# The Project-And-Lift Algorithm for the Computation of Toric Gröbner Bases

## An Implementation in Mathematica

Stephan Ritscher

Department of Computer Science
TU München

13th of April 2010

# Outline

# Integer Linear Programming

## Standard Form

$$\min\{cy : Ay = b, y \in \mathbb{N}^n\}$$

## Selected Applications

- Travelling Salesman
- Knapsack, Bin Packing
- Schedule Optimization
- Frequency Planning for Mobile Phone Networks
- Capacity Planning for Telecommunication Networks

# Applying Gröbner-Bases

## Definition (Test Set)

$T$ is a test set for $\min\{cy : Ay = b, y \in \mathbb{N}^n\}$ if

1. all $y \in T$ have negative cost ($cy < 0$),
2. all $y \in T$ solve $Ay = 0$ and
3. for each non-optimal solution $y_s \in \mathbb{N}^n$ of $Ay_s = b$ there is a $y \in T$ s.t. $y_s + y \in \mathbb{N}^n$.

# Applying Gröbner-Bases

## Definition (Test Set)

$T$ is a test set for $\min\{cy : Ay = b, y \in \mathbb{N}^n\}$ if

1. all $y \in T$ have negative cost ($cy < 0$),
2. all $y \in T$ solve $Ay = 0$ and
3. for each non-optimal solution $y_s \in \mathbb{N}^n$ of $Ay_s = b$ there is a $y \in T$ s.t. $y_s + y \in \mathbb{N}^n$.

## Optimization Algorithm

1. Find *any* solution $y_s \in \mathbb{N}^n$ of $Ay_s = b$.
2. While $\exists y \in T : y_s + y \in \mathbb{N}^n$, let $y_s \leftarrow y_s + y$.
3. Return $y_s$.

# Applying Gröbner-Bases

## Definition (Test Set)

$T$ is a test set for $\min\{cy : Ay = b, y \in \mathbb{N}^n\}$ if

1. all $y \in T$ have negative cost ($cy < 0$),
2. all $y \in T$ solve $Ay = 0$ and
3. for each non-optimal solution $y_s \in \mathbb{N}^n$ of $Ay_s = b$ there is a $y \in T$ s.t. $y_s + y \in \mathbb{N}^n$.

## Optimization Algorithm

1. Find *any* solution $y_s \in \mathbb{N}^n$ of $Ay_s = b$.
2. While $\exists y \in T : y_s + y \in \mathbb{N}^n$, let $y_s \leftarrow y_s + y$.
3. Return $y_s$.

How can we find a *finite* (small) test set???

# Polynomial Ideals

## Notation

- $\mathbb{K}[X]$ = ring of polynomials over $\mathbb{K}$ in variables $X = \{x_1, \ldots, x_n\}$
- $I \subset \mathbb{K}[X]$ is an ideal iff for all $a, b \in I, r \in \mathbb{K}[X]$
  1. $a + b \in I$ and
  2. $ar \in I$.
- $\langle f_1, \ldots, f_s \rangle$ = ideal generated by $f_1, \ldots, f_s \in \mathbb{K}[X]$

# Polynomial Ideals

## Notation

- $\mathbb{K}[X]$ = ring of polynomials over $\mathbb{K}$ in variables $X = \{x_1, \dots, x_n\}$
- $I \subset \mathbb{K}[X]$ is an ideal iff for all $a, b \in I, r \in \mathbb{K}[X]$
  1. $a + b \in I$ and
  2. $ar \in I$.
- $\langle f_1, \dots, f_s \rangle$ = ideal generated by $f_1, \dots, f_s \in \mathbb{K}[X]$

## Example

- $\mathbb{Q}[x_1, x_2, x_3] \ni x_1^2 - x_2x_3, x_1x_2^2x_3 - 1, x_2^5x_3^3 - 1$
- $\left\langle x_1^2 - x_2x_3, x_1x_2^2x_3 - 1, x_2^5x_3^3 - 1 \right\rangle \ni x_1 - x_2^3x_3^2 = x_2^2x_3(x_1^2 - x_2x_3) - x_1(x_1x_2^2x_3 - 1)$

# Monomial Orderings

## Definition

A total ordering $\prec$ of the monomials $x^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ is admissible iff for all $\alpha, \beta, \gamma \in \mathbb{N}^n$ ($0 \in \mathbb{N}$)

1. $x^\alpha \prec x^\beta \Rightarrow x^\alpha x^\gamma \prec x^\beta x^\gamma$ and
2. $1 \prec x^\alpha$ for $x^\alpha \neq 1$.

$\mathrm{LM}(f)$ = largest monomial of $f \in \mathbb{K}[X]$ wrt. $\prec$

# Monomial Orderings (2)

## Example

- Lexicographic Ordering:
  $x^\alpha \prec x^\beta$ iff first nonzero entry of $\alpha - \beta$ is negative.

$$x_1 \succ x_2^3 x_3^2 \succ x_2^2 x_3^3$$

- Graded Reverse Lexicographic Ordering:
  $x^\alpha \prec x^\beta$ iff $\deg(x^\alpha) < \deg(x^\beta)$ or $\deg(x^\alpha) = \deg(x^\beta)$ and last nonzero entry of $\alpha - \beta$ is positive.

$$x_2^3 x_3^2 \succ x_2^2 x_3^3 \succ x_1$$

# Monomial Orderings (3)

## Definition (Matrix Orderings)

Given a matrix $C \in \mathbb{K}^{s,n}$, let $x^\alpha \prec x^\beta$ iff the first nonzero entry of $C\alpha - C\beta$ is positive.

# Monomial Orderings (3)

## Definition (Matrix Orderings)

Given a matrix $C \in \mathbb{K}^{s,n}$, let $x^\alpha \prec x^\beta$ iff the first nonzero entry of $C\alpha - C\beta$ is positive.

## Notes

Any admissible monomial ordering can be

- represented by a matrix $C \in \mathbb{R}^{n,n}$.
- approximated up to an arbitrary, fixed degree by a matrix $C \in \mathbb{Z}^{n,n}$.

Common monomial orderings can be represented by a matrix $C \in \mathbb{Z}^{n,n}$

# Monomial Orderings (3)

## Definition (Matrix Orderings)

Given a matrix $C \in \mathbb{K}^{s,n}$, let $x^\alpha \prec x^\beta$ iff the first nonzero entry of $C\alpha - C\beta$ is positive.

## Notes

Any admissible monomial ordering can be

- represented by a matrix $C \in \mathbb{R}^{n,n}$.
- approximated up to an arbitrary, fixed degree by a matrix $C \in \mathbb{Z}^{n,n}$.

Common monomial orderings can be represented by a matrix $C \in \mathbb{Z}^{n,n}$

## Example ($n = 3$)

Lexicographic Ordering: $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Graded Reverse-lexicographic Ordering: $\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{pmatrix}$

# Gröbner Bases

## Definition

$G$ is a Gröbner basis of $I$ wrt. a monomial ordering $\prec$ iff

1. $I = \langle G \rangle$ and
2. $\langle \mathrm{LM}(I) \rangle = \langle \mathrm{LM}(G) \rangle$.

# Gröbner Bases

### Definition

$G$ is a Gröbner basis of $I$ wrt. a monomial ordering $\prec$ iff

1. $I = \langle G \rangle$ and
2. $\langle \mathrm{LM}(I) \rangle = \langle \mathrm{LM}(G) \rangle$.

### Example

Consider $I = \left\langle x_1^2 - x_2x_3, x_1x_2^2x_3 - 1, x_2^5x_3^3 - 1 \right\rangle$ and the lexicographic monomial ordering.

- $G_1 = \left\{x_1^2 - x_2x_3, x_1x_2^2x_3 - 1, x_2^5x_3^3 - 1\right\}$ is no Gröbner basis of $I$ since $x_1 \in \langle \mathrm{LM}(I) \rangle$ but $x_1 \notin \langle \mathrm{LM}(G_1) \rangle$
- $G_2 = \left\{x_1^2 - x_2x_3, x_1x_2^2x_3 - 1, x_2^5x_3^3 - 1, x_1 - x_2^3x_3^2\right\}$ is a Gröbner basis of $I$.

# Problem Statement

## Definition (Toric Ideals)

Given a matrix $A \in \mathbb{Z}^{k,n}$, the associated toric ideal is defined as

$$I(A) = \left\langle x^{\alpha} - x^{\beta} : \alpha, \beta \in \mathbb{N}^{n} : \alpha - \beta \in \ker(A) \right\rangle$$

# Problem Statement

## Definition (Toric Ideals)

Given a matrix $A \in \mathbb{Z}^{k,n}$, the associated toric ideal is defined as

$$I(A) = \left\langle x^\alpha - x^\beta : \alpha, \beta \in \mathbb{N}^n : \alpha - \beta \in \ker(A) \right\rangle$$

## Example

$A = \begin{pmatrix} 1 & -3 & 5 \end{pmatrix}$

$\Rightarrow \; I(A) = \langle x_2^5 x_3^3 - 1, x_1 - x_2^3 x_3^2 \rangle$

# Problem Statement

## Definition (Toric Ideals)

Given a matrix $A \in \mathbb{Z}^{k,n}$, the associated toric ideal is defined as

$$I(A) = \left\langle x^\alpha - x^\beta : \alpha, \beta \in \mathbb{N}^n : \alpha - \beta \in \ker(A) \right\rangle$$

## Example

$A = \begin{pmatrix} 1 & -3 & 5 \end{pmatrix}$

$\Rightarrow \; I(A) = \langle x_2^5 x_3^3 - 1, x_1 - x_2^3 x_3^2 \rangle$

## Task

Given

- matrix $A \in \mathbb{Z}^{k,n}$ and
- monomial ordering $\prec$ (as matrix)

compute a Gröbner basis of $I(A)$ wrt. $\prec$.

# "Blue Print" of an Toric Ideal Algorithm

**Input**: Matrix A, monomial ordering $\prec$ defined by matrix $C$
**Output**: Gröbner basis of $I(A)$
Calculate lattice basis $B$ of $\ker_{\mathbb{Z}}(A)$
Compute Markov basis $M$ of $\ker_{\mathbb{Z}}(A)$ resp. ideal basis $F$ of $I(A)$
Compute Gröbner basis of $I(A)$

# Lattice Basis

## Definition

A lattice $L$ is a set of the form

$$L = \mathbb{Z}v_1 + \mathbb{Z}v_2 + \ldots + \mathbb{Z}v_s \subset \mathbb{Z}^n.$$

The set $\{v_1, \ldots, v_s\}$ is a lattice basis of $L$.

# Lattice Basis

## Definition

A lattice $L$ is a set of the form

$$L = \mathbb{Z}v_1 + \mathbb{Z}v_2 + \ldots + \mathbb{Z}v_s \subset \mathbb{Z}^n.$$

The set $\{v_1, \ldots, v_s\}$ is a lattice basis of $L$.

## Example

$A = \begin{pmatrix} 1 & -3 & 5 \end{pmatrix}$

$\Rightarrow \ker_{\mathbb{Z}}(A) = \mathbb{Z}\begin{pmatrix} 2 & -1 & -1 \end{pmatrix}^T + \mathbb{Z}\begin{pmatrix} 1 & 2 & 1 \end{pmatrix}^T$

# Lattice Basis

## Definition

A lattice $L$ is a set of the form

$$L = \mathbb{Z}v_1 + \mathbb{Z}v_2 + \ldots + \mathbb{Z}v_s \subset \mathbb{Z}^n.$$

The set $\{v_1, \ldots, v_s\}$ is a lattice basis of $L$.

## Example

$A = \begin{pmatrix} 1 & -3 & 5 \end{pmatrix}$

$\Rightarrow \ker_{\mathbb{Z}}(A) = \mathbb{Z}\begin{pmatrix} 2 & -1 & -1 \end{pmatrix}^T + \mathbb{Z}\begin{pmatrix} 1 & 2 & 1 \end{pmatrix}^T$

## Computation

- Since $A \in \mathbb{Z}^{k,n}$, $L = \ker_{\mathbb{Z}}(A)$ is a lattice.

1. Triangulate $A$ with unimodular operations (Hermite decomposition).
2. Lattice basis $B$ of $L = \ker_{\mathbb{Z}}(A)$ can be read off the triangular form.

# Markov Basis

Let $\alpha^+$ be defined by $\alpha_i^+ = \max\{\alpha_i, 0\}$ and $\alpha^- = (-\alpha)^+$.

## Definition

Given a lattice $L = \ker_{\mathbb{Z}}(A)$, $B$ is a Markov basis of $L$ iff $J(B) = \{x^{\alpha^+} - x^{\alpha^-} : \alpha \in B\}$ is an ideal basis of $I(A)$.

# Markov Basis

Let $\alpha^+$ be defined by $\alpha_i^+ = \max\{\alpha_i, 0\}$ and $\alpha^- = (-\alpha)^+$.

## Definition

Given a lattice $L = \ker_{\mathbb{Z}}(A)$, $B$ is a Markov basis of $L$ iff $J(B) = \{x^{\alpha^+} - x^{\alpha^-} : \alpha \in B\}$ is an ideal basis of $I(A)$.

## Example

$J(\begin{pmatrix}2 & -1 & -1\end{pmatrix}^T, \begin{pmatrix}1 & 2 & 1\end{pmatrix}^T) = \{x_1^2 - x_2x_3, x_1x_2^2x_3 - 1\}$
is a Markov basis of $A = \begin{pmatrix}1 & -3 & 5\end{pmatrix}$

# Markov Basis

Let $\alpha^+$ be defined by $\alpha_i^+ = \max\{\alpha_i, 0\}$ and $\alpha^- = (-\alpha)^+$.

## Definition

Given a lattice $L = \ker_{\mathbb{Z}}(A)$, $B$ is a Markov basis of $L$ iff $J(B) = \{x^{\alpha^+} - x^{\alpha^-} : \alpha \in B\}$ is an ideal basis of $I(A)$.

## Example

$J(\begin{pmatrix}2 & -1 & -1\end{pmatrix}^T, \begin{pmatrix}1 & 2 & 1\end{pmatrix}^T) = \{x_1^2 - x_2x_3, x_1x_2^2x_3 - 1\}$
is a Markov basis of $A = \begin{pmatrix}1 & -3 & 5\end{pmatrix}$

## Computation

1. Saturation algorithm (Sturmfels, Bigatti et. al)
2. Shadow algorithm (Kesh, Metha)
3. Project-and-Lift algorithm (Hemmecke, Malkin)

# Saturation Algorithm

### Definition

$I : f^\infty = \left\{ h \in \mathbb{K}[X] : f^k h \in I \text{ for some } k \in \mathbb{N} \right\}$ saturation of $I$ wrt. $f$.

# Saturation Algorithm

## Definition

$I : f^\infty = \left\{ h \in \mathbb{K}[X] : f^k h \in I \text{ for some } k \in \mathbb{N} \right\}$ saturation of $I$ wrt. $f$.

## Fact

Given a basis $B$ of the lattice $L = \ker_{\mathbb{Z}}(A)$, $\langle J(B) \rangle : (x_1 \cdots x_n)^\infty = I(A)$.

# Saturation Algorithm

## Definition

$I : f^\infty = \left\{ h \in \mathbb{K}[X] : f^k h \in I \text{ for some } k \in \mathbb{N} \right\}$ saturation of $I$ wrt. $f$.

## Fact

Given a basis $B$ of the lattice $L = \ker_{\mathbb{Z}}(A)$, $\langle J(B) \rangle : (x_1 \cdots x_n)^\infty = I(A)$.

## Computation

- $\langle g_1, \ldots, g_s \rangle : f^\infty = \langle g_1, \ldots, g_s, tf - 1 \rangle \cap \mathbb{K}[X]$

$\Rightarrow$ Compute Gröbner basis of $\langle g_1, \ldots, g_s, tx_1 \cdots x_n - 1 \rangle$ wrt. elimination ordering (e.g. lexicographic).

$\lightning$ Extra variable necessary.

# Saturation Algorithm (2)

## Optimization

$I$ $\omega$-graded iff for all $x^\alpha - x^\beta \in I : \omega \cdot \alpha = \omega \cdot \beta$, $\omega_i > 0$

- If $I$ is $\omega$-graded, then $I : x_k^\infty = \langle \mathrm{GB}(I, \prec_{\omega,k}) : x_k^\infty \rangle$.
- $I : (x_1 \cdots x_n)^\infty = (\cdots((I : x_1^\infty) : x_2^\infty) \cdots) : x_n^\infty)$

# Saturation Algorithm (2)

## Optimization

$I$ $\omega$-graded iff for all $x^\alpha - x^\beta \in I : \omega \cdot \alpha = \omega \cdot \beta$, $\omega_i > 0$

- If $I$ is $\omega$-graded, then $I : x_k^\infty = \langle \mathrm{GB}(I, \prec_{\omega,k}) : x_k^\infty \rangle$.
- $I : (x_1 \cdots x_n)^\infty = (\cdots((I : x_1^\infty) : x_2^\infty) \cdots) : x_n^\infty)$

**Input**: $F = J(B)$ for basis $B$ of lattice $L$
**Output**: $G =$ ideal basis of $I(L)$
$G \leftarrow F$
**for** $k \leftarrow 1$ **to** $n$ **do**
  $G \leftarrow \mathrm{GB}(G, \prec_{\omega,k}) : x_k^\infty$
**end**

# Saturation Algorithm (2)

## Optimization

$I$ $\omega$-graded iff for all $x^\alpha - x^\beta \in I : \omega \cdot \alpha = \omega \cdot \beta$, $\omega_i > 0$

- If $I$ is $\omega$-graded, then $I : x_k^\infty = \langle \mathrm{GB}(I, \prec_{\omega,k}) : x_k^\infty \rangle$.
- $I : (x_1 \cdots x_n)^\infty = (\cdots((I : x_1^\infty) : x_2^\infty) \cdots) : x_n^\infty)$

**Input**: $F = J(B)$ for basis $B$ of lattice $L$
**Output**: $G =$ ideal basis of $I(L)$
$G \leftarrow F$
**for** $k \leftarrow 1$ **to** $n$ **do**
    $G \leftarrow \mathrm{GB}(G, \prec_{\omega,k}) : x_k^\infty$
**end**

- Can be optimized to maximal $\frac{n}{2}$ saturation steps.

↯ $\frac{n}{2}$ Gröbner basis computations in $n$ variables.

↯ Only for graded ideals, otherwise extra variable.

# Shadow Algorithm

## Projections

Let $\pi_k$ be the projection on the first $k$ variables $(x_{k+1}, \ldots, x_n \leftarrow 1)$.

- Further optimizes Saturation Algorithm
- $G = \mathrm{GB}_k(I, \prec_{\omega,k}) \subset I$ such that $\pi_k(G) = \mathrm{GB}(\pi_k(I), \prec_{\omega,k})$

# Shadow Algorithm

## Projections

Let $\pi_k$ be the projection on the first $k$ variables $(x_{k+1}, \ldots, x_n \leftarrow 1)$.

- Further optimizes Saturation Algorithm
- $G = \mathrm{GB}_k(I, \prec_{\omega,k}) \subset I$ such that $\pi_k(G) = \mathrm{GB}(\pi_k(I), \prec_{\omega,k})$

**Input**: $F = J(B)$ for basis $B$ of lattice $L$
**Output**: $G =$ ideal basis of $I(L)$
$G \leftarrow F$
**for** $k \leftarrow 1$ **to** $n$ **do**
    $G \leftarrow \mathrm{GB}_k(G, \prec_{\omega,k}) : x_k^\infty$
**end**

# Shadow Algorithm

## Projections

Let $\pi_k$ be the projection on the first $k$ variables $(x_{k+1}, \ldots, x_n \leftarrow 1)$.

- Further optimizes Saturation Algorithm
- $G = \mathrm{GB}_k(I, \prec_{\omega,k}) \subset I$ such that $\pi_k(G) = \mathrm{GB}(\pi_k(I), \prec_{\omega,k})$

**Input**: $F = J(B)$ for basis $B$ of lattice $L$
**Output**: $G =$ ideal basis of $I(L)$
$G \leftarrow F$
**for** $k \leftarrow 1$ **to** $n$ **do**
$\quad G \leftarrow \mathrm{GB}_k(G, \prec_{\omega,k}) : x_k^\infty$
**end**

↯ Only for graded ideals, otherwise extra variable.

# Project-and-Lift Algorithm

## Markov Bases

- Also uses projections: $\pi_\sigma$ projects onto $\mathbb{K}[x_i : i \in \sigma]$.
- Instead of computing Gröbner bases, compute Markov bases.
- Sometimes we can simply add a polynomial to lift to larger projection.
- In remaining cases ideal is graded, so no extra variable.

# Project-and-Lift Algorithm (2)

**Input**: Basis $B$ of lattice $L$
**Output**: $M$ = Markov basis of $L$
Find (maximal) $\sigma \subset \{1, \dots, n\}$ s.t. $\ker(\pi_\sigma(L)) = \{0\}$.
Find $M \subset L$ s.t. $\pi_\sigma(M)$ is Markov basis of $\pi_\sigma(L)$.
**while** $\exists k \in \overline{\sigma} = \{1, \dots, n\} \setminus \sigma$ **do**
  **if** $\exists u \in L : \pi_\sigma(u) \geq 0, u_k > 0$ **then**
    $M \leftarrow M \cup \{u\}$
  **else**
    Find $c \in \mathbb{Q}_+^n$ s.t. $\pi_{\overline{\sigma}}(c) = 0$ and $\forall u \in L : c \cdot u = -u_k$.
    $M \leftarrow \mathrm{GB}_\sigma(M, \prec_c)$
  **end**
  $\sigma \leftarrow \sigma \cup \{k\}$
**end**

# Project-and-Lift Algorithm (3)

## Optimizations

- Preprocess lattice $L$ with `LatticeReduce` $\Rightarrow$ short vectors.
- Use `Minimize` or extreme ray algorithm in order to find $u \in L : \pi_\sigma(u) \geq 0, u_k > 0 \Rightarrow$ short vectors.
- Lift variables $x_k$ last for which no Gröbner basis computation is necessary.

# S-Polynomial

## Definition

$$\mathrm{S}(f, g, \prec) = \frac{\mathrm{LM}(g)}{\mathrm{gcd}(\mathrm{LM}(f), \mathrm{LM}(g))} \cdot f - \frac{\mathrm{LM}(f)}{\mathrm{gcd}(\mathrm{LM}(f), \mathrm{LM}(g))} \cdot g$$

## Example

$$S(x_1^2 - x_2x_3, x_1x_2^2x_3 - 1) = x_2^2x_3(x_1^2 - x_2x_3) - x_1(x_1x_2^2x_3 - 1) = x_1 - x_2^3x_3^2$$

# Reduction

## $\mathrm{Reduce}(f, F, \prec)$

**Input**: Polynomial $f$, $F = \{f_1, \ldots, f_s\}$, monomial ordering $\prec$
**Output**: $h \equiv f \mod \langle f_1, \ldots, f_s\rangle$ s.t. no monomial of $h$ is in $\langle \mathrm{LM}(F)\rangle$

$h \to 0$
**while** $f \neq 0$ **do**
  **if** $\exists f_i : \mathrm{LM}(f_i) \mid \mathrm{LM}(f)$ **then**
    $f \to f - \frac{\mathrm{LM}(f)}{\mathrm{LM}(f_i)} f_i$
  **else**
    $h \to h + \mathrm{LM}(f)$
    $f \to f - \mathrm{LM}(f)$
  **end**
**end**

### Example

$\mathrm{Reduce}(x_1x_2^2x_3, \{x_1 - x_2^3x_3^2, x_2^5x_3^3 - 1\}, \prec_{\mathrm{lex}})$: $x_1x_2^2x_3 \xrightarrow{x_1 - x_2^3x_3^2} x_2^5x_3^3 \xrightarrow{x_2^5x_3^3 - 1} 1$

# Buchberger Algorithm

**Input**: Basis $F$ of ideal $I$, monomial ordering $\prec$
**Output**: $G$ = Gröbner basis of $I$

$G \leftarrow F$
$\mathit{Pairs} \leftarrow \{(i,j) : 1 \leq i < j \leq \#G\}$
**while** $\exists (i,j) \in \mathit{Pairs}$ **do**
    $\mathit{Pairs} \leftarrow \mathit{Pairs} \setminus \{(i,j)\}$
    $h \leftarrow \mathrm{S}(g_i, g_j, \prec)$
    $h \leftarrow \mathrm{Reduce}(h, G, \prec)$
    **if** $h \neq 0$ **then**
        $G \leftarrow G \cup \{h\}$
        $\mathit{Pairs} \leftarrow \mathit{Pairs} \cup \{(i, \#G) : 1 \leq i < \#G\}$
    **end**
**end**

# Geometric Buchberger Algorithm

## Criteria

- Buchberger: Discard $(i, j)$ if $\mathrm{LM}(g_i), \mathrm{LM}(g_j)$ relatively prime.
- Gebauer-Möller: Basis of syzygies is sufficient. $\Rightarrow$ Discard superfluous pairs (only approximation, still expensive).
- Bigatti,Hemmecke-Malkin: Criterion-Tail (only for toric ideals) Discard $(i, j)$ if $S(g_i, g_j)$ reduces to polynomial with smaller grading.

# Geometric Buchberger Algorithm

## Criteria

- Buchberger: Discard $(i, j)$ if $\mathrm{LM}(g_i), \mathrm{LM}(g_j)$ relatively prime.
- Gebauer-Möller: Basis of syzygies is sufficient. $\Rightarrow$ Discard superfluous pairs (only approximation, still expensive).
- Bigatti,Hemmecke-Malkin: Criterion-Tail (only for toric ideals) Discard $(i, j)$ if $S(g_i, g_j)$ reduces to polynomial with smaller grading.

## Vector Arithmetic

- Use vectors $\alpha - \beta$ instead of binomials $x^\alpha - x^\beta$.
- Automatically cancels common factors (i.e. partly saturates).
- Only works for toric ideals.

## Example

$$x_1^3 x_2^2 x_3 - x_1 x_2^5 x_3^3 \longrightarrow \begin{pmatrix} 2 & -3 & -2 \end{pmatrix}^T \longrightarrow x_1^2 - x_2^3 x_3^2$$

# Geometric Buchberger Algorithm (2)

## Data Structures

- Originally, $G$ is a vector.
- Many searches in $G$ for $\mathrm{LM}(g_i)|m$ for given monomial $m$.
- Vector notation: search $g_i^+ \leq m$.

$\Rightarrow$ Use multidimensional search data structure.

# Geometric Buchberger Algorithm (2)

## Data Structures

- Originally, $G$ is a vector.
- Many searches in $G$ for $\mathrm{LM}(g_i)|m$ for given monomial $m$.
- Vector notation: search $g_i^+ \leq m$.

$\Rightarrow$ Use multidimensional search data structure.

## Suggestions

- kd-Lists
- kd-Trees

# Geometric Buchberger Algorithm (3)

## kd-Lists

- First level: sorted array of first coordinates
- Each entry points to substructure representing all elements with this value.
- $i$-th level: sorted arrays of $i$-th coordinates of resp. substructure

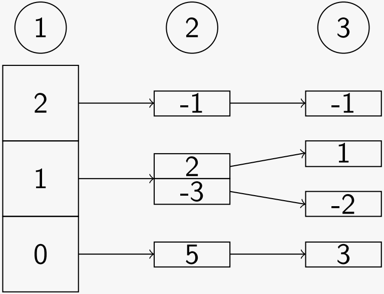


Figure: kd-List for $\begin{pmatrix}2 & -1 & -1\end{pmatrix}$, $\begin{pmatrix}1 & 2 & 1\end{pmatrix}$, $\begin{pmatrix}0 & 5 & 3\end{pmatrix}$, $\begin{pmatrix}1 & -3 & -2\end{pmatrix}$

# Geometric Buchberger Algorithm (4)

## kd-Trees

- Each level: two subtrees
- Vectors stored at leaves
- Equally balances if possible

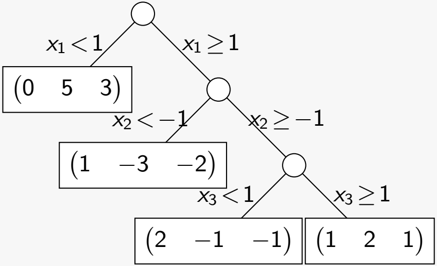


Figure: kd-Tree for $\begin{pmatrix} 2 & -1 & -1 \end{pmatrix}$, $\begin{pmatrix} 1 & 2 & 1 \end{pmatrix}$, $\begin{pmatrix} 0 & 5 & 3 \end{pmatrix}$, $\begin{pmatrix} 1 & -3 & -2 \end{pmatrix}$

# Experimental Results

| Input | P&L (uses LR) List | KDL | KDT | ToricGB | ToricGB + LR |
|---|---|---|---|---|---|
| Random 1x5 | 0.06 | 0.12 | 0.11 | 0.00 | 0.00 |
| Random 2x5 | 0.25 | 0.48 | 0.42 | 0.01 | 0.01 |
| Random 3x5 | 2.45 | 3.42 | 3.54 | 0.35 | 0.31 |
| Random 4x5 | 0.74 | 1.00 | 1.00 | 2.90 | 0.01 |
| Random 5x5 | 0.11 | 0.22 | 0.22 | 0.00 | 0.00 |
| Frobenius 6x 10 dig. | 7.56 | 26.86 | 14.33 | 429.76 | 0.49 |
| Frobenius 7x 10 dig. | 168.33 | 714.25 | 293.49 | 383.29 | 56.80 |
| Frobenius 8x 5 dig. | 22.87 | 64.75 | 44.31 | 5.55 | 1.25 |
| Frobenius 10x 4 dig. | 24.98 | 50.32 | 43.51 | 3.38 | 1.56 |

Attention:

- ToricGB is implemented natively, P&L not.
- Statistic biased by memory leaks.

# Conclusion

## Data Structures

- Performance gain possible if implemented natively.
- kd-Lists can degenerate, thus kd-trees more promising

# Conclusion

## Data Structures

- Performance gain possible if implemented natively.
- kd-Lists can degenerate, thus kd-trees more promising

## Mathematica Implementation

- Add Criterion Tail to `GroebnerBasis‘ToricGroebnerBasis`.
- Add capability of projections to `GroebnerBasis‘ToricGroebnerBasis`.
- Use Project-and-Lift algorithm for saturation.

Thank you!