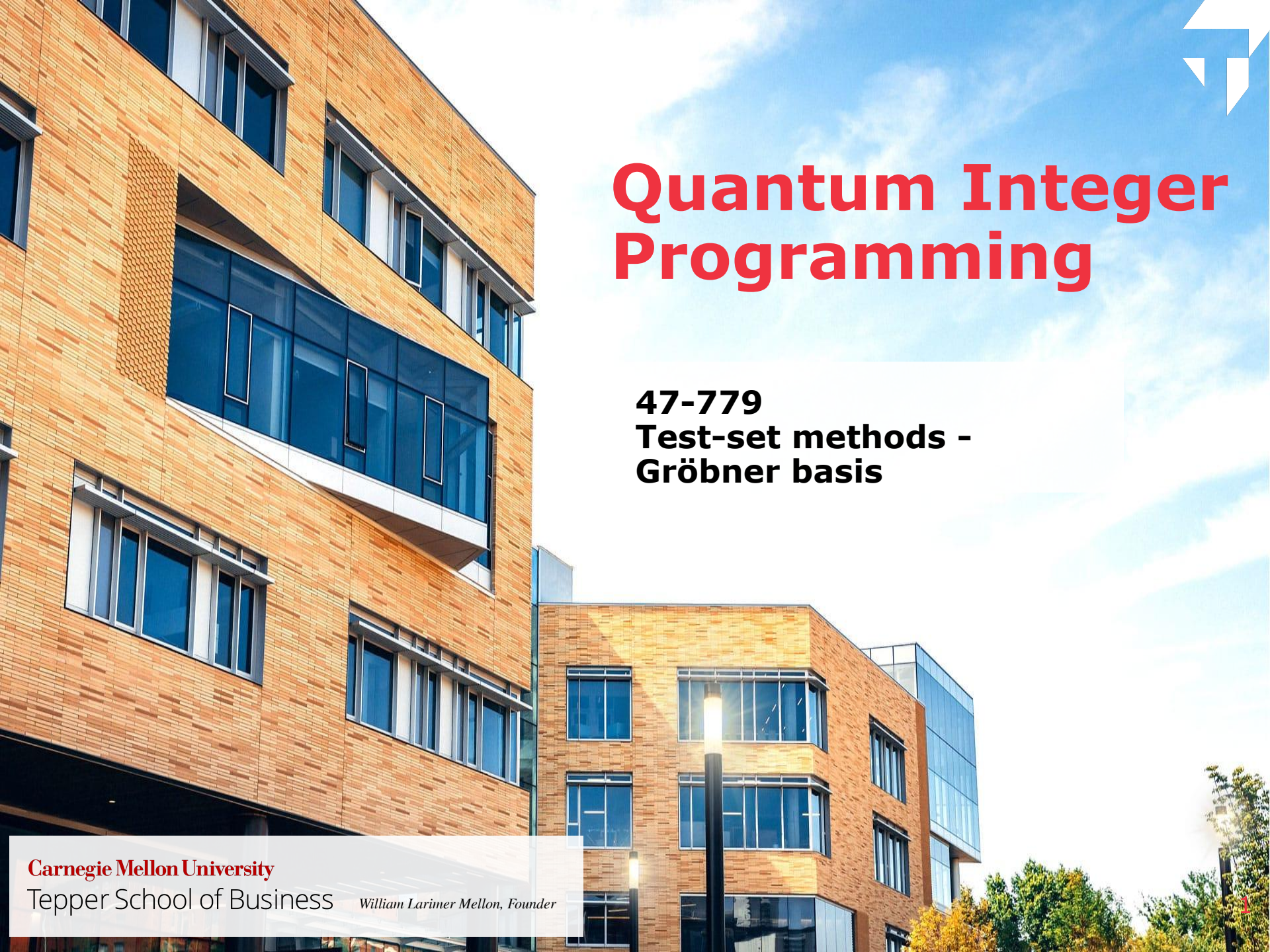


Quantum Integer Programming

47-779
Test-set methods - Gröbner basis

Carnegie Mellon University
Tepper School of Business
William Larimer Mellon, Founder

Quiz 1

What is the optimal objective function value of the following problem?

$$\min_{\mathbf{x}} 2x_0 + 4x_1 + 4x_2 + 4x_3 + 4x_4 + 4x_5 + 5x_6 + 4x_7 + 5x_8 + 6x_9 + 5x_{10}$$

$$s.t. \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\mathbf{x} \in \{0,1\}^{11}$$

$$\min \sum_i c_i x_i$$
$$A\mathbf{x} = b$$
$$\mathbf{x} \in \{0,1\}^n$$

What is the optimal objective function of $\min \sum_i \exp(c_i x_i^2)$?

$$A\mathbf{x} = b$$
$$\mathbf{x} \in \{0,1\}^n$$

Which variable(s) is(are) 1 in the optimal solution of $\min \sum_i \log(c_i + x_i)$?

$$A\mathbf{x} = b$$
$$\mathbf{x} \in \{0,1\}^n$$

Agenda

- Algebraic Geometry introduction
- Gröbner basis and polynomial equations (triangulation)
- Computing Gröbner basis: Buchberger algorithm
- Application in graph theory: coloring
- Algebraic geometry in combinatorial optimization
- Test-set methods
- How to obtain test-sets for IP
  - 5 steps

Acknowledgements

A considerable part of this material is based on Scipy's Gröbner basis tutorials and the Lecture by Maria Isabel Hartillo for IMUS-MSRI2016

What is algebraic geometry?

Algebraic geometry is the study of geometric objects defined by polynomial equations, using algebraic means. Its roots go back to Descartes' introduction of coordinates to describe points in Euclidean space and his idea of describing curves and surfaces by algebraic equations.

Basic correspondence in algebraic geometry

$$\text{Algebraic varieties} \simeq \text{Polynomial rings}$$

Example: Circle

$$\text{Algebraic variety: } \mathcal{V} := \{(x, y) \in \mathbb{R}^2 : x^2 + y^2 - 1 = 0\}.$$

$$\text{Polynomial ring: } \mathbb{Q}[x, y]/ < x^2 + y^2 - 1 > = \text{polynomials mod } x^2 + y^2 - 1$$

Gröbner basis - Example

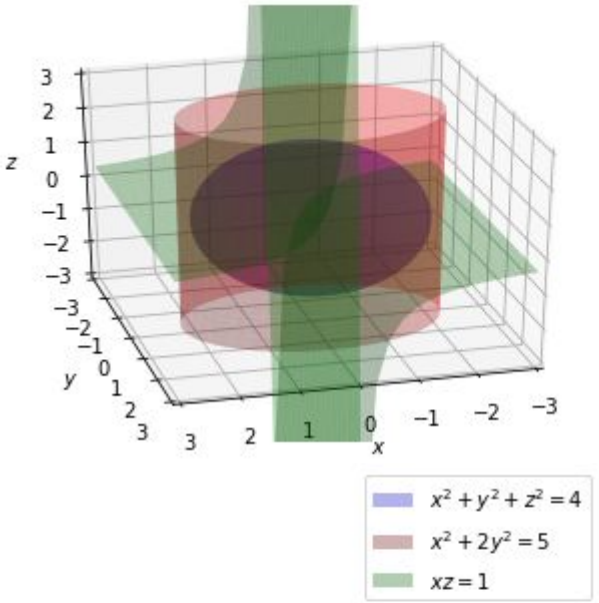
Consider the system of equations

$$\mathcal{S} = \{x^2 + y^2 + z^2 - 4, x^2 + 2y^2 - 5, xz - 1\}$$

We would like to find the solution of this system (variety $\mathcal{V}(\mathcal{S})$)

It would be easier if we could perform an upper triangularization technique:

- Finding polynomial equations such that we can solve one variable at a time



For that. we compute the Gröbner basis for $\mathcal{S}$

Solving this system can be understood geometrically as finding the intersection between a sphere a cylinder and a bilinear curve

[1] https://arxiv.org/abs/1810.01440
[2] Miller, E., & Sturmfels, B. (1999). Monomial Ideals and Planar Graphs. AAECC.

Gröbner basis

Let’s go to the code

https://colab.research.google.com/github/bernalde/QuIP/blob/master/notebooks/Notebook%202%20-%20Groebner%20basis.ipynb

What is algebraic geometry?

Notation

- Let k be a **field** (set where addition, subtraction, multiplication and division work as with the rationals and reals)
- Let $\mathbf{x}^\alpha = x_0^\alpha \cdots x_{n-1}^\alpha$ be a **monomial** in $x_0, \cdots, x_{n-1}$
- Let $c\mathbf{x}^\alpha$ where $c \in k$ be a **term** in $x_0, \cdots, x_{n-1}$
- Let $f = \sum_{\alpha \in \mathbb{Z}_+^n} c_\alpha \mathbf{x}^\alpha$ be a **polynomial** in n variables
- Let $\mathbb{Q}[\mathbf{x}] = \mathbb{Q}[x_0, \cdots, x_{n-1}]$ be the **polynomial ring** in n variables
- Let $\mathbb{A}^n = \mathbb{A}^n(k)$ be the affine space over k

- 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 $\mathbf{x}^{\alpha+\gamma} > \mathbf{x}^{\beta+\gamma}$
  - There exists a smallest element under >
  - Useful to determine **elimination orders**

What is algebraic geometry?

For the purpose of this lecture

- Let $\mathcal{S}$ be a set of polynomials $f \in \mathbb{Q}[x_0, \ldots, x_{n-1}]$
- $\mathcal{V}(\mathcal{S})$ is the affine variety defined by the polynomials $f \in \mathcal{S}$, that is, the set of common zeros of the equations: $\mathcal{V}(\mathcal{S}) = ((z_1, \cdots, z_n) | f_k(z_1, \cdots, z_n) = 0, \forall f_k \in S)$
- The system $\mathcal{S}$ generates an ideal $\mathcal{I}$ by taking all linear combinations over the polynomial ring $\mathbb{Q}[x_0, \ldots, x_{n-1}]$ in $\mathcal{S}$
  - $f_1, f_2 \in \mathcal{I} \rightarrow f_1 + f_2 \in \mathcal{I}$
  - $f \in \mathcal{I}, r \in \mathbb{R} \rightarrow rf \in \mathcal{I}$
  - Property: $\mathcal{V}(\mathcal{S}) = \mathcal{V}(\mathcal{I})$

Every ideal can be finitely generated

$\mathcal{I} = \{\sum_{i=0}^{t-1} h_i g_i | h_i \in \mathbb{Q}[\mathbf{x}]\} \Leftrightarrow \mathcal{I} = \langle g_0, \cdots, g_{t-1} \rangle$

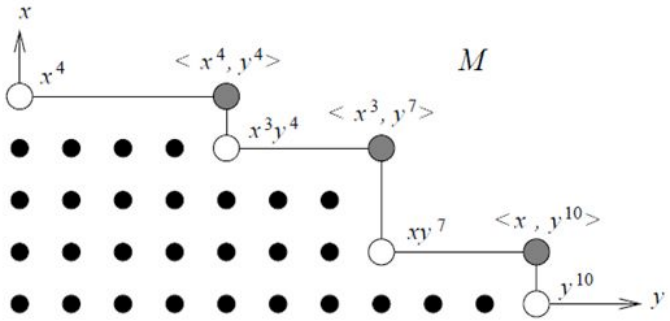
- The ideal $\mathcal{I}$ reveals the hidden polynomials that are consequence of the generating polynomials in
  - For instance is one hidden polynomial is the constant 1 (i.e. $1 \in \mathcal{I}$) then the system $\mathcal{S}$ is inconsistent (since $1 \neq 0$)
- Strictly speaking, the set of all hidden polynomials is given by the radical ideal $\sqrt{\mathcal{I}}$, which is defined by $\sqrt{\mathcal{I}} = \{g \in \mathbb{Q}[x_0, \ldots, x_{n-1}] | \, \exists r \in \mathbb{N} : \, g^r \in \mathcal{I}\}$
  - In practice $\sqrt{\mathcal{I}}$ is infinite, so we represent it with a finite set called the reduced Gröbner basis $\mathcal{B}$

Gröbner basis - Properties

Interesting properties

Properties of the variety $\mathcal{V}(\mathcal{S}) = \mathcal{V}(\mathcal{I})$ can be inferred from the reduced Gröbner basis: $\mathcal{B}$

- In fact $\mathcal{V}(\mathcal{S}) = \mathcal{V}(\mathcal{I}) = \mathcal{V}(\sqrt{\mathcal{I}}) = \mathcal{V}(\mathcal{B})$
- If $\mathcal{B} = \langle \mathcal{I} \rangle$(or if $1 \in \mathcal{B}$) then $\mathcal{V}(\mathcal{I}) = \emptyset$
- The size of $\mathcal{V}(\mathcal{S}) = \mathcal{V}(\mathcal{I})$can be obtained using staircase diagram (without solving equations)



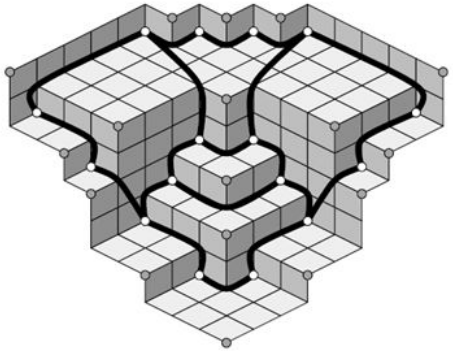


Fig. 1. The monomial ideal $M = \langle x^4, x^3y^4, xy^7, y^{10} \rangle$, with its generators (*white circles*), standard monomials (*black dots*), and irreducible components (*shaded circles*)

[1] https://arxiv.org/abs/1810.01440
[2] Miller, E., & Sturmfels, B. (1999). Monomial Ideals and Planar Graphs. AAECC.

Gröbner basis
How to compute them?

Monomial orders

- Lexicographic order $\mathbf{x}^\alpha >_{lex} \mathbf{x}^\beta$
  - For every $\alpha, \beta \in \mathbb{Z}_+^n$ $\alpha >_{lex} \beta$ if the leftmost nonzero entry of $\alpha - \beta$ is positive.
    $\alpha_1 > \beta_1$, or $\alpha_1 = \beta_1$ and $\alpha_2 > \beta_2$ …
    Example: $x^3 >_{lex} x^2 z^2 >_{lex} xy^2 z >_{lex} z^2$
    Power of x dominates
- Graded lex order $\mathbf{x}^\alpha >_{grlex} \mathbf{x}^\beta$
  - For every $\alpha, \beta \in \mathbb{Z}_+^n$ $\alpha >_{grlex} \beta$ if
    $|\alpha| = \sum_{i=0}^{n-1} \alpha_i > |\beta| = \sum_{i=0}^{n-1} \beta_i$ or $\alpha_1 = \beta_1$ and $\alpha >_{lex} \beta$
    Example: $x^2 z^2 >_{grlex} xy^2 z >_{grlex} x^3 >_{grlex} z^2$
    Total degree dominates, power of x breaks the tie between first 2 polys
- Graded releverse lex order $\mathbf{x}^\alpha >_{grevlex} \mathbf{x}^\beta$
  - Same as graded lex order, but ties are broken in reverse lexicographic order
    Example: $xy^2 z >_{grevlex} x^2 z^2 >_{grevlex} x^3 >_{grevlex} z^2$
    Total degree dominates, power of z breaks the tie between first 2 polys

Gröbner basis
How to compute them?

Leading terms

Given a monomial order > and a set of polynomials $f \in \mathbb{Q}[x_0, \ldots, x_{n-1}]$ we write $f = c_\alpha \mathbf{x}^\alpha +$terms with exponent vectors $\alpha \neq \beta$ such that $c \neq 0$ and $\mathbf{x}^\alpha > \mathbf{x}^\beta$ wherever $\mathbf{x}^\beta$ appears in a nonzero term of f, then

- **Leading term** $LT(f) = c\mathbf{x}^\alpha$
- **Leading monomial** $LM(f) = \mathbf{x}^\alpha$
- **Leading coefficient** $LC(f) = c$

S-Polynomials

Given two multivariate polynomials $f, g \in \mathbb{Q}[\mathbf{x}]$ considering the least common multiple of their leading monomials with respect to an ordering >, $L = \text{lcm}(LM(f), LM(g))$ we define the S polynomial as

$$S(f,g) = \frac{L}{LT(f)} f - \frac{L}{LT(g)} g$$

Normal form or generalized division algorithm

A polynomial $f \in \mathbb{Q}[\mathbf{x}]$ is reduced wrt $G \subset \mathbb{Q}[\mathbf{x}]$ if no monomial of f is contained in the ideal $\langle LM(g) | g \in G \rangle$

Gröbner basis
How to compute them?

Gröbner Basis G

Formal definition $\forall g_i, g_j \in G, \mathrm{mod}\,(S(g_i, g_j), G) = 0$

Buchberger algorithm

Input: A polynomial set $\mathcal{S} = \{f_0, \cdots, f_{n-1}\}$ that generates $\mathcal{I}$

Output: A Gröbner basis $G = \{g_0, \cdots, g_{t-1}\}$ that generates $\mathcal{I}$

$G := \mathcal{S}$

$M := \{\{f_i, f_j\} : f_i, f_j \in G \text{ and } f_i \neq f_j\}$

WHILE $M \neq \emptyset$

$\{p, q\} :=$ a pair in M

$M := M \setminus \{p, q\}$

$S := S(p, q)$

$h := \mathrm{reduced}\,(S, G)$

IF $h \neq 0$

$M := M \cup \{\{g, h\} \forall g \in G\}$

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

[1] Weisstein, Eric W. "Buchberger's Algorithm". MathWorld.
[2] https://www3.risc.jku.at/people/buchberg/

Gröbner basis
How to compute them?

Reduced Gröbner Basis $\mathcal{B}$

A Gröbner basis is reduced if for every $g \in \mathcal{B}$

- $LT(g)$ divides no term of any element of $\mathcal{B} \setminus \{g\}$
- $LC(g) = 1$

Important specializations of Gröbner basis:

- If the system of equations is linear it reduces to Gauss' algorithm
- If the polynomials are univariate it reduces to Euclid's division algorithm

[1] https://www.britannica.com/biography/Carl-Friedrich-Gauss
[2] https://www.britannica.com/biography/Euclid-Greek-mathematician

Gröbner basis

Let’s go back to the code

https://colab.research.google.com/github/bernalde/QuIP/blob/master/notebooks/Notebook%202%20-%20Groebner%20basis.ipynb

Gröbner basis Applications

Applications

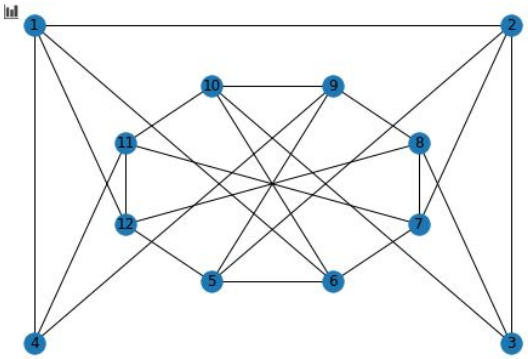
- Algebraic Geometry
- Coding Theory
- Cryptography
- Invariant Theory
- **Integer Programming**
- **Graph Theory**
- Statistics
- Symbolic Integration
- Symbolic Summation
- Differential Equations
- Systems Theory

Example

Graph coloring

$$\mathcal{G}(V, E) =$$

$$K = \{1, \cdots, k\}$$



Can be posed as an Integer Program (actually as a SAT since objective is irrelevant)

$$\min_{\mathbf{x}} 1$$
$$s.t. \sum_{j \in K} x_{ij} = 1, \forall i \in V$$
$$x_{uj} + x_{vj} \leq 1, \forall j \in K, \forall (u, v) \in E$$
$$x_{ij} \in \{0, 1\}, \forall j \in K, \forall i \in V$$

Or rather as a set of polynomial equations

$$\mathcal{S} = \{x_i^k = 1, \forall i \in V$$
$$x_u^k - x_j^k = 0, \forall (u, v) \in E\}$$

[1] Buchberger, Bruno. "Gröbner bases and systems theory." Multidimensional systems and signal processing 12.3-4 (2001): 223-251.

Gröbner basis

Let’s go back to the code

https://colab.research.google.com/github/bernalde/QuIP/blob/master/notebooks/Notebook%202%20-%20Groebner%20basis.ipynb

Algebraic Geometry in Combinatorial Optimization

Given an optimization problem

$$\begin{aligned} &\operatorname{argmin}_{\mathbf{x}} f(\mathbf{x}) \\ &s.t.\ g(\mathbf{x}) = 0 \\ &\mathbf{x} \in \{0,1\}^n \end{aligned}$$

Where $f(\mathbf{x}), g(\mathbf{x}) \in \mathbb{Q}[\mathbf{x}]$ **Algebraic Geometry appears naturally**

The feasible solutions are the variety of the ideal generated by the constraints and a polynomial enforcing the variables being binary $x_i^2 - x_i$

We can generate a new ideal involving the objective function

$$\mathcal{I} = \langle z - f(\mathbf{x}), g(\mathbf{x}), x_i^2 - x_i \rangle \subset \mathbb{Q}[z, x_0, \cdots, x_{n-1}]$$

Proposed by Bertsimas, Perakis, and Tayur (2000). **BPT method** from now on

Example

$$\begin{aligned} &\operatorname{argmin}_{\mathbf{x}} x_1 + 2x_2 + 3x_3 + 4x_4 \\ &s.t.\ x_1 + x_2 + 2x_3 + x_4 = 3 \\ &\mathbf{x} \in \{0,1\}^n \end{aligned}$$

[1] Bertsimas, Dimitris, Georgia Perakis, and Sridhar Tayur. "A new algebraic geometry algorithm for integer programming." Management Science 46.7 (2000): 999-1008.

Gröbner basis

Let’s go back to the code

BPT method

https://colab.research.google.com/github/bernalde/QuIP/blob/master/notebooks/Notebook%202%20-%20Groebner%20basis.ipynb

Algebraic Geometry in Combinatorial Optimization

Given an integer linear program

$$\begin{aligned} &\operatorname{argmin}_{\mathbf{x}} \mathbf{c}^\top \mathbf{x} \\ &s.t.\ \mathbf{A}\mathbf{x} = \mathbf{b} \\ &\mathbf{x} \in \mathbb{Z}^n \end{aligned}$$

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

We introduce for each constraint a variable $z_i^{\mathbf{a}_{i1}x_1 + \cdots + \mathbf{a}_{in}x_n} = z_i^{b_i}$

$$\prod_{i=1}^m \prod_{j=1}^n (z_i^{\mathbf{a}_{ij}x_j}) = \prod_{i=1}^m z_i^{b_i}$$

With this we define the mapping $\phi : \mathbb{Q}[w_1, \cdots, w_n] \to \mathbb{Q}[z_1, \cdots, z_m]$ such that

$$\phi(w_j) = \prod_{i=1}^m (z_i^{\mathbf{a}_{ij}x_j})$$

then for $g \in \mathbb{Q}[\mathbf{w}]$

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

Proposed by Conti and Traverso (1991). **CT method**

[1]Conti, Pasqualina, and Carlo Traverso. "Buchberger algorithm and integer programming." International Symposium on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes. Springer, Berlin, Heidelberg, 1991.

Algebraic Geometry in Combinatorial Optimization

Example

$$\begin{aligned} 4x_1 \quad &+5x_2 + x_3 \quad &&=37 \\ 2x_1 \quad &+3x_2 \quad &+x_4 \;&=20 \end{aligned}$$

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

Where a set of feasible solutions satisfies

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

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

$$\mathcal{I} = \langle f_1 - w_1, \cdots, f_n - w_n \rangle \subset \mathbb{Q}[\mathbf{z}, \mathbf{w}]$$

[1]Conti, Pasqualina, and Carlo Traverso. "Buchberger algorithm and integer programming." International Symposium on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes. Springer, Berlin, Heidelberg, 1991.

Algebraic Geometry in Combinatorial Optimization

This method is also able to handle negative coefficients in the constraints

Example

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

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

Where a set of feasible solutions satisfies

$$\phi(w_1^{x_1} w_2^{x_2} w_3^{x_3}) = z_1^4 z_2^5$$

Let the ideal be $\mathcal{I} = \langle w_1 z_2 - z_1^2, w_2 z_1 - z_2^2, w_3 - z_1 \rangle$

We can strengthen it with an extra equation to uses an extra variable t to allow for invertibility (not always necessary)

$$\mathcal{I} = \langle w_1 z_2 - z_1^2, w_2 z_1 - z_2^2, w_3 - z_1, 1 - t z_1 z_2 \rangle$$

[1]Conti, Pasqualina, and Carlo Traverso. "Buchberger algorithm and integer programming." International Symposium on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes. Springer, Berlin, Heidelberg, 1991.

Gröbner basis

Let’s go back to the code

CT method

https://colab.research.google.com/github/bernalde/QuIP/blob/master/notebooks/Notebook%202%20-%20Groebner%20basis.ipynb

Algebraic Geometry in Combinatorial Optimization

Given an integer linear program

$$\begin{aligned} &\operatorname{argmin}_{\mathbf{x}} \mathbf{c}^\top \mathbf{x} \\ &s.t.\ \mathbf{A}\mathbf{x} = \mathbf{b} \\ &\mathbf{x} \in \mathbb{Z}^n \end{aligned}$$

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

We introduce for each constraint a variable $z_i^{\mathbf{a}_{i1}x_1 + \cdots + \mathbf{a}_{in}x_n} = z_i^{b_i}$

$$\prod_{i=1}^m \prod_{j=1}^n (z_i^{\mathbf{a}_{ij}x_j}) = \prod_{i=1}^m z_i^{b_i}$$

With this we define the mapping $\phi : \mathbb{Q}[w_1, \cdots, w_n] \to \mathbb{Q}[z_1, \cdots, z_m, z_1^{-1}, \cdots, z_m^{-1}]$ such that

$$\phi(w_j) = \prod_{i=1}^m (z_i^{\mathbf{a}_{ij}x_j})$$

Each column can be written as $\mathbf{a}_j = \mathbf{a}_j^+ - \mathbf{a}_j^-$ with $\mathbf{a}_j^+, \mathbf{a}_j^- \geq \mathbf{0}$

Then the Ideal becomes $\mathcal{I} = \langle \mathbf{z}^{\mathbf{a}_j^-} w_j - \mathbf{z}^{\mathbf{a}_j^+}, 1 - z_1 \cdots z_m \rangle$

Proposed by Conti and Traverso (1991).

[1]Conti, Pasqualina, and Carlo Traverso. "Buchberger algorithm and integer programming." International Symposium on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes. Springer, Berlin, Heidelberg, 1991.

Improvements for Combinatorial Optimization

Independence from the right hand side

Considering $\mathcal{I} = \langle \mathbf{z}^{\mathbf{a}_j^-} w_j - \mathbf{z}^{\mathbf{a}_j^+}, 1 - z_1 \cdots z_m \rangle = \langle \mathbf{x}^{\mathbf{u}} - \mathbf{x}^{\mathbf{v}} : \mathbf{A}\mathbf{u} = \mathbf{A}\mathbf{v}, \mathbf{u}, \mathbf{v} \in \mathbb{Z}_+^n \rangle$

With a given Gröbner basis with respect to a monomial order that eliminates $\mathbf{z}, t$ we have that $\mathcal{B} \cap \mathbb{Q}[\mathbf{w}]$ is a Gröbner basis of the ideal $\mathcal{I} \cap \mathbb{Q}[\mathbf{w}] = \mathcal{I}_{\mathbf{A}}$

Given $g(\mathbf{w}) \in \mathcal{I} \cap \mathbb{Q}[\mathbf{w}] \Rightarrow g(\mathbf{w}) \in \ker(\phi) = \mathcal{I}_{\mathbf{A}}$

This is called the **toric ideal** of $\mathbf{A}$ and is independent from the right hand side.

Objective function

A monomial order $>_{\mathbf{c}}$ can be defined wrt the objective $\mathbf{c} \in \mathbb{R}_+^n$ such that

$$\alpha >_{\mathbf{c}} \beta \Rightarrow \mathbf{c}^\top \alpha > \mathbf{c}^\top \beta \text{ or}$$
$$\mathbf{c}^\top \alpha = \mathbf{c}^\top \beta \text{ and } \alpha > \beta$$

Theorem

Let $\mathcal{B}_{>_{\mathbf{c}}}$ be the reduced Gröbner basis with respect to $>_{\mathbf{c}}$ of the ideal $\mathcal{I}_{\mathbf{A}}$, then for any right hand side, integer constraint matrix, term order, and nonoptimal feasible solution $\mathbf{z}_0$ there is some such that is a better feasible solution $\mathbf{z}_1 = \mathbf{z}_0 - \mathbf{u} + \mathbf{v}$

Improvements for Combinatorial Optimization

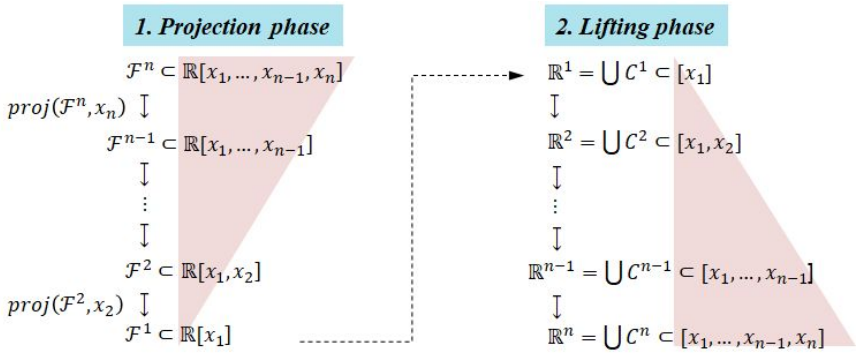
Improvements on the Buchberger algorithm

Although extremely general, the main complication is that the Buchberger algorithm has a double exponential time complexity d^{2^n} with d the degree of the polynomial and n the number of polynomials.

The current best algorithms are developed by Charles Faugere (F4 and F5) and are based on a transformation into linear algebraic problems, where more efficient algorithms can be applied.

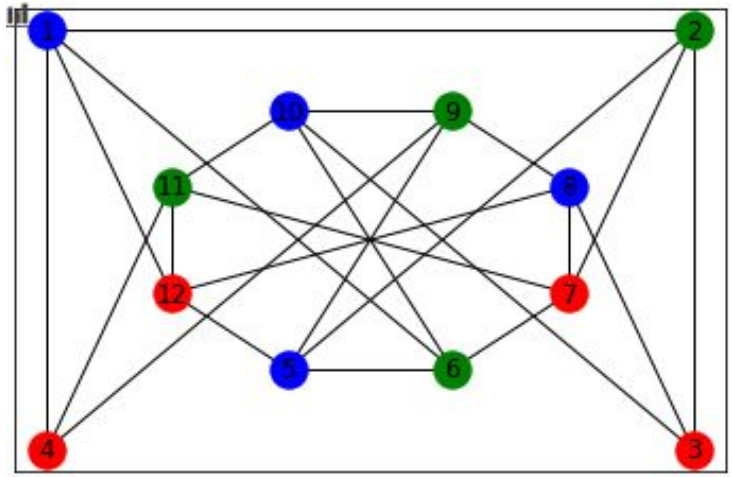
Project-and-lift algorithms

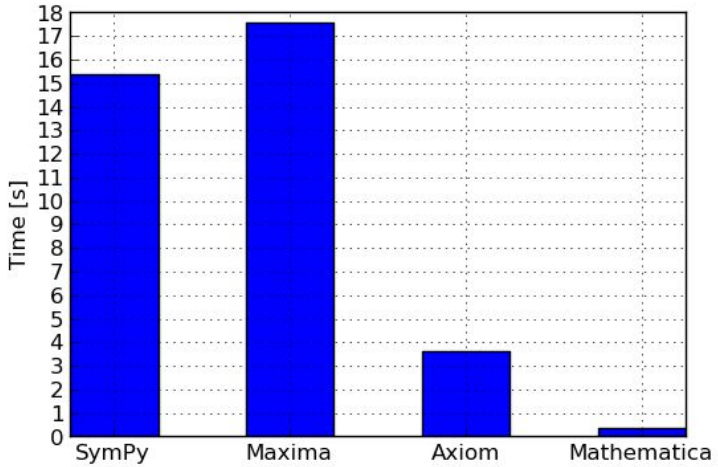
For computing Gröbner basis of Toric ideals the most efficient algorithms try to avoid the the zero reduction of the S-polynomial. The most efficient implementation of this algorithm is available in the software [4ti2](4ti2).



Computing Gröbner basis

Coloring example





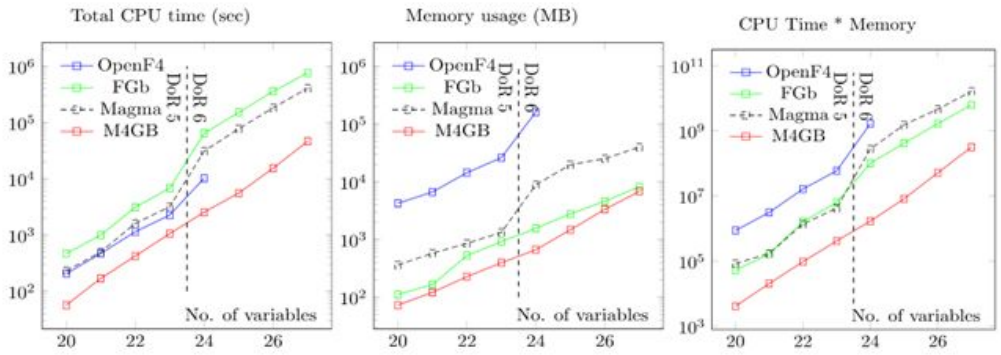
(1) 3-coloring of Graph G(V,E), (3) Average timing for computing Gröbner basis of graph G(V, E) in different software packages

[1] https://mattpap.github.io/masters-thesis/html/src/Gröbner.html#

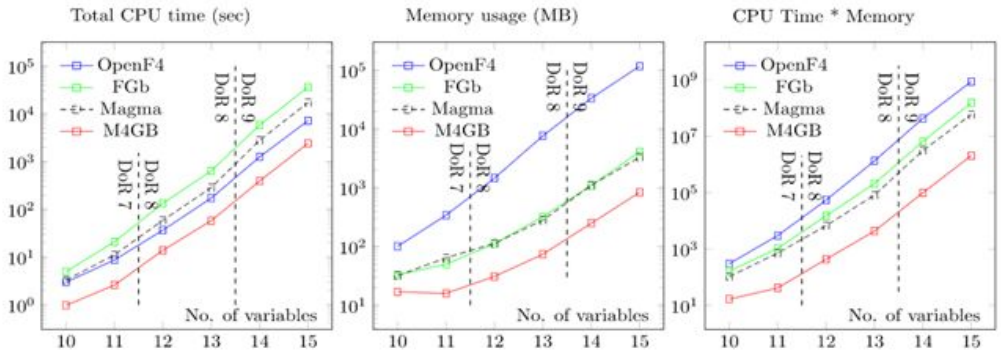
Computing Gröbner basis

General Benchmarks

Benchmarks of M4GB, Magma, FGBLib & OpenF4 over random dense quadratic polynomial systems over GF(31) with *#equations = 2 * #variables*:



Benchmarks of M4GB, Magma, FGBLib & OpenF4 over random dense quadratic polynomial systems over GF(31) with *#equations = #variables + 1*:



Gröbner basis computation with specific field size 31 (GF31) comparing various efficient implementations of algorithms F4 and F5.
Software compared:

- OpenF4
- FGb
- Magma
- M4GB

[1] Rusydi Makarim, Marc Stevens, M4GB: An efficient Gröbner-basis algorithm, ISSAC 2017
https://marc-stevens.nl/research/papers/ISSAC17-MS-M4GB.pdf

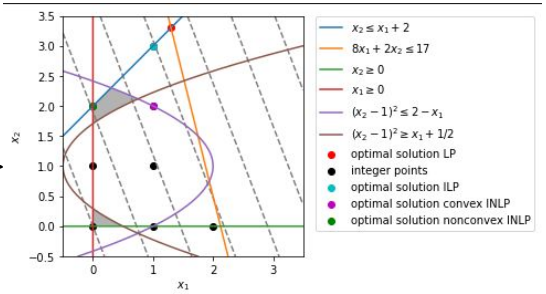
Solving more general problems

How to deal with complicating constraints?

Original problem

$$\begin{aligned}&\operatorname{argmin}_{\mathbf{x}} \mathbf{c}^\top \mathbf{x}\\ &s.t.\ \mathbf{A}\mathbf{x} = \mathbf{b}\\ &g(\mathbf{x}) \leq 0\\ &\mathbf{x} \in \mathbb{Z}^n\end{aligned}$$

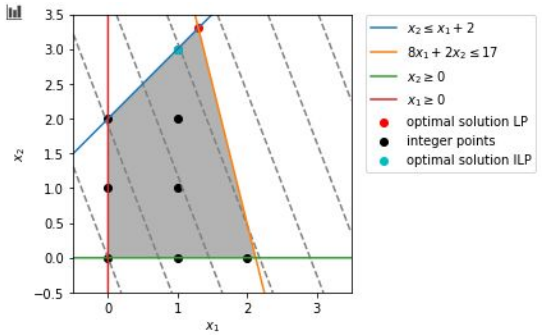
$\Leftrightarrow$



Linear relaxation

$$\begin{aligned}&\operatorname{argmin}_{\mathbf{x}} \mathbf{c}^\top \mathbf{x}\\ &s.t.\ \mathbf{A}\mathbf{x} = \mathbf{b}\\ &\mathbf{x} \in \mathbb{Z}^n\end{aligned}$$

$\Leftrightarrow$



Solve linear relaxation and **walk-back** from that solution to check if it works for original problem.

How to walk-back?

You need to make sure that you stay feasible for relaxation, such that you can eventually reach optimal solution of original problem.

Gröbner basis!

Proposed by Tayur, Thomas, and Natraj (1995)

[1] Tayur, Sridhar R., Rekha R. Thomas, and N. R. Natraj. "An algebraic geometry algorithm for scheduling in presence of setups and correlated demands." Mathematical Programming 69.1-3 (1995): 369-401.

Solving more general problems

How to deal with complicating constraints?

Proposed by Tayur, Thomas, and Natraj (1995)

In original paper the complicating constraints were stochastic and nonlinear constraints, but procedure works in general.

Test-set

Given an integer linear program $\min_{\mathbf{x}} \mathbf{c}^\top \mathbf{x}$ s.t. $\mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \in \mathbb{Z}^n$ there exists a finite set denotes test-set $\mathcal{T} = \{\mathbf{t}^1, \cdots, \mathbf{t}^N\}$ that only depends on $\mathbf{A}$, $\mathbf{c}$ that assures that a feasible solution $\mathbf{x}^*$ is optimal if and only if $\mathbf{c}^\top(\mathbf{x}^* + \mathbf{t}^i) \geq \mathbf{c}^\top(\mathbf{x}^*)$ whenever $(\mathbf{x}^* + \mathbf{t}^i)$ is feasible.

Clever discovery

The Gröbner basis of the toric ideal $\mathcal{I}_{\mathbf{A}}$ with respect to the weighted order for $\mathbf{c}$ $\mathcal{B}_{>_\mathbf{c}}$ are **test-sets**.

[1] Tayur, Sridhar R., Rekha R. Thomas, and N. R. Natraj. "An algebraic geometry algorithm for scheduling in presence of setups and correlated demands." Mathematical Programming 69.1-3 (1995): 369-401.

Solution methods for Combinatorial Optimization

Current status and perspectives

Classical methods

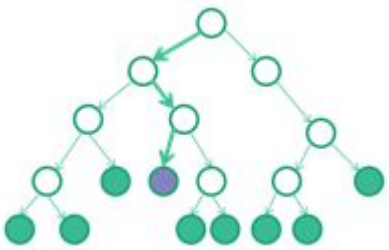
Methods based on divide-and-conquer

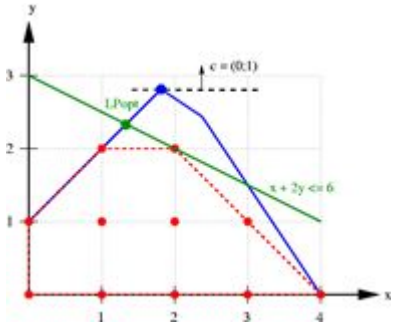
- Branch-and-Bound algorithms
- Harness advances in polyhedral theory
- With global optimality guarantees
- Very efficient codes available
- Exponential complexity

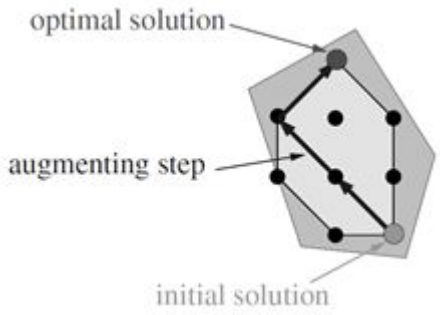
Not very popular classical methods

Methods based on test-sets

- Algorithms based on “augmentation”
- Use tools from algebraic geometry
- Global convergence guarantees
- Very few implementations out there
- Polynomial **oracle** complexity **once we have test-set**







[1] https://de.wikipedia.org/wiki/Branch-and-Cut
[2] Algebraic And geometric ideas in the theory of discrete optimization. De Loera, Hemmecke, Köppe. 2012

Test-set methods - Example

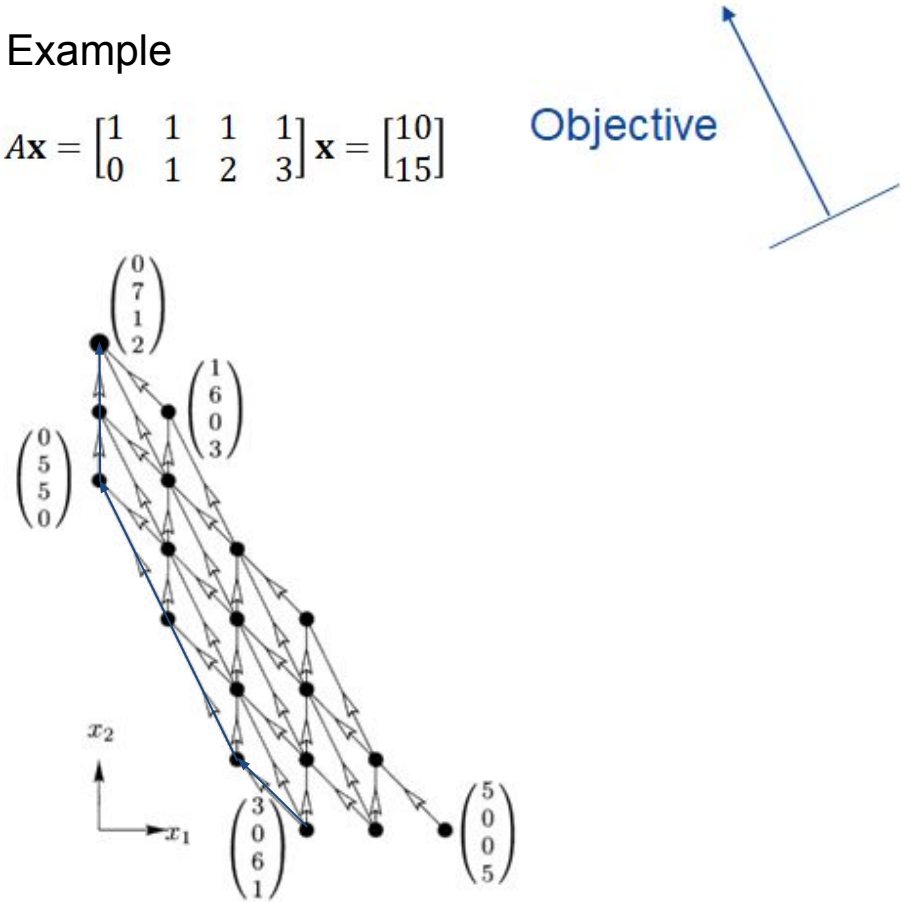
Primal method for Integer Programs

We require:

- An initial feasible solution
- An oracle to compare objective function
- The test-set (set of directions)
- Given the objective, the test set will point us a direction where to improve it, and if no improvement, we have the optimal solution.
- The Gröbner basis test-set only depends on the constraints and objective and can be computed for equality constraints with integer variables

Example

$$A\mathbf{x} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 10 \\ 15 \end{bmatrix}$$



[1] Gröbner Bases and Integer Programming, G. Ziegler. 1997

[2] Integer Programming (1st ed. 2014) by Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli

William Larimer Mellon, Founder

Test-set methods

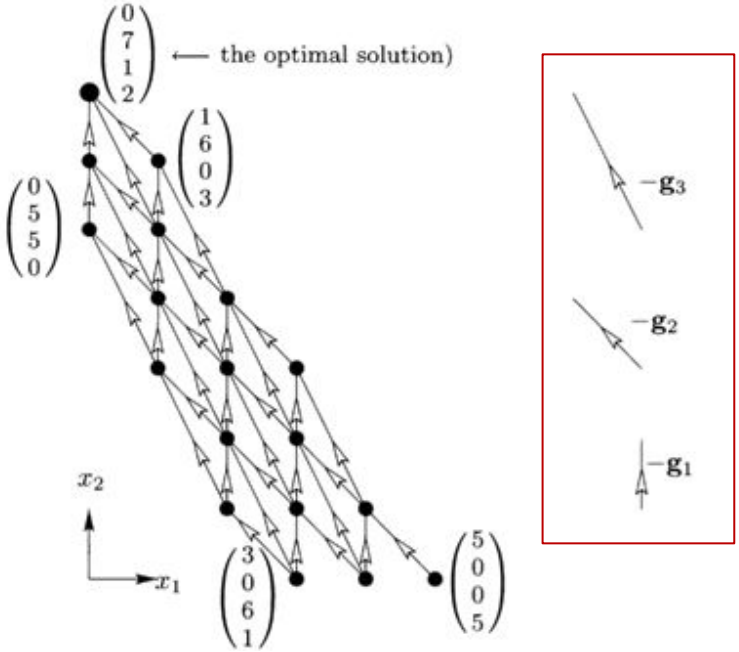
Combinatorial problems are usually NP
Unless P=NP, they don't accept polynomial algorithms.

Where is the NP?

Obtaining the test-set!

Example

$$A\mathbf{x} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 10 \\ 15 \end{bmatrix}$$



[1] Gröbner Bases and Integer Programming, G. Ziegler. 1997

Walk-back procedure

How to deal with complicating constraints?

Starting from the optimal solution of the linear relaxation and a feasible solution of the original problem, compute Gröbner basis and walk back from solution of relaxation using complicating constraints as oracles

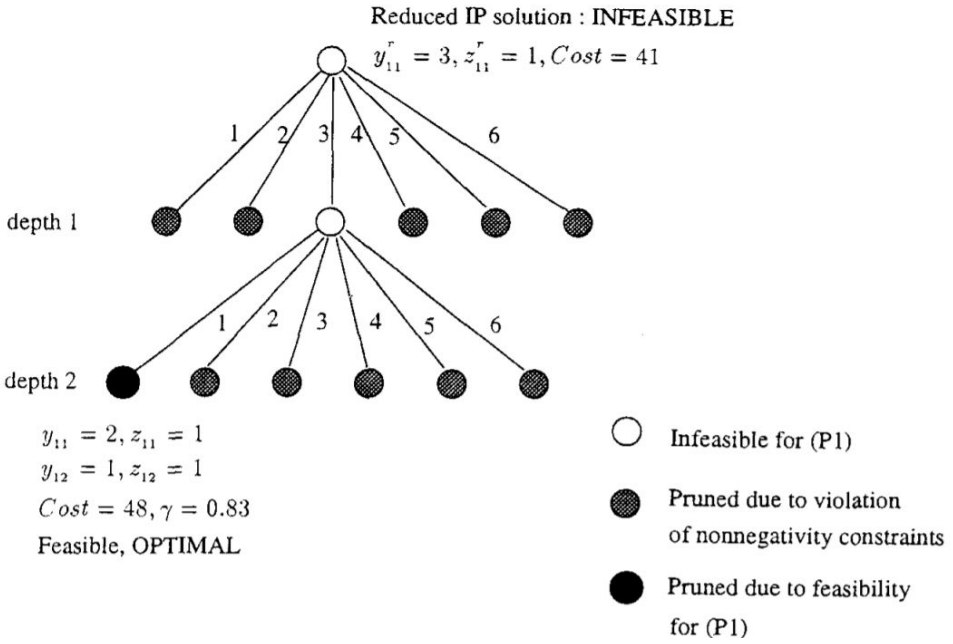
$$\text{minimize} \quad 5z_{11} + 3z_{12} + 12y_{11} + 16y_{12}$$

such that:

$$y_{11} + y_{12} = 3$$
$$y_{11} - 3z_{11} \leqslant 0$$
$$y_{12} - 3z_{12} \leqslant 0$$
$$4y_{12} + z_{11} \leqslant 10$$
$$4y_{12} + z_{12} \leqslant 10$$
$$\text{Prob}\{D_1 \tfrac{1}{3} y_{11} + z_{11} \leqslant 10,\ D_1 \tfrac{1}{3} y_{12} + z_{12} \leqslant 10\} \geqslant 0.6$$
$$z_{ij} \in \{0, 1\}, \quad y_{ij} \in \{0, 1, 2, 3\}.$$

$\mathcal{B}_{>_c}$

(1) $y_{12}a_{11} - y_{11}a_{12}$,
(2) $y_{12}^2 b_{11} - z_{11} y_{11}^2 a_{11} a_{12}^2$,
(3) $z_{12}a_{12}^3 - b_{12}$,
(4) $z_{12} y_{12} a_{12}^2 b_{11} - z_{11} y_{11} a_{11}^2 b_{12}$,
(5) $z_{11}a_{11}^3 - b_{11}$, and
(6) $z_{11} y_{11} a_{11}^2 a_{12} - y_{12} b_{11}$.



[1] Tayur, Sridhar R., Rekha R. Thomas, and N. R. Natraj. "An algebraic geometry algorithm for scheduling in presence of setups and correlated demands." Mathematical Programming 69.1-3 (1995): 369-401.

Walk-back procedure

Application from machine scheduling

Similar formulation as previously but with some strengthening constraints and walking improvements. Gröbner basis calculation done in < 1s with 4ti2.

Table 2 4 machines, 7 jobs, $M = 2$, improved model, 250 records, Time in seconds

	Walk-back				WB new ordering				Gurobi		Mosek		Cbc	
	Total		Optimum		Total		Optimum							
γ	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Total	Optimum	Total	Optimum	Total	Optimum
0.696	0.3	125	0.1	121	0.2	63	0.1	44	0.7	0.1	0.8	0.7	10.3	7.5
0.710	0.3	125	0.1	121	0.2	63	0.1	44	0.9	0.1	0.9	0.9	16.0	8.6
0.750	0.4	190	0.2	179	0.3	99	0.1	61	1.6	1.0	2.7	2.6	9.5	8.0
0.770	0.5	226	0.3	215	0.3	99	0.1	61	1.1	1.0	1.6	1.5	13.6	7.0
0.850	1.1	396	0.4	311	1.1	278	0.3	130	7.7	6.0	Max	117.2	41.9	23.3
0.888	3.4	934	1.3	873	2.8	643	0.2	100	5.1	4.0	Max	337.4	31.7	31.3
0.900	3.4	934	1.3	873	2.7	643	0.2	100	17.4	17.0	Max	265.6	18.1	17.0
0.932	5.4	1299	1.4	893	4.1	913	0.1	63	12.9	2.0	48.1	3.7	17.8	12.3
0.956	176.9	15,220	130.2	14,670	64.8	8511	2.1	1145	12.7	1.0	Max	2.7	48.8	41.5
0.960	534.9	28,575	429.3	27,799	263.7	19,072	62.5	7973	41.3	34.0	Max	294.0	241.4	229.4
0.980		Max		NNP	6355.7	98,294	11.7	3360	39.6	16.0	Max	NNP	75.7	38.8

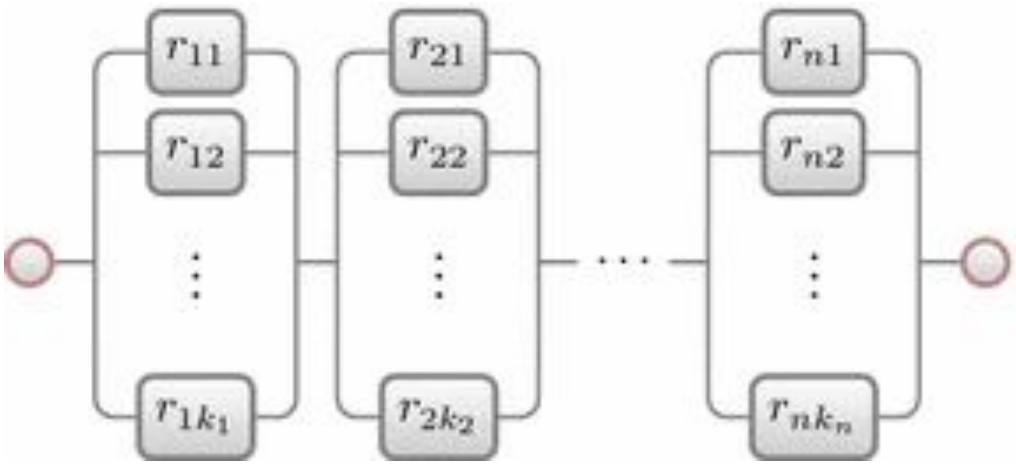
MINLP solvers failed. Chance constraint can be reformulated and solved with ILP solvers. For verification of optimal solution is still competitive.

[1] Gago-Vargas, Jesús, et al. "An improved test set approach to nonlinear integer problems with applications to engineering design." Computational Optimization and Applications 62.2 (2015): 565-588.

Example: Process reliability

How many backup units should you install in order to ensure reliability of your problem?

You want to minimize your operation cost while making sure you satisfy a reliability threshold R_0



$$\min_x \sum_{i,j} c_{ij} x_{ij}$$

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

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

$$x_{ij} \in \mathbb{Z}$$

$$\prod_{i=1}^{n} (1 - \prod_{j=1}^{k_j} (1 - r_{ij})^{x_{ij}}) = R(x) \geq R_0$$

This is a convex integer nonlinear program

[1] Gago-Vargas, Jesús, et al. "An improved test set approach to nonlinear integer problems with applications to engineering design." Computational Optimization and Applications 62.2 (2015): 565-588.

Example: Process reliability

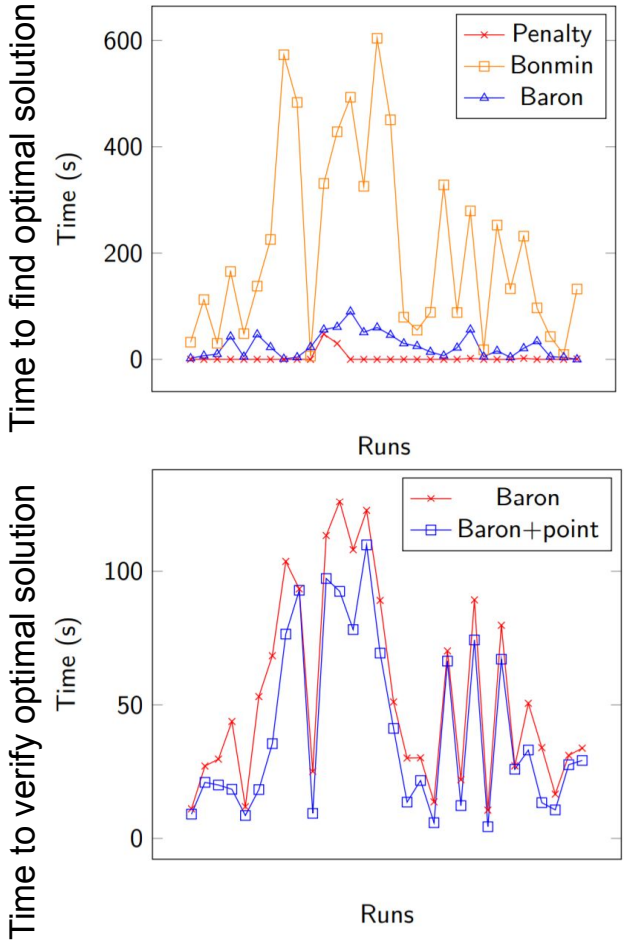
Linear relaxation to find Gröbner basis

$$\min_x \sum_{i=1}^{n} \sum_{j=1}^{k_i} c_{ij} x_{ij}$$
$$x_{ij} + t_{ij} = u_{ij}, \forall i \in I$$
$$\sum_{j=1}^{k_i} x_{ij} - d_{ij} = 1, \forall i \in I, \forall j \in J$$
$$\sum_{i=1}^{n} \sum_{j=1}^{k_i} c_{ij} x_{ij} = b = c_0$$
$$x_{ij} \in \mathbb{Z}, \forall i \in I, \forall j \in J$$

For this simple problem you can even compute analytical Gröbner basis

$$\mathcal{B}_{<_c} = \{\underline{x_{ik} d_i} - t_{ik} b^{c_{ik}}, \underline{x_{iq} t_{ip}} - x_{ip} t_{iq} b^{c_{iq} - c_{ip}}\}$$

The authors also proposed a way to inform the walk-back procedure of the objective function.



[1] Gago-Vargas, Jesús, et al. "An improved test set approach to nonlinear integer problems with applications to engineering design." Computational Optimization and Applications 62.2 (2015): 565-588.

Complexity introduction

When analyzing an algorithm we are interested in the amount of time and memory that it will take to solve it.

- To be safe this is usually answered in the **worst case scenario**
- The discussion here will be mainly the time complexity of algorithms.
- We will use the "big-O" notation where given two functions $f : S \to \mathbb{R}_+$ $g : S \to \mathbb{R}_+$, where S is an unbounded subset of $\mathbb{R}_+$we write that if there exists a positive real number $M$and $x_0 \in S$ such that $f(x) \leq Mg(x)$ for every $x > x_0$ then $f(x) = O(g(x))$
- For us S is going to be the set of instances or problems, and an *algorithm* is a procedure that will give a correct answer is a finite amount of time.
- An algorithm solves a problem in *polynomial time* if the function that measures its arithmetic operations $f : S \to \mathbb{R}_+$ is polynomially bounded by the function encoding the size of the problem $g : S \to \mathbb{R}_+$

Complexity introduction

- If there exists an algorithm that solves a problem in polynomial time, the that problem becomes to the complexity class P
  - For example LP belongs to P because the interior point algorithm solves it in poly-time

- A decision problem is one with answer “yes” or “no”
- The complexity class NP, non-deterministic polynomial, is the class of all decision problems where the “yes”-answer can be verified in poly-time.

- If all the decision problems in NP can be reduced in poly-time to a problem *Q*, then *Q* is said to be NP-complete
  - “Is a (mixed-)integer linear set empty?” belongs to NP and is actually NP-complete

[1] Adapted from Integer Programming (1st ed. 2014) by Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli
[2] Cook, Stephen A. The Complexity of Theorem-Proving Procedures (1971)

Complexity introduction

- A problem *Q* can be called NP-hard if all problems in NP can be reduced to *Q* in poly-time
  - Integer Programming is NP-hard
    - Since we can transform any NP problem into an integer program in poly-time, if there existed an algorithm to solve IP in poly-time, then we could solve any NP problem in poly-time: P=NP

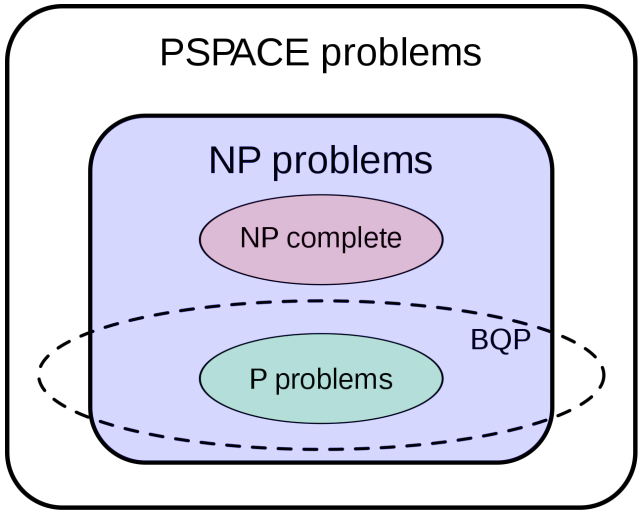
- Integer programs with quadratic constraints are proved to be undecidable
  - Even after a long time without finding a solution, we cannot conclude none exists…
  - MINLP are **tough**!

[1] Adapted from Integer Programming (1st ed. 2014) by Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli
[2] Cook, Stephen A, The Complexity of Theorem-Proving Procedures (1971)

William Larimer Mellon, Founder

Complexity introduction

- A problem is said to belong to the complexity class BPP, bounded-error probabilistic polynomial time, if there is an algorithm that solves it such that
  - It is allowed to flip coins and make random decisions
  - It is guaranteed to run in polynomial time
  - On any given run of the algorithm, it has a probability of at most 1/3 of giving the wrong answer, whether the answer is YES or NO.
- There exists another complexity class called BQP, bounded-error quantum polynomial time, which is the quantum analogue of BPP
  - We hope that some problems belong to BQP and not to BPP to observe Quantum Advantage
    - E.g. Integer factorization



The suspected relationship of BQP to other problem spaces

[1] Michael Nielsen and Isaac Chuang (2000). Quantum Computation and Quantum Information. Cambridge: Cambridge University Press. ISBN 0-521-63503-9.

Take home message

Integer Programs lie at a very special intersection since they are:

- Interesting from an academic point of view
- Useful from a practical point of view
- Challenging from a computational point of view

We do not expect to observe Quantum Advantage by solving Integer Programs using Quantum Computers (but who knows right? Maybe P=BPP=BQP=NP)

We are still dealing with complicated problems that require answers, so we are going to try our best to solve them.

Welcome to Quantum Integer Programming!

More thoughts

Slide taken from Ed Rothberg (key developer of CPLEX and the RO in guRObi) on a talk of parallelization for (M)IP