GRIN: An Implementation of Gröbner Bases for Integer Programming

Serkan Hoşten[1] and Bernd Sturmfels[2]

[1] School of Operations Research and Industrial Engineering
Cornell University, Ithaca, New York 14853, USA
[2] Department of Mathematics, University of California,
Berkeley, California 94720, USA

1 Introduction

Integer programming (IP) is concerned with solving linear equations over the non-negative integers, subject to the requirement that a linear cost function is minimized. We consider families of integer programs where the matrix and the cost function are fixed but the right hand side varies. The algebraic technique of Gröbner bases provides tools for preprocessing such a family. The result of this preprocessing is a test set (in the sense of [13] and [14], §17.3) which allows the solution of each instance by following an edge path similar to the simplex algorithm for linear programming. The Gröbner basis approach was first proposed by Conti & Traverso [4] and Pottier [12], and it was further developed in [6, 11, 16, 17, 18, 19].

In this paper we present a computer program (GRIN) for solving and analyzing integer programs using Gröbner bases. The algorithms coded in GRIN are drawn from both commutative algebra and the standard IP repertoire (e.g. Lovasz' reduced lattice bases). We present two new algorithms for computing generators of toric ideals. One of them is due to DiBiase and Urbanke [6]. The emphasis of our discussion lies on experiments and practical computability. A comparison to existing integer programming software (CPLEX) is given as well.

2 Gröbner Bases as a Test Set

We begin with a brief review and problem specification. Consider the integer program

$$IP_{A,c}(b): \qquad \text{minimize } c \cdot u \text{ subject to } A \cdot u \;=\; b \text{ and } u \geq 0,\ u \in \mathbf{Z}^n,$$

where c is a generic vector in $\mathbf{R}^n$, A is a $d \times n$ integral matrix of rank d, and $b \in \mathbf{Z}^d$. Let $IP_{A,c}$ denote the family of all such programs with fixed matrix A and cost vector c, obtained by allowing b to vary over $\mathbf{Z}^d$. A *test set* for $IP_{A,c}$ is a finite subset $\mathcal{G}$ of $ker(A) \cap \mathbf{Z}^n$ such that, for every right hand side b and every feasible solution u of $IP_{A,c}(b)$, either u is the optimal solution of $IP_{A,c}(b)$ or there exists $g \in \mathcal{G}$ such that $u - g \geq 0$ and $c \cdot g > 0$.

Lemma 1. *There exists a unique minimal set of vectors $\alpha_1, \ldots, \alpha_t$ in $\mathbf{N}^n$ such that the set of non-optimal solutions to all programs in $IP_{A,c}$ equals $\bigcup_{i=1}^{t}(\alpha_i + \mathbf{N}^n)$.*

Let β_i be the unique optimal solution of $IP_{A,c}(A\alpha_i)$. Note that α_i and β_i have disjoint support, since α_i is a minimal element in the set of non-optimal solutions.

Proposition 2. [18] *The set $\mathcal{G}_c = \{\, \alpha_i - \beta_i \, : \, i = 1, \ldots, t \,\}$ is a minimal test set for the family of integer programs $IP_{A,c}$.*

We call $\mathcal{G}_c$ the *reduced Gröbner basis* of $IP_{A,c}$. It was introduced by Conti-Traverso [4] in the more general context of Gröbner bases for polynomial ideals. For an introduction we refer to the text books [1, 3, 5]. Our IP problem is encoded algebraically as follows. Let k be any field and $k[x_1, \ldots, x_n]$ the polynomial ring. Monomials $\mathbf{x}^u = x_1^{u_1} \ldots x_n^{u_n}$ are identified with lattice points $u = (u_1, \ldots, u_n)$ in $\mathbf{N}^n$. A *term order* $\succ$ is a total order on $\mathbf{N}^n$ which is compatible with sums and has $\mathbf{0}$ as its minimum element. If we fix a tie-breaking term order $\succ$, then every $c \in \mathbf{R}^n$ defines a term order $\succ_c$ by setting $\mathbf{x}^u \succ_c \mathbf{x}^v$ if $c \cdot u > c \cdot v$, or if $c \cdot u = c \cdot v$ and $u \succ v$. Our matrix A is encoded in the *toric ideal*

$$I_A \quad = \quad \langle\, \mathbf{x}^u - \mathbf{x}^v \, : \, u, v \in \mathbf{N}^n, \, u \, - \, v \, \in ker(A) \,\rangle.$$

Remark. The set of monomial differences $\{\, \mathbf{x}^{\alpha_i} - \mathbf{x}^{\beta_i} \, : \, \alpha_i - \beta_i \in \mathcal{G}_c \,\}$ is the reduced Gröbner basis of the toric ideal I_A with respect to the term order $\succ_c$.

The punch line is this: using the *Buchberger Algorithm* we can compute the test set $\mathcal{G}_c$ from any generating set of I_A. If v is any feasible solution of $IP_{A,c}(b)$, and $\mathbf{x}^u$ is the *normal form* of $\mathbf{x}^v$ with respect to $\mathcal{G}_c$, then u is the optimal solution of $IP_{A,c}(b)$. Once computed, the Gröbner basis $\mathcal{G}_c$ is an extremely valuable resource which has many other applications, including sensitivity analysis [16], sampling of feasible solutions [7], IP in the presence of stochastic constraints [11], and triangulations of polytopes [16]. See [15] for further details.

3 GRIN

GRIN (GRöbner basis for INteger programming) is an experimental software developed for computing Gröbner basis of toric ideals, in particular, for solving integer programs using Gröbner bases. It is intended to be a tool for both Combinatorial Optimization and Computational Algebra to support the research in the intersection of both fields by making basic computations in reasonable time. To be more specific, GRIN enables the user to compute the reduced Gröbner basis $\mathcal{G}_c$ of $IP_{A,c}$ where c is an integer vector. Once $\mathcal{G}_c$ is computed, the user can make multiple reductions which correspond to finding optimal solutions with respect to various right-hand-side vectors b. GRIN provides several input options: for instance, one may either input the matrix A or any binomial ideal J such that

$(J : (x_1 \cdots x_n)^\infty) = I_A$; see Proposition 3 below. A main feature is a built-in option for finding $\mathcal{G}_c$ by Algorithm 1 below. Furthermore, other state-of-the-art techniques such as Algorithm 2 due to Urbanke and DiBiase are also implemented. Both algorithms produce Gröbner bases for toric ideals considerably faster than any previously known method.

While efficient software to do Gröbner basis computations for general ideals is readily available (e.g. MACAULAY [2]), it is our aim to exploit the specific structure of toric ideals to achieve maximum speed. First experiments in the same direction within the system ALPI were reported in [4]. A more systematic study and implementation was carried out by C.Moulinet [10], who ran many experiments. When running some of Moulinet's examples in GRIN, we typically observed a speed-up by a factor of 10.

The toric ideal I_A that underlies the computations is very simple in the sense that it is generated by "binomials", that is, differences of monomials. The Buchberger Algorithm maintains this property throughout the entire computation. Binomials require very simple data structures since they can be identified with pairs of lattice points [18]. GRIN uses conventional Gröbnerian techniques to speed up the computation whenever this is suitable. For instance, we make effective use of criteria to cut down the number of S-pairs, which is a bottleneck during the computations. However, criteria which proved to be inefficient for the binomial case, such as Gebauer's B-criterion [8], are "switched off". Attention is paid to eliminating repetition of costly reductions by careful "house keeping" and heuristics. For instance, it is a common strategy to keep the set of binomials throughout the entire Gröbner basis computation as reduced as possible. We implemented this idea by doing global reductions periodically (whenever new elements of the size of a fixed percentage of the current basis are created) as opposed to doing it every time a new binomial is added. Another important strategy is the extraction of common monomial factors in every newly created S-pair. This extraction is justified by the fact that I_A is a prime ideal not containing any monomials. The above idea proved to be very effective, leading to reductions as much as 50-60 % in running times.

GRIN with its user manual is publicly available by anonymous ftp at the site `ftp.orie.cornell.edu`. We are particularly interested in having practitioners of integer programming test our program, and to provide us with feedback and suggestions. It is our belief that there is still much room for further improvements in the efficiency and applicability of the Gröbner basis approach. A particularly exciting recent development is the work of Urbaniak, Weismantel and Ziegler [19] who give a variant of Buchberger algorithm for integer programming with *fixed* right hand side. In this restricted setting, which has obvious advantages for 0-1 problems, their implementation is faster than GRIN.

4 A Gröbner Basis Method for Integer Programming

In the end of Section 2 it was remarked that $\mathcal{G}_c$ can be computed from any generating set of I_A. This is correct but generally irrelevant in practice since an explicit finite generating set of I_A is almost never available. Moreover, if A is a general matrix, then Theorem 4.1 in [16] implies that finding a generating set for I_A is as hard as finding $\mathcal{G}_c$ itself. In this section we give a new algorithm for solving both problems. Our results are further developments of the ideas and questions presented by Conti & Traverso [4].

The following standard method (for non-negative A) is frequently proposed in the literature [1, 4, 18]: Add "slack variables" $y_1, \ldots, y_d$ and form the ideal $I := \langle \mathbf{y}^{a_i} - x_i : i = 1, 2, \ldots, n \rangle$, where the monomial $\mathbf{y}^{a_i} := y_1^{a_{1i}} \cdots y_d^{a_{di}}$ encodes the i-th column of A. Then the toric ideal can be found as the elimination ideal $I_A = I \cap k[x_1, x_2, \ldots, x_n]$, which amounts to a Gröbner basis computation using an elimination term order [1, 5]. We found this formulation virtually useless for our purposes. The inefficiency of this approach is explained partly by the fact that the Buchberger algorithm is highly sensitive to the number of variables, which is $d + n$ instead of n as in I_A. Moreover, the intermediate Gröbner basis for I is much larger (often by a factor of 10) than the desired output $\mathcal{G}_c$.

Let $\mathcal{L} \subset \mathbf{Z}^n$ denote the (integer) kernel of A. Our toric ideal I_A equals $\langle \mathbf{x}^{u_+} - \mathbf{x}^{u_-} : u \in \mathcal{L} \rangle$, where $u = u_+ - u_-$ is the usual decomposition into positive and negative part. With every lattice basis $u^{(1)}, \ldots, u^{(n-d)}$ of $\mathcal{L}$ we associate a subideal of I_A, namely,

$$J \quad := \quad \langle \, \mathbf{x}^{u_+^{(i)}} - \mathbf{x}^{u_-^{(i)}} \; : i = 1, 2, \ldots, n - d \, \rangle.$$

Our approach is based on the following two propositions.

Proposition 3. *The toric ideal I_A equals the ideal quotient $(J : (x_1 x_2 \cdots x_n)^\infty) = \{ f \in k[x_1, \ldots, x_n] : f \cdot (x_1 x_2 \cdots x_n)^r \in J$ for some integer $r \}$.*

The ideal quotient can be computed one variable at a time as follows:

$$(J : (x_1 x_2 \cdots x_n)^\infty) \quad = \quad ((\cdots (J : x_1^\infty) : x_2^\infty) \cdots) : x_n^\infty).$$

Suppose that the row space of A contains a positive vector $\omega = (\omega_1, \ldots, \omega_n)$. Then the toric ideal I_A and its subideal J are homogeneous with respect to the grading $deg(x_i) = \omega_i$.

Proposition 4. *Fix the ω-graded reverse lexicographic term order induced by $x_1 > \cdots > x_n$, and let $\mathcal{F}$ be the reduced Gröbner basis of I. Then a Gröbner basis of $(I : x_n^\infty)$ is obtained by dividing each element $f \in \mathcal{F}$ by the highest power of x_n that divides f.*

This gives rise to the following algorithm.

Algorithm 1: *How to compute the Gröbner basis $\mathcal{G}_c$ of I_A*
1. Find any lattice basis $\mathcal{B}$ for $\mathcal{L} = ker(A)$ (e.g. using the Hermite normal

form algorithm,[14], §5.3)

2. *Replace $\mathcal{B}$ by the reduced lattice basis $\mathcal{B}_{red}$ (in the sense of Lovasz [9]; [14], §6.2).*
3. *Let $J_0 := \langle \mathbf{x}^{u_+} - \mathbf{x}^{u_-} \,:\, u \in \mathcal{B}_{red} \rangle$.*
4. *For $i = 1, 2, \ldots, n$: Compute $J_i := (J_{i-1} : x_i^\infty)$ using Proposition 4, i.e., by making x_i the reverse lexicographically cheapest variable.*
5. *Compute the reduced Gröbner basis $\mathcal{G}_c$ of $J_n = I_A$ with respect to $\succ_c$.*

Starting from the reduced lattice basis computed in Step 2 results in significant time savings in the subsequent Gröbner basis computations. Note that Step 4 involves n Gröbner basis computations. But our experiments show that they are short and easy calculations. This can be explained partially by the "optimal" starting basis $\mathcal{B}_{red}$, but the main improvement comes from the strategy of extracting common factors. Indeed, monomial extraction is an operation which interacts well with reverse lexicography and ideal quotients. The experiments we conducted with Algorithm 1 gave encouraging results, to be detailed in Section 6.

5 The Algorithm of Di Biase and Urbanke

We give an exposition of an alternative algorithm for computing a generating set for the toric ideal I_A. This algorithm is due to Di Biase and Urbanke [6]. It is based on the following two lemmas. We retain the notation from Section 4.

Lemma 5. *If one of the basis vectors $u^{(i)}$ has all coordinates positive, then the toric ideal I_A coincides with $J = \langle \mathbf{x}^{u_+^{(i)}} - \mathbf{x}^{u_-^{(i)}} \,:\, i = 1, 2, \ldots, n-d \rangle$.*

Proof. The hypothesis implies that all variables are invertible modulo J. This is equivalent to $J = \big(J : (x_1 x_2 \cdots x_n)^\infty\big)$. Now apply Proposition 4.1.

Let A_i denote the matrix which is obtained from A by reversing the signs in the i-th column. In what follows we let $\mathbf{x}^a, \mathbf{x}^b, \mathbf{x}^{a_j}, \mathbf{x}^{b_j}$ denote monomials which do not contain the variable x_i. Then a binomial $x_i^r \mathbf{x}^a - \mathbf{x}^b$ lies in I_{A_i} if and only if $\mathbf{x}^a - \mathbf{x}^b x_i^r$ lies in I_A. Let $\prec$ be any term order on $k[x_1, \ldots, x_n]$ which eliminates x_i, that is, all monomials containing x_i are higher than those not containing x_i.

Lemma 6. *Let $\mathcal{G}_i = \{x_i^{r_j} \mathbf{x}^{a_j} - \mathbf{x}^{b_j} : j = 1, 2, \ldots, m\}$ be a Gröbner basis for I_{A_i} with respect to $\prec$. Then $\mathcal{G} = \{\mathbf{x}^{a_j} - x_i^{r_j} \mathbf{x}^{b_j} : j = 1, 2, \ldots, m\}$ is a generating set for the ideal I_A.*

Proof. Let $\mathbf{x}^a - x_i^r \mathbf{x}^b$ be a binomial in I_A. Clearly, I_A is generated by binomials of this form. Then $x_i^r \mathbf{x}^a - \mathbf{x}^b$ lies in I_{A_i} and therefore reduces to zero modulo the Gröbner basis $\mathcal{G}_i$. Hence there exist polynomials $f_1, \ldots, f_m$ such that

$$x_i^r \mathbf{x}^a - \mathbf{x}^b \quad = \quad \sum_{j=1}^{m} f_j(x_1, x_2, \ldots, x_n) \cdot (x_i^{r_j} \mathbf{x}^{a_j} - \mathbf{x}^{b_j}),$$

and the variable x_i occurs with degree at most $r - r_j$ in f_j. (In particular, we have $f_j = 0$ whenever $r < r_j$.) This implies that the identity

$$\mathbf{x}^a - x_i^r \mathbf{x}^b \quad = \quad \sum_{j=1}^{m} x_i^{r-r_j} \cdot f_j(x_1, \ldots, x_{i-1}, \frac{1}{x_i}, x_{i+1}, \ldots, x_n) \cdot (\mathbf{x}^{a_j} - x_i^{r_j} \mathbf{x}^{b_j})$$

expresses the left hand side as a *polynomial* linear combination of the elements in $\mathcal{G}$. We conclude that $I_A = \langle \mathcal{G} \rangle$.

Algorithm 2 : *How to compute a generating set for the toric ideal* I_A

1. Choose a subset $\{i_1, \ldots, i_r\}$ *of* $\{1, \ldots, n\}$ *such that the kernel of* $A_{i_1 i_2 \cdots i_r} := (\cdots (A_{i_1})_{i_2} \cdots)_{i_r}$ *contains a strictly positive vector.*

2. Find a (nice) lattice basis $\mathcal{B}$ *for the kernel of* $A_{i_1 i_2 \cdots i_r}$ *which contains a strictly positive vector.*

3. The ideal $I_{A_{i_1 i_2 \cdots i_r}}$ *is generated by* $\{\mathbf{x}^{u_+} - \mathbf{x}^{u_-} : u \in \mathcal{B}\}$ *(by Lemma 5).*

4. Let $\ell := r$.

5. While $\ell \geq 1$ *do*

5.1. Compute a Gröbner basis $\mathcal{G}_{i_1 i_2 \cdots i_\ell}$ *for* $I_{A_{i_1 i_2 \cdots i_\ell}}$ *with respect to a term order which eliminates* x_{i_ℓ}.

5.2. Flip the variable x_{i_ℓ} *as in Lemma 6 to get generators for* $I_{A_{i_1 i_2 \cdots i_{\ell-1}}}$.

5.3 $\ell := \ell - 1$

6. Output the resulting generating set for I_A.

Algorithm 2 is due to DiBiase and Urbanke and appears in [6]. Our experimental results in the next section demonstrate that the efficiency of this algorithm is of the same general order of magnitude as that of Algorithm 1.

6 Experiments and Comparison with CPLEX

We ran Algorithm 1 and Algorithm 2 (implemented in GRIN) on randomly generated integer matrices A of various sizes (ranging from 4×8 to 8×16) with nonnegative entries in a prespecified range between 0 and 20. Once $\mathcal{G}_c$ was found, we selected many random feasible solutions v (and thereby random right hand sides b), and we reduced each v to the corresponding optimal solution u using the Gröbner basis $\mathcal{G}_c$. For each test instance (A, c, b) a comparison was made with a pure branch-and-bound algorithm (implemented in CPLEX 3.0).

If A was chosen to be a sparse 0-1-matrix, then the IP optimum was often "very close" to the LP optimum, and CPLEX outperformed GRIN by orders of magnitude. However, for random matrices A with entries of slightly larger magnitude ($0 - 20$, for example), we found that GRIN was competitive to CPLEX, and sometimes even faster. In these cases the IP optimum was "quite far" from the LP optimum, and CPLEX required an immense number of branching nodes. In particular, after the initial preprocessing of finding $\mathcal{G}_c$, it takes GRIN only a fraction of a second to determine the optimum, while CPLEX has to start "from scratch" and typically needs about ten seconds to compute the optimum.

In the table below we present a typical example from our experiments, which were performed on a SPARC10. The 4×8 matrix A has entries which are chosen uniformly at random from [0,20]. The Lovasz-reduced basis $\mathcal{B}_{red}$ of $ker(A) \cap \mathbf{Z}^8$ consists of the rows of the second matrix L. It took 26.0 seconds to find a first Gröbner basis for I_A. We chose four random cost functions c and transformed our first Gröbner basis into $\mathcal{G}_c$. This is now much faster, namely between 3.9 and 6.8 seconds. The size of $\mathcal{G}_c$ ranges from 233 to 294. We finally chose four feasible vectors v and reduced each of them modulo $\mathcal{G}_c$ to get the optimal solution u. Each reduction took less than 0.2 seconds.

As a comparison we solved the same problem using CPLEX. The CPLEX running time was between 6.97 and 21.27 seconds, and the number of branching nodes was between 6,180 and 19,549.

$$A = \begin{bmatrix} 15 & 4 & 14 & 19 & 2 & 1 & 10 & 17 \\ 18 & 11 & 13 & 5 & 16 & 16 & 8 & 19 \\ 11 & 7 & 8 & 19 & 15 & 18 & 14 & 6 \\ 17 & 10 & 13 & 17 & 16 & 14 & 15 & 18 \end{bmatrix}$$

$$L = \begin{bmatrix} 3 & -7 & -4 & 1 & 3 & 0 & -2 & 2 \\ -2 & 1 & -5 & -1 & -8 & 6 & 4 & 5 \\ -8 & 4 & -3 & 6 & 0 & 3 & -9 & 7 \\ -18 & -13 & 17 & -4 & 1 & 6 & 5 & 6 \end{bmatrix}$$

Time to compute a first GB of the toric ideal of A : 26.0 *secs*

cost vector c: **Sample feasible solution v:** **Optimal solution u:**	**Size of $\mathcal{G}_c$ /Time to compute $\mathcal{G}_c$** **Time to reduce / CPLEX time (# nodes)**
$c = [1\ 8\ 8\ 16\ 2\ 2\ 2\ 2]$: $v = [2\ 60\ 16\ 100\ 46\ 77\ 38\ 29]$: $u = [71\ 9\ 1\ 43\ 7\ 86\ 137\ 2]$	294/6.8 $*$[3]/13.57 (11975)
$c = [6\ 1\ 15\ 8\ 0\ 4\ 15\ 8]$: $v = [56\ 39\ 65\ 17\ 7\ 43\ 59\ 74]$: $u = [126\ 81\ 3\ 51\ 41\ 1\ 1\ 48]$	256/5.2 $*$/13.03 (10860)
$c = [0\ 8\ 18\ 1\ 0\ 12\ 15\ 14]$: $v = [7\ 33\ 93\ 75\ 96\ 93\ 99\ 63]$: $u = [154\ 42\ 2\ 124\ 197\ 0\ 2\ 2]$	233/3.9 $*$/21.27 (19549)
$c = [18\ 4\ 16\ 1\ 2\ 8\ 18\ 6]$: $v = [21\ 28\ 96\ 72\ 37\ 11\ 61\ 49]$: $u = [3\ 59\ 0\ 121\ 7\ 50\ 0\ 119]$	255/5.1 $*$/6.97 (6180)

When we use Algorithm 2 in "Phase I" to find a generating set for I_A, we get even more promising running times. In the above example if one starts with the lattice basis L of $ker(I_A)$, it only takes 10.6 seconds to compute such a generating set. The biggest factor in the speed-up is that the algorithm does $n/2$ iterations in the worst case, whereas Algorithm 1 has to do n iterations for the same problem. Moreover, a careful choice of the elimination order keeps the size of the intermediate Gröbner bases small, leading to further improvements. The conducted experiments showed that a block term order that makes the associated variable in each iteration the most expensive one and that uses graded reverse lexicographic order on the rest of the variables is a good choice. One should note that Algorithm 2 heavily depends on the starting lattice basis. A "better" basis $\mathcal{B}$ will reduce the number of iterations needed for the computation, and this will further improve the running times.

We conclude with a summary of the results of the experiments we have done with GRIN, shown in the following table. The range of the entries used in the problems are given in the second column. The third and fourth columns give the times for computing an initial Gröbner basis for each problem with Algorithm 1 and 2 respectively. Column 5 shows the range of the sizes of Gröbner bases of I_A we get when five randomly generated positive cost vectors are used. The corresponding range of running times is given in the following column. The last column is the range of CPLEX running times on the same problems. The

[3] all instances were under 0.2 secs

timings are in CPU seconds on a SPARC10. All of the test problems and detailed information on running times for individual problems can be obtained from `ftp.orie.cornell.edu` via anonymous ftp.

Problem	*Entries*	*Alg.1*	*Alg.2*	*Size*	*Time*	*CPLEX*
mat4x8.1	0-20	26.0	10.6	233-294	3.9-6.8	7.0-21.3
mat4x8.2	0-20	5.3	10.1	131-205	1.0-2.5	59.3-130.0
mat4x8.3	0-20	32.0	17.8	283-481	6.0-26.1	6.23-130.2
mat5x10.1	0-4	32.9	21.8	204-266	2.6-5.6	0.08-0.27
mat5x10.2	0-4	82.0	42.4	242-350	4.4-10.0	0.03-0.28
mat5x10.3	0-4	68.7	101.2	276-414	5.2-14.9	0.07-1.38
mat6x12.1	0-3	132.2	75.8	359-561	10.4-29.5	0.03-0.35
mat6x12.2	0-3	166.3	93.9	211-413	3.3-16.2	0.05-0.35
mat6x12.3	0-3	232.7	275.6	503-709	23.8-58.6	0.12-0.42
mat8x16.1	0-1	7.7	7.8	82-126	0.5-1.3	0.05-0.10
mat8x16.2	0-1	12.9	8.1	127-168	1.2-2.2	0.05-0.10
mat8x16.3	0-1	12.7	7.9	187-241	2.9-4.4	0.05-0.10

Acknowledgements. This project was supported in part by the David and Lucile Packard Foundation and the National Science Foundation. We wish to thank Lisa Fleischer for her contribution to the GRIN project during its initial phase January-May 1994.

References

1. W.W. Adams and P. Loustaunau, *An Introduction to Gröbner Bases*, American Mathematical Society, Graduate Studies in Mathematics, Vol. III, 1994.
2. D. Bayer & M. Stillman, Macaulay: a computer algebra system for algebraic geometry, available by anonymous ftp from `zariski.harvard.edu`.
3. T. Becker and V. Weispfenning, *Gröbner bases: a computational approach to commutative algebra*, Graduate Texts in Mathematics **141**, Springer-Verlag 1993.
4. P. Conti and C. Traverso, Gröbner bases and integer programming, Proceedings AAECC-9 (New Orleans), Springer Verlag, LNCS **539** (1991) 130–139.
5. D. Cox, J. Little and D. O'Shea, *Ideals, Varieties and Algorithms*, Springer Verlag, New York, 1992.
6. Di Biase, F., Urbanke, R., An algorithm to compute the kernel of certain polynomial ring homomorphisms, *Experimental Mathematics*, to appear.
7. P. Diaconis and B. Sturmfels, Algebraic algorithms for sampling from conditional distributions, *Annals of Statistics*, to appear.
8. R. Gebauer and H.M. Möller, On an installation of Buchberger's algorithm, *Journal of Symbolic Computation* **6** (1988) 275–286.

9. L. Lovász, *An Algorithmic Theory of Numbers, Graphs and Convexity*, CBMS-NSF Regional Conference Series in Applied Mathematics **50**, Society for Industrial and Applied Mathematics (SIAM), Philadelphia 1986.
10. C. Moulinet-Ossola, *Algoritmique des Réseaux et des Systèmes Diophantiens Linéares*, Dissertation, L'Université de Nice Sophia-Antipolis, 1995.
11. N.R. Natraj, S.R. Tayur and R.R. Thomas, An algebraic geometry algorithm for scheduling in the presence of setups and correlated demands, *Mathematical Programming*, to appear.
12. L. Pottier, Minimal solutions to linear diophantine systems: bounds and algorithms, in *Proceedings RTA '91*, Como, Springer LNCS **488**, pp. 162–173.
13. H.E. Scarf, Neighborhood systems for production sets with indivisibilities, *Econometrica* **54** (1986) 507-532.
14. A. Schrijver, *Theory of Linear and Integer Programming*, Wiley-Interscience Series in Discrete Mathematics and Optimization, New York, 1986.
15. B. Sturmfels, *Gröbner Bases and Convex Polytopes*, American Mathematical Society, 1996, to appear
16. B. Sturmfels and R. R. Thomas, Variation of cost functions in integer programming, Manuscript, 1994.
17. B. Sturmfels, R. Weismantel and G. Ziegler, Gröbner bases of lattices, corner polyhedra and integer programming, Manuscript, 1994.
18. R.R. Thomas, A geometric Buchberger algorithm for integer programming, *Mathematics of Operations Research*, to appear.
19. R. Urbaniak, R. Weismantel and G.Ziegler, A variant of the Buchberger algorithm for integer programming, Manuscript, Konrad-Zuse-Zentrum für Informationstechnik, Berlin, 1995.

This article was processed using the LaTeX macro package with LLNCS style