, we note that since the start solution $\mathbf{x}' \geq \mathbf{0}$, we have (by definition of $\mathbf{t}$) $\mathbf{t}_N \geq \mathbf{0}$. We find

$$\begin{aligned}\mathbf{y} \in \mathscr{F}_t &\iff (\mathbf{I}, \bar{\mathbf{A}}, \bar{\mathbf{a}})\,\mathbf{y} = \bar{\mathbf{b}}, \mathbf{y} \geq \mathbf{0} \iff (\mathbf{I}, \bar{\mathbf{A}}, \bar{\mathbf{A}}\mathbf{t}_N)\,\mathbf{y} = \bar{\mathbf{b}}, \mathbf{y} \geq \mathbf{0} \\ &\iff \mathbf{y}_B + \bar{\mathbf{A}}(\mathbf{y}_N + \hat{y}\,\mathbf{t}_N) = \bar{\mathbf{b}}, (\mathbf{y}_B, \mathbf{y}_N, \hat{y}) \geq \mathbf{0},\end{aligned}$$

so by setting $\mathbf{x} := (\mathbf{y}_B, \mathbf{y}_N + \hat{y}\,\mathbf{t}_N)$ we obtain $(\mathbf{I}, \bar{\mathbf{A}})\mathbf{x} = \bar{\mathbf{b}}$. Since $\mathbf{y} \in \mathscr{F}'$ implies $\mathbf{y} \geq \mathbf{0}$ and $\mathbf{t}_N \geq \mathbf{0}$ we also find $\mathbf{x} \geq \mathbf{0}$, which means $\mathbf{x} \in \mathscr{F}$.

Adding the constraint $\hat{y} \leq 1$ to $\mathscr{F}_t$ is admissible. This introduces the basic slack variable $\hat{s}$. Since by construction we have $-\mathbf{t}_B = \bar{\mathbf{a}} \leq \bar{\mathbf{b}} = \mathbf{x}_B$, this row may be selected for pivoting in $\hat{y}$. Putting $\hat{y}$ into the basis B' brings $\hat{y}$ to value 1 in the basic solution of $\mathscr{F}'$ returned by the algorithm. By the formula given above, the solution returned corresponds to $\mathbf{x}'$, so it is feasible. □

In many cases one may get by without any of the above transformations by analyzing the structure of the matrix $\mathbf{A}$. Example 2.6 describes a method for the case where all $\mathbf{x}$ variables are binary. Other special techniques can be applied when the matrix $\mathbf{A}$ contains special types of rows, for example set-packing constraints.

Example 2.6 (Basis construction in the $0/1$ *setting).* If all variables $\mathbf{x}$ in the problem are binary, one can construct a simplex basis without lifting the problem. Assume again that the feasible solution $(\mathbf{x}^0, \mathbf{s}^0)$ is given, and w.l.o.g. let $x_i^0 > 0$ for $i \in \{1, \ldots, k\}$, $x_i^0 = 0$ for $i \in \{k+1, \ldots, n\}$. Now replace each variable x_i for $i \in \{1, \ldots, k\}$ by its complement, so that the problem becomes

$$\begin{array}{lllll} \max \sum_{i=1}^k c_i & -(c_1, \ldots, c_k)\bar{\mathbf{x}}_{1,\ldots,k} & +(c_{k+1}, \ldots, c_n)\mathbf{x}_{k+1,\ldots,n} & & \\ \text{s.t.} & -(\mathbf{A}_{\cdot,1}, \ldots, \mathbf{A}_{\cdot,k})\bar{\mathbf{x}}_{1,\ldots,k} & +(\mathbf{A}_{\cdot,k+1}, \ldots, \mathbf{A}_{\cdot,n})\mathbf{x}_{k+1,\ldots,n} & +\mathbf{I}\mathbf{s} = \mathbf{b} - \sum_{i=1}^k \mathbf{A}_{\cdot,i}, \\ & \bar{\mathbf{x}} \in \mathbf{Z}_+^k,\ \mathbf{x} \in \mathbf{Z}_+^n,\ \mathbf{s} \in \mathbf{Z}_+^m. & & \end{array} \tag{2.6}$$

In this reformulated problem the solution $(\mathbf{0}, \mathbf{s})$ represents the original solution $(\mathbf{x}^0, \mathbf{s}^0)$, and it is basic feasible for the basis consisting of all $\mathbf{s}$ variables.

3. Integer Tableau Manipulations

To simplify the notation in the following we will resort to our initial system (IP). We denote again by $\mathscr{F}$ the feasible region and assume that from now on a basic feasible integer solution is known. Let B denote the feasible basis for (IP), yielding the simplex tableau

$$\begin{aligned} \max\ & \gamma + \bar{\mathbf{c}}^\top \mathbf{x}_N \\ \text{s.t.}\ & \mathbf{x}_B + \bar{\mathbf{A}}_N \mathbf{x}_N = \bar{\mathbf{b}}, \\ & \mathbf{x} \geq \mathbf{0}, \\ & \mathbf{x} \in \mathbf{Z}^n, \end{aligned} \tag{3.1}$$

where $\bar{\mathbf{b}} \in \mathbf{Z}^m_+$ and $B \cup N = \{1, \dots, n\}$ and $B \cap N = \emptyset$. Associated with this tableau is the feasible solution $(\bar{\mathbf{b}}, \mathbf{0})$ attaining an objective function value of γ.

We reformulate below the original integer program (IP) by manipulating the non-basic part of the tableau (3.1) in a way that all feasible solutions can still be represented. By performing these operations we will either detect an augmenting direction or prove that our feasible solution $(\bar{\mathbf{b}}, \mathbf{0})$ is optimal.

Definition 3.1. (a) *A vector* $\mathbf{t} \in \mathbf{Z}^n$ *is a* feasible direction *w.r.t.* $(\bar{\mathbf{b}}, \mathbf{0})$ *if* $(\bar{\mathbf{b}}, \mathbf{0}) + \mathbf{t} \in \mathscr{F}$.
(b) *A vector* $\mathbf{t} \in \mathbf{Z}^n$ *is an* augmenting direction *w.r.t.* $(\bar{\mathbf{b}}, \mathbf{0})$ *if it is a feasible direction w.r.t.* $(\bar{\mathbf{b}}, \mathbf{0})$ *and* $\bar{\mathbf{c}}^\top \mathbf{t} > 0$.

Note that in this tableau setting the basic part $\mathbf{t}_B$ of a feasible direction $\mathbf{t}$ w.r.t. $(\bar{\mathbf{b}}, \mathbf{0})$ is the negative of a non-negative integral combination of the non-basic columns:

Lemma 3.2. *For the tableau* (3.1) *a vector* $\mathbf{t} \in \mathbf{Z}^n$ *is a feasible direction w.r.t.* $(\bar{\mathbf{b}}, \mathbf{0})$ *if and only if* $\mathbf{t} = (-\bar{\mathbf{A}}_N \mathbf{z}, \mathbf{z})$ *for some*

$$\mathbf{z} \in \mathscr{F}_N := \{\, \mathbf{z} \in \mathbf{Z}^{n-m}_+ : \ \bar{\mathbf{A}}_N \mathbf{z} \leq \bar{\mathbf{b}} \,\}.$$

Proof. Let $\mathbf{t} \in \mathbf{Z}^n$ be a feasible direction w.r.t. $(\bar{\mathbf{b}}, \mathbf{0})$. Then $(\bar{\mathbf{b}}, \mathbf{0}) + \mathbf{t} \in \mathscr{F}$. This implies that $\mathbf{t}_B = -\bar{\mathbf{A}}_N \mathbf{t}_N$. With $\mathbf{z} := \mathbf{t}_N$ we have $\mathbf{z} \in \mathscr{F}_N$ such that $\mathbf{t} = (-\bar{\mathbf{A}}_N \mathbf{z}, \mathbf{z})$. For the converse, let $\mathbf{z} \in \mathscr{F}_N$ and $\mathbf{t} := (-\bar{\mathbf{A}}_N \mathbf{z}, \mathbf{z})$. Then $(\mathbf{b}, \mathbf{0}) + \mathbf{t} \in \mathscr{F}$, i.e., $\mathbf{t}$ is a feasible direction w.r.t. $(\bar{\mathbf{b}}, \mathbf{0})$. □

We will now derive another representation of the problem by eliminating one non-basic column of positive reduced cost and substitute it by a couple of other columns in a way that the basic part of every feasible direction w.r.t. $(\bar{\mathbf{b}}, \mathbf{0})$ is the negative of a non-negative integral combination of the new non-basic columns. This is what we call a "proper reformulation" of a tableau.

Definition 3.3. *For a tableau* (3.1)*, let*

$$\mathscr{F}_N = \{\, \mathbf{z} \in \mathbf{Z}^{n-m}_+ : \bar{\mathbf{A}}_N \mathbf{z} \leq \bar{\mathbf{b}} \,\}.$$

For $j \in N$ *and* $\{\mathbf{t}^1, \dots, \mathbf{t}^r\} \subseteq \mathbf{Z}^{n-m}_+$ *we define*

$$\tilde{\mathscr{F}}_N = \Big\{ (\mathbf{x}_{N\setminus\{j\}}, \mathbf{y}) \in \mathbf{Z}^{n-m-1+r}_+ : \sum_{k \in N\setminus\{j\}} \bar{\mathbf{A}}_k x_k + \sum_{i=1}^{r} (\bar{\mathbf{A}}\mathbf{t}^i) y_i \leq \bar{\mathbf{b}} \Big\}.$$

Associated with $\tilde{\mathscr{F}}_N$ *is the tableau*

$$\begin{aligned} \max\ & \gamma && + \bar{\mathbf{c}}^\top_{N\setminus\{j\}} \mathbf{x}_{N\setminus\{j\}} && + \bar{\mathbf{g}}^\top \mathbf{y} && \\ \text{s.t.}\ & \mathbf{x}_B && + \bar{\mathbf{A}}_{N\setminus\{j\}} \mathbf{x}_{N\setminus\{j\}} && + \bar{\mathbf{D}}\mathbf{y} && = \bar{\mathbf{b}}, \\ & \mathbf{x}_B \in \mathbf{Z}^m_+,\ \mathbf{x}_{N\setminus\{j\}} \in \mathbf{Z}^{n-m-1}_+,\ \mathbf{y} \in \mathbf{Z}^r_+, \end{aligned} \tag{3.2}$$

where

$$\bar{g}_i = \bar{\mathbf{c}}^\top \mathbf{t}^i, \quad i = 1, \ldots r, \tag{3.3}$$

$$\bar{\mathbf{D}}_{\cdot,i} = \bar{\mathbf{A}}\mathbf{t}^i, \quad i = 1, \ldots r. \tag{3.4}$$

We call $\tilde{\mathscr{F}}_N$ *and tableau* (3.2) *a* proper reformulation *of* $\mathscr{F}_N$ *and tableau* (3.1)*, respectively, if the following condition holds:* $\mathbf{z} \in \mathscr{F}_N$ *if and only if there exist* $\mathbf{x} \in \mathbf{Z}_+^{n-m-1}$, $\mathbf{y} \in \mathbf{Z}_+^r$ *such that* $(\mathbf{x}, \mathbf{y}) \in \tilde{\mathscr{F}}_N$,

$$z_j = \sum_{i=1}^{r} t_j^i y_i, \tag{3.5a}$$

$$z_k = x_k + \sum_{i=1}^{r} t_k^i y_i \quad \textit{for } k \in \{1, \ldots, n-m\} \setminus \{j\}. \tag{3.5b}$$

Remark 3.4. In order to save some notation, we refrain from defining the obvious generalization of this construction that consists of a simultaneous substitution of several columns (rather than the single column j).

One general way of properly reformulating a given tableau is achieved using irreducible points.

Definition 3.5. *Let*

$$S := \{\mathbf{z} \in \mathbf{Z}_+^d : \mathbf{C}\mathbf{z} \le \mathbf{f}\}, \quad \mathbf{C} \in \mathbf{Z}^{m\times n}, \quad \mathbf{f} \in \mathbf{Z}_+^m. \tag{3.6}$$

(a) *A vector* $\mathbf{z} \in S$ *is* reducible *if* $\mathbf{z} = \mathbf{0}$ *or there exist* $k \ge 2$ *vectors* $\mathbf{z}^1, \ldots, \mathbf{z}^k \in S \setminus \{\mathbf{0}\}$ *and integral multipliers* $\lambda_1, \ldots, \lambda_k \ge 1$ *such that* $\mathbf{z} = \sum_{i=1}^k \lambda_i \mathbf{z}^i$*. Otherwise,* $\mathbf{z}$ *is* irreducible.
(b) *An* integral generating set *of* S *is a subset* $S_0 \subseteq S$ *such that every vector* $\mathbf{z} \in S$ *is a non-negative integral linear combination of vectors of* S_0*. It is called a* basis *if it is minimal w.r.t. inclusion.*

Note that in the homogeneous situation (i.e., when $\mathbf{f} = \mathbf{0}$) an integral basis is known under the name *Hilbert basis* of the pointed cone $\mathscr{C} = \{\mathbf{z} \in \mathbf{R}_+^d : \mathbf{C}\mathbf{z} \le \mathbf{0}\}$. We deal here with the inhomogeneous case, where the situation is slightly more complicated because S is not closed under non-negative integral combinations.

Resorting to the notion of irreducible points we can now implement a proper reformulation of a tableau.

Lemma 3.6. *Let* $\mathscr{F}_N$ *be defined as in Lemma 3.2. For* $j \in N$ *let* $\{\mathbf{t}^1, \ldots, \mathbf{t}^r\}$ *be the set of all the irreducible solutions in* $\mathscr{F}_N$ *such that* $t_j^i > 0$ *for* $i \in \{1, \ldots, r\}$*. Then*

$$\tilde{\mathscr{F}}_N = \left\{ (\mathbf{x}_{N\setminus\{j\}}, \mathbf{y}) \in \mathbf{Z}_+^{n-m-1+r} : \sum_{k\in N\setminus\{j\}} \bar{\mathbf{A}}_{\cdot,k} x_k + \sum_{i=1}^{r} (\bar{\mathbf{A}} t^i) y_i \le \bar{\mathbf{b}} \right\}.$$

is a proper reformulation of $\mathscr{F}_N$.

Proof. Let $\mathbf{z}_{N\setminus\{j\}} \in \mathbf{Z}_+^{n-m-1}$ and $z_j \in \mathbf{Z}_+$ such that $\mathbf{z}_N \in \mathscr{F}_N$. If $z_j = 0$, we have that $(\mathbf{z}_{N\setminus\{j\}}, \mathbf{0}) \in \tilde{\mathscr{F}}_N$, and the conditions (3.5) are trivially satisfied. Otherwise, $z_j > 0$. Let H be the set of all the irreducible solutions in $\mathscr{F}_N$. Then $\{\mathbf{t}^1, \dots, \mathbf{t}^r\} \subseteq H$. Hence we can write H in the following form,

$$H = \{\mathbf{h}^1, \dots, \mathbf{h}^l\} \cup \{\mathbf{t}^1, \dots, \mathbf{t}^r\},$$

where $h_j^i = 0$ for all $i = 1, \dots, l$. We conclude that

$$\mathbf{z}_N = \sum_{i=1}^{l} \lambda_i \mathbf{h}^i + \sum_{i=1}^{r} y_i \mathbf{t}^i$$

with $\lambda_i \in \mathbf{Z}_+$ for all $i = 1, \dots, l$ and $y_i \in \mathbf{Z}_+$ for all $i = 1, \dots, r$. Let $\mathbf{x}_N = \sum_{i=1}^{l} \lambda_i \mathbf{h}^i$. Then, $x_j = 0$. Hence,

$$(\mathbf{x}_{N\setminus\{j\}}, \mathbf{y}) \in \tilde{\mathscr{F}}_N,$$

and the conditions (3.5) hold.

For the converse direction, let $\mathbf{x} \in \mathbf{Z}_+^{n-m-1}$, $\mathbf{y} \in \mathbf{Z}_+^r$ such that $(\mathbf{x}, \mathbf{y}) \in \tilde{\mathscr{F}}_N$. Let $\mathbf{z}$ be defined by (3.5). Then $\mathbf{z} \in \mathbf{Z}_+^{n-m}$, and

$$\begin{aligned} \bar{\mathbf{A}}_N \mathbf{z} &= \sum_{k \in N\setminus\{j\}} \bar{\mathbf{A}}_{\cdot,k} z_k + \bar{\mathbf{A}}_{\cdot,j} z_j \\ &= \sum_{k \in N\setminus\{j\}} \bar{\mathbf{A}}_{\cdot,k} x_k + \sum_{k \in N\setminus\{j\}} \sum_{i=1}^{r} \bar{\mathbf{A}}_{\cdot,k} t_k^i y_i + \sum_{i=1}^{r} \bar{\mathbf{A}}_{\cdot,j} t_j^i y_i \\ &= \sum_{k \in N\setminus\{j\}} \bar{\mathbf{A}}_{\cdot,k} x_k + \sum_{i=1}^{r} (\bar{\mathbf{A}} \mathbf{t}^i) y_i \le \bar{\mathbf{b}}, \end{aligned}$$

hence $\mathbf{z} \in \mathscr{F}_N$. □

Lemma 3.6 suggests an algorithm for manipulating our initial tableau: if all the reduced cost are negative, we have a proof that our given basic feasible integer solution is optimal. Otherwise, we select a non-basic variable with index j. We eliminate this column and introduce r new non-basic columns $\bar{\mathbf{A}}\mathbf{t}^1, \dots, \bar{\mathbf{A}}\mathbf{t}^r$ that correspond to the irreducible solutions $\mathbf{t}^1, \dots, \mathbf{t}^r$ in $\mathscr{F}_N$ such that $t_j^i > 0$ for all $i = 1, \dots, r$. According to Lemma 3.6 this step performs a proper reformulation of the original tableau. This means that every feasible direction w.r.t. our basic feasible solution is representable by the new non-basic variables in the modified tableau. We obtain the rudimentary form of the *Integral Basis Method*.

4. The Integral Basis Method

With the preparations that we met in Section 3 we are ready to analyze the basic form of the *Integral Basis Method*. We start with tableau (3.1) and maintain the problem in tableau form throughout the algorithm.

Let $j \in N$ be the index of a non-basic variable with $\bar{c}_j > 0$ and $\bar{\mathbf{A}}_N \mathbf{e}^j \not\le \bar{\mathbf{b}}$, i.e., $\mathbf{e}^j$ is the non-basic part of a vector that promises to be "improving" but which is not a feasible direction w.r.t. the basic solution $(\bar{\mathbf{b}}, \mathbf{0})$. We will replace column $\bar{\mathbf{A}}_{\cdot j}$ by a set of columns corresponding to new non-basic integral variables according to Lemma 3.6. The new columns to be added are computed by determining the set of all irreducible integral solutions to a system (3.6) of linear inequalities.

Table 3 summarizes the Integral Basis Method.

Table 3. The Integral Basis Method

input A tableau (3.1) with data $\bar{\mathbf{A}} \in \mathbf{Q}^{m \times n-m}$, $\bar{\mathbf{b}} \in \mathbf{Z}_+^m$ and reduced cost vector $\bar{\mathbf{c}}$. Let B be the basis and N the non-basic variables.

$$\mathscr{F}_N = \left\{ \mathbf{z} \in \mathbf{Z}_+^{n-m} : \bar{\mathbf{A}}_N \mathbf{z} \le \bar{\mathbf{b}} \right\}.$$

output a certificate that the point $(\bar{\mathbf{b}}, \mathbf{0})$ is optimal or an augmenting direction w.r.t. $(\bar{\mathbf{b}}, \mathbf{0})$.
$N_+ \leftarrow \{ i \in N : \bar{c}_i > 0 \}$.
while $N_+ \neq \emptyset$ **do**
  Select $j \in N_+$.
  if $\mathbf{e}^j \in \mathscr{F}_N$ **then**
    return the vector $(-\bar{\mathbf{A}}_j, \mathbf{e}^j)$ as an augmenting direction w.r.t. $(\bar{\mathbf{b}}, \mathbf{0})$;
  else
    Determine the set $I(j) = \{\mathbf{t}^1, \ldots, \mathbf{t}^r\}$ of all irreducible solutions $\mathbf{z} \in \mathscr{F}_N$ with $z_j > 0$.
    Delete the column with index j.
    Introduce r columns corresponding to the elements in $I(j)$, i.e., the new column i has entries $\bar{\mathbf{A}}\mathbf{t}^i$ and reduced cost $\bar{\mathbf{c}}^\top \mathbf{t}^i$.
    Set $n \leftarrow n - 1 + r$ and update N and N_+.
return "Optimal."

From our discussions in Section 3 it follows that the Integral Basis Method is solving the underlying integer programming problem. It remains to verify whether the algorithm is finite. This result requires a little bit of further preparation. We first need to check that each reformulation step is finite. This follows from Theorem 4.1.

Theorem 4.1. *Let*

$$S := \{ \mathbf{z} \in \mathbf{Z}_+^d : \mathbf{C}\mathbf{z} \le \mathbf{f} \}, \quad \mathbf{C} \in \mathbf{Z}^{m \times d}, \quad \mathbf{f} \in \mathbf{Z}_+^m. \tag{4.1}$$

There exists a unique finite integral basis $\mathscr{H}(S)$ of S.

Proof. We split the proof into two parts. Part (a) shows that there exists a finite integral generating set of S. In Part (b) we establish uniqueness of an integral basis for S.

(a) We define a subset $P \subseteq \mathbf{Z}_+^{d+2m}$ by

$$P := \{ (\mathbf{x}, (\mathbf{C}\mathbf{x})^+, (\mathbf{C}\mathbf{x})^-) : \mathbf{x} \in S \setminus \{\mathbf{0}\} \},$$

where $\mathbf{z}^+$ and $\mathbf{z}^-$ denote the positive and negative part of $\mathbf{z}$, respectively, $\mathbf{z} = \mathbf{z}^+ - \mathbf{z}^-$. By the Gordan Lemma there exists a unique minimal and finite set

$$P_0 = \{(\mathbf{x}^1, (\mathbf{C}\mathbf{x}^1)^+, (\mathbf{C}\mathbf{x}^1)^-), \ldots, (\mathbf{x}^t, (\mathbf{C}\mathbf{x}^t)^+, (\mathbf{C}\mathbf{x}^t)^-)\}$$

of elements in P such that for every vector $\mathbf{p} \in P$ there exists a vector in P_0 dominated by $\mathbf{p}$, i.e., there is an index $j \in \{1, \dots, t\}$ with $\mathbf{p} \geq (\mathbf{x}^j, (\mathbf{C}\mathbf{x}^j)^+, (\mathbf{C}\mathbf{x}^j)^-)$. We claim that the set $\{\mathbf{x}^1, \dots, \mathbf{x}^t\}$ is an integral generating set of S.

By definition, $\{\mathbf{x}^1, \dots, \mathbf{x}^t\} \subseteq S$. Let $\mathbf{y} \in S \setminus \{\mathbf{0}\}$. Then there exists an index $j \in \{1, \dots, t\}$ such that $(\mathbf{y}, (\mathbf{C}\mathbf{y})^+, (\mathbf{C}\mathbf{y})^-) \geq (\mathbf{x}^j, (\mathbf{C}\mathbf{x}^j)^+, (\mathbf{C}\mathbf{x}^j)^-)$. For all row indices $i \in \{1, \dots, m\}$, the following holds: If $(\mathbf{C}\mathbf{y})_i \geq 0$, then $(\mathbf{C}\mathbf{x}^j)^-_i \leq (\mathbf{C}\mathbf{y})^-_i = 0$, thus $(\mathbf{C}\mathbf{x}^j)_i \geq 0$ and $(\mathbf{C}\mathbf{y})_i - (\mathbf{C}\mathbf{x}^j)_i \leq (\mathbf{C}\mathbf{y})_i \leq f_i$. Otherwise, we have $(\mathbf{C}\mathbf{y})_i \leq 0$ which implies $(\mathbf{C}\mathbf{x}^j)_i \leq 0$, and therefore $(\mathbf{C}\mathbf{y})_i - (\mathbf{C}\mathbf{x}^j)_i = (\mathbf{C}\mathbf{x}^j)^-_i - (\mathbf{C}\mathbf{y})^-_i \leq 0 \leq f_i$ holds. This yields

$$\mathbf{C}(\mathbf{y} - \mathbf{x}^j) = \mathbf{C}\mathbf{y} - \mathbf{C}\mathbf{x}^j \leq \mathbf{f}.$$

Moreover, $\mathbf{y} - \mathbf{x}^j \in \mathbf{Z}^d_+$ and therefore $\mathbf{y}' := \mathbf{y} - \mathbf{x}^j \in S$. If $\mathbf{y}' \neq \mathbf{0}$, apply the previous arguments iteratively on $\mathbf{y}'$ instead of $\mathbf{y}$. Since the ℓ_1-norms decrease in every step by at least 1, this procedure terminates, showing that $\mathbf{y}$ is a non-negative integral combination of the vectors $\mathbf{x}^1, \dots, \mathbf{x}^t$.

(b) It is clear that every integral generating set of S must contain the set $\mathscr{H}(S)$ of irreducible vectors of S. On account of (a), $\mathscr{H}(S)$ is finite. We claim that $\mathscr{H}(S)$ is already an integral generating set of S. Suppose the contrary. Let $\mathbf{y} \in S \setminus \{\mathbf{0}\}$ be a point of minimal ℓ_1-norm that cannot be represented as a non-negative integer combination of the elements in $\mathscr{H}(S)$. By definition of $\mathscr{H}(S)$, we have $\mathbf{y} = \sum_{i=1}^k \lambda_i \mathbf{v}^i$ with $k \geq 2$ vectors $\mathbf{v}^1, \dots, \mathbf{v}^k \in S \setminus \{\mathbf{0}\}$ and integral multipliers $\lambda_1, \dots, \lambda_k \geq 1$. We obtain

$$\sum_{i=1}^k \lambda_i \|\mathbf{v}^i\|_1 = \|\mathbf{y}\|_1, \quad \|\mathbf{v}^i\|_1 > 0 \quad \text{for} \quad i = 1, \dots, k.$$

Since $\|\mathbf{v}^i\|_1 < \|\mathbf{y}\|_1$ for $i = 1, \dots, k$, all the summands $\mathbf{v}^i$ can be written as non-negative integral combinations of the elements in $\mathscr{H}(S)$, and hence, $\mathbf{y}$ too. □

Theorem 4.2. *The Integral Basis Method is finite.*

Proof. Let $j \in N_+$. We eliminate the column associated with j and introduce r new non-basic columns $\bar{\mathbf{A}}\mathbf{t}^i$ with reduced cost $\bar{\mathbf{c}}^\top \mathbf{t}^i$ that correspond to all the irreducible solutions $\{\mathbf{t}^1, \dots, \mathbf{t}^r\}$ in $\mathscr{F}_N$ such that $t^i_j > 0$ for all $i = 1, \dots, r$. Theorem 4.1 asserts that r is a finite number. The reduced cost of a new column i is $\bar{\mathbf{c}}^\top \mathbf{t}^i$. If one of these new columns has positive reduced cost, the vector

$$\mathbf{t} = (-\bar{\mathbf{A}}_N \mathbf{t}^i, \mathbf{t}^i)$$

is an augmenting direction w.r.t. $(\bar{\mathbf{b}}, \mathbf{0})$. Otherwise the number of columns with positive reduced cost of the tableau strictly decreases. Hence, after a finite number of iterations we have either detected an augmenting direction w.r.t. $(\bar{\mathbf{b}}, \mathbf{0})$ or a linear programming certificate that $(\bar{\mathbf{b}}, \mathbf{0})$ is optimal. □

We apply the Integral Basis Method to the following two pathological examples.

Example 4.3. For $k = 1, 2, \dots$ let us consider the 0/1 linear integer program

$$\begin{aligned}
\max\ & \textstyle\sum_{i=1}^{k}(x_i - 2y_i) \\
\text{s.t.}\ & 2x_i - y_i + s_i = 1 \quad \text{for } i = 1,\dots,k, \\
& x_i + \bar{x}_i = 1 \quad \text{for } i = 1,\dots,k, \\
& y_i + \bar{y}_i = 1 \quad \text{for } i = 1,\dots,k, \\
& x_i,\ y_i,\ s_i,\ \bar{x}_i,\ \bar{y}_i \in \mathbf{Z}_+ \quad \text{for } i = 1,\dots,k.
\end{aligned} \tag{4.2}$$

The obvious basic feasible solution $\mathbf{s} = \mathbf{1}$, $\bar{\mathbf{x}} = \mathbf{1}$, $\bar{\mathbf{y}} = \mathbf{1}$ (with non-basic variables $\mathbf{x} = \mathbf{0}$, $\mathbf{y} = \mathbf{0}$) is already the optimal solution of (4.2), yet LP-based branch-and-bound codes have a hard time proving its optimality: In every node, the linear-programming relaxation will yield $x_i = \frac{1}{2}$, $y_i = 0$, $s_i = 0$ for all variables that have not been fixed yet. Branching on one of these fractional x_i variables will lead to two subproblems sharing the same property. Therefore, an exponential number of branching nodes will be visited.

Our example illustrates the "combinatorial explosion" that is inherent in branch-and-bound. The example has a lot of symmetry, so branch-and-bound methods reiterate the same branching decisions in many subproblems.

In fact, for the sake of simplicity, the problem (4.2) has been constructed as a disjoint union of k copies of the same "little" program. It should be no surprise that commercial IP solvers like CPLEX 7.1 detect the simple situation in (4.2) and resolve it in the preprocessing stage, leading to the problem

$$\begin{aligned}
\max\ & \textstyle\sum_{i=1}^{k}(x_i - 2y_i) \\
\text{s.t.}\ & x_i - y_i + s_i = 0 \quad \text{for } i = 1,\dots,k, \\
& x_i + \bar{x}_i = 1 \quad \text{for } i = 1,\dots,k, \\
& y_i + \bar{y}_i = 1 \quad \text{for } i = 1,\dots,k, \\
& x_i,\ y_i,\ s_i,\ \bar{x}_i,\ \bar{y}_i \in \mathbf{Z}_+ \quad \text{for } i = 1,\dots,k,
\end{aligned} \tag{4.2$'$}$$

which can be solved without branching. Alternatively, an IP solver could simply be adding Gomory cuts $x_i - y_i \le 0$ for $i = 1, \dots, k$, in order to obtain a problem that can be solved in the root.

However, it is straightforward to come up with a more complicated example so that the available commercial codes can neither detect the decomposability of this integer program, nor reformulate the problem with preprocessing techniques.

For the ease of exposition, we continue analyzing the simple example and show that the Integral Basis Method can solve the program (4.2) with a linear number of steps. In fact, the Integral Basis Method does not depend on the a-priori knowledge how a given integer program decomposes. Hence, the Integral Basis Method will show a good performance on many more complicated instances as well.

We shall start with the basic feasible solution $\mathbf{s} = \mathbf{1}, \bar{\mathbf{x}} = \mathbf{1}, \bar{\mathbf{y}} = \mathbf{1}$, so that (4.2) serves as the corresponding tableau.

At the beginning of the algorithm, the set N_+ consists of the indices of all the non-basic variables x_i for $i = 1, \dots, k$ because their rediced cost are all positive. The algorithm selects one index from N_+, corresponding to some variable x_i. Due to the first set of rows in the tableau (4.2), we have that

$$(\mathbf{e}^i, \mathbf{0}) \notin \mathscr{F}_N = \{\, (\mathbf{x}, \mathbf{y}) \in \mathbf{Z}_+^{2k} : 2x_i - y_i \le 1,\ x_i, y_i \le 1 \text{ for } i = 1, \dots, k \,\}.$$

Therefore, the algorithm proceeds to determine the set of irreducible solutions $(\mathbf{x}, \mathbf{y})$ to $\mathscr{F}_N$ with $x_i > 0$.

The constraint $2x_i - y_i \leq 1$ implies that every solution $(\mathbf{x}, \mathbf{y}) \in \mathscr{F}_N$ with $x_i > 0$ must have $y_i > 0$. The vector $(\mathbf{x}^1, \mathbf{y}^1) = (\mathbf{e}^i, \mathbf{e}^i)$ is one such solution, and it turns out to be the only irreducible solution $(\mathbf{x}, \mathbf{y}) \in \mathscr{F}_N$ with $x_i > 0$. The algorithm deletes the column of x_i and introduces a new column corresponding to the solution $(\mathbf{e}^i, \mathbf{e}^i)$, i.e., the new column is the sum of the x_i and y_i columns. We denote the non-basic variable corresponding to the new column by x_i'. Note that x_i' has negative reduced cost, so it does not enter the set N_+.

This procedure is repeated until all the variables x_i for $i = 1, \dots, k$ have been processed. This yields the reformulated tableau

$$\begin{array}{lllll} \max & \sum_{i=1}^k(-x_i - 2y_i) & & & \\ \text{s.t.} & x_i' - y_i + s_i & = 1 & \text{for } i = 1, \dots, k, & \\ & x_i' + y_i \quad + \bar{x}_i & = 1 & \text{for } i = 1, \dots, k, & \\ & y_i \qquad + \bar{y}_i & = 1 & \text{for } i = 1, \dots, k, & \\ & x_i',\ y_i,\ s_i,\ \bar{x}_i,\ \bar{y}_i \in \mathbf{Z}_+ & & \text{for } i = 1, \dots, k, & \end{array} \tag{4.2''}$$

providing a linear-programming certificate for optimality.

Example 4.4. Let us now compare the strength of an operation of the Integral Basis Method to that of Chvátal–Gomory cuts. For $k = 1, 2, \dots$ we consider the problem

$$\begin{array}{ll} \max & x_2 \\ \text{s.t.} & kx_1 + x_2 + x_3 = k \\ & -kx_1 + x_2 + x_4 = 0 \\ & x_1, x_2, x_3, x_4 \geq 0 \\ & x_1, x_2, x_3, x_4 \in \mathbf{Z}. \end{array} \tag{CG_k}$$

There are only two integer solutions to (CG_k), namely $(0, 0, k, 0)$ and $(1, 0, 0, k)$, which are both optimal. We start with the basic feasible integer solution $(0, 0, k, 0)$. The LP solution is $(\frac{1}{2}, \frac{1}{2}k, 0, 0)$. Note that the Chvátal rank 1 closure of (CG_k) is (CG_{k-1}). Therefore the inequality $x_2 \leq 0$, which describes a facet of the integer polytope, has a Chvátal rank of k.

We will now show that the Integral Basis Method can solve problem (CG_k) in a single step. Note that (CG_k) is already a tableau corresponding to the basic feasible integer solution $\mathbf{x}$ with basic variables $x_3 = k$ and $x_4 = 0$. Hence, we can apply the Integral Basis Method directly to

$$\mathscr{F}_N = \left\{ (x_1, x_2) \in \mathbf{Z}_+^2 : kx_1 + x_2 \leq k,\ -kx_1 + x_2 \leq 0 \right\}.$$

The set N_+ only contains the index of variable x_2. According to the second constraint, $-kx_1 + x_2 \leq 0$, column x_2 could, in principle, be replaced by columns corresponding to $x_1 + 1x_2, \dots, x_1 + kx_2$; see section 6. Each of these columns however will violate the generalized upper-bound constraint given by the first constraint. Therefore, there is no irreducible solution with a strictly positive value of x_2. So the replacement columns can simply be dropped. The resulting tableau only has a column for x_1, so we have proved optimality.

5. A Variant of the Method Using Discrete Relaxations

The core of the Integral Basis Method is to perform column substitutions. In each of the substitution steps of the algorithm, we need to compute all the irreducible solutions $\mathbf{z} \in \mathscr{F}_N$ with $z_j > 0$ (for some variable index j). This can, at least in principle, be accomplished with a combinatorial Buchberger-type algorithm, see the papers [CT91, Tho95, UWZ97] and the survey [Wei98]. We omit all the details of such an algorithm here because this approach is computationally intractable in general, even for very small instances. In fact, also the finiteness proof (Theorem 4.2) suggests that the substitution steps of the algorithm are "too strong": At most $n - m$ substitution steps are needed to find an augmentation vector (or prove optimality), hence each of these steps is at least as "hard" as the augmentation problem (in the sense of computational complexity).

For these reasons, we propose a reformulation technique that is based upon systems ("discrete relaxations") that partially describe the underlying problem but for which integral bases can be easily computed by other means. In other words, we weaken the substitution steps in order to obtain an algorithm that is computationally tractable. In this section, we discuss the general algorithm and prove its finiteness (for bounded programs); in section 6, we turn to the study of a particular class of relaxations.

Definition 5.1. *For a tableau* (3.1) *let*

$$\mathscr{F}_N = \{\, \mathbf{z} \in \mathbf{Z}_+^{n-m} : \bar{\mathbf{A}}_N \mathbf{z} \le \bar{\mathbf{b}} \,\}.$$

We call a set

$$\mathscr{F}'_N = \{\, \mathbf{z} \in \mathbf{Z}_+^{n-m} : \mathbf{A}'\mathbf{z} \le \mathbf{b}' \,\}$$

a discrete relaxation *of $\mathscr{F}_N$ if $\mathscr{F}_N \subseteq \mathscr{F}'_N$.*

By using the term *discrete relaxation*, we emphasize that we are not interested in the polyhedral properties of the relaxation but rather wish to exploit the system in an integral fashion.

We are now ready to describe a variant of the Integral Basis Method that resorts to discrete relaxations; see Table 4. In fact, also this variant of the Integral Basis Method that resorts to discrete relaxations is correct. This is easily established with Lemma 5.2.

Lemma 5.2. *Let $\mathscr{F}_N$ be defined as in Lemma 3.2. For $j \in N$ let $\mathscr{F}'_N$ be a discrete relaxation of $\mathscr{F}_N$. Let $\{\mathbf{t}^1, \dots, \mathbf{t}^r\}$ be all the irreducible solutions in $\mathscr{F}'_N$ such that $t^i_j > 0$ for $i = 1, \dots, r$. Then*

$$\tilde{\mathscr{F}}_N = \Big\{ (\mathbf{x}_{N\setminus\{j\}}, \mathbf{y}) \in \mathbf{Z}_+^{n-m-1+r} : \sum_{k \in N\setminus\{j\}} \bar{\mathbf{A}}_k x_k + \sum_{i=1}^{r} (\bar{\mathbf{A}}\mathbf{t}^i) y_i \le \bar{\mathbf{b}} \Big\}.$$

is a proper reformulation of $\mathscr{F}_N$.

Proof. The result follows quite easily by adapting the arguments from the proof of Lemma 3.6. Let $j \in N$. We need to show that the non-basic part of every feasible direction $\mathbf{t}$ w.r.t. $(\bar{\mathbf{b}}, \mathbf{0})$ is still expressable as a non-negative integer combination of the non-basic columns in the reformulated tableau associated with $\tilde{\mathscr{F}}_N$. From Lemma 3.2,

Table 4. The Integral Basis Method: a variant using discrete relaxations

input as in Table 3.
output as in Table 3.
$N_+ \leftarrow \{\, i \in N : \bar{c}_i > 0 \,\}$.
while $N_+ \neq \emptyset$ **do**
  Select $j \in N_+$.
  if $\mathbf{e}^j \in \mathscr{F}_N$ **then**
    return the vector $(-\bar{\mathbf{A}}_j, \mathbf{e}^j)$ as an augmenting direction w.r.t. $(\bar{\mathbf{b}}, \mathbf{0})$.
  else
    Determine a discrete relaxation $\mathscr{F}'_N$ of $\mathscr{F}_N$ such that $\mathbf{e}^j \notin \mathscr{F}'_N$.
    Compute the set $I(j) = \{\mathbf{t}^1, \ldots, \mathbf{t}^r\}$ of all irreducible solutions $\mathbf{z} \in \mathscr{F}'_N$ with $z_j > 0$.
    Delete the column with index j.
    Introduce r new columns corresponding to the elements in $I(j)$ as outlined in Table 3, i.e., the new column i has entries $\bar{\mathbf{A}}\mathbf{t}^i$ and reduced cost $\bar{\mathbf{c}}^\top \mathbf{t}^i$.
    Set $n \leftarrow n - 1 + r$ and update N and N_+.
return "Optimal."

it follows that a feasible direction $\mathbf{t}$ w.r.t. $(\bar{\mathbf{b}}, \mathbf{0})$ has the form $\mathbf{t} = (-\bar{\mathbf{A}}_N \mathbf{t}_N, \mathbf{t}_N)$ where $\mathbf{t}_N \in \mathscr{F}_N$. If $t_j = 0$, the statement is obviously correct. Let $t_j > 0$ and $\{\mathbf{t}^1, \ldots, \mathbf{t}^r\}$ be all the irreducible solutions of $\mathscr{F}'_N$ such that $t^i_j > 0$ for $i = 1, \ldots, r$.

We have that $\mathbf{t}_N \in \mathscr{F}_N \subseteq \mathscr{F}'_N$. Let H be all the irreducible solutions in $\mathscr{F}'_N$. Then we can write H as

$$H = \{\mathbf{h}^1, \ldots, \mathbf{h}^l\} \cup \{\mathbf{t}^1, \ldots, \mathbf{t}^r\},$$

where $h^i_j = 0$ for all $i = 1, \ldots, l$. This implies that there exist $\lambda_1, \ldots, \lambda_l \in \mathbf{Z}_+$ and $y_1, \ldots, y_r \in \mathbf{Z}_+$ such that

$$\mathbf{t}_N = \sum_{i=1}^{l} \lambda_i \mathbf{h}^i + \sum_{i=1}^{r} \mathbf{t}^i y_i.$$

Let $\mathbf{x}_N = \sum_{i=1}^{l} \lambda_i \mathbf{h}^i$. Then, $x_j = 0$. Hence, $(\mathbf{x}_{N\setminus\{j\}}, \mathbf{y}) \in \tilde{\mathscr{F}}_N$, and the conditions (3.5) hold. The proof of the converse direction is the same as in the proof of Lemma 3.6. □

Corollary 5.3. *The Integral Basis Method as outlined in Table 4 is correct.*

It remains to analyze whether the Integral Basis Method as shown in Table 4 is finite. This is however not always guaranteed as the following example shows.

Example 5.4 (Unbounded programs). Let us consider the following integer program, whose feasible set turns out to be unbounded.

$$\begin{array}{lrl} \max & x_1 & \\ \text{s.t.} & 2x_1 - x_2 + x_3 + s_1 &= 1 \\ & x_1 + x_2 - x_3 + s_2 &= 1 \\ & s_1, s_2, x_1, x_2, x_3 &\geq 0 \\ & s_1, s_2, x_1, x_2, x_3 &\in \mathbf{Z}. \end{array} \tag{5.1}$$

s_1 and s_2 are the two basic variables. x_1, x_2 and x_3 are non-basic. It is easy to see that

$$\mathscr{F}_N = \left\{ (0, x_2, x_3) \in \mathbf{Z}^3_+ : |x_2 - x_3| \leq 1. \right\}$$

We will apply the variant of the Integral Basis Method as shown in Table 4, using the discrete relaxation given by one constraint of the problem only. At the beginning of the algorithm, we have $N_+ = \{1\}$, i.e., the non-basic variable x_1 is the only variable promising an improvement. Since the vector $\mathbf{e}^1$ is not a feasible direction, the algorithm attempts to substitute the column corresponding to x_1. We will use the discrete relaxation $\mathscr{F}'_N$ given by the first constraint only,

$$\mathscr{F}'_N = \left\{ (x_1, x_2, x_3) \in \mathbf{Z}^3_+ : 2x_1 - x_2 + x_3 \leq 1 \right\}.$$

The vector $\mathbf{t}^1 = (1, 1, 0)^\top$ turns out to be the only irreducible solution $\mathbf{z} \in \mathscr{F}'_N$ with $z_1 > 0$. The algorithm therefore substitutes the non-basic column of variable x_1 by a new non-basic column, corresponding to the vector $\mathbf{t}^1$. We call the new variable x'_1. This leads to the tableau

$$\begin{array}{ll} \max & x'_1 \\ \text{s.t.} & x'_1 - x_2 + x_3 + s_1 \qquad = 1 \\ & 2x'_1 + x_2 - x_3 \qquad + s_2 = 1 \\ & x'_1, x_2, x_3, s_1, s_2 \geq 0 \\ & x'_1, x_2, x_3, s_1, s_2 \in \mathbf{Z}, \end{array} \tag{5.1$'$}$$

which is just the same as (5.1), only with permuted rows (slack variables) and columns. Thus, the procedure of substituting columns can be repeated ad infinitum, without making any progress.

Indeed, if we work with discrete relaxations, we can deal with bounded programs only. As before, let $\bar{\mathbf{A}} \in \mathbf{Q}^{m \times n-m}$, $\bar{\mathbf{b}} \in \mathbf{Z}^m_+$ and $\bar{\mathbf{c}} \in \mathbf{Q}^{n-m}$. Let B be the basis, N the set of non-basic variables and

$$\mathscr{F}_N = \left\{ \mathbf{z} \in \mathbf{Z}^{n-m}_+ : \bar{\mathbf{A}}_N \mathbf{z} \leq \bar{\mathbf{b}} \right\}.$$

If the underlying integer program is bounded, then there exists a number $U \in \mathbf{Z}_+$ such that $\sum_{i \in N} x_i \leq U$ for every $\mathbf{x}_N \in \mathscr{F}_N$.

We add this bound-constraint to our system. This way $\bar{\mathbf{A}}_N$ is enlarged by introducing a row with index $m + 1$ that consists of all ones. We also add an $(m + 1)$-st component to $\bar{\mathbf{b}}$ with entry U and introduce a slack variable associated with this row that is assigned to the basis. Then Algorithm 4 is applied. The modifications keep track on the norm in the initial space of the variables of the columns that arise in the course of the algorithm.

Lemma 5.5. *Let $U \in \mathbf{Z}_+$ such that $\sum_{i \in N} x_i \leq U$ for all $\mathbf{x}_N \in \mathscr{F}_N$. Then the Integral Basis Method as outlined in Table 4 is finite.*

Proof. In the course of the algorithm we remove a column with index $j \in N$ and add new columns corresponding to the vectors $\{\mathbf{t}^1, \ldots, \mathbf{t}^r\}$ consisting of all the irreducible solutions in the discrete relaxation $\mathscr{F}'_N$ of $\mathscr{F}_N$ such that $t^i_j > 0$ for $i = 1, \ldots, r$. This substitution step also has an effect on the $(m + 1)$-st row of $\bar{\mathbf{A}}$, namely the entries for these new columns in row $m+1$ become $\|\mathbf{t}^i\|_1$. If for one of these entries $\|\mathbf{t}^i\|_1 > U$, then the column $\mathbf{t}^i$ can be removed. Since in each reformulation step we have that $\mathbf{e}^j \notin \mathscr{F}'_N$, the vectors $\{\mathbf{t}^1, \ldots, \mathbf{t}^r\}$ that become new columns will consist of non-negative integer combinations of the original columns involving more than the column with index j. Hence $\|\mathbf{t}^i\|_1 > \alpha$ for all $i = 1, \dots, r$, where α denotes the entry in row $m + 1$ of the old column j to be removed. Therefore, each reformulation step causes the entries in row $m+1$ to increase. After a finite number of steps the newly generated columns will touch the norm bound U. Thus the procedure is finite. □

6. Irreducible Solutions from Knapsack Relaxations

So far we have established the theory of an augmentation algorithm that performs column substitutions. In order to make such a method computationally tractable, it is important to understand the trade-off between the "strength" of column substitutions on the one hand, and the number of generated columns and the time to compute these objects on the other hand.

As a general rule one can say that the "closer" the discrete relaxation is to the feasible region, the more time is needed to compute the replacement columns. Our philosophy in this section is to begin with a very weak discrete relaxation that comes from generating a Gomory cut from a single row of the tableau (section 6.1). The advantage is that for this special relaxations we have an explicit formula for the irreducible solutions. Moreover, the number of columns added in a single substitution step is low. We demonstrate on an example that this technique can require many iterations to terminate, however.

On the other extreme, one can study the irreducible solutions to one row of the system (section 6.2). This results in a very strong substitution method at the expense of generating many new columns. Therefore the applicability of this approach is limited to rows whose coefficients are not too big in absolute value.

A compromise between these two extreme strategies is investigated in section 6.3. There we study a family of relaxations that is parameterized by an integral "shift" value, controlling the strength of the relaxation. In addition, we do not only use the information from one row of the tableau but also take all rows with non-negative coefficients into account when computing the substitution columns. This strategy is precisely the one used in our implementation; see section 7.

6.1. Irreducible Solutions Corresponding to Gomory Cuts

Theorem 6.1. *Let $j \in N$ such that $\mathbf{e}^j \notin \mathscr{F}_N$. Let $\mathbf{a}^\top \mathbf{x}_N \leq b$ denote a row of tableau* (3.1) *violated by $\mathbf{e}^j$, i.e., $b < a_j$. Let $N_<$ denote the index set of non-basic variables with negative coefficients in $\mathbf{a}$. For $k \in N_<$ let $\mu_k := \lceil -a_k/a_j \rceil$. Then the tableau obtained from* (3.1) *by removing column $\bar{\mathbf{A}}\mathbf{e}^j$ and adding the columns*

$$\bar{\mathbf{A}}(\mathbf{e}^k + 1\mathbf{e}^j), \dots, \bar{\mathbf{A}}(\mathbf{e}^k + \mu_k \mathbf{e}^j) \quad \text{for } k \in N_< \tag{6.1}$$

is a proper reformulation of tableau (3.1).

Proof. For $j \in N$ such that $\mathbf{e}^j \notin \mathscr{F}_N$, let $\mathbf{a}^\top \mathbf{x}_N \leq b$ denote a row of tableau (3.1) such that $b < a_j$. The constraint reads

$$\sum_{k \in N \setminus N_<} a_k z_k - \sum_{k \in N_<} (-a_k) z_k \leq b.$$

From this constraint, we obtain the Gomory cut

$$\sum_{k \in N \setminus N_<} \left\lfloor \frac{a_k}{a_j} \right\rfloor z_k - \sum_{k \in N_<} \left\lceil \frac{-a_k}{a_j} \right\rceil z_k \leq 0 = \left\lfloor \frac{b}{a_j} \right\rfloor. \tag{6.2}$$

We will consider the discrete relaxation $\mathscr{F}'_N$ of $\mathscr{F}_N$, given by one constraint dominated by (6.2),

$$\mathscr{F}'_N = \left\{ \mathbf{z} \in \mathbf{Z}_+^{n-m} : \ z_j - \sum_{k \in N_<} \left\lceil \frac{-a_k}{a_j} \right\rceil z_k \leq 0 \right\}.$$

The set

$$\{ \mathbf{e}^k + 1\mathbf{e}^j, \dots, \mathbf{e}^k + \mu_k \mathbf{e}^j : k \in N_< \}$$

corresponds to all the irreducible solutions $\mathbf{z} \in \mathscr{F}'_N$ such that $z_j > 0$. On account of Lemma 5.2 the statement follows. ◻

Table 5 shows a variant of the Integral Basis Method that uses the construction of Theorem 3.1 in every step. By Lemma 5.2 and Theorem 5.5, this algorithm computes an augmenting direction or proves optimality of the current solution in a finite number of steps if the program is bounded.

Table 5. The Integral Basis Method: a variant using knapsack relaxations

input as in Table 3.
output as in Table 3.
$N_+ \leftarrow \{ k \in N : \bar{c}_k > 0 \}$.
while $N_+ \neq \emptyset$ **do**
  Select $j \in N_+$.
  if $\mathbf{e}^j \in \mathscr{F}_N$ **then**
    return the vector $(-\bar{\mathbf{A}}_j, \mathbf{e}^j)$ as an augmenting direction w.r.t. $(\bar{\mathbf{b}}, \mathbf{0})$.
  else
    Select a row i with $\bar{A}_{ij} > \bar{b}_i$.
    $I(j) \leftarrow \emptyset$.
    for all $k \in N$ **do**
      if $\bar{A}_{ik} < 0$ **then**
        Let $\mu_k \leftarrow \lceil -\bar{A}_{ik} / \bar{A}_{ij} \rceil$.
        $I(j) \leftarrow I(j) \cup \{ \mathbf{e}^k + 1\mathbf{e}^j, \dots, \mathbf{e}^k + \mu_k \mathbf{e}^j \}$.
    Delete the column with index j.
    Introduce $|I(j)|$ new columns corresponding to the elements in $I(j)$ as outlined in Table 3.
    Set $n \leftarrow n - 1 + |I(j)|$ and update N and N_+.
return “Optimal.”

Example 6.2. Let us consider the linear integer program

$$\begin{array}{llr}
\max & x_1 + x_2 - x_3 & \\
\text{s.t.} & 9x_1 + 5x_2 - 3x_3 - 4x_4 + s_1 = 0 & \\
& -x_1 - 2x_3 + x_4 + s_2 = 0 & \\
& 4x_3 - x_4 + s_3 = 1 & (6.3) \\
& -3x_3 + x_4 + s_4 = 1 & \\
& x_1 + x_2 - 5x_3 + x_4 + s_5 = 1 & \\
& x_1, \dots, x_4; s_1, \dots, s_5 \in \mathbf{Z}_+, &
\end{array}$$

given as the tableau corresponding to the basic feasible solution $(0, 0, 0, 0; 0, 0, 1, 1, 1)$. Table 6 shows the first few possible steps of the procedure. We note that quite a few more iterations are needed until an augmentation vector is constructed that leads to a solution corresponding to the unique optimal solution $(3, 1, 2, 7; 2, 0, 0, 0, 0)$ to the program (6.3).

6.2. Irreducible Solutions From Primitive Partition Identities

The construction above is applicable to every non-optimal tableau, is proven to terminate for bounded programs and is computationally very cheap. On the other hand, the observation made in Example 6.2 suggests that the construction of Theorem 6.1 may be rather weak.

We propose to integrate stronger, "combinatorial" substitution steps into the procedure of Table 5. The idea is to perform special substitutions (leading to better reformulations of the problem) whenever those substitutions are applicable. If no such step is possible, we can always resort to the general-purpose construction of Theorem 6.1.

We present one family of combinatorial substitution steps in the following: We consider one row of the tableau that has "small" integral coefficients and compute all the irreducible solutions to the discrete relaxation given by this row. This can be accomplished via the connection to primitive partition identities (PPI).

Definition 6.3 (Primitive partition identities). *An equation of the form*

$$p_1 + \dots + p_k = q_1 + \dots + q_l \tag{6.4}$$

with numbers $p_1, \dots, p_k; q_1, \dots, q_l \in \mathbf{N}$ is called a partition identity. *The numbers $p_1, \dots, p_k; q_1, \dots, q_l$ are called the* parts *of the identity, and their maximum is called the* largest part. *The identity* (6.4) *is called* primitive *if no proper partial sum on the left equals one on the right.*

We shall always assume that $p_i \neq q_j$ for $i \in \{1, \dots, k\}$ and $j \in \{1, \dots, l\}$, i.e., (6.4) does not contain trivial sub-identities of the form $a = a$. In the following, we will represent the partition identities with largest part at most n by integral vectors $\mathbf{x} \in \mathbf{Z}^n$ via the setting

$$x_i := \left|\{\, j \in \{1, \dots, k\} : p_j = i \,\}\right| - \left|\{\, j \in \{1, \dots, l\} : q_j = i \,\}\right| \tag{6.5}$$

for $i \in \{1, \dots, n\}$. We note that these vectors satisfy the condition $\mathbf{n}^\top \mathbf{x} = 0$, where $\mathbf{n} = (1, \dots, n)^\top$. Next we introduce some terms and notations that are useful for dealing with the vectors arising from partition identities.

Table 6. The first steps of the substitution procedure of Table 5 on Example 6.2. The current basis is shown in boldface. A dagger (†) marks the element corresponding to the column and row selected in steps 5 and 9 of the algorithm. Variables that have been substituted by new variables are marked with an asterisk (⋆) in the reduced-cost row.

x_1	x_2	x_3	x_4	s_1	s_2	s_3	s_4	s_5	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	
(a) Initial tableau																	
1	1	−1	0	0	0	0	0	0									0
9†	5	−3	−4	**1**													0
−1		−2	1		**1**												0
		4	−1			**1**											1
		−3	1				**1**										1
1	1	−5	1					**1**									1
(b) Tableau after substituting x_1 according to the first row																	
⋆	1	−1	0	0	0	0	0	0	0	1							0
	5	−3	−4	**1**					6	5							0
		−2	1		**1**				−3								0
		4	−1			**1**			4	−1							1
		−3	1				**1**		−3	1							1
	1	−5	1					**1**	−4	2†							1
(c) Tableau after substituting w_2 according to the fifth row																	
⋆	1	−1	0	0	0	0	0	0	0	⋆	0	−1	−2	1	1		0
	5	−3	−4	**1**					6		2	−1	−4	11	17		0
		−2	1		**1**				−3		−2	−4	−6	−3	−6		0
		4	−1			**1**			4		3	7	11	3	7†		1
		−3	1				**1**		−3		−2	−5	−8	−2	−5		1
	1	−5	1					**1**	−4		−3	−8	−13	−2	−6		1
(d) Tableau after substituting w_7 according to the third row																	
⋆	1	−1	0	0	0	0	0	0	0	⋆	0	−1	−2	1	⋆	1	0
	5	−3	−4	**1**					6		2	−1	−4	11		13	0
		−2	1		**1**				−3		−2	−4	−6	−3		−5	0
		4	−1			**1**			4		3	7	11	3		6	1
		−3	1				**1**		−3		−2	−5	−8	−2		−4	1
	1	−5	1					**1**	−4		−3	−8	−13	−2		−5	1

Definition 6.4. (a) *For two numbers a, b, we write $a \preceq b$ if a and b have the same sign and $|a| \leq |b|$ holds. For vectors* $\mathbf{v}$, $\mathbf{w} \in \mathbf{Z}^n$, $\mathbf{n}^\top \mathbf{v} = \mathbf{n}^\top \mathbf{w} = 0$, *we write* $\mathbf{v} \preceq \mathbf{w}$ *if* $\mathbf{v}$ reduces $\mathbf{w}$, *i.e.,* $v_i \preceq w_i$ *for* $i \in \{1, \dots, n\}$.

(b) *A vector* $\mathbf{w} \in \mathbf{Z}^n$ *with* $\mathbf{n}^\top \mathbf{w} = 0$ *is* irreducible *w.r.t.* $\mathbf{n}$ *if* $\mathbf{w} \neq \mathbf{0}$ *and there is no proper reducer* $\mathbf{v} \preceq \mathbf{w}$, *where* $\mathbf{n}^\top \mathbf{v} = 0$ *and* $\mathbf{0} \neq \mathbf{v} \neq \mathbf{w}$.

It is easy to see that a partition identity is primitive if and only if its associated vector $\mathbf{x}$ is irreducible w.r.t. $\mathbf{n}$. In the following, let P_n denote the set of irreducible vectors $\mathbf{x} \in \mathbf{Z}^n \cap \ker \mathbf{n}^\top$ with $x_{\max \operatorname{supp} \mathbf{x}} > 0$, i.e., the largest part of the associated identity shall appear on the left side. Furthermore, for $t \in \mathbf{Z}_+$ let P_n^t denote the set of vectors $\mathbf{x} \in P_n$ with $x_n = t$.

Remark 6.5. The terminology used here comes from the study of *Graver test sets* for integer programming problems; see [AWW00] or [Wei98] for an overview. In fact, the set $\pm P_n = \{\, \mathbf{x}, -\mathbf{x} : \mathbf{x} \in P_n \,\}$ is the Graver test set for the family of knapsack master problems

$$\max\{\, \mathbf{c}^\top \mathbf{x} : \mathbf{n}^\top \mathbf{x} = b,\ \mathbf{l} \le \mathbf{x} \le \mathbf{u},\ \mathbf{x} \in \mathbf{Z}^n \,\} \tag{6.6}$$

with varying data $\mathbf{c} \in \mathbf{R}^n$, $\mathbf{l}, \mathbf{u} \in \mathbf{Z}^n$, and $b \in \mathbf{Z}$.

The following estimate was established by Lambert [Lam87] and independently by Diaconis–Graham–Sturmfels [DGS96]; note that stronger bounds were given by Henk–Weismantel [HW00].

Lemma 6.6. *For* $\mathbf{x} \in P_n$*,*

$$\|\mathbf{x}\|_1 \le \max \operatorname{supp} \mathbf{x}^+ + \max \operatorname{supp} \mathbf{x}^- \le 2n - 1.$$

As an immediate consequence we obtain:

Corollary 6.7. *For* $\mathbf{x} \in P_n$*, we have* $x_n \le n - 1$.

It is possible to partially characterize the irreducible vectors; we have obtained the following two results.

Lemma 6.8. *Let* $\mathbf{v} \in P^t_{n+1}$ *and* $\mathbf{w} := \mathbf{v} + \mathbf{e}^{n+1} - \mathbf{e}^i - \mathbf{e}^{(n+1)-i}$ *for some index* $i \in \{1, \dots, n\}$*.*

(a) *If* v_i*,* $v_{(n+1)-i} > 0$*, then* $\mathbf{w} \in P^{t+1}_{n+1}$*.*
(b) *If* v_i*,* $v_{(n+1)-i} \le 0$*, then* $\mathbf{w} \notin P^{t+1}_{n+1}$*.*

Proof. Let v_i, $v_{(n+1)-i} > 0$. Suppose $\mathbf{w} = \mathbf{w}^1 + \mathbf{w}^2$ with vectors $\mathbf{w}^1$, $\mathbf{w}^2$ such that $\mathbf{0} \ne \mathbf{w}^1, \mathbf{w}^2 \le \mathbf{w}$. Without loss of generality, we may assume that $w^1_{n+1} < w_{n+1}$. Then $w^1_{n+1} \le v_{n+1}$ and $w^1_i < v_i$, thus $\mathbf{w}^1 \le \mathbf{v}$ and $\mathbf{w}^1 \ne \mathbf{v}$. This contradicts the irreduciblity of $\mathbf{v}$.

Now let v_i, $v_{(n+1)-i} \le 0$. Then $\mathbf{0} \ne \mathbf{v} \ne \mathbf{w}$ and $\mathbf{v} \le \mathbf{w}$. Hence, $\mathbf{w}$ is reducible. □

Theorem 6.9. *Let* $\mathbf{v} \in P^t_{n+1}$ *and* $i \in \{1, \dots, n\}$ *such that* $v_i \le 0$ *and* $v_{(n+1)-i} > 0$*. Let* $\mathbf{w} := \mathbf{v} + \mathbf{e}^{n+1} - \mathbf{e}^i - \mathbf{e}^{(n+1)-i}$*. Then either* $\mathbf{w}$ *is irreducible, i.e.,* $\mathbf{w} \in P^{t+1}_{n+1}$*, or* $\mathbf{w}$ *decomposes into two irreducible vectors* $\mathbf{w}^1$*,* $\mathbf{w}^2$ *with*

$$w^1_i = w_i, \qquad w^2_i = 0, \tag{6.7a}$$

$$w^1_{n+1} = 0, \qquad w^2_{n+1} = w_{n+1}, \tag{6.7b}$$

where $\mathbf{w}^2$ *is obtained from a vector* $\mathbf{u} \in P^t_{n+1}$ *by the construction of Lemma 6.8 a.*

Proof. Let $\mathbf{w} = \mathbf{w}^1 + \dots + \mathbf{w}^l$ for some $l \in \mathbf{N}$ and irreducible vectors $\mathbf{w}^1, \dots, \mathbf{w}^l \le \mathbf{w}$. Since

$$0 \le w^k_{n+1-i} \le w_{n+1-i} < v_{n+1-i}, \tag{6.8}$$

we have $\mathbf{v} \ne \mathbf{w}^k$ for $k \in \{1, \dots, l\}$. Therefore, irreducibility of $\mathbf{w}^k$ implies $\mathbf{v} \not\le \mathbf{w}^k$. Thus for each $k \in \{1, \dots, l\}$ there exists an index $j(k) \in \{1, \dots, n+1\}$ such that $v^k_{j(k)} \not\le w^k_{j(k)}$. Since $w^k_j \le w_j \le v_j$ for $j \notin \{i,\ n+1\}$, we have that $j(k) \in \{i,\ n+1\}$ for $k \in \{1, \dots, l\}$. But $w_i = v_i - 1$ and $w_{n+1} = v_{n+1} + 1$, hence $l \le 2$. In the case $l = 1$, $\mathbf{w}$ is irreducible. If $l = 2$, the vectors $\mathbf{w}^1$ and $\mathbf{w}^2$ satisfy the conditions (6.7) up to transposition.

Now let $\mathbf{u} := \mathbf{w}^2 - \mathbf{e}^{n+1} + \mathbf{e}^i + \mathbf{e}^{(n+1)-i}$. Suppose that $\mathbf{u}$ is reducible. Since $u_i = 1$, there is a reducer $\mathbf{u}^2 \leq \mathbf{u}$ such that $u_i^2 = 0$ and $\mathbf{u}^2 \neq \mathbf{0}$. But $\mathbf{u}^2$ also reduces $\mathbf{v}$, which is a contradiction to the irreducibility of $\mathbf{v}$. Hence, $\mathbf{u}$ is irreducible. By (6.8) and (6.7a), we have $w_{n+1-i}^2 \geq 0$ and $w_i^2 = 0$; thus $\mathbf{w}^2$ is obtained from $\mathbf{u}$ by the construction of Lemma 6.8 a. □

The characterization of the vectors in P_{n+1}^{t+1} in terms of those from P_{n+1}^{t} suggests the use of a recursive computation scheme. By Corollary 6.7, the maximum value of t to consider is known in advance. Exploiting the conditions of Lemma 6.8 and Theorem 6.9 greatly reduces the number of expensive reducibility checks. Table 7 shows the resulting algorithm. The remaining reduction check in line 12 can be performed by an orthogonal

Table 7. Computation of the primitive partition identities

1: **input** P_n;
2: **output** P_{n+1};
3: $P_{n+1}^0 \leftarrow \{\mathbf{0}\} \cup \pm P_n$;
4: **for** $t \leftarrow 0$ **to** $n-2$ **do**
5: $\quad P_{n+1}^{t+1} \leftarrow \emptyset$;
6: $\quad$ **for all** $\mathbf{v} \in P_{n+1}^t$ **do**
7: $\quad\quad$ **for** $i \leftarrow 1$ **to** $\lfloor \frac{n}{2} \rfloor$ **do**
8: $\quad\quad\quad \mathbf{w} \leftarrow \mathbf{v} + \mathbf{e}_{n+1} - \mathbf{e}_i - \mathbf{e}_{(n+1)-i}$;
9: $\quad\quad\quad$ **if** $v_i > 0$ and $v_{(n+1)-i} > 0$ **then**
10: $\quad\quad\quad\quad P_{n+1}^{t+1} \leftarrow P_{n+1}^{t+1} \cup \{\mathbf{w}\}$
11: $\quad\quad\quad$ **else if** $v_i > 0$ or $v_{(n+1)-i} > 0$ **then**
12: $\quad\quad\quad\quad$ **if** $\nexists\, \mathbf{u} \in \pm P_n$ with $\mathbf{u} \leq \mathbf{w}$ **then**
13: $\quad\quad\quad\quad\quad P_{n+1}^{t+1} \leftarrow P_{n+1}^{t+1} \cup \{\mathbf{w}\}$;
14: **return** $P_{n+1} \leftarrow \bigcup_{t=0}^{n} P_{n+1}^t$;

range searching procedure in a static data structure for P_n; see the thesis [Köp99] for a discussion and implementation notes.

Table 8 shows the cardinality of P_k for $k \leq 27$. In the following, we will assume that the set P_k has been computed for some value k. We will now show how one can use this pre-computed set to determine all the irreducible solutions to a single inequality with small integral coefficients. For the sake of simplicity, we restrict ourselves to the case of pairwise distinct coefficients and a zero right-hand side:

Lemma 6.10. *Let k be a positive integer. Consider the discrete relaxation*

$$\begin{gathered}\sum_{j=1}^{n} (\sigma_j a_j) x_j \leq 0 \\ \mathbf{x} \in \mathbf{Z}_+^n,\end{gathered} \tag{6.9}$$

where $a_j \in \{2, \dots, k\}$ and $\sigma_j \in \{\pm 1\}$ for all $j \in \{1, \dots, n\}$ and $a_j \neq a_{j'}$ for $j \neq j'$. Let $a_0 = \sigma_0 = 1$. Then the irreducible solutions to (6.9) *are the vectors $\mathbf{x}$ defined by*

$$x_j = |v_{a_j}| \quad \text{for } j \in \{1, \dots, n\}, \tag{6.10}$$

where $\mathbf{v} \in \{\, \mathbf{v} \in P_k : \sigma_j v_{a_j} \geq 0$ for $j \in \{0, \dots, n\}$ and $v_i = 0$ for $i \notin \{a_0, \dots, a_n\} \,\}$.

Table 8. Cardinalities of PPI sets

k	$\|P_k\|$	k	$\|P_k\|$
2	1	15	58 171
3	5	16	99 328
4	15	17	181 514
5	47	18	287 239
6	102	19	502 116
7	276	20	775 710
8	578	21	1 239 710
9	1 261	22	1 956 334
10	2 465	23	3 210 736
11	5 362	24	4 660 786
12	9 285	25	7 297 823
13	18 900	26	10 997 235
14	33 269	27	16 536 803

A proof and a generalization of this lemma can be found in [Köp99].

Example 6.11. Let us consider the first row of the tableau (6.3),

$$9x_1 + 5x_2 - 3x_3 - 4x_4 + s_1 = 0, \tag{6.11}$$

where s_1 is the (basic) slack variable. The largest coefficient in (6.11) is 9, hence we can obtain the irreducible solutions from the PPI table P_9 via Lemma 6.10; see Table 9. We can now substitute x_1 and x_2 simultaneously by new non-basic variables $y_1, \ldots, y_{14}$,

Table 9. Irreducible solutions to the equation $9x_1 + 5x_2 - 3x_3 - 4x_4 + s_1 = 0$ where $x_1, \ldots, x_4, s_1 \in \mathbf{Z}_+$. Zero entries have been left blank. The solutions with $x_1 > 0$ or $x_2 > 0$ give rise to new non-basic variables y_i, which are denoted aside the respective solutions.

	x_1	x_2	x_3	x_4	s_1		x_1	x_2	x_3	x_4	s_1
			1		3	y_7	2		2	3	
				1	4	y_8	2		1	4	1
y_1		2	1	2	1	y_9	2			5	2
y_2		2		3	2	y_{10}	2	2		7	
y_3		2	2	1		y_{11}	1	3		6	
y_4	1	1	1	3	1	y_{12}		4		5	
y_5	1	1		4	2	y_{13}	3	1		8	
y_6	1	1	2	2		y_{14}	4			9	

corresponding to the irreducible solutions to (6.11) with $x_1 > 0$ or $x_2 > 0$; cf. Remark 3.4. This leads to the tableau shown in Table 10 a. By this substitution, we have exploited all the combinatorial information from the first row. In fact, the first row is redundant in the new formulation: all non-basic coefficients have become non-positive.

Now we can perform two non-degenerate integral pivot steps; see Table 10 b, c. In this example, the resulting tableau happens to provide a linear-programming certificate for the optimality of the basic integer solution with objective value 2, given by $y_4 = 1$, $y_8 = 1$, and $s_1 = 2$, all other variables being zero.

The optimal solution to the original program can be computed via Table 9: Since y_4 corresponds to $(x_1, x_2, x_3, x_4) = (1, 1, 1, 3)$ and y_8 to $(2, 0, 1, 4)$, the optimal solution to the original program is $(3, 1, 2, 7; 2, 0, 0, 0, 0)$.

Table 10. Simultaneous substitution of two variables using primitive partition identities, followed by two non-degenerate integral simplex steps leading to the optimal solution. The current basis is shown in boldface, whereas the pivot element is marked with an asterisk ($\star$). The negative of the objective function value is shown in the upper right corner of each tableau.

x_3	x_4	s_1	s_2	s_3	s_4	s_5	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	
(a) Tableau obtained from Tableau (6.3) by performing a PPI substitution																					
−1	0	0	0	0	0	0	1	2	0	1	2	0	0	1	2	4	4	4	4	4	0
−3	−4	**1**					−1	−2		−1	−2			−1	−2						0
−2	1		**1**					3	−3		3	−3	−3		3	5	5	5	5	5	0
4	−1			**1**			2	−3	7	$1^\star$	−4	6	5		−5	−7	−6	−5	−8	−9	1
−3	1				**1**		−1	3	−5		4	−4	−3	1	5	7	6	5	8	9	1
−5	1					**1**	−1	5	−7		6	−6	−5	1	7	11	10	9	12	13	1
(b) Tableau after pivoting y_4 into the basis																					
−5	1	0	0	−1	0	0	−1	5	−7	0	6	−6	−5	1	7	11	10	9	12	13	−1
	−5	**1**		1			1	−5	7		−6	6	5	−1	−7	−7	−6	−5	−8	−9	1
−2	1		**1**					3	−3		3	−3	−3		3	5	5	5	5	5	0
4	−1			1			2	−3	7	**1**	−4	6	5		−5	−7	−6	−5	−8	−9	1
−3	1				**1**		−1	3	−5		4	−4	−3	1	5	7	6	5	8	9	1
−5	1					**1**	−1	5	−7		6	−6	−5	$1^\star$	7	11	10	9	12	13	1
(c) Tableau after pivoting y_8 into the basis																					
0	0	0	0	−1	0	−1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	−2
−4	−4	**1**		1		1										4	4	4	4	4	2
−2	1		**1**					3	−3		3	−3	−3		3	5	5	5	5	5	0
4	−1			1			2	−3	7	**1**	−4	6	5		−5	−7	−6	−5	−8	−9	1
2					**1**	−1		−2	2		−2	2	2		−2	−4	−4	−4	−4	−4	0
−5	1					1	−1	5	−7		6	−6	−5	**1**	7	11	10	9	12	13	1

6.3. *Irreducible Solutions to Strengthened 0/1 Knapsack Relaxations*

For an efficient implementation of the Integral Basis Method it is important to recognize the rôle of *generalized upper-bound* (GUB) constraints, i.e., valid inequalities $\mathbf{d}^\top \mathbf{x} \le \delta$ with $\mathbf{d} \in \mathbf{Z}_+^{n+m}$ and $\delta \in \mathbf{Z}_+$. In every step of the Integral Basis Method, the newly introduced non-basic variables are automatically coupled via all GUB constraints that their constituting variables (i.e., the non-basic variables with indices in the support of $\mathbf{t}$) were members of. In particular, the replacement procedure builds GUB constraints from simple upper-bound constraints.

It is trivial to realize that the set $I(j) = \{\mathbf{t}^1, \ldots, \mathbf{t}^r\}$ of all irreducible solutions $\mathbf{z} \in \mathscr{F}'_N$ with $z_j > 0$ can be "truncated" by taking GUB constraints into account. In fact, every vector $\mathbf{t} \in I(j)$ that violates such a GUB constraint, i.e., $\mathbf{d}_N^\top \mathbf{t} > \delta$, can be dropped from $I(j)$ because in the reformulated problem the new non-basic variable corresponding to $\mathbf{t}$ would attain a value of zero for every feasible integer solution.

In the following, we describe a practical procedure that computes all the irreducible solutions to a discrete relaxation consisting of one knapsack constraint and a set of GUB constraints. For the sake of simplicity, we restrict ourselves to the case of 0/1 variables.

To this end, we consider the i-th row of the tableau (3.1) and read it as a linear inequality for the non-basic variables,

$$\bar{\mathbf{A}}_{i,\cdot}\mathbf{x}_N \leq \bar{b}_i. \tag{6.12}$$

Let $j \in N$ with $\bar{A}_{ij} > \bar{b}_i$. Let $r \in \mathbf{Z}_+$ denote the number of negative entries in the i-th row of $\bar{\mathbf{A}}$. Now sort the indices $k_1, \dots, k_r \in N$ of the columns where $\bar{\mathbf{A}}_{i,\cdot}$ has negative entries so that $\bar{A}_{i,k_\sigma} \leq \bar{A}_{i,k_\tau} < 0$ for $\sigma < \tau$ and $\bar{A}_{i,k} \geq 0$ for $k \in N \setminus \{k_1, \dots, k_r\}$. Then the inequality

$$\bar{A}_{ij}x_j + \sum_{\sigma=1}^{r} \bar{A}_{i,k_\sigma} x_{k_\sigma} \leq \bar{b}_i + \mathit{shift} \tag{6.13}$$

is valid for $\mathscr{F}_N$ for every $\mathit{shift} \geq 0$. The discrete relaxation $\mathscr{F}'_N$ of $\mathscr{F}_N$ to be considered is the set given by the inequality (6.13) and some GUB constraints $\mathbf{d}_N^\top \mathbf{x}_N \leq \delta$ represented by a set $\mathscr{D}$ of pairs $(\mathbf{d}_N, \delta) \in \mathbf{Z}_+^n \times \mathbf{Z}_+$:

$$\begin{aligned}
\bar{A}_{ij}x_j + \textstyle\sum_{\sigma=1}^{r} \bar{A}_{i,k_\sigma} x_{k_\sigma} &\leq \bar{b}_i + \mathit{shift}, && && (6.14a)\\
\mathbf{d}_N^\top \mathbf{x}_N &\leq \delta && \text{for all } (\mathbf{d}, \delta) \in \mathscr{D}, && (6.14b)\\
\mathbf{x}_N &\in \mathbf{Z}_+^n, && && (6.14c)
\end{aligned}$$

where the non-negative *shift* parameter can be used to control the strength of the relaxation.

Remark 6.12. For $\mathit{shift} = \bar{A}_{ij} - \bar{b}_i - 1$, we have

$$I(j) = \{\, \mathbf{e}^{k_i} + \mathbf{e}^j : i \in \{1, \dots, r\} \,\} \cap \mathscr{F}'_N. \tag{6.15}$$

In other words, the set $I(j)$ of irreducible solutions $\mathbf{t} \in \mathscr{F}'_N$ with $t_j > 0$ coincides with that of Theorem 6.1, truncated by the implicit variable bound $x_j \leq 1$ and by the GUB constraints.

The algorithm shown in Table 11 computes the set $I(j)$ of irreducible solutions $\mathbf{t} \in \mathscr{F}'_N$ with $t_j > 0$, for arbitrary *shift* values. The algorithm starts off with the vector $\mathbf{e}^j$ and systematically builds partial sums

$$\mathbf{s} = \mathbf{e}^j + \mathbf{e}^{k_{\sigma_1}} + \mathbf{e}^{k_{\sigma_2}} + \cdots + \mathbf{e}^{k_{\sigma_p}} \quad \text{with} \quad \sigma_1 < \cdots < \sigma_p \tag{6.16}$$

by adding unit vectors corresponding to the variables with negative coefficients in inequality (6.14a), starting with the most negative coefficients. This procedure continues until the resulting vector $\mathbf{s}$ satisfies the knapsack constraint (6.14a) or violates one of the GUB or compatibility constraints.

Theorem 6.13. *The algorithm shown in Table 11 is correct.*

Table 11. Algorithm that computes irreducible vectors for strengthened 0/1 knapsack relaxations

input constraint (6.14a); the *shift* value; a set $\mathscr{D}$ of pairs $(\mathbf{d}_N, \delta) \in \mathbf{Z}^n_+ \times \mathbf{Z}_+$ representing GUB constraints $\mathbf{d}_N^\top \mathbf{x}_N \leq \delta$.
output the set $I(j)$ of all irreducible solutions $\mathbf{t}$ to the system (6.14) with $t_j > 0$.
Initialize $I(j) \leftarrow \emptyset$.
(∗ *We maintain a set S of pairs $(\sigma, \mathbf{s})$, where $\mathbf{s}$ is some partial sum* (6.16) *that does not yet satisfy the knapsack constraint* (6.14a)*, and σ is the index σ_p in* (6.16) ∗)
$S \leftarrow \{(0, \mathbf{e}^j)\}$.
while $S \neq \emptyset$ **do**
  Let $(\sigma, \mathbf{s}) \in S$ and set $S \leftarrow S \setminus \{(\sigma, \mathbf{s})\}$.
  for $\hat\sigma \leftarrow \sigma + 1$ **to** r **do**
    Let $k \leftarrow k_{\hat\sigma}$ and $\hat{\mathbf{s}} \leftarrow \mathbf{s} + \mathbf{e}^k$.
    if $(k, l) \in E$ for all $l \in \operatorname{supp} \mathbf{s}$ **then**
      if $\mathbf{d}_N^\top \hat{\mathbf{s}} \leq \delta$ for all $(\mathbf{d}, \delta) \in \mathscr{D}$ **then**
        if $\mathbf{a}^\top \hat{\mathbf{s}} > \beta + \mathit{shift}$ **then**
          (∗ $\hat{\mathbf{s}}$ *does not satisfy the knapsack constraint* (6.14a) ∗)
          $S \leftarrow S \cup \{(\hat\sigma, \hat{\mathbf{s}})\}$.
        else
          $I(j) \leftarrow I(j) \cup \{\hat{\mathbf{s}}\}$.
return $I(j)$.

Proof. Let I denote the set computed by the algorithm of Table 11. First we show that all elements of I are irreducible solutions to the system (6.14) with $t_j > 0$, i.e., $I \subseteq I(j)$.

To this end, let $\mathbf{t} \in I$. By construction, we have $t_j = 1$. Clearly, $\mathbf{t} \in \mathscr{F}'_N$ because all the constraints in the system (6.14) are checked in the algorithm. Let $\sigma_1 < \cdots < \sigma_p$ denote the values of $\hat\sigma$ in the computation chain of $\mathbf{t}$, i.e., $\mathbf{t} = \mathbf{e}^j + \sum_{\alpha=1}^p \mathbf{e}^{k_{\sigma_\alpha}}$.

Now suppose that $\mathbf{t}$ is reducible w.r.t. $\mathscr{F}'_N$, i.e., there exist $k \geq 2$ solutions $\mathbf{t}^1, \dots, \mathbf{t}^k \in \mathscr{F}'_N \setminus \{\mathbf{0}\}$ such that $\mathbf{t} = \sum_{i=1}^k \mathbf{t}^i$. W.l.o.g. let $t^1_j = 1$ and $t^2_j = \cdots = t^k_j = 0$. Let $1 \leq q < p$ and $\tau_1, \dots, \tau_q \in \{\sigma_1, \dots, \sigma_p\}$, ordered such that $\tau_1 < \cdots < \tau_q$, and $\mathbf{t}^1 = \mathbf{e}^j + \sum_{\alpha=1}^q \mathbf{e}^{k_{\tau_\alpha}}$. Let $\tilde{\mathbf{t}} = \mathbf{e}^j + \sum_{\alpha=1}^q \mathbf{e}^{k_{\sigma_\alpha}}$. We claim that $\tilde{\mathbf{t}} \in \mathscr{F}'_N$. Since $\tilde{\mathbf{t}} \leq \mathbf{t}$, all the constraints (6.14b) are satisfied by $\tilde{\mathbf{t}}$. Since $\tau_\alpha \geq \sigma_\alpha$ for $\alpha \in \{1, \dots, q\}$, we have $\bar A_{i,k_{\tau_\alpha}} \geq \bar A_{i,k_{\sigma_\alpha}}$ and hence

$$\bar{\mathbf{A}}_{i,\cdot} \tilde{\mathbf{t}} = \bar A_{ij} + \sum_{\alpha=1}^{q} \bar A_{i,k_{\sigma_\alpha}} \leq \bar A_{ij} + \sum_{\alpha=1}^{q} \bar A_{i,k_{\tau_\alpha}} = \bar{\mathbf{A}}_{i,\cdot} \mathbf{t}^1 \leq \bar b_i + \mathit{shift}.$$

On the other hand, $\tilde{\mathbf{t}}$ appeared in the computation chain of $\mathbf{t}$, which is a contradiction. This proves that $\mathbf{t} \in I(j)$.

Conversely, let $\mathbf{t} \in I(j)$. Suppose that $t_k > 0$ for some index $k \in N \setminus \{j, k_1, \dots, k_r\}$. Then $\mathbf{e}^k \leq \mathbf{t}$ and $\mathbf{t} - \mathbf{e}^k \leq \mathbf{t}$, hence both vectors satisfy the constraints (6.14b). Moreover, both vectors satisfy the constraint (6.14a) because variable x_k is not involved there. Hence, $\mathbf{e}^k, \mathbf{t} - \mathbf{e}^k \in \mathscr{F}'_N$, which is a contradiction to $\mathbf{t} \in I(j)$.

Hence, there are indices $\sigma_1 < \cdots < \sigma_p$ such that $\mathbf{t} = \mathbf{e}^j + \sum_{\alpha=1}^p \mathbf{e}^{k_{\sigma_\alpha}}$. Therefore, $\mathbf{t}$ is computed by the algorithm of Table 11. □

Table 12 illustrates the effect of the *shift* value on the family of relaxations arising from a knapsack row of the problem R1100 from the MIPLIB [BCMS98]. This row is very dense, hence computing all the irreducible solutions is not tractable for $\mathit{shift} = 0$. In fact, for $\mathit{shift} < 53$, the number of irreducible solutions is greater than 100 000. By

Table 12. Number of irreducible solutions and average solution length in the one-parameter family of relaxations arising from row `R1100` in the MIPLIB problem `p0548`. For *shift* < 53, there are more than 100 000 irreducible solutions.

shift	Solutions	Avg. $\lvert\mathbf{t}\rvert$	*shift*	Solutions	Avg. $\lvert\mathbf{t}\rvert$	*shift*	Solutions	Avg. $\lvert\mathbf{t}\rvert$
53	90 769	5.66	73	8 905	4.60	93	807	3.37
54	81 746	5.61	74	7 819	4.53	94	720	3.31
55	73 588	5.55	75	6 887	4.46	95	590	3.29
56	66 268	5.50	76	6 094	4.37	96	419	3.25
57	59 293	5.46	77	5 525	4.31	97	327	3.12
58	52 862	5.41	78	4 931	4.25	98	283	3.02
59	47 254	5.36	79	4 384	4.20	99	229	2.88
60	42 236	5.30	80	4 016	4.14	100	206	2.78
61	37 632	5.23	81	3 518	4.12	101	206	2.75
62	33 591	5.17	82	3 027	4.07	102	190	2.70
63	29 964	5.12	83	2 652	4.02	103	164	2.64
64	26 484	5.06	84	2 292	3.94	104	164	2.64
65	23 621	5.01	85	1 980	3.83	105	139	2.56
66	21 181	4.96	86	1 751	3.75	106	128	2.52
67	18 942	4.91	87	1 549	3.66	107	118	2.47
68	17 017	4.86	88	1 326	3.58	108	94	2.30
69	15 284	4.81	89	1 158	3.56	109	79	2.13
70	13 569	4.76	90	1 057	3.51	110	76	2.08
71	11 873	4.71	91	947	3.46	111	73	2.00
72	10 266	4.67	92	903	3.42			

choosing a high *shift* value, we can however reduce the number of irreducible solutions. For *shift* $= 111$, only the 73 vectors corresponding to Theorem 6.1 remain. Table 12 also shows the average length of the irreducible vectors. For *shift* $= 111$, this value is 2 since the irreducible vectors are sums of two unit vectors. For smaller *shift* values, longer solution vectors are obtained.

7. Integral Basis Method Implementation Notes

We will now discuss details of our implementation of the Integral Basis Method. In the following we will work with a feasible simplex tableau (3.1), obtained by one of the methods described in section 2. We assume that all non-basic variables are 0/1, whereas the basic variables may be general integer variables. This setting is general enough to deal with $0/1$ integer programs, for which computational results are shown in section 8.

In our implementation, we compute irreducible solutions to knapsack relaxations of the original programs, which are strengthened by taking GUB constraints into account, as discussed in subsection 6.3. For this reason, we keep the set of known GUB constraints in a specialized data structure, rather than reading them off from the simplex tableau every time.

In the pure 0/1 setting, GUB constraints with a right-hand side of 1 play a special rôle. We note that the action of the replacement procedure on those constraints can, in fact, be interpreted as an action on a *set-packing subproblem*. This interpretation suggests to keep a *compatibility graph* $\mathscr{G}_c = (B \cup N, E)$ available during the course of the algorithm. This graph has a node for every basic and non-basic variable in the current problem formulation and an edge $(v, w) \in E$ whenever the variables x_v and x_w are *compatible*, i.e., they can simultaneously attain a non-zero value in a feasible integral solution $\mathbf{x} \in \mathscr{F}$.

In fact, our implementation of the Integral Basis Method maintains a formulation of the integer programming problem in the following form:

$$\begin{aligned}
\max \quad & c_0 + \bar{\mathbf{c}}^\top \mathbf{x}_N && && (7.1a)\\
\text{s.t.} \quad & (\mathbf{I}, \bar{\mathbf{A}})(\mathbf{x}_B, \mathbf{x}_N) = \bar{\mathbf{b}}, && && (7.1b)\\
& \mathbf{x} \leq \mathbf{u}, && && (7.1c)\\
& x_v + x_w \leq 1 && \text{for all } (v, w) \notin E, && (7.1d)\\
& (\mathbf{d}_B^\top, \mathbf{d}_N^\top)(\mathbf{x}_B, \mathbf{x}_N) \leq \delta && \text{for all } (\mathbf{d}, \delta) \in \mathscr{D}, && (7.1e)\\
& \mathbf{x} \in \mathbf{Z}_+^{n+m}. && && (7.1f)
\end{aligned}$$

In the preprocessing run (Table 13), we compute the variable bounds $\mathbf{u}$, the set $\mathscr{D}$ of known valid GUB constraints and the compatibility graph $\mathscr{G}_\mathrm{c} = (B \cup N, E)$. The only

Table 13. Bound tightening and preprocessing procedure

1: **input** an integer programming problem in tableau form (3.1).
2: **output** an equivalent formulation of the problem in the form (7.1) with a compatibility graph $\mathscr{G}_\mathrm{c} = (V, E)$, a vector of known upper bounds $\mathbf{u} \in \overline{\mathbf{R}}^{n+m}$ for all variables, and a set $\mathscr{D}$ of known valid GUB constraints.
3: Initialize $V \leftarrow B \cup N$ and $E \leftarrow \{\, (v, w) \in V \times V : v \neq w \,\}$.
4: Solve linear programs

$$\begin{aligned}
\max \quad & x_j \\
\text{s.t.} \quad & (\mathbf{I}, \bar{\mathbf{A}})(\mathbf{x}_B, \mathbf{x}_N) = \bar{\mathbf{b}}, \\
& \mathbf{x} \in \mathbf{R}_+^{n+m},
\end{aligned} \qquad (7.2)$$

to determine upper variable bounds $\mathbf{u} \in \mathbf{R}^{n+m}$. Delete variables fixed at zero.
5: Set $\mathscr{D} \leftarrow \left\{ (\mathbf{e}^i, u_i) : i \in B \cup N \right\}$.
6: **for all** $i \in B$ with $\bar{\mathbf{A}}_{i,\cdot} \geq \mathbf{0}$ **do**
7: Set $\mathscr{D} \leftarrow \mathscr{D} \cup \{((\mathbf{e}^i, \bar{\mathbf{A}}_{i,\cdot}), \bar{b}_i)\}$.
8: Let $N_> \leftarrow \{\, j \in N : \bar{A}_{ij} > 0 \,\}$.
9: **for all** $j_1, j_2 \in N_>$ with $\bar{A}_{ij_1} + \bar{A}_{ij_2} > \bar{b}_i$ **do**
10: Set $E \leftarrow E \setminus \{(j_1, j_2)\}$.
11: **return** $\mathscr{G}_\mathrm{c} = (V, E)$, $\mathbf{u}$, $\mathscr{D}$.

step that needs explanation in this algorithm is the loop 9–10. Since variables are non-negative, and matrix coefficients of GUB-rows are non-negative too, two variables cannot simultaneously attain a value of 1, if the condition checked in this step is fulfilled. Therefore we may delete the edge signifying compatibility between them from $\mathscr{G}_\mathrm{c}$.

In each substitution step in the course of the Integral Basis Method, the set $\mathscr{D}$ of GUB constraints is updated in the same way as we modify the tableau in the algorithm shown in Table 4 (step 12). The update of the compatibility graph $\mathscr{G}_\mathrm{c}$ is based on the following observation:

Proposition 7.1. *Let $I(j) = \{\mathbf{t}^1, \dots, \mathbf{t}^r\}$ be the set of irreducible solutions to the discrete relaxation $\mathscr{F}'_N$ with $t^k_j > 0$. For $i \in \{1, \dots, r\}$, let y_i be the new non-basic variable corresponding to the irreducible solution $\mathbf{t}^i \in I(j)$.*

(a) *The new variables $y_1, \dots, y_r$ are pairwise incompatible.*
(b) *Let $i \in \{1, \dots, r\}$. If x_u and x_v are incompatible and x_v is a "constituent" of y_i, i.e., $t^i_v > 0$, then x_u and y_i are incompatible in the reformulated system.*

The update of the GUB constraints and of the compatibility graph is shown in Table 14.

Using the algorithm shown in Table 11, we can calculate the set $I(j)$ very fast. However, we note that the column substitution according to step 12 of the algorithm shown in Table 4 introduces extra "symmetry" in the problem formulation in the following sense: A feasible solution to the original problem corresponds to several feasible solutions to the reformulated problem. This is to be avoided, since it slows down computations further on in the algorithm. In the following, we discuss how one may add combinatorial, *symmetry-breaking* constraints to the problem formulation, which reduce the number of representations of a feasible solution to the original problem.

To this end, let $\mathbf{t}^1, \mathbf{t}^2 \in I(j)$ be irreducible solutions and let y_1 and y_2 denote the variables in the reformulated problem corresponding to $\mathbf{t}^1$ and $\mathbf{t}^2$, respectively. Let $\mathbf{t} = \max\{\mathbf{t}^1, \mathbf{t}^2\}$ denote the component-wise maximum of $\mathbf{t}^1, \mathbf{t}^2$ and let $\hat{\mathbf{t}} \in \mathbf{Z}^n_+$ be an arbitrary vector with $\hat{\mathbf{t}} \geq \mathbf{t}$. This situation is illustrated by the diagram on the right.

$$\begin{tikzcd}[column sep=small]
 & \hat{\mathbf{t}} \arrow[d, dash] & \\
 & \mathbf{t} = \max\{\mathbf{t}^1, \mathbf{t}^2\} \arrow[dl, dash] \arrow[dr, dash] & \\
\mathbf{t}^1 \arrow[dr, dash] & & \mathbf{t}^2 \arrow[dl, dash] \\
 & \mathbf{e}^j \arrow[d, dash] & \\
 & \mathbf{0} &
\end{tikzcd}$$

We note that the vector $\mathbf{t}$ has *two* representations in the reformulated problem, namely

$$y_1 = 1 \quad \text{and} \quad x_k = 1 \text{ for } k \in \operatorname{supp} \mathbf{t}^2 \setminus \operatorname{supp} \mathbf{t}^1$$

and

$$y_2 = 1 \quad \text{and} \quad x_k = 1 \text{ for } k \in \operatorname{supp} \mathbf{t}^1 \setminus \operatorname{supp} \mathbf{t}^2,$$

all other variables being zero. The property of having two representations also carries over to all vectors $\hat{\mathbf{t}} \geq \mathbf{t}$.

The idea is to forbid one of these representations by adding a GUB constraint

$$y_1 + \sum_{k \in \operatorname{supp} \mathbf{t}^2 \setminus \operatorname{supp} \mathbf{t}^1} x_k \leq \left|\operatorname{supp} \mathbf{t}^2 \setminus \operatorname{supp} \mathbf{t}^1\right| \tag{7.3}$$

to the problem formulation (7.1) or, equivalently, by removing edges from the compatibility graph $\mathscr{G}_c$. This is accomplished by the algorithm shown in Table 15.

Proposition 7.2. *After adding symmetry-breaking constraints by the algorithm shown in Table 15, still all solutions to the original problem have a representation in the reformulated problem.*

In Table 16 we summarize our implementation of the Integral Basis Method.

Table 14. Procedure that updates the GUB constraints the compatibility graph

input the set $I(j) = \{\mathbf{t}^1, \dots, \mathbf{t}^r\}$ of irreducible solutions to the discrete relaxation $\mathscr{F}'_N$ with $t^k_j > 0$; the column indices $j_1, \dots, j_r \in N'$ of the new columns corresponding to these solutions; the set $\mathscr{D}$ of known valid GUB constraints; the compatibility graph $\mathscr{G}_c = (B \cup N, E)$.
output the updated set $\mathscr{D}'$ and the updated graph $\mathscr{G}_c{}' = (B \cup N', E')$.
$\mathscr{D}' \leftarrow \emptyset$.
for all $(\mathbf{d}, \delta) \in \mathscr{D}$ **do**
  Define $\mathbf{d}' \in \mathbf{Z}_+^{m+n'}$ by

$$d'_i = \begin{cases} d_i & \text{for } i \in B \cup N \setminus \{j\}, \\ \mathbf{d}_N^\top \mathbf{t}^k & \text{for } i \in N' \setminus N \text{ with } i = j_k. \end{cases}$$

  $\mathscr{D}' \leftarrow \mathscr{D}' \cup \{(\mathbf{d}', \delta)\}$.
$E' \leftarrow E \setminus \{\, (v, w) \in E : v = j \text{ or } w = j \,\}$.
for $i \leftarrow 1$ **to** r **do**
  $U \leftarrow \bigcap_{v \in \operatorname{supp} \mathbf{t}^i} \{\, u \in B \cup N : (u, v) \in E \,\}$.
  $E' \leftarrow E' \cup \{\, (j_i, u) : u \in U \,\}$.
return $\mathscr{D}', \mathscr{G}_c{}' = (B \cup N', E')$.

Table 15. Algorithm for breaking symmetry

input the set $I(j)$ of irreducible solutions $\mathbf{t}$ to the discrete relaxation $\mathscr{F}'_N$ with $t_j > 0$.
Fix some linear order $\preceq$ on $I(j)$.
for all $\mathbf{t}^1, \mathbf{t}^2 \in I(j)$ with $\mathbf{t}^1 \prec \mathbf{t}^2$ **do**
  Let i be the index of the column corresponding to $\mathbf{t}^1$.
  Let $S = \operatorname{supp} \mathbf{t}^2 \setminus \operatorname{supp} \mathbf{t}^1$.
  if $|S| = 1$ **then**
    Let $\{k\} = S$.
    $E \leftarrow E \setminus \{(k, i)\}$.
  else if $|S| \geq 2$ **then**
    $\mathscr{D} \leftarrow \mathscr{D} \cup \{(\mathbf{e}^i + \sum_{k \in S} \mathbf{e}^k, |S|)\}$.

We remark that the algorithm shown in Table 16 leaves unspecified how to choose the column index j, the row index i, and the *shift* value in steps 6, 15, and 16, respectively. Various strategies for these choices can be implemented, which we refrain from discussing in this paper.

8. Computational Results

In this section, we present computational results for the algorithms described in this paper. All computations were carried out on a SUN Enterprise 450 with 480 MHz.

8.1. Optimality Proofs

The first application of the Integral Basis Method is as a tool for proving optimality. We take an optimal solution, generate a simplex tableau for this solution, and start with

Table 16. Integral Basis Method using strengthened knapsack relaxations

input An integer programming problem in tableau form

$$\max c_0 + \bar{\mathbf{c}}^\top \mathbf{x}_N \ : \ \mathbf{x} \in \mathscr{F} = \left\{ \mathbf{x} \in \mathbf{Z}_+^{n+m} : (\mathbf{I}, \bar{\mathbf{A}})(\mathbf{x}_B, \mathbf{x}_N) = \bar{\mathbf{b}} \right\}$$

with data $\bar{\mathbf{A}} \in \mathbf{Z}^{m\times n}$, $\bar{\mathbf{b}} \in \mathbf{Z}_+^m$, $\bar{\mathbf{c}} \in \mathbf{Z}^n$, and $c_0 \in \mathbf{Z}$, where all non-basic variables are 0/1.

output an optimal point $\mathbf{x}^*$ with an optimality certificate.

Preprocess the problem with the algorithm shown in Table 13, computing a vector $\mathbf{u} \in \overline{\mathbf{R}}^{n+m}$ of known upper bounds for all variables, a set $\mathscr{D}$ of known valid GUB constraints and a compatibility graph $\mathscr{G}_c = (B \cup N, E)$.

$N_+ \leftarrow \{\, j \in N : \bar{c}_j > 0 \,\}$.

while $N_+ \neq \emptyset$ **do**

  Select $j \in N_+$.

  if $u_j < 1$ **then**

    Delete the column with index j.

  else if $\mathbf{e}^j \in \mathscr{F}_N$ **then**

    (∗ *the vector* $(-\bar{\mathbf{A}}_{\cdot j}, \mathbf{e}^j)$ *is an augmentation direction w.r.t.* $(\bar{\mathbf{b}}, \mathbf{0})$ ∗)

    Let $i \in B$ be the index of a row such that $\bar{b}_i = 1$ and $\bar{A}_{ij} = 1$ (if no such row exists, add the constraint $x_j \leq 1$ to the tableau).

    Perform a non-degenerate simplex pivot operation on the binary variable x_j and the i-th row, updating N and B. Recompute N_+.

  else

    (∗ *the* j*-th column cannot be applied, so we replace it by other columns corresponding to irreducible solutions to some discrete relaxation* ∗)

    Choose the index $i \in B$ of a tableau row $(\mathbf{e}^{i\top}, \bar{\mathbf{A}}_{i,\cdot})(\mathbf{x}_B, \mathbf{x}_N) = \bar{b}_i$ with $\bar{A}_{ij} > \bar{b}_i$.

    Choose a *shift* value.

    Determine the set $I(j) = \{\mathbf{t}^1, \ldots, \mathbf{t}^r\}$ of all irreducible solutions $\mathbf{z} \in \mathscr{F}'_N$ with $z_j > 0$ to the discrete relaxation (6.14) using the algorithm shown in Table 11.

    Delete the column with index j.

    Introduce r columns corresponding to the elements in $I(j)$, i.e., the new column i has entries $\bar{\mathbf{A}}\mathbf{t}^i$ and reduced cost $\bar{\mathbf{c}}^\top \mathbf{t}^i$.

    Set $n \leftarrow n - 1 + r$ and update N and N_+.

    Update the GUB constraints and the compatibility graph $\mathscr{G}_c$ (Table 14).

    Break symmetry (Table 15).

    Compute upper bounds u_k for all new columns by solving linear programs of the form (7.2).

return "Optimal."

the preprocessing algorithm (Table 13). Since the problems under consideration have only binary variables, we construct a simplex basis for the solution in the manner of Example 2.6.

Table 17 shows a few problem instances from the MIPLIB [BCMS98] where setting up a simplex tableau for the optimal integral solution and applying the preprocessing algorithm (Table 13) suffice to prove optimality. The table shows the number of columns and rows of the problem and the time spent in the preprocessing stage. Not a single iteration of the Integral Basis Method is necessary in these instances.

In general, however, proving optimality of a solution involves actual application of the algorithm shown in Table 16. Table 18 displays computation times for a number of MIPLIB-instances. It lists initial problem size and the number of iterations of the algorithm, together with the computation times including preprocessing. The maximum

Table 17. Verifying optimality by preprocessing

	Columns	Rows	Time (s)
enigma	121	21	0.0
nw08	434	24	0.9
cracpb1	572	144	2.1
air03	10757	124	8935.6

Table 18. Applying the Integral Basis Method at an optimal solution

	Columns	Rows	Iterations	Max. Size	Time (s)
lseu	89	28	95	159	7.0
mod008	319	6	27	285	54.7
p0033	33	15	31	46	0.6
p0040	40	23	9	18	0.2
p0201	201	133	123	266	23.3
p0282	282	465	22	142	5.2
p0291	291	252	7	391	21.5
p0548	548	307	197	979	55.5
p2756	2756	1700	162	3288	493.1
cap6000	6000	2176	2128	825	1263.1
l152lav	1989	192	320	592	1015.2

number of columns in the tableau during the course of the algorithm is listed under "Max. Size".

8.2. Near the Optimum

We also test the performance of the Integral Basis Method performs when applied to a non-optimal solution. Using CPLEX 6.6 without probing and allowing only limited cuts we obtain initial solutions that still leave a few percent gap to the optimum. We construct simplex tableaux for these solutions and run the algorithms shown in Tables 13 and 16. Whenever an augmenting direction is detected in step 10 of the Integral Basis Method, the algorithm performs an integral simplex pivot step, and then continues with the resulting tableau. Table 19 lists the number of columns currently in the tableau and the number of columns with positive reduced cost remaining in each augmentation step for the problems p0033, p0201, p0282, and p0548, respectively.

8.3. The Effect of Symmetry-Breaking Constraints

In Table 20 we illustrate the effect of the symmetry-breaking constraints discussed in section 7. We run a number of MIPLIB instances both with the normal implementation and with an implementation where we turn off generation of symmetry-breaking constraints. For both cases, the table shows the number of iterations, the maximum number of columns during the course of the algorithm, and the computation time.

Table 19. Starting the Integral Basis Method from a suboptimal point

	Iteration	Time (s)	Columns	Positives	Objective
		A. p0033			
start	0	0.0	33		3438
augmentation	4	0.2	34	13	3188
augmentation	8	0.3	41	11	3089
optimality	39	0.7	44	0	3089
		B. p0201			
start	0	0.0	201		7715
augmentation	248	45.4	261	23	7615
optimality	265	48.6	69	0	7615
		C. p0282			
start	0	0.0	282	40	263797
augmentation	0	1.9	280	31	263010
augmentation	183	39.3	313	99	260888
augmentation	243	42.4	179	7	260576
augmentation	247	43.1	148	4	258682
augmentation	247	43.2	148	2	258561
optimality	248	43.7	77	0	258411
		D. p0548			
start	0	0.0	548	80	8837
augmentation	34	11.6	540	114	8795
augmentation	39	12.6	571	125	8775
augmentation	39	12.8	571	123	8759
augmentation	39	13.1	571	121	8717
augmentation	44	18.3	570	110	8707
augmentation	348	120.0	247	35	8701
augmentation	356	121.3	90	1	8691
optimality	356	121.4	40	0	8691

9. The Integral Basis Method in an Augmentation Framework for Integer Programming

The computational results presented so far rely on our ability to detect a solution close to the optimum. In fact, our computational tests show, that if one starts with an initial solution that is far away from the optimum, a high number of iterations is required for the algorithm to terminate. Therefore we were interested in applying a heuristic to improve the original feasible solution before starting with the Integral Basis Method.

9.1. An Augmentation Heuristic Using Lattice Reduction

We will present an augmentation heuristic that solely relies on the linear integer programming formulation, and can therefore be used instead of problem-specific methods that are typically available for combinatorial problems.

Table 20. The effect of symmetry-breaking constraints on some MIPLIB instances. An asterisk ($\star$) in the "Initial objective value" column indicates the objective value of the optimal solution. For the entries that have been left blank, the computation did not finish within 15 hours of CPU time.

		With breaking symmetry			Without breaking symmetry		
	Initial obj.	Iterations	Max. Size	Time (s)	Iterations	Max. Size	Time (s)
`lseu`	1120*	95	159	6.8	102	203	8.5
`mod008`	307*	27	285	55.0	53	483	235.1
`p0033`	3089*	31	46	0.5	36	67	0.6
	3438	39	59	0.7	51	90	0.9
`p0201`	7615*	123	266	23.2	160	266	34.0
	7715	265	436	49.3	272	502	59.6
	7945	667	1131	301.8	807	1456	625.7
	8535	1368	2324	2012.0	1949	3139	3833.4
`p0282`	258411*	22	142	5.2	23	144	5.7
	263797	248	556	44.2			
`p0548`	8691*	197	979	55.6	270	1150	125.7
	8837	356	887	122.8			
`p2756`	3124*	162	3288	491.3	163	3441	670.4
`cap6000`	2451377*	2128	825	1263.4			

In this section we consider integer programming problems in a form which may seem a little uncommon at first. It provides the most convenient setting however, and conversion from other problem formulations presents no difficulty:

$$\begin{aligned} \max \quad & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} \quad \mathbf{l} \leq & \begin{pmatrix} \mathbf{I} \\ \mathbf{A} \end{pmatrix} \mathbf{x} \leq \mathbf{u} \\ & \mathbf{x} \in \mathbf{Z}^n \end{aligned} \tag{9.1}$$

where all data is assumed to be integral. We presume that an initial feasible solution $\mathbf{x}^0$ to the set of restrictions is given.

The problem formulation (9.1) contains three types of restrictions: the bound constraints on the variables, range constraints on the product $\mathbf{Ax}$, and—if components of $\mathbf{l}$ and $\mathbf{u}$ are equal—equality constraints on some rows of $\mathbf{Ax}$. Intuitively, an augmentation vector $\mathbf{t}$ for the feasible solution $\mathbf{x}^0$ therefore can be found if it satisfies any equality constraints that might exist, and is "short" in the sense that it does not change $\mathbf{x}^0$ so much as to make $\mathbf{x}^0 + \mathbf{t}$ violate the other restrictions; see Figure 1.

Given a set of n linearly independent integral vectors $\{\mathbf{l}^1, \dots, \mathbf{l}^n\}$ in d-dimensional Euclidean space $\mathbf{R}^d$, the set Λ of all integral linear combinations of them is called the *lattice* generated by $\mathbf{l}^1, \dots, \mathbf{l}^n$, i.e., $\Lambda = \{\, \mathbf{v} \in \mathbf{Z}^d : \mathbf{v} = \sum_{i=1}^n \lambda_i \mathbf{l}^i,\ \lambda_i \in \mathbf{Z} \,\}$. Any such set of linearly independent vectors is called a *basis* of the lattice. Let $\|\mathbf{v}\| := \langle \mathbf{v}, \mathbf{v} \rangle_{\mathbf{M}}^{1/2}$ denote an arbitrary elliptical norm on $\mathbf{R}^d$, i.e. $\langle \mathbf{v}, \mathbf{v} \rangle_{\mathbf{M}} = \mathbf{v}^\top \mathbf{M} \mathbf{v}$ for a positive definite matrix $\mathbf{M}$.

In a lattice Λ given by a basis, the *Shortest Vector Problem* is the task to find a vector $\mathbf{v} \in \Lambda$ such that $\mathbf{v} \neq \mathbf{0}$ and $\|\mathbf{v}\| = \min \{\, \|\mathbf{v}\| \ :\ \mathbf{v} \in \Lambda \,\}$. It is a classical result of Minkowski that every d-dimensional lattice contains a vector with length at most $2(\det \Lambda)^{1/d} V_d^{-1/d}$, where V_d denotes the volume of the d-dimensional unit ball. It is known (see [Mic98]) that the Shortest Vector Problem is $\mathcal{NP}$-hard under randomized reductions, but no

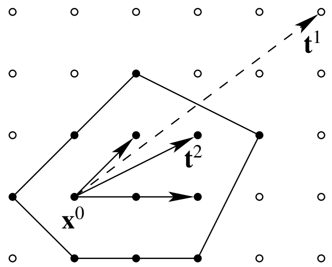

Fig. 1. Why short vectors are suitable for augmentation. The vector $\mathbf{t}^1$ is so long that $\mathbf{x}^0 + \mathbf{t}^1$ is already outside the polyhedron. Augmenting with vectors like $\mathbf{t}^2$ increases the chance of remaining in the feasible region.

deterministic proof of its $\mathscr{NP}$-hardness exists. One can, however, find a "longer short vector" using the Lovász lattice reduction algorithm in polynomial time:

Theorem 9.1 (Lenstra, Lenstra, Lovász 1982). *For any lattice Λ in R^d the Lovász lattice reduction algorithm [LLL82] computes a nonzero vector $\mathbf{v} \in \Lambda$ with norm $\|\mathbf{v}\|$ at most $2^{d(d-1)/4}(\det \Lambda)^{1/d}$ in polynomial time.*

The Lovász lattice reduction algorithm has a quality parameter δ. (It determines the "factor of decrease" allowed between two consecutive vectors in the Gram–Schmidt orthogonalization of the current lattice basis, and is usually set as $\delta = \frac{3}{4}$.) We set $\delta = 0.99$ for our computations, which seems to be a good choice for practical purposes, see for example [SE94].

Seeing that rather short lattice vectors can be found using lattice reduction, we are also interested in the reverse question. A vector $\mathbf{v} \in \Lambda$ is called *primitive*, if for every $k > 1$ we have $\frac{1}{k}\mathbf{v} \notin \Lambda$. Can every primitive lattice vector be found with an algorithm that finds short lattice vectors?

Theorem 9.2. *Every primitive lattice vector $\mathbf{v} = (v_1, \ldots, v_d)^\top$ is shortest with respect to some elliptical norm.*

Proof. We can assume that the lattice is not contained in a linear subspace of $\mathbf{R}^d$, so that $d = n$. Let $\mathbf{v}$ be primitive and choose $\mathbf{b}^2, \ldots, \mathbf{b}^n$ primitive, such that $(\mathbf{v}, \mathbf{b}^2, \ldots, \mathbf{b}^n)$ form a basis of the lattice. Now let $\mathbf{A} := (\mathbf{v}, \frac{1}{2}\mathbf{b}^2, \ldots, \frac{1}{2}\mathbf{b}^n)^{-1}$ be the matrix mapping $\mathbf{v}$ to $\mathbf{e}^1$ (the first unit vector), and each $\mathbf{b}^i$ to $2\mathbf{e}^i$. Then $\mathbf{A}^\top\mathbf{A}$ is positive definite, defines a scalar product, and using bilinearity can be seen to generate an elliptical norm where $\mathbf{v}$ and $-\mathbf{v}$ are the only vectors of Λ with norm 1, while all other nonzero lattice vectors have length > 1. $\square$

This theorem guarantees that every vector one could want to apply to augment a given feasible solution is shortest if the right elliptical norm is used to measure it. It justifies the reverse procedure: applying lattice reduction to the standard lattice while choosing a suitable elliptical norm to derive augmentation vectors for the problem (9.1). This method is oulined in table 21.

Table 21. Augmentation procedure based on lattice reduction

1: **input** integer program in the form (9.1) with m rows, and a feasible solution $\mathbf{x}^0$.
2: **output** feasible solution $\mathbf{x}'$.
3: Choose an initial elliptical norm.
4: Initialize $\mathbf{x}' \leftarrow \mathbf{x}^\mathbf{0}$.
5: **loop**
6:   Apply Lovász lattice reduction to the matrix $\begin{pmatrix}\mathbf{I}\\ \mathbf{A}\end{pmatrix}$ with respect to the chosen norm and with quality parameter $\delta = 0.99$, yielding the matrix $\begin{pmatrix}\mathbf{L}\\ \mathbf{A}'\end{pmatrix}$.
7:   **for all** $\mathbf{t} \in \{\mathbf{L}_{\cdot,i} : i \in \{1,\ldots,n\}\}$ **do**
8:     **if** $\mathbf{t}$ is an augmenting direction for $\mathbf{x}'$ **then**
9:       Apply $\mathbf{t}$, i.e. $\mathbf{x}' \leftarrow \mathbf{x}' + \mathbf{t}$.
10:     **else if** $-\mathbf{t}$ is an augmenting direction for $\mathbf{x}'$ **then**
11:       Apply $-\mathbf{t}$, i.e. $\mathbf{x}' \leftarrow \mathbf{x}' - \mathbf{t}$.
12:   Identify why augmentation with (a subset of) the current lattice basis fails and modify the norm appropriately.
13:   **if** progress has been too slow or gap to dual bound is small enough **then**
14:     **return** current solution $\mathbf{x}'$.

There are a number of choices left open in this algorithm. It is easy to incorporate a steering parameter that counts the number of unsuccessful rounds in the loop starting at step 5 and terminates the algorithm after recurrent failure. Note that the order of testing applicability of the generated vectors in step 7 is relevant. One may decide to use the vectors in the order that they have in the lattice basis, or (in a greedy fashion), in reverse order, or test various possible choices each time in an attempt to maximize progress per lattice basis generated. The other large degree of freedom is step 12, details of which have deliberately been omitted in the presentation of the algorithm. Example 9.3 presents an explicit construction that works nicely for small examples, while for larger problems more tuning is necessary. In practice we choose to work only with norms defined by $||\mathbf{v}||^2 = \mathbf{v}^\mathsf{T}\mathbf{M}\mathbf{v}$ where $\mathbf{M}$ is a diagonal matrix (with positive entries), and in a first phase increase entries on the diagonal of $\mathbf{M}$ until all equality rows (those with $l_i = u_i$ in (9.1)) are satisfied by most lattice basis vectors. This can easily be achieved with relatively small multipliers. After that we switch to the method presented in Example 9.3. A time limit is used to stop the procedure.

Example 9.3. Consider the system (9.2) which describes a cube with the three vertices adjacent to the feasible point $x^0 := \begin{pmatrix}1\\1\\1\end{pmatrix}$ cut off:

$$\begin{array}{lc}
\min & (\;1\;\;1\;\;1\;)\,\mathbf{x} \\
\text{s.t.} & \begin{pmatrix}0\\0\\0\\-1\\-1\\-1\end{pmatrix} \le \begin{pmatrix}1&0&0\\0&1&0\\0&0&1\\1&1&-1\\1&-1&1\\-1&1&1\end{pmatrix}\mathbf{x} \le \begin{pmatrix}1\\1\\1\\1\\1\\1\end{pmatrix} \\
 & \mathbf{x} \in \mathbf{Z}^3
\end{array} \tag{9.2}$$

We start with the basis of unit vectors $\mathbf{e}^1, \mathbf{e}^2, \mathbf{e}^3$ for the standard lattice. The columns of (9.2) are not changed by applying Lovász lattice reduction w.r.t. the standard Euclidean norm given by $||\mathbf{v}||^2 := \sum_{i=1}^{6} v_i^2$. None of the vectors $\pm\mathbf{e}^1, \pm\mathbf{e}^2, \pm\mathbf{e}^3$ is a feasible direction at $\mathbf{x}_0$, so we want to modify the norm to have the lattice reduction algorithm produce new potential augmentation vectors.

A simple-minded way is the following: We select the first vector $\mathbf{l}^1$ in the reduced lattice basis (it is, up to the quality parameter δ, approximately the shortest vector in it), and check how much each ranged inequality is violated for $\mathbf{x}_0 + \mathbf{l}^1$ and for $\mathbf{x}_0 - \mathbf{l}^1$, i.e.,

$$\Delta_i^{\pm} := \max\{\, \mathbf{a}_i^{\top}(\mathbf{x}_0 \pm \mathbf{l}^1) - u_i,\ l_i - \mathbf{a}_i^{\top}(\mathbf{x}_0 \pm \mathbf{l}^1) \,\},$$

and count how many rows they each violate, $k^{\pm} = \left|\{\, \Delta_i^{\pm} : \Delta_i^{\pm} > 0 \,\}\right|$.

In the example we find that $\mathbf{x}_0 + \mathbf{e}^1$ violates three inequalities (the first, fourth, and fifth), each by a value of 1. The point $\mathbf{x}_0 - \mathbf{e}^1$ violates only the sixth row, again by a value of 1. We choose the norm in such a way, that each row i of the problem gets an additional weight of $k^+\Delta_i^+ + k^-\Delta_i^-$, i.e., the norm becomes

$$\|\mathbf{v}\|^2 := (4\mathbf{v}_1)^2 + \mathbf{v}_2^2 + \mathbf{v}_3^2 + (4\mathbf{v}_4)^2 + (4\mathbf{v}_5)^2 + (2\mathbf{v}_6)^2.$$

Applying Lovász lattice reduction with quality parameter $\delta = 0.9$ to the columns of (9.2) yields the basis $\left(\begin{smallmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{smallmatrix}\right)$. The negative of the first basis vector is now applicable (once), improving the objective value by -2. Noting that we can now apply $-\mathbf{e}^2$ for an additional improvement of -1 we arrive at the point $\mathbf{0}$. In fact, we could once more have modified the norm after the first augmentation step using the same technique as above, and would have found $\mathbf{e}^2$ in the resulting basis, too.

We therefore replace the steps 3 and 12 of the algorithm shown in Table 21 by the following (denoting the matrix rows of $\left(\begin{smallmatrix} \mathbf{I} \\ \mathbf{A} \end{smallmatrix}\right)$ in (9.1) by $\hat{\mathbf{a}}_i$):

3': Start with the Euclidean norm, i.e., $\|\mathbf{v}\| \leftarrow \sqrt{\mathbf{v}^{\top}\mathbf{M}\mathbf{v}}$ with $\mathbf{M} := \mathbf{I}$, the identity matrix.

12': Determine how much the first lattice basis vector violates each row $\hat{\mathbf{a}}_i$ of (9.1): Set

$$\Delta_i^{\pm} \leftarrow \max\{\, \hat{\mathbf{a}}_i^{\top}(\mathbf{x}' \pm \mathbf{L}_{\cdot,1}) - u_i,\ l_i - \hat{\mathbf{a}}_i^{\top}(\mathbf{x}' \pm \mathbf{L}_{\cdot,1}) \,\}$$

for all $i \in \{1, \ldots, m\}$, and $k^{\pm} \leftarrow \left|\{\, \Delta_i^{\pm} : \Delta_i^{\pm} > 0,\ i \in \{1, \ldots, m\} \,\}\right|$. Update the norm-defining matrix $\mathbf{M}$ by setting $M_{i,i} \leftarrow \left(\sqrt{M_{i,i}} + (k^+\Delta_i^+ + k^-\Delta_i^-)\right)^2$ for all $i \in \{1, \ldots, m\}$.

9.2. *Results*

Finally, we will present a complete run for the `p0548` problem. For problems of this size (larger than a few hundred variables), lattice reduction is a bottleneck, since our heuristic augmentation procedure computes a reduced lattice basis after each change of the norm, amounting to a hundred or more calls to the Lovász basis reduction algorithm per problem. Note however, that for small changes in the norm the reduced basis does not change very much, so subsequent calls to the Lovász basis reduction algorithm take less time than the first one.

Table 22. Blockwise lattice reduction for quality parameter $\delta = 0.99$

	Computation time (s)		
Blocks	`p0201`	`p0282`	`p0548`
1	6.0	49.4	695.5
2	3.0	20.3	489.9
4	2.3	20.2	441.1
8	2.1	21.7	633.8
16	2.0	22.2	647.1

We implemented a simple heuristic divide-and-conquer algorithm to speed up lattice reduction: The lattice generators are divided into a number of subsets, then the standard lattice reduction algorithm is applied to each subset. Pairs of subsets are joined, and the larger sets are subjected to lattice reduction again. This is repeated until only one set remains, yielding a reduced basis of the original lattice. We choose to group generators by similar Euclidean length, and join subsets by successively adjoining vectors from one set to the other, reducing each new vector immediately. This method usually reduces overall computation time; an additional speedup can be seen on large problems when using a multi-processor machine, where lattice reduction on the blocks can be distributed onto different processors. There has been some research on parallelization of the lattice reduction algorithm itself, see for example [HT98] and [Hec95], but since the focus in those papers was on lattices from number-theoretic applications (very big coefficients of the generators, relatively low dimension), the results are not directly applicable to our type of problems.

Table 22 shows the computation times needed to complete Lovász basis reduction for the lattice generated by the columns of some MIPLIB-problems, after transforming them into the form (9.1), for different numbers of blocks.

To conclude we present Table 23 and Figure 2, where all algorithms of this paper were combined. We start with a feasible solution with objective value 13577 that is obtained by running CPLEX 6.6 for a few seconds. At this solution we set up the problem in the form (9.1), and run the augmentation procedure (Table 21) with a time limit of 1 hour. The augmentation vectors thus derived improve the solution to an objective value of 9973. Here we construct a simplex basis using the method of Example 2.6, and start the Integral Basis Method.

The table also contains a column showing a dual bound obtained by maximizing the tableau using an LP solver after dropping integrality demands at some steps. Note that the continuous reformulation that is done in the column substitution steps of the Integral Basis Method also has a strong effect on the quality of the linear programming solution. Figure 2 displays a plot of the changes in the LP bound as well as the objective value of the current feasible solution over the course of the algorithm.

Table 23. Optimizing `p0548` from scratch

	Iteration	Time	Columns	Positives	Objective	LP-Bound
start			548		13577	
step to heuristic solution	0	0.0	548		9973	
LP-bound change	4	11.4	505	93	9973	572.4
LP-bound change	31	14.1	540	126	9973	3874.2
LP-bound change	32	14.1	540	126	9973	4682.7
LP-bound change	33	14.3	552	127	9973	4711.0
LP-bound change	34	14.5	553	127	9973	5312.4
augmentation	40	16.0	603	137	9972	5358.6
LP-bound change	40	21.0	587	130	9972	6568.3
augmentation	42	21.5	595	128	9910	6568.3
LP-bound change	42	26.2	592	128	9910	6624.6
augmentation	46	27.0	609	132	9721	6700.7
augmentation	46	27.3	609	131	9619	6700.7
augmentation	46	27.6	609	127	9393	6700.7
augmentation	51	33.2	599	119	9338	6700.7
LP-bound change	52	38.3	603	120	9338	6738.7
LP-bound change	57	39.2	641	126	9338	7019.3
augmentation	110	60.2	1033	347	9328	7930.5
LP-bound change	131	77.2	990	291	9328	8022.6
augmentation	255	120.7	1021	273	8963	8496.0
augmentation	1305	386.2	2705	864	8960	8683.2
augmentation	1337	428.8	2654	712	8824	8683.2
LP-bound change	1375	471.4	394	70	8824	8684.5
augmentation	1470	498.6	653	225	8780	8684.5
augmentation	1470	498.9	653	221	8745	8684.5
LP-bound change	1470	504.5	232	42	8745	8686.7
LP-bound change	1474	505.2	255	44	8745	8688.3
augmentation	1492	512.5	348	59	8738	8688.3
augmentation	1495	515.1	270	34	8734	8688.3
LP-bound change	1500	516.7	256	32	8734	8688.5
augmentation	1507	517.7	286	37	8700	8688.5
LP-bound change	1507	519.0	75	4	8700	8691.0
augmentation	1509	519.1	76	3	8691	8691.0
optimality	1510	519.2	0	0	8691	8691.0

References

[AHL00] K. Aardal, C. A. J. Hurkens, and A. K. Lenstra, *Solving a system of diophantine equations with lower and upper bounds on the variables*, Mathematics of Operations Research **25** (2000), 427–442.

[AWW00] K. Aardal, R. Weismantel, and L. Wolsey, *Non-standard approaches to integer programming*, CORE Discussion Paper, no. 2000/2, Université Catholique de Louvain, CORE, Voie du Roman Pays 34, B-1348 Louvain-la-Neuve, Belgium, January 2000.

[BCMS98] R. E. Bixby, S. Ceria, C. M. McZeal, and M. W. P. Savelsbergh, *An updated mixed integer programming library:* MIPLIB *3.0*, Optima **58** (1998), 12–15.

[BIC62] A. Ben-Israel and A. Charnes, *On some problems of diophantine programming*, Cahiers du Centre d'Études de Recherche Opérationnelle **4** (1962), 215–280.

[BP75] E. Balas and M. Padberg, *On the set-covering problem: II. An algorithm for set partitioning*, Operations Research **23** (1975), 74–90.

[CRSS93] W. Cook, T. Rutherford, H. E. Scarf, and D. Shallcross, *An implementation of the generalized basis reduction algorithm for integer programming*, ORSA Journal on Computing **5** (1993), no. 2, 206–212.

[CT91] P. Conti and C. Traverso, *Buchberger algorithm and integer programming*, Proceedings of the AAECC-9, Lecture Notes in Computer Science 539, Springer, Berlin, 1991, pp. 130–139.

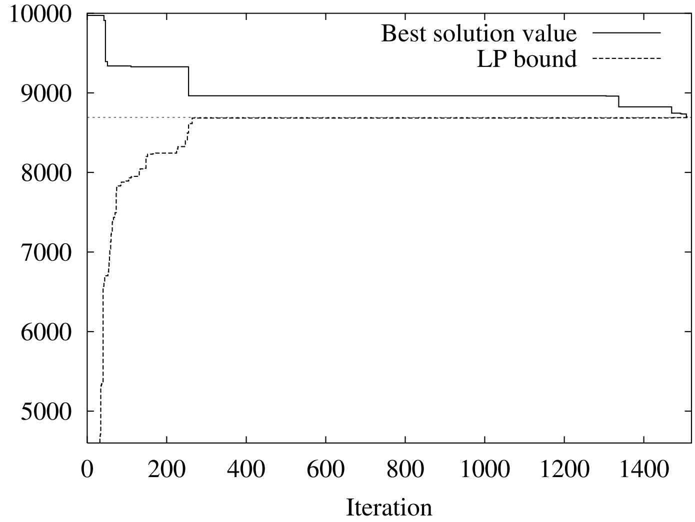

Fig. 2. Plot of optimization progress and dual bound for `p0548`

[DGS96] P. Diaconis, R. L. Graham, and B. Sturmfels, *Primitive partition identities*, Combinatorics, Paul Erdős is Eighty, Volume 2 (D. Miklós, V. T. Sós, and D. Szőnyi, eds.), Bolyai Society Mathematical Studies, vol. 2, 1996, pp. 173–192.

[GHK+01] C. Gentile, U.-U. Haus, M. Köppe, G. Rinaldi, and R. Weismantel, *On the way to perfection: A new combinatorial algorithm for stable sets in graphs*, Preprint no. 24, Fakultät für Mathematik, Otto-von-Guericke-Universität Magdeburg, 2001, available from URL `http://www.math.uni-magdeburg.de/~mkoeppe/art/primstab.ps`.

[Glo68] F. Glover, *A new foundation for a simplified primal integer programming algorithm*, Operations Res. **16** (1968), 727–740.

[GN72] R. S. Garfinkel and G. L. Nemhauser, *Integer programming*, Wiley, New York, 1972.

[Gom58] R. E. Gomory, *Outline of an algorithm for integer solutions to linear programs*, Bulletin of the American Mathematical Society **64** (1958), 275–278.

[Gom60] R. E. Gomory, *An Algorithm for the Mixed Integer Problem*, Tech. Report RM-2597-PR, The RAND Corporation, Santa Monica, CA, 1960.

[Gom63] R. E. Gomory, *An all-integer integer programming algorithm*, Industrial Scheduling (J. F. Muth and G. L. Thompson, eds.), Prentice Hall, Englewood Cliffs, N.J., 1963, pp. 193–206.

[Hec95] C. Heckler, *Automatische Parallelisierung und parallele Gitterbasisreduktion*, Dissertation, Institut für Mikroelektronik, Universität des Saarlandes, Saarbrücken, Germany, 1995.

[HKW01] U.-U. Haus, M. Köppe, and R. Weismantel, *The Integral Basis Method for integer programming*, Mathematical Methods of Operations Research **53** (2001), no. 3, 353–361.

[HT98] C. Heckler and L. Thiele, *Complexity analysis of a parallel lattice basis reduction algorithm*, SIAM J. Comput. **27** (1998), no. 5, 1295–1302 (electronic).

[Hu69] T. C. Hu, *Integer programming and network flows*, Addison–Wesley, Reading, Mass., 1969.

[HW00] M. Henk and R. Weismantel, *On minimal solutions of diophantine equations*, Beiträge zur Algebra und Geometrie **41** (2000), no. 1, 49–55.

[Köp99] M. Köppe, *Erzeugende Mengen für gemischt-ganzzahlige Programme*, Diploma thesis, Otto-von-Guericke-Universität Magdeburg, 1999, available from URL `http://www.math.uni-magdeburg.de/~mkoeppe/art/mkoeppe-diplom.ps`.

[Lam87] J. L. Lambert, *Une borne pour les générateurs des solutions entiéres positives d'une équation diophantienne linéaire*, C. R. Acad. Sci. Paris **305** (1987), 39–40.

[Len83] H. W. Lenstra, Jr., *Integer programming with a fixed number of variables*, Mathematics of Operations Research **8** (1983), 538–548.

[LLL82] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász, *Factoring polynomials with rational coefficients*, Mathematische Annalen **261** (1982), 515–534.

[Mic98] D. Micciancio, *The shortest vector in a lattice is hard to approximate to within some constant*, IEEE Symposium on Foundations of Computer Science, 1998, pp. 92–98.

[PH80] M. W. Padberg and S. Hong, *On the symmetric travelling salesman problem: a computational study*, Math. Programming Stud. (1980), no. 12, 78–107.

[SE94] C.-P. Schnorr and M. Euchner, *Lattice basis reduction: improved practical algorithms and solving subset sum problems*, Math. Programming **66** (1994), no. 2, Ser. A, 181–199.

[Tho95] R. R. Thomas, *A geometric Buchberger algorithm for integer programming*, Mathematics of Operations Research **20** (1995), 864–884.

[UWZ97] R. Urbaniak, R. Weismantel, and G. Ziegler, *A variant of Buchberger's algorithm for integer programming*, SIAM Journal on Discrete Mathematics **1** (1997), 96–108.

[Wei98] R. Weismantel, *Test sets of integer programs*, Mathematical Methods of Operations Research **47** (1998), 1–37.

[You68] R. D. Young, *A simplified primal (all-integer) integer programming algorithm*, Operations Research **16** (1968), no. 4, 750–782.

[You75] R. D. Young, *The eclectic primal algorithm: Cutting-plane method that accommodates hybrid subproblem solution techniques.*, Math. Program. **9** (1975), 294–312.