ELSEVIER

Theoretical Computer Science 263 (2001) 37–46

Theoretical Computer Science

www.elsevier.com/locate/tcs

On computing Hilbert bases via the Elliot–MacMahon algorithm

Dmitrii V. Pasechnik *

Faculty of Technical Mathematics and Informatics, Delft University of Technology, Mekelweg 4, 2628 CD Delft, Netherlands

Accepted April 2000

Abstract

The ways of using the Elliot–MacMahon algorithm to compute the Hilbert base of a system of linear Diophantine equations known so far are either not efficient or can fail to terminate. We present a version of an algorithm exploiting this range of ideas, which however is reasonably efficient as well as finite.

Keywords: Linear Diophantine equations; Elliot–MacMahon algorithm; Hilbert basis

1. Introduction

The problem of computing the minimal generating set H of the semigroup Σ of the nonnegative integer solutions of the system of equations

$$Ax = 0, \tag{1}$$

where A is an $d \times n$ integer matrix, recently received a lot of attention, particularly in computer science (AC-unification, Petri nets), in integer linear programming, as well as in combinatorics and computational commutative algebra. We refer the reader to [1, 2, 6, 7, 10–14] for more details, references and applications. We will call H *the Hilbert base* of A.

It was observed recently that a modification of the classical Elliott–MacMahon algorithm for computing the graded generating function of Σ can be used to compute H. It works by repeated transformations of a set S of certain subsets S_i of vectors in $\mathbb{Z}^n_{\geqslant 0}$ in the following way. In each step a *critical pair* of vectors $\{x, y\}$ belonging to some

* Corresponding author.
E-mail address: d.pasechnik@twi.tudelft.nl (D.V. Pasechnik).

0304-3975/01/$ - see front matter
PII: S0304-3975(00)00229-2

$S_i \in S$ is chosen in such a way that the vector $A(x+y)$ is in certain sense "closer" to the origin than Ax and Ay. Then S_i is replaced by two sets $S_i \cup \{x+y\} - \{x\}$ and $S_i \cup \{x+y\} - \{y\}$. (Optionally, other $S_k \supseteq \{x, y\}$ are replaces simultaneously, as well.) The starting set S consists of just one set, namely the set of standard basis vectors of $\mathbb{Z}^n_{\geqslant 0}$. Stanley [9] observed that when all $S_k \supseteq \{x, y\}$ are replaced simultaneously, the procedure can be interpreted as a sequence of subdivisions of a simplicial fan in $\mathbb{R}^n_{\geqslant 0}$.

Several obstacles lie on the way of making the procedure just described practically feasible. First, it is necessary to avoid computing the same element of H repeatedly. This can be accomplished via the Stanley's idea just mentioned. However, a naive way of using it might lead to an explosive exponential growth of the memory needed to store S. In Section 2 we will show (see Proposition 2.1) that the simplicial fan S can be recovered from its 1-skeleton, thus providing a crucial reduction in the memory usage.

Second, it is crucial to ensure a speedy termination of the procedure. Here the issue of the strategy to choose the critical pair $P = \{x, y\}$ needs to be addressed. Geometrically, it is natural to choose P so that

$$x^{\mathrm{T}} A^{\mathrm{T}} A y < 0 \tag{2}$$

holds. However, as Tomás [13] shows, the strategy of choosing P on the basis of (2) alone can lead to failure of termination. The only strategy proved to be finite so far basically solves the equations of (1) one by one. This is well-known to be in general rather inefficient, as the size of the Hilbert base of these intermediate systems can be huge compared to the size of H we seek. In Section 4 we will show that the strategy of selecting P minimizing

$$\deg x + \deg y = \sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i \tag{3}$$

among P's satisfying (2) is finite.

The remained of the paper is organized as follows. Section 2 concerns notation and preliminaries. The algorithm, for a range of strategies of choosing a critical pair, is described and shown to be complete in Section 3. Section 4 is devoted to a termination proof of our strategy, and Section 5 concludes the paper with a discussion and some data obtained from an experimental implementation of our algorithm.

2. Preliminaries

The notation used is mainly taken from Ziegler [15], Sturmfels [11] and Stanley [9].

2.1. Cones and fans

For a *cone* C, one can construct the complex $F(C)$ in the usual way, with the i-dimensional faces of C corresponding to $(i-1)$-dimensional simplices of $F(C)$. A cone C is *simplicial* is $F(C)$ is a simplex. A simplicial cone $C \subseteq \mathbb{R}^n_{\geqslant 0}$ generated by

linearly independent vectors $c_1,\ldots,c_k$ (notation $C=\mathrm{cone}(c_1,\ldots,c_k)$) is called a *lattice* cone (or a *unimodular* cone) if any vector in $C\cap\mathbb{Z}^n$ can be expressed as a nonnegative integer linear combination of $c_1,\ldots,c_k$. The latter is equivalent to the existence of $n-k$ integer vectors $c_{k+1},\ldots,c_n$ such that

$$\det(c_1^{\mathrm{T}},\ldots,c_n^{\mathrm{T}})=\pm 1. \tag{4}$$

Geometrically, it means the absence of integer vectors in the interior of the parallelepiped defined by $0,c_1,\ldots,c_k$.

A *fan* $\mathscr{F}$ is a set of cones in $\mathbb{R}^k$ satisfying the following two properties.

(i) Every nonempty face F of a cone $C\in\mathscr{F}$ also belongs to $\mathscr{F}$.

(ii) The intersection of two cones in $\mathscr{F}$ is a face of both of them.

A fan is called *simplicial* if all its cones are simplicial. A simplicial complex can be naturally associated with a simplicial fan. Now we are going to define two operations which, given a simplicial fan $\mathscr{F}$ in $\mathbb{R}^n_{\geqslant 0}$, produce a new one.

S_{xy}: *Edge split.* Given a cone $E=\mathrm{cone}(x,y)\in\mathscr{F}$, replace each cone $C=\mathrm{cone}(x,y,c_1,\ldots,c_k)\in\mathscr{F}$ by a pair of cones $C'=\mathrm{cone}(x,x+y,c_1,\ldots,c_k)$, $C''=\mathrm{cone}(y,x+y,c_1,\ldots,c_k)$.

R_x: *Vertex removal.* Given a ray $x\in\mathscr{F}$, remove from $\mathscr{F}$ all the cones containing x.

We shall consider fans obtained from $\mathbb{R}^n_{\geqslant 0}$ by repeatedly applying edge splits and vertex removals. Formally, we define a family $\Phi=\bigcup_{i\geqslant 0}\Phi_i$ of simplicial fans inductively by setting $\Phi_0=\{\mathbb{R}^n_{\geqslant 0}\}$, and $\Phi_{i+1}=S_{xy}(\Phi_i)$ for some $\mathrm{cone}(x,y)\in\Phi_i$ or $\Phi_{i+1}=R_x(\Phi_i)$ for some $\mathrm{cone}(x)\in\Phi_i$.

Denote by $G=G(\mathscr{F})$ the 1-skeleton of the simplicial complex of $\mathscr{F}\in\Phi$. As usual, $V(G)$ and $E(G)$ denote the sets of vertices and edges of G, respectively.

Proposition 2.1. *The cliques of $G=G(\mathscr{F})$ are in the natural one-to-one correspondence with the cones of $\mathscr{F}\in\Phi$.*

Remark 1. It is essential that $\mathscr{F}\in\Phi$. For instance for a fan $\mathscr{F}$ with the maximal cones $\mathrm{cone}(a,b)$, $\mathrm{cone}(a,c)$ and $\mathrm{cone}(b,c)$, where $a=(1\,0\,0)$, $b=(0\,1\,0)$ and $c=(0\,0\,1)$, the clique $\{a,b,c\}$ of $G(\mathscr{F})$ does not correspond to any cone of $\mathscr{F}$.

Proof. We proceed by induction. Observe that the statement holds for the fan $\{\mathbb{R}^n_{\geqslant 0}\}$ and assume that it holds for $\mathscr{F}_0\in\Phi$.

Trivially, a cone $C\in\mathscr{F}_0$ corresponds to a clique $G(C)$ of $G_0=G(\mathscr{F}_c0)$. Hence, in particular, the proposition holds for $\mathscr{F}=R_x(\mathscr{F}_0)$.

Next, let $\mathscr{F}=S_{xy}(\mathscr{F}_0)$. Observe that the graph G is obtained from the graph G_0 by replacing the edge (x,y) with the two-path (x,w,y), where $w=x+y$, and joining w to the common neighbours of x and y in G_0. Clearly, it suffices to prove the statement for the maximal cliques only. Let C be a maximal clique in G. If $w\notin C$ then C is a clique in G_0 unaffected by S_{xy}. Thus $\mathrm{cone}(C)$ is a cone in $\mathscr{F}\cap\mathscr{F}_0$, and we are done. Now assume $w\in C$. Observe that $C-\{w\}$ is a clique of G_0 consisting of some of the common neighbours of x and y in G_0, as well as of either x or y. (The

latter follows from the maximality of C.) Without loss of generality, $x \in C$. Moreover, $C_0 = C \cup \{y\} - \{w\}$ is a clique of G_0. Hence, by induction, $\mathrm{cone}(C_0) = \mathrm{cone}(C - \{w\}, y)$ is a cone of $\mathscr{F}_0$ and $\mathrm{cone}(C) \in \mathscr{F}$ by definition of S_{xy}. □

Remark 2. Tomás observed, after a preliminary version of this paper was distributed, that a rather similar idea appears in [4]. The exposition we give here is however quite different from the one in [4] and is easily accessible for a reader unfamiliar with "constraint propagation" technique of [4].

The proposition just proved shows that the combinatorics of $\mathscr{F} \in \Phi$ is completely encoded in $G(\mathscr{F})$. The following proposition, derived from Stanley [9, Lemma 3.7], deals with the geometry of $\mathscr{F}$.

Proposition 2.2 (Stanley [9]). *The cones of $\mathscr{F} \in \Phi$ are lattice cones.*

Proof (*Stanley* [9]). For the fan $\{\mathbb{R}^n_{\geqslant 0}\}$ the statement holds. Assume that is holds for $\mathscr{F}_0 \in \Phi$. Then it clearly holds for $\mathscr{F} = R_x(\mathscr{F}_0)$. Let $\mathscr{F} = S_{xy}(\mathscr{F}_0)$ and $w = x + y \in C \in \mathscr{F}$, with C maximal (obviously it suffices to prove the statement for the maximal cones). Thus we can assume $x \in C$ and $C = \mathrm{cone}(w, x, c_1, \ldots, c_k)$. Then $C_0 = \mathrm{cone}(x, y, c_1, \ldots, c_k) \in \mathscr{F}_0$ by definition of S_{xy} and is a lattice cone by assumption. Thus,

$$\begin{aligned} \pm 1 &= \det(x^{\mathrm{T}}, y^{\mathrm{T}}, c_1^{\mathrm{T}} \cdots c_k^{\mathrm{T}}, d_1^{\mathrm{T}} \cdots d_{n-k-2}^{\mathrm{T}}) \\ &= \det((x+y)^{\mathrm{T}}, y^{\mathrm{T}}, c_1^{\mathrm{T}} \cdots c_k^{\mathrm{T}}, d_1^{\mathrm{T}} \cdots d_{n-k-2}^{\mathrm{T}}), \end{aligned}$$

applying (4) to get the first equality, and a well-known determinantal identity to get the second one. Hence the proposition. □

2.2. Matrices and vectors

The vectors $v \in \mathbb{Z}^n_{\geqslant 0}$ satisfying (1) form the semigroup Σ. The *Hilbert base* of A is the (unique, cf. e.g. [8, Theorem 14.6]) set H of vectors $v \in \mathbb{Z}^n_{\geqslant 0}$ satisfying (1) and such that

1. $x = \sum_{h \in H} \alpha_h h$, $\alpha_h \in \mathbb{Z}_{\geqslant 0}$ for any $x \in \Sigma$ (i.e. H *generates* Σ);
2. $u - v \notin \mathbb{Z}^n_{\geqslant 0}$ for any $u, v \in H$ (i.e. H is *minimal*).

In what follows $\|x\|$ stands for the Euclidean norm of $x \in \mathbb{R}^n$ (i.e. $\|x\| = \sqrt{\sum_{i=1}^n x_i^2}$). The *support* of x is the set $\mathrm{supp}\, x = \{j \mid 1 \leqslant j \leqslant n,\ x_j \neq 0\}$.

3. Description of the algorithm

In this section we describe an algorithm $\mathbf{HB} = \mathbf{HB}(A, R, b, R', b')$ computing the intersection $H \cap P$ of the Hilbert base H of A with the set

$$P = \{x \in \mathbb{R}^n_{\geqslant 0} \mid Rx \leqslant b, R'x < b'\}, \quad R^{(')} = (r_{ij}),\ r_{ij} \geqslant 0.$$

We allow $R = R' = 0$, so that $H \cap P = H$ in this particular case.

HB works by updating a fan $\mathscr{F}$ by applying edge splits (and, optionally, vertex removals) starting from $\mathscr{F} = \{\mathbb{R}^n_{\geqslant 0}\}$. $\mathscr{F}$ is stored as $G = G(\mathscr{F})$, as Proposition 2.1 allows. As $x \in V(G)$ satisfying (1) appear, they are stored in H, which is initially empty. We shall denote

$$\begin{aligned} E_S &= E_S(G) \\ &= \{(x,y) \in E(G) \mid x^{\mathrm{T}} A^{\mathrm{T}} A y < 0,\ R(x+y) \leqslant b,\ R'(x+y) < b', \\ &\qquad \text{there is no } w \in H \text{ with } x + y \geqslant w\}. \end{aligned} \tag{5}$$

The edges (x,y) for the splitting S_{xy} will always be selected from E_S. The set E_S is maintained by calling **HB** recursively. This allows to describe the procedure in a greater generality. Other possibilities for maintaining E_S are described in Section 5.

The algorithm in pseudo-code is given in Fig. 1. First, we assume that the step (**Optional**) is not performed at all.

Lemma 3.1. *Let $w \in H \cap P$ lie in the interior of a cone $C \in \mathscr{F}$, where $G = G(\mathscr{F})$ is obtained at some stage of the execution of the algorithm (assuming* (**Optional**) *not being performed). Then $E_S = E_S(G) \neq \emptyset$.*

Proof. As $C = \mathrm{cone}(c_1, \ldots, c_k)$ is a lattice cone (cf. Proposition 2.2) and w is in its interior,

$$w = \sum_{i=1}^{k} \alpha_i c_i \quad \text{with } \alpha_i \geqslant 1 \text{ for } 1 \leqslant i \leqslant k.$$

Applying A to both sides of this expression yields

$$0 = \sum_{i=1}^{k} \alpha_i A c_i.$$

The latter implies that $c_i^{\mathrm{T}} A^{\mathrm{T}} A c_j < 0$ for some $1 \leqslant i < j \leqslant k$. As $\alpha_i \geqslant 1$, $\alpha_j \geqslant 1$, and $w \in P$, we have $c_i + c_j \in P$. Hence $(c_i, c_j) \in E_S \neq \emptyset$. □

The following can be proved by a straightforward inspection of the definition of E_S.

Lemma 3.2. *Let (x,y,v) be a triangle in G with (x,v) and (y,v) not in E_S. Then $(x+y,v)$, should it arise during the run of algorithm, cannot belong to E_S.*

Thus the vertices of G removed during the step (**Optional**) are "redundant", as they do not participate in updating of E_S. Hence this step does not affect the outcome of the algorithm.

By Lemma 3.1, if the algorithm terminates then $w \in H$ will appear in the block **Choice** of the algorithm as $w = x + y$ for some $(x,y) \in E_S$. Thus H will be returned upon termination.

Input: 1. A - $d \times n$ integer matrix
2. R - $r \times n$ matrix of nonnegative integers
3. b - vector in $\mathbb{R}^r_{\geqslant 0}$.
4. R' - $r' \times n$ matrix of nonnegative integers
5. b' - vector in $\mathbb{R}^{r'}_{\geqslant 0}$.

Output: $\{x \in H \mid Rx \leqslant b, R'x < b'\}$, where H is the Hilbert base of A

Initialize: $G := G(\mathbb{R}^n_{\geqslant 0})$; $H := \emptyset$;
for $v \in \mathrm{V}(G)$ *do*
if $Rv \not\leqslant b$ *or* $R'v \not< b'$ *then* $G := R_v(G)$; *fi*;
if $Av = 0$ *then* $G := R_v(G)$; $H := H \cup \{v\}$; *fi*;
od
Set E_S using (5);

Main loop: *while* $E_s \neq \emptyset$ *do*
Choice: Choose $(x, y) \in E_S$

$$w := x + y;\ H_w := \mathbf{HB}\left(A, R, b, \begin{pmatrix} R' \\ I \end{pmatrix}, \begin{pmatrix} b' \\ w \end{pmatrix}\right);$$

if $H_w = \emptyset$ *then*
if $Aw = 0$ *then*
$H := H \cup \{w\}$; $E_S := E_S - \{(x, y)\}$;
else $G := S_{xy}(G)$;
$E_S := E_S \cup \{(w, u) \in E(G) \mid w^{\mathrm{T}}A^{\mathrm{T}}Au < 0,\ R(w + u) \leqslant b,\ R'(w + u) < b'\}$
fi;
else $E_S := E_S - \{(x, y)\}$;
fi;

(Optional): *for* $v \in \mathrm{V}(G)$ *do*
if $\{u \mid (u, v) \in E_S\} = \emptyset$ *then* $G := R_v(G)$; *fi*
od;
od;
return H;
END **HB**;

Fig. 1. The algorithm $\mathbf{HB}(A, R, b, R', b')$.

4. Termination

The way the algorithm is described in Section 3 leaves the question of choosing w in the step **Choice** widely open. In fact, it is shown in [13] that for certain ways

of performing this step, in particular when (x, y) with the least value of $x^{\mathrm{T}}A^{\mathrm{T}}Ay$ is selected in E_S, can lead to failure of termination. However, if we process $V(G)$ degree by degree (recall that $\deg x = \sum_i x_i$) then, as we shall show, the situation is quite satisfactory. We use a generalization of the technique due to Contejean and Devie [2, Section 5]. Namely, we prove the following.

Theorem 4.1. *If* $(x, y) \in E_S$ *of minimal* $\deg x + \deg y$ *is always selected at the step* **Choice**, *then the algorithm terminates.*

Proof. Assume the contrary. Then in the graph G we will have an infinite degree-increasing path $v_1, v_2, v_3, \ldots, v_i, \ldots$, where $v_{i+1} = v_i + x_i$ for some $x_i \in V(G)$ with $v_i^{\mathrm{T}}A^{\mathrm{T}}Ax_i < 0$ for every $i \geqslant 1$. □

We shall use following generalization of [2, Lemma 5.1].

Claim 4.2.

$$\lim_{k\to\infty} \frac{||Av_k||}{\deg v_k} = 0.$$

Proof of the Claim. It suffices to show that for *any* vertex $v_k \in V(G)$ we have $||Av_k||^2 \leqslant C \deg v_k$, for a constant C. We proceed by induction. Using $v_{k-1}^{\mathrm{T}}A^{\mathrm{T}}Ax_{k-1} < 0$ we derive

$$\begin{aligned} ||Av_k||^2 &= ||Av_{k-1}||^2 + ||Ax_{k-1}||^2 + 2v_{k-1}^{\mathrm{T}}A^{\mathrm{T}}Ax_{k-1} \\ &\leqslant C \deg v_{k-1} + C \deg x_{k-1} = C \deg v_k. \end{aligned}$$

Now define $u_k = v_k / \deg v_k$. Then $\deg u_k = 1$ and $\lim_{k\to\infty} ||Au_k|| = 0$.[1] As u_k is a sequence on the n-dimensional simplex, which is a compact set, there is an adherence point ℓ. Namely, ℓ is the limit of the subsequence $(u_{\phi(k)})_{k\geqslant 1}$, where ϕ is an increasing mapping on $\mathbb{Z}_{>0}$. By the continuity of the mapping $x \mapsto ||Ax||$,

$$||A\ell|| = \lim_{k\to\infty} ||Au_{\phi(k)}|| = 0.$$

Thus ℓ satisfies (1).

Starting from ℓ, we construct a nonzero solution $r \in \mathbb{Z}^n_{\geqslant 0}$ to (1) with $\operatorname{supp} r \subseteq \operatorname{supp} \ell$. Note that ℓ lies in the *rational* cone $\{x \in \mathbb{R}^n_{\geqslant 0} \mid Ax = 0\}$ (rational means that the extremal rays of it are rational vectors). As it is finite-dimensional, these rays can be given by vectors r_i's in $\mathbb{Z}^n_{\geqslant 0}$. Thus ℓ is a sum of r_i's with nonnegative coefficients. Hence there is a constant N such that $N\ell \geqslant r$, for some $r = r_i$, an extremal ray of the cone.

[1] At this point it suffices to refer the reader to Contejean and Devie [2, Proposition 5.3] to complete the proof. As the second part of their proof (starting from the middle of p. 153) can be considerably streamlined, we would nevertheless like to give a complete proof here.

From the definition of u_k we know that for $j \in \operatorname{supp} \ell$, the jth entry of v_k increases without bound as k increases. Thus for some k_0 we have $v_{k_0} \geqslant r$, the desired contradiction. □

5. Discussion

5.1. The implementation

The author has done an experimental C language implementation of the algorithm described in Section 3, Fig. 1, with the minimal degree edges (x, y) chosen at the step **Choice**, as justified by Theorem 4.1. The graph G is maintained using balanced binary trees to represent $V(G)$ and the adjacencies of the vertices.

The recursive call to **HB** in **Choice** in the implementation is replaced by a search in the set H kept as an hierarchical structure. Due to the order of the edges in E_S processed, H always contains the elements of the Hilbert base of A of degree less than the one currently processed. While this works well when $|H|$ is small, this becomes the bottleneck of algorithm when, roughly speaking, $|H|$ becomes comparable with $|V(G)|$. The alternatives would be to try the recursive call to **HB**, or to some other algorithm computing the Hilbert base, e.g. the one described in [2]. Perhaps the most promising is to try to keep some vertices of G (removed by the step (**Optional**) in the current implementation), so that the recursive calls to **HB** would not have to rebuild (parts of) G. This needs further analysis.

Also, one could try to keep H in the data structures known in computational geometry such as *Kd-trees* and *range trees*. For them, however, the worst case analysis does not give too much ground for optimism, either (the "rectangular query range" cannot be searched faster than $O(\log^{n-1} |H|)$). See [3] for details on this.

Finally, we observed that for large G's the search of vertices in $V(G)$ and the vertex adjacencies slowed down. More experiments are needed here with efficient ways of maintaining large sets.

5.2. Comparison and conclusion

It appears that the most efficient algorithm known to date is described in [2]. The major attraction of it is that the space needed by it is just a constant $O(dn)$. However, it generally needs to perform more additions of vectors of length d, i.e. linear combinations of columns of A, than Elliott–MacMahon algorithm. For instance for the (1×2)-matrix $A = (a, b)$ it needs to perform roughly ab vector additions, while Elliott–MacMahon takes at most $\max\{|a|, |b|\}$.

The preliminary results of testing of the implementation appear to be encouraging. For instance, on the (proposed in [11, Chapter 6]) benchmarks *homogeneous primitive partition identities*, where A is the $(2 \times 2n)$-matrix

$$A = A_n = \left\{ \pm \begin{pmatrix} i \\ 1 \end{pmatrix} \;\middle|\; 1 \leqslant i \leqslant n \right\},$$

our program outperforms the program implementing [2], as much as by a factor of 100 for $n \geqslant 6$. As well, on some tests from Tables II and III of [2] our program is faster (while on some, especially "small" ones, slower).

The memory problems start to affect the performance when G gets large. This is certainly the price one pays for the faster performance, not so uncommon in the theory of algorithms.

Finally, we must stress that there is apparently no particular algorithm for finding the Hilbert base which is much better than the rest of them. Specific domains need specific algorithms. We hope that Elliott–MacMahon algorithm will have its place, as well.

Acknowledgements

The author thanks L. Pottier, F. Laburthe, and especially A.P. Tomás for helpful discussions. The author thanks E. Contejean for kindly supplying an implementation of algorithm from [2]. A part of this research was concluded while the author held a position at Research Institute for Applications of Computer Algebra, Technical University of Eindhoven. The author is currently supported by the NWO grant SWON 4-53603 awarded to C. Roos and T. Terlaky.

References

[1] M. Adi, C. Kirchner, AC-unification race: the system solving approach, implementation and benchmarks. J. Symbolic Comput. 14(1) (1992) 51–70.
[2] E. Contejean, H. Devie, An efficient incremental algorithm for solving systems of linear Diophantine equations, Inform. and Comput. 113(1) (1994) 143–172.
[3] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, Computational Geometry: Algorithms and Applications, Springer, Berlin, 1997.
[4] E. Domenjoud, A.P. Tomás, From Elliott–MacMahon to an algorithm for general linear constraints on naturals., in: H. Montanari, F. Rossi (Eds.), Proceedings Principles and Practice of Constraint Programming – CP'95, vol. 976, Lecture Notes in Computer Science, Springer, Berlin, 1995, pp. 18–35.
[5] M. Filgueiras, A.P. Tomás, A note on the implementation of the Elliott–MacMahon algorithm, Tech. Rep. Department of Computer Science, University of Porto, June 1992.
[6] M. Henk, R. Weismantel, On Hilbert bases of polyhedral cones, Results Math. 32 (1997) 298–303.
[7] L. Pottier, Minimal solutions of linear Diophantine systems: bounds and algorithms, in: R.V. Book (Ed.), Proceedings of the 4th International Conference on Rewriting Techniques and Applications, vol. 488, Lecture Notes in Computer Science, Springer, Berlin, 1991, pp. 162–173.
[8] A. Schrijver, Theory of Linear and Integer Programming, Wiley, New York, 1986.
[9] R.P. Stanley, Magic labellings of graphs, symmetric magic squares, systems of parameters and Cohen-Macaulay rings, Duke Math. J. 40 (1973) 607–632.
[10] R.P. Stanley, Enumerative combinatorics, vol. 1, Cambridge Studies in Advanced Mathematics, vol. 49, Cambridge University Press, Cambridge, 1997. With a foreword by Gian-Carlo Rota, Corrected reprint of the 1986 original.
[11] B. Sturmfels, Grobner Bases and Convex Polytopes, University Lecture Series, vol. 8, American Mathematical Society, Providence, RI, 1996.
[12] R.R. Thomas, A geometric Buchberger algorithm for integer programming, Math. Oper. Res. 20 (1995) 864–884.

[13] A.P. Tomás, On solving linear Diophantine constraints, Ph.D. Thesis, Department of Computer Science, University of Porto, January 1997.
[14] G. Ziegler, Gröbner bases and integer programming, in: H.S.A.M. Cohen, H. Cuypers (Eds.), Some Tapas of Computer Algebra, Springer, Berlin, 1999.
[15] G.M. Ziegler, Lectures on Polytopes, Graduate Texts in Mathematics, vol. 152, Springer, New York, 1995.