The Integral Basis Method for Integer Programming

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

This paper introduces an exact algorithm for solving integer programs, neither using cutting planes nor enumeration techniques. It is a primal augmentation algorithm that relies on iteratively substituting one column by columns that correspond to irreducible solutions of certain linear diophantine inequalities. We demonstrate the algorithm's potential by testing it on some instances of the MIPLIB with up to 6000 variables.

1 Introduction

Many computational experiments in the past twenty years have revealed that linear programming based branch-and-bound algorithms with cuts added are currently the most successful tool for attacking combinatorial and linear integer programming problems without any given structure. Such algorithms are known as *branch-and-cut* and may be interpreted as dual type methods. Current implementations of branch-and-cut algorithms heavily rely on the polyhedral theory for designing the cutting phase of such an algorithm.

This is in contrast with the situation on the primal side. It is undoubtable that from a theoretical point of view primal methods have been the starting point for many deep algorithmic results related to the theory about flows, matroids, and matchings. The practical applicability of this theoretical knowledge for general combinatorial or integer programs has however never been proved. No theory enters the design of the primal phase of today's integer programming algorithms, i.e., all codes resort to usually pretty obvious rounding heuristics for generating feasible solutions.

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

There are however alternative approaches for solving integer programs available that depend explicitly on the discrete nature of the set of feasible solutions. Perhaps most important is the use of lattice basis theory. This theory has become extremely important in the theory of integer programming due to Lenstra's result [Len83] and Lovász's basis reduction algorithm [LLL82]. Computationally, reduced bases of a lattice have not been used too often in linear integer programming. One reason might be that the computation of a reduced basis of a lattice becomes quite expensive when the number of rows and columns of the matrix are in the range of a couple of hundreds. Secondly, the procedure of computing a reduced basis of a lattice must be combined with a branch and bound algorithm. Although theoretically the number of nodes to be explored is controllable, practically it can still be expensive. However, reformulation of the problem using lattice basis reduction turns out to be a successful method for solving a hard class of integer programs, see [ABH$^+$00, Was00].

The situation is worse when it comes to test sets. Test sets are a collection of lattice points such that every feasible non-optimal point of a linear integer program can be improved by applying a vector from the test set. Although test sets are finite under modest assumptions, they are usually huge. This fact is not surprising because minimal test sets w.r.t. inclusion can be obtained as the union of Hilbert bases of rational polyhedral cones. In particular, the union is taken over all orthants of the space of variables, which is an exponential construction. We are not aware that test sets have been employed with computational success. Also, the situation does not improve when one resorts to other approaches for determining test sets such as neighbors of the origin or Gröbner bases for integer programs.

This paper introduces another alternative for solving integer programs of the form

$$\max c^\top x : \ x \in \mathscr{F} := \{ x \in \mathbf{Z}^n : Ax = b, \ x \geq 0 \}, \tag{IP}$$

with integral data $A \in \mathbf{Z}^{m\times n}$, $b \in \mathbf{Z}^m$, and $c \in \mathbf{Z}^n$.

The method does neither use any cutting planes nor enumeration techniques. It performs lattice operations only and, hence, explicitly relies on the discrete structure of the feasible set $\mathscr{F}$. It is an exact algorithm that starts off with a feasible solution $x \in \mathscr{F}$. In each major step the algorithm either returns an augmenting direction that is applicable at x and yields a new feasible point with better objective function value or provides a proof that the point x maximizes c. The algorithm iteratively substitutes one column by columns that correspond to irreducible solutions of a system of linear diophantine inequalities. Computational results will be given on some instances of the MIPLIB with several hundred variables to support the hypothesis that various integer programs can efficiently be solved without cutting planes.

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 The Integral Basis Method

Let us begin with an informal description of the major steps of the integral basis method. Starting with an integral feasible solution x^0 we want to detect an augmenting direction that is applicable at x^0 or provide a proof that x^0 is optimal. To achieve this we derive in a first step a system of linear inequalities such that x^0 becomes a basic feasible solution of this system. This can be trivially achieved, if $\mathscr{F}$ is contained in the unit cube. In the general integer case we might have to add an additional column to the original system to turn x^0 into a basic feasible solution, see [FSW99]. Once x^0 has been made a basic feasible solution of a system describing $\mathscr{F}$, we make use of the simplex method for performing our task. Clearly, if the reduced cost of all the non-basic variables are non-positive, we have a certificate that x^0 is optimal. Suppose that this is not the case. Then there exist non-basic variables in the current tableau with positive reduced cost. The usual simplex method would then perform a pivot operation on a column with positive reduced cost. This is of course not feasible in an integer setting because in general after the execution of a simplex pivot the new basic feasible solution is no longer integral. Ways to overcome this difficulty have been proposed by Gomory [Gom63] and Young [You68], see also [Hu69].

The procedure is to generate a Chvátal–Gomory cut from one of the legitimate pivot rows in the simplex tableau such that the coefficient of the former pivot element attains the value one in the cut inequality and, secondly, that the cut itself will become a legitimate pivot row. Performing a simplex pivot operation on the selected pivot column and the new row corresponding to the cut will guarantee that the new basic solution (after the pivot operation has been performed) is integral, basic and feasible. This approach has been suggested by Young [You68]. It is in fact a modification of Gomory's all-integer algorithm in the sense that Gomory's algorithm always maintains a dual feasible integral solution by applying the dual simplex method. The approach of Young, however, performs primal simplex steps and works during the course of the procedure with a primal feasible integral solution.

To the best of our knowledge both of these approaches have computationally never proven to be effective. Although both algorithms can be turned into finite procedures, they tend to require extremely many pivot operations until they terminate. Moreover, there are various numerical problems when trying to make this machinery work in practice, see also Nemhauser and Wolsey [NW88].

The integral basis method that we introduce below refrains from adding cuts, but replaces the cutting step by a step in which we manipulate the columns of the given system. Once again let us assume that a basic feasible integral solution x^0 of an appropriate simplex tableau is given. Let B and N denote the index sets of basic and non-basic variables, respectively, $\bar{c}$ the vector of reduced cost and $\bar{A}_N = A_B^{-1}A_N$, $\bar{b} = A_B^{-1}b \geq 0$. Suppose there exist non-basic variables in the current tableau with positive reduced cost. Select a column with positive reduced cost. We may assume that if we perform a pivot operation on this column, then the new basic feasible solution will become fractional. Even more, we may assume that the pivot ratio is strictly less than one, that is, this column on its own is not applicable at x^0. Let us denote the index of this column by j.

Note that the basic part of every possible augmentation vector at x^0 is a non-negative integer combination of the non-basic columns, that is, the non-basic columns form a (non-negative) *integral basis* for the set of all augmentation vectors. Therefore, every augmentation vector that involves the pivot column with index j corresponds to a solution z with $z_j > 0$ of the system

$$\bar{A}_N z \leq \bar{b}, \quad z \in \mathbf{Z}_+^N. \tag{R}$$

Our algorithm computes a set $I(j)$ of solutions z to (R) with $z_j > 0$ such that the columns $\bar{A}_N z$ for $z \in I(j)$ complete the tableau obtained by deleting column j w.r.t. the integral basis property. Thus we arrive at a strengthened IP formulation. In fact, we choose $I(j)$ as the set of certain "irreducible" solutions to (R); the concept of irreducibility used here will be made explicit in section 3.

In other words, the algorithm iteratively replaces one integrally non-applicable column by other columns, maintaining the integral basis property of the tableau as an invariant; hence, the name *integral basis method*.

Algorithm 2.1. (The integral basis method)
Input. $x^0 \in \mathscr{F}$, $c \in \mathbf{Z}^n$.
Output. "Optimal," if x^0 maximizes c; otherwise, $t \in \mathbf{Z}^n$ such that $c^\top t > 0$ and $x^0 + t \in \mathscr{F}$.

(1) Make x^0 a basic feasible solution of a system describing $\mathscr{F}$.

(2) Compute reduced cost $\bar{c}_i$ for all $i \in N$. Determine tableau data $\bar{A}_N = A_B^{-1} A_N$ and $\bar{b} = A_B^{-1} b$. Set $N_+ := \{\, i \in N : \bar{c}_i > 0 \,\}$.

(3) While $N_+ \neq \varnothing$ perform the following steps:

  (3.1) Select $j \in N_+$.

  (3.2) If $\{\, i \in \{1,\dots,m\} : \bar{A}_{ij} \not\leq \bar{b}_i \,\} = \varnothing$, return the augmenting vector $t \in \mathbf{Z}^n$ that corresponds to the non-basic column $\bar{A}_{\cdot j}$ as output of the algorithm.

  (3.3) Determine the set $I(j)$ of all irreducible solutions z to the system $\bar{A}_N z \leq \bar{b}$, $z \in \mathbf{Z}_+^N$ which satisfy $z_j > 0$.

  (3.4) Delete column j from the current tableau and add the columns $\bar{A}_N z$ for $z \in I(j)$ to the tableau, updating N, $\bar{A}_N$, $\bar{c}$, and N_+.

(4) Return "Optimal."

We demonstrate below on an example how the integral basis method works.

Example 2.2. Consider the basic feasible solution

$$(y_1, y_2, \bar{x}_1, x_1, \bar{x}_2, x_2, \bar{x}_3, x_3) = (1,2,1,0,1,0,1,0),$$

of the system

$$\begin{array}{llllllllll}
\max & & & & & -5x_1 & -3x_2 & +4x_3 & & \\
\text{s.t.} & y_1 & & & & -x_1 & -2x_2 & +2x_3 & = & 1 \\
 & & y_2 & & & -2x_1 & +2x_2 & +x_3 & = & 2 \\
 & & & \bar{x}_1 & & +x_1 & & & = & 1 \\
 & & & & \bar{x}_2 & & +x_2 & & = & 1 \\
 & & & & \bar{x}_3 & & & +x_3 & = & 1 \\
 & \multicolumn{9}{l}{x_i, \bar{x}_i, y_i \in \mathbf{Z}_+.}
\end{array}$$

Notice that $N = \{x_1, x_2, x_3\}$. Since $N_+ = \{x_3\}$, the integral basis method will select the variable $j = x_3$ for performing Step (3). Since the coefficient of x_3 in the first row is greater than the right-hand side, column j on its own is not applicable, and the index set in Step (3.2) is non-empty. Therefore, we consider the system (R),

$$-z_1 - 2z_2 + 2z_3 \leq 1, \quad -2z_1 + 2z_2 + z_3 \leq 2, \quad z_1, z_2, z_3 \in \{0,1\}.$$

The solutions to this system are $(1,0,0)$, $(0,1,0)$, $(1,0,1)$, and $(1,1,1)$. Because $(1,1,1)$ is the sum of the solutions $(1,0,1)$ and $(0,1,0)$—it is "reducible"—, we obtain $I(j) = \{(1,0,1)\}$. Hence, variable x_3 can be replaced by a variable corresponding to the sum of the old variable x_3 and the variable x_1. Call this new variable x_3'. Updating the matrix according to Step (3.4) yields a new system,

$$\begin{array}{llllllllll}
\max & & & & & -5x_1 & -3x_2 & -x_3' & & \\
\text{s.t.} & y_1 & & & & -x_1 & -2x_2 & +x_3' & = & 1 \\
 & & y_2 & & & -2x_1 & +2x_2 & -x_3' & = & 2 \\
 & & & \bar{x}_1 & & +x_1 & & +x_3' & = & 1 \\
 & & & & \bar{x}_2 & & +x_2 & & = & 1 \\
 & & & & \bar{x}_3 & & & +x_3' & = & 1 \\
 & \multicolumn{9}{l}{x_i, \bar{x}_i, y_i, x_3' \in \mathbf{Z}_+.}
\end{array}$$

Since the reduced cost of all the non-basic variables are now non-positive, we can assert that the initial basic feasible solution $(y_1, y_2, \bar{x}_1, x_1, \bar{x}_2, x_2, \bar{x}_3, x_3)$ is indeed optimal.

Let us mention that Step (3.3) in our algorithm is similar to a test set approach in integer programming with the following major distinctions. We need to determine irreducible vectors by incorporating the non-basic variables only. Even more, many of the non-basic variables remain untouched because their reduced cost are already non-positive. Secondly, since we start off from a simplex tableau we need to work with only one orthant in the space of variables instead of an exponential number of orthants. In particular, these two reasons make the approach computationally tractable.

3 Irreducible Lattice Points

This section makes precise what we mean by the concept of irreducibility mentioned before and proves correctness and finiteness of the integral basis method. Throughout this section we deal with integral systems of the form

$$S := \{z \in \mathbf{Z}^n_+ : Az \le b\}, \quad A \in \mathbf{Z}^{m\times n}, \quad b \in \mathbf{Z}^m_+.$$

Definition 3.1. (a) A vector $z \in S$ is *reducible* if $z = 0$ or there exist $k \ge 2$ vectors $z^1, \dots, z^k \in S \setminus \{0\}$ and integral multipliers $\lambda_1, \dots, \lambda_k \ge 1$ such that $z = \sum_{i=1}^k \lambda_i z^i$. Otherwise, z is *irreducible*.

(b) An *integral generating set* of S is a subset $S_0 \subseteq S$ such that every vector $z \in S$ 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 $b = 0$) an integral basis is a (minimal) Hilbert basis of the pointed cone $C = \{z \in \mathbf{R}^n_+ : Az \le 0\}$. It is well-known that there is a uniquely determined minimal Hilbert basis of a pointed rational polyhedral cone, which consists of the integral vectors of C that are "irreducible", i.e., they are not the sum of two other non-zero integral vectors of C; see [vdC31]. In the inhomogeneous case, the situation is slightly more complicated because S is not closed w.r.t. non-negative integral combinations.

Example 3.2. Consider the integral system

$$\begin{array}{rcl} -z_1 + z_2 + z_3 & \le & 1 \\ z_1 - z_2 + z_3 & \le & 1 \\ z_1 + z_2 - z_3 & \le & 1 \\ z_1, z_2, z_3 \in \mathbf{Z}_+. \end{array} \tag{1}$$

The unit vectors $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$ are solutions to (1). The vector $(1,1,1)$ is a solution to (1) that is generated by the unit vectors but it is not the sum of *two* other solutions.

It can be shown that existence and uniqueness of a Hilbert basis extend to the more general situation. Consequently, the integral basis method turns out to be a finite procedure.

4 Computational Results and Extensions

We have implemented the integral basis method and applied it to a couple of instances from the MIPLIB. In fact, we did not implement the rudimentary version of it, but a modification. Instead of Step (3.3) in Algorithm 2.1 we select a candidate row i, say, and determine the set $I_i(j)$ of all irreducible solutions with $z_j > 0$ of the system $\bar{A}_{i\cdot} z \leq \bar{b}_i$, $z \in \mathbf{Z}_+^N$. That is, we are working with a discrete relaxation of the original problem. The reason for this modification is that our computational tests have revealed that the number of irreducible solutions to one inequality is usually significantly smaller than the number of irreducible solutions to a system of inequalities. Moreover, we designed a special algorithm for computing irreducible solutions w.r.t. one knapsack constraint that heavily exploits the fact that we work with one constraint only. Working with one row at a time has the price that we need to perform Step (3.3) more often. However, in total, this results in a substantial speed-up.

The current implementation of our algorithm has been designed for 0/1 integer programming problems. Below we discuss its performance on some MIPLIB instances. In order to apply it to general integer instances, some further modifications in the computation of the irreducible solutions in Step (3.3) need to be incorporated into the system.

Table 1: Problems from the MIPLIB

	Columns	Rows	Iterations	Max. Size[a]	Time[b]
`p0033`	33	16	71	54	1
`p0201`	201	133	191	278	48
`p0282`	282	241	34	243	14
`p0548`	548	176	549	1424	610
`lseu`	28	89	329	310	38
`cap6000`	6000	2176	12	134	1592
`mod008`	319	6	636	848	409

[a]Maximum number of non-basic variables during the course of the algorithm.
[b]In seconds, on a Sun Enterprise 450 with 300 MHz.

In our experiments we always start with an optimal solution that we get from running CPLEX, turn it into a basic feasible solution and then try to prove optimality using our algorithm, without resorting to a branching procedure or applying cutting-plane techniques. Table 1 shows our results for some pure 0/1 MIPLIB instances. The second and third column represent the original number of columns and rows in the problem, respectively. The fourth column represents the number of times we enter the loop (3) of Algorithm 2.1. This number reflects how many columns with positive reduced cost had to be substituted. The fifth column indicates the maximal number of columns that arise during the course of the algorithm. This number is indeed an indicator for the speed of the algorithm because our algorithm will have a substantial problem when the number of irreducible solutions that we need to add to the current system in order to replace a column with positive reduced cost is exploding. So far, this situation could be avoided by choosing the pivot rows appropriately, or by relaxing the right-hand side of the constraints. In fact, in some of the test examples the maximum size is quite small, even smaller than the number of columns. This is achieved by solving linear programs maximizing x_i s.t. the feasibility constraints. If the value of this LP is smaller than the upper bound on x_i, then the upper bound on x_i can be decreased, or x_i can even be fixed at zero. The computation times for proving optimality are summarized in the rightmost column.

Note that the whole procedure is without any branching. It is trivial to realize that the running times of the integral basis method can be significantly improved when the method is incorporated into an enumeration scheme. We are however not aware of any pure cutting-plane method that can solve all these instances faster, although cutting planes have been explored extensively in the last thirty years.

The algorithm is very sensitive with the selection of the pivot row on which we perform the computation of the irreducible solutions. To stabilize the performance of the code we always perform an analysis of the rows and estimate the number of irreducible solutions arising from each row before doing the actual calculation. This analysis is important. Indeed, when the number of columns that we need to add to the system is under control, then the algorithm is always pretty fast and robust.

The major difficulty that one faces is a situation where for each candidate column j the set $I(j)$ is very large. Here are two suggestions that are of help in this respect.

1. Relax the right-hand side of a constraint. This increase in the right-hand side can result in a drastic reduction of the number of irreducible solutions that satisfy the relaxed constraint.

2. Using information from a pre-computed data base, it is sometimes possible to compute the number of irreducible 0/1 solutions of a constraint with large support in a compact way. One such example is when the coefficients of the constraint are integer values between 0 and 20, say, then the irreducible solutions to such a system can be read off from the irreducible solutions of an integer knapsack constraint with 20 variables.

Our computational tests have shown that Suggestion 1 is indeed very helpful. So far we have however not tested the effect of Suggestion 2. The current computational results indicate at least that the integral basis method in its current version is promising. For us it was a surprise that the theory of integral bases can be turned into a practical tool. We are sure that this theory deserves further attention, both from the theoretical as well as from the computational point of view.

Acknowledgment. The authors wish to thank Giovanni Rinaldi for many interesting discussions related to this approach. We are also very grateful to Tom Liebling, who has always supported the idea of developing an algorithm for integer programming based on ideas related to integral bases.

References

[ABH$^+$00] K. Aardal, R. E. Bixby, C. A. J. Hurkens, A. K. Lenstra, and J. W. Smeltink, *Market split and basis reduction: towards a solution of the Cornuéjols-Dawande instances*, INFORMS J. Comput. **12** (2000), no. 3, 192–202.

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

[FSW99] R. T. Firla, B. Spille, and R. Weismantel, *A primal analogue of cutting plane algorithms*, Preprint No. 22, Universität Magdeburg, 1999.

[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.

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

[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.

[NW88] G. L. Nemhauser and L. Wolsey, *Integer and combinatorial optimization*, Wiley, Chichester, 1988.

[vdC31] J. G. van der Corput, *Über Systeme von linear-homogenen Gleichungen und Ungleichungen*, Proceedings Koninklijke Akademie van Wetenschappen te Amsterdam, vol. 34, 1931, pp. 368–371.

[Was00] A. Wassermann, *Attacking the market split problem with lattice basis reduction*, Preprint, Universität Bayreuth, 2000.

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