Some Tapas of Computer Algebra

Editors:

A.M. Cohen, H. Cuypers, H. Sterk

Eindhoven University of Technology

The Netherlands

Table of Contents

Gröbner bases and integer programming

Günter M. Ziegler

MA 6-1, Department of Mathematics, Technical University Berlin
10623 Berlin, Germany

1. Introduction

'Integer programming' is a basic mathematical problem, of central importance in Optimization and Operations Research. While a systematic body of theory has been developed for it in the last fifty years [14], it has been realized only very recently, first by Conti & Traverso [5], that the Buchberger algorithm (cf. Chapter 1) provides a solution strategy for integer programming problems, in particular in the case of families of programs with 'varying right-hand side.'

Section 2. gives a short introduction to 'what integer programming is about'. Then we discuss a basic version of the Buchberger algorithm applied to integer programming (Section 3.). In Section 4. we show how, in the special case of the binomial ideals that arise from integer programs, the Buchberger algorithm can be formulated as a combinatorial-geometric algorithm that operates on lattice vectors. A surprisingly simple variation of the Buchberger algorithm for integer programming is presented in Section 5..

The problems to this chapter treat the relation between lattice vectors and binomials, and the 'Gröbner basis of a lattice', in more detail.

2. 'What is Integer Programming'

A *polyhedron* P is any intersection $P := \{\mathbf{x} \in \mathbb{R}^n : A\mathbf{x} \leq \mathbf{b}\}$ of closed halfspaces, in some $\mathbb{R}^n$. Here $A \in \mathbb{R}^{m \times n}$ is a matrix, while $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^m$ are column vectors. A bounded polyhedron is a *polytope*.

Given any linear function $\mathbf{x} \longmapsto \mathbf{c}^\top \mathbf{x}$ on $\mathbb{R}^n$, the *linear programming problem* is to determine a point $\mathbf{x}$ in the polyhedron P for which the linear function $\mathbf{c}^\top \mathbf{x}$ is minimal. The points in P are called *feasible*. If P is a non-empty polytope, then the existence of an *optimal* point $\mathbf{x}_0$ is guaranteed. Furthermore, if the linear function $\mathbf{c}^\top$ is *generic* (with respect to the inequalities that define P), then the optimal point $\mathbf{x}_0$ is unique.

It is both useful and customary to deal only with a restricted class of rational polyhedra in some 'standard form.' That is, one considers, for example, only polyhedra of the form

$$P \quad = \quad \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0}\},$$

with the additional assumptions that A and $\mathbf{b}$ have only integral coordinates. This is not much of a loss of generality: non-rational data do not occur 'in nature,' and general polyhedra can be represented by polyhedra in this *equality standard form* by suitable coordinate transformation, introduction of slack variables, multiplication by common denominators, etc.

Linear programming is a well-established theory. In a practical sense the linear programming problem is 'solved': after essential progress in the eighties (including the construction of polynomial algorithms such as the 'ellipsoid method', the rise of 'interior point methods' that are both theoretically polynomial and practically competitive, and considerable refinements of the classical 'simplex method'), we have now codes available (such as CPLEX, by Bob Bixby) which will solve virtually every linear program that you can cook up and store in a computer.

The situation is vastly different for *integer programming*, the task to compute an *integral* vector in P that minimizes $\mathbf{c}^\top\mathbf{x}$. In this situation, the points in $P \cap \mathbb{Z}^n$ are called *feasible*. A basic result of polyhedral theory states that the convex hull

$$P_I \quad = \quad \operatorname{conv}\{\mathbf{x} \in \mathbb{Z}^n \mid A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0}\},$$

is also a polyhedron with finitely many facets. However, the facet-defining inequalities for the polyhedron P_I are not usually known in general (otherwise we would be done by linear programming), they are hard to determine, and their number may be huge. Thus the integer programming problem — of essential importance in many practical applications — is still a great challenge. It is still not difficult to produce integer programs of reasonable size ($m = 20$ and $n = 30$, say) that none of the currently available codes can solve. Here *solution*, as we will see, really comprises two separate tasks, both of them non-trivial: to *find* an optimal solution, and to *prove* that it is optimal.

While several 'good', or at least 'interesting', solution strategies exist, integer programming is still a difficult problem. The object of this chapter is an introduction to one (relatively new) such strategy: the construction of *test sets* for integer programming via the Buchberger algorithm. In a basic version, we will present this in the following section.

3. A Buchberger Algorithm for Integer Programming

For the following exposition, we consider a family of integer programs for which $A \in \mathbb{N}^{m\times n}$ is a fixed, non-negative integer matrix. The right hand side vector, $\mathbf{b} \in \mathbb{N}^m$, is considered as variable. Thus we consider the integer polyhedra

$$P_I(\mathbf{b}) \quad := \quad \operatorname{conv}\{\mathbf{x} \in \mathbb{N}^n \mid A\mathbf{x} = \mathbf{b}\}.$$

Now let $\mathbf{c} \neq \mathbf{0}$ be a (fixed) linear objective function. To make life easier, we also assume that the linear program

$$LP(\mathbf{b}) \qquad \min\ \mathbf{c}^\top\mathbf{x} : \quad A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0}$$

is bounded for every $\mathbf{b}$. This is not much of a restriction: for example, it is satisfied if $\mathbf{c} \geq \mathbf{0}$. In particular, it implies that the integer programs

$$\mathrm{IP}(\mathbf{b}) \qquad \min\ \mathbf{c}^\top\mathbf{x} : \quad A\mathbf{x} = \mathbf{b},\ \mathbf{x} \in \mathbb{N}^n$$

are bounded.

In the following, we also need that the objective function is generic. To enforce this, we choose a term order $\prec$ (lexicographic, for example) that can be used as a tie breaker, and define

$$\mathbf{x} \prec_{\mathbf{c}} \mathbf{y} \qquad :\Longleftrightarrow \qquad \begin{cases} \mathbf{c}^\top\mathbf{x} < \mathbf{c}^\top\mathbf{y}, & \text{or} \\ \mathbf{c}^\top\mathbf{x} = \mathbf{c}^\top\mathbf{y} & \text{and } \mathbf{x} \prec \mathbf{y}. \end{cases}$$

We will use a total order such as $\prec_{\mathbf{c}}$, derived from the "original" objective function, as the input for the integer programming algorithms of this chapter. Note that $\prec_{\mathbf{c}}$ is a term order (in the sense of Gröbner basis theory) if and only if $\mathbf{c} \geq \mathbf{0}$.

With the above assumptions, we get that each of the integer programs

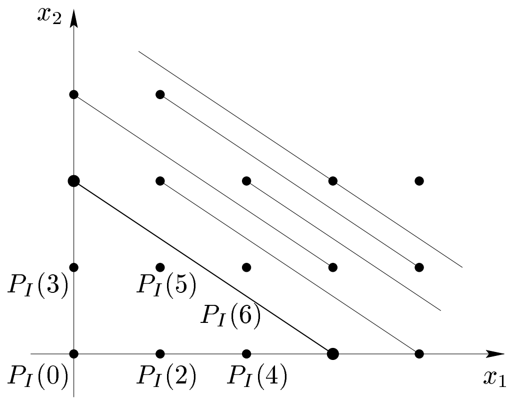
$$\mathrm{IP}(\mathbf{b}) \qquad \min\nolimits_{\prec_{\mathbf{c}}} \{\mathbf{x} \in \mathbb{N}^n \mid A\mathbf{x} = \mathbf{b}\}$$

either has no feasible points, or it has a unique optimal solution. (The optimal solution, but not its objective function value, will in general depend on the tie breaker used to define $\prec_{\mathbf{c}}$.) We use

$$\mathrm{IP}_{A,\mathbf{c}}$$

to denote the whole family of these integer programs, with fixed A and $\mathbf{c}$, but varying right-hand side $\mathbf{b}$. The key idea is to consider this whole class of programs simultaneously.

Example 3.1. For $n = 2$ we can draw figures such as the following, which is obtained for $A = (2\ 3)$, where $P_I(b_1)$ denotes the convex hull of the feasible (integer) points of $\mathrm{IP}(\mathbf{b})$, for $\mathbf{b} = (b_1)$. We get that $P_I(0) = \emptyset$, the set $P_I(b_1)$ is a point for $b_1 \in \{0, 2, 3, 4, 5, 7\}$, while $P_I(b_1)$ is a (bounded) line segment for $b_1 = 6$ and for $b_1 \geq 8$.



We call a vector $\mathbf{x} \in \mathbb{N}^n$ *non-optimal* if there is another vector $\mathbf{y} \in \mathbb{N}^n$ that is feasible for the same right-hand side (that is, $A\mathbf{x} = A\mathbf{y}$), and that is "better" than $\mathbf{x}$ in the sense that $\mathbf{y} \prec_{\mathbf{c}} \mathbf{x}$. A simple but crucial observation is that if $\mathbf{x}$ is non-optimal, and if $\mathbf{x}' \geq \mathbf{x}$ is componentwise larger than $\mathbf{x}$, then $\mathbf{x}'$ is non-optimal as well. Thus the Gordan-Dickson lemma, according to which every subset of $\mathbb{N}^n$ has only finitely many minimal elements (for the componentwise order), yields the following key fact.

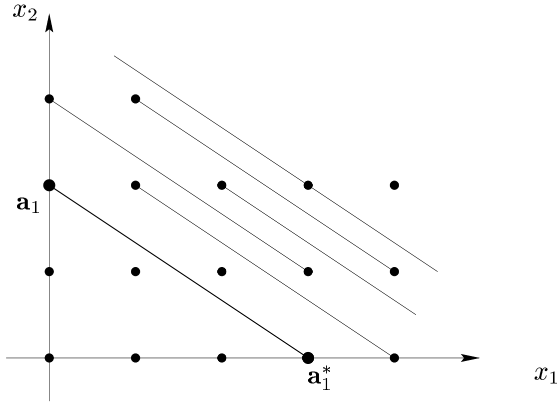
Lemma 3.2 (Minimal non-optimal points). *The minimal (with respect to inclusion) set of vectors $\mathbf{a}_i \in \mathbb{N}^n$ such that*

$$\{\mathbf{x} \in \mathbb{N}^n \mid \mathbf{x} \textit{ non-optimal}\} \quad = \quad \{\mathbf{x} \in \mathbb{N}^n \mid \mathbf{x} \geq \mathbf{a}_i \textit{ for some } i\}$$

is unique and finite, and thus it can be written as $\{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_t\}$.

This lemma is important since it yields the indexing set for both the minimal test set for $\mathrm{IP}_{A,\mathbf{c}}$, as follows, and for the Gröbner basis of the associated ideal, see Theorem 3.7 below.

Example 3.3. For $n \leq 2$ we necessarily have $t = 1$ (Exercise!). Our figure depicts the situation for $n = 2$, $A = (2\ 3)$, $\mathbf{b} = (6)$, and $\mathbf{c} = (1\ 4)$. The shaded region covers all the non-optimal integral points for this family of programs.



Here both $\mathbf{a}_1$ and $\mathbf{a}_1^*$ are contained in $P_I(\mathbf{b})$, for $\mathbf{b} = A\mathbf{a}_1 = A\mathbf{a}_1^* = (6)$.

Definition 3.4. A subset $\mathcal{G}_{\mathbf{c}} \subseteq \mathbb{Z}^n$ is a *test set* for the family $\mathrm{IP}_{A,\mathbf{c}}$ of integer programs if and only if

$A\mathbf{g} = \mathbf{0}$ for all $\mathbf{g} \in \mathcal{G}_{\mathbf{c}}$,

$\mathbf{g} \succ_{\mathbf{c}} \mathbf{0}$ for all $\mathbf{g} \in \mathcal{G}_{\mathbf{c}}$, and

for every non-optimal point $\mathbf{x} \in \mathbb{N}^n$, there is some $\mathbf{g} \in \mathcal{G}_{\mathbf{c}}$ with $\mathbf{x} - \mathbf{g} \geq \mathbf{0}$.

The definition of a test set immediately provides us with the following algorithm for integer programming — once we have a feasible point to start with, and we know how to compute a test set. One might note the similarity to 'improvement heuristics', such as the ones used to find good solutions to traveling salesman problems.

Algorithm 3.5. To solve programs in a family $\mathrm{IP}_{A,\mathbf{c}}$:

Input a test set $\mathcal{G}_\mathbf{c}$ for the family $\mathrm{IP}_{A,\mathbf{c}}$
and some $\mathbf{x} \in \mathbb{N}^n$ $\quad$ ($\mathbf{x}$ is feasible for $\mathrm{IP}(A\mathbf{x})$)
Repeat find $\mathbf{g} \in \mathcal{G}_\mathbf{c}$ such that $\mathbf{x} - \mathbf{g} \geq \mathbf{0}$,
$\mathbf{x} \longrightarrow \mathbf{x} - \mathbf{g}$
Until optimal.

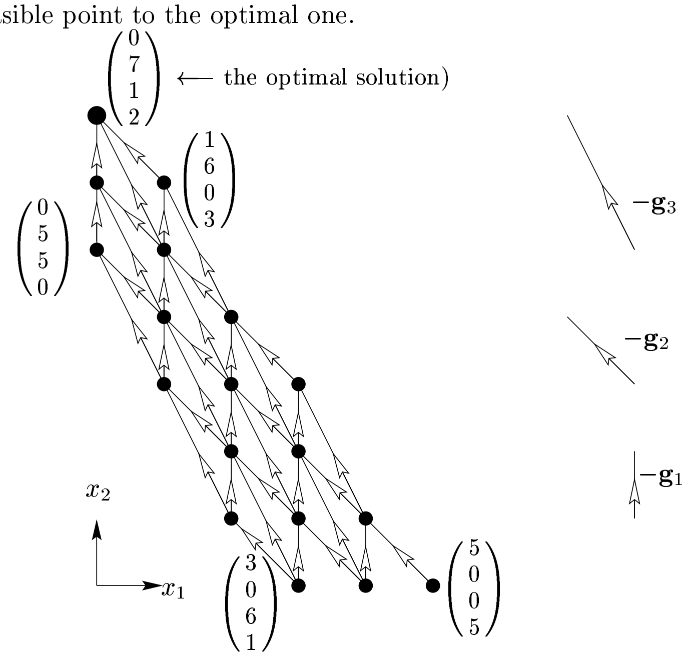
Example 3.6. (Thomas [18])
For the family of integer programming problems that are given by

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{pmatrix} \qquad \text{and} \qquad \mathbf{c}^\top = (1\ 3\ 14\ 17)$$

a minimal test set consists of just three vectors,

$$\mathbf{g}_1 = \begin{pmatrix} 0 \\ -1 \\ 2 \\ -1 \end{pmatrix}, \qquad \mathbf{g}_2 = \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix}, \qquad \mathbf{g}_3 = \begin{pmatrix} 1 \\ 2 \\ 1 \\ 0 \end{pmatrix},$$

and these three are sufficient to do the optimization for an arbitrary right-hand side. For $\mathbf{b} = (10\ 15)^\top$, we obtain the situation displayed in the figure, which shows the projection to the (x_1, x_2)-plane: there are 18 feasible points, and the three test set vectors are sufficient (and necessary!) to get you from any feasible point to the optimal one.



Theorem 3.7. (Thomas [18, Cor. 2.1.10]) *The unique minimal test set for the family* $\mathrm{IP}_{A,\mathbf{c}}$ *of integer programs is given by*

$$\begin{aligned}\mathcal{G}_{\mathbf{c}} \quad := \quad \big\{\mathbf{a}_i - \mathbf{a}_i^* \;\big|\; & \mathbf{a}_i \in \min_{\leq}\{\mathbf{x} \in \mathbb{N}^n \textit{ non-optimal}\}, \\ & \mathbf{a}_i^* \textit{ optimal for } \mathrm{IP}(A\mathbf{a}_i^*),\textit{ where } A\mathbf{a}_i = A\mathbf{a}_i^*\big\}.\end{aligned}$$

The connection from Integer Programming to Gröbner Basis Theory can now be made by observing that the minimal test of Theorem 3.7 corresponds to the reduced Gröbner basis of the binomial ideal

$$I_A \quad := \quad \big\langle \mathbf{X}^{\mathbf{a}^+} - \mathbf{X}^{\mathbf{a}^-} \mid A\mathbf{a} = \mathbf{0},\ \mathbf{a} \in \mathbb{Z}^n \big\rangle$$

with respect to the term order $\prec_{\mathbf{c}}$. (Here $\mathbf{a}^+$ is our shorthand for the vector we obtain by replacing all negative coordinates of $\mathbf{a}$ by zero. Similarly, we use $\mathbf{a}^- := (-\mathbf{a})^+$, so that we have $\mathbf{a} = \mathbf{a}^+ - \mathbf{a}^-$, with $\mathbf{a}^+, \mathbf{a}^- \geq \mathbf{0}$. As is customary, $\mathbf{X}^{\mathbf{a}}$ is a notation for $X_1^{a_1} \cdots X_n^{a_n}$, etc.)

The information that $\mathcal{G}_{\mathbf{c}}$ is a Gröbner basis of I_A is not terribly helpful, since in general we do not know a generating set for the ideal — so we can't compute a Gröbner basis, either. Thus we use a small dirty trick: we create a larger integer program, which has an obvious integer feasible point, and for which the ideal has a nice generating set to start from. (Versions of this trick appear both in algebra, see [6, Sect. 3.3], and in linear programming, where slack variables are introduced to obtain "Phase I" problems that have feasible starting basis, see the "big-M method" in [14, Sect. 11.2].)

For this, we consider the 'extended integer programs'

$$\mathrm{EIP}(\mathbf{b}) \qquad \min\nolimits_{\prec_{(M\mathbf{1},\mathbf{c})}} \Big\{ \begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix} \in \mathbb{N}^{n+m} \;\Big|\; I_m\mathbf{y} + A\mathbf{x} = \mathbf{b} \Big\}$$

where $M \in \mathbb{N}$ is a large constant, I_m is the $m \times m$ identity matrix, and $\mathbf{1}$ denotes the vector of all ones. We use

$$\mathrm{EIP}_{A,\mathbf{c}}$$

to denote the whole family of these integer programs, with fixed A and $\mathbf{c}$, but varying right-hand side $\mathbf{b}$. What have we gained? On the one hand, all of the programs $\mathrm{EIP}(\mathbf{b})$ are feasible: they have the obvious solution $\mathbf{x} = \mathbf{0}$, $\mathbf{y} = \mathbf{b}$. However, an optimal solution will satisfy $\mathbf{y} = \mathbf{0}$, $\mathbf{x} = \mathbf{x}_0$ if the program $\mathrm{IP}(\mathbf{b})$ is feasible, because M was chosen to be *very* large. If $\mathrm{IP}(\mathbf{b})$ is infeasible, then the extended program $\mathrm{EIP}(\mathbf{b})$ has an optimal solution with $\mathbf{y} \neq \mathbf{0}$. The binomial ideal that corresponds to $\mathrm{EIP}_{A,\mathbf{c}}$ *does* have a nice generating set that we can use to start a Buchberger algorithm.

Proposition 3.8. (Conti & Traverso [5]) *The ideal*

$$I_{(I_m,A)} \quad := \quad \Big\langle \mathbf{Y}^{\mathbf{a}_1^+}\mathbf{X}^{\mathbf{a}_2^+} - \mathbf{Y}^{\mathbf{a}_1^-}\mathbf{X}^{\mathbf{a}_2^-} \;\Big|\; (I_m, A)\begin{pmatrix}\mathbf{a}_1\\ \mathbf{a}_2\end{pmatrix} = \mathbf{0},\ \begin{pmatrix}\mathbf{a}_1\\ \mathbf{a}_2\end{pmatrix} \in \mathbb{Z}^{m+n} \Big\rangle$$

is generated by the binomials

$$\mathbf{Y}^{A\mathbf{e}_j} - X_j \qquad \textit{for } 1 \leq j \leq n.$$

The reduced Gröbner basis of $I_{(I_m,A)}$ *with respect to the term order* $\prec_{(M\mathbf{1},\mathbf{c})}$ *yields the minimal test set* $\mathcal{G}_{(M\mathbf{1},\mathbf{c})}$ *for the family* $\mathrm{EIP}_{A,\mathbf{c}}$*, via the canonical bijection*

$$\begin{pmatrix}\mathbf{a}_1\\ \mathbf{a}_2\end{pmatrix} \quad\longleftrightarrow\quad \mathbf{Y}^{\mathbf{a}_1^+}\mathbf{X}^{\mathbf{a}_2^+} \;-\; \mathbf{Y}^{\mathbf{a}_1^-}\mathbf{X}^{\mathbf{a}_2^-}.$$

The binomials $\mathbf{Y}^{A\mathbf{e}_j} - X_i$ form a Gröbner basis for the ideal that they generate, for a lexicographic term order with $X_i \succ_{\mathrm{lex}} Y_j$. To see that this is the whole ideal $I_{(I_m,A)}$, start with any binomial in $I_{(I_m,A)}$, reduce it to a binomial that contains no X-variables using the generators of $I_{(I_m,A)}$, and then conclude that you have arrived at the zero binomial, using Exercise 6.4.

Putting things together, we have an extremely simple algorithm for integer programming: we 'only' need to compute a reduced Gröbner basis with respect to the term order $\prec_{(M\mathbf{1},\mathbf{c})}$, and then use this with the above algorithm to solve the extended programs $\mathrm{EIP}_{A,\mathbf{c}}$.

Algorithm 3.9 (Integer programming via Buchberger's algorithm). The following procedure solves the extended integer program

$$\mathrm{IP}(\mathbf{b}) \qquad\qquad \min_{\prec_{\mathbf{c}}} \{\mathbf{x} \in \mathbb{N}^n \,|\, A\mathbf{x} = \mathbf{b}\}$$

for $A \in \mathbb{N}^{m\times n}$, $\mathbf{b} \in \mathbb{N}^m$, $\mathbf{c} \in \mathbb{N}^n$.

First Phase: Compute a test set

Input A, $\mathbf{c}$
Compute the reduced Gröbner basis $\mathcal{G}_{(M\mathbf{1},\mathbf{c})}$ for $I := \langle \mathbf{Y}^{A\mathbf{e}_j} - X_j \rangle$,
Output the test set $\mathcal{G}_{(M\mathbf{1},\mathbf{c})}$.

Second Phase: Reduction

Input $\mathcal{G}_{(M\mathbf{1},\mathbf{c})}$, $\mathbf{b}$
Reduce the monomial $\mathbf{Y}^{\mathbf{b}}$ with respect to $\mathcal{G}_{(M\mathbf{1},\mathbf{c})}$, get $\mathbf{Y}^{\mathbf{a}_1}\mathbf{X}^{\mathbf{a}_2}$.
Output If $\mathbf{a}_1 \neq \mathbf{0}$, return 'infeasible'.
If $\mathbf{a}_1 = \mathbf{0}$, return '$\mathbf{x}_0 = \mathbf{a}_2$ is optimal'.

While the first phase of this algorithm (computation of a Gröbner basis) amounts to hard work, the second one should typically be quite easy & fast (if we manage to efficiently search the Gröbner basis, which may be huge). But even if we cannot obtain a complete Gröbner basis from the first phase, then we can still use any partial basis to reduce the monomial $\mathbf{Y}^{\mathbf{b}}$, which may yield a feasible, or even the optimal, point.

However, we are still making quite a detour in Algorithm 3.9: one can formulate the Buchberger algorithm so that it operates directly on lattice points (no ideals, binomials, etc. involved!) This geometric formulation (given in the next section) yields an extremely simple algorithm for integer programming: also one that is very easy to implement! The basic version is not terribly efficient: but we will discuss a few basic ideas about 'how to speed it up'.

4. A Geometric Buchberger Algorithm

The Buchberger algorithm for integer programming is a special case of the general Buchberger algorithm. However, there is a lot of special features in the special situation of 'toric ideals' that we are dealing with here. In particular, one only has to deal with 'binomials with disjoint supports': thus we can get an entirely geometric formulation of the algorithm, dealing with lattice vectors in $\mathbb{Z}^n$ — no polynomials whatsoever appear. This simplifies the data structures considerably!

The translation process may be done as follows. The first observation is that, by definition

$$I_{(I_m,A)} \quad = \quad \langle \mathbf{Y}^{A\mathbf{e}_j} - X_j : 1 \leq j \leq n \rangle$$

is a *binomial ideal*, an ideal generated by binomials. Any S-pair of two binomials is a binomial (see Problem 6.3 for a sharper version of this fact). Also the reduction of binomials by binomials leads to binomials. Thus the entire Buchberger process produces only binomials during its lifetime, and any reduced Gröbner basis of $I_{(I_m,A)}$ consists of binomials.

As the second step in our translation process we notice that, whenever a binomial appears in the computation whose two terms have a common factor, we may remove that factor, and the corresponding 'reduced' binomial is also contained in $I_{(I_m,A)}$. This follows from the stronger statement that I is a 'lattice ideal', in the following way.

A *lattice* is a discrete additive subgroup $\mathcal{L} \subseteq \mathbb{Z}^n$, that is, the set of all integral linear combinations of a finite set of linearly independent vectors in $\mathbb{R}^n$. With every lattice $\mathcal{L} \subseteq \mathbb{Z}^n$ we associate associate the *lattice ideal*

$$I_{\mathcal{L}} \quad := \quad \langle \mathbf{X}^{\mathbf{a}^+} - \mathbf{X}^{\mathbf{a}^-} \ \mid \ \mathbf{a} \in \mathcal{L} \rangle.$$

Thus, Proposition 3.8 shows that $I_{(I_m,A)}$ is a lattice ideal. Note that in the definition of $I_{\mathcal{L}}$, we can replace $\mathcal{L}$ by the subset of all lattice vectors that are positive with respect to the ordering $\succ$ that we consider, that is, by

$$\mathcal{L}^{\succ \mathbf{0}} \ := \ \{\mathbf{a} \in \mathcal{L} : \mathbf{a} \succ \mathbf{0}\}.$$

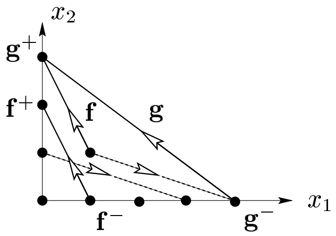
Thus the Buchberger algorithm can immediately remove the common factors from all binomials that it produces. (In particular, any reduced Gröbner basis of I contains only binomials of the form $\mathbf{X}^{\mathbf{a}^+} - \mathbf{X}^{\mathbf{a}^-}$ with $\mathbf{a} \in \mathcal{L}^{\succ \mathbf{0}}$.) Thus the Buchberger algorithm can really be formulated as a geometric algorithm operating on lattice vectors. So we get the following two algorithms to compute the *reduced Gröbner basis of a lattice*, that is, the finite subset $\mathcal{G} \subseteq \mathcal{L}^{\succ \mathbf{0}}$ that corresponds to the reduced Gröbner basis of $I_{\mathcal{L}}$. The assumption for this is that we know a 'good' generating set for the lattice, i.e., a subset of the lattice corresponding to a set of binomials that generates $I_{\mathcal{L}}$.

Algorithm 4.1 (Reduction). The following algorithm computes the reduction of a vector $\mathbf{f} \in \mathbb{Z}^{m+n}$ by a set $\mathcal{G}$ of integer vectors.

(Compare it to the algorithm *Reduce* of Chapter 1!)

Input $\mathcal{G} \subseteq \mathcal{L}^{\succ 0}$, $\mathbf{f} \succ \mathbf{0}$.
Repeat
If there is some $\mathbf{g} \in \mathcal{G}$ with $\mathbf{g}^+ \leq \mathbf{f}^+$, then replace $\mathbf{f}$ by $\pm(\mathbf{f} - \mathbf{g}) \succ \mathbf{0}$.
If there is some $\mathbf{g} \in \mathcal{G}$ with $\mathbf{g}^+ \leq \mathbf{f}^-$, then replace $\mathbf{f}$ by $\mathbf{f} + \mathbf{g}$.
Output $\overline{\mathbf{f}} := \mathbf{f}$.

Our figure illustrates the first case in the reduction algorithm, where we have $\mathbf{f}^+ \leq \mathbf{g}^+$, and the reduced vector arises as a difference. (Lattice vectors such as $\mathbf{f}$ can be drawn with the head at $\mathbf{f}^+$ and the tail at $\mathbf{f}^-$.)



You should check that this reduction process corresponds to the reduction of $\mathbf{X}^{\mathbf{f}^+} - \mathbf{X}^{\mathbf{f}^-} = X_2^3 - X_1^4$ by $\mathbf{X}^{\mathbf{g}^+} - \mathbf{X}^{\mathbf{g}^-} = X_2^2 - X_1$, where the resulting polynomial $X_1^4 - X_1X_2$ has a common factor X_1, whose removal corresponds to a translation of the dotted vector.

Algorithm 4.2 (Buchberger algorithm on lattice vectors).
The following algorithm computes the reduced Gröbner basis of the lattice $\mathcal{L}$, for a fixed term order $\succ$.

First Step: Construct a Gröbner basis

Input A basis $\{\mathbf{a}_1, \ldots, \mathbf{a}_n\} \subseteq \mathcal{L}$ of the lattice $\mathcal{L}$ such that the binomials $\mathbf{X}^{\mathbf{a}^+} - \mathbf{X}^{\mathbf{a}^-}$ generate $I_{\mathcal{L}}$. (See Problem 6.5!)
Set $\mathcal{G}_{old} := \emptyset$, $\mathcal{G} := \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$
Repeat While $\mathcal{G}_{old} \neq \mathcal{G}$, repeat the following steps
$\mathcal{G}_{old} := \mathcal{G}$
(S-pairs) construct the pairs $\mathbf{g} := \mathbf{a} - \mathbf{a}' \succ \mathbf{0}$ with $\mathbf{a}, \mathbf{a}' \in \mathcal{G}$.
(Reduction) reduce the vectors $\mathbf{g}$ by the vectors in $\mathcal{G}_{old}$. If $\overline{\mathbf{g}} \neq \mathbf{0}$, set $\mathcal{G} := \mathcal{G} \cup \overline{\mathbf{g}}$.

Second Step: Construct a minimal Gröbner basis

Repeat If for some $\mathbf{g} \in \mathcal{G}$ the point $\mathbf{g}^+$ can be reduced by some $\mathbf{g}' \in \mathcal{G} \backslash \mathbf{g}$, then delete $\mathbf{g}$ from $\mathcal{G}$.

Third Step: Construct the reduced Gröbner basis

Repeat If for some $\mathbf{g} \in \mathcal{G}$ the point $\mathbf{g}^-$ can be reduced by some $\mathbf{g}' \in \mathcal{G} \backslash \mathbf{g}$, then replace $\mathbf{g}$ by the corresponding reduced vector: $\mathcal{G} := \mathcal{G} \backslash \mathbf{g} \cup \overline{\mathbf{g}}$.
Output $\mathcal{G}_{red} := \mathcal{G}$.

All these operations are easy to visualize (at least in the 2-dimensional situation). They are also easily implemented — just do it. There is also a lot of flexibility: in fact, for a successful implementation it is important to reduce earlier, otherwise the Gröbner bases constructed in the 'First Step' will be too large. See [5, 12, 21] for further ideas about how to make this efficient.

We just remark that the elements that can occur in a reduced Gröbner basis can be characterized geometrically in a different way. The following theorem is due to Sturmfels & Thomas [16].

Theorem 4.3. *The universal Gröbner basis* (*that is, the union of all the reduced Gröbner bases* $\mathcal{G}_{\mathbf{c}}$ *of* $\mathrm{IP}_{A,\mathbf{c}}$*, for all objective functions* $\mathbf{c}$) *consists of all the primitive lattice vectors* $\mathbf{a} \in \mathbb{Z}^n$ (*with* $A\mathbf{a} = \mathbf{0}$*, and* $\frac{1}{\lambda}\mathbf{a} \notin \mathbb{Z}^n$ *for* $\lambda > 1$) *such that* $[\mathbf{a}^+, \mathbf{a}^-]$ *is an edge of the polyhedron*

$$P_I(\mathbf{a}^+) \quad = \quad \mathrm{conv}\{\mathbf{x} \in \mathbb{N}^n \mid A\mathbf{x} = A\mathbf{a}^+\}.$$

This theorem can be applied as well to the extended integer programs EIP_A, where we know how the minimal test sets (Gröbner bases) can be computed via Buchberger's algorithm. Sturmfels & Thomas [16] also have a technique to compute universal Gröbner bases via one single application of a Buchberger algorithm (to a larger problem).

5. A Variant of the Buchberger Algorithm

The following presents a variation of the Buchberger algorithm that may be even more useful for integer programming.

Given a matrix $A \in \mathbb{N}^{m\times n}$, an objective function $\mathbf{c} \in \mathbb{N}^n$, and a right hand side vector $\mathbf{b} \in \mathbb{N}^m$, we denote by $\mathrm{IP}_{A,\mathbf{b},\mathbf{c}}$ the optimization problem

$$\max_{\prec_{\mathbf{c}}} \{\mathbf{x} \in \mathbb{N}^n \mid A\mathbf{x} \leq \mathbf{b},\ \mathbf{0} \leq \mathbf{x} \leq \mathbf{u}\}.$$

This is a special but quite common type of integer program, which we call a problem in *inequality standard form with upper bounds.* See also Exercise 6.1. Again, in order to avoid dealing with degenerate cases we refine the objective function $\mathbf{c}^\top\mathbf{x}$ to get a term order $\prec_{\mathbf{c}}$. A *test set* for a problem of the type $\mathrm{IP}_{A,\mathbf{b},\mathbf{c}}$ is a set $\mathcal{G}$ of vectors $\mathbf{g} \succ_{\mathbf{c}} \mathbf{0}$ such that every non-optimal feasible point can be improved by one of the test set vectors. Algebraically, both $\mathbf{u}$ and $\mathbf{b}$ provide "degree bounds" for Buchberger algorithms. Thus test sets for families of problems of the type $\mathrm{IP}_{A,\mathbf{b},\mathbf{c}}$ correspond to certain truncated Gröbner bases. However, our discussion in the following stays in the elementary geometry setting of [21]; The algebraic picture can be found in [20].

Roughly speaking, a test set, $\mathcal{G}$ say, can be computed as follows. Start with the n unit vectors, i.e., set $\mathcal{G} := \{e_i \mid 1 \leq i \leq n\}$. Iteratively, compute the difference vectors between all pairs of vectors that are in $\mathcal{G}$ and direct each such difference vector such that it is greater than $\mathbf{0}$ with respect to the order. All such difference vectors are added to $\mathcal{G}$, if they are not already in $\mathcal{G}$, and if they are differences of feasible points for $\mathrm{IP}_{A,\mathbf{b},\mathbf{c}}$. The algorithm terminates when no more vectors are added to $\mathcal{G}$.

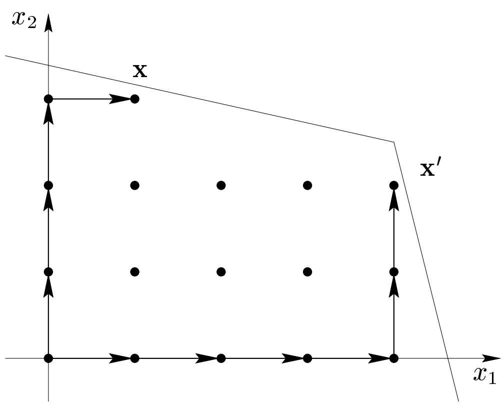
More precisely, the basic algorithm can be formulated as follows:

Algorithm 5.1. To computes a test set for integer programs $\mathrm{IP}_{A,\mathbf{b},\mathbf{c}}$ in inequality standard form with upper bounds,

Input A and $\prec_{\mathbf{c}}$
Initialize Set $\mathcal{G}_{old} := \emptyset$, $\mathcal{G} := \{e_i \mid 1 \leq i \leq n\}$.
While $\mathcal{G}_{old} \neq \mathcal{G}$ perform the following steps:
Set $\mathcal{G}_{old} := \mathcal{G}$.
For all pairs of vectors $\mathbf{v}, \mathbf{w} \in \mathcal{G}$ such that
$$\mathbf{w} \succ_{\mathbf{c}} \mathbf{v}, \quad -\mathbf{b} \leq A(\mathbf{w}-\mathbf{v}) \leq \mathbf{b} \quad \text{and} \quad -\mathbf{u} \leq \mathbf{w}-\mathbf{v} \leq \mathbf{u},$$
set $\mathcal{G} := \mathcal{G} \cup \{\mathbf{w}-\mathbf{v}\}$.

Whenever the loop in this algorithm is executed (except for the last time), a new vector is added to the set $\mathcal{G}_{old}$. Since the number of integral vectors $\mathbf{x}$ satisfying $-\mathbf{u} \leq \mathbf{x} \leq \mathbf{u}$ is bounded by $\prod_{i=1}^{n}(2u_i+1)$, the above algorithm terminates after finitely many steps.

Let us now show that the set $\mathcal{G}$ generated by the above algorithm is a test set for $\mathrm{IP}_{A,\mathbf{b},\mathbf{c}}$. Suppose that $\mathbf{x}$ is a feasible point ($A\mathbf{x} \leq \mathbf{b}$, $\mathbf{0} \leq \mathbf{x} \leq \mathbf{u}$) that cannot be improved by any element in $\mathcal{G}$, and let $\mathbf{x}'$ be a feasible vector with $\mathbf{x}' \succ_{\mathbf{c}} \mathbf{x}$. Then $\mathbf{x}' - \mathbf{x}$ is not an element of $\mathcal{G}$. However, as $\mathbf{x}' - \mathbf{x}$ can be written as a linear combination of unit vectors and since unit vectors are elements of $\mathcal{G}$, we can decrease from $\mathbf{x}$ to reach $\mathbf{0}$, then increase to reach $\mathbf{x}'$.



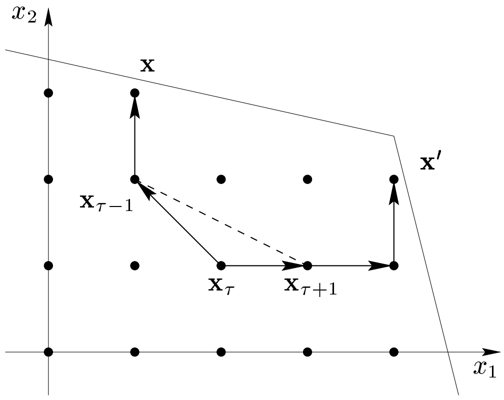
Hence, there exists a sequence $P = (\mathbf{x}^0, \ldots, \mathbf{x}^p)$ of vectors $\mathbf{x}^i$ and a number $1 \leq \tau < p$ with the properties:

(i) $\mathbf{x}^0 = \mathbf{x}$, $\mathbf{x}^p = \mathbf{x}'$,
(ii) for all $i = 1, \ldots, \tau$, $-(\mathbf{x}^i - \mathbf{x}^{i-1}) \in \mathcal{G}$,
(iii) for all $i = \tau+1, \ldots, p$, $(\mathbf{x}^i - \mathbf{x}^{i-1}) \in \mathcal{G}$,
(iv) every vector in P is feasible.

Let $p_{\min}$ be the smallest number such that there exists some sequence

$$P_{\min} \quad = \quad (\mathbf{x}^0, \ldots, \mathbf{x}^{p_{\min}})$$

of vectors $\mathbf{x}^i$ and a number $1 \leq \tau < p_{\min}$ satisfying (i), (ii), (iii) and (iv).



Since $-(\mathbf{x}^\tau - \mathbf{x}^{\tau-1}) \in \mathcal{G}$ and $(\mathbf{x}^{\tau+1} - \mathbf{x}^\tau) \in \mathcal{G}$, the difference vector

$$\mathbf{v} := (\mathbf{x}^{\tau+1} - \mathbf{x}^\tau) \; - \; (-(\mathbf{x}^\tau - \mathbf{x}^{\tau-1})) \quad = \quad \mathbf{x}^{\tau+1} \; - \; \mathbf{x}^{\tau-1}$$

has been computed in Step 2.2 of Algorithm 5.1. Moreover, both vectors $\mathbf{x}^{\tau+1}$ and $\mathbf{x}^{\tau-1}$ are feasible. It follows that $-\mathbf{b} \leq A\mathbf{v} \leq \mathbf{b}$ and $-u_i \leq v_i \leq u_i$ for all i.

In case that $\mathbf{0} \prec_{\mathbf{c}} \mathbf{v}$, the vector $\mathbf{v}$ was added to $\mathcal{G}$. Consequently,

$$P' \quad := \quad (\mathbf{x}^0, \ldots, \mathbf{x}^{\tau-1}, \mathbf{x}^{\tau+1}, \ldots, \mathbf{x}^{p_{\min}})$$

and $\tau - 1$ again satisfy properties (i) – (iv), yet involving $p_{\min} - 1$ vectors, a contradiction.

Therefore $\mathbf{v} \prec_{\mathbf{c}} \mathbf{0}$. In this case the vector $-\mathbf{v}$ was added to $\mathcal{G}$ in Algorithm 5.1. Then,

$$P' \quad := \quad (\mathbf{x}^0, \ldots, \mathbf{x}^{\tau-1}, \mathbf{x}^{\tau+1}, \ldots, \mathbf{x}^{p_{\min}})$$

and τ satisfy properties (i)-(iv). Since again only $p_{\min} - 1$ vectors belong to P', we obtain a contradiction.

Thus we have proved the following theorem.

Theorem 5.2. *Algorithm 5.1 terminates after a finite number of steps. The output is a test set for the integer programming problem* $\mathrm{IP}_{A,\mathbf{b},\mathbf{c}}$.

Compared to Algorithm 3.9, this extremely simple algorithm has some essential advantages. In particular, it works without the increase in dimension to obtain the extended problem: the computation takes place in the original space. However, Algorithm 5.1 still has the problem that it computes too many elements: the partial Gröbner basis computed is way too large. (For the basic version of the algorithm presented here, nearly all difference vectors of feasible points will be contained in $\mathcal{G}$!) Thus one has to work with reduction, and thus discard superfluous elements during the computation (see [21]). Also, this algorithm sometimes makes way too many comparisons, while generating only relatively few new basis elements. Such observations, made on a practical implementation, led to further variations of the algorithm that are currently still under investigation.

6. Problems

Exercise 6.1 (Standard forms of integer programs). Show that the 'equality standard form'

$$\min\{\mathbf{c}^\top \mathbf{x} \mid A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0}\},$$

and the 'inequality standard form'

$$\max\{\mathbf{c}^\top \mathbf{x} \mid A\mathbf{x} \leq \mathbf{b},\ \mathbf{0} \leq \mathbf{x} \leq \mathbf{u}\}.$$

of linear programs are equivalent: for any problem in one form we can construct a problem in the other form that solves it.
(Assume that $A \in \mathbb{N}^n$, A has no zero columns, $\mathbf{b} \in \mathbb{N}^m$, and $\mathbf{c} \in \mathbb{Z}^n$.)

Exercise 6.2 (Upper bounds). For a problem of the form

$$\max\{\mathbf{c}^\top \mathbf{x} \mid A\mathbf{x} \leq \mathbf{b},\ \mathbf{x} \geq \mathbf{0}\},$$

with $A \in \mathbb{N}^n$, $\mathbf{b} \in \mathbb{N}^m$, and $\mathbf{c} \in \mathbb{Z}^n$, how can we compute upper bounds u_i for the variables x_i? What happens in the special case when A has a zero-column?

The following problem sharpens our observation that S-pair formation corresponds to difference of vectors: we explicitly identify the 'superfluous' monomial factors that occur in the formation of S-pairs.

Exercise 6.3 (S-pairs and difference vectors). For $\mathbf{a}, \mathbf{b} \in \mathbb{Z}^n$, $\mathbf{a}, \mathbf{b} \succ \mathbf{0}$, the S-polynomial of $\mathbf{X}^{\mathbf{a}^+} - \mathbf{X}^{\mathbf{a}^-} \in I_{\mathcal{L}}$ and $\mathbf{X}^{\mathbf{b}^+} - \mathbf{X}^{\mathbf{b}^-} \in I_{\mathcal{L}}$ is

$$\mathbf{X}^{\min(\mathbf{a}^+, \mathbf{b}^+)}\,(\mathbf{X}^{(\mathbf{a}-\mathbf{b})^+} - \mathbf{X}^{(\mathbf{a}-\mathbf{b})^-}),$$

a monomial times the binomial corresponding to $\mathbf{a} - \mathbf{b}$.

Exercise 6.4 (Binomial criterion). A binomial $\mathbf{X}^{\mathbf{a}} - \mathbf{X}^{\mathbf{b}}$, with $\mathbf{a}, \mathbf{b} \geq \mathbf{0}$, is contained in $I_{\mathcal{L}}$ if and only if $\mathbf{a} - \mathbf{b} \in \mathcal{L}$.
(Hint: $\mathbf{X}^{\mathbf{a}}$ and $\mathbf{X}^{\mathbf{b}}$ reduce to the same standard monomial.)

Exercise 6.5 (Ideal of a lattice: generators). Assume that the lattice $\mathcal{L}$ is generated by the columns of a nonnegative matrix $A \in \mathbb{N}^{n \times n}$. Show that then the ideal $I_{\mathcal{L}}$ is generated by the binomials

$$\mathbf{X}^{\mathbf{a}^+} - \mathbf{X}^{\mathbf{a}^-}.$$

Show that this can fail if we do not assume A to be non-negative.
(A more general version of this is [21, Lemma 2.1].)

Exercise 6.6 (Gröbner bases of a lattice: an example). Let $\mathcal{L}_A \subseteq \mathbb{Z}^2$ be the 2-dimensional lattice generated by the columns of

$$A = \begin{pmatrix} 1 & 4 \\ 4 & 3 \end{pmatrix}.$$

Compute all the (four) different reduced Gröbner bases for the corresponding ideal. Describe the structure of the various Gröbner basis elements.
How many standard monomials are there in each case?

Exercise 6.7 (Gröbner bases of a lattice: geometry). Show that if $\mathcal{L}$ is a 2-dimensional integral lattice, then the universal Gröbner basis (the union of all the reduced Gröbner bases) consists of the following lattice vectors:

- the vertices $\mathbf{a} \in \mathbb{Z}^2$ of the polyhedron $\mathrm{conv}(\mathcal{L} \cap \mathbb{N}^2 \backslash \mathbf{0})$, and
- the vectors $\mathbf{a} \in \mathcal{L}$ that have one positive and one negative component, and for which $\mathbf{0}$ and $\mathbf{a}$ are two adjacent vertices of the polyhedron

$$\mathrm{conv}(\mathcal{L} \cap \{\mathbf{x} \in \mathbb{Z}^2 \mid \mathbf{x} \geq -\mathbf{a}^-\}).$$

(Remark: a similar structure theorem is true in higher dimensions as well, but harder to prove [17].)

Exercise 6.8 (A variant of Buchberger's algorithm: an example). Apply Algorithm 5.1 to compute a test set $\mathcal{G}$ for the 0/1 knapsack problem

$$\max\{x_1 + 2x_2 + 3x_3 \mid x_1 + 2x_2 + 3x_3 \leq 3,\ x_i \in \{0,1\},\ i = 1,2,3\}.$$

Identify a minimal test set $\mathcal{G}_{\min} \subseteq \mathcal{G}$.

7. Notes

While the theory of Gröbner bases [1, 2, 3, 6] yields basic ideas and tools, the discussion in this chapter stays in an 'elementary geometry' setting. Chvátal [4] and Schrijver [14] are excellent guides to all topics related to Linear and Integer Programming. [22] is a recent exposition of the geometry and combinatorics of polytopes.

As mentioned in the introduction, the basic ideas of Section 3. are due to Conti & Traverso [5]. The connection between lattices, binomial ideals and

Gröbner bases is relevant to interesting aspects in the theory of integer programming (the 'local situation', Gomory's [9] 'group problem'), but also, for example, to the ideals of toric varieties. The key reference for these directions is Sturmfels [15]. See [8] for more on binomial ideals.

Our presentation in Sections 3. and 4. is based on Thomas [18] [19, Chap. 2]. I am very grateful to Rekha Thomas for many helpful comments and discussions on this chapter, and for her permission to report about and draw on her materials.

The ideas for Section 5. are from [21]. An algebraic interpretation of the situation in terms of "truncated Gröbner bases" was given by Weismantel & Thomas [20].

We refer to Hoşten & Sturmfels for an alternative approach to the "phase I" problem. Recently, Li, Guo, Ida & Darlington [11] have described a combination of truncated Gröbner bases with the Hoşten-Sturmfels approach. They also reported some computational tests: on random problems of sizes up to "8×16" — which must still be considered very modest for all practical purposes. Computational results (also for larger, structured problems) are also presented in [10], in [7], and in [21].

A successful application of the Buchberger approach to a class of integer programming problems arising in practice was reported in Natraj, Tayur & Thomas [13].

References

1. W. W. Adams & P. Loustaunau: An Introduction to Gröbner Bases, Graduate Studies in Math., Vol. III, American Math. Soc., Providence RI 1994.
2. T. Becker & V. Weispfennig: Gröbner Bases: A Computational Approach to Commutative Algebra, Graduate Texts in Mathematics **141**, Springer-Verlag, New York 1993.
3. B. Buchberger: Gröbner bases: An algorithmic method in polynomial ideal theory, in: N.K. Bose (ed.), 'Multidimensional Systems Theory', D. Reidel 1985, 184-232.
4. V. Chvátal: Linear Programming, Freeman, New York 1983.
5. P. Conti & C. Traverso: Buchberger Algorithm and Integer Programming, Proceedings AAECC-9 (New Orleans), Springer LNCS **539**, 1991, pp. 130-139.
6. D. A. Cox, J. B. Little & D. O'Shea: Ideals, Varieties, and Algorithms. An Introduction to Computational Algebraic Geometry and Commutative Algebra, Undergraduate Texts in Mathematics, Springer-Verlag, New York 1992.
7. F. Di Biase & R. Urbanke: An algorithm to calculate the kernel of certain polynomial ring homomorphisms, *Experimental Math.* **4** (1995), 227-234.
8. D. Eisenbud & B. Sturmfels: Binomial ideals, *Duke Math. J.* **84** (1996), 1-45.
9. R. E. Gomory: Some polyhedra related to combinatorial problems, *Linear Algebra and its Applications* **2** (1969), 451-455.
10. S. Hoşten & B. Sturmfels: GRIN: An implementation of Gröbner bases for integer programming, in: "Integer Programming and Combinatorial Optimization" (E. Balas, J. Clausen, eds.), Proc. 4th Int. IPCO Conference (Copenhagen, May 1995), *Lecture Notes in Computer Science* **920**, Springer-Verlag 1995, 267-276.

11. Q. Li, Y. Guo, T. Ida & J. Darlington: The minimised geometric Buchberger algorithm: An optimal algebraic algorithm for integer programming, in: Proc. ISSAC'97, ACM Press 1997, pp. 331-338.
12. C. Moulinet & L. Pottier: Gröbner bases of toric ideals: properties, algorithms, and applications, preprint, INRIA Sophia Antipolis, 10 pages.
13. N. R. Natraj, S. R. Tayur & R. R. Thomas: An algebraic geometry algorithm for scheduling in presence of setups and correlated demands, *Math. Programming* **69A** (1995), 369-401.
14. A. Schrijver: Theory of Linear and Integer Programming, Wiley-Interscience, Chichester 1986.
15. B. Sturmfels: Gröbner Bases and Convex Polytopes, AMS University Lecture Series, Vol. 8, American Math. Soc., Providence RI 1995.
16. B. Sturmfels & R. R. Thomas: Variation of cost functions in integer programming, *Math. Programming* **77** (1997), 357-387.
17. B. Sturmfels, R. Weismantel & G. M. Ziegler: Gröbner bases of lattices, corner polyhedra, and integer programming, *Beiträge Algebra und Geometrie/Contributions to Algebra and Geometry* **36** (1995), 281-298.
18. R. R. Thomas: A geometric Buchberger algorithm for integer programming, *Math. Operations Research* **20** (1995), 864-884.
19. R. R. Thomas: Gröbner basis methods for integer programming, Ph. D. Thesis, Cornell University 1994, 157 pages.
20. R. R. Thomas & R. Weismantel: Truncated Gröbner bases for integer programming, *Applicable Algebra in Engineering, Communication and Computing (AAIECC)*, **8** (1997), 241-257.
21. R. Urbaniak, R. Weismantel & G. M. Ziegler: A variant of Buchberger's algorithm for integer programming, *SIAM J. Discrete Math.* **10** (1997), 96-108.
22. G. M. Ziegler: Lectures on Polytopes, Graduate Texts in Mathematics, Springer-Verlag, New York 1995.