Gröbner bases, Graver bases and Integer Optimization

María Isabel Hartillo
hartillo@us.es

Seville, 23th June

Gröbner bases

What are Gröbner bases?

- They are a representation of an algebraic object
- It gives a general method of computing with multivariate polynomials
- It generalises well-known methods:
  - Gaussian elimination
  - Euclidean algorithm

Notation

- k field (often algebraically closed)
- $\mathbf{x}^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ monomial in $x_1, \ldots, x_n$
- $c\mathbf{x}^\alpha$ $c \in k$ term in $x_1, \ldots, x_n$
- $f = \sum_{\alpha \in \mathbb{Z}_+^n} c_\alpha \mathbf{x}^\alpha$ polynomial in n variables
- $k[\mathbf{x}] = k[x_1, \ldots, x_n]$ polynomial ring in n variables
- $\mathbb{A}^n = \mathbb{A}^n(k)$ affine space over k

Monomial orders

Definition

A monomial order is a total order $>$ on the sets of monomials $\mathbf{x}^\alpha$ such that:

- If $\mathbf{x}^\alpha > \mathbf{x}^\beta$ and $\gamma \in \mathbb{Z}_+^n$ then $\mathbf{x}^{\alpha+\gamma} > \mathbf{x}^{\beta+\gamma}$
- Any nonempty subset of monomials has a smallest element under $>$ (it is a well-ordering)

We often think of a monomial order as a total order on the set of exponent vectors $\alpha \in \mathbb{Z}_+^n$.
Being a well-ordering is equivalent to $\alpha > 0$, for $\alpha \neq 0$ (or $\mathbf{x}^\alpha > 1$)

Examples of monomial orders

Examples

- **Lexicographic order** For every $\alpha, \beta \in \mathbb{Z}^n_+$ we say $\alpha >_{lex} \beta$ (or $\mathbf{x}^\alpha >_{lex} \mathbf{x}^\beta$) if the leftmost nonzero entry of $\alpha - \beta$ is positive. That is:

$$\alpha_1 > \beta_1, \quad \text{or} \quad \alpha_1 = \beta_1 \quad \text{and} \quad \alpha_2 > \beta_2, \ldots$$

$(1, 2, 0) >_{lex} (1, 1, 3)$

- **Graded lex order** For every $\alpha, \beta \in \mathbb{Z}^n_+$ we say $\alpha >_{grlex} \beta$ if

$$|\alpha| = \sum_{i=1}^n \alpha_i > |\beta| = \sum_{i=1}^n \beta_i, \quad \text{or} \quad |\alpha| = |\beta| \quad \text{and} \quad \alpha >_{lex} \beta$$

$(0, 2, 5) >_{grlex} (1, 2, 3)$

Using SAGE and monomial orders

```
sage: P.<x,y,z> = PolynomialRing(QQ, 3, order='lex')
sage: x > y
True
sage: x > y^2
True
sage: x > 1
True
sage: x^1*y^2 > y^3*z^4
True
sage: x^3*y^2*z^4 < x^3*y^2*z^1
False
```

Using SAGE and monomial orders

```
sage: P.<x,y,z> = PolynomialRing(QQ, 3, order='deglex')
sage: x > y
True
sage: x > y^2*z
False
sage: x > 1
True
sage: x^1*y^2*z^3 > x^3*y^2*z^0
True
sage: x^2*y*z^2 > x*y^3*z
True
```

Examples of monomial orders

More examples

- **Graded reverse lex order**For every $\alpha, \beta \in \mathbb{Z}_+^n$ we say $\alpha >_{grevlex} \beta$ if

$$|\alpha| = \sum_{i=1}^{n} \alpha_i > |\beta| = \sum_{i=1}^{n} \beta_i, \quad \text{or} \quad |\alpha| = |\beta| \quad \text{and}$$

  the rightmost nonzero entry of $\alpha - \beta$ is negative
  $(5, 2, 0) >_{grevlex} (2, 2, 3)$
- **Vector induced order** Let $c \in \mathbb{R}_+^n$ for every $\alpha, \beta \in \mathbb{Z}_+^n$ we say $\alpha >_c \beta$ if

$$c'\alpha > c'\beta, \quad \text{or} \quad c'\alpha = c'\beta \quad \text{and} \quad \alpha >_{lex} \beta$$

  If $c = (2, 3, 1)$ then $(0, 3, 0) >_c (2, 1, 1)$

Using SAGE and monomial orders

```
sage: P.<x,y,z>=PolynomialRing(QQ,3,order='degrevlex')
sage: x > y
True
sage: x > y^2*z
False
sage: x > 1
True
sage: x^1*y^5*z^2 > x^4*y^1*z^3
True
sage: x^2*y*z^2 > x*y^3*z
False
```

Using SAGE and monomial orders

```
sage: P.<x,y,z> =
PolynomialRing(QQ,3,order=TermOrder('wdeglex',(1,2,3)))
sage: x > y
False
sage: x > x^2
False
sage: x > 1
True
sage: x^1*y^2 > x^2*z
False
sage: y*z > x^3*y
False
```

Using SAGE and monomial orders

```
sage: P.<x,y,z>=
PolynomialRing(QQ,3,order=TermOrder('wdegrevlex',(1,2,3)))
sage: x > y
False
sage: x > x^2
False
sage: x > 1
True
sage: x^1*y^2 > x^2*z
True
sage: y*z > x^3*y
False
```

Examples of monomial orders

More examples

- **Matrix induced order** Let M be a square $n \times n$ matrix for every $\alpha, \beta \in \mathbb{Z}_+^n$ we say $\alpha >_M \beta$ if

$$M\alpha >_{lex} M\beta$$

- Let $>_1$ a monomial order on $\mathbb{Z}_+^n$ and $>_2$ a monomial order on $\mathbb{Z}_+^m$. The **product order** (or block order) $> := (>_1, >_2)$ on $\mathbb{Z}_+^{n+m}$ is defined as: $(\alpha_1, \beta_1) > (\alpha_2, \beta_2)$ if

$$\alpha_1 >_1 \alpha_2 \quad \text{or} \quad \alpha_1 = \alpha_2 \quad \text{and} \quad \beta_1 >_2 \beta_2$$

Using SAGE and monomial orders

```
sage: m = matrix(2,[2,3,0,1]); m
[2 3]
[0 1]
sage: T = TermOrder(m); T
Matrix term order with matrix
[2 3]
[0 1]
sage: P.<a,b> = PolynomialRing(QQ,2,order=T)
sage: P
Multivariate Polynomial Ring in a, b over Rational Field
sage: a > b
False
sage: a^3 < b^2
True
sage: S = TermOrder('M(2,3,0,1)')
sage: T == S
```

Using SAGE and monomial orders

```
sage: P.<a,b,c,d,e,f> =
PolynomialRing(QQ, 6,order='degrevlex(4),neglex(2)')
sage: a > c^4
False
sage: a > e^4
True
sage: e > f^2
False
```

Using SAGE and monomial orders

```
sage: T1 = TermOrder('degrevlex',4)
sage: T2 = TermOrder('neglex',2)
sage: T = T1 + T2
sage: P.<a,b,c,d,e,f> = PolynomialRing(QQ, 6, order=T)
sage: a > c^4
False
sage: a > e^4
True
```

Leading terms

Definition

Fix a monomial order $>$ and let $f \in k[\mathbf{x}]$ be nonzero. Write

$$f = c_\alpha \mathbf{x}^\alpha + \text{ terms with exponent vectors } \beta \neq \alpha$$

such that $c \neq 0$ and $\mathbf{x}^\alpha > \mathbf{x}^\beta$ wherever $\mathbf{x}^\beta$ appears in a nonzero term of f, then:

- $LT(f) = c\mathbf{x}^\alpha = in(f)$ is the leading term or initial term of f
- $LM(f) = \mathbf{x}^\alpha$ is the leading monomial of f
- $LC(f) = c$ is the leading coefficient of f

Using SAGE and monomial orders

```
sage: t = TermOrder('negwdeglex',(1,2,3))
sage: P.<x,y,z> = PolynomialRing(QQbar, 3, order=t)
sage: f=x^2+2*x*y^2; f.lt()
2*x*y^2
sage: f=x^2+2*x*y^2; f.lm()
x*y^2
sage: f=x^2+2*x*y^2; f.lc()
2
```

Ideals

Definition

- An ideal is a subset $I \subset k[\mathbf{x}]$ satisfying:
  - $0 \in I$
  - If $f, g \in I$ then $f + g \in I$
  - If $f \in I$ and $h \in k[\mathbf{x}]$, then $hf \in I$
- Given an ideal I, we define an affine variety $V(I)$

$$V(I) = \{\mathbf{z} \in \mathbb{A}^n \mid f(\mathbf{z}) = 0 \text{ for all } f \in I\}$$

Ideals

The Hilbert Basis Theorem

Every ideal $I \subset k[\mathbf{x}]$ is finitely generated, i.e., there exists $g_1, \ldots, g_t \in I$ such that

$$I = \left\{ \sum_{i=1}^{t} h_i g_i \mid h_1, \ldots, h_t \in k[\mathbf{x}] \right\}$$

We note $I = \langle g_1, \ldots, g_t \rangle$

If $I = \langle g_1, \ldots, g_t \rangle$, then

$$V(I) = \{\mathbf{z} \in \mathbb{A}^n \mid g_i(\mathbf{z}) = 0 \quad i = 0, \ldots, t\}$$

Nullstellensatz

The Weak Nullstellensatz

Fix an ideal $I \subset k[\mathbf{x}]$ where k is algebraically closed

$$V(I) = \emptyset \Leftrightarrow I = k[\mathbf{x}]$$

Hilbert's Nullstellensatz

The polynomials $f, f_1, \ldots f_s \in k[\mathbf{x}]$ satisfy the relation
$f \in I(V(\langle f_1, \ldots, f_s \rangle)) \Leftrightarrow f^m \in \langle f_1, \ldots, f_s \rangle$ for some $m \geq 1$

Division Algorithm

Division Algorithm

Given nonzero polynomials $f, f_1, \dots, f_s \in k[\mathbf{x}]$ and a monomial order $>$, there exist $r, q_1, \dots, q_s \in k[\mathbf{x}]$ with the following properties:

- $f = q_1 f_1 + \cdots + q_s f_s + r$
- No term of r is divisible by any of $LT(f_1), \dots, LT(f_s)$
- $LT(f) = \max\{LT(q_i)LT(f_i), q_i \neq 0\}$

Definition

Any representation

$$f = q_1 f_1 + \cdots + q_s f_s$$

satisfying the third bullet is a standard representation of f

Division Algorithm

Definition

Let $f, g \in k[\mathbf{x}]$ with $LM(f) = \mathbf{x}^\alpha$, $LM(g) = \mathbf{x}^\beta$. Set
$\gamma = lcm(\alpha, \beta) = (\max\{\alpha_1, \beta_1\}, \ldots, \max\{\alpha_n, \beta_n\})$
We define the S polynomial of f and g as

$$S(f, g) = \mathbf{x}^{\gamma-\alpha} f - \frac{LC(f)}{LC(g)} \mathbf{x}^{\gamma-\beta} g$$

Definition

Let $f \in k[\mathbf{x}]$, $G \subset k[\mathbf{x}]$ f is reduced wrt G if no monomial of f is contained in $\langle LM(g) \,|\, g \in G \rangle$

Division Algorithm

Computing Normal Form;
Data: $f \in k[\mathbf{x}]$, $G \subset k[\mathbf{x}]$
Result: $NF(f, G)$
$h = f$;
while $h \neq 0$ *and* $G_h = \{g \in G \mid LM(g) \text{ divides } LM(h)\} \neq \emptyset$ **do**
  choose $g \in G_h$;
  $h = S(h, g)$;
end
return h;

Initial Ideal

Definition

Given an ideal $I \subset k[\mathbf{x}]$ and a monomial order $>$, the initial ideal is the monomial ideal

$$in(I) = \langle LT(f) \mid f \in I \rangle$$

If $I = \langle f_1, \ldots, f_s \rangle$ then

$$\langle LT(f_1), \ldots, LT(f_s) \rangle \subset in(I)$$

though equality need not occur.

Gröbner bases

Definition

Given an ideal $I \subset k[\mathbf{x}]$ a finite set $G \subset I \backslash \{0\}$ is a Gröbner basis for I under $>$ if

$$\langle LT(g) \,|\, g \in G \rangle = in(I)$$

Definition

A Gröbner basis G is reduced if for every $g \in G$

- $LT(g)$ divides no term of any element of $G \backslash \{g\}$
- $LC(g) = 1$

Theorem

Every ideal has a unique reduced Gröbner basis under $>$

Gröbner bases

Property

Given an ideal $I \subset k[\mathbf{x}]$ and $G \subset I$ a Gröbner basis.

- $f \in I \Leftrightarrow NF(f, G) = 0$
- If $NF(-, G)$ is reduced then it is unique

Buchberger's Criterion

Given an ideal $I \subset k[\mathbf{x}]$ and $G \subset I$. The following are equivalent:

- G is a Gröbner basis of I
- $NF(f, G) = 0$ for all $f \in I$
- $I = \langle G \rangle$ and $NF(S(g, g'), G) = 0$ for all $g, g' \in G$

Using SAGE and ideals

```
sage: R= PolynomialRing(QQ,'x',5,order='lex')
sage: I=R.ideal([x0-3*x1+5*x2-7*x3-5,
x1+2*x3-x4+1,x0-2*x1+4*x3-5*x4,x2+x3+x4])
sage: B=I.groebner_basis()
sage: B
[x0 + 3, x1 + 15/14*x4 + 17/14,
x2 - 5/14*x4 - 15/14, x3 - 5/7*x4 - 1/7]
```

Using SAGE and ideals

```
sage: x,y,z = QQ['x,y,z'].gens()
sage: I = ideal(x^5 + y^4 + z^3 - 1,
x^3 + y^3 + z^2 - 1)
sage: B = I.groebner_basis()
sage: B
[y^6+x*y^4+2*y^3*z^2+x*z^3+z^4-2*y^3-2*z^2-x+1,
x^2*y^3-y^4+x^2*z^2-z^3-x^2+1,
x^3+y^3+z^2-1]
sage: f,g,h = B
sage: (2*x*f + g).reduce(B)
0
```

Using SAGE and ideals

```
sage: (2*x*f + g) in I
True

sage: (2*x*f + 2*z*h + y^3).reduce(B)
y^3

sage: (2*x*f + 2*z*h + y^3) in I
False
```

Nullstellensatz and Gröbner bases

Theorem

Given an ideal $I \subset k[\mathbf{x}]$ where k is algebraically closed, the following are equivalent:

- $I \neq k[\mathbf{x}]$
- $1 \notin I$
- $V(I) \neq \emptyset$
- I has a Gröbner basis consisting of nonconstant polynomials
- I has a reduced Gröbner basis $\neq \{1\}$

Nullstellensatz and Gröbner bases, 0 dimensional case

Theorem

Given an ideal $I \subset k[\mathbf{x}]$ where k is algebraically closed, the following are equivalent:

- $V(I) \subset \mathbb{A}^n$ is finite
- $k[\mathbf{x}]/I$ is a finite-dimensional vector space
- Only finitely many monomials are not in $in(I)$

Using SAGE and ideals

```
sage: x,y,z = QQ['x,y,z'].gens()
sage: I=ideal(x^2*z-y,x^2+x*y-y*z,x*z^2+x*z-x)
sage: B=I.groebner_basis()
sage: B
[x^2 - y*z - y, x*y + y, x*z^2 + x*z - x,
y^2 - y*z, y*z^2 + y*z - y]
sage: I.dimension()
1
```

Elimination

Definition

Given an ideal $I \subset k[\mathbf{x}]$ the l-th elimination ideal I_l is

$$I_l = I \cap k[x_{l+1}, \ldots, x_n]$$

The Elimination Theorem

Given an ideal $I \subset k[\mathbf{x}]$ and let G be the Gröbner basis with respect to the lexicographic order, where $x_1 > x_2 > \ldots > x_n$. Then for every $0 \leq l \leq n-1$ the set

$$G_l = G \cap k[x_{l+1}, \ldots, x_n]$$

is a Gröbner basis of the l-th elimination ideal I_l

Elimination

The Extension Theorem

Let $I = \langle q_1, \dots, q_s \rangle \subset k[\mathbf{x}]$ and let I_1 be the first elimination ideal of I. For each $1 \leq i \leq s$ we can write q_i in the form

$$q_i = h_i(x_2, \dots, x_n) x_1^{N_i} + \text{terms with } x_1 \text{ smaller degree than } N_i$$

where $N_i \geq 0$ and $h_i \neq 0$.
If

$$(a_2, \dots, a_n) \in V(I_1)$$

and

$$(a_2, \dots, a_n) \notin V(h_1, \dots, h_s),$$

then there exists $a_1 \in k$ such that $(a_1, a_2, \dots, a_n) \in V(I)$

Using SAGE and ideals

```
sage: x,y,z = QQ['x,y,z'].gens()
sage: I=ideal(x^2*z-1,x^2+x*y-y*z,x*z^2+x*z-x)
sage: B=I.groebner_basis()
sage: B
[x + (-2)*y*z + 2*y + z, y^2+y*z+y-z-3/2, z^2+z-1]
sage: I.dimension()
0
```

Stable sets

Stable sets

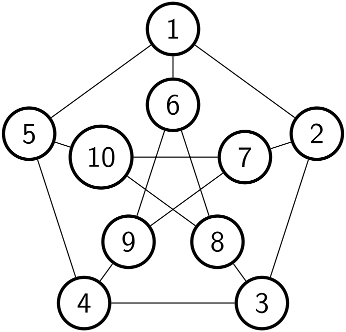
Let $G = (V, E)$ be a graph. For a given positive integer k, consider the following polynomial system:

$$x_i^2 - x_i = 0, \; \forall i \in V$$

$$x_i x_j = 0, \; \forall (i, j) \in E,$$

$$\sum_{i \in V} x_i = k$$

This system is feasible if and only if G has a stable set of size k.



Is there a stable set of size 5 for the Petersen graph?

Using SAGE and ideals

```
sage: R.<x1,x2,x3,x4,x5,x6,x7,x8,x9,x10> =
PolynomialRing(QQ,order='lex')
sage: I=R.ideal([x1^2-x1,x2^2-x2,x3^2-x3,
x4^2-x4,x5^2-x5,x6^2-x6,x7^2-x7,x8^2-x8,
x9^2-x9,x10^2-x10,x1*x2,x1*x5,x1*x6,x2*x3,
x2*x7,x3*x4,x3*x8,x4*x9,x4*x5,x5*x10,
x6*x8,x6*x9,x7*x9,x7*x10,x8*x10,
x1+x2+x3+x4+x5+x6+x7+x8+x9+x10-5])
sage: B=I.groebner_basis()
sage: B
[1]
```

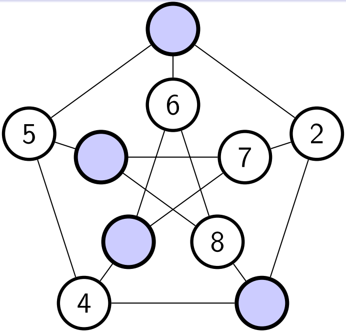
```
sage: I=R.ideal([x1^2-x1,x2^2-x2,x3^2-x3,
x4^2-x4,x5^2-x5,x6^2-x6,x7^2-x7,x8^2-x8,
x9^2-x9,x10^2-x10,x1*x2,x1*x5,x1*x6,x2*x3,
x2*x7,x3*x4,x3*x8,x4*x9,x4*x5,x5*x10,
x6*x8,x6*x9,x7*x9,x7*x10,x8*x10,
x1+x2+x3+x4+x5+x6+x7+x8+x9+x10-4])
sage: B=I.groebner_basis()
sage: B
[x1 -x8-2*x9*x10+x9, x2+2*x9*x10-x9-x10,
x3+x8-x9*x10+x10-1, x4-x8+x9-x10,
x5+x8+x9*x10-x9+x10-1,
x6+x8+x9*x10-1, x7-x9*x10+x9+x10-1,
x8^2-x8, x8*x9+x9*x10-x9,
x8*x10, x9^2-x9, x10^2-x10]
sage: I.dimension()
0
```

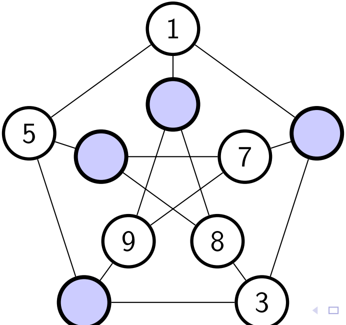
```
sage: I.normal_basis()
[x9*x10, x10, x9, x8, 1]
```

There are 5 solutions. We can construct them from the Gröbner basis. Looking at the normal basis, we can start fixing $x_{10} = 1$ then:

```
sage: I=R.ideal([x1^2-x1,x2^2-x2,x3^2-x3,
x4^2-x4,x5^2-x5,x6^2-x6,x7^2-x7,x8^2-x8,
x9^2-x9,x10^2-x10,x1*x2,x1*x5,x1*x6,x2*x3,
x2*x7,x3*x4,x3*x8,x4*x9,x4*x5,x5*x10,
x6*x8,x6*x9,x7*x9,x7*x10,x8*x10,
x1+x2+x3+x4+x5+x6+x7+x8+x9+x10-4,x10-1])
sage: B=I.groebner_basis()
sage: B
[x1 -x9, x2+x9-1, x3-x9, x4+x9-1, x5,
x6+x9-1, x7, x8, x9^2-x9, x10-1]
sage: I.normal_basis()
[x9, 1]
```

If $x_9 = 1$, we have the solution $\{1, 3, 9, 10\}$, if $x_9 = 0$ the solution is $\{2, 4, 6, 10\}$

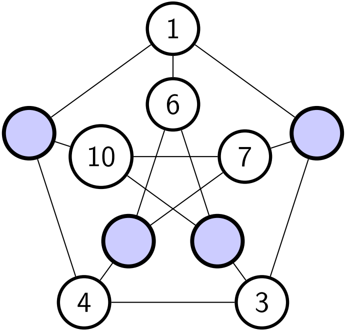

6
5
2
7
8
4


1
5
7
9
8
3

If we choose $x_{10} = 0$

```
sage: I=R.ideal([x1^2-x1,x2^2-x2,x3^2-x3,
x4^2-x4,x5^2-x5,x6^2-x6,x7^2-x7,x8^2-x8,
x9^2-x9,x10^2-x10,x1*x2,x1*x5,x1*x6,x2*x3,
x2*x7,x3*x4,x3*x8,x4*x9,x4*x5,x5*x10,
x6*x8,x6*x9,x7*x9,x7*x10,x8*x10,
x1+x2+x3+x4+x5+x6+x7+x8+x9+x10-4,x10])
sage: B=I.groebner_basis()
sage: B
[x1 -x8+x9, x2-x9, x3+x8-1, x4-x8+x9,
x5+x8-x9-1, x6+x8-1, x7+x9-1, x8^2-x8,
x8*x9-x9, x9^2-x9, x10]
sage: I.normal_basis()
[x9, x8, 1]
```

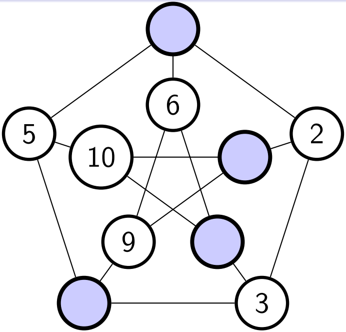
If $x_9 = 1$, we have the solution $\{2, 5, 8, 9\}$

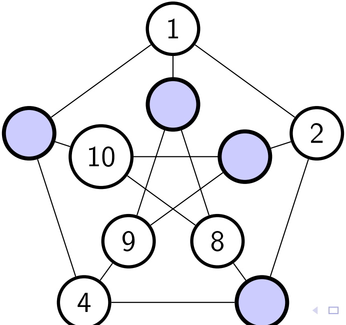

1
6
10
7
4
3

If we choose $x_{10} = 0$ and $x_9 = 0$

```
sage: I=R.ideal([x1^2-x1,x2^2-x2,x3^2-x3,
x4^2-x4,x5^2-x5,x6^2-x6,x7^2-x7,x8^2-x8,
x9^2-x9,x10^2-x10,x1*x2,x1*x5,x1*x6,x2*x3,
x2*x7,x3*x4,x3*x8,x4*x9,x4*x5,x5*x10,
x6*x8,x6*x9,x7*x9,x7*x10,x8*x10,
x1+x2+x3+x4+x5+x6+x7+x8+x9+x10-4,x10,x9])
sage: B=I.groebner_basis()
sage: B
[x1-x8, x2, x3+x8-1, x4-x8, x5+x8-1,
x6+x8-1, x7-1,x8^2-x8, x9, x10]
sage: I.normal_basis()
[x8, 1]
```

If $x_8 = 1$, we have the solution $\{1, 4, 7, 8\}$, if $x_8 = 0$ the solution is $\{3, 5, 6, 7\}$


6
5
10
2
9
3


1
10
2
9
8
4

k-Colorable graphs

k-Colorable graphs

Let $G = (V, E)$ be a graph. For a positive integer k, consider the following polynomial system of $|V| + |E|$ equations:

$$x_i^k - 1 = 0, \ \forall i \in V$$

$$\sum_{s=0}^{k-1} x_i^{k-1-s} x_j^s = 0, \ \forall (i, j) \in E,$$

The graph G is k-colorable if and only if this system has a complex solution. Furthermore, when k is odd, G is k-colorable if and only if this system has a common root over $\overline{\mathbb{F}_2}$, the algebraic closure of the finite field with two elements.

k-Colorable graphs

We are using the Nullstellensatz over $\mathbb{C}$ an algebraically closed ring. The equation $x_i^k - 1 = 0$, assign a k-th root of unity to each vertex (a color).
If we take an edge (i, j), as these vertices have a color,

$$0 = 1 - 1 = x_i^k - x_j^k = (x_i - x_j)(x_i^{k-1} + x_i^{k-2}x_j + \cdots + x_j^{k-1})$$

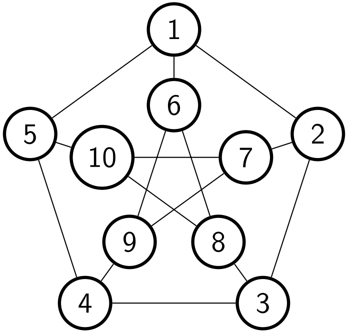
As those vertices are joined by an edge they have different colors, then the second factor must be zero

k-Colorable graphs

Conversely, if there is solution of the above polynomials, we have a color for each vertex. We need to prove that any adjacent vertex has different color. If (i, j) is an edge and both vertices have the same root of unity β, then

$$x_i^{k-1} + x_i^{k-2} x_j + \cdots + x_j^{k-1} = \beta^{k-1} + \beta^{k-1} + \cdots + \beta^{k-1} = k\beta^{k-1} = 0$$

Over $\mathbb{C}$ clearly $\beta = 0$.



Is the Petersen graph 3-colorable?

Using SAGE and ideals

```
sage: R.<x1,x2,x3,x4,x5,x6,x7,x8,x9,x10> =
PolynomialRing(QQ,order='lex')
sage: I=R.ideal([x1^3-1,x2^3-1,x3^3-1,
x4^3-1,x5^3-1,x6^3-1,x7^3-1,x8^3-1,
x9^3-1,x10^3-1,x1^2+x1*x2+x2^2,
x1^2+x1*x5+x5^2,
...
x8^2+x8*x10+x10^2])
sage: B=I.groebner_basis()
sage: B
Polynomial Sequence with 33 Polynomials in 10 Variables
sage: I.dimension()
0
sage: I.normal_basis()
Polynomial Sequence with 120 Polynomials in 10 Variables
```

Applications to binary optimization

Let $\mathbf{A} \in \mathbb{Z}^{m \times n}$ and $\mathbf{b} \in \mathbb{Z}^m$ and consider the system

$$\begin{array}{rcl} \mathbf{Ax} & = & \mathbf{b} \\ \mathbf{x} & \in & \{0,1\}^n \end{array}$$

We can use the polynomial $x_i^2 - x_i = 0$ to assure $x_i \in \{0,1\}$
Let $f_1 = \mathbf{a}_i\mathbf{x} - b_i$ and $g_i = x_i^2 - x_i$
Then

$$I = \langle f_1, \ldots, f_m, g_1, \ldots, g_n \rangle$$

Feasible and Gröbner basis;
Data: $\mathbf{A} \in \mathbb{Z}^{m \times n}$, $\mathbf{b} \in \mathbb{Z}^m$
Result: A feasible solution $(a_1, \ldots, a_n)$ or a infeasibility certificate
Compute G Gröbner basis of the ideal J for lex order $x_1 > \ldots > x_n$;
if $G \neq \{1\}$ **then**
    for $1 \leq l \leq n$ consider $G_l = G \cap k[x_{l+1}, \ldots, x_n]$;
    Starting from index $n - 1$;
    Find $a_n \in V(G_{n-1})$;
    Extend a_n to (a_{n-1}, a_n) such that $(a_{n-1}, a_n) \in V(G_{n-2})$;
    $\ldots$;
    return $(a_1, \ldots, a_n)$;
else
    There is no feasible solution
end

Using SAGE and binary optimization

Consider the system

$$x_1 + 2x_2 + 3x_3 + 4x_4 + 5x_5 + 15x_6 = 15 \quad \mathbf{x} \in \{0,1\}^6$$

```
sage: R.<x1,x2,x3,x4,x5,x6> = PolynomialRing(QQ,order='lex
sage: I=R.ideal([x1^2-x1,x2^2-x2,x3^2-x3,x4^2-x4,
x5^2-x5,x6^2-x6,x1+2*x2+3*x3+4*x4+5*x5+15*x6-15])
sage: B=I.groebner_basis()
sage: B
[x1+x6-1, x2+x6-1, x3+x6-1, x4+x6-1, x5 +x6-1,
x6^2 -x6]
```

Using SAGE and binary optimization

Consider the system

$$x_1 + 2x_2 + 3x_3 + 4x_4 + 6x_5 = 6 \quad \mathbf{x} \in \{0,1\}^5$$

```
sage: R.<x1,x2,x3,x4,x5> = PolynomialRing(QQ,order='lex')
sage: I=R.ideal([x1^2-x1,x2^2-x2,x3^2-x3,
x4^2-x4,x5^2-x5,x1+2*x2+3*x3+4*x4+6*x5-6])
sage: B=I.groebner_basis()
sage: B
[x1 + x4 + x5 - 1, x2 + x5 - 1, x3 + x4 + x5 - 1,
x4^2 - x4, x4*x5, x5^2- x5]
```

Applications to binary optimization

We next use Gröbner bases to solve the optimization problem

$$\begin{array}{rrcl} \min & \mathbf{c}'\mathbf{x} & & \\ \text{subject to} & \mathbf{A}\mathbf{x} & = & \mathbf{b} \\ & \mathbf{x} & \in & \{0,1\}^n \end{array}$$

We can use the polynomial $h = y - \sum_{j=1}^{n} c_j x_j$
We let $f_1 = \mathbf{a}_i\mathbf{x} - b_i$ and $g_i = x_i^2 - x_i$
Then

$$I = \langle f_1, \ldots, f_m, g_1, \ldots, g_n, h \rangle$$

and a term order such that $x_1 > \ldots > x_n > y$

Consider the problem

$$\begin{array}{rll} \min & x_1 + 2x_2 + 3x_3 + 3x_4 & \\ \text{subject to} & x_1 + x_2 + 2x_3 + x_4 & = 3 \\ & \mathbf{x} \in \{0,1\}^4 & \end{array}$$

```
sage: R.<x1,x2,x3,x4,y> = PolynomialRing(QQ,order='lex')
sage: I=R.ideal([x1^2-x1,x2^2-x2,x3^2-x3,x4^2-x4,
x1+x2+2*x3+x4-3,y-x1-2*x2-3*x3-3*x4])
sage: B=I.groebner_basis()
sage: B
[x1 + x3 - 1/2*y^2 + 11/2*y - 16,
 x2 + x3 + y^2 - 10*y + 23,
 x3^2 - x3,
 x3*y - 6*x3 - y + 6,
 x4 - 1/2*y^2 + 9/2*y - 10,
 y^3 - 15*y^2 + 74*y - 120]
```

Applications to optimization

$$\begin{array}{rrcl} \min & \mathbf{c}'\mathbf{x} & & \\ \text{subject to} & \mathbf{A}\mathbf{x} & = & \mathbf{b} \\ & \mathbf{x} & \in & \mathbb{Z}^n \end{array}$$

with $\mathbf{A} \in \mathbb{Z}_+^{m \times n}$, $\mathbf{b} \in \mathbb{Z}_+^m$ $\mathbf{c} \in \mathbb{Z}_+^n$. We introduce a new variable z_i for the i-constraint, so

$$z_i^{a_{i1}x_1 + \ldots + a_{in}x_n} = z_i^{b_i}$$

$$\prod_{i=1}^{m}\prod_{j=1}^{n}\left(z_i^{a_{ij}}\right)^{x_j} = \prod_{j=1}^{n}\prod_{i=1}^{m}\left(z_i^{a_{ij}}\right)^{x_j} = \prod_{j=1}^{n}\left(\prod_{i=1}^{m} z_i^{a_{ij}}\right)^{x_j} = \prod_{i=1}^{m} z_i^{b_i}$$

Applications to optimization

We define the mapping $\phi : k[w_1, \ldots, w_n] \to k[z_1, \ldots, z_m]$ such that

$$\phi(w_j) = \prod_{i=1}^{m} z_i^{a_{ij}}$$

then, for $g \in k[\mathbf{w}]$

$$\phi(g(w_1, \ldots, w_n)) = g(\phi(w_1), \ldots, \phi(w_n))$$

Proposition

A vector $\mathbf{x} \in \mathbb{Z}_+^n$ is feasible if and only if ϕ maps the monomial $w_1^{x_1} \cdots w_n^{x_n}$ to the monomial $\mathbf{z}^{\mathbf{b}}$

Applications to optimization

If we consider the problem

$$\begin{array}{lllll} 4x_1+ & 5x_2 & +x_3 & & = 37 \\ 2x_1+ & 3x_2 & & +x_4 & = 20 \end{array}$$

The mapping is given by

$$\phi(w_1) = z_1^4 z_2^2 \quad \phi(w_2) = z_1^5 z_2^3 \quad \phi(w_3) = z_1 \quad \phi(w_4) = z_2$$

The set of feasible solutions are all the integers points (x_1, x_2, x_3, x_4) such that

$$\phi(w_1^{x_1} w_2^{x_2} w_3^{x_3} w_4^{x_4}) = z_1^{37} z_2^{20}$$

Applications to optimization

Let $f_j = \phi(w_j) = \prod_{i=1}^{m} z_i^{a_{ij}}$, we can consider the ideal

$$I = \langle f_1 - w_1, \ldots, f_n - w_n \rangle \subset k[\mathbf{z}, \mathbf{w}]$$

and a term order, being an elimination order of $\mathbf{z}$

```
sage: R.<z1,z2,w1,w2,w3,w4> =
PolynomialRing(QQ,order='lex')
sage: I=R.ideal([z1^4*z2^2-w1,z1^5*z2^3-w2,z1-w3,z2-w4])
sage: B=I.groebner_basis()
sage: B
[z1 - w3, z2 - w4, w1 - w3^4*w4^2, w2 - w3^5*w4^3]
sage: (z1^37*z2^20).reduce(B)
w3^37*w4^20
```

```
sage: T1 = TermOrder('lex',2)
T2 = TermOrder('wdeglex',(1,2,3,4))
sage: R.<z1,z2,w1,w2,w3,w4> =
PolynomialRing(QQ,order=T1+T2)
sage: I=R.ideal([z1^4*z2^2-w1,z1^5*z2^3-w2,z1-w3,z2-w4])
sage: B=I.groebner_basis()
sage: B
[z1-w3, z2-w4, w3^4*w4^2-w1,
w2*w3^3*w4-w1^2, w1^5*w4^2-w2^4,
w2^2*w3^2 - w1^3, w2^3*w3 - w1^4*w4, w1*w3*w4 - w2]
sage: (z1^37*z2^20).reduce(B)
w1^8*w2*w4
```

Solving LIPP;

Data: $\mathbf{A} \in \mathbb{Z}_+^{m\times n}$, $\mathbf{b} \in \mathbb{Z}_+^m$,$\mathbf{c} \in \mathbb{Z}_+^n$

Result: The solution $(x_1^*, \ldots, x_n^*)$ or a infeasibility certificate

Compute G Gröbner basis of the ideal I for term order such that $z_1 > \ldots > z_m > w_1 > \ldots w_n$ and $\mathbf{c}'\mathbf{x}_1 > \mathbf{c}'\mathbf{x}_2$ then $\mathbf{w}^{\mathbf{x}_1} > \mathbf{w}^{\mathbf{x}_2}$;

if $g = NF(\prod_{i=1}^m z_i^{b_i}, G) \in k[\mathbf{w}]$ **then**

  $g = w_1^{x_1^*} \cdots w_n^{x_n^*}$;

  return $(x_1^*, \ldots, x_n^*)$;

else

  There is no feasible solution

end

Applications to optimization

If we consider the problem

$$\begin{array}{rrrl} 2x_1 + & -x_2 & +x_3 & = 4 \\ -x_1 + & 2x_2 & & = 5 \end{array}$$

The mapping can be extended by

$$\phi(w_1) = \frac{z_1^2}{z_2} \quad \phi(w_2) = \frac{z_2^2}{z_1} \quad \phi(w_3) = z_1$$

So

$$J = \langle w_1 z_2 - z_1^2, w_2 z_1 - z_2^2, w_3 - z_1 \rangle$$

```
sage: R.<z1,z2,w1,w2,w3> =
PolynomialRing(QQ,order='lex')
sage: I=R.ideal([z1^2-w1*z2,z2^2-w2*z1,z1-w3])
sage: B=I.groebner_basis()
sage: B
[z1-w3, z2^2-w2*w3, z2*w1-w3^2,
z2*w3^2-w1*w2*w3, w1^2*w2*w3 -w3^4]
sage: (w1^2*w2-w3^3).reduce(B)
w1^2*w2-w3^3
```

Applications to optimization

We consider now the general case

$$\begin{array}{rrcl} \min & \mathbf{c}'\mathbf{x} & & \\ \text{subject to} & \mathbf{A}\mathbf{x} & = & \mathbf{b} \\ & \mathbf{x} & \in & \mathbb{Z}^n \end{array}$$

with $\mathbf{A} \in \mathbb{Z}^{m\times n}$, $\mathbf{b} \in \mathbb{Z}^m$ $\mathbf{c} \in \mathbb{Z}^n_+$. The mapping $\phi : k[w_1, \ldots, w_n] \to k[z_1, \ldots, z_m, z_1^{-1}, \ldots, z_m^{-1}]$ such that

$$\phi(w_j) = \prod_{i=1}^{m} z_i^{a_{ij}}$$

We can always write any column $\mathbf{a}_j = \mathbf{a}_j^+ - \mathbf{a}_j^-$ with $\mathbf{a}_j^+, \mathbf{a}_j^- \geq \mathbf{0}$
We introduce the polynomials:

$$I = \langle \mathbf{z}^{\mathbf{a}_j^-} w_j - \mathbf{z}^{\mathbf{a}_j^+}, 1 - t z_1 \cdots z_m \rangle$$

```
sage: R.<t,z1,z2,w1,w2,w3> =
PolynomialRing(QQ,order='lex')
sage: I=R.ideal([z1^2-w1*z2,z2^2-w2*z1,z1-w3,1-t*z1*z2])
sage: B=I.groebner_basis()
sage: B
[t*w1*w2-1, t*w2*w3^2-z2, t*w3^3-w1, z1 -w3, z2^2-w2*w3,
z2*w1-w3^2, z2*w3 - w1*w2, w1^2*w2 - w3^3]
sage: (w1^2*w2-w3^3).reduce(B)
0
```

```
sage: T1 = TermOrder('lex',3)
T2 = TermOrder('wdeglex',(1,2,3))
sage: R.<t,z1,z2,w1,w2,w3>=PolynomialRing(QQ,order=T1+T2)
sage: I=R.ideal([z1^2-w1*z2,z2^2-w2*z1,z1-w3,1-t*z1*z2])
sage: B=I.groebner_basis()
sage: B
[t*w2*w3^2-z2, t*w1*w2-1, z1-w3, z2^2-w2*w3,
z2*w3-w1*w2, z2*w1-w3^2, w3^3-w1^2*w2]
sage: (z1^4*z2^5).reduce(B)
w1^3*w2^4*w3^2
```

Improving the algorithm

We are considering

$$I = \langle \mathbf{z}^{\mathbf{a}_j^-} w_j - \mathbf{z}^{\mathbf{a}_j^+}, 1 - tz_1 \cdots z_m \rangle$$

Given G a Gröbner basis with respect to a term order which eliminates t and $\mathbf{z}$ we have that $G \cap k[\mathbf{w}]$ is a Gröbner basis of the ideal:

$$I \cap k[\mathbf{w}] = I_A$$

Given $g(\mathbf{w}) \in I \cap k[\mathbf{w}] \Rightarrow g(\mathbf{w}) \in \ker(\phi) = I_A$
This ideal is called the toric ideal of $\mathbf{A}$ and it does not depend on the right hand side of the constraints

Improving the algorithm

Proposition

The toric ideal I_A is a k-vector space spanned by the binomials:

$$\{\mathbf{x}^{\mathbf{u}} - \mathbf{x}^{\mathbf{v}} : \ \mathbf{A}\mathbf{u} = \mathbf{A}\mathbf{v}, \ \mathbf{u}, \ \mathbf{v} \in \mathbb{Z}_+^n\}$$

and therefore

$$I_A = \langle \mathbf{x}^{\mathbf{u}} - \mathbf{x}^{\mathbf{v}} : \ \mathbf{A}\mathbf{u} = \mathbf{A}\mathbf{v}, \ \mathbf{u}, \ \mathbf{v} \in \mathbb{Z}_+^n \rangle$$

Improving the algorithm

Using Hilbert's basis theorem, there exist a finite number of binomials which generate I_A. We can restrict to binomials $\mathbf{x}^{\mathbf{u}} - \mathbf{x}^{\mathbf{v}}$ with disjoint support, that is, $supp(\mathbf{u}) \cap supp(\mathbf{v}) = \emptyset$. If not

$$gcd(\mathbf{x}^{\mathbf{u}}, \mathbf{x}^{\mathbf{v}}) = \mathbf{x}^{\gamma} \Rightarrow \mathbf{x}^{\mathbf{u}-\gamma} - \mathbf{x}^{\mathbf{v}-\gamma} \in I_A$$

can replace the previous element in the set of generators. Any $\mathbf{w} \in \ker(A) \cap \mathbb{Z}^n$ can be expressed as a binomial with disjoint support

$$\mathbf{x}^{\mathbf{w}^+} - \mathbf{x}^{\mathbf{w}^-}$$

Improving the algorithm

Fixed $>$ a term order and $\mathbf{c} \in \mathbb{R}^n_+$, we define the product order $>_{\mathbf{c}}$ as

$$\alpha >_{\mathbf{c}} \beta \Leftrightarrow \begin{cases} \mathbf{c}'\alpha > \mathbf{c}'\beta & \text{or} \\ \mathbf{c}'\alpha = \mathbf{c}'\beta & \text{and } \alpha > \beta \end{cases}$$

Theorem

Let $>$ be any term order, $\mathbf{A} \in \mathbb{Z}^{m \times n}$ a fixed matrix, and $\mathbf{c} \in \mathbb{R}^n_+$ a fixed cost vector. Moreover, let $G_{>_{\mathbf{c}}}$ be the reduced minimal Gröbner basis of I_A with respect to $>_{\mathbf{c}}$. Then for any right-hand side vector $\mathbf{b}$ and any nonoptimal feasible solution $\mathbf{z}_0$ there is some binomial $\mathbf{x}^{\mathbf{u}} - \mathbf{x}^{\mathbf{v}} \in G_{>_{\mathbf{c}}}$ such that $z_0 - \mathbf{u} + \mathbf{v}$ is a better feasible solution than z_0.

Improving the algorithm

During the Buchberger algorithm, one must check whether the S-polynomial of every critical pair reduces to 0. Checking reduction to 0 is computationally expensive.

The project-and-lift algorithms to compute generating sets and Gröbner bases of lattice ideals are implemented in the software package `4ti2`

`4ti2` can be called from sage

```
sage: from sage.interfaces.four_ti_2 import four_ti_2
sage: four_ti_2.write_matrix([[2,-1,1],[-1,2,0]],
"test_file.mat")
sage: four_ti_2.write_matrix([[1,2,3]], "test_file.cost")
sage: four_ti_2.call("groebner", "test_file", False)
sage: four_ti_2.read_matrix("test_file.gro")
[-2 -1  3]
```

We are interested in solving the minimum number of nickels and quarters, such that using pennies (1ct), nickels (5ct), dimes (10ct) and quarters (25ct), they sum up 99 cents and they are exactly 11 coins, that is:

$$\begin{array}{rrl} \text{min} & x_2 + x_4 & \\ \text{subject to} & x_1 + x_2 + x_3 + x_4 & = 11 \\ & x_1 + 5x_2 + 10x_3 + 25x_4 & = 99 \end{array}$$

```
sage: from sage.interfaces.four_ti_2 import four_ti_2
sage: four_ti_2.write_matrix([[1,1,1,1],[1,5,10,25]],
 "4coins.mat")
sage: four_ti_2.write_matrix([[0,1,0,1]],"4coins.cost")
sage: four_ti_2.call("groebner", "4coins", False)
sage: four_ti_2.write_matrix([[4,4,0,3]],"4coins.feas")
sage: four_ti_2.call("normalform","4coins")
sage: four_ti_2.read_matrix("4coins.nf")
[4 1 4 2]
```

Nonlinear integer programming with linear objective function

In this talk we introduce some refinements of a general setting to treat the following problem **(P)**:

$$
\begin{array}{rl}
\min & \mathbf{c}'\mathbf{x}, \\
\\
\text{subjecto to} & A\mathbf{x} = \mathbf{b} \\
& g_1(\mathbf{x}) \leq C_1 \\
& \vdots \\
& g_m(\mathbf{x}) \leq C_m, \\
\\
& \mathbf{x} = (x_1, \ldots, x_n)' \in \mathbb{Z}^n_{\geq 0}, \quad m \geq 1.
\end{array}
$$

with A an integer matrix, a nonlinear integer programming problem with linear objective function.

Our contributions give an alternative to treat real size problems in this form.

Tayur, Thomas and Natraj '1995

Our initial inspiration

Tayur, Thomas and Natraj, in An algebraic geometry algorithm for scheduling in presence of setups and correlated demands [**Math. Programming '1995**], presented a way of providing an exact solution for a class of stochastic integer programming problem.

Tayur, Thomas and Natraj '1995

Our initial inspiration

Tayur, Thomas and Natraj, in An algebraic geometry algorithm for scheduling in presence of setups and correlated demands [**Math. Programming '1995**], presented a way of providing an exact solution for a class of stochastic integer programming problem.

Their method can be generalized, in principle, to any (**P**) as the one described before. They used an idea of Sturmfels in Convex Polytopes to visit all the feasible points of a linear integer programming problem.

Tayur, Thomas and Natraj '1995

The method is based on:

- The calculation of a test set for a linear subproblem **(LP)** of **(P)**.
- An inverse search process, called walk-back, in order to reach, starting at the optimum of **(LP)**, the optimum of **(P)**.

Walk-back: test set

Test-sets [cf. Schrijver '1998]

Given a integer linear programming

$$\min\{\mathbf{c}'\mathbf{x} \mid A\mathbf{x} = \mathbf{b}, \mathbf{x} \in \mathbb{Z}^n_{\geq 0}\}$$

there exists a finite set $\mathcal{T} = \{\mathbf{t}^1, \ldots, \mathbf{t}^N\}$ (depending only on A and $\mathbf{c}'$) that assures that a feasible solution $\mathbf{x}^\star$ is optimal if and only if

$$\mathbf{c}'(\mathbf{x}^\star + \mathbf{t}^i) \geq \mathbf{c}'\mathbf{x}^\star$$

whenever $(\mathbf{x}^\star + \mathbf{t}^i)$ is feasible, $i = 1, \ldots, N$. Such a $\mathcal{T}$ is called a test-set with respect to $(A, \mathbf{c}')$.

Properties of test sets

- A test set provides a method which solves an IPP, given a feasible point

Properties of test sets

- A test set provides a method which solves an IPP, given a feasible point
- At each step, there is an element of the test set which improves the cost, or there is no improvement, so we have reached the optimum

Properties of test sets

- A test set provides a method which solves an IPP, given a feasible point
- At each step, there is an element of the test set which improves the cost, or there is no improvement, so we have reached the optimum
- The process ends whenever the cost is bounded

Properties of test sets

- A test set provides a method which solves an IPP, given a feasible point
- At each step, there is an element of the test set which improves the cost, or there is no improvement, so we have reached the optimum
- The process ends whenever the cost is bounded

Properties of test sets

- A test set provides a method which solves an IPP, given a feasible point
- At each step, there is an element of the test set which improves the cost, or there is no improvement, so we have reached the optimum
- The process ends whenever the cost is bounded

Algebraic test set

Gröbner basis of toric ideal I_A with respect to weighted orders for c are test sets

Properties of test sets

- A test set provides a method which solves an IPP, given a feasible point
- At each step, there is an element of the test set which improves the cost, or there is no improvement, so we have reached the optimum
- The process ends whenever the cost is bounded

Algebraic test set

Gröbner basis of toric ideal I_A with respect to weighted orders for c are test sets

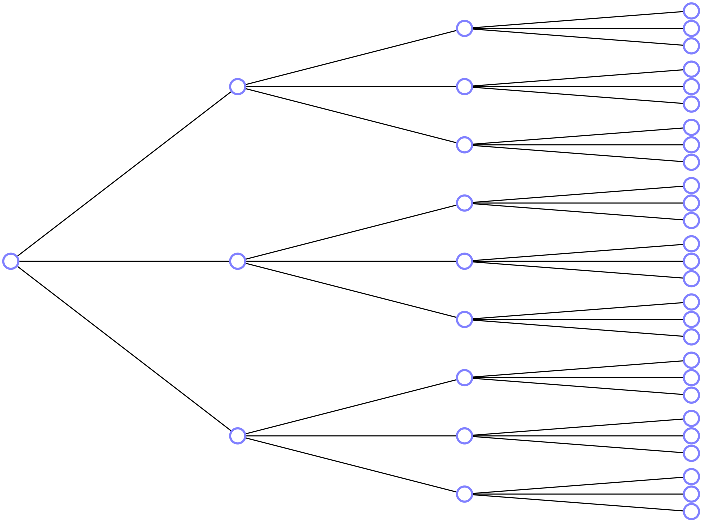
This step may be the bottleneck

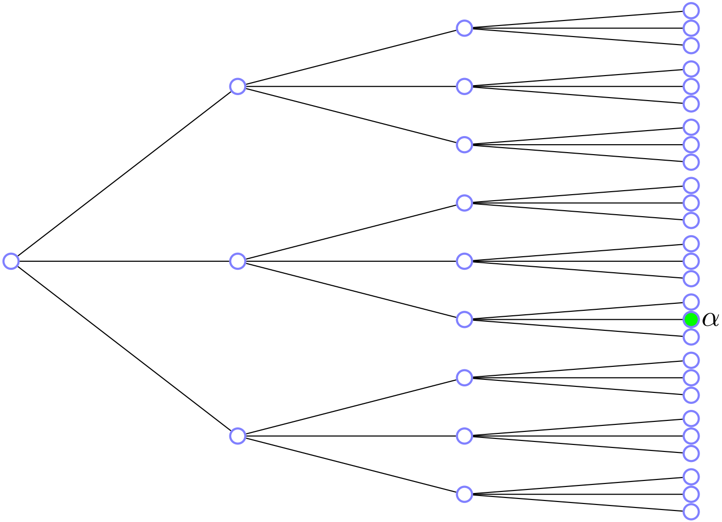
Properties of test sets

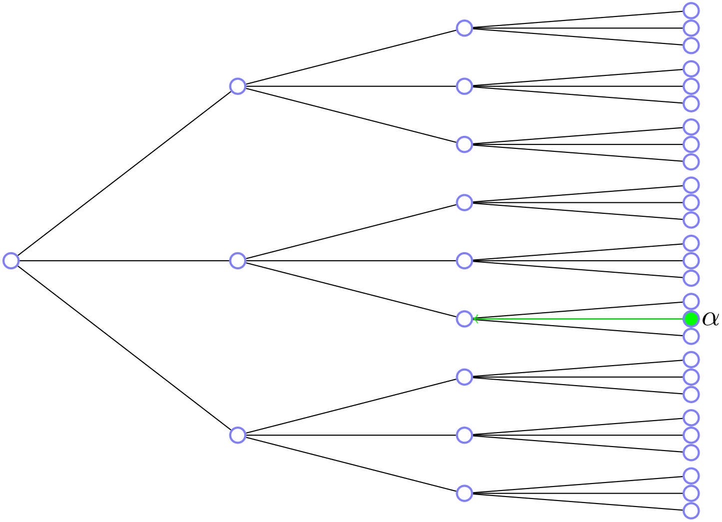
- A test set provides a method which solves an IPP, given a feasible point
- At each step, there is an element of the test set which improves the cost, or there is no improvement, so we have reached the optimum
- The process ends whenever the cost is bounded

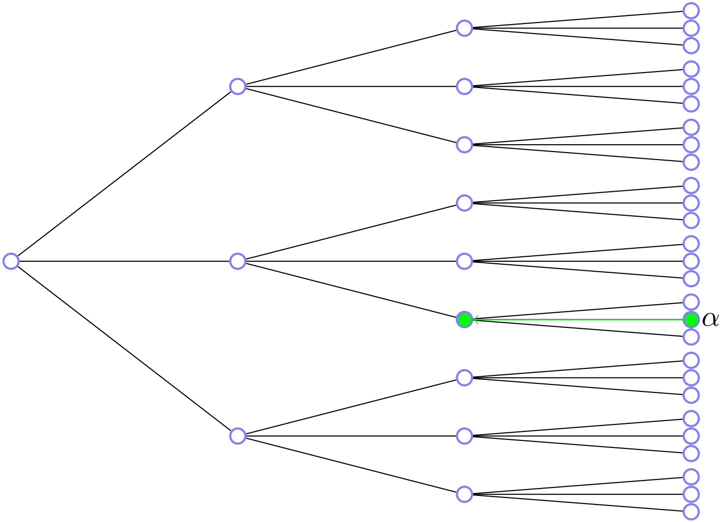
Algebraic test set

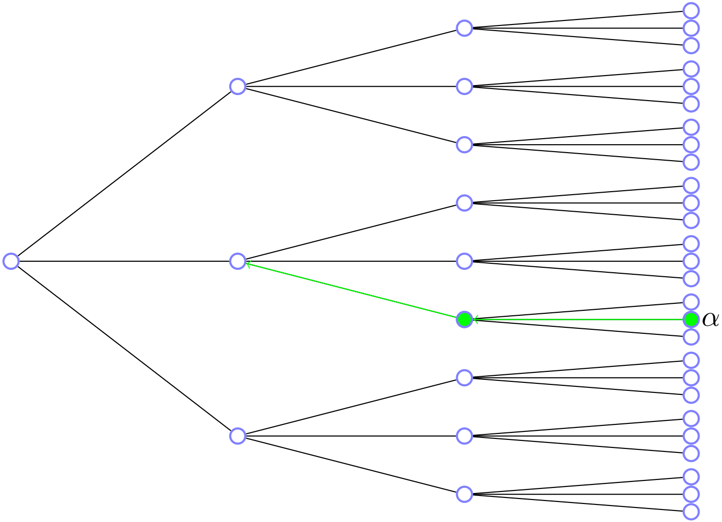
Gröbner basis of toric ideal I_A with respect to weighted orders for c are test sets
This step may be the bottleneck
In some cases a closed formula for the test set can be given.

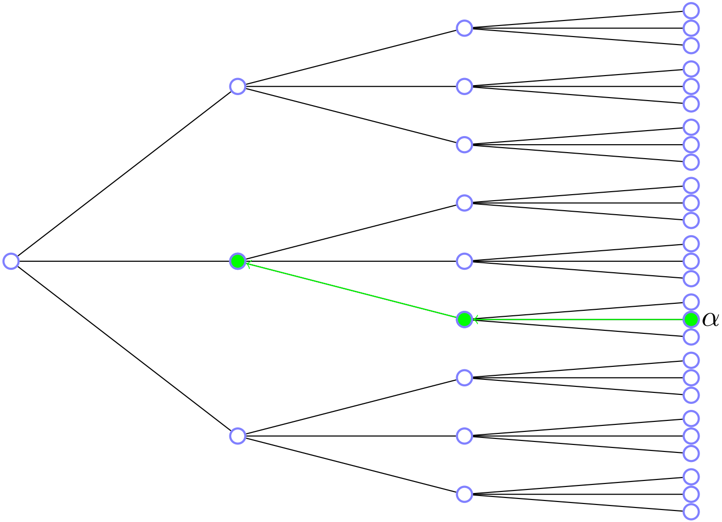


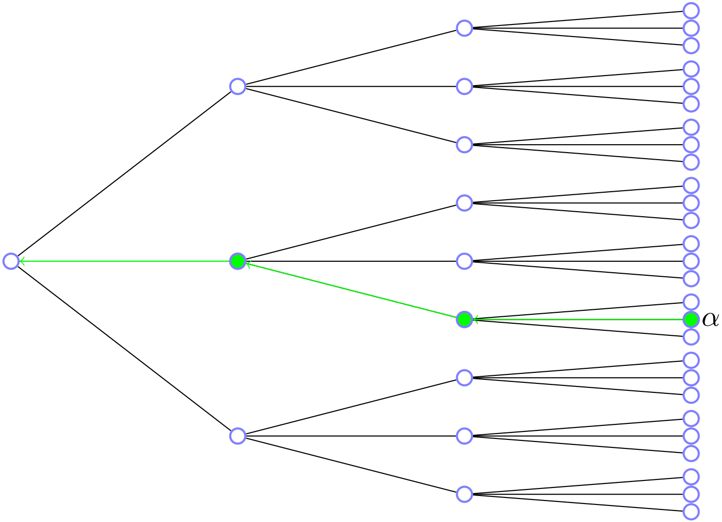

α

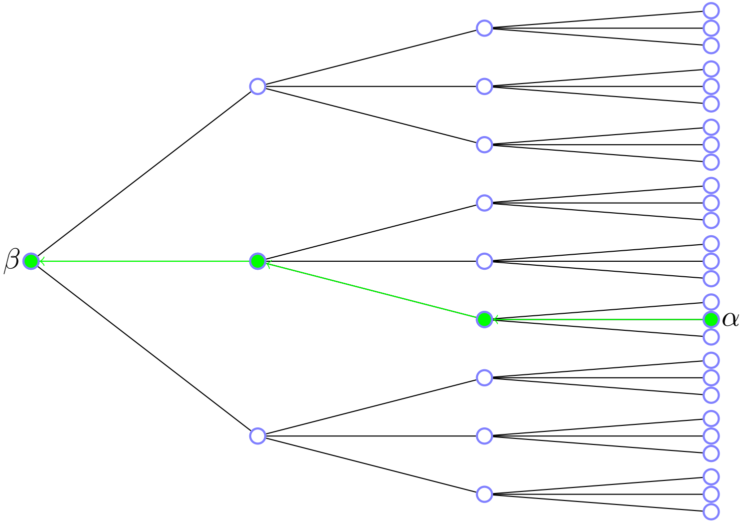
α


α


α


α


α


β
α

Solving **(P)**

- Let c_0 be the cost for y^0 feasible point for **(P)**

Solving **(P)**

- Let c_0 be the cost for y^0 feasible point for **(P)**
- We calculate β optimum for **(LP)**.

Solving **(P)**

- Let c_0 be the cost for y^0 feasible point for **(P)**
- We calculate β optimum for **(LP)**.
- If β is feasible for **(P)**, it is optimum for **(P)**.

Solving (P)

- Let c_0 be the cost for y^0 feasible point for **(P)**
- We calculate β optimum for **(LP)**.
- If β is feasible for **(P)**, it is optimum for **(P)**.
- If β is not feasible for **(P)**, we use the reverse skeleton $\mathcal{G}'_{>_c}$.

Solving **(P)**

- Let c_0 be the cost for y^0 feasible point for **(P)**
- We calculate β optimum for **(LP)**.
- If β is feasible for **(P)**, it is optimum for **(P)**.
- If β is not feasible for **(P)**, we use the reverse skeleton $\mathcal{G}'_{>_c}$.
- For any γ obtained from $\mathcal{G}'_{>_c}$:

Solving **(P)**

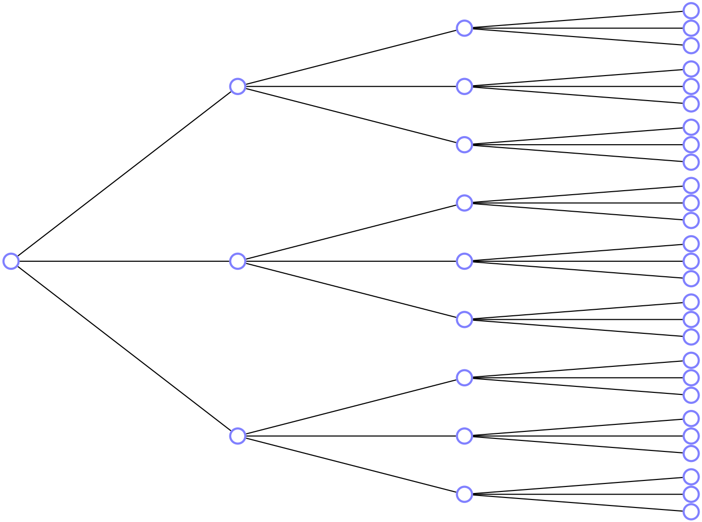
- Let c_0 be the cost for y^0 feasible point for **(P)**
- We calculate β optimum for **(LP)**.
- If β is feasible for **(P)**, it is optimum for **(P)**.
- If β is not feasible for **(P)**, we use the reverse skeleton $\mathcal{G}'_{>_c}$.
- For any γ obtained from $\mathcal{G}'_{>_c}$:
  - If $c(\gamma) > c_0$, we prune the branch

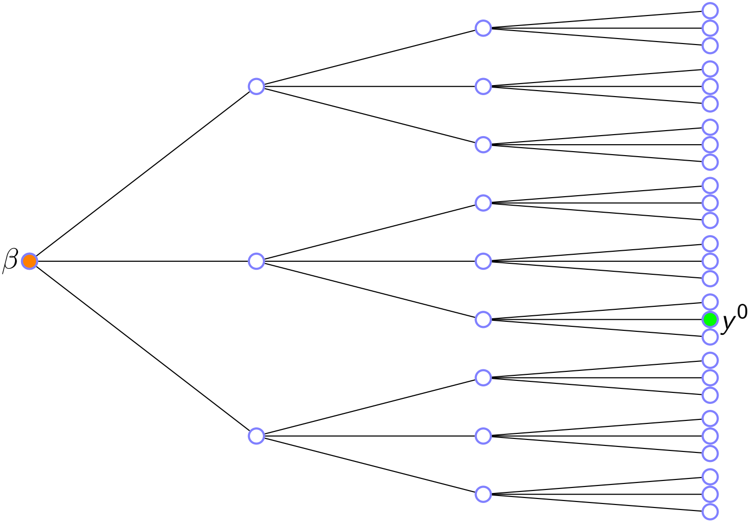
Solving **(P)**

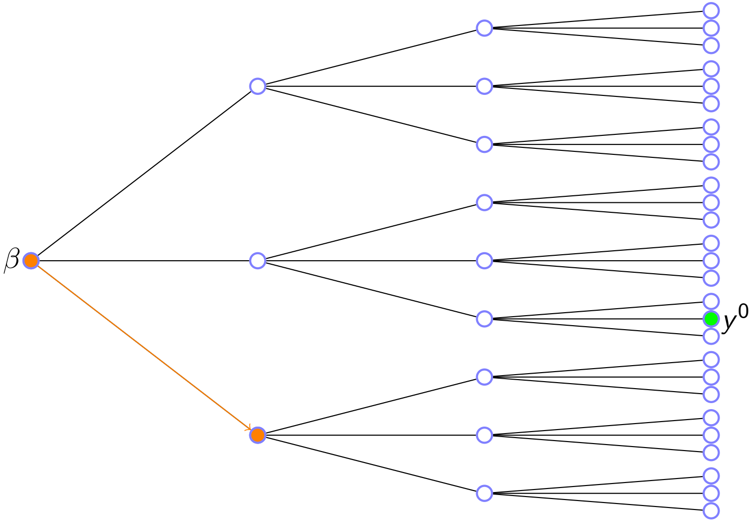
- Let c_0 be the cost for y^0 feasible point for **(P)**
- We calculate β optimum for **(LP)**.
- If β is feasible for **(P)**, it is optimum for **(P)**.
- If β is not feasible for **(P)**, we use the reverse skeleton $\mathcal{G}'_{>_c}$.
- For any γ obtained from $\mathcal{G}'_{>_c}$:
  - If $c(\gamma) > c_0$, we prune the branch
  - If $\gamma_i < 0$ we prune the branch

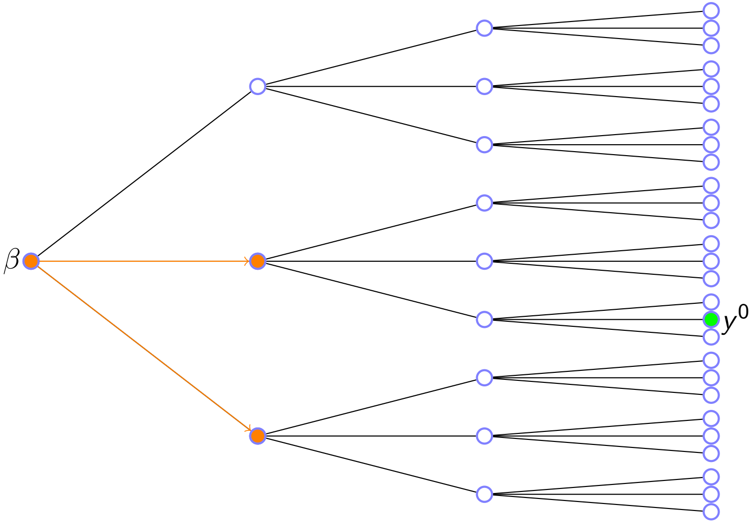
Solving (P)

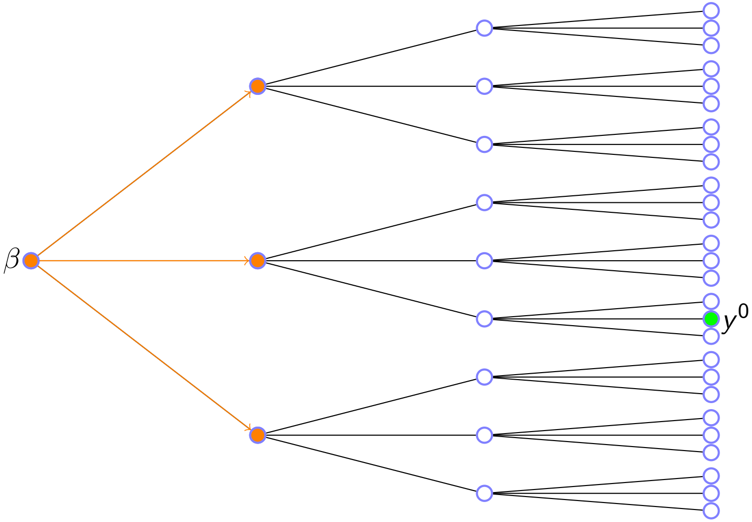
- Let c_0 be the cost for y^0 feasible point for **(P)**
- We calculate β optimum for **(LP)**.
- If β is feasible for **(P)**, it is optimum for **(P)**.
- If β is not feasible for **(P)**, we use the reverse skeleton $\mathcal{G}'_{>_c}$.
- For any γ obtained from $\mathcal{G}'_{>_c}$:
  - If $c(\gamma) > c_0$, we prune the branch
  - If $\gamma_i < 0$ we prune the branch
  - If γ is feasible for **(P)**, and $c(\gamma) < c_0$, we actualize c_0 and y^0.

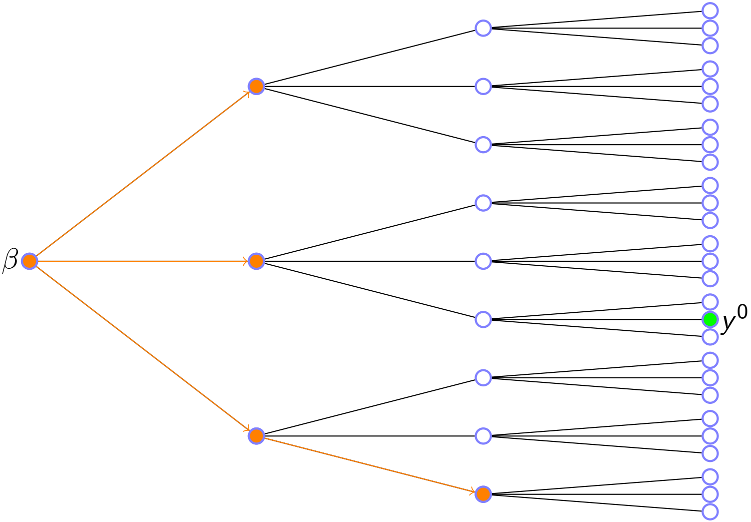


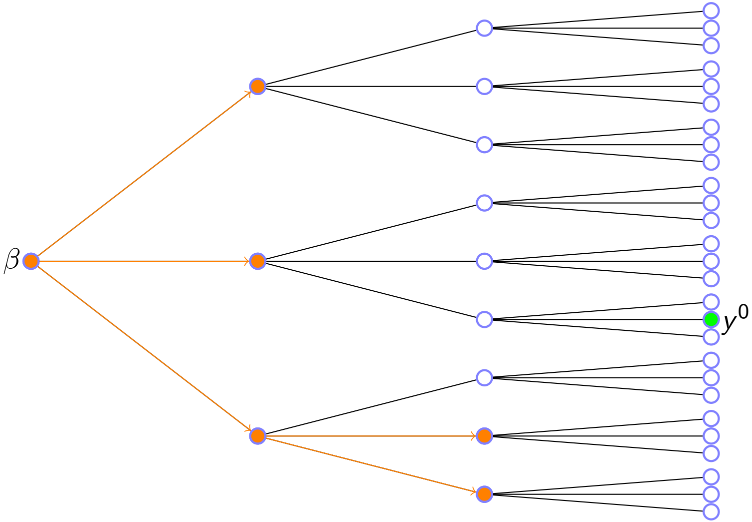

β
y^0

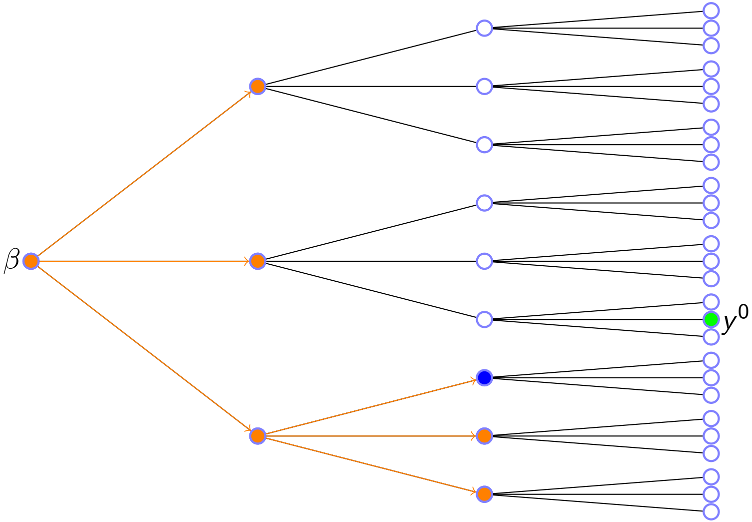

β
y^0

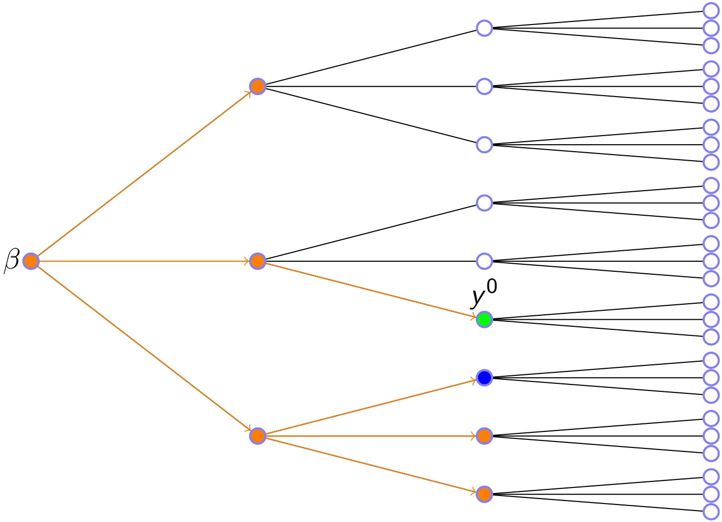

β
y^0

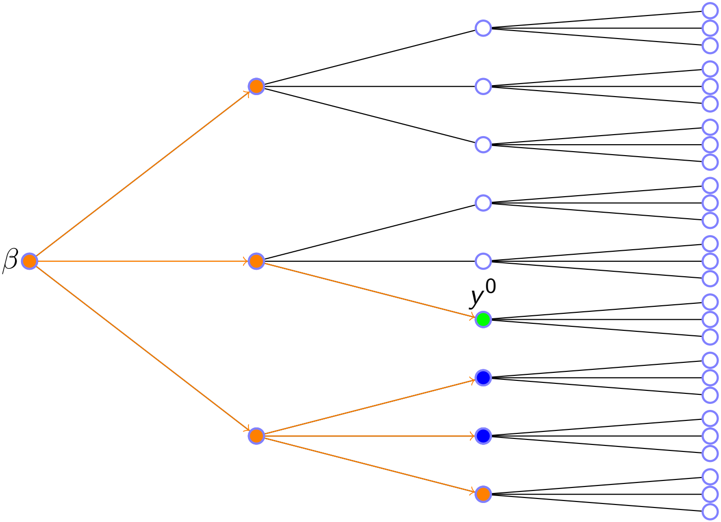

β
y^0

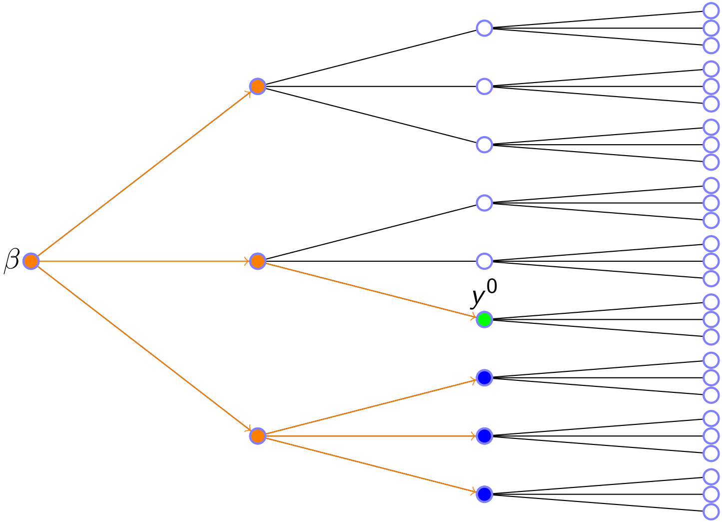

β
y^0

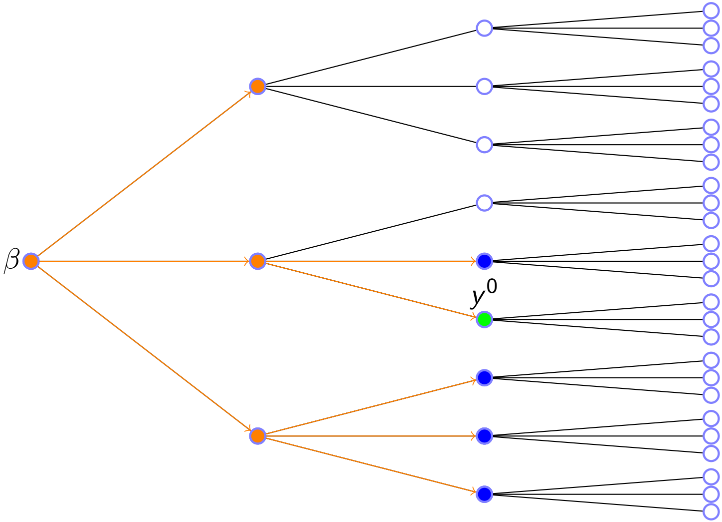

β
y^0

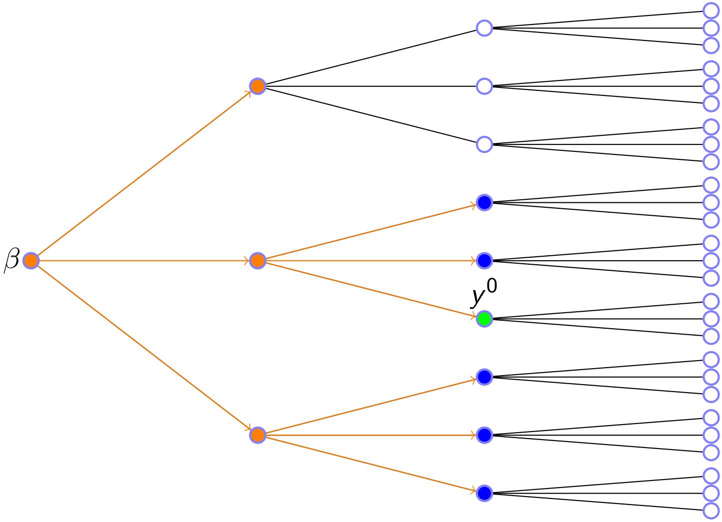

β
y^0

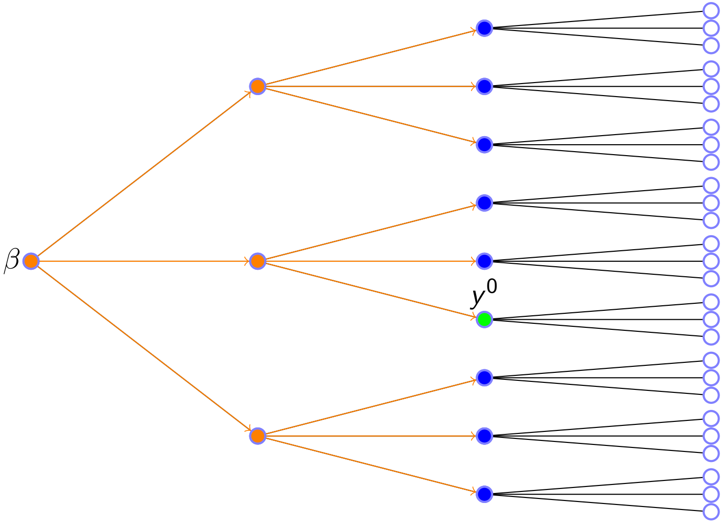

β
y^0


β
y0


β
y^0


β
y^0


β
y^0


β
y^0

Advantages and disadvantages of the search

Advantages

- The walk back gives, in an ordered way by the cost, all the feasible points of **(P)**.

Advantages and disadvantages of the search

Advantages

- The walk back gives, in an ordered way by the cost, all the feasible points of **(P)**.
- A new feasible point y^0 which improves the cost, discards all the pending nodes of greater or equal cost.

Advantages and disadvantages of the search

Advantages

- The walk back gives, in an ordered way by the cost, all the feasible points of **(P)**.
- A new feasible point y^0 which improves the cost, discards all the pending nodes of greater or equal cost.

Disadvantages

Advantages and disadvantages of the search

Advantages

- The walk back gives, in an ordered way by the cost, all the feasible points of **(P)**.
- A new feasible point y^0 which improves the cost, discards all the pending nodes of greater or equal cost.

Disadvantages

- If the feasible points of **(P)** are very far from β (optimum for **(LP)**), the number of nodes to be processed is huge.

Advantages and disadvantages of the search

Advantages

- The walk back gives, in an ordered way by the cost, all the feasible points of **(P)**.
- A new feasible point y^0 which improves the cost, discards all the pending nodes of greater or equal cost.

Disadvantages

- If the feasible points of **(P)** are very far from β (optimum for **(LP)**), the number of nodes to be processed is huge.
- If we can add constraints which shrink the feasible region, the test set changes, and the size may increase

Problems in the search

Breadth First Search

Basically, the Walk-back method performs a Breadth First Search using the test set.

Problems in the search

Breadth First Search

Basically, the Walk-back method performs a Breadth First Search using the test set.

Sturmfels en "Gröbner bases and Convex polytopes"

"One drawback of Algorithm 5.7 as presented is that the set `Active` can grow very large during the computation. This problem can be resolved by applying the "reverse search" technique of (Avis & Fukuda 1992). The reserve search variant requires no intermediate storage whatsoever, and it runs in linear time in the size of the output"

Improving the search

Processing

The size of the `Active` nodes list grows very quickly

- If we order them by cost, the computational cost is very high

Improving the search

Processing

The size of the `Active` nodes list grows very quickly

- If we order them by cost, the computational cost is very high
- It takes a long time to reach the **best** points

Improving the search

Processing

The size of the `Active` nodes list grows very quickly

- If we order them by cost, the computational cost is very high
- It takes a long time to reach the **best** points
- We need to order the pending nodes list with a balance between costs and feasibility for **(P)**.

Notation

$$
\textbf{(P)} \quad \begin{aligned} &\min c'x \\ &x \in \mathcal{A} \\ &x \in \mathcal{B} \end{aligned}
$$

Notation

$$\textbf{(P)} \quad \begin{aligned} & \min c'x \\ & x \in \mathcal{A} \\ & x \in \mathcal{B} \end{aligned}$$

$$\textbf{(P)} \quad \begin{aligned} & \min c'x \\ & Ax = b \\ & g_1(x) \leq C_1 \\ & \vdots \\ & g_m(x) \leq C_m \end{aligned}$$

Notation

$$\textbf{(P)} \quad \begin{aligned} &\min c'x \\ &x \in \mathcal{A} \\ &x \in \mathcal{B} \end{aligned}$$

$$\textbf{(P)} \quad \begin{aligned} &\min c'x \\ &Ax = b \\ &g_1(x) \leq C_1 \\ &\vdots \\ &g_m(x) \leq C_m \end{aligned}$$

$$\textbf{(LP)} \quad \begin{aligned} &\min c'x \\ &x \in \mathcal{A} \end{aligned}$$

Notation

$$\textbf{(P)} \quad \begin{aligned} &\min c'x \\ &x \in \mathcal{A} \\ &x \in \mathcal{B} \end{aligned}$$

$$\textbf{(P)} \quad \begin{aligned} &\min c'x \\ &Ax = b \\ &g_1(x) \leq C_1 \\ &\vdots \\ &g_m(x) \leq C_m \end{aligned}$$

$$\textbf{(LP)} \quad \begin{aligned} &\min c'x \\ &x \in \mathcal{A} \end{aligned}$$

$$\textbf{(LP)} \quad \begin{aligned} &\min c'x \\ &Ax = b \end{aligned}$$

Penalized cost function

Penalized cost function

- $c_p(x) = c(x) + P(x)$

Penalized cost function

Penalized cost function

- $c_p(x) = c(x) + P(x)$
- $P(x) = p(d(x, \mathcal{B}))$, d distance from x to $\mathcal{B}$.

Penalized cost function

Penalized cost function

- $c_p(x) = c(x) + P(x)$
- $P(x) = p(d(x, \mathcal{B}))$, d distance from x to $\mathcal{B}$.
  - If $\mathcal{B}$ is given by $g_i(x) \leq C_i$: $P(x) = \sum \max(g_i(x) - C_i, 0)$

Penalized cost function

Penalized cost function

- $c_p(x) = c(x) + P(x)$
- $P(x) = p(d(x, \mathcal{B}))$, d distance from x to $\mathcal{B}$.
    - If $\mathcal{B}$ is given by $g_i(x) \leq C_i$: $P(x) = \sum \max(g_i(x) - C_i, 0)$
    - $c_p(x) = c(x) + \mu P(x)$ (static penalization)

Penalized cost function

Penalized cost function

- $c_p(x) = c(x) + P(x)$
- $P(x) = p(d(x, \mathcal{B}))$, d distance from x to $\mathcal{B}$.
    - If $\mathcal{B}$ is given by $g_i(x) \leq C_i$: $P(x) = \sum \max(g_i(x) - C_i, 0)$
    - $c_p(x) = c(x) + \mu P(x)$ (static penalization)
    - $r(x) = (\max(g_1(x) - C_1, 0), \ldots, \max(g_m(x) - C_m), 0)$

Penalized cost function

Penalized cost function

- $c_p(x) = c(x) + P(x)$
- $P(x) = p(d(x, \mathcal{B}))$, d distance from x to $\mathcal{B}$.
  - If $\mathcal{B}$ is given by $g_i(x) \leq C_i$: $P(x) = \sum \max(g_i(x) - C_i, 0)$
  - $c_p(x) = c(x) + \mu P(x)$ (static penalization)
  - $r(x) = (\max(g_1(x) - C_1, 0), \ldots, \max(g_m(x) - C_m), 0)$
  - $c_p(x) = c(x) + \lambda(t) \parallel r(x) \parallel$ (adaptive penalization)

Penalized cost function

Penalized cost function

- $c_p(x) = c(x) + P(x)$
- $P(x) = p(d(x, \mathcal{B}))$, d distance from x to $\mathcal{B}$.
    - If $\mathcal{B}$ is given by $g_i(x) \leq C_i$: $P(x) = \sum \max(g_i(x) - C_i, 0)$
    - $c_p(x) = c(x) + \mu P(x)$ (static penalization)
    - $r(x) = (\max(g_1(x) - C_1, 0), \ldots, \max(g_m(x) - C_m), 0)$
    - $c_p(x) = c(x) + \lambda(t) \parallel r(x) \parallel$ (adaptive penalization)
- $c_p(x) = c(x) \cdot (2 - D(x))$

Penalized cost function

Penalized cost function

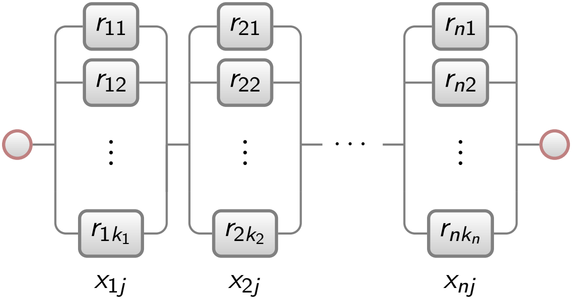
- $c_p(x) = c(x) + P(x)$
- $P(x) = p(d(x, \mathcal{B}))$, d distance from x to $\mathcal{B}$.
  - If $\mathcal{B}$ is given by $g_i(x) \leq C_i$: $P(x) = \sum \max(g_i(x) - C_i, 0)$
  - $c_p(x) = c(x) + \mu P(x)$ (static penalization)
  - $r(x) = (\max(g_1(x) - C_1, 0), \ldots, \max(g_m(x) - C_m), 0)$
  - $c_p(x) = c(x) + \lambda(t) \parallel r(x) \parallel$ (adaptive penalization)
- $c_p(x) = c(x) \cdot (2 - D(x))$
  - $D(x) = (\prod d_i(x))^{1/m}$

Penalized cost function

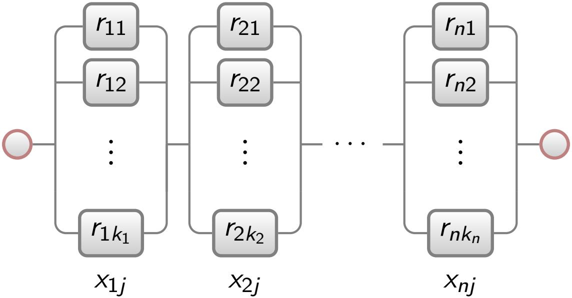
Penalized cost function

- $c_p(x) = c(x) + P(x)$
- $P(x) = p(d(x, \mathcal{B}))$, d distance from x to $\mathcal{B}$.
  - If $\mathcal{B}$ is given by $g_i(x) \leq C_i$: $P(x) = \sum \max(g_i(x) - C_i, 0)$
  - $c_p(x) = c(x) + \mu P(x)$ (static penalization)
  - $r(x) = (\max(g_1(x) - C_1, 0), \ldots, \max(g_m(x) - C_m), 0)$
  - $c_p(x) = c(x) + \lambda(t) \parallel r(x) \parallel$ (adaptive penalization)
- $c_p(x) = c(x) \cdot (2 - D(x))$
  - $D(x) = (\prod d_i(x))^{1/m}$
  - $$d_i(x) = \begin{cases} 1 & \text{if } g_i(x) \leq C_i \\ \left| \frac{C_i}{g_i(x)} \right| & \text{if } g_i(x) > C_i \end{cases}$$

Series-parallel systems



Series-parallel systems



$$R(x) = \prod_{i=1}^{n}(1 - \prod_{j=1}^{k_j}(1 - r_{ij})^{x_{ij}})$$

Problem formulation

$$\textbf{(RP)} \qquad \min \sum_{i,j} c_{ij} x_{ij}$$

$$\text{s.t.} \quad R(x) = \prod_{i=1}^{n} \left(1 - \prod_{j=1}^{k_j} (1 - r_{ij})^{x_{ij}}\right) \geq R_0,$$

$$0 \leq x_{ij} \leq u_{ij}$$

$$\sum_j x_{ij} \geq 1$$

Problem formulation

$$\textbf{(RP)} \qquad \min \sum_{i,j} c_{ij} x_{ij}$$

$$\text{s.t.} \quad R(x) = \prod_{i=1}^{n} \left(1 - \prod_{j=1}^{k_j} (1 - r_{ij})^{x_{ij}}\right) \geq R_0,$$

$$0 \leq x_{ij} \leq u_{ij}$$

$$\sum_j x_{ij} \geq 1$$

$$\textbf{(LRP)} \quad \min \sum_{i,j} c_{ij} x_{ij}$$

$$0 \leq x_{ij} \leq u_{ij}$$

$$\sum_j x_{ij} \geq 1$$

Test set

Test set

$$
\textbf{(LRP)} \quad \begin{array}{ll}
\min & \sum_{i=1}^{n} \sum_{j=1}^{k_i} c_{ij} x_{ij} \\
\text{s.t.} & \\
& \sum_{j=1}^{k_i} x_{ij} - d_i = 1, \quad i = 1, \ldots, n, \\
& x_{ij} + t_{ij} = u_{ij}, \qquad\quad i = 1, \ldots, n, \\
& \qquad\qquad\qquad\qquad\quad j = 1, \ldots, k_i,
\end{array}
$$

Test set

$$
\textbf{(LRP)} \quad \begin{array}{ll} \min & \sum_{i=1}^{n} \sum_{j=1}^{k_i} c_{ij} x_{ij} \\ \text{s.t.} & \\ & \sum_{j=1}^{k_i} x_{ij} - d_i = 1, \quad i = 1, \ldots, n, \\ & x_{ij} + t_{ij} = u_{ij}, \qquad i = 1, \ldots, n, \\ & \qquad\qquad\qquad\quad\;\; j = 1, \ldots, k_i, \end{array}
$$

$$
\mathcal{G} = \{\underline{x_{ik} d_i} - t_{ik}, \underline{x_{iq} t_{ip}} - x_{ip} t_{iq}\} \quad c_{iq} > c_{ip}
$$

Main strategy

1. Starting from the optimum of (**LP**) we use the reverse test set to walk back into the feasible region of (**P**).

Main strategy

1. Starting from the optimum of (**LP**) we use the reverse test set to walk back into the feasible region of (**P**).
2. We calculate a feasible point y^0 for **(RP)**, with cost c^0

Main strategy

1. Starting from the optimum of (**LP**) we use the reverse test set to walk back into the feasible region of (**P**).
2. We calculate a feasible point y^0 for **(RP)**, with cost c^0
3. Given a new node **w**, with reliability R_w we order the active nodes by

$$c_p(w) = \mathbf{c}^{\mathbf{t}} \cdot \mathbf{w} + \mu \cdot \max\{0, R_0 - R_w\}$$

Main strategy

1. Starting from the optimum of (**LP**) we use the reverse test set to walk back into the feasible region of (**P**).
2. We calculate a feasible point y^0 for **(RP)**, with cost c^0
3. Given a new node **w**, with reliability R_w we order the active nodes by

$$c_p(w) = \mathbf{c^t} \cdot \mathbf{w} + \mu \cdot \max\{0, R_0 - R_w\}$$

Main strategy

1. Starting from the optimum of (**LP**) we use the reverse test set to walk back into the feasible region of (**P**).
2. We calculate a feasible point y^0 for **(RP)**, with cost c^0
3. Given a new node **w**, with reliability R_w we order the active nodes by

$$c_p(w) = \mathbf{c}^{\mathbf{t}} \cdot \mathbf{w} + \mu \cdot \max\{0, R_0 - R_w\}$$

with

$$\mu = \frac{c_Y - c\beta}{R_0 - R_\beta}$$

c_Y best cost for a feasible point for **(RP)**, initially $c_Y = c^0$

Improving the process

Advantages

- It improves the “naked” walk-back method

Improving the process

Advantages

- It improves the “naked” walk-back method
- The process reaches very quickly a very good point

Improving the process

Advantages

- It improves the "naked" walk-back method
- The process reaches very quickly a very good point

Drawbacks

- The optimality certification is very slow due to the list of pending nodes.

Improving the process

Advantages

- It improves the “naked” walk-back method
- The process reaches very quickly a very good point

Drawbacks

- The optimality certification is very slow due to the list of pending nodes.

We can combine our method with a Breadth First Search, to improve the performance of the pending nodes

Improved main strategy

1. Starting from the optimum of (**LP**) we use the reverse test set to walk back into the feasible region of (**P**).

Improved main strategy

1. Starting from the optimum of (**LP**) we use the reverse test set to walk back into the feasible region of (**P**).
2. We calculate a feasible point y^0 for **(RP)**, with cost c^0

Improved main strategy

1. Starting from the optimum of (**LP**) we use the reverse test set to walk back into the feasible region of (**P**).
2. We calculate a feasible point y^0 for **(RP)**, with cost c^0
3. Given a new node **w**, with reliability R_w we order the active nodes by

$$c_p(w) = \begin{cases} \mathbf{c}^{\mathbf{t}} \cdot \mathbf{w} + \mu \cdot \max\{0, R_0 - R_w\} & \text{if visited nodes} \leq L \\ -\mathbf{c}^{\mathbf{t}} \cdot \mathbf{w} & \text{if visited nodes} > L \end{cases}$$

Computational results

Table: **Correlation, high reliabilities**
$r_{ij} \in [0.99, 0.998]$, $c_{ij} \in [10, 20]$, $u_{ij} = 4$, $R_0 = 0.90$, average time and average number of nodes

		Walk-back			Penalty		
n	k	T	Nodes	$>$ Limit	T	Nodes	$>$ Limit
15	3	322.8	6965	1	42.3	4795	0
15	4	571.1	17629	13	432.6	21843	8
17	2	92.7	6465	1	15.4	3813	1

Computational results

Table: **Correlation, lower reliabilities**
$r_{ij} \in [0.90, 0.99]$, $c_{ij} \in [10, 20]$, $u_{ij} = 4$, $R_0 = 0.90$, average time and average number of nodes

		Walk-back			Penalty		
n	k	T	Nodes	$>$ Limit	T	Nodes	$>$ Limit
7	5	23.6	6157	0	17.9	5301	0
8	4	593.5	37728	7	373.4	34754	1

Comparison with other solvers

Table: Comparison with other solvers in the case $n = 8, k = 4$ of table 2

	WB Penalty		Baron		Couenne		Bonmin	
Ex	T	Cost	T	Cost	T	Cost	T	Cost
04	158.4	119	43.8	119	N/F		165.4	119
15	650.4	123	89.1	124(*)	N/F		604.1	123
16	192.2	121	51.1	121	1017.5	121	456.8	121
18	54.3	113	30.2	113	247.8	113	146.3	113
19	15.8	114	13.7	114	68.6	114	88.5	114
30	112.6	124	33.8	125(*)	805.5	124	178.8	124

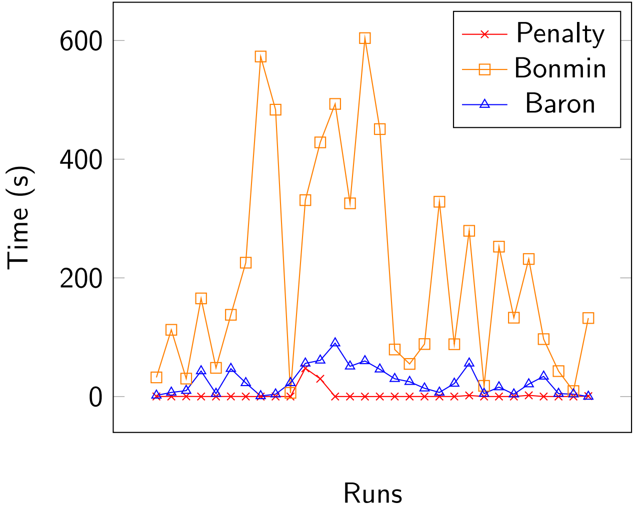


Figure: Spent time to reach the optimum: Penalty, Bonmin and Baron

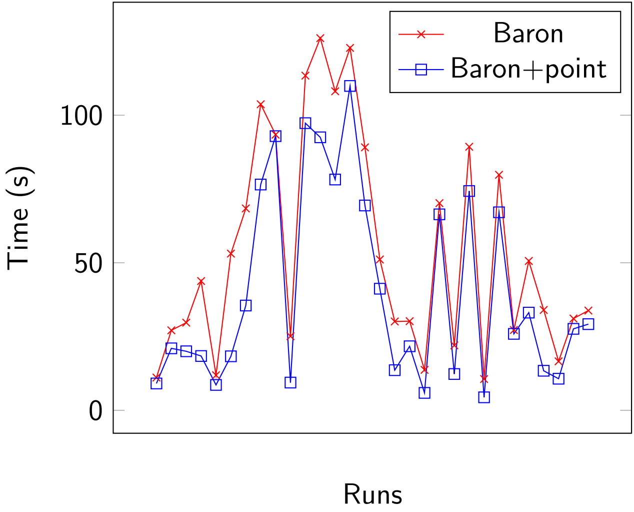


Figure: Verification Time: Baron and Baron with initial point provided by Penalty

Assignment of jobs

The treated problem in Tayur, Thomas and Natraj, consists in assigning jobs to machines, with given production and correlated setup costs, capacity constraints and probability to reach a given demand.

Assignment of jobs

The treated problem in Tayur, Thomas and Natraj, consists in assigning jobs to machines, with given production and correlated setup costs, capacity constraints and probability to reach a given demand.
This probability constraint is of the following form:

$$\mathrm{Prob}(\tilde{T}x \leq \boldsymbol{C}) \geq \gamma$$

where $\tilde{T}$ is the technology matrix, and represents a joint probabilistic constraint, where it is important to have all constraints satisfied simultaneously and there may be dependence between random variables in different rows.

Model

- n number of job types, indexed by i,

Model

- n number of job types, indexed by i,
- m number of machines, indexed by j,

Model

- n number of job types, indexed by i,
- m number of machines, indexed by j,
- $(D_1, \ldots, D_n)$ random vector of demands,

Model

- n number of job types, indexed by i,
- m number of machines, indexed by j,
- $(D_1, \ldots, D_n)$ random vector of demands,
- C_j capacity (time) for each machine,

Model

- n number of job types, indexed by i,
- m number of machines, indexed by j,
- $(D_1, \ldots, D_n)$ random vector of demands,
- C_j capacity (time) for each machine,
- $(\hat{D}_1, \ldots, \hat{D}_n)$ means vector of the probability distribution of demand,

Model

- n number of job types, indexed by i,
- m number of machines, indexed by j,
- $(D_1, \dots, D_n)$ random vector of demands,
- C_j capacity (time) for each machine,
- $(\hat{D}_1, \dots, \hat{D}_n)$ means vector of the probability distribution of demand,
- S_{ij} setup time for job type i on machine j,

Model

- n number of job types, indexed by i,
- m number of machines, indexed by j,
- $(D_1, \ldots, D_n)$ random vector of demands,
- C_j capacity (time) for each machine,
- $(\hat{D}_1, \ldots, \hat{D}_n)$ means vector of the probability distribution of demand,
- S_{ij} setup time for job type i on machine j,
- K_{ij} setup cost for job type i on machine j,

Model

- M_i lot splitting,

Model

- M_i lot splitting,
- L'_{ij} the cost of producing a unit of product type i on machine j,

Model

- M_i lot splitting,
- L'_{ij} the cost of producing a unit of product type i on machine j,
- $L_{ij} = (\hat{D}_i / M_i) L'_{ij}$,

Model

- M_i lot splitting,
- L'_{ij} the cost of producing a unit of product type i on machine j,
- $L_{ij} = (\hat{D}_i / M_i) L'_{ij}$,
- p_{ij} processing time for a unit of job type i on machine j,

Model

- M_i lot splitting,
- L'_{ij} the cost of producing a unit of product type i on machine j,
- $L_{ij} = (\hat{D}_i/M_i)L'_{ij}$,
- p_{ij} processing time for a unit of job type i on machine j,
- γ probability of no shortfall.

Model

- M_i lot splitting,
- L'_{ij} the cost of producing a unit of product type i on machine j,
- $L_{ij} = (\hat{D}_i/M_i)L'_{ij}$,
- p_{ij} processing time for a unit of job type i on machine j,
- γ probability of no shortfall.
- z_{ij} equals 1 if job type i is scheduled on machine j, 0 otherwise,

Model

- M_i lot splitting,
- L'_{ij} the cost of producing a unit of product type i on machine j,
- $L_{ij} = (\hat{D}_i/M_i)L'_{ij}$,
- p_{ij} processing time for a unit of job type i on machine j,
- γ probability of no shortfall.
- z_{ij} equals 1 if job type i is scheduled on machine j, 0 otherwise,
- y_{ij} multiples of $1/M_i$ of demand of product i s scheduled on machine j.

Model

$$(\mathrm{SP}) \min \sum_i \sum_j (K_{ij} z_{ij} + L_{ij} y_{ij})$$

$$\text{s.t.} \sum_{j=1}^{m} y_{ij} = M_i, i = 1, 2, \ldots, n, \tag{1}$$

$$M_i z_{ij} \geq y_{ij}, i = 1, 2, \ldots, n, j = 1, 2, \ldots, m, \tag{2}$$

$$\sum_{i=1}^{n} p_{ij} \left(\hat{D}_i / M_i \right) y_{ij} + \sum_{i=1}^{n} S_{ij} z_{ij} \leq C_j, j = 1, 2, \ldots, m, \tag{3}$$

$$\mathrm{Prob} \left\{ \sum_{i=1}^{n} p_{ij} \left(D_i / M_i \right) y_{ij} + \sum_{i=1}^{n} S_{ij} z_{ij} \leq C_j, j = 1, 2, \ldots, m \right\} \geq \gamma, \tag{4}$$

$$z_{ij} \in \{0, 1\}, y_{ij} \in \{0, 1, \ldots, M_i\} \tag{5}$$

Relaxed problem

$$(\mathrm{LSP})\ \min \sum_i \sum_j (K_{ij} z_{ij} + L_{ij} y_{ij})$$

s.t.

$$\sum_{j=1}^{m} y_{ij} = M_i, i = 1, 2, \ldots, n, \tag{6}$$

$$M_i z_{ij} \geq y_{ij}, i = 1, 2, \ldots, n, j = 1, 2, \ldots, m, \tag{7}$$

$$z_{ij} \in \{0, 1\}, y_{ij} \in \{0, 1, \ldots, M_i\}. \tag{8}$$

Penalized cost function

If $\boldsymbol{x}$ is the set of variables y_{ij}, z_{ij}, we consider:

$$G_0(\boldsymbol{x}) = \gamma - g_0(\boldsymbol{x}),$$

$$g_0(\boldsymbol{x}) = \mathrm{Prob}\left\{\sum_{i=1}^{n} p_{ij}\left(D_i/M_i\right)y_{ij} + \sum_{i=1}^{n} S_{ij}z_{ij} \le C_j, j = 1, 2, \ldots, m\right\},$$

$$G_j(\boldsymbol{x}) = g_j(\boldsymbol{x}) - C_j,$$

$$g_j(\boldsymbol{x}) = \sum_{i=1}^{n} p_{ij}\left(\hat{D}_i/M_i\right)y_{ij} + \sum_{i=1}^{n} S_{ij}z_{ij}, j = 1, 2, \ldots, m.$$

Penalized cost function

The first penalized function is: $T_0(\boldsymbol{x}) = c(\boldsymbol{x}) + \mu P(\boldsymbol{x})$

Penalized cost function

The first penalized function is: $T_0(\boldsymbol{x}) = c(\boldsymbol{x}) + \mu P(\boldsymbol{x})$ where

$$P(\boldsymbol{x}) = \sum_{j=0}^{m} \max(G_j(\boldsymbol{x}), 0).$$

in order to calculate μ, we consider

Penalized cost function

The first penalized function is: $T_0(\boldsymbol{x}) = c(\boldsymbol{x}) + \mu P(\boldsymbol{x})$ where

$$P(\boldsymbol{x}) = \sum_{j=0}^{m} \max(G_j(\boldsymbol{x}), 0).$$

in order to calculate μ, we consider

- $\boldsymbol{p}$ optimum of the relaxed problem (LSP),

Penalized cost function

The first penalized function is: $T_0(\boldsymbol{x}) = c(\boldsymbol{x}) + \mu P(\boldsymbol{x})$ where

$$P(\boldsymbol{x}) = \sum_{j=0}^{m} \max(G_j(\boldsymbol{x}), 0).$$

in order to calculate μ, we consider

- $\boldsymbol{p}$ optimum of the relaxed problem (LSP),
- $\rho = G_0(\boldsymbol{p})$, $c_0 = c(\boldsymbol{p})$

Penalized cost function

The first penalized function is: $T_0(\boldsymbol{x}) = c(\boldsymbol{x}) + \mu P(\boldsymbol{x})$ where

$$P(\boldsymbol{x}) = \sum_{j=0}^{m} \max(G_j(\boldsymbol{x}), 0).$$

in order to calculate μ, we consider

- $\boldsymbol{p}$ optimum of the relaxed problem (LSP),
- $\rho = G_0(\boldsymbol{p})$, $c_0 = c(\boldsymbol{p})$
- $c_1 = \sum_i \sum_j (K_{ij} + M_i L_{ij})$

Penalized cost function

The first penalized function is: $T_0(\boldsymbol{x}) = c(\boldsymbol{x}) + \mu P(\boldsymbol{x})$ where

$$P(\boldsymbol{x}) = \sum_{j=0}^{m} \max(G_j(\boldsymbol{x}), 0).$$

in order to calculate μ, we consider

- $\boldsymbol{p}$ optimum of the relaxed problem (LSP),
- $\rho = G_0(\boldsymbol{p})$, $c_0 = c(\boldsymbol{p})$
- $c_1 = \sum_i \sum_j (K_{ij} + M_i L_{ij})$

$$\mu_0 = \frac{c_1 - c_0}{\rho}, \alpha = \lfloor \log(\frac{c_0}{\mu_0}) \rfloor, \text{ and } \mu = \alpha \mu_0.$$

Penalized cost function

The second penalized function is:

$$T_1(\boldsymbol{x}) = c(\boldsymbol{x})(2 - D(\boldsymbol{x}))$$

Penalized cost function

The second penalized function is:

$$T_1(\boldsymbol{x}) = c(\boldsymbol{x})(2 - D(\boldsymbol{x}))$$

where $D(\boldsymbol{x}) = \left(\prod_{j=0}^{m} d_j(\boldsymbol{x})\right)^{1/(m+1)}$ and

Penalized cost function

The second penalized function is:

$$T_1(\boldsymbol{x}) = c(\boldsymbol{x})(2 - D(\boldsymbol{x}))$$

where $D(\boldsymbol{x}) = \left(\prod_{j=0}^{m} d_j(\boldsymbol{x})\right)^{1/(m+1)}$ and

$$d_0(\boldsymbol{x}) = \begin{cases} 1 & \text{if } g_0(\boldsymbol{x}) \geq \gamma, \\ \frac{g_0(\boldsymbol{x})}{\gamma} & \text{if } g_0(\boldsymbol{x}) < \gamma. \end{cases}$$

$$d_j(\boldsymbol{x}) = \begin{cases} 1 & \text{if } g_j(\boldsymbol{x}) \leq C_j, \\ \frac{C_j}{g_j(\boldsymbol{x})} & \text{if } g_j(\boldsymbol{x}) \geq C_j. \end{cases} \quad j = 1, 2, \ldots, m,$$

Penalized cost function

The second penalized function is:

$$T_1(\boldsymbol{x}) = c(\boldsymbol{x})(2 - D(\boldsymbol{x}))$$

where $D(\boldsymbol{x}) = \left(\prod_{j=0}^{m} d_j(\boldsymbol{x})\right)^{1/(m+1)}$ and

$$d_0(\boldsymbol{x}) = \begin{cases} 1 & \text{if } g_0(\boldsymbol{x}) \geq \gamma, \\ \frac{g_0(\boldsymbol{x})}{\gamma} & \text{if } g_0(\boldsymbol{x}) < \gamma. \end{cases}$$

$$d_j(\boldsymbol{x}) = \begin{cases} 1 & \text{if } g_j(\boldsymbol{x}) \leq C_j, \\ \frac{C_j}{g_j(\boldsymbol{x})} & \text{if } g_j(\boldsymbol{x}) \geq C_j. \end{cases} \quad j = 1, 2, \ldots, m,$$

Computational results

Table: 4 machines, 7 jobs, $M = 2$, original model

	Walk-back				Penalty T_0			
	Total		Optimum		Total		Optimum	
γ	T	Nodes	T	Nodes	T	Nodes	T	Nodes
0.888	148.7	13708	91.2	12409	54.3	8861	0.8	596
0.900	149.3	13709	92.1	12410	54.8	8865	0.8	600
0.932	237.3	18560	29.2	7132	196.3	18498	1.1	777
0.956		Max		NNP		Max	6194.5	83857
0.960		Max		NNP		Max	14397.0	116797
0.980		Max		NNP		Max		NNP

Computational results

Table: 4 machines, 7 jobs, $M = 2$, original model

	Walk-back				Penalty T_1			
	Total		Optimum		Total		Optimum	
γ	T	Nodes	T	Nodes	T	Nodes	T	Nodes
0.888	148.7	13708	91.2	12409	54.1	8972	1.1	733
0.900	149.3	13709	92.1	12410	53.9	8931	1.1	737
0.932	237.3	18560	29.2	7132	192.8	18516	1.3	898
0.956		Max		NNP		Max	239.7	18184
0.960		Max		NNP		Max	5614.5	78284
0.980		Max		NNP		Max		NNP

Improving the model

The model for the original problemadmits an additional valid inequality

$$z_{ij} \leq y_{ij}, i = 1, \dots, n, j = 1, \dots, m. \tag{9}$$

Improving the model

The model for the original problemadmits an additional valid inequality

$$z_{ij} \leq y_{ij}, i = 1, \ldots, n, j = 1, \ldots, m. \tag{9}$$

This constraint reduces the size of the feasible region

Improving the model

The model for the original problemadmits an additional valid inequality

$$z_{ij} \leq y_{ij}, i = 1, \ldots, n, j = 1, \ldots, m. \quad (9)$$

This constraint reduces the size of the feasible region
The computation of the Gröbner basis of the linear problem remains very quickly (less than 1s)

Computational results

Table: 4 machines, 7 jobs, $M = 2$, improved model

	Walk-back				Penalty T_0			
	Total		Optimum		Total		Optimum	
γ	T	Nodes	T	Nodes	T	Nodes	T	Nodes
0.888	3.4	934	1.3	873	2.8	643	0.2	100
0.900	3.4	934	1.3	873	2.7	643	0.2	100
0.932	5.4	1299	1.4	893	4.1	913	0.1	63
0.956	176.9	15220	130.2	14670	64.8	8511	2.1	1145
0.960	534.9	28575	429.3	27799	263.7	19072	62.5	7973
0.980		Max		NNP	6355.7	98294	11.7	3360

Computational results

Table: 4 machines, 7 jobs, $M = 2$, improved model

	Walk-back				Penalty T_1			
	Total		Optimum		Total		Optimum	
γ	T	Nodes	T	Nodes	T	Nodes	T	Nodes
0.888	3.4	934	1.3	873	2.7	641	0.1	97
0.900	3.4	934	1.3	873	2.7	641	0.2	97
0.932	5.4	1299	1.4	893	4.1	913	0.1	63
0.956	176.9	15220	130.2	14670	60.9	8064	1.2	727
0.960	534.9	28575	429.3	27799	194.1	16172	22.4	4497
0.980		Max		NNP	6316.1	98920	87.5	9905

Comparison with other solvers

Other solvers

- The solver COUENNE returns the message “System error” and stops the execution without any point returned.

Comparison with other solvers

Other solvers

- The solver COUENNE returns the message "System error" and stops the execution without any point returned.
- BARON returns a point that is not feasible for the chance constraint, and it does not give any message about that.

Comparison with other solvers

Other solvers

- The solver COUENNE returns the message "System error" and stops the execution without any point returned.
- BARON returns a point that is not feasible for the chance constraint, and it does not give any message about that.
- BONMIN does not handle problems with a non convex feasible region, but it may return a feasible point. In this case, the process stops with a point that does not verify the chance constraint.

Integral bases

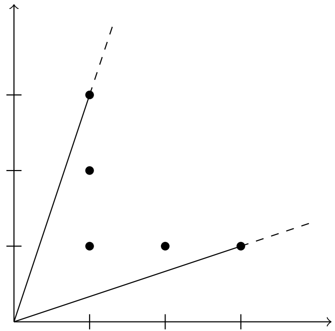
Integral generating set

Let $\mathcal{F} \subset \mathbb{Z}^n$. A set $H \subset \mathcal{F}$ is an **integral generating set** of $\mathcal{F}$ if for every $\mathbf{x} \in \mathcal{F}$ there exist $\{\mathbf{h}_1, \ldots, \mathbf{h}_k\} \subset H$ and multipliers $\lambda_1, \ldots, \lambda_k \in \mathbb{Z}_+$ such that

$$\mathbf{x} = \sum_{i=1}^{k} \lambda_i \mathbf{h}_i$$

An integral generating set H of $\mathcal{F}$ is called an **integral basis** if it is minimal with respect to inclusion.

Let $C = \{\mathbf{x} \in \mathbb{R}^2 \mid x_1 = \lambda_1 + 3\lambda_2,\ x_2 = 3\lambda_1 + \lambda_2,\ \lambda_1,\ \lambda_2 \geq 0\}$ C is the generated cone by the vectors $(1,3)', (3,1)'$



$$H = \{(1,3)', (1,2)', (1,1)', (2,1)', (3,1)'\}$$

is an integral basis of $\mathcal{F} = C \cap \mathbb{Z}^2$

Theorem

The set of all integer points in a rational polyhedral cone has a finite integral generating set, called Hilbert basis

Theorem

If a rational polyhedral cone C is pointed, then $\mathcal{F} = C \cap \mathbb{Z}^n$ has a unique integral basis

Graver basis

Fixed A an $m \times n$ matrix with integer entries.
We note $\mathbb{O}_j = j^{\text{th}}$ orthant of $\mathbb{R}^n$.
H_j =(unique) minimal Hilbert basis of $\ker_{\mathbb{R}^n}(A) \cap \mathbb{O}_j$

$$Gr(A) = \bigcup H_j \backslash \{0\}$$

is called the **Graver basis** of A.

Definition

We call $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ sign-compatible or **conformal** if $u_j v_j \geq 0$ for all components $j = 1, \ldots, n$.

Examples

The vectors $(4, -2, 0)$ and $(2, -3, 5)$ are conformal. And $(2, -3, 5)$ and $(-1, -4, 5)$ are not

$\sqsubseteq$

For $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ we say $\mathbf{u} \sqsubseteq \mathbf{v}$ if $\mathbf{u}$ and $\mathbf{v}$ are conformal and if $|u_j| \leq |v_j|$ for all $j = 1, \ldots n$; that is, if $\mathbf{u}$ belongs to the same orthant as $\mathbf{v}$ and if its components are not greater in absolute value than the corresponding components of $\mathbf{v}$.

Examples

$(4, -2, 0) \not\sqsubseteq (2, -3, 5)$, neither $(2, -3, 5) \not\sqsubseteq (4, -2, 0)$.
$(4, -2, 0) \sqsubseteq (7, -3, 5)$

Primitive binomials

A binomial $\mathbf{x}^{\mathbf{u}^+} - \mathbf{x}^{\mathbf{u}^-}$ in I_A is called **primitive** if there exists no other binomial $\mathbf{x}^{\mathbf{v}^+} - \mathbf{x}^{\mathbf{v}^-}$ in I_A such that $\mathbf{x}^{\mathbf{v}^+}$ divides $\mathbf{x}^{\mathbf{u}^+}$ and $\mathbf{x}^{\mathbf{v}^-}$ divides $\mathbf{x}^{\mathbf{u}^-}$. A primitive element is $\sqsubseteq$ minimal element in I_A

Gordan-Dickson lemma

Every infinite set $S \subset \mathbb{Z}_+^n$ contains only finitely many $\leq$-minimal points

Gordan-Dickson lemma, $\sqsubseteq$ version

Every sequence $\{\mathbf{p}_1, \mathbf{p}_2, \ldots\}$ of points in $\mathbb{Z}^n$ such that $\mathbf{p}_i \not\sqsubseteq \mathbf{p}_j$ whenever $i < j$ is finite

Every infinite set $S \subset \mathbb{Z}^n$ contains only finitely many $\sqsubseteq$-minimal points

$\sqsubseteq$-minimal elements

For any given matrix $A \in \mathbb{Z}^{m \times n}$ the set of minimal $\sqsubseteq$ elements in $\ker_{\mathbb{Z}^n}(A) \backslash \{0\}$ is finite

Lemma

$Gr(A)$ is the set of minimal $\sqsubseteq$ elements in $\ker_{\mathbb{Z}^n}(A) \backslash \{0\}$

Positive sum property

$Gr(A)$ has the **positive sum property** with respect to $\ker_{\mathbb{Z}^n}(A)$, that is, every $\mathbf{z} \in \ker_{\mathbb{Z}^n}(A)$ possesses a $\sqsubseteq$-representation with respect to $Gr(A)$:

$$z = \sum \alpha_i g_i \quad \alpha_i \in \mathbb{Z}_+, g_i \in Gr(A), \quad g_i \sqsubseteq z$$

$Gr(A)$ and PSP

$Gr(A)$ is the unique inclusion-minimal subset of $\ker_{\mathbb{Z}^n}(A)$ that has the positive sum property with respect to $\ker_{\mathbb{Z}^n}(A)$

Circuits and Graver basis

Circuits

A vector in $\ker_{\mathbb{Z}^n}(A)$ is called a circuit of A if its support is inclusion minimal among all elements in $\ker_{\mathbb{Z}^n}(A)$ and its components are integer and relatively prime.
Equivalently, a circuit is an irreducible binomial $\mathbf{x}^{\mathbf{u}^+} - \mathbf{x}^{\mathbf{u}^-}$ in I_A which has minimal support

Lemma

If $\mathbf{u}$ is a circuit of A then $supp(\mathbf{u})$ has at most $m+1$ elements.

$\mathcal{C}_A \subset Gr(A)$

Every circuit is primitive.
So, the set of circuits forms a subset of the Graver basis of A.

Universal Gröbner basis and Graver basis

Universal Gröbner basis

The universal Gröbner basis is the union of all reduced Gröbner bases and a Gröbner basis with respect to any monomial order. The universal Gröbner basis is finite

Lemma

Every binomial $\mathbf{x}^{\mathbf{u}^+} - \mathbf{x}^{\mathbf{u}^-}$ in the universal Gröbner basis $\mathcal{U}_A$ is an element of the Graver basis

$$\mathcal{C}_A \subset \mathcal{U}_A \subset Gr(A)$$

Let $A = (i\ j\ k) \in \mathbb{N}^{1\times 3}$ pairwise relatively prime, then $\mathcal{C}_A = \{x_1^j - x_2^i, x_1^k - x_3^i, x_2^k - x_3^j\}$. We can consider the following three cases:

- If $A = (1\ 2\ 3)$ then $\mathcal{U}_A = Gr(A) = \mathcal{C}_A \cup \{x_3 - x_1x_2, x_1x_3 - x_2^2\}$
- If $A = (1\ 2\ 4)$ then $\mathcal{C}_A = \mathcal{U}_A = \{x_1^2 - x_2, x_1^4 - x_3, x_2 - x_3^2\}$ and $Gr(A) \backslash \mathcal{U}_A = \{x_3 - x_1^2 - x_2\}$
- If $A = (1\ 2\ 5)$ then $\mathcal{U}_A \backslash \mathcal{C}_A = \{x_3 - x_1x_2^2, x_1x_3 - x_2^3\}$ and $Gr(A) \backslash \mathcal{U}_A = \{x_3 - x_1^3x_2\}$

Theorem

If A is a totally unimodular matrix

$$\mathcal{C}_A = \mathcal{U}_A = Gr(A)$$

Lawrence lifting

Consider the enlarged matrix

$$\Lambda(A) = \begin{pmatrix} A & \mathbf{0} \\ I_n & I_n \end{pmatrix}$$

Where I_n is the identity matrix and $\mathbf{0}$ is the $m \times n$ zero matrix. This $(m+n) \times 2n$-matrix is called the Lawrence lifting of A

Toric ideal

$$I_{\Lambda(A)} = \{\mathbf{x}^{\mathbf{u}^+}\mathbf{y}^{\mathbf{u}^-} - \mathbf{x}^{\mathbf{u}^-}\mathbf{y}^{\mathbf{u}^+} \,:\, \mathbf{u} \in \ker(A)\}$$

Theorem

For a Lawrence type matrix $\Lambda(A)$ the following sets of binomials coincide:

- The Graver basis of $\Lambda(A)$
- The universal Gröbner basis of $\Lambda(A)$
- Any reduced Gröbner basis of $I_{\Lambda(A)}$
- Any minimal generating set of $I_{\Lambda(A)}$ (up to scalar multiples)

Computing Graver basis;
Data: $A \in \mathbb{Z}^{m \times n}$
Result: $Gr(A)$
Choose any term order $>$ on $k[\mathbf{x}, \mathbf{y}]$;
Compute the reduced Gröbner basis G of $I_{\Lambda(A)}$ with respect to $>$;
Substitute $y_i \mapsto 1$ for any $g \in G$;
return G;

Pottier's algorithm

This algorithm computes the set of $\sqsubseteq$-minimal elements in a lattice $\mathcal{L}\backslash\{0\}$. We choose $\mathcal{L} = \ker_{\mathbb{Z}^n}(A)$.

Infinite test Criterion for PSP

A symmetric set $G \subset \mathcal{L}$ has the Positive Sum Property with respect to $\mathcal{L}$ if and only if every $z \in \mathcal{L}$ is $\sqsubseteq$-representable with respect to G

Finite test Criterion for PSP

A symmetric set $G \subset \mathcal{L}$ has the Positive Sum Property with respect to $\mathcal{L}$ if and only if G generates $\mathcal{L}$ over $\mathbb{Z}$ and if every sum $\mathbf{u} + \mathbf{v}$, $\mathbf{u}$, $\mathbf{v} \in G$, is $\sqsubseteq$-representable with respect to G

Normal form algorithm

Normal form

We give the algorithm to compute Normal Form $\mathbf{r}$ of an element $\mathbf{s}$ in the lattice $\mathcal{L}$ with respect to $G \subset \mathcal{L}$, such that $\mathbf{s} = \sum \alpha_i \mathbf{g}_i + \mathbf{r}$ with $\alpha_i \in \mathbb{Z}_+$, $\mathbf{g}_i, \mathbf{r} \sqsubseteq \mathbf{s}$ and $\mathbf{g}_i \in G$ and $g \not\sqsubseteq \mathbf{r}$ for all $g \in G$

Normal form algorithm;
Data: $\mathbf{s} \in \mathcal{L}$, set $G \subset \mathcal{L}$
Result: vector $\mathbf{r}$ =NormalForm($\mathbf{s}$, G);
$\mathbf{r} = \mathbf{s}$;
while $\exists \mathbf{g} \in G$ *with* $\mathbf{g} \sqsubseteq \mathbf{r}$ **do**
  $\mathbf{r} = \mathbf{r} - \mathbf{g}$;
end
return $\mathbf{r}$;

Completion procedure

Pottier's algorithm;

Data: a generating set F of $\mathcal{L} \subset \mathbb{Z}^n$

Result: a set $G \subset \mathcal{L}$ containing all the $\sqsubseteq$-minimal elements in $\mathcal{L}\setminus\{0\}$;

$G = F \cup (-F)$;

$C = \cup_{\mathbf{f},\mathbf{g} \in G}\{\mathbf{f} + \mathbf{g}\}$;

while $C \neq \emptyset$ **do**

  $\mathbf{s} =$an element in C;

  $C = C\setminus\{\mathbf{s}\}$;

  $\mathbf{r} =$NormalForm($\mathbf{s}$, G);

  if $\mathbf{r} \neq 0$ **then**

    $C = C \cup \{\mathbf{r} + \mathbf{g} : \mathbf{g} \in G\}$;

    $G = G \cup \{\mathbf{r}\}$;

  end

end

return G;

Drawbacks of Pottier's algorithm

- The set G might contain many elements of $\mathcal{L}$ that are not $\sqsubseteq$-minimal.
- The computation of the normal form of $\mathbf{s}$ with respect to G is very costly

Project-and-lift approach

Best algorithm

- Apply Pottier's algorithm to achieve Graver basis property on a **subset of all variables**. All vectors in $\ker(A)$ (in particular: all Graver bases elements) can be generated by increasing norm on these variables(Project phase).
- Apply Pottier's algorithm again, but to all variables.
  - Fewer sums $f + g$ have to be considered. (f and g should have the same sign pattern on the chosen variables.)
  - Only those sums $f + g$ have to be considered that fulfill upper bound conditions on the chosen variables.

```
sage: from sage.interfaces.four_ti_2 import four_ti_2
sage: four_ti_2.write_matrix([[1,1,1,1],[1,5,10,25]],
 "4coins.mat")
sage: four_ti_2.call("graver", "4coins", False)
sage: four_ti_2.read_matrix("4coins.gra")
[ 5 -6  0  1]
[ 5 -9  4  0]
[ 0  3 -4  1]
[ 5 -3 -4  2]
[ 5  0 -8  3]
```

Let $A \in \mathbb{Z}^{m \times n}$, $\mathbf{b} \in \mathbb{Z}^m$ $\mathbf{l}$, $\mathbf{u} \in \mathbb{Z}^n$ and an objective function $f : \mathbb{R}^n \to \mathbb{R}$ be given.

$$IP_{A,\mathbf{b},\mathbf{l},\mathbf{u},f} : \min\{f(\mathbf{z}) : \; A\mathbf{z} = \mathbf{b}, \; \mathbf{l} \leq \mathbf{z} \leq \mathbf{u}, \; \mathbf{z} \in \mathbb{Z}^n\}$$

As we developed in the linear case, we give a test set for this problem in certain conditions.

Test set for $IP_{A,\mathbf{b},\mathbf{l},\mathbf{u},f}$

$\mathcal{T} \subset \mathbb{Z}^n$ is a **test set** for $IP_{A,\mathbf{b},\mathbf{l},\mathbf{u},f}$ if, for every nonoptimal feasible solution $\mathbf{z}_0$ of $IP_{A,\mathbf{b},\mathbf{l},\mathbf{u},f}$ there exists a vector $\mathbf{t} \in \mathcal{T}$ and some positive integer α such that

- $\mathbf{z}_0 + \alpha\mathbf{t}$ is feasible and
- $f(\mathbf{z}_0 + \alpha\mathbf{t}) < f(\mathbf{z}_0)$

Lemma

Let $f(\mathbf{z}) = \sum_{j=1}^{n} f_j(z_j)$ be separable convex, let $\mathbf{z} \in \mathbb{R}^n$, and $\mathbf{g}_1, \ldots \mathbf{g}_r \in \mathbb{R}^n$ be vectors with the same sign pattern; that is, they belong to a common orthant of $\mathbb{R}^n$. Then we have

$$f\left(\mathbf{z} + \sum_{i=1}^{r} \alpha_i \mathbf{g}_i\right) - f(\mathbf{z}) \geq \sum_{i=1}^{r} \alpha_i (f(\mathbf{z} + \mathbf{g}_i) - f(\mathbf{z}))$$

for arbitrary integers $\alpha_1, \ldots, \alpha_r \in \mathbb{Z}_+$

Lemma

The set $Gr(A)$ is an optimality certificate for $IP_{A,\mathbf{b},\mathbf{l},\mathbf{u},f}$ for any vectors $\mathbf{b} \in \mathbb{Z}^m$ $\mathbf{l}$, $\mathbf{u} \in \mathbb{Z}^n$ and for any separable convex function f

Graver-best augmentation algorithm

Graver-best augmentation algorithm;

Data: $A \in \mathbb{Z}^{m \times n}$, $\mathbf{b} \in \mathbb{Z}^m$ $\mathbf{l}$, $\mathbf{u} \in \mathbb{Z}^n$, $f : \mathbb{R}^n \to \mathbb{R}$, a finite test set $\mathcal{T}$ for $IP_{A,\mathbf{b},\mathbf{l},\mathbf{u},f}$, a feasible solution $\mathbf{z}_0$ to $IP_{A,\mathbf{b},\mathbf{l},\mathbf{u},f}$

Result: a optimal solution $\mathbf{z}_{min}$ of $IP_{A,\mathbf{b},\mathbf{l},\mathbf{u},f}$;

while *There are* $\mathbf{t} \in \mathcal{T}, \alpha \in \mathbb{Z}_+$ *with* $\mathbf{z}_0 + \alpha\mathbf{t}$ *feasible and* $f(\mathbf{z}_0 + \alpha\mathbf{t}) < f(\mathbf{z}_0)$ **do**

    Among all such pairs $\mathbf{t} \in \mathcal{T}$, $\alpha \in \mathbb{Z}_+$ choose one with $f(\mathbf{z}_0 + \alpha\mathbf{t})$ minimal;

    $\mathbf{z}_0 = \mathbf{z}_0 + \alpha\mathbf{t}$;

end

return $\mathbf{z}_0$;