New Algebraic Techniques in Integer Optimization

Jesús A. De Loera, UC Davis

all new results are based on papers joint work with subsets of Hemmecke, Köppe, Onn, Weismantel

August 31, 2008

Le Menu

- Appetizer: What is Integer Optimization? and why do we need new mathematics to deal with it?
- Main Dish: A Taste of Two New Techniques.
  - Graver bases for Integer Linear Programming
  - Rational Functions for Non-linear Mixed Integer Programming
- Dessert: Closing Comments and Future directions.

Le Menu

- Appetizer: What is Integer Optimization? and why do we need new mathematics to deal with it?
- Main Dish: A Taste of Two New Techniques.
  - Graver bases for Integer Linear Programming
  - Rational Functions for Non-linear Mixed Integer Programming
- Dessert: Closing Comments and Future directions.

Le Menu

- Appetizer: What is Integer Optimization? and why do we need new mathematics to deal with it?
- Main Dish: A Taste of Two New Techniques.
  - Graver bases for Integer Linear Programming
  - Rational Functions for Non-linear Mixed Integer Programming
- Dessert: Closing Comments and Future directions.

What is Integer Optimization?

Why we need new tools?

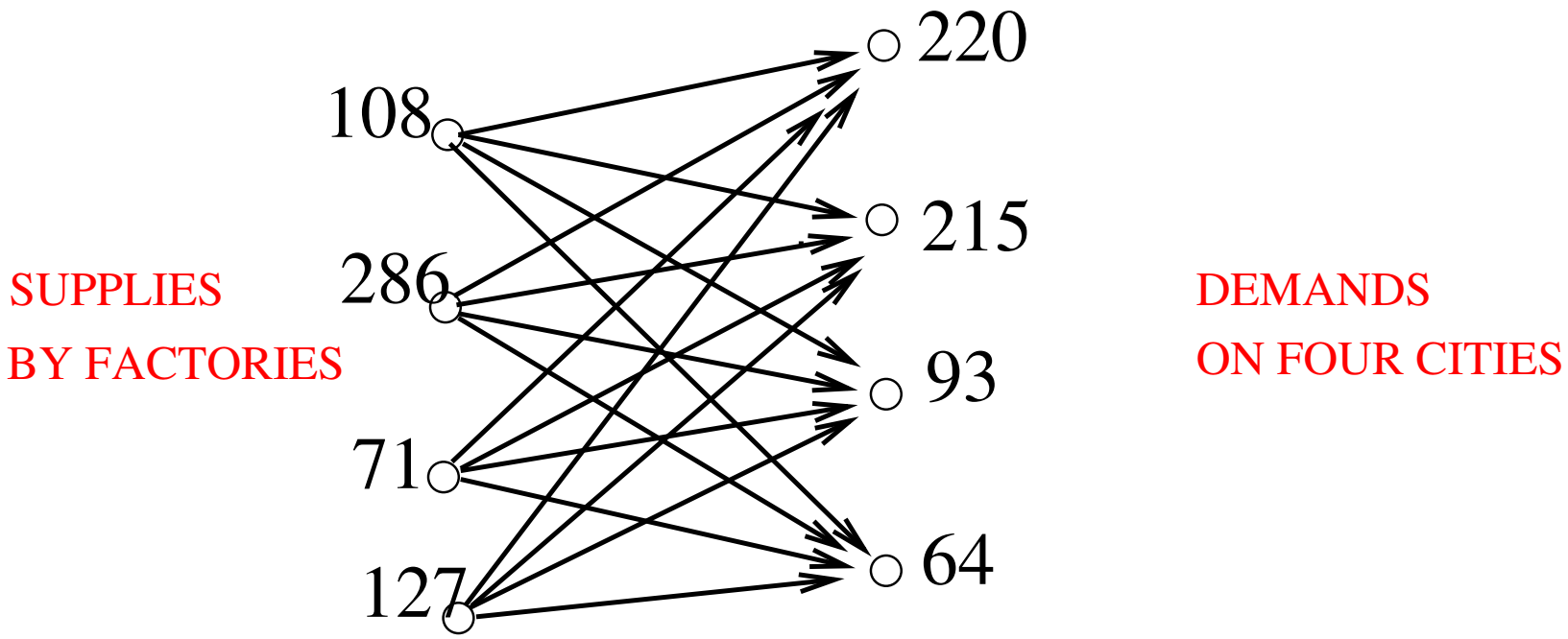
- A part of Applied Mathematics, its main problem: Given a finite set X, each of whose elements has an assigned cost, price or optimality criteria, find the cheapest such object.
- Problems come from bioinformatics, industrial engineering, management, operations planning, finances, any area where the best solution is required!
- History starts with the WWII Initial work by Kantorovich (1939), T.C Koopmans (1941), von Neumann (1947), Dantzig (1950), Ford and Fulkerson (1956). Invention of linear programming and the simplex method.

- A part of Applied Mathematics, its main problem: Given a finite set X, each of whose elements has an assigned cost, price or optimality criteria, find the cheapest such object.
- Problems come from bioinformatics, industrial engineering, management, operations planning, finances, any area where the best solution is required!
- History starts with the WWII Initial work by Kantorovich (1939), T.C Koopmans (1941), von Neumann (1947), Dantzig (1950), Ford and Fulkerson (1956). Invention of linear programming and the simplex method.

- A part of Applied Mathematics, its main problem: Given a finite set X, each of whose elements has an assigned cost, price or optimality criteria, find the cheapest such object.
- Problems come from bioinformatics, industrial engineering, management, operations planning, finances, any area where the best solution is required!
- History starts with the WWII Initial work by Kantorovich (1939), T.C Koopmans (1941), von Neumann (1947), Dantzig (1950), Ford and Fulkerson (1956). Invention of linear programming and the simplex method.

My Favorite Example

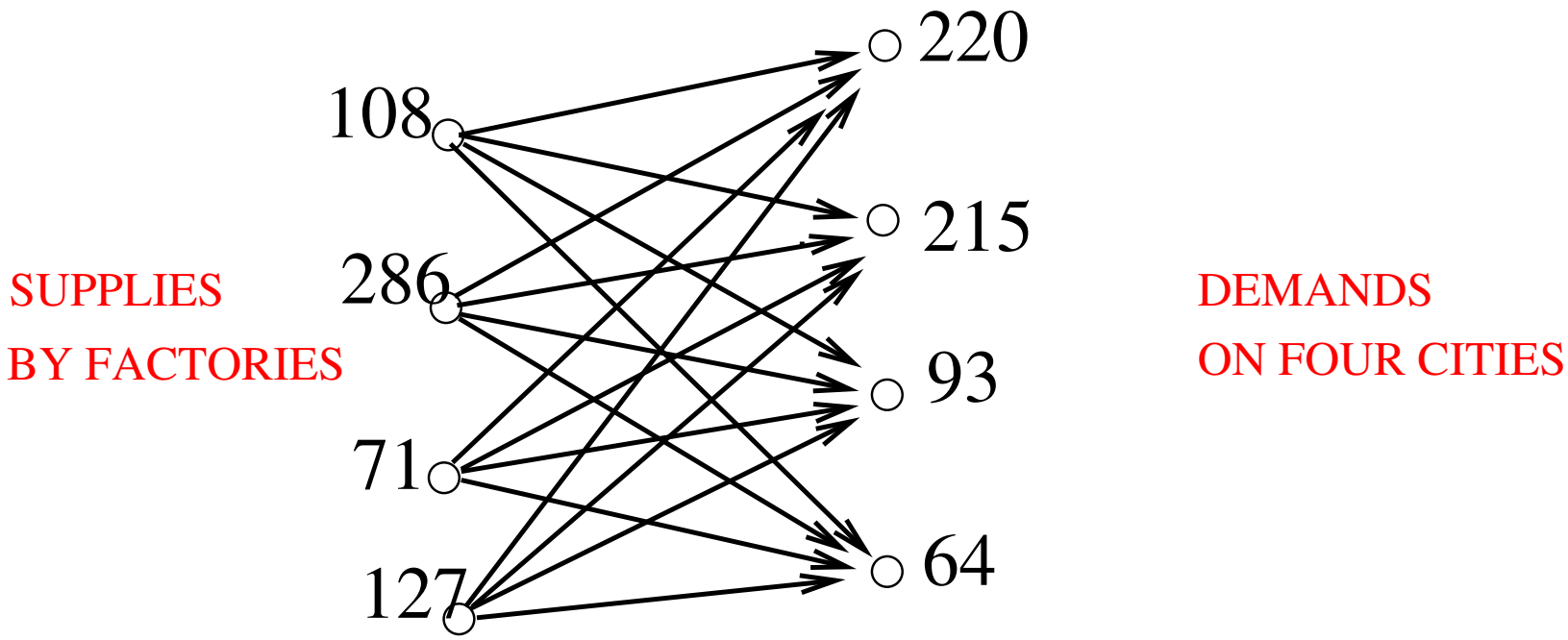
- The Transportation problem: A company builds laptops in four factories, each with certain supply power. Four cities have laptop demands. There is a cost $c_{i,j}$ for tranporting a laptop from factor i to city j. What is the best assignment of transport in order to minimize the cost?



- A silly way to solve this: run through all possibilities! Well how do I do this?? Not so easy... If supply and demand are all ONE and if number of cities and factories is $n = 35$, and a computer took 10^{-9} seconds to check one possibility, it would take 200,000 years to solve!

My Favorite Example

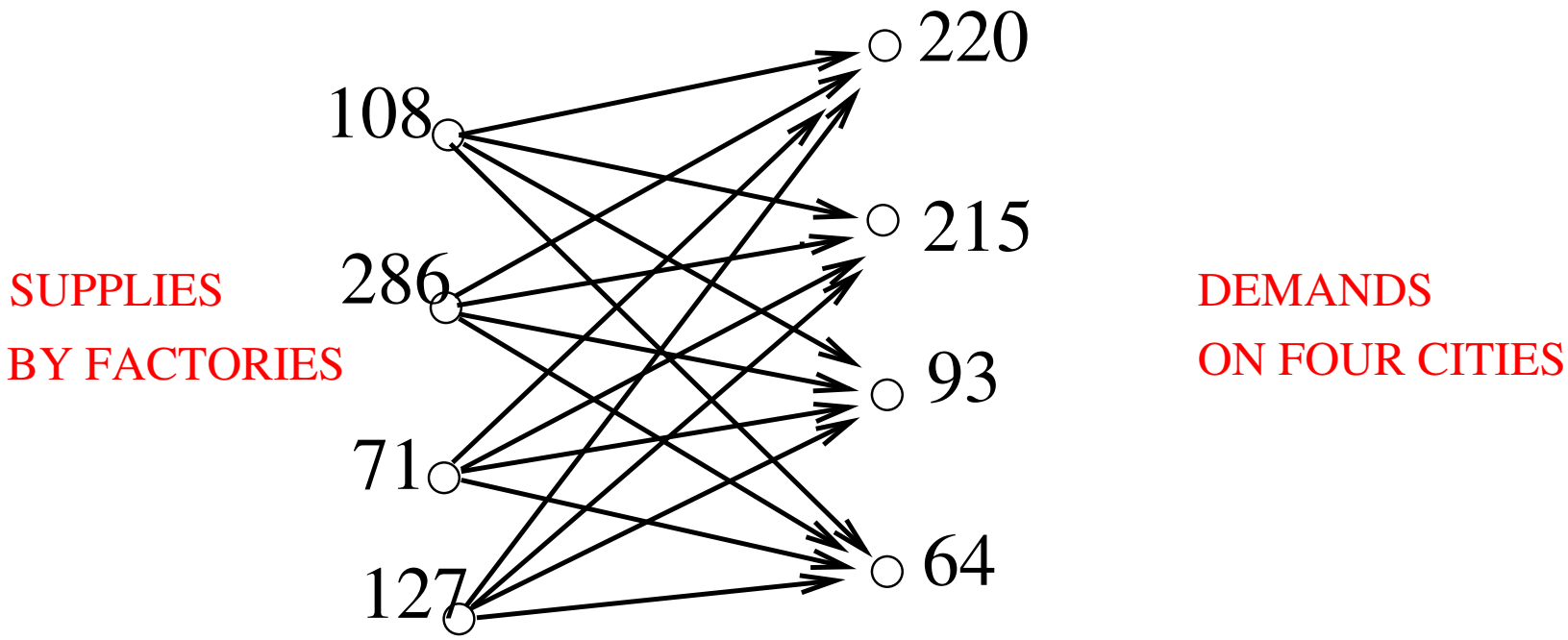
- The Transportation problem: A company builds laptops in four factories, each with certain supply power. Four cities have laptop demands. There is a cost $c_{i,j}$ for tranporting a laptop from factor i to city j. What is the best assignment of transport in order to minimize the cost?



- A silly way to solve this: run through all possibilities! Well how do I do this?? Not so easy... If supply and demand are all ONE and if number of cities and factories is $n = 35$, and a computer took 10^{-9} seconds to check one possibility, it would take 200,000 years to solve!

My Favorite Example

- The Transportation problem: A company builds laptops in four factories, each with certain supply power. Four cities have laptop demands. There is a cost $c_{i,j}$ for tranporting a laptop from factor i to city j. What is the best assignment of transport in order to minimize the cost?



- A silly way to solve this: run through all possibilities! Well how do I do this?? Not so easy... If supply and demand are all ONE and if number of cities and factories is $n = 35$, and a computer took 10^{-9} seconds to check one possibility, it would take 200,000 years to solve!

Integer Linear Programming

Linear programs

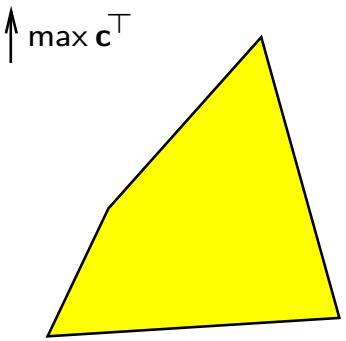
$$\begin{aligned} \max \quad & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \end{aligned}$$

Mixed integer programs

$$\begin{aligned} \max \quad & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ & \text{some } x_i \text{ integer} \end{aligned}$$

Integer programs

$$\begin{aligned} \max \quad & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ & \text{all } x_i \text{ integer} \end{aligned}$$



Integer Linear Programming

Linear programs

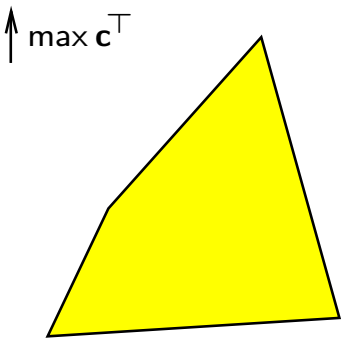
$$\begin{aligned} \max \quad & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \end{aligned}$$

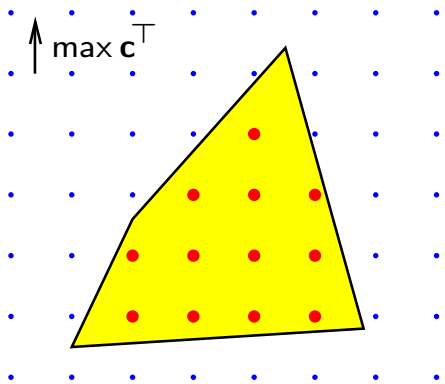
Mixed integer programs

$$\begin{aligned} \max \quad & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ & \text{some } x_i \text{ integer} \end{aligned}$$

Integer programs

$$\begin{aligned} \max \quad & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ & \text{all } x_i \text{ integer} \end{aligned}$$

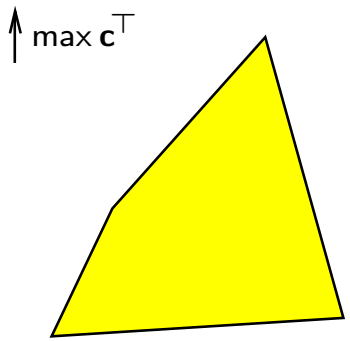




Integer Linear Programming

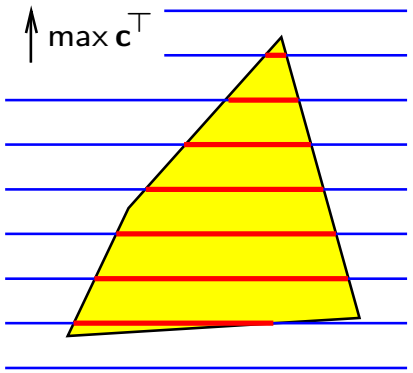
Linear programs

$$\begin{aligned} \max \quad & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \end{aligned}$$



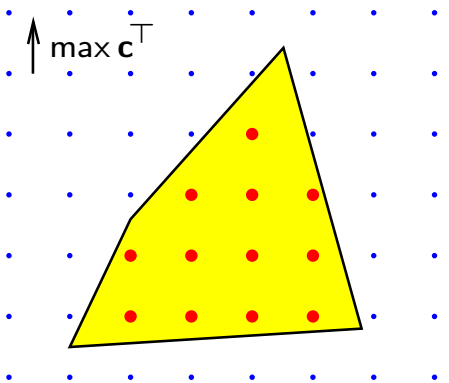
Mixed integer programs

$$\begin{aligned} \max \quad & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ & \text{some } x_i \text{ integer} \end{aligned}$$



Integer programs

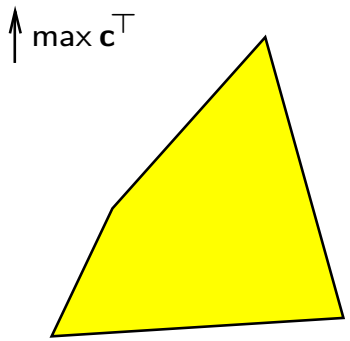
$$\begin{aligned} \max \quad & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ & \text{all } x_i \text{ integer} \end{aligned}$$



Integer Linear Programming

Linear programs

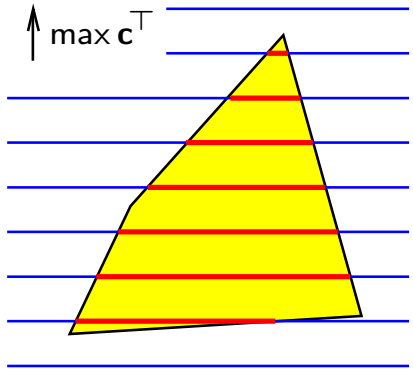
$$\begin{aligned} \max \quad & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \end{aligned}$$



Easy
(polynomial-time solvable)

Mixed integer programs

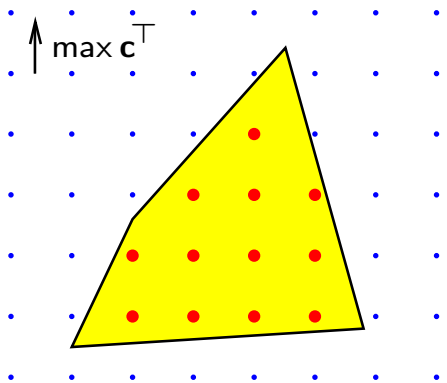
$$\begin{aligned} \max \quad & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ & \text{some } x_i \text{ integer} \end{aligned}$$



Hard
(NP-hard)

Integer programs

$$\begin{aligned} \max \quad & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ & \text{all } x_i \text{ integer} \end{aligned}$$



Hard
(NP-hard)

A way to model it via equations and inequalities

- Let $x_{i,j}$ be a variable indicating number of laptops factory i provides to city j. $x_{i,j}$ can only take non-negative integer values, $x_{i,j} \geq 0$.
- Then Since factory i produces a_i laptops we have

$$\sum_{j=1}^{n} x_{i,j} = a_i, \text{ for all } i = 1, \ldots, n.$$

and since city j needs b_i laptops

$$\sum_{i=1}^{n} x_{i,j} = b_j, \text{ for all } j = 1, \ldots, n.$$

- Now we minimize $\sum c_{i,j} x_{i,j}$.

A way to model it via equations and inequalities

- Let $x_{i,j}$ be a variable indicating number of laptops factory i provides to city j. $x_{i,j}$ can only take non-negative integer values, $x_{i,j} \geq 0$.
- Then Since factory i produces a_i laptops we have

$$\sum_{j=1}^{n} x_{i,j} = a_i, \text{ for all } i = 1, \ldots, n.$$

and since city j needs b_i laptops

$$\sum_{i=1}^{n} x_{i,j} = b_j, \text{ for all } j = 1, \ldots, n.$$

- Now we minimize $\sum c_{i,j} x_{i,j}$.

A way to model it via equations and inequalities

- Let $x_{i,j}$ be a variable indicating number of laptops factory i provides to city j. $x_{i,j}$ can only take non-negative integer values, $x_{i,j} \geq 0$.
- Then Since factory i produces a_i laptops we have

$$\sum_{j=1}^{n} x_{i,j} = a_i, \text{ for all } i = 1, \ldots, n.$$

and since city j needs b_i laptops

$$\sum_{i=1}^{n} x_{i,j} = b_j, \text{ for all } j = 1, \ldots, n.$$

- Now we minimize $\sum c_{i,j} x_{i,j}$.

A way to model it via equations and inequalities

- Let $x_{i,j}$ be a variable indicating number of laptops factory i provides to city j. $x_{i,j}$ can only take non-negative integer values, $x_{i,j} \geq 0$.
- Then Since factory i produces a_i laptops we have

$$\sum_{j=1}^{n} x_{i,j} = a_i, \text{ for all } i = 1, \ldots, n.$$

and since city j needs b_i laptops

$$\sum_{i=1}^{n} x_{i,j} = b_j, \text{ for all } j = 1, \ldots, n.$$

- Now we minimize $\sum c_{i,j} x_{i,j}$.

Integer Linear Programming: The state of the art

Traditional Algorithms

Dual (polyhedral) techniques

Cutting plane algorithms
– based on polyhedral theory

Enumeration

Branch-and-bound

Primal methods

Primal heuristic

Mathematical modelling – Strong initial IP formulation

Integer Linear Programming: The state of the art

Traditional Algorithms

Dual (polyhedral) techniques

Cutting plane algorithms
– based on polyhedral theory

Enumeration

Branch-and-bound

Primal methods

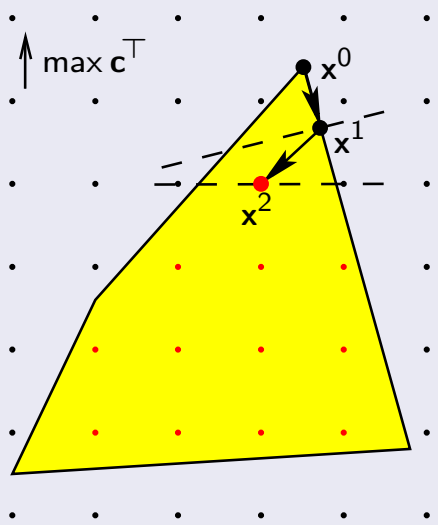
Primal heuristic

Mathematical modelling – Strong initial IP formulation

Integer Linear Programming: The state of the art

Traditional Algorithms

Dual (polyhedral) techniques



Cutting plane algorithms
– based on polyhedral theory

Enumeration

Branch-and-bound

Primal methods

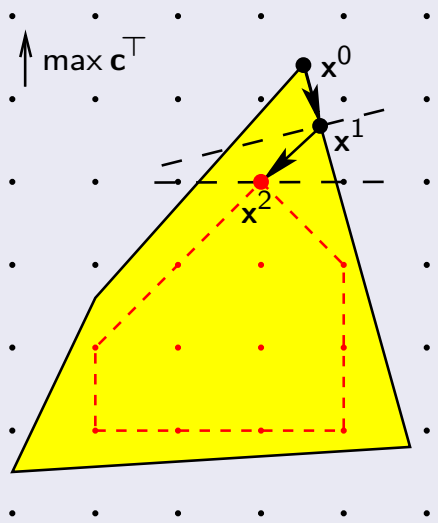
Primal heuristic

Mathematical modelling – Strong initial IP formulation

Integer Linear Programming: The state of the art

Traditional Algorithms

Dual (polyhedral) techniques



Cutting plane algorithms
– based on polyhedral theory

Enumeration

Branch-and-bound

Primal methods

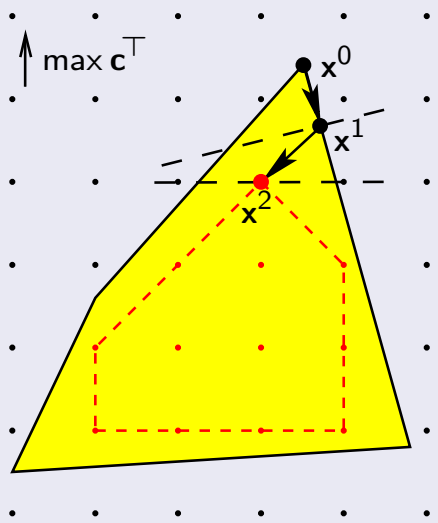
Primal heuristic

Mathematical modelling – Strong initial IP formulation

Integer Linear Programming: The state of the art

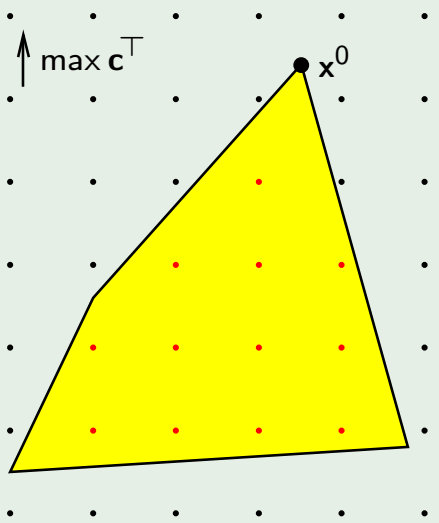
Traditional Algorithms

Dual (polyhedral) techniques



Cutting plane algorithms
– based on polyhedral theory

Enumeration



Branch-and-bound

Primal methods

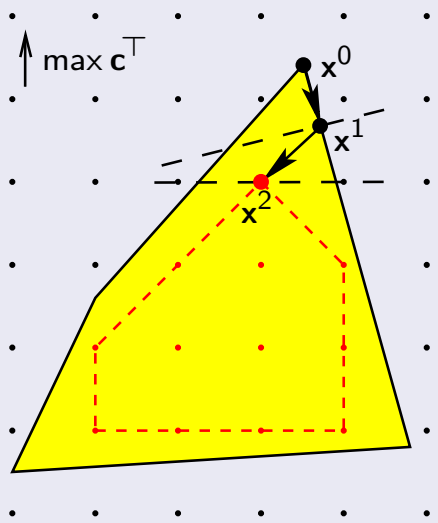
Primal heuristic

Mathematical modelling – Strong initial IP formulation

Integer Linear Programming: The state of the art

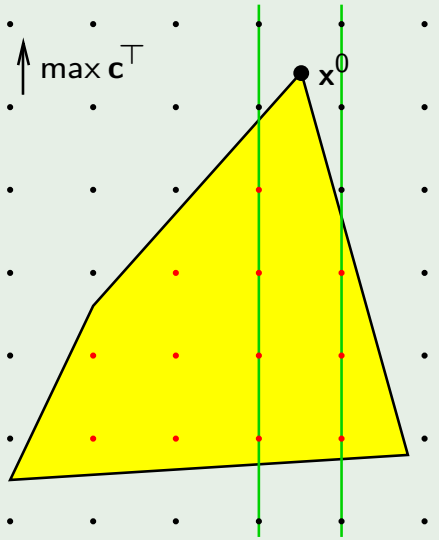
Traditional Algorithms

Dual (polyhedral) techniques



Cutting plane algorithms
– based on polyhedral theory

Enumeration



Branch-and-bound

Primal methods

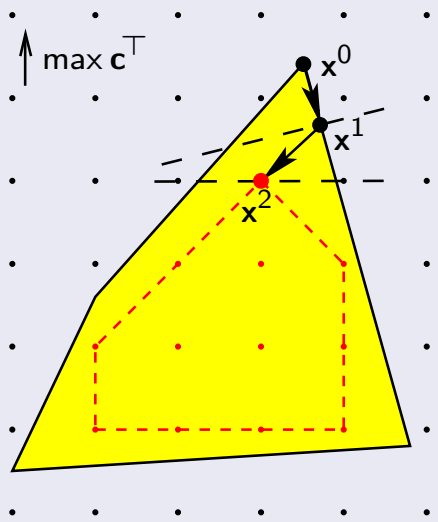
Primal heuristic

Mathematical modelling – Strong initial IP formulation

Integer Linear Programming: The state of the art

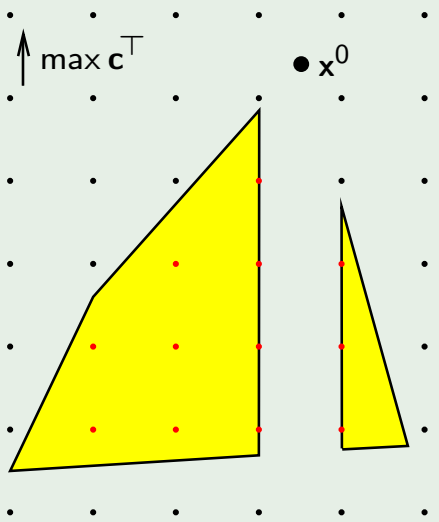
Traditional Algorithms

Dual (polyhedral) techniques



Cutting plane algorithms
– based on polyhedral theory

Enumeration



Branch-and-bound

Primal methods

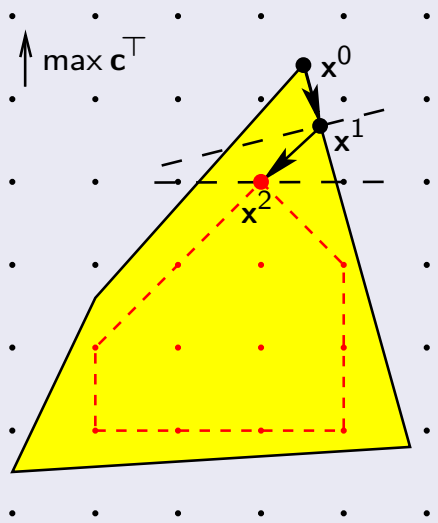
Primal heuristic

Mathematical modelling – Strong initial IP formulation

Integer Linear Programming: The state of the art

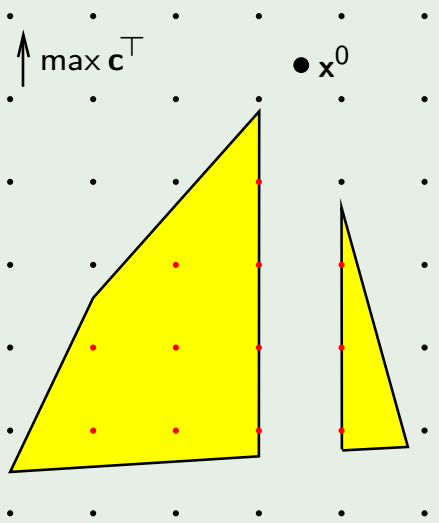
Traditional Algorithms

Dual (polyhedral) techniques



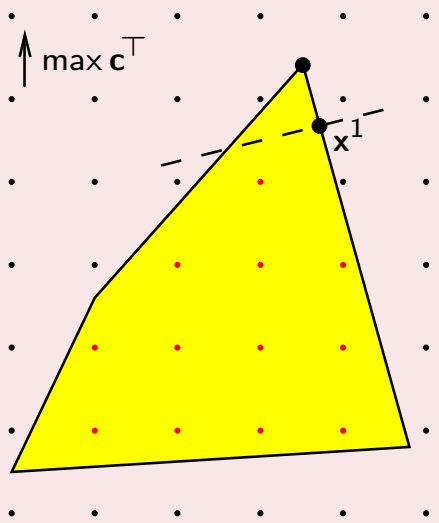
Cutting plane algorithms
– based on polyhedral theory

Enumeration



Branch-and-bound

Primal methods



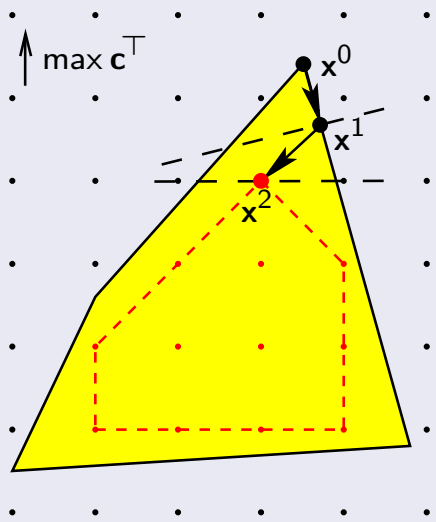
Primal heuristic

Mathematical modelling – Strong initial IP formulation

Integer Linear Programming: The state of the art

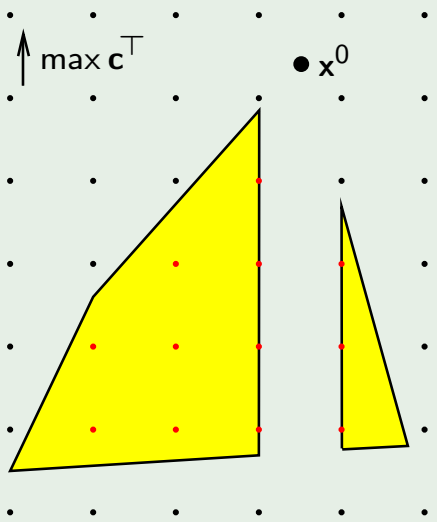
Traditional Algorithms

Dual (polyhedral) techniques



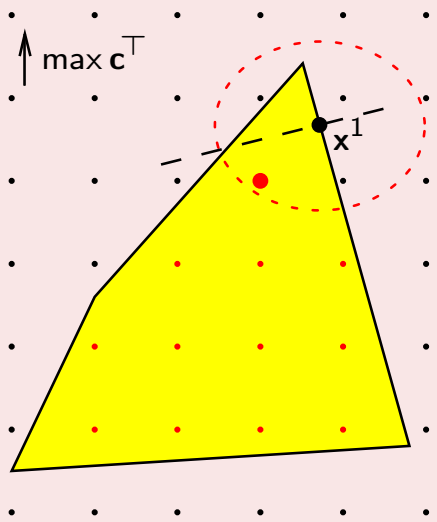
Cutting plane algorithms
– based on polyhedral theory

Enumeration



Branch-and-bound

Primal methods



Primal heuristic

Mathematical modelling – Strong initial IP formulation

WE NEED NEW MATHEMATICAL METHODS!!

REASON ONE: Reality is NON-LINEAR!

Non-linear Mixed Integer Optimization

$$\max/\min\ f(x_1, \ldots, x_d)$$
$$\text{subject to } g_j(x_1, \ldots, x_d) \leq 0,$$
$$\text{for } j = 1 \ldots s, \text{ and with}$$

some x_i integer!

bad news!!

The problem is INCREDIBLY HARD
It is UNDECIDABLE already when f, g_i's are
polynomials and even with number of variables=10.

where

- The constraints f and g_i's can be non-linear functions now!
- Problem has huge modeling power!

REASON ONE: Reality is NON-LINEAR!

Non-linear Mixed Integer Optimization

$$\max/\min\ f(x_1, \ldots, x_d)$$
$$\text{subject to } g_j(x_1, \ldots, x_d) \leq 0,$$
$$\text{for } j = 1 \ldots s, \text{ and with}$$

some x_i integer!

bad news!!

The problem is INCREDIBLY HARD
It is UNDECIDABLE already when f,g_i's are
polynomials and even with number of variables=10.

where

- The constraints f and g_i's can be non-linear functions now!
- Problem has huge modeling power!

REASON TWO: Even baby problems unsolvable with traditional techniques!

- Market Share problem (Cornéujols- Dawande, Williams)

  minimize $\sum_{i=1}^{m} |s_i|$ subject to the constraints

  $$\sum_{j=1}^{n} a_{i,j} x_j + s_i = d_i, \quad i = 1, \ldots, m$$

  $$x_j \in \{0, 1\}, \; j = 1, \ldots, n, \text{ and all } s_i \in \text{integer}$$

- Nasty Knapsack problems (Aardal, Bixby et al)

  Minimize or maximize $\sum_{i=1}^{10} x_i$, subject to $x_i \geq 0$ and
  $3719x_1 + 20289x_2 + 29067x_3 + 60517x_4 + 64354x_5 + 65633x_6 + 76969x_7 +$
  $102024x_8 + 106036x_9 + 119930x_{10} = 13385100$

REASON TWO: Even baby problems unsolvable with traditional techniques!

- Market Share problem (Cornéujols- Dawande, Williams)

  minimize $\sum_{i=1}^{m} |s_i|$ subject to the constraints

  $$\sum_{j=1}^{n} a_{i,j} x_j + s_i = d_i, \quad i = 1, \ldots, m$$

  $$x_j \in \{0, 1\}, \; j = 1, \ldots, n, \text{ and all } s_i \in \text{integer}$$

- Nasty Knapsack problems (Aardal, Bixby et al)

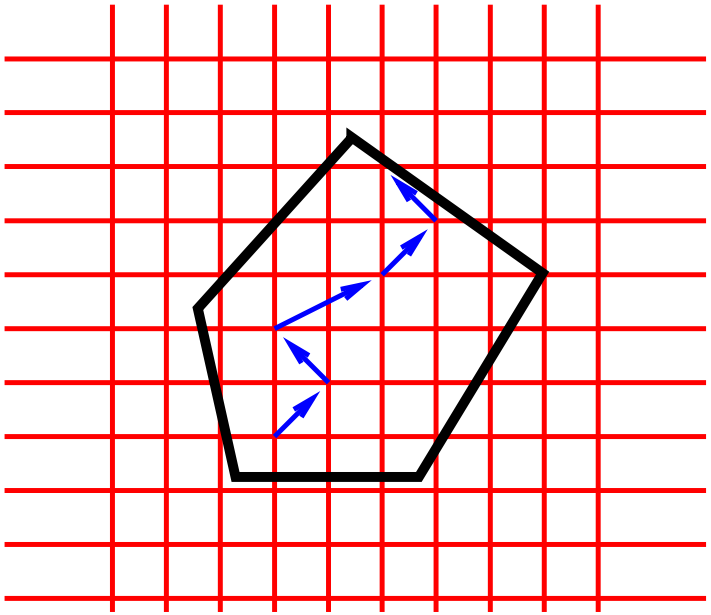
  Minimize or maximize $\sum_{i=1}^{10} x_i$, subject to $x_i \geq 0$ and
  $3719x_1 + 20289x_2 + 29067x_3 + 60517x_4 + 64354x_5 + 65633x_6 + 76969x_7 +$
  $102024x_8 + 106036x_9 + 119930x_{10} = 13385100$

A Taste of Two New Algebraic Techniques

Graver bases for ILP

Test Sets and Augmentation Methods

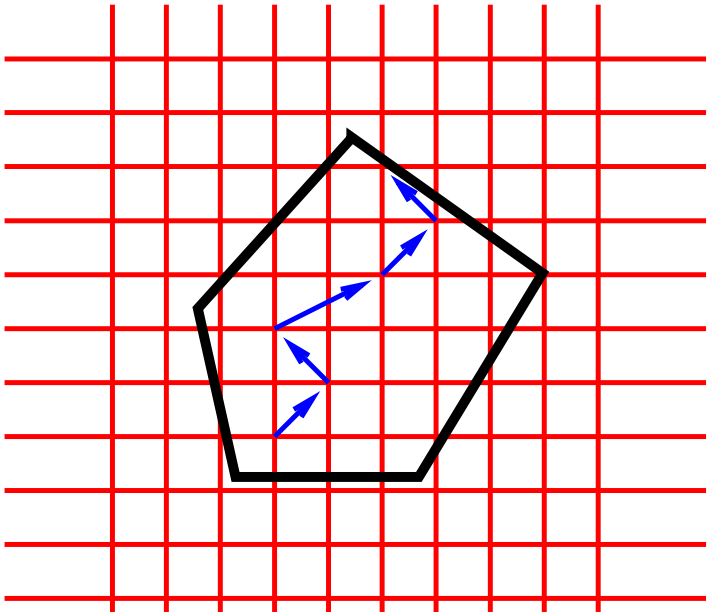
- A TEST SET is a finite collection of integral vectors with the property that every feasible non-optimal solution of an integer program can be improved by adding a vector in the test set.



- **We focus here on Algebraic Test Sets:** Graver and Gröbner bases, Hilbert bases, integral basis method. Work by Graver, Scarf, Sturmfels, Weismantel et al. and many others.

Test Sets and Augmentation Methods

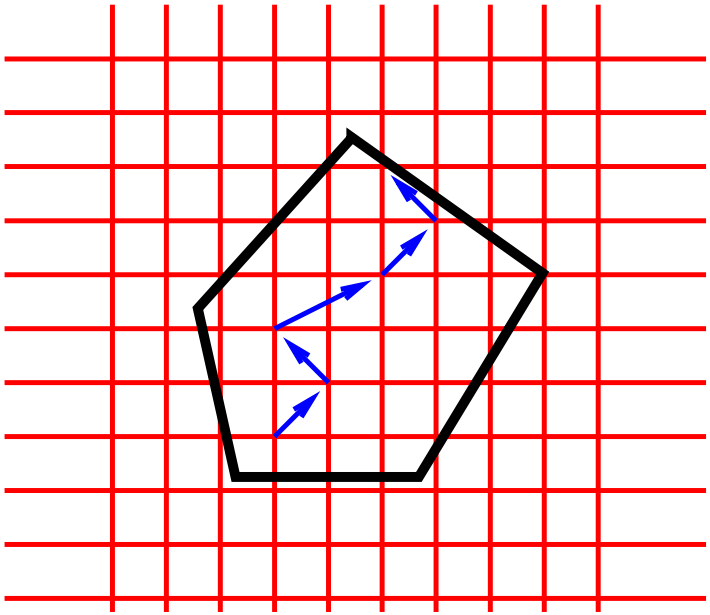
- A TEST SET is a finite collection of integral vectors with the property that every feasible non-optimal solution of an integer program can be improved by adding a vector in the test set.



- **We focus here on Algebraic Test Sets:** Graver and Gröbner bases, Hilbert bases, integral basis method. Work by Graver, Scarf, Sturmfels, Weismantel et al. and many others.

Test Sets and Augmentation Methods

- A TEST SET is a finite collection of integral vectors with the property that every feasible non-optimal solution of an integer program can be improved by adding a vector in the test set.



- **We focus here on Algebraic Test Sets:** Graver and Gröbner bases, Hilbert bases, integral basis method. Work by Graver, Scarf, Sturmfels, Weismantel et al. and many others.

Graver Bases

- The lattice $L(A) = \{x \in \mathbf{Z}^n : Ax = 0\}$ has a natural partial order. For $u, v \in \mathbf{Z}^n$ we say that u is *conformal* to v, denoted $u \sqsubset v$, if $|u_i| \leq |v_i|$ and $u_i v_i \geq 0$ for $i = 1, \ldots, n$, that is, u and v lie in the same orthant of $\mathbf{R}^n$ and each component of u is bounded by the corresponding component of v in absolute value.
- The Graver basis of an integer matrix A is the set of conformal-minimal nonzero integer dependencies on A.
- **Example:** If $A = [1\ 2\ 1]$ then its Graver basis is

$$\pm\{[2, -1, 0], [0, -1, 2], [1, 0, -1], [1, -1, 1]\}$$

.

Graver Bases

- The lattice $L(A) = \{x \in \mathbf{Z}^n : Ax = 0\}$ has a natural partial order. For $u, v \in \mathbf{Z}^n$ we say that u is *conformal* to v, denoted $u \sqsubset v$, if $|u_i| \leq |v_i|$ and $u_i v_i \geq 0$ for $i = 1, \ldots, n$, that is, u and v lie in the same orthant of $\mathbf{R}^n$ and each component of u is bounded by the corresponding component of v in absolute value.
- The Graver basis of an integer matrix A is the set of conformal-minimal nonzero integer dependencies on A.
- **Example:** If $A = [1\ 2\ 1]$ then its Graver basis is

$$\pm\{[2, -1, 0], [0, -1, 2], [1, 0, -1], [1, -1, 1]\}$$

.

Graver Bases

- The lattice $L(A) = \{x \in \mathbf{Z}^n : Ax = 0\}$ has a natural partial order. For $u, v \in \mathbf{Z}^n$ we say that u is *conformal* to v, denoted $u \sqsubset v$, if $|u_i| \leq |v_i|$ and $u_i v_i \geq 0$ for $i = 1, \ldots, n$, that is, u and v lie in the same orthant of $\mathbf{R}^n$ and each component of u is bounded by the corresponding component of v in absolute value.
- The Graver basis of an integer matrix A is the set of conformal-minimal nonzero integer dependencies on A.
- **Example:** If $A = [1\ 2\ 1]$ then its Graver basis is

$$\pm\{[2, -1, 0], [0, -1, 2], [1, 0, -1], [1, -1, 1]\}$$

.

- **Theorem** [J. Graver 1975] Graver bases for A can be used to solve the augmentation problem Given $A \in \mathbf{Z}^{m \times n}$, $x \in \mathbf{N}^n$ and $c \in \mathbf{Z}^n$, either find an improving direction $g \in \mathbf{Z}^n$, namely one with $x - g \in \{y \in \mathbf{N}^n : Ay = Ax\}$ and $cg > 0$, or assert that no such g exists.
- The fastest algorithm to compute Graver bases is based on a completion and project-and-lift method. Implemented in `4ti2` (by R. Hemmecke and P. Malkin). Equivalent to the computation of minimal Hilbert bases.
- Graver bases contain, and generalize, the LP test set given by the circuits of the matrix A. Circuits contain all possible edges of polyhedra in the family

$$P(b) := \{x |\ Ax = b, x \geq 0\}$$

.

- **Theorem** The Graver basis contains all edges for all integer hulls $conv(\{x |\ Ax = b, x \geq 0,\ x \in \mathbf{Z}^n\})$ as b changes.

- **Theorem** [J. Graver 1975] Graver bases for A can be used to solve the augmentation problem Given $A \in \mathbf{Z}^{m \times n}$, $x \in \mathbf{N}^n$ and $c \in \mathbf{Z}^n$, either find an improving direction $g \in \mathbf{Z}^n$, namely one with $x - g \in \{y \in \mathbf{N}^n : Ay = Ax\}$ and $cg > 0$, or assert that no such g exists.
- The fastest algorithm to compute Graver bases is based on a completion and project-and-lift method. Implemented in `4ti2` (by R. Hemmecke and P. Malkin). Equivalent to the computation of minimal Hilbert bases.
- Graver bases contain, and generalize, the LP test set given by the circuits of the matrix A. Circuits contain all possible edges of polyhedra in the family

$$P(b) := \{x |\ Ax = b, x \geq 0\}$$

.
- **Theorem** The Graver basis contains all edges for all integer hulls $conv(\{x |\ Ax = b, x \geq 0,\ x \in \mathbf{Z}^n\})$ as b changes.

- **Theorem** [J. Graver 1975] Graver bases for A can be used to solve the augmentation problem Given $A \in \mathbf{Z}^{m \times n}$, $x \in \mathbf{N}^n$ and $c \in \mathbf{Z}^n$, either find an improving direction $g \in \mathbf{Z}^n$, namely one with $x - g \in \{y \in \mathbf{N}^n : Ay = Ax\}$ and $cg > 0$, or assert that no such g exists.
- The fastest algorithm to compute Graver bases is based on a completion and project-and-lift method. Implemented in `4ti2` (by R. Hemmecke and P. Malkin). Equivalent to the computation of minimal Hilbert bases.
- Graver bases contain, and generalize, the LP test set given by the circuits of the matrix A. Circuits contain all possible edges of polyhedra in the family

$$P(b) := \{x |\ Ax = b, x \geq 0\}$$

.

- **Theorem** The Graver basis contains all edges for all integer hulls $conv(\{x |\ Ax = b, x \geq 0,\ x \in \mathbf{Z}^n\})$ as b changes.

- **Theorem** [J. Graver 1975] Graver bases for A can be used to solve the augmentation problem Given $A \in \mathbf{Z}^{m \times n}$, $x \in \mathbf{N}^n$ and $c \in \mathbf{Z}^n$, either find an improving direction $g \in \mathbf{Z}^n$, namely one with $x - g \in \{y \in \mathbf{N}^n : Ay = Ax\}$ and $cg > 0$, or assert that no such g exists.
- The fastest algorithm to compute Graver bases is based on a completion and project-and-lift method. Implemented in `4ti2` (by R. Hemmecke and P. Malkin). Equivalent to the computation of minimal Hilbert bases.
- Graver bases contain, and generalize, the LP test set given by the circuits of the matrix A. Circuits contain all possible edges of polyhedra in the family

$$P(b) := \{x | \ Ax = b, x \geq 0\}$$

.

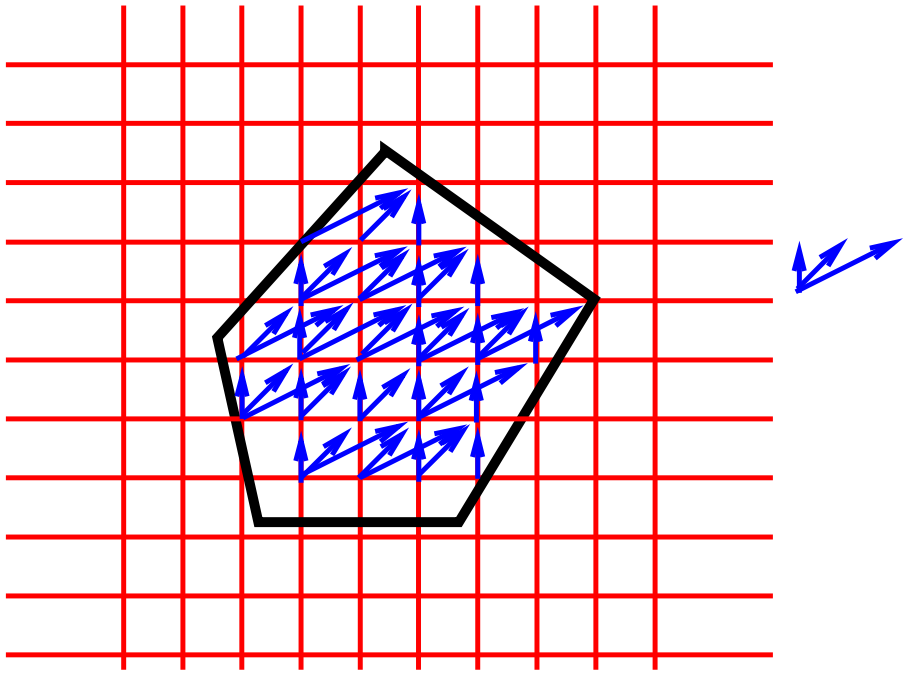
- **Theorem** The Graver basis contains all edges for all integer hulls $conv(\{x | \ Ax = b, x \geq 0, \ x \in \mathbf{Z}^n\})$ as b changes.

- For a fixed cost vector c, we can visualize a Graver basis of of an integer program by creating a graph!!
- Here is how to construct it, consider

$$L(b) := \{x |\ Ax = b, x \geq 0,\ x \in \mathbf{Z}^n\}$$

.

Nodes are lattice points in $L(b)$ and the Graver basis elements give directed edges departing from each lattice point $u \in L(b)$.

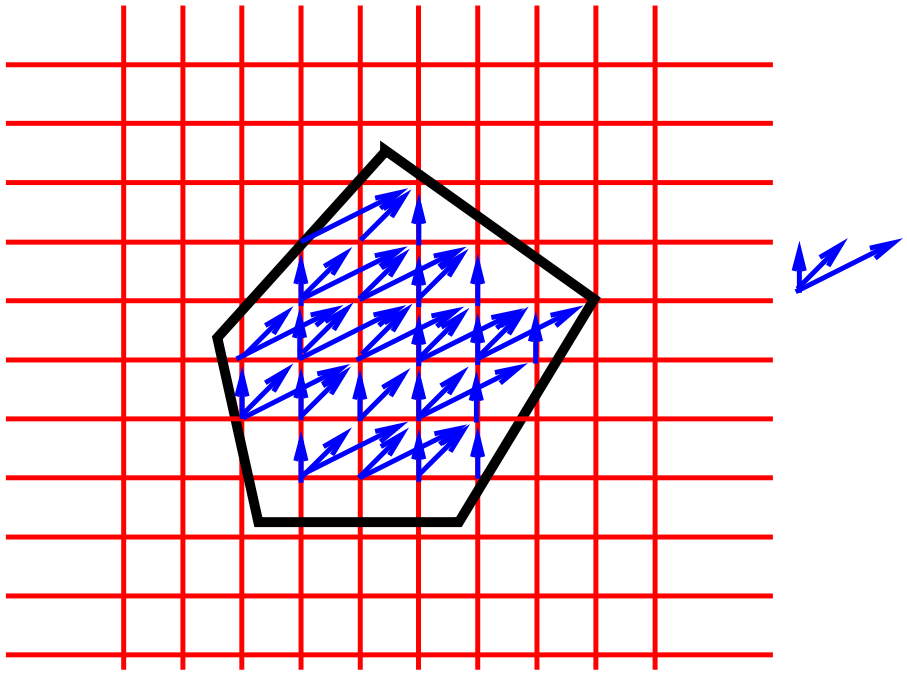


- For a fixed cost vector c, we can visualize a Graver basis of of an integer program by creating a graph!!
- Here is how to construct it, consider

$$L(b) := \{x |\ Ax = b, x \geq 0,\ x \in \mathbf{Z}^n\}$$

.

Nodes are lattice points in $L(b)$ and the Graver basis elements give directed edges departing from each lattice point $u \in L(b)$.



BAD NEWS!!

- Graver test sets can be exponentially large even in fixed dimension!
- People typically store a list of the whole test set. Very large indeed. (New ways to store them available, using algebra!).
- Very hard to compute, you don't want to do this too often.
- GOAL: Make test sets efficient at least in special cases.
- OUR NEW RESULTS: We show a useful cases where Graver bases become very manageable.

BAD NEWS!!

- Graver test sets can be exponentially large even in fixed dimension!
- People typically store a list of the whole test set. Very large indeed. (New ways to store them available, using algebra!).
- Very hard to compute, you don't want to do this too often.
- GOAL: Make test sets efficient at least in special cases.
- OUR NEW RESULTS: We show a useful cases where Graver bases become very manageable.

BAD NEWS!!

- Graver test sets can be exponentially large even in fixed dimension!
- People typically store a list of the whole test set. Very large indeed. (New ways to store them available, using algebra!).
- Very hard to compute, you don't want to do this too often.
- GOAL: Make test sets efficient at least in special cases.
- OUR NEW RESULTS: We show a useful cases where Graver bases become very manageable.

BAD NEWS!!

- Graver test sets can be exponentially large even in fixed dimension!
- People typically store a list of the whole test set. Very large indeed. (New ways to store them available, using algebra!).
- Very hard to compute, you don't want to do this too often.
- GOAL: Make test sets efficient at least in special cases.
- OUR NEW RESULTS: We show a useful cases where Graver bases become very manageable.

N-fold Systems

Fix any pair of integer matrices A and B with the same number of columns, of dimensions $r \times q$ and $s \times q$, respectively. The n-fold matrix of the ordered pair A, B is the following $(s + nr) \times nq$ matrix,

$$[A, B]^{(n)} \quad := \quad (\mathbf{1}_n \otimes B) \oplus (I_n \otimes A) \quad = \quad \begin{pmatrix} B & B & B & \cdots & B \\ A & 0 & 0 & \cdots & 0 \\ 0 & A & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A \end{pmatrix} .$$

N-fold systems DO appear in applications! Multiway Transportation problems are examples!

Theorem Fix any integer matrices A, B of sizes $r \times q$ and $s \times q$, respectively. Then there is a polynomial time algorithm that, given any n and any integer vectors b and c, solves the corresponding n-fold integer programming problem.

$$\min\{cx : \ [A, B]^{(n)} x = b, \ x \in \mathbf{N}^{nq}\} \quad .$$

N-fold Systems

Fix any pair of integer matrices A and B with the same number of columns, of dimensions $r \times q$ and $s \times q$, respectively. The n-fold matrix of the ordered pair A, B is the following $(s + nr) \times nq$ matrix,

$$[A, B]^{(n)} \quad := \quad (\mathbf{1}_n \otimes B) \oplus (I_n \otimes A) \quad = \quad \begin{pmatrix} B & B & B & \cdots & B \\ A & 0 & 0 & \cdots & 0 \\ 0 & A & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A \end{pmatrix} .$$

N-fold systems DO appear in applications! Multiway Transportation problems are examples!

Theorem Fix any integer matrices A, B of sizes $r \times q$ and $s \times q$, respectively. Then there is a polynomial time algorithm that, given any n and any integer vectors b and c, solves the corresponding n-fold integer programming problem.

$$\min\{cx : \ [A, B]^{(n)} x = b, \ x \in \mathbf{N}^{nq}\} \quad .$$

N-fold Systems

Fix any pair of integer matrices A and B with the same number of columns, of dimensions $r \times q$ and $s \times q$, respectively. The n-fold matrix of the ordered pair A, B is the following $(s + nr) \times nq$ matrix,

$$[A, B]^{(n)} \quad := \quad (\mathbf{1}_n \otimes B) \oplus (I_n \otimes A) \quad = \quad \begin{pmatrix} B & B & B & \cdots & B \\ A & 0 & 0 & \cdots & 0 \\ 0 & A & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A \end{pmatrix} \quad .$$

N-fold systems DO appear in applications! Multiway Transportation problems are examples!

Theorem Fix any integer matrices A, B of sizes $r \times q$ and $s \times q$, respectively. Then there is a polynomial time algorithm that, given any n and any integer vectors b and c, solves the corresponding n-fold integer programming problem.

$$\min\{cx : \ [A, B]^{(n)} x = b, \ x \in \mathbf{N}^{nq}\} \quad .$$

- **Key Lemma** Fix any pair of integer matrices $A \in \mathbf{Z}^{r \times q}$ and $B \in \mathbf{Z}^{s \times q}$. Then there is a polynomial time algorithm that, given n, computes the Graver basis $G([A, B]^{(n)})$ of the n-fold matrix $[A, B]^{(n)}$. In particular, the cardinality and the bit size of $G([A, B]^{(n)})$ are bounded by a polynomial function of n.
- **Key Idea (from Commutative Algebra!)** [Aoki-Takemura, Santos-Sturmfels, Hosten-Sullivant] For every pair of integer matrices $A \in \mathbf{Z}^{r \times q}$ and $B \in \mathbf{Z}^{s \times q}$, there exists a constant $g(A, B)$ such that for all n, the Graver basis of $[A, B]^{(n)}$ consists of vectors with at most $g(A, B)$ the number nonzero components.
  The smallest constant $g(A, B)$ possible is the Graver complexity of A, B.

- **Key Lemma** Fix any pair of integer matrices $A \in \mathbf{Z}^{r \times q}$ and $B \in \mathbf{Z}^{s \times q}$. Then there is a polynomial time algorithm that, given n, computes the Graver basis $G([A, B]^{(n)})$ of the n-fold matrix $[A, B]^{(n)}$. In particular, the cardinality and the bit size of $G([A, B]^{(n)})$ are bounded by a polynomial function of n.
- **Key Idea (from Commutative Algebra!)** [Aoki-Takemura, Santos-Sturmfels, Hosten-Sullivant] For every pair of integer matrices $A \in \mathbf{Z}^{r \times q}$ and $B \in \mathbf{Z}^{s \times q}$, there exists a constant $g(A, B)$ such that for all n, the Graver basis of $[A, B]^{(n)}$ consists of vectors with at most $g(A, B)$ the number nonzero components.
  The smallest constant $g(A, B)$ possible is the Graver complexity of A, B.

Example: Consider the matrices $A = [1\ 1]$ and $B = I_2$. The Graver complexity of the pair A, B is $g(A, B) = 2$.

$$[A, B]^{(2)} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \quad G([A, B]^{(2)}) = \pm \begin{pmatrix} 1 & -1 & -1 & 1 \end{pmatrix} .$$

By our theorem, the Graver basis of the 4-fold matrix

$$[A, B]^{(4)} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix},$$

$$G([A, B]^{(4)}) = \pm \begin{pmatrix} 1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 \end{pmatrix}.$$

A Taste of Two New Algebraic Techniques

Rational Functions for Non-Linear Integer Optimization

Going Beyond Convex Integer Optimization!!!

Problem type

$$\begin{aligned} \max \quad & f(x_1, \ldots, x_d) \\ \text{subject to} \quad & (x_1, \ldots, x_d) \in P \cap \mathbf{Z}^d, \end{aligned}$$

where

- P is a polytope (bounded polyhedron) given by linear constraints,
- f is a (multivariate) polynomial function non-negative over $P \cap \mathbf{Z}^d$,
- the dimension d is fixed.

Prior Work

- Integer Linear Programming can be solved in polynomial time
  (H. W. Lenstra Jr, 1983)
- Convex polynomials f can be minimized in polynomial time
  (Khachiyan and Porkolab, 2000)
- Optimizing an arbitrary degree-4 polynomial f for $d = 2$ is NP-hard

Fully Polynomial-Time Approximation Scheme (FPTA S)

For every $\epsilon > 0$, there exists an algorithm $\mathcal{A}_\epsilon$ with running time polynomial in the input size and $1/\epsilon$, which computes an approximation $\mathbf{x}_\epsilon$ wi th

$$|f(\mathbf{x}_\epsilon) - f(\mathbf{x}^{\max})| \leq \epsilon f(\mathbf{x}^{\max}).$$

Going Beyond Convex Integer Optimization!!!

Problem type

$$\begin{aligned} \max \quad & f(x_1, \ldots, x_d) \\ \text{subject to} \quad & (x_1, \ldots, x_d) \in P \cap \mathbf{Z}^d, \end{aligned}$$

where

- P is a polytope (bounded polyhedron) given by linear constraints,
- f is a (multivariate) polynomial function non-negative over $P \cap \mathbf{Z}^d$,
- the dimension d is fixed.

Prior Work

- Integer Linear Programming can be solved in polynomial time
  (H. W. Lenstra Jr, 1983)
- Convex polynomials f can be minimized in polynomial time
  (Khachiyan and Porkolab, 2000)
- Optimizing an arbitrary degree-4 polynomial f for $d = 2$ is NP-hard

Fully Polynomial-Time Approximation Scheme (FPTA S)

For every $\epsilon > 0$, there exists an algorithm $\mathcal{A}_\epsilon$ with running time polynomial in the input size and $1/\epsilon$, which computes an approximation $\mathbf{x}_\epsilon$ wi th

$$|f(\mathbf{x}_\epsilon) - f(\mathbf{x}^{\max})| \leq \epsilon f(\mathbf{x}^{\max}).$$

Going Beyond Convex Integer Optimization!!!

Problem type

$$\begin{aligned} \max \quad & f(x_1, \ldots, x_d) \\ \text{subject to} \quad & (x_1, \ldots, x_d) \in P \cap \mathbf{Z}^d, \end{aligned}$$

where

- P is a polytope (bounded polyhedron) given by linear constraints,
- f is a (multivariate) polynomial function non-negative over $P \cap \mathbf{Z}^d$,
- the dimension d is fixed.

Prior Work

- Integer Linear Programming can be solved in polynomial time (H. W. Lenstra Jr, 1983)
- Convex polynomials f can be minimized in polynomial time (Khachiyan and Porkolab, 2000)
- Optimizing an arbitrary degree-4 polynomial f for $d = 2$ is NP-hard

Fully Polynomial-Time Approximation Scheme (FPTA S)

For every $\epsilon > 0$, there exists an algorithm $\mathcal{A}_\epsilon$ with running time polynomial in the input size and $1/\epsilon$, which computes an approximation $\mathbf{x}_\epsilon$ wi th

$$|f(\mathbf{x}_\epsilon) - f(\mathbf{x}^{\max})| \leq \epsilon f(\mathbf{x}^{\max}).$$

Going Beyond Convex Integer Optimization!!!

Problem type

$$\begin{aligned} \max \quad & f(x_1, \ldots, x_d) \\ \text{subject to} \quad & (x_1, \ldots, x_d) \in P \cap \mathbf{Z}^d, \end{aligned}$$

where

- P is a polytope (bounded polyhedron) given by linear constraints,
- f is a (multivariate) polynomial function non-negative over $P \cap \mathbf{Z}^d$,
- the dimension d is fixed.

Prior Work

- Integer Linear Programming can be solved in polynomial time (H. W. Lenstra Jr, 1983)
- Convex polynomials f can be minimized in polynomial time (Khachiyan and Porkolab, 2000)
- Optimizing an arbitrary degree-4 polynomial f for $d = 2$ is NP-hard

Fully Polynomial-Time Approximation Scheme (FPTA S)

For every $\epsilon > 0$, there exists an algorithm $\mathcal{A}_\epsilon$ with running time polynomial in the input size and $1/\epsilon$, which computes an approximation $\mathbf{x}_\epsilon$ wi th

$$|f(\mathbf{x}_\epsilon) - f(\mathbf{x}^{\max})| \leq \epsilon f(\mathbf{x}^{\max}).$$

Going Beyond Convex Integer Optimization!!!

Problem type

$$\begin{aligned} \max \quad & f(x_1, \ldots, x_d) \\ \text{subject to} \quad & (x_1, \ldots, x_d) \in P \cap \mathbf{Z}^d, \end{aligned}$$

where

- P is a polytope (bounded polyhedron) given by linear constraints,
- f is a (multivariate) polynomial function non-negative over $P \cap \mathbf{Z}^d$,
- the dimension d is fixed.

Prior Work

- Integer Linear Programming can be solved in polynomial time (H. W. Lenstra Jr, 1983)
- Convex polynomials f can be minimized in polynomial time (Khachiyan and Porkolab, 2000)
- Optimizing an arbitrary degree-4 polynomial f for $d = 2$ is NP-hard

Fully Polynomial-Time Approximation Scheme (FPTA S)

For every $\epsilon > 0$, there exists an algorithm $\mathcal{A}_\epsilon$ with running time polynomial in the input size and $1/\epsilon$, which computes an approximation $\mathbf{x}_\epsilon$ wi th

$$|f(\mathbf{x}_\epsilon) - f(\mathbf{x}^{\max})| \leq \epsilon f(\mathbf{x}^{\max}).$$

Key Idea: Represent Sets of Lattice Points as Rational Function

Given $K \subset \mathbf{R}^d$ we define the sum

$$f(K) = \sum_{\alpha \in K \cap \mathbf{Z}^d} z_1^{\alpha_1} z_2^{\alpha_2} \dots z_n^{\alpha_n}.$$

Think of the lattice points as monomials!!! EXAMPLE: $(7, 4, -3)$ is $z_1^7 z_2^4 z_3^{-3}$. When K is a rational convex polyhedron, i.e. $K = \{x \in \mathbf{R}^n | Ax = b,\ Bx \leq b'\}$, where A, B are integral matrices and b, b' are integral vectors, The generating function $f(K)$, and thus **ALL** the lattice points of the polyhedron K, **can be encoded in a sum of rational functions!**

Key Idea: Represent Sets of Lattice Points as Rational Function

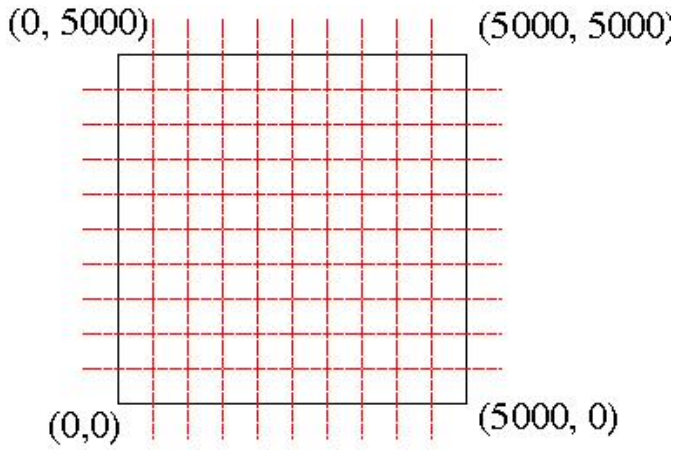
Given $K \subset \mathbf{R}^d$ we define the sum

$$f(K) = \sum_{\alpha \in K \cap \mathbf{Z}^d} z_1^{\alpha_1} z_2^{\alpha_2} \dots z_n^{\alpha_n}.$$

Think of the lattice points as monomials!!! EXAMPLE: $(7, 4, -3)$ is $z_1^7 z_2^4 z_3^{-3}$. When K is a rational convex polyhedron, i.e. $K = \{x \in \mathbf{R}^n | Ax = b,\ Bx \leq b'\}$, where A, B are integral matrices and b, b' are integral vectors, The generating function $f(K)$, and thus **ALL** the lattice points of the polyhedron K, **can be encoded in a sum of rational functions!**

Example

Let P be the square with vertices $V_1 = (0, 0)$, $V_2 = (5000, 0)$, $V_3 = (5000, 5000)$, and $V_4 = (0, 5000)$.



The generating function $f(P)$ has over 25,000,000 monomials,
$f(P) = 1 + z_1 + z_2 + z_1^1 z_2^2 + z_1^2 z_2 + \cdots + z_1^{5000} z_2^{5000}$,

But it can be written using only four rational functions

$$\frac{1}{(1-z_1)(1-z_2)} + \frac{{z_1}^{5000}}{(1-{z_1}^{-1})(1-z_2)} + \frac{{z_2}^{5000}}{(1-{z_2}^{-1})(1-z_1)} + \frac{{z_1}^{5000}{z_2}^{5000}}{(1-{z_1}^{-1})(1-{z_2}^{-1})}$$

Also, $f(tP, z)$ is

$$\frac{1}{(1-z_1)(1-z_2)} + \frac{{z_1}^{5000\cdot t}}{(1-{z_1}^{-1})(1-z_2)} + \frac{{z_2}^{5000\cdot t}}{(1-{z_2}^{-1})(1-z_1)} + \frac{{z_1}^{5000\cdot t}{z_2}^{5000\cdot t}}{(1-{z_1}^{-1})(1-{z_2}^{-1})}$$

Conclusions and Future work

- Traditional Methods are not sufficient to solve all current integer optimization models, even the simple linear ones!
- There is new demand to solve NON-LINEAR INTEGER optimization problems, not just model things linearly anymore.
- Tools from Commutative Algebra, Number Theory, Functional Analysis, Probability, and Convex Geometry are bound to play an stronger role in the foundations of new algorithmic tools!
- Not just the foundations need to be studied, new software is beginning to appear that uses all these ideas: `4ti2`, `LattE`.

Conclusions and Future work

- Traditional Methods are not sufficient to solve all current integer optimization models, even the simple linear ones!
- There is new demand to solve NON-LINEAR INTEGER optimization problems, not just model things linearly anymore.
- Tools from Commutative Algebra, Number Theory, Functional Analysis, Probability, and Convex Geometry are bound to play an stronger role in the foundations of new algorithmic tools!
- Not just the foundations need to be studied, new software is beginning to appear that uses all these ideas: `4ti2`, `LattE`.

Conclusions and Future work

- Traditional Methods are not sufficient to solve all current integer optimization models, even the simple linear ones!
- There is new demand to solve NON-LINEAR INTEGER optimization problems, not just model things linearly anymore.
- Tools from Commutative Algebra, Number Theory, Functional Analysis, Probability, and Convex Geometry are bound to play an stronger role in the foundations of new algorithmic tools!
- Not just the foundations need to be studied, new software is beginning to appear that uses all these ideas: `4ti2`, `LattE`.

Conclusions and Future work

- Traditional Methods are not sufficient to solve all current integer optimization models, even the simple linear ones!
- There is new demand to solve NON-LINEAR INTEGER optimization problems, not just model things linearly anymore.
- Tools from Commutative Algebra, Number Theory, Functional Analysis, Probability, and Convex Geometry are bound to play an stronger role in the foundations of new algorithmic tools!
- Not just the foundations need to be studied, new software is beginning to appear that uses all these ideas: `4ti2`, `LattE`.

Thank you

Gracias

Merci

Danke