Exploiting Symmetries in the Computation of Graver Bases

Raymond Hemmecke
Otto-von-Guericke-University Magdeburg
raymond@hemmecke.de

Abstract

Many challenging Graver bases computations, like for multi-way tables in statistics, have a highly symmetric problem structure that is not exploited so far computationally. In this paper we present a Graver basis algorithm for sublattices of $\mathbb{Z}^n$ that exploits existing symmetry.

1 Introduction

Graver bases, originally introduced by Graver [3] for use in integer programming, have a variety of interesting applications. Besides providing improving directions for integer programs [3, 5, 13], for stochastic integer programs [6], and even for certain convex integer programs [7, 10], Graver bases can for example also be used as Markov bases for sampling in statistics [2], or as a superset from which a universal Gröbner basis of the toric ideal $I_A := \langle x^u - x^v : Au = Av, u, v \in \mathbb{Z}^n_+ \rangle$ can be extracted [12].

Unfortunately, the size of Graver bases increases quickly with the dimension, making it very hard if not impossible to compute them in practice. In several applications, however, as for example in algebraic statistics, the problems involve a high symmetry that should make it much easier to compute the Graver basis in terms of (relatively few) representatives. In [8, 9], the authors exploit existing symmetry of lattices that arise from a single constraint of the form $a^\intercal x \equiv 0 \pmod p$, $p \in \mathbb{Z}_+$, in a similar way as we do in Lemma 1.2 and in Corollary 2.4 below.

Let us start our presentation by defining the notion of a Graver basis and by giving an example that demonstrates the problem we are interested in.

The **Graver basis** $\mathcal{G}(\Lambda)$ associated to a lattice $\Lambda \subseteq \mathbb{Z}^n$ consists exactly of all $\sqsubseteq$-minimal *nonzero* elements in Λ, where for $u, v \in \mathbb{Z}^n$ we say that $u \sqsubseteq v$ if $u^{(j)}v^{(j)} \geq 0$ and $|u^{(j)}| \leq |v^{(j)}|$ for all components $j = 1, \ldots, n$, that is, if u belongs to the same orthant as v and its components are not greater in absolute value than the corresponding components of v. Note that Graver originally defined this set only for the case $\Lambda = \ker(A) \cap \mathbb{Z}^n$ for given matrix $A \in \mathbb{Z}^{d \times n}$, but his definition can be readily extended to the definition we gave above.

Example 1.1 Consider the set of all 3×3 tables/arrays whose entries are filled with integer numbers in such a way that the sums along each row and along each column are 0. One particular example is the table

$$\begin{pmatrix} 1 & -1 & 0 \\ -1 & 3 & -2 \\ 0 & -2 & 2 \end{pmatrix}.$$

If we encode the 9 entries of the table as $z_1, \ldots, z_9$, then the set of 3×3 tables coincides with the integer vectors in the kernel of the matrix

$$A_{3\times 3} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix},$$

that is, with all $z \in \mathbb{Z}^9$ satisfying $A_{3\times 3} z = 0$. As defined above, the Graver basis of $A_{3\times 3}$ consists of all $\sqsubseteq$-minimal *nonzero* tables among them. The particular 3×3 table above does not belong to the Graver basis of $A_{3\times 3}$, since

$$\begin{pmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \sqsubseteq \begin{pmatrix} 1 & -1 & 0 \\ -1 & 3 & -2 \\ 0 & -2 & 2 \end{pmatrix}.$$

Using the computer program 4ti2 [4], we find that the following 15 vectors (and their negatives) constitute the Graver basis of $A_{3\times 3}$:

$$\begin{array}{rrrrrrrrr}
(1, & -1, & 0, & -1, & 1, & 0, & 0, & 0, & 0) \\
(0, & 0, & 0, & 1, & 0, & -1, & -1, & 0, & 1) \\
(1, & 0, & -1, & -1, & 0, & 1, & 0, & 0, & 0) \\
(1, & -1, & 0, & 0, & 0, & 0, & -1, & 1, & 0) \\
(0, & 0, & 0, & 1, & -1, & 0, & -1, & 1, & 0) \\
(1, & -1, & 0, & -1, & 0, & 1, & 0, & 1, & -1) \\
(0, & -1, & 1, & 1, & 0, & -1, & -1, & 1, & 0) \\
(1, & -1, & 0, & 0, & 1, & -1, & -1, & 0, & 1) \\
(1, & 0, & -1, & 0, & 0, & 0, & -1, & 0, & 1) \\
(0, & -1, & 1, & 0, & 1, & -1, & 0, & 0, & 0) \\
(0, & 1, & -1, & 1, & -1, & 0, & -1, & 0, & 1) \\
(0, & 0, & 0, & 0, & 1, & -1, & 0, & -1, & 1) \\
(0, & -1, & 1, & 0, & 0, & 0, & 0, & 1, & -1) \\
(1, & 0, & -1, & 0, & -1, & 1, & -1, & 1, & 0) \\
(1, & 0, & -1, & -1, & 1, & 0, & 0, & -1, & 1)
\end{array}$$

However, there is an obvious symmetry group $S_3 \times S_3 \times S_2$ operating on the set of 3×3 tables whose elements transform a given table $v \in \ker(A_{3\times 3})$ into another table $w \in \ker(A_{3\times 3})$ by suitably rearranging components (permuting rows or columns, flipping the table along the main diagonals). If we take these symmetries into account, we see that among these 15 elements there are in fact only two essentially different elements:

$$\begin{array}{cccccccccc} (1, & -1, & 0, & -1, & 1, & 0, & 0, & 0, & & 0) \\ (1, & -1, & 0, & -1, & 0, & 1, & 0, & 1, & & -1) \end{array}$$

or, in a more array-like notation:

$$\begin{pmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & -1 \end{pmatrix}.$$

It should be clear that for bigger or for higher-dimensional tables, this difference in sizes becomes far more striking, since the acting symmetry groups are much bigger. □

In the following, let a sublattice $\Lambda \subseteq \mathbb{Z}^n$ be given and let $\mathcal{S}_\Lambda \subseteq S_n$ be a group of symmetries such that for all $v \in \Lambda$ and for all $\sigma \in \mathcal{S}_\Lambda$ we have that also $\sigma(v) := (v^{\sigma(1)}, \ldots, v^{\sigma(n)}) \in \Lambda$. Finally, denote by $\mathrm{orb}_{\mathrm{S}_\Lambda}(v) := \{\sigma(v) : \sigma \in \mathcal{S}_\Lambda\}$ the orbit of v under $\mathcal{S}_\Lambda$.

Lemma 1.2 *Let $\Lambda \subseteq \mathbb{Z}^n$ be a lattice, let $\mathcal{S}_\Lambda \subseteq S_n$ be the group of its symmetries, and let $g, g', s, s' \in \Lambda$. Then the following holds:*

- *If $g \sqsubseteq s$ then $\sigma(g) \sqsubseteq \sigma(s)$ for every $\sigma \in \mathcal{S}_\Lambda$.*
- *If there is no $g' \in \mathrm{orb}_{\mathrm{S}_\Lambda}(g)$ with $g' \sqsubseteq s$, then for every $s' \in \mathrm{orb}_{\mathrm{S}_\Lambda}(s)$ there is no $g'' \in \mathrm{orb}_{\mathrm{S}_\Lambda}(g)$ with $g'' \sqsubseteq s'$.*
- *If $v \in \mathcal{G}(\Lambda)$ then $\mathrm{orb}_{\mathrm{S}_\Lambda}(v) \subseteq \mathcal{G}(\Lambda)$.*

Proof. The first statement follows immediately from the definition of $\sqsubseteq$.

For the second statement, let $s' = \sigma(s)$ for some $\sigma \in \mathcal{S}_\Lambda$ and assume there is some $g'' \in \mathrm{orb}_{\mathrm{S}_\Lambda}(g)$ with $g'' \sqsubseteq s'$. Then $g' := \sigma^{-1}(g'') \sqsubseteq \sigma^{-1}(s') = s$ and $g' \in \mathrm{orb}_{\mathrm{S}_\Lambda}(g)$. A contradiction to the assumed non-existence of such g'.

For the last statement we have to show that with $v \in \mathcal{G}(\Lambda)$ also the full orbit $\mathrm{orb}_{\mathrm{S}_\Lambda}(v)$ lies in $\mathcal{G}(\Lambda)$. For this it suffices to assume that there is some $\sigma(v) \in \mathrm{orb}_{\mathrm{S}_\Lambda}(v)$ that could be written non-trivially as $\sigma(v) = w_1 + w_2$ with $w_1, w_2 \sqsubseteq \sigma(v)$. But this would imply $v = \sigma^{-1}(w_1) + \sigma^{-1}(w_2)$ with nonzero $\sigma^{-1}(w_1), \sigma^{-1}(w_2) \sqsubseteq v$, which contradicts $v \in \mathcal{G}(\Lambda)$. Thus, $\sigma(v)$ must belong to $\mathcal{G}(\Lambda)$. □

As a consequence of this lemma, $\mathcal{G}(\Lambda)$ decomposes completely into *full* orbits. Our task of computing $\mathcal{G}(\Lambda)$ thus reduces to computing representatives of these orbits, and to collect them into a set $\mathcal{G}^{\mathrm{sym}}(\Lambda)$.

By the previous lemma, we recover $\mathcal{G}(\Lambda)$ via

$$\mathcal{G}(\Lambda) = \bigcup_{v \in \mathcal{G}^{\mathrm{sym}}(\Lambda)} \mathrm{orb}_{\mathrm{S}_\Lambda}(v).$$

Note that this last expression does not compute a superset of $\mathcal{G}(\Lambda)$. In contrast to this, the last statement of Lemma 1.2 fails to be true in general for minimal toric Gröbner bases or for minimal Markov bases associated with the lattice Λ.

2 Computing the Graver Basis

In this section, we adapt Pottier's algorithm [11] to deal with the symmetries of Λ in the computation of $\mathcal{G}(\Lambda)$. Note that this algorithm is not the fastest way to compute Graver bases directly. However, by adapting this algorithm, it will be easier for us to exploit the given symmetries (and to present the main ideas). The state-of-the-art algorithm that is based on the positive sum property of Graver bases [5] needs to break the symmetry, see Section 3. Nonetheless, we show how to exploit the symmetries also in this situation and arrive at an even faster "symmetric" algorithm.

Algorithm 2.1 *(Algorithm to Compute $\mathcal{G}(\Lambda)$)*

Input: generating set F of Λ over $\mathbb{Z}$

Output: a set G which contains $\mathcal{G}(\Lambda)$

$G := F \cup -F$

$C := \bigcup_{f,g \in G} \{f + g\}$

while $C \neq \emptyset$ *do*

$s :=$ *an element in* C

$C := C \setminus \{s\}$

$f := \mathrm{normalForm}(s, G)$

if $f \neq 0$ *then*

$G := G \cup \{f\}$

$C := C \cup \bigcup_{g \in G} \{f + g\}$

return G.

Behind the function $\mathrm{normalForm}(s, G)$ there is the following algorithm. It aims at finding a relation $s = g_1 + \ldots + g_r$, $g_i \in G$, $g_i \sqsubseteq s$, $i = 1, \ldots, r$. The function $\mathrm{normalForm}(s, G)$ returns 0 if it found such vectors $g_i \in G$, or it returns a vector $f \in \Lambda$ such that such a desired relation exists with elements from $G \cup \{f\}$.

Algorithm 2.2 *(Normal Form Algorithm)*

Input: a vector s, a set G of vectors

Output: a normal form of s with respect to G

while there is some $g \in G$ such that $g \sqsubseteq s$ do

$s := s - g$

return s

This algorithm due to Pottier always terminates and the set $\mathcal{G}(\Lambda)$ is exactly the set of all $\sqsubseteq$-minimal vectors in the final output G. Termination is guaranteed by the following lemma, which we will employ again later.

Lemma 2.3 *(Gordan-Dickson Lemma, Sequence version, [1])*

Let $\{p_1, p_2, \ldots\}$ be a sequence of points in $\mathbb{Z}_+^n$ such that $p_i \not\leq p_j$ whenever $i < j$. Then this sequence is finite.

Now let us adapt the algorithm to exploit the symmetries. Let us start with an immediate consequence of Lemma 1.2.

Corollary 2.4 *If $s = g_1 + \ldots + g_r$, $g_i \sqsubseteq s$, $i = 1, \ldots, r$, then $\sigma(s) = \sigma(g_1) + \ldots + \sigma(g_r)$, $\sigma(g_i) \sqsubseteq \sigma(s)$, $i = 1, \ldots, r$ for every $\sigma \in \mathcal{S}_\Lambda$. Thus, if $G = \bigcup_{v \in G^{sym}} \mathrm{orb}_{\mathrm{S}_\Lambda}(v)$ and if $\mathrm{normalForm}(s, G) = 0$ then $\mathrm{normalForm}(\sigma(s), G) = 0$ for every $\sigma \in \mathcal{S}_\Lambda$.*

In other words, if a representation $s = g_1 + \ldots + g_r$, $g_i \in G$, $g_i \sqsubseteq s$, $i = 1, \ldots, r$ has been found, the symmetry of Λ already guarantees existence of a similar representation for every element in $\mathrm{orb}_{\mathrm{S}_\Lambda}(s)$.

Finally, we are in the position to exploit symmetries in Pottier's algorithm. The main differences to the original algorithm will be that instead of keeping the sets G and C in memory, we only store their representatives under the given symmetry in sets G^{sym} and C^{sym}. (At any point during the "symmetric" algorithm we may go back to the original algorithm by replacing all elements in G^{sym} and C^{sym} by the vectors from their orbits under $\mathcal{S}_\Lambda$.)

Moreover, there a few more changes. Once we have found a nonzero vector f that is to be added to G (and to G^{sym} as a new representative), we assume that we add the full orbit $\mathrm{orb}_{\mathrm{S}_\Lambda}(f)$ to G, as we know that the Graver basis would contain the full orbit if f was in fact a Graver basis element. Accordingly, instead of adding only the vectors

$$\bigcup_{g \in G} \{f + g\}$$

to C, we immediately include the vectors

$$\bigcup_{f' \in \mathrm{orb}_{S_\Lambda}(f), g \in G} \{f' + g\}.$$

As G will always be a union of full orbits, this last expression can be transformed to

$$\bigcup_{f' \in \mathrm{orb}_{S_\Lambda}(f), g \in G} \{f' + g\} = \bigcup_{g \in G^{\mathrm{sym}}} \bigcup_{\substack{f' \in \mathrm{orb}_{S_\Lambda}(f) \\ g' \in \mathrm{orb}_{S_\Lambda}(g)}} \{f' + g'\} = \bigcup_{g \in G^{\mathrm{sym}}} \bigcup_{g' \in \mathrm{orb}_{S_\Lambda}(g)} \mathrm{orb}_{S_\Lambda}(f + g').$$

Therefore, we update C^{sym} as follows:

$$C^{\mathrm{sym}} = C^{\mathrm{sym}} \cup \bigcup_{g \in G^{\mathrm{sym}}} \bigcup_{g' \in \mathrm{orb}_{S_\Lambda}(g)} \{f + g'\}.$$

Note that since

$$\bigcup_{g' \in \mathrm{orb}_{S_\Lambda}(g)} \mathrm{orb}_{S_\Lambda}(f + g') = \bigcup_{f' \in \mathrm{orb}_{S_\Lambda}(f)} \mathrm{orb}_{S_\Lambda}(f' + g)$$

we have a choice in adding either all vectors $\{f+g'\}$ or all vectors $\{f'+g\}$ to C^{sym}. Clearly, one would choose to add as few new representatives to C^{sym} as possible to keep the number of S-vectors that need to be reduced small. After all, these reductions are the most expensive part of the algorithm. In the following algorithm, $\mathrm{rep}_{S_\Lambda}(H)$ for a set $H \subseteq \Lambda$ of vectors shall denote a set of representatives of H under the symmetry group $\mathcal{S}_\Lambda$.

Algorithm 2.5 *(Algorithm to Compute $G^{sym}(\Lambda)$)*

Input: generating set F of Λ over $\mathbb{Z}$

Output: a set G which contains $\mathcal{G}(\Lambda)$

$G^{sym} := \mathrm{rep}_{S_\Lambda}(F \cup -F)$ $\qquad$ $G := \mathrm{orb}_{S_\Lambda}(F \cup -F)$

$C^{sym} := \mathrm{rep}_{S_\Lambda}\left(\bigcup_{f,g \in G} \{f + g\}\right)$ $\qquad$ $C := \bigcup_{f,g \in G} \{f + g\}$

while $C^{sym} \neq \emptyset$ *do*

$s :=$ *an element in* C^{sym}

$C^{sym} := C^{sym} \setminus \{s\}$ $\qquad$ $C := C \setminus \mathrm{orb}_{S_\Lambda}(s)$

$f := \mathrm{normalForm}(s, G^{sym}) := \mathrm{normalForm}(s, G)$ $\qquad$ $f := \mathrm{normalForm}(s, G)$

if $f \neq 0$ *then*

$G^{sym} := G^{sym} \cup \{f\}$ $\qquad$ $G := G \cup \mathrm{orb}_{S_\Lambda}(f)$

$C^{sym} := C^{sym} \cup \bigcup_{g \in G} \bigcup_{g' \in \mathrm{orb}_{S_\Lambda}(g)} \{f + g'\}$ $\qquad$ $C := C \cup \bigcup_{f' \in \mathrm{orb}_{S_\Lambda}(f), g \in G} \{f' + g\}$

return $G = \mathrm{orb}_{S_\Lambda}(G^{sym})$.

In this algorithm, we compute normalForm(s, G^{sym}) via normalForm(s, G^{sym}) := normalForm(s, G). Clearly, from a practical perspective, one would not want to keep the huge set G in memory. Then, of course, one needs to think about how to compute normalForm(s, G^{sym}) efficiently if only G^{sym} instead of G is available. (For example, G might be simply too big to be kept in memory.) This is still an open question and any significant improvement in the solution of this problem would lead to an equally significant improvement of the overall algorithm.

Lemma 2.6 *Algorithm 2.5 always terminates and returns a set G containing $\mathcal{G}(\Lambda)$.*

Proof. To prove termination, consider the sequence of vectors in $G^{\mathrm{sym}} \setminus \mathrm{rep}_{\mathrm{S}_\Lambda}(F \cup -F) = \{f_1, f_2, \ldots\}$ as they are added to G^{sym} during the run of the algorithm. By construction, we have $f_i \not\sqsubseteq f_j$, that is $(f_i^+, f_i^-) \not\leq (f_j^+, f_j^-)$ whenever $i < j$. Thus, by the Gordan-Dickson Lemma, this sequence must be finite and the algorithm terminates.

Note that throughout the run of the algorithm, we always have $G = \mathrm{orb}_{\mathrm{S}_\Lambda}(G^{\mathrm{sym}})$. Upon termination we know that normalForm$(f + g, G) = 0$ for every pair of vectors $f, g \in G$. Thus, G must contain the Graver basis $\mathcal{G}(\Lambda)$. □

3 Computing the Graver basis faster

In this section we introduce a special generating set of Λ. This set not only decreases the number of sums $f + g$ that need to be added to C, but more importantly, it reduces the amount of work to compute normalForm(s, G) tremendously. As the latter computation is the most expensive part of Pottier's algorithm, this heavily speeds up the computation of $\mathcal{G}(\Lambda)$. Moreover, we introduce a so-called critical-pair selection strategy that chooses the next element s from C according to a certain rule. This, together with the special input set, will imply that the set G returned by our algorithm is *exactly* the Graver basis $\mathcal{G}(\Lambda)$. Not a single unnecessary vector is computed!

In Pottier's algorithm, one has to wait until the very end to extract the Graver basis from the returned set G. In contrast to this, our algorithm provides a *certificate* of $\sqsubseteq$-minimality for each vector that is added to G. Consequently, at any point during the computation, a subset G of $\mathcal{G}(\Lambda)$ is known.

Let us assume from now on that Λ is generated by d vectors and that the first d components of the vectors of Λ are linearly independent, that is, the only d-dimensional vector orthogonal to the projection of Λ to the first d components is the zero vector. (This condition can easily be achieved algorithmically on a lattice basis of Λ by integer row operations and by switching components.) Let π denote the projection of an n-dimensional vector onto its first d components. As the last $n - d$ components of Λ are linearly dependent on the first d components, a vector $\pi(v) \in \pi(\Lambda)$ can be uniquely lifted back to $v \in \Lambda$. Moreover, this can easily be done algorithmically.

Next let us define a norm $\|.\|$ on vectors $v \in \Lambda$ as follows: $\|v\| := \|\pi(v)\|_1$, where $\|.\|_1$ denotes the

L_1-norm on $\mathbb{R}^d$. It can easily be checked that, under our assumptions on Λ, this defines indeed a norm on Λ. It is this norm definition that breaks existing symmetry of the given problem.

Finally, we are ready to state the algorithm behind the implementation in 4ti2 that seemingly defines the current state-of-the-art in the computation of Graver bases.

Let $\pi(\bar{F})$ denote set of all $\sqsubseteq$-minimal nonzero vectors in $\pi(\Lambda)$. Note that $\pi(\bar{F})$ has the positive sum property with respect to $\pi(\Lambda)$, that is, every vector $\pi(v) \in \pi(\Lambda)$ can be written as a positive integer linear combination $\pi(v) = \sum \alpha_i \pi(f_i)$ of vectors $\pi(f_i)$ from the set $\pi(\bar{F})$ that lie all in the same orthant as $\pi(v)$. In other words, $\alpha_i \in \mathbb{Z}_{>0}$ and $\pi(f_i) \sqsubseteq \pi(v)$ for all i in this linear combination. Clearly, this nice integer linear combination $\pi(v) = \sum \alpha_i \pi(f_i)$ can be uniquely lifted to $v = \sum \alpha_i f_i$ showing that $\bar{F}$ is in particular a generating set of Λ over $\mathbb{Z}$.

Moreover, this set $\pi(\bar{F})$ can be computed for example via Pottier's algorithm once a lattice basis F of $\pi(\Lambda) \subseteq \mathbb{Z}^d$ over $\mathbb{Z}$ is given as input. Note that all these $\sqsubseteq$-minimal vectors in $\pi(\bar{F})$ must lift to $\sqsubseteq$-minimal elements in Λ, since they are already indecomposable/minimal on the first d components. Therefore, the computation of $\pi(\bar{F})$ is usually far less expensive than computing $\mathcal{G}(\Lambda)$ itself. In our computational experiments, it often happened that this set $\pi(\bar{F})$ simply contained the unit vectors in $\mathbb{Z}^d$ and their negatives.

Algorithm 3.1 *(Faster Algorithm to Compute $\mathcal{G}(\Lambda)$)*

Input: set $\bar{F} \subseteq \Lambda$ such that $\pi(\bar{F})$ is the set of $\sqsubseteq$-minimal nonzero vectors in $\pi(\Lambda)$

Output: a set G which contains $\mathcal{G}(\Lambda)$

$G := \bar{F}$

$C := \bigcup_{f,g \in G : \pi(f) \text{ and } \pi(g) \text{ lie in the same orthant of } \mathbb{R}^d} \{f+g\}$

while $C \neq \emptyset$ *do*

$s :=$ *an element in* C *with smallest* $\|.\|$*-norm*

$C := C \setminus \{s\}$

$f := \text{normalForm}(s, G)$

if $f \neq 0$ *then*

$G := G \cup \{f\}$

$C := C \cup \bigcup_{g \in G : \pi(f) \text{ and } \pi(g) \text{ lie in the same orthant of } \mathbb{R}^d} \{f+g\}$

return G.

Due to our special input set, the function normalForm(s, G) can be sped up as follows.

Algorithm 3.2 *(Faster Normal Form Algorithm)*

Input: a vector s, a set G of vectors

Output: a normal form of s with respect to G

if there is some $g \in G$ such that $g \sqsubseteq s$ return 0

return s

Once a vector $g \in G$ with $g \sqsubseteq s$ is found, a representation $s = g_1 + \ldots + g_r$, $g_i \in G$, $g_i \sqsubseteq s$, $i = 1, \ldots, r$, must exist. Therefore, the function normalForm(s, G) can return 0 immediately without explicitly constructing such a relation.

Since we started with a very special input set, only those pairs of vectors $f, g \in G$ lead to a critical vector in C, for which the projections $\pi(f)$ and $\pi(g)$ lie in the same orthant of $\mathbb{R}^d$. Finally, let us prove our claims.

Lemma 3.3 *Algorithm 3.1 always terminates and returns a set G containing $\mathcal{G}(\Lambda)$.*

Proof. To prove termination, consider the sequence of vectors in $G \backslash \bar{F} = \{f_1, f_2, \ldots\}$ as they are added to G during the run of the algorithm. By construction, we have $f_i \not\sqsubseteq f_j$, that is $(f_i^+, f_i^-) \not\leq (f_j^+, f_j^-)$ whenever $i < j$. Thus, by the Gordan-Dickson Lemma, this sequence must be finite and the algorithm terminates.

To prove correctness, let us assume that $z \in \mathcal{G}(\Lambda)$ is not contained in the final output set G of Algorithm 3.1. Without loss of generality we may assume that z has a smallest norm $\|z\|$ among all such vectors from $\mathcal{G}(\Lambda)$. Therefore, we can assume that all Graver basis elements g with $\|g\| < \|z\|$ are contained in G. In the following, we construct a contradiction to the assumption $z \notin G$ and correctness of Algorithm 3.1 is proved.

As $\bar{F}$ is contained in G, there is a representation $z = \sum \alpha_i v_i$ with positive integers α_i and vectors $v_i \in G$ with $\pi(v_i) \sqsubseteq \pi(z)$. From the set of all such linear integer combinations choose one such that $\sum \alpha_i \|v_i\|_1$ is minimal.

Let us assume first that $\sum \alpha_i \|v_i\|_1 > \|z\|_1$. Therefore, there have to exist vectors v_{i_1}, v_{i_2} in this representation which have some component $k = k_0$ of different signs. By construction, $k_0 > d$, as the v_i have all the same sign as z on the first d components.

The vector $v_{i_1} + v_{i_2}$ was added to C during the run of Algorithm 3.1 (as $\pi(v_{i_1})$ and $\pi(v_{i_2})$ lie in the same orthant of $\mathbb{R}^d$ by construction). If $\|v_{i_1} + v_{i_2}\| = \|z\|$, then all other v_i, $i \neq i_1, i_2$, must satisfy $\|v_i\| = 0$ as $\pi(v_i) \sqsubseteq \pi(z)$ for all i. But $\pi(v_i) = 0$ implies $v_i = 0$ and thus $v_{i_1} + v_{i_2} = z$. Since $v_{i_1} + v_{i_2} (= z)$ is a vector that was added to C during the run of the algorithm, the vector z is eventually chosen as $s \in C$. Being $\sqsubseteq$-minimal, we have normalForm$(z, G) = z$ and thus, z must have been added to G, in contradiction to our assumption $z \notin G$.

Therefore, we may assume that $\|v_{i_1} + v_{i_2}\| < \|z\|$. However, since all Graver basis elements v with norm $\|v\| < \|z\|$ are assumed to be in G, there must exist a representation $v_{i_1} + v_{i_2} = \sum \beta_j v'_j$ for finitely many $\beta_j \in \mathbb{Z}_{>0}$, $v'_j \in G$, and $\beta_j v'_j \sqsubseteq v_{i_1} + v_{i_2}$ for all j. This implies that we have for each component $k = 1, \ldots, n$,

$$\sum_j \beta_j |v'^{(k)}_j| = |\sum_j \beta_j v'^{(k)}_j| = |(v_{i_1} + v_{i_2})^{(k)}| \leq |v^{(k)}_{i_1}| + |v^{(k)}_{i_2}|,$$

where the last inequality is strict for $k = k_0$ by construction. Summing up over $k = 1, \ldots, n$, yields $\sum \beta_j \|v'_j\|_1 = \|v_{i_1} + v_{i_2}\|_1 < \|v_{i_1}\|_1 + \|v_{i_2}\|_1$. But now z can be represented as

$$\begin{aligned} z &= \alpha_{i_1} v_{i_1} + \alpha_{i_2} v_{i_2} + \sum_{i \neq i_1, i_2} \alpha_i v_i \\ &= \sum \beta_j v'_j + (\alpha_{i_1} - 1) v_{i_1} + (\alpha_{i_2} - 1) v_{i_2} + \sum_{i \neq i_1, i_2} \alpha_i v_i \end{aligned}$$

and it holds

$$\sum \beta_j \|v'_j\|_1 + (\alpha_{i_1} - 1)\|v_{i_1}\|_1 + (\alpha_{i_2} - 1)\|v_{i_2}\|_1 + \sum_{i \neq i_1, i_2} \alpha_i \|v_i\|_1 < \sum \alpha_i \|v_i\|_1$$

in contradiction to the minimality required on $\sum \alpha_i \|v_i\|_1$. Thus, our assumption $\sum \alpha_i \|v_i\|_1 > \|z\|_1$ was wrong and $\sum \alpha_i \|v_i\|_1 = \|z\|_1$ must hold.

But this last equation implies that $v_i \sqsubseteq z$ for all i, contradicting $\sqsubseteq$-minimality of z unless the representation $z = \sum \alpha_i v_i$ is trivial, that is $z = v_1 \in G$. This, however, again contradicts our initial assumption $z \notin G$ and thus $\mathcal{G}(\Lambda) \subseteq G$. □

It should be noted that we did not make use of our selection strategy to prove termination and correctness of Algorithm 3.1. It is the following Lemma that provides a certificate for $\sqsubseteq$-minimality of vectors in G.

Lemma 3.4 *The set G returned by Algorithm 3.1 equals $\mathcal{G}(\Lambda)$.*

Proof. The main observation needed in this proof is that the norms $\|s\|$ of the vectors s that are chosen from C form a non-decreasing sequence. This follows from the definition that normalForm(s, g) only returns either 0 or s and from our condition that only vectors $f + g$ are added to C whose first d components have the same sign pattern. The latter implies $\|f + g\| = \|f\| + \|g\|$.

Now assume that some $z \in G$ is not contained in $\mathcal{G}(\Lambda)$. Thus, there is some $g \in \mathcal{G}(\Lambda)$ with $g \sqsubseteq z$. Under our assumptions on Λ, $g \sqsubseteq z$ implies $\|g\| < \|s\|$. Since G contains $\mathcal{G}(\Lambda)$ (and thus in particular the vector g) at the end of the algorithm and since after z only vectors f are added to G that have a norm $\|f\| \geq \|z\|$, the vector g must have been contained in G already at the time when z was added to G. This however, implies that Algorithm 3.1 must have computed normalForm$(z, G) = 0$, a contradiction to the assumption that z was added to G. □

4 Computing the symmetric Graver basis faster

Although the definition of $\|.\|$ in the previous section broke most if not all existing symmetry in the problem, we will now combine the ideas of Sections 2 and 3 to a faster algorithm to compute symmetric Graver bases. The main idea is to use the norm $\|.\|$ defined on $\mathbb{Z}^n$ to define a norm on orbits: For $T \subseteq \Lambda$, we define $\|T\| := \min\{\|v\| : v \in T\}$. Then the new symmetric Graver basis algorithm looks as follows.

Algorithm 4.1 *(Faster Algorithm to Compute $G^{sym}(\Lambda)$)*

Input: set $\bar{F} \subseteq \Lambda$ such that $\pi(\bar{F})$ is the set of $\sqsubseteq$-minimal nonzero vectors in $\pi(\Lambda)$

Output: a set G which contains $\mathcal{G}(\Lambda)$

$G^{sym} := \mathrm{rep}_{\mathrm{S}_\Lambda}(\bar{F})$

$C^{sym} := \mathrm{rep}_{\mathrm{S}_\Lambda}\left(\bigcup_{f,g \in G}\{f+g\}\right)$

while $C^{sym} \neq \emptyset$ *do*

$s :=$ *an element in* C^{sym} *with smallest value of* $\|\mathrm{orb}_{\mathrm{S}_\Lambda}(s)\|$

$C^{sym} := C^{sym} \setminus \{s\}$

$f := \mathrm{normalForm}(s, G^{sym})$

if $f \neq 0$ *then*

$G^{sym} := G^{sym} \cup \{f\}$

$C^{sym} := C^{sym} \cup \bigcup_{g \in G} \bigcup_{g' \in \mathrm{orb}_{\mathrm{S}_\Lambda}(g)} \{f + g'\}$

return $G = \mathrm{orb}_{\mathrm{S}_\Lambda}(G^{sym})$.

Due to our special input set and our norm defined on orbits, we can again simplify and speed up the normal form computation $\mathrm{normalForm}(s, G^{\mathrm{sym}}) := \mathrm{normalForm}(s, \mathrm{orb}_{\mathrm{S}_\Lambda}(G^{\mathrm{sym}}))$ by using Algorithm 3.2 instead of Algorithm 2.2. Again, from a practical perspective, one would want to compute $\mathrm{normalForm}(s, G^{\mathrm{sym}})$ without recovering or storing the huge set $\mathrm{orb}_{\mathrm{S}_\Lambda}(G^{\mathrm{sym}})$.

On the other hand, it should be noted that we do not, as in Algorithm 3.1, have an orthant condition in Algorithm 4.1 that reduces the number of vectors added to C^{sym}. This is done to "undo" the symmetry breaking caused by $\|.\|$.

Lemma 4.2 *Algorithm 4.1 always terminates and returns the set $G = \mathcal{G}(\Lambda)$.*

Proof. The subsequent proof follows similar lines as the proof of Lemma 3.3.

To prove termination, we consider again the sequence of vectors in $G^{\mathrm{sym}} \setminus \mathrm{rep}_{S_\Lambda}(\bar{F}) = \{f_1, f_2, \ldots\}$ as they are added to G^{sym} during the run of the algorithm. By construction, we have $f_i \not\sqsubseteq f_j$, that is $(f_i^+, f_i^-) \not\leq (f_j^+, f_j^-)$ whenever $i < j$. Thus, by the Gordan-Dickson Lemma, this sequence must be finite and the algorithm terminates.

Next we show that $\mathcal{G}(\Lambda) \subseteq G$. Assume on the contrary that this is not the case. Among all elements $z \in \mathcal{G}(\Lambda) \setminus G$ choose one with $\|\mathrm{orb}_{S_\Lambda}(z)\|$ smallest. Moreover, we may assume that z is a representative of $\mathrm{orb}_{S_\Lambda}(z)$ with $\|\pi(z)\|_1 = \|\mathrm{orb}_{S_\Lambda}(z)\|$. By our generating assumption on the set $\bar{F} \subseteq G$, there exists a non-trivial representation $z = \sum \alpha_i v_i$ with positive integers α_i and vectors $v_i \in G$ with $\pi(v_i) \sqsubseteq \pi(z)$. From the set of all such linear integer combinations choose one such that $\sum \alpha_i \|v_i\|_1$ is minimal. Note that from $\pi(v_i) \sqsubseteq \pi(z)$ and the fact that the relation $z = \sum \alpha_i v_i$ is non-trivial, we conclude $\|\pi(v_i)\|_1 < \|\pi(z)\|_1$ and thus $\|\mathrm{orb}_{S_\Lambda}(v_i)\| < \|\mathrm{orb}_{S_\Lambda}(z)\|$ for all i.

Let us assume first that $\sum \alpha_i \|v_i\|_1 > \|z\|_1$. Therefore, there have to exist vectors v_{i_1}, v_{i_2} in this representation which have some component $k = k_0$ of different signs. By construction, $k_0 > d$, as the v_i have all the same sign as z on the first d components.

The orbit of the vector $v_{i_1} + v_{i_2}$ was added to C^{sym} during the run of the algorithm. If $\|v_{i_1} + v_{i_2}\| = \|z\|$, then all other v_i, $i \neq i_1, i_2$ must satisfy $\|v_i\| = 0$ as $\pi(v_i) \sqsubseteq \pi(z)$ for all i. But $\pi(v_i) = 0$ implies $v_i = 0$ and thus $v_{i_1} + v_{i_2} = z$. Since $v_{i_1} + v_{i_2} (= z)$ is a vector whose orbit (or better: a representative of it) was added to C^{sym} during the run of the algorithm, a representative of the orbit of the vector z is eventually chosen as $s \in C^{\mathrm{sym}}$. Being $\sqsubseteq$-minimal, we have $\mathrm{normalForm}(\mathrm{rep}_{S_\Lambda}(\mathrm{orb}_{S_\Lambda}(z)), G^{\mathrm{sym}}) = \mathrm{rep}_{S_\Lambda}(\mathrm{orb}_{S_\Lambda}(z))$ and thus, $\mathrm{rep}_{S_\Lambda}(\mathrm{orb}_{S_\Lambda}(z))$ must have been added to G^{sym}, in contradiction to our assumption $z \notin G$.

Therefore, we may assume that $\|v_{i_1} + v_{i_2}\| < \|z\|$. However, since all Graver basis elements with norm strictly smaller than $\|z\|$ are assumed to be in G, we may continue literally as in the proof of Lemma 3.3: rewrite $z = \sum \alpha_i v_i$ and arrive at a contradiction to the minimality of $\sum \alpha_i v_i$.

Thus, we must have $\sum \alpha_i \|v_i\|_1 = \|z\|_1$, which implies that $v_i \sqsubseteq z$ for all i. As z is $\sqsubseteq$-minimal, this is only possible if the relation $z = \sum \alpha_i v_i$ is trivial, that is $z = v_1$. As $v_1 \in G$ we now also have $z \in G$ as desired.

To show that not only $\mathcal{G}(\Lambda) \subseteq G$ but in fact $\mathcal{G}(\Lambda) = G$ is true, assume $G \setminus \mathcal{G}(\Lambda) \neq \emptyset$. Among all elements $z \in G \setminus \mathcal{G}(\Lambda)$ choose one with $\|\mathrm{orb}_{S_\Lambda}(z)\|$ smallest. Moreover, we may again assume that z is a representative of $\mathrm{orb}_{S_\Lambda}(z)$ with $\|\pi(z)\|_1 = \|\mathrm{orb}_{S_\Lambda}(z)\|$. Note that by Lemma 1.2 and since $z \notin \mathcal{G}(\Lambda)$, not a single element in $\mathrm{orb}_{S_\Lambda}(z)$ belongs to $\mathcal{G}(\Lambda)$.

We now start with a non-trivial representation $z = \sum \alpha_i v_i$ with positive integers α_i and vectors $v_i \in \bar{F} \subseteq G$ with $\pi(v_i) \sqsubseteq \pi(z)$. Clearly, $\pi(v_i) \sqsubseteq \pi(z)$ implies $\|\pi(v_i)\|_1 < \|\pi(z)\|_1$ and thus $\|\mathrm{orb}_{S_\Lambda}(v_i)\| < \|\mathrm{orb}_{S_\Lambda}(z)\|$ for all i. Next, we follow similar steps as above (or more precisely: as in the proof of Lemma 3.3) to change $z = \sum \alpha_i v_i$ into a representation $z = \sum \beta_j w_j$ with $\beta_j > 0$, $w_j \in G$ and $w_j \sqsubseteq z$ for all j. As z is not $\sqsubseteq$-minimal, this representation is not trivial.

Note that in $z = \sum \alpha_i v_i$, $z = \sum \beta_j w_j$, and in any intermediate representation $z = \sum \beta_k u_k$, we have that the summands v_i, w_j, u_k have a strictly smaller norm on the first d components as z. In fact, the

same property holds for the sum of two summands that are iteratively needed for the rewriting steps. Thus, their corresponding orbits also have a strictly smaller norm than $\|\operatorname{orb}_{S_\Lambda}(z)\|$. We conclude that at the time $\operatorname{rep}_{S_\Lambda}(\operatorname{orb}_{S_\Lambda}(z))$ was chosen from C^{sym} and then added to G^{sym}, representatives for all orbits $\operatorname{orb}_{S_\Lambda}(w_j)$ were already in G^{sym}. Thus, $\operatorname{normalForm}(\operatorname{rep}_{S_\Lambda}(\operatorname{orb}_{S_\Lambda}(z)), G^{\mathrm{sym}})$ must have returned 0 in contradiction to the assumption that $\operatorname{rep}_{S_\Lambda}(\operatorname{orb}_{S_\Lambda}(z))$ was added to G^{sym}.

We conclude $G \setminus \mathcal{G}(\Lambda) = \emptyset$ and hence $G = \mathcal{G}(\Lambda)$. □

5 Computational Experiments

In this section we report on computational experience with a few examples that have symmetry. These problem deal all with 3-way tables, that is, with $k_1 \times k_2 \times k_3$ tables of unit cubes, each containing an integer number. For each problem, the lattice Λ is the set of lattice points in the kernel of the matrix that encodes the conditions that the sums along each one-dimensional row parallel to a coordinate axis are 0. Example 1.1 presented the case of 3×3 tables.

The following running times demonstrate that, as expected, our symmetric algorithm heavily speeds up the computation. (Times are given in seconds on a Sun UltraSparc III+ with 1.05 GHz.)

Problem	$\lvert\mathcal{S}_\Lambda\rvert$	Size of Graver basis	$\lvert G^{\mathrm{sym}}\rvert$	Algorithm 3.1	Algorithm 4.1
$3 \times 3 \times 3$	$1,296$	795	7	2	1
$3 \times 3 \times 4$	$1,728$	$19,722$	27	$1,176$	9
$3 \times 3 \times 5$	$8,640$	$263,610$	61	$560,517$	526
$3 \times 4 \times 4$	$6,912$	$4,617,444$	784	$--$	$260,590$

References

[1] D. Cox, J. Little, and D. O'Shea. Ideals, Varieties, Algorithms. Springer, 1992.

[2] P. Diaconis and B. Sturmfels. Algebraic algorithms for sampling from conditional distributions. Annals of Statistics **26** (1998), 363–397.

[3] J. E. Graver. On the foundation of linear and integer programming I. Mathematical Programming **9** (1975), 207–226.

[4] R. Hemmecke and R. Hemmecke. `4ti2` Version 1.1–Computation of Hilbert bases, Graver bases, toric Gröbner bases, and more. Available at `http://www.4ti2.de/`, September 2003.

[5] R. Hemmecke. On the positive sum property and the computation of Graver test sets. Mathematical Programming **96** (2003), 247–269.

[6] R. Hemmecke and R. Schultz. Decomposition of test sets in stochastic integer programming. Mathematical Programming **94** (2003), 323-341.

[7] R. Hemmecke. Test sets for integer programs with Z-convex objective function. e-print available from `http://front.math.ucdavis.edu/math.CO/0309154`, 2003.

[8] M. Jach. Gruppentheoretische Relaxierung ganzzahliger Programme. Diploma thesis, University of Magdeburg, April 2004.

[9] M. Jach, M. Köppe, and R. Weismantel. Nondecomposable group solutions and an application to polyhedral combinatorics. Manuscript, University of Magdeburg, October 2004.

[10] K. Murota, H. Saito, and R. Weismantel. Optimality criterion for a class of nonlinear integer programs. Operations Research Letters **32** (2004), 468–472.

[11] L. Pottier. Euclide's algorithm in dimension n. Research report, ISSAC **96**, ACM Press, 1996.

[12] B. Sturmfels. Gröbner Bases and Convex Polytopes. American Mathematical Society, Providence, Rhode Island, 1995.

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