Using Saturation Matrix to Efficiently Remove Redundant Inequalities

Chirantan Mukherjee

University of Western Ontario

June 25, 2024

Content

What is the Fourier-Motzkin Elimination?

Fourier-Motzkin elimination (FME) is a method to project polyhedral sets on lower dimensions.

What is the Fourier-Motzkin Elimination?

Fourier-Motzkin elimination (FME) is a method to project polyhedral sets on lower dimensions. The basic idea is similar to Gaussian elimination (GE) for equality systems.

$$\begin{array}{rcl} -2\,x_1 + 4\,x_2 - 3\,x_3 = 0 & & -2\,x_1 + 4\,x_2 - 3\,x_3 = 0 \\ -13\,x_1 + 24\,x_2 - 20\,x_3 = 0 & \xrightarrow{1 \; step \;\; GE} & 0\,x_1 - 2\,x_2 - \frac{1}{2}\,x_3 = 0 \\ -26\,x_1 + 54\,x_2 - 39\,x_3 = 0 & & 0\,x_1 + 2\,x_2 - 0\,x_3 = 0 \end{array}$$

What is the Fourier-Motzkin Elimination?

Fourier-Motzkin elimination (FME) is a method to project polyhedral sets on lower dimensions. The basic idea is similar to Gaussian elimination (GE) for equality systems.

$$\begin{aligned} -2\,x_1 + 4\,x_2 - 3\,x_3 &= 0 \\ -13\,x_1 + 24\,x_2 - 20\,x_3 &= 0 \\ -26\,x_1 + 54\,x_2 - 39\,x_3 &= 0 \end{aligned} \xrightarrow{1\ \ step\ \ GE} \begin{aligned} -2\,x_1 + 4\,x_2 - 3\,x_3 &= 0 \\ 0\,x_1 - 2\,x_2 - \tfrac{1}{2}\,x_3 &= 0 \\ 0\,x_1 + 2\,x_2 - 0\,x_3 &= 0 \end{aligned}$$

$$\begin{aligned} 3\,x_1 - 2\,x_2 + 1\,x_3 &\le 7 \\ -2\,x_1 + 2\,x_2 - 1\,x_3 &\le 12 \\ -4\,x_1 + 1\,x_2 - 3\,x_3 &\le 15 \end{aligned} \xrightarrow{1\ \ step\ \ FME} \begin{aligned} 0\,x_1 + 2\,x_2 - 1\,x_3 &\le 50 \\ 0\,x_1 - 5\,x_2 - 13\,x_3 &\le 73 \end{aligned}$$

Example

Eliminating t_1 from

$$A = \begin{cases} a_1 : 3t_1 - 2t_2 + t_3 \leq 7 \\ a_2 : -2t_1 + 2t_2 - t_3 \leq 12 \\ a_3 : -4t_1 + t_2 + 3t_3 \leq 15 \end{cases}$$

Example

Eliminating t_1 from

$$A = \begin{cases} a_1 : 3t_1 - 2t_2 + t_3 \leq 7 \\ a_2 : -2t_1 + 2t_2 - t_3 \leq 12 \\ a_3 : -4t_1 + t_2 + 3t_3 \leq 15 \end{cases}$$

$partition(A) = \{a_1\}, \{a_2, a_3\}$

Example

Eliminating t_1 from

$$A = \begin{cases} a_1 : 3t_1 - 2t_2 + t_3 \leq 7 \\ a_2 : -2t_1 + 2t_2 - t_3 \leq 12 \\ a_3 : -4t_1 + t_2 + 3t_3 \leq 15 \end{cases}$$

$partition(A) = \{a_1\}, \{a_2, a_3\}$
$combine(a_1, a_2) = combine(2a_1 + 3a_2) = 2t_2 - t_3 \leq 50$
$combine(a_1, a_3) = combine(4a_1 + 3a_3) = -5t_2 - 13t_3 \leq 73$

$$A' = \begin{cases} 2t_2 - t_3 \leq 50 \\ -5t_2 - 13t_3 \leq 73 \end{cases}$$

Example

Eliminating t_2 from

$$A' = \begin{cases} a_4 : 2t_2 - t_3 \leq 50 \\ a_5 : -5t_2 - 13t_3 \leq 73 \end{cases}$$

Example

Eliminating t_2 from

$$A' = \begin{cases} a_4 : 2t_2 - t_3 \leq 50 \\ a_5 : -5t_2 - 13t_3 \leq 73 \end{cases}$$

$partition(A) = \{a_4\}, \{a_5\}$

Example

Eliminating t_2 from

$$A' = \begin{cases} a_4 : 2t_2 - t_3 \leq 50 \\ a_5 : -5t_2 - 13t_3 \leq 73 \end{cases}$$

$partition(A) = \{a_4\}, \{a_5\}$
$combine(a_1, a_2) = combine(5a_4 + 2a_5) = -31t_3 \leq 396$

$$A'' = \begin{cases} -31t_3 \leq 396 \end{cases}$$

Background

- Projection of polyhedral sets has many applications in computer science
  - scheduling
  - dependence analysis (automatic parallelization)

Background

- Projection of polyhedral sets has many applications in computer science
  - scheduling
  - dependence analysis (automatic parallelization)
- The basic algorithm, proposed by Fourier (1827) and Motzkin (1936), is called *Fourier-Motzkin Elimination* (FME)

Background

- Projection of polyhedral sets has many applications in computer science
  - scheduling
  - dependence analysis (automatic parallelization)
- The basic algorithm, proposed by Fourier (1827) and Motzkin (1936), is called *Fourier-Motzkin Elimination* (FME)
  - Its complexity is $\mathcal{O}(m^{2^d})$ due to many redundant inequalities

Background

- Projection of polyhedral sets has many applications in computer science
  - scheduling
  - dependence analysis (automatic parallelization)
- The basic algorithm, proposed by Fourier (1827) and Motzkin (1936), is called *Fourier-Motzkin Elimination* (FME)
  - Its complexity is $\mathcal{O}(m^{2^d})$ due to many redundant inequalities
  - Removing intermediate redundant inequalities significantly improves running time and output size

Background

- Projection of polyhedral sets has many applications in computer science
  - scheduling
  - dependence analysis (automatic parallelization)
- The basic algorithm, proposed by Fourier (1827) and Motzkin (1936), is called *Fourier-Motzkin Elimination* (FME)
  - Its complexity is $\mathcal{O}(m^{2^d})$ due to many redundant inequalities
  - Removing intermediate redundant inequalities significantly improves running time and output size
  - Using linear programming (LP) for removing redundant inequalities, complexity drops to $\mathcal{O}(d^2 m^{2d} LP(d, 2^d hd^2 m^d))$ for an input polyhedron in dimension d, with m facets and coefficient height h.

Background

- Projection of polyhedral sets has many applications in computer science
  - scheduling
  - dependence analysis (automatic parallelization)
- The basic algorithm, proposed by Fourier (1827) and Motzkin (1936), is called *Fourier-Motzkin Elimination* (FME)
  - Its complexity is $\mathcal{O}(m^{2^d})$ due to many redundant inequalities
  - Removing intermediate redundant inequalities significantly improves running time and output size
  - Using linear programming (LP) for removing redundant inequalities, complexity drops to $\mathcal{O}(d^2 m^{2d} LP(d, 2^d hd^2 m^d))$ for an input polyhedron in dimension d, with m facets and coefficient height h.
- Chernikov [Ch60] and Kohler [Ko67] proposed procedures for removing redundant inequalities based on linear algebra instead of LP. The current implementation in Maple uses matrix arithmetic [JMT20].

Background

- Projection of polyhedral sets has many applications in computer science
  - scheduling
  - dependence analysis (automatic parallelization)
- The basic algorithm, proposed by Fourier (1827) and Motzkin (1936), is called *Fourier-Motzkin Elimination* (FME)
  - Its complexity is $\mathcal{O}(m^{2^d})$ due to many redundant inequalities
  - Removing intermediate redundant inequalities significantly improves running time and output size
  - Using linear programming (LP) for removing redundant inequalities, complexity drops to $\mathcal{O}(d^2 m^{2d} LP(d, 2^d h d^2 m^d))$ for an input polyhedron in dimension d, with m facets and coefficient height h.
- Chernikov [Ch60] and Kohler [Ko67] proposed procedures for removing redundant inequalities based on linear algebra instead of LP. The current implementation in Maple uses matrix arithmetic [JMT20].
- In [JMXY24] proposed a method using Saturation Matrix.

Definition (H-representation)

A polyhedron P is a set which can be expressed as the intersection of finite number of (closed) half spaces, that is $\{\overrightarrow{x} \in \mathbb{R}^n \mid A\overrightarrow{x} \leq \overrightarrow{b}\}$.

Definition (H-representation)

A polyhedron P is a set which can be expressed as the intersection of finite number of (closed) half spaces, that is $\{\overrightarrow{x} \in \mathbb{R}^n \mid A\overrightarrow{x} \leq \overrightarrow{b}\}$.

Definition (V-representation)

The dual representation of the polyhedron P can be expressed as a combination of rays R (forming the polyhedral cone) and vertices V (forming the polytope), that is
$\{x \in \mathbb{R}^n \mid \mu R + \rho V$ such that $\mu, \rho \geq 0$ and $\mu + \rho = 1\}$.

We will denote it as $\mathcal{VR}(F)$, where F is the H-representation of P.

Definition

Having two inequality: $a_1x_1 + \cdots + a_nx_n \leq d_1$ and $b_1x_1 + \cdots + b_nx_n \leq d_2$ such that $a_1 > 0$ and $b_1 < 0$, we can eliminate x_1 by multiplying the first inequality by $|b_1|$ and the second one by a_1 and add them together. The result of combining these inequalities is:

$$(a_2|b_1| + b_2a_1)x_2 + \cdots + (a_n|b_1| + b_na_1)x_n \leq |b_1|d_1 + a_1d_2.$$

Definition

Having two inequality: $a_1x_1 + \cdots + a_nx_n \leq d_1$ and $b_1x_1 + \cdots + b_nx_n \leq d_2$ such that $a_1 > 0$ and $b_1 < 0$, we can eliminate x_1 by multiplying the first inequality by $|b_1|$ and the second one by a_1 and add them together. The result of combining these inequalities is:

$$(a_2|b_1| + b_2a_1)x_2 + \cdots + (a_n|b_1| + b_na_1)x_n \leq |b_1|d_1 + a_1d_2.$$

Definition

Having a linear inequality system S with m inequalities and n variables of the form $a_{i1}x_1 + \cdots + a_{in}x_n \leq d_i$, We can partition the inequalities in three groups with respect to x_1:

- A^+ set of inequalities with positive x_1 coefficient.
- A^- set of inequalities with negative x_1 coefficient.
- A^0 set of inequalities with zero x_1 coefficient.

Idea

Theorem

Let A' be the union of combination of all inequalities in A^+ with all inequalities in A^- and inequalities in A^0 such that A' does not have x_1. Then,

$$(x_2, \cdots, x_n) \in Sol(A') \Longleftrightarrow \exists x_1 \; (x_1, x_2, \cdots, x_n) \in Sol(A)$$

where $Sol(A)$ is a set of real points which satisfies all inequalities in A.

Idea

Theorem

Let A' be the union of combination of all inequalities in A^+ with all inequalities in A^- and inequalities in A^0 such that A' does not have x_1. Then,

$$(x_2, \cdots, x_n) \in Sol(A') \Longleftrightarrow \exists x_1 \ (x_1, x_2, \cdots, x_n) \in Sol(A)$$

where $Sol(A)$ is a set of real points which satisfies all inequalities in A.

FME Algorithm

Western
UNIVERSITY · CANADA

Idea

Theorem

Let A' be the union of combination of all inequalities in A^+ with all inequalities in A^- and inequalities in A^0 such that A' does not have x_1. Then,

$$(x_2, \cdots, x_n) \in Sol(A') \Longleftrightarrow \exists x_1 \ (x_1, x_2, \cdots, x_n) \in Sol(A)$$

where $Sol(A)$ is a set of real points which satisfies all inequalities in A.

FME Algorithm

- Select variables one after another.

Idea

Theorem

Let A' be the union of combination of all inequalities in A^+ with all inequalities in A^- and inequalities in A^0 such that A' does not have x_1. Then,

$$(x_2, \cdots, x_n) \in Sol(A') \iff \exists x_1 \ (x_1, x_2, \cdots, x_n) \in Sol(A)$$

where $Sol(A)$ is a set of real points which satisfies all inequalities in A.

FME Algorithm

- Select variables one after another.
- Partition A^+, A^- and A^0 with respect to the variable.

Western
UNIVERSITY · CANADA

Idea

Theorem

Let A' be the union of combination of all inequalities in A^+ with all inequalities in A^- and inequalities in A^0 such that A' does not have x_1. Then,

$$(x_2, \cdots, x_n) \in Sol(A') \Longleftrightarrow \exists x_1 \ (x_1, x_2, \cdots, x_n) \in Sol(A)$$

where $Sol(A)$ is a set of real points which satisfies all inequalities in A.

FME Algorithm

- Select variables one after another.
- Partition A^+, A^- and A^0 with respect to the variable.
- Combine inequalities in A^+ and A^- and form the resulting union.

Complexity

Eliminating variable x_1 from a system with m inequalities with d variables:

Complexity

Eliminating variable x_1 from a system with m inequalities with d variables:

- Partitioning inequalities with positive and negative x_1 coefficient is $\mathcal{O}(m)$.

Complexity

Eliminating variable x_1 from a system with m inequalities with d variables:

- Partitioning inequalities with positive and negative x_1 coefficient is $\mathcal{O}(m)$.
- In the worst case, the system has $\frac{m}{2}$ positive coefficients and $\frac{m}{2}$ negative coefficients.

Complexity

Eliminating variable x_1 from a system with m inequalities with d variables:

- Partitioning inequalities with positive and negative x_1 coefficient is $\mathcal{O}(m)$.
- In the worst case, the system has $\frac{m}{2}$ positive coefficients and $\frac{m}{2}$ negative coefficients.
- The final system would have $\mathcal{O}(\frac{m}{2})^2$ inequalities.

Complexity

Eliminating variable x_1 from a system with m inequalities with d variables:

- Partitioning inequalities with positive and negative x_1 coefficient is $\mathcal{O}(m)$.
- In the worst case, the system has $\frac{m}{2}$ positive coefficients and $\frac{m}{2}$ negative coefficients.
- The final system would have $\mathcal{O}(\frac{m}{2})^2$ inequalities.
- Hence, the complexity of eliminating d variables is $\mathcal{O}(m^{2^d})$.

Complexity

Eliminating variable x_1 from a system with m inequalities with d variables:

- Partitioning inequalities with positive and negative x_1 coefficient is $\mathcal{O}(m)$.
- In the worst case, the system has $\frac{m}{2}$ positive coefficients and $\frac{m}{2}$ negative coefficients.
- The final system would have $\mathcal{O}(\frac{m}{2})^2$ inequalities.
- Hence, the complexity of eliminating d variables is $\mathcal{O}(m^{2^d})$.

Improve Complexity

FME's complexity is double exponential. Most of the inequalities generated by FME algorithm are redundant. Detecting these redundant inequalities and removing them can significantly improve algorithm's complexity.

Definition

For $F : \{A\overrightarrow{x} \leq \overrightarrow{b}\}$ a consistent system of linear inequalities, P be the polyhedron represented by F, an inequality $\ell : \overrightarrow{a}^t\overrightarrow{x} \leq \overrightarrow{b}$ of F is called,

Definition

For $F : \{A\overrightarrow{x} \leq \overrightarrow{b}\}$ a consistent system of linear inequalities, P be the polyhedron represented by F, an inequality $\ell : \overrightarrow{a}^t\overrightarrow{x} \leq \overrightarrow{b}$ of F is called,

- Redundant in F, if $F \setminus \{\overrightarrow{a}^t\overrightarrow{x} \leq \overrightarrow{b}\}$ represents the same polyhedron P.

Definition

For $F : \{A\overrightarrow{x} \leq \overrightarrow{b}\}$ a consistent system of linear inequalities, P be the polyhedron represented by F, an inequality $\ell : \overrightarrow{a}^t\overrightarrow{x} \leq \overrightarrow{b}$ of F is called,

- Redundant in F, if $F \setminus \{\overrightarrow{a}^t\overrightarrow{x} \leq \overrightarrow{b}\}$ represents the same polyhedron P.
- Otherwise, it is irredundant.

Definition

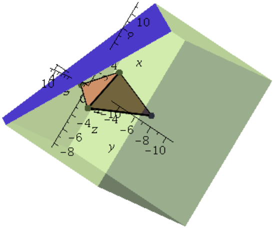
For $F : \{A\overrightarrow{x} \leq \overrightarrow{b}\}$ a consistent system of linear inequalities, P be the polyhedron represented by F, an inequality $\ell : \overrightarrow{a}^t\overrightarrow{x} \leq \overrightarrow{b}$ of F is called,

- Redundant in F, if $F \setminus \{\overrightarrow{a}^t\overrightarrow{x} \leq \overrightarrow{b}\}$ represents the same polyhedron P.
- Otherwise, it is irredundant.
- Strongly redundant if $\overrightarrow{a}^t\overrightarrow{x} < \overrightarrow{b}$ for all $\overrightarrow{x} \in P$.

Definition

For $F : \{A\overrightarrow{x} \leq \overrightarrow{b}\}$ a consistent system of linear inequalities, P be the polyhedron represented by F, an inequality $\ell : \overrightarrow{a}^t\overrightarrow{x} \leq \overrightarrow{b}$ of F is called,

- Redundant in F, if $F \setminus \{\overrightarrow{a}^t\overrightarrow{x} \leq \overrightarrow{b}\}$ represents the same polyhedron P.
- Otherwise, it is irredundant.
- Strongly redundant if $\overrightarrow{a}^t\overrightarrow{x} < \overrightarrow{b}$ for all $\overrightarrow{x} \in P$.
- Weakly redundant if it is redundant and $\overrightarrow{a}^t\overrightarrow{x} = \overrightarrow{b}$ holds for some $\overrightarrow{x} \in P$.



Using LP

Removing all these redundancies is equivalent to giving a minimal representation of the projected polyhedron.

Using LP

Removing all these redundancies is equivalent to giving a minimal representation of the projected polyhedron.

- Leonid Khachiyan explained in [Kh09] how linear programming (LP) could be used to remove all redundant inequalities, thus reducing the cost of Fourier-Motzkin elimination to a singly exponential time.

Using LP

Removing all these redundancies is equivalent to giving a minimal representation of the projected polyhedron.

- Leonid Khachiyan explained in [Kh09] how linear programming (LP) could be used to remove all redundant inequalities, thus reducing the cost of Fourier-Motzkin elimination to a singly exponential time.
- The earliest FME algorithms uses LP to detect redundancy.

Using LP

Removing all these redundancies is equivalent to giving a minimal representation of the projected polyhedron.

- Leonid Khachiyan explained in [Kh09] how linear programming (LP) could be used to remove all redundant inequalities, thus reducing the cost of Fourier-Motzkin elimination to a singly exponential time.
- The earliest FME algorithms uses LP to detect redundancy.
  - It needs to run the Simplex algorithm [Sc86] for all inequalities.

Using LP

Removing all these redundancies is equivalent to giving a minimal representation of the projected polyhedron.

- Leonid Khachiyan explained in [Kh09] how linear programming (LP) could be used to remove all redundant inequalities, thus reducing the cost of Fourier-Motzkin elimination to a singly exponential time.
- The earliest FME algorithms uses LP to detect redundancy.
  - It needs to run the Simplex algorithm [Sc86] for all inequalities.
  - It is not practical for large cases.

Using LP

Removing all these redundancies is equivalent to giving a minimal representation of the projected polyhedron.

- Leonid Khachiyan explained in [Kh09] how linear programming (LP) could be used to remove all redundant inequalities, thus reducing the cost of Fourier-Motzkin elimination to a singly exponential time.
- The earliest FME algorithms uses LP to detect redundancy.
  - It needs to run the Simplex algorithm [Sc86] for all inequalities.
  - It is not practical for large cases.
- It neither has a good theoretical complexity, nor is it effective in practice because of its dependence on LP solvers.

Using LP

Removing all these redundancies is equivalent to giving a minimal representation of the projected polyhedron.

- Leonid Khachiyan explained in [Kh09] how linear programming (LP) could be used to remove all redundant inequalities, thus reducing the cost of Fourier-Motzkin elimination to a singly exponential time.
- The earliest FME algorithms uses LP to detect redundancy.
  - It needs to run the Simplex algorithm [Sc86] for all inequalities.
  - It is not practical for large cases.
- It neither has a good theoretical complexity, nor is it effective in practice because of its dependence on LP solvers.
- Since, FME are essentially an adaptation of GE, it is desirable to achieve the redundancy via linear algebra instead of using LP.

Definition

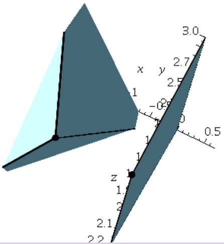
The projection $proj(\cdot, I)$ acts on the V -representation of a polyhedron, and returns a set of vertices and rays representing the projected polyhedron, which can be obtained by simply erasing the coordinates corresponding to the variables in I.

Definition

The projection $proj(\cdot, I)$ acts on the V -representation of a polyhedron, and returns a set of vertices and rays representing the projected polyhedron, which can be obtained by simply erasing the coordinates corresponding to the variables in I.

Example

Consider the polyhedron defined by $\{x + 2\,y - z \leq 2, 2\,x - 3\,y + 6\,z \leq 2, -2\,x + 3\,y + 4\,z \leq 20\}$. Its projection on $[y, z]$ is the polyhedron represented by $\{z \leq \frac{11}{5},\ y + \frac{2}{7}z \leq \frac{24}{7}\}$.



- In [Ba98], Balas observed that if the matrix B is invertible, then we can find a cone such that its extreme rays are in one-to-one correspondence with the facets of the projection of the polyhedron.

- In [Ba98], Balas observed that if the matrix B is invertible, then we can find a cone such that its extreme rays are in one-to-one correspondence with the facets of the projection of the polyhedron.
- Using this fact, Balas developed an algorithm to find all redundant inequalities.

- In [Ba98], Balas observed that if the matrix B is invertible, then we can find a cone such that its extreme rays are in one-to-one correspondence with the facets of the projection of the polyhedron.
- Using this fact, Balas developed an algorithm to find all redundant inequalities.
- Rui-Juan Jing, Marc Moreno-Maza, and Delaram Talaashrafi in [JMT20] combined an improved version of Bala's algorithm with Kohler's algorithm to detect all all redundant inequalities.

- In [Ba98], Balas observed that if the matrix B is invertible, then we can find a cone such that its extreme rays are in one-to-one correspondence with the facets of the projection of the polyhedron.
- Using this fact, Balas developed an algorithm to find all redundant inequalities.
- Rui-Juan Jing, Marc Moreno-Maza, and Delaram Talaashrafi in [JMT20] combined an improved version of Bala's algorithm with Kohler's algorithm to detect all all redundant inequalities.
  - First construct the initial test cone from the input polyhedron.

- In [Ba98], Balas observed that if the matrix B is invertible, then we can find a cone such that its extreme rays are in one-to-one correspondence with the facets of the projection of the polyhedron.
- Using this fact, Balas developed an algorithm to find all redundant inequalities.
- Rui-Juan Jing, Marc Moreno-Maza, and Delaram Talaashrafi in [JMT20] combined an improved version of Bala's algorithm with Kohler's algorithm to detect all all redundant inequalities.
  - First construct the initial test cone from the input polyhedron.
  - This cone can be used to find the "polar cone" of the polyhedron after projection.

- In [Ba98], Balas observed that if the matrix B is invertible, then we can find a cone such that its extreme rays are in one-to-one correspondence with the facets of the projection of the polyhedron.
- Using this fact, Balas developed an algorithm to find all redundant inequalities.
- Rui-Juan Jing, Marc Moreno-Maza, and Delaram Talaashrafi in [JMT20] combined an improved version of Bala's algorithm with Kohler's algorithm to detect all all redundant inequalities.
  - First construct the initial test cone from the input polyhedron.
  - This cone can be used to find the "polar cone" of the polyhedron after projection.
  - Redundant inequalities can be detected using extreme rays of the polar cone.

- In [Ba98], Balas observed that if the matrix B is invertible, then we can find a cone such that its extreme rays are in one-to-one correspondence with the facets of the projection of the polyhedron.
- Using this fact, Balas developed an algorithm to find all redundant inequalities.
- Rui-Juan Jing, Marc Moreno-Maza, and Delaram Talaashrafi in [JMT20] combined an improved version of Bala's algorithm with Kohler's algorithm to detect all all redundant inequalities.
  - First construct the initial test cone from the input polyhedron.
  - This cone can be used to find the "polar cone" of the polyhedron after projection.
  - Redundant inequalities can be detected using extreme rays of the polar cone.
- For a non-empty, full-dimensional, and pointed polyhedron $P \subset \mathbb{Q}^n$ as input, given by a system of m linear inequalities of height h, the complexity is $\mathcal{O}(m^{\frac{5n}{2}} n^{\omega+1+\epsilon} h^{1+\epsilon})$ bit operations, for any $\epsilon > 0$, where ω is the exponent of matrix multiplication.

New Algorithm in Maple

Saturation Matrix

Definition

The saturation matrix of a system of linear inequalities F is the Boolean matrix $\mathrm{satM}(F) \in \mathbb{Q}^{m \times k}$, whose (i, j)-th element is equal to 1, if the j-th element of $\mathcal{VR}(F)$ saturates the i-th inequality of F, 0 otherwise.

Saturation Matrix

Definition

The saturation matrix of a system of linear inequalities F is the Boolean matrix $\text{satM}(F) \in \mathbb{Q}^{m \times k}$, whose (i, j)-th element is equal to 1, if the j-th element of $\mathcal{VR}(F)$ saturates the i-th inequality of F, 0 otherwise.

Example

Consider the system F with the set $\mathcal{VR}(F)$ and the saturation matrix $\text{satM}(F)$ given below.

F	$\mathcal{VR}(F)$
$\ell_1 : x + y \leq 1$	$\vec{v}_1 : (0, 1)$
$\ell_2 : -x - y \leq 1$	$\vec{v}_2 : (1, 0)$
$\ell_3 : x - y \leq 1$	$\vec{v}_3 : (-1, 0)$
$\ell_4 : -x + y \leq 1$	$\vec{v}_4 : (0, -1)$

$\text{satM}(F)$

	$\vec{v}_1$	$\vec{v}_2$	$\vec{v}_3$	$\vec{v}_4$
ℓ_1	1	1	0	0
ℓ_2	0	0	1	1
ℓ_3	0	1	0	1
ℓ_4	1	0	1	0

Notation:

Let ℓ in F be an inequality such that, we denote

Notation:

Let ℓ in F be an inequality such that, we denote

- $\mathcal{S}^{\mathcal{VR}}(\ell)$, the collection of vertices and rays in $\mathcal{VR}(F)$ saturated by the hyperplane $\mathcal{H}_\ell$ of ℓ.

Notation:

Let ℓ in F be an inequality such that, we denote

- $\mathcal{S}^{\mathcal{VR}}(\ell)$, the collection of vertices and rays in $\mathcal{VR}(F)$ saturated by the hyperplane $\mathcal{H}_\ell$ of ℓ.
- $\mathcal{S}^{\mathcal{H}}(\overrightarrow{v})$, the collections of inequalities in F that $\overrightarrow{v}$ saturates.

Notation:

Let ℓ in F be an inequality such that, we denote

- $\mathcal{S}^{\mathcal{VR}}(\ell)$, the collection of vertices and rays in $\mathcal{VR}(F)$ saturated by the hyperplane $\mathcal{H}_\ell$ of ℓ.
- $\mathcal{S}^{\mathcal{H}}(\overrightarrow{v})$, the collections of inequalities in F that $\overrightarrow{v}$ saturates.

Observation:

The composition of the above two can be denotes by $\mathcal{S}^{\mathcal{H}}(\mathcal{S}^{\mathcal{VR}}(\ell))$, which is the collection of inequalities saturated by all the vertices or rays saturating ℓ.

Theorem (Redundancy tests)

Let ℓ be an inequality in F. The following properties hold:

Theorem (Redundancy tests)

Let ℓ be an inequality in F. The following properties hold:

1. *The inequality ℓ is strongly redundant in F if and only if $\mathcal{S}^{\mathcal{VR}}(\ell)$ is empty.*

Theorem (Redundancy tests)

Let ℓ be an inequality in F. The following properties hold:

1. *The inequality ℓ is strongly redundant in F if and only if $\mathcal{S}^{\mathcal{VR}}(\ell)$ is empty.*
2. *If $\mathcal{S}^{\mathcal{VR}}(\ell)$ is non-empty and its cardinality is less than n, then the inequality ℓ is weakly redundant in F.*

Theorem (Redundancy tests)

Let ℓ be an inequality in F. The following properties hold:

1. *The inequality ℓ is strongly redundant in F if and only if $\mathcal{S}^{\mathcal{VR}}(\ell)$ is empty.*
2. *If $\mathcal{S}^{\mathcal{VR}}(\ell)$ is non-empty and its cardinality is less than n, then the inequality ℓ is weakly redundant in F.*
3. *The inequality ℓ is weakly redundant in F if and only if the set $\mathcal{S}^{\mathcal{H}}(\mathcal{S}^{\mathcal{VR}}(\ell)) \setminus \{\ell\}$ is not empty.*

Theorem (Redundancy tests)

Let ℓ be an inequality in F. The following properties hold:

1. *The inequality ℓ is strongly redundant in F if and only if $\mathcal{S}^{\mathcal{VR}}(\ell)$ is empty.*
2. *If $\mathcal{S}^{\mathcal{VR}}(\ell)$ is non-empty and its cardinality is less than n, then the inequality ℓ is weakly redundant in F.*
3. *The inequality ℓ is weakly redundant in F if and only if the set $\mathcal{S}^{\mathcal{H}}(\mathcal{S}^{\mathcal{VR}}(\ell)) \setminus \{\ell\}$ is not empty.*

Corollary (Criteria using the saturation matrix)

1. *If* $\mathrm{satM}(F)[\ell]$ *contains zero elements, then ℓ is strongly redundant.*

Theorem (Redundancy tests)

Let ℓ be an inequality in F. The following properties hold:

1. *The inequality ℓ is strongly redundant in F if and only if $\mathcal{S}^{\mathcal{VR}}(\ell)$ is empty.*
2. *If $\mathcal{S}^{\mathcal{VR}}(\ell)$ is non-empty and its cardinality is less than n, then the inequality ℓ is weakly redundant in F.*
3. *The inequality ℓ is weakly redundant in F if and only if the set $\mathcal{S}^{\mathcal{H}}(\mathcal{S}^{\mathcal{VR}}(\ell)) \setminus \{\ell\}$ is not empty.*

Corollary (Criteria using the saturation matrix)

1. *If $\mathrm{satM}(F)[\ell]$ contains zero elements, then ℓ is strongly redundant.*
2. *If the number of nonzero elements of $\mathrm{satM}(F)[\ell]$ is positive and less than the dimension n, then ℓ is weakly redundant.*

Theorem (Redundancy tests)

Let ℓ be an inequality in F. The following properties hold:

1. *The inequality ℓ is strongly redundant in F if and only if $\mathcal{S}^{\mathcal{VR}}(\ell)$ is empty.*
2. *If $\mathcal{S}^{\mathcal{VR}}(\ell)$ is non-empty and its cardinality is less than n, then the inequality ℓ is weakly redundant in F.*
3. *The inequality ℓ is weakly redundant in F if and only if the set $\mathcal{S}^{\mathcal{H}}(\mathcal{S}^{\mathcal{VR}}(\ell)) \setminus \{\ell\}$ is not empty.*

Corollary (Criteria using the saturation matrix)

1. *If $\mathrm{satM}(F)[\ell]$ contains zero elements, then ℓ is strongly redundant.*
2. *If the number of nonzero elements of $\mathrm{satM}(F)[\ell]$ is positive and less than the dimension n, then ℓ is weakly redundant.*
3. *If $\mathrm{satM}(F)[\ell]$ is contained in $\mathrm{satM}(F)[\ell_1]$ for some $\ell_1 \in F \setminus \{\ell\}$, then ℓ is weakly redundant.*

From the saturation matrix $\mathrm{satM}(F) = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$, it is easy to obtain the following identities:

$$\begin{aligned} \mathcal{S}^{\mathcal{VR}}(\ell_1) &= \{\overrightarrow{v}_1, \overrightarrow{v}_2\} & \mathcal{S}^{\mathcal{H}}(\overrightarrow{v}_1) &= \{\ell_1, \ell_4\} \\ \mathcal{S}^{\mathcal{VR}}(\ell_2) &= \{\overrightarrow{v}_3, \overrightarrow{v}_4\} & \mathcal{S}^{\mathcal{H}}(\overrightarrow{v}_2) &= \{\ell_1, \ell_3\} \\ \mathcal{S}^{\mathcal{VR}}(\ell_3) &= \{\overrightarrow{v}_2, \overrightarrow{v}_4\} & \mathcal{S}^{\mathcal{H}}(\overrightarrow{v}_3) &= \{\ell_2, \ell_4\} \\ \mathcal{S}^{\mathcal{VR}}(\ell_4) &= \{\overrightarrow{v}_1, \overrightarrow{v}_3\} & \mathcal{S}^{\mathcal{H}}(\overrightarrow{v}_4) &= \{\ell_2, \ell_3\} \end{aligned}$$

Algorithm 1 Check Redundancy

Require: 1. the inequality system F with m inequalities;
2. the saturation matrix satM(F).

Ensure: the minimal irredundant system F_{ired}.

$F_{\text{irred}} := \{ \}$ and $\text{satM}_{\text{irred}} := [\ ]$.
for i from 1 to m **do**
  Let *Redundant* := *False*.
  if the number of nonzero elements in satM[i] is less than n **then**
    next. /* Corollary 1 and 2*/
  end if
  for j from 1 to $i - 1$ **do**
    if satM[i] = satM[i]&satM[j] **then**
      Redundant := True.
      break. /* Corollary 3 */
    end if
  end for
  if not Redundant **then**
    $F_{\text{irred}} := F_{\text{irred}} \cup \{f_i\}$ and append satM[i] to $\text{satM}_{\text{irred}}$.
  end if
end for

Updating the Saturation Matrix

The saturation matrix is traversed both row-wise (to compute bit-wise AND) and column-wise (to compute bit-wise OR).

Updating the Saturation Matrix

The saturation matrix is traversed both row-wise (to compute bit-wise AND) and column-wise (to compute bit-wise OR).

- $\mathcal{S}^{\mathcal{VR}}(\ell)$ is obtained by the bit-wise AND of the Boolean vectors satM$(F)[\ell_{pos}]$ and satM$(F)[\ell_{neg}]$.

Updating the Saturation Matrix

The saturation matrix is traversed both row-wise (to compute bit-wise AND) and column-wise (to compute bit-wise OR).

- $\mathcal{S}^{\mathcal{VR}}(\ell)$ is obtained by the bit-wise AND of the Boolean vectors $\mathrm{satM}(F)[\ell_{pos}]$ and $\mathrm{satM}(F)[\ell_{neg}]$.
- Merging is to form the saturation matrix corresponding to the subspace of the remaining coordinates.

Updating the Saturation Matrix

The saturation matrix is traversed both row-wise (to compute bit-wise AND) and column-wise (to compute bit-wise OR).

- $\mathcal{S}^{\mathcal{VR}}(\ell)$ is obtained by the bit-wise AND of the Boolean vectors $\mathrm{satM}(F)[\ell_{pos}]$ and $\mathrm{satM}(F)[\ell_{neg}]$.
- Merging is to form the saturation matrix corresponding to the subspace of the remaining coordinates.
- For any vertices $\{v_1, \ldots, v_e\}$ so that $\mathrm{proj}(v_1, \{x\}) = \cdots = \mathrm{proj}(v_e, \{x\})$ holds, we merge in $\mathrm{satM}(F)$ the saturation information contained by these columns indexed by $v_1, \ldots, v_e$ as follows:

Updating the Saturation Matrix

The saturation matrix is traversed both row-wise (to compute bit-wise AND) and column-wise (to compute bit-wise OR).

- $\mathcal{S}^{\mathcal{VR}}(\ell)$ is obtained by the bit-wise AND of the Boolean vectors $\mathrm{satM}(F)[\ell_{pos}]$ and $\mathrm{satM}(F)[\ell_{neg}]$.
- Merging is to form the saturation matrix corresponding to the subspace of the remaining coordinates.
- For any vertices $\{v_1, \ldots, v_e\}$ so that $\mathrm{proj}(v_1, \{x\}) = \cdots = \mathrm{proj}(v_e, \{x\})$ holds, we merge in $\mathrm{satM}(F)$ the saturation information contained by these columns indexed by $v_1, \ldots, v_e$ as follows:
    - we compute the bit-wise OR of the columns (regarded as bit-vectors) of $\mathrm{satM}(F)$ indexed by $v_1, \ldots, v_e$

Updating the Saturation Matrix

The saturation matrix is traversed both row-wise (to compute bit-wise AND) and column-wise (to compute bit-wise OR).

- $\mathcal{S}^{\mathcal{VR}}(\ell)$ is obtained by the bit-wise AND of the Boolean vectors $\mathrm{satM}(F)[\ell_{pos}]$ and $\mathrm{satM}(F)[\ell_{neg}]$.
- Merging is to form the saturation matrix corresponding to the subspace of the remaining coordinates.
- For any vertices $\{v_1, \ldots, v_e\}$ so that $\mathrm{proj}(v_1, \{x\}) = \cdots = \mathrm{proj}(v_e, \{x\})$ holds, we merge in $\mathrm{satM}(F)$ the saturation information contained by these columns indexed by $v_1, \ldots, v_e$ as follows:
    - we compute the bit-wise OR of the columns (regarded as bit-vectors) of $\mathrm{satM}(F)$ indexed by $v_1, \ldots, v_e$
    - we replace the columns indexed by $v_1, \ldots, v_e$ by this bit-wise OR.

Example

- $\mathcal{S}^{\mathcal{VR}}(\ell_1) = \{\overrightarrow{v}_1, \overrightarrow{v}_2\}$ and $\mathcal{S}^{\mathcal{VR}}(\ell_4) = \{\overrightarrow{v}_1, \overrightarrow{v}_3\}$.

Example

- $\mathcal{S}^{\mathcal{VR}}(\ell_1) = \{\overrightarrow{v}_1, \overrightarrow{v}_2\}$ and $\mathcal{S}^{\mathcal{VR}}(\ell_4) = \{\overrightarrow{v}_1, \overrightarrow{v}_3\}$.
- Then, $\mathcal{S}^{\mathcal{VR}}(\ell_1) \cap \mathcal{S}^{\mathcal{VR}}(\ell_4) = \{\overrightarrow{v}_1\}$ and $\mathrm{proj}(\mathcal{S}^{\mathcal{VR}}(\ell_1) \cap \mathcal{S}^{\mathcal{VR}}(\ell_4), \{x\}) = \mathrm{proj}(\{\overrightarrow{v}_1\}, \{x\}) = \{(1)\}$.

Example

- $\mathcal{S}^{\mathcal{VR}}(\ell_1) = \{\overrightarrow{v}_1, \overrightarrow{v}_2\}$ and $\mathcal{S}^{\mathcal{VR}}(\ell_4) = \{\overrightarrow{v}_1, \overrightarrow{v}_3\}$.
- Then, $\mathcal{S}^{\mathcal{VR}}(\ell_1) \cap \mathcal{S}^{\mathcal{VR}}(\ell_4) = \{\overrightarrow{v}_1\}$ and $\mathrm{proj}(\mathcal{S}^{\mathcal{VR}}(\ell_1) \cap \mathcal{S}^{\mathcal{VR}}(\ell_4), \{x\}) = \mathrm{proj}(\{\overrightarrow{v}_1\}, \{x\}) = \{(1)\}$.
- $\mathrm{satM}(F)[\ell_1] = (1, 1, 0, 0), \mathrm{satM}(F)[\ell_4] = (1, 0, 1, 0)$.

Example

- $\mathcal{S}^{\mathcal{VR}}(\ell_1) = \{\overrightarrow{v}_1, \overrightarrow{v}_2\}$ and $\mathcal{S}^{\mathcal{VR}}(\ell_4) = \{\overrightarrow{v}_1, \overrightarrow{v}_3\}$.
- Then, $\mathcal{S}^{\mathcal{VR}}(\ell_1) \cap \mathcal{S}^{\mathcal{VR}}(\ell_4) = \{\overrightarrow{v}_1\}$ and $\text{proj}(\mathcal{S}^{\mathcal{VR}}(\ell_1) \cap \mathcal{S}^{\mathcal{VR}}(\ell_4), \{x\}) = \text{proj}(\{\overrightarrow{v}_1\}, \{x\}) = \{(1)\}$.
- $\text{satM}(F)[\ell_1] = (1, 1, 0, 0), \text{satM}(F)[\ell_4] = (1, 0, 1, 0)$.
- So, we have $\text{satM}(F)[\ell_1] \& \text{satM}(F)[\ell_4] = (1, 0, 0, 0)$.

Example

- $\mathcal{S}^{\mathcal{VR}}(\ell_1) = \{\overrightarrow{v}_1, \overrightarrow{v}_2\}$ and $\mathcal{S}^{\mathcal{VR}}(\ell_4) = \{\overrightarrow{v}_1, \overrightarrow{v}_3\}$.
- Then, $\mathcal{S}^{\mathcal{VR}}(\ell_1) \cap \mathcal{S}^{\mathcal{VR}}(\ell_4) = \{\overrightarrow{v}_1\}$ and $\mathrm{proj}(\mathcal{S}^{\mathcal{VR}}(\ell_1) \cap \mathcal{S}^{\mathcal{VR}}(\ell_4), \{x\}) = \mathrm{proj}(\{\overrightarrow{v}_1\}, \{x\}) = \{(1)\}$.
- $\mathrm{satM}(F)[\ell_1] = (1, 1, 0, 0), \mathrm{satM}(F)[\ell_4] = (1, 0, 1, 0)$.
- So, we have $\mathrm{satM}(F)[\ell_1] \& \mathrm{satM}(F)[\ell_4] = (1, 0, 0, 0)$.
- Note that $\mathrm{proj}(\{v_2\}, \{x\}) = \mathrm{proj}(\{v_3\}, \{x\}) = (0)$, we should merge the information on the second and third position in $\mathrm{satM}(F)[\ell_1] \& \mathrm{satM}(F)[\ell_4]$.

Example

- $\mathcal{S}^{\mathcal{VR}}(\ell_1) = \{\overrightarrow{v}_1, \overrightarrow{v}_2\}$ and $\mathcal{S}^{\mathcal{VR}}(\ell_4) = \{\overrightarrow{v}_1, \overrightarrow{v}_3\}$.
- Then, $\mathcal{S}^{\mathcal{VR}}(\ell_1) \cap \mathcal{S}^{\mathcal{VR}}(\ell_4) = \{\overrightarrow{v}_1\}$ and $\text{proj}(\mathcal{S}^{\mathcal{VR}}(\ell_1) \cap \mathcal{S}^{\mathcal{VR}}(\ell_4), \{x\}) = \text{proj}(\{\overrightarrow{v}_1\}, \{x\}) = \{(1)\}$.
- $\text{satM}(F)[\ell_1] = (1, 1, 0, 0), \text{satM}(F)[\ell_4] = (1, 0, 1, 0)$.
- So, we have $\text{satM}(F)[\ell_1] \& \text{satM}(F)[\ell_4] = (1, 0, 0, 0)$.
- Note that $\text{proj}(\{v_2\}, \{x\}) = \text{proj}(\{v_3\}, \{x\}) = (0)$, we should merge the information on the second and third position in $\text{satM}(F)[\ell_1] \& \text{satM}(F)[\ell_4]$.
- $\text{Merge}(\text{satM}(F)[\ell_1] \& \text{satM}(F)[\ell_4]) = (1, 0, 0)$.
- $\text{proj}(\{\ell_1, \ell_4\}, \{x\}) = \{x \leq 1\}$.

Example

- $\mathcal{S}^{\mathcal{VR}}(\ell_1) = \{\overrightarrow{v}_1, \overrightarrow{v}_2\}$ and $\mathcal{S}^{\mathcal{VR}}(\ell_4) = \{\overrightarrow{v}_1, \overrightarrow{v}_3\}$.
- Then, $\mathcal{S}^{\mathcal{VR}}(\ell_1) \cap \mathcal{S}^{\mathcal{VR}}(\ell_4) = \{\overrightarrow{v}_1\}$ and $\mathrm{proj}(\mathcal{S}^{\mathcal{VR}}(\ell_1) \cap \mathcal{S}^{\mathcal{VR}}(\ell_4), \{x\}) = \mathrm{proj}(\{\overrightarrow{v}_1\}, \{x\}) = \{(1)\}$.
- $\mathrm{satM}(F)[\ell_1] = (1, 1, 0, 0), \mathrm{satM}(F)[\ell_4] = (1, 0, 1, 0)$.
- So, we have $\mathrm{satM}(F)[\ell_1] \& \mathrm{satM}(F)[\ell_4] = (1, 0, 0, 0)$.
- Note that $\mathrm{proj}(\{v_2\}, \{x\}) = \mathrm{proj}(\{v_3\}, \{x\}) = (0)$, we should merge the information on the second and third position in $\mathrm{satM}(F)[\ell_1] \& \mathrm{satM}(F)[\ell_4]$.
- $\mathrm{Merge}(\mathrm{satM}(F)[\ell_1] \& \mathrm{satM}(F)[\ell_4]) = (1, 0, 0)$.
- $\mathrm{proj}(\{\ell_1, \ell_4\}, \{x\}) = \{x \leq 1\}$.
- Among the four vertices, only $\mathrm{proj}(\overrightarrow{v}_1, \{x\})$ saturates $\mathrm{proj}(\{\ell_1, \ell_4\}, \{x\})$.

Example

- $\mathcal{S}^{\mathcal{VR}}(\ell_1) = \{\overrightarrow{v}_1, \overrightarrow{v}_2\}$ and $\mathcal{S}^{\mathcal{VR}}(\ell_4) = \{\overrightarrow{v}_1, \overrightarrow{v}_3\}$.
- Then, $\mathcal{S}^{\mathcal{VR}}(\ell_1) \cap \mathcal{S}^{\mathcal{VR}}(\ell_4) = \{\overrightarrow{v}_1\}$ and $\text{proj}(\mathcal{S}^{\mathcal{VR}}(\ell_1) \cap \mathcal{S}^{\mathcal{VR}}(\ell_4), \{x\}) = \text{proj}(\{\overrightarrow{v}_1\}, \{x\}) = \{(1)\}$.
- $\text{satM}(F)[\ell_1] = (1, 1, 0, 0), \text{satM}(F)[\ell_4] = (1, 0, 1, 0)$.
- So, we have $\text{satM}(F)[\ell_1]\&\text{satM}(F)[\ell_4] = (1, 0, 0, 0)$.
- Note that $\text{proj}(\{v_2\}, \{x\}) = \text{proj}(\{v_3\}, \{x\}) = (0)$, we should merge the information on the second and third position in $\text{satM}(F)[\ell_1]\&\text{satM}(F)[\ell_4]$.
- $\text{Merge}(\text{satM}(F)[\ell_1]\&\text{satM}(F)[\ell_4]) = (1, 0, 0)$.
- $\text{proj}(\{\ell_1, \ell_4\}, \{x\}) = \{x \leq 1\}$.
- Among the four vertices, only $\text{proj}(\overrightarrow{v}_1, \{x\})$ saturates $\text{proj}(\{\ell_1, \ell_4\}, \{x\})$.

With the techniques of updating the saturation matrix, we provide Algorithm to compute the minimal projected representation of a polyhedron.

Algorithm 2 Minimal projected representation

Require: 1. an inequality system F;
2. a variable order $x_1 > x_2 > \ldots > x_d$.

Ensure: the minimal projected representation *res* of F.

Compute the V-representation V of F;
Set $res :=$ table().
Sort the elements in V w.r.t. the reverse lexico order.
$F := \mathrm{CheckRedundancy}(F)$.
$res[x_1] := F^{x_1}$.
for i from 1 to $n-1$ **do**
  $(F^p, F^n, F^0) :=$ partition(F).
  Let $V_{new} :=$ proj$(V, \{x_i\})$.
  Let $F_{new} := F^0$.
  for each $f_p \in F^p$ and $f_n \in F^n$ **do**
    Append proj$((f_p, f_n), \{x_i\})$ to F_{new}
  end for
  $F := \mathrm{CheckRedundancy}(F_{new})$.
  $V := V_{new}$, $res[x_{i+1}] := F^{x_{i+1}}$.
end for
return *res*.

Algorithm 3 Minimal projected representation

Require: 1. an inequality system F;
2. a variable order $x_1 > x_2 > \ldots > x_d$.

Ensure: the minimal projected representation *res* of F.

Compute the V-representation V of F;
Set $res := \text{table}()$.
Sort the elements in V w.r.t. the reverse lexico order.
Compute the saturation matrix satM**.**
F, satM $:= \mathrm{CheckRedundancy}(F,$ satM(F)$)$.
$res[x_1] := F^{x_1}$.
for i from 1 to $n-1$ **do**
    $(F^p, F^n, F^0) := \text{partition}(F)$.
    Let $V_{new} := \text{proj}(V, \{x_i\})$.
    Merging: satM := Merge(satM)**.**
    Let $F_{new} := F^0$ **and** $\text{satM}_{new} := \text{satM}[F^0]$.
    for each $f_p \in F^p$ and $f_n \in F^n$ **do**
        Append $\text{proj}((f_p, f_n), \{x_i\})$ to F_{new}
        Append $\text{satM}[f_p]\&\text{satM}[f_n]$ **to** satM_{new}.
    end for
    F, satM $:= \mathrm{CheckRedundancy}(F_{new}, \text{satM}_{\mathbf{new}})$.
    $V := V_{new}$, $res[x_{i+1}] := F^{x_{i+1}}$.
end for
return *res*.

Complexity

Input H-representation $(A, \overrightarrow{b})$ with $A \in \mathbb{Q}^{m \times n}$, $\overrightarrow{b} \in \mathbb{Q}^m$ and $\text{height}([A, \overrightarrow{b}]) = h$.

Complexity

Input H-representation $(A, \overrightarrow{b})$ with $A \in \mathbb{Q}^{m \times n}$, $\overrightarrow{b} \in \mathbb{Q}^m$ and height$([A, \overrightarrow{b}]) = h$.

- Computing the V-representation [Lemma 9 [JMT20]] –> $\mathcal{O}(m^{n+2} n^{\omega+\varepsilon} h^{1+\varepsilon})$.

Complexity

Input H-representation $(A, \overrightarrow{b})$ with $A \in \mathbb{Q}^{m \times n}$, $\overrightarrow{b} \in \mathbb{Q}^m$ and height$([A, \overrightarrow{b}]) = h$.

- Computing the V-representation [Lemma 9 [JMT20]] –> $\mathcal{O}(m^{n+2} n^{\omega+\varepsilon} h^{1+\varepsilon})$.
- Height of the V-representation [Lemma 8 of [JMT20]] –> $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.

Complexity

Input H-representation $(A, \overrightarrow{b})$ with $A \in \mathbb{Q}^{m \times n}$, $\overrightarrow{b} \in \mathbb{Q}^m$ and $\text{height}([A, \overrightarrow{b}]) = h$.

- Computing the V-representation [Lemma 9 [JMT20]] –> $\mathcal{O}(m^{n+2} n^{\omega+\varepsilon} h^{1+\varepsilon})$.
- Height of the V-representation [Lemma 8 of [JMT20]] –> $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
- Computing the initial satM –> $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
  - It is obtained by multiplying $A \in \mathbb{Q}^{m \times n}$ and $(V, R) \in \mathbb{Q}^{n \times k}$.

Complexity

Input H-representation $(A, \overrightarrow{b})$ with $A \in \mathbb{Q}^{m \times n}$, $\overrightarrow{b} \in \mathbb{Q}^m$ and height$([A, \overrightarrow{b}]) = h$.

- Computing the V-representation [Lemma 9 [JMT20]] –> $\mathcal{O}(m^{n+2} n^{\omega+\varepsilon} h^{1+\varepsilon})$.
- Height of the V-representation [Lemma 8 of [JMT20]] –> $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
- Computing the initial satM –> $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
  - It is obtained by multiplying $A \in \mathbb{Q}^{m \times n}$ and $(V, R) \in \mathbb{Q}^{n \times k}$.
  - Note that height$((V, R))$ is at most $\mathcal{O}(n \log n + nh)$.

Complexity

Input H-representation $(A, \overrightarrow{b})$ with $A \in \mathbb{Q}^{m \times n}$, $\overrightarrow{b} \in \mathbb{Q}^m$ and height$([A, \overrightarrow{b}]) = h$.

- Computing the V-representation [Lemma 9 [JMT20]] –> $\mathcal{O}(m^{n+2} n^{\omega+\varepsilon} h^{1+\varepsilon})$.
- Height of the V-representation [Lemma 8 of [JMT20]] –> $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
- Computing the initial satM –> $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
  - It is obtained by multiplying $A \in \mathbb{Q}^{m \times n}$ and $(V, R) \in \mathbb{Q}^{n \times k}$.
  - Note that height$((V, R))$ is at most $\mathcal{O}(n \log n + nh)$.
  - This multiplication requires at most $\mathcal{O}(mn^{2+\varepsilon} kh) = \mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.

Complexity

Input H-representation $(A, \overrightarrow{b})$ with $A \in \mathbb{Q}^{m \times n}$, $\overrightarrow{b} \in \mathbb{Q}^m$ and $\text{height}([A, \overrightarrow{b}]) = h$.

- Computing the V-representation [Lemma 9 [JMT20]] –> $\mathcal{O}(m^{n+2} n^{\omega+\varepsilon} h^{1+\varepsilon})$.
- Height of the V-representation [Lemma 8 of [JMT20]] –> $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
- Computing the initial satM –> $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
  - It is obtained by multiplying $A \in \mathbb{Q}^{m \times n}$ and $(V, R) \in \mathbb{Q}^{n \times k}$.
  - Note that $\text{height}((V, R))$ is at most $\mathcal{O}(n \log n + nh)$.
  - This multiplication requires at most $\mathcal{O}(mn^{2+\varepsilon} kh) = \mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
- Redundancy detection in the initial input system: –> $\mathcal{O}(m^{n+2})$ bit operations.

Complexity

Input H-representation $(A, \overrightarrow{b})$ with $A \in \mathbb{Q}^{m\times n}$, $\overrightarrow{b} \in \mathbb{Q}^m$ and height$([A, \overrightarrow{b}]) = h$.

- Computing the V-representation [Lemma 9 [JMT20]] –> $\mathcal{O}(m^{n+2} n^{\omega+\varepsilon} h^{1+\varepsilon})$.
- Height of the V-representation [Lemma 8 of [JMT20]] –> $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
- Computing the initial satM –> $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
  - It is obtained by multiplying $A \in \mathbb{Q}^{m\times n}$ and $(V, R) \in \mathbb{Q}^{n\times k}$.
  - Note that height$((V, R))$ is at most $\mathcal{O}(n \log n + nh)$.
  - This multiplication requires at most $\mathcal{O}(mn^{2+\varepsilon} kh) = \mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
- Redundancy detection in the initial input system: –> $\mathcal{O}(m^{n+2})$ bit operations.
  - For a fixed inequality ℓ in F, find the index set I of all the 1's in satM$[\ell]$.

Complexity

Input H-representation $(A, \overrightarrow{b})$ with $A \in \mathbb{Q}^{m\times n}$, $\overrightarrow{b} \in \mathbb{Q}^m$ and height$([A, \overrightarrow{b}]) = h$.

- Computing the V-representation [Lemma 9 [JMT20]] –> $\mathcal{O}(m^{n+2} n^{\omega+\varepsilon} h^{1+\varepsilon})$.
- Height of the V-representation [Lemma 8 of [JMT20]] –> $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
- Computing the initial satM –> $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
  - It is obtained by multiplying $A \in \mathbb{Q}^{m\times n}$ and $(V, R) \in \mathbb{Q}^{n\times k}$.
  - Note that height$((V, R))$ is at most $\mathcal{O}(n \log n + nh)$.
  - This multiplication requires at most $\mathcal{O}(mn^{2+\varepsilon} kh) = \mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
- Redundancy detection in the initial input system: –> $\mathcal{O}(m^{n+2})$ bit operations.
  - For a fixed inequality ℓ in F, find the index set I of all the 1's in satM$[\ell]$.
  - Then, apply bit-wise AND to column vectors of satM$[1..-1, I]$. This requires $m \cdot |I|$ bit operations, where $|I| < k$ is the cardinality of I.

Complexity

Input H-representation $(A, \overrightarrow{b})$ with $A \in \mathbb{Q}^{m \times n}$, $\overrightarrow{b} \in \mathbb{Q}^m$ and $\text{height}([A, \overrightarrow{b}]) = h$.

- Computing the V-representation [Lemma 9 [JMT20]] –> $\mathcal{O}(m^{n+2} n^{\omega+\varepsilon} h^{1+\varepsilon})$.
- Height of the V-representation [Lemma 8 of [JMT20]] –> $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
- Computing the initial satM –> $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
  - It is obtained by multiplying $A \in \mathbb{Q}^{m \times n}$ and $(V, R) \in \mathbb{Q}^{n \times k}$.
  - Note that $\text{height}((V, R))$ is at most $\mathcal{O}(n \log n + nh)$.
  - This multiplication requires at most $\mathcal{O}(mn^{2+\varepsilon} kh) = \mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
- Redundancy detection in the initial input system: –> $\mathcal{O}(m^{n+2})$ bit operations.
  - For a fixed inequality ℓ in F, find the index set I of all the 1's in $\text{satM}[\ell]$.
  - Then, apply bit-wise AND to column vectors of $\text{satM}[1..-1, I]$. This requires $m \cdot |I|$ bit operations, where $|I| < k$ is the cardinality of I.
  - Redundancy detection for one inequality requires at most m^{n+1} bit operations. Therefore, the the redundancy detection for the input system F requires at most m^{n+2} bit operations.

Comparision of Algorithms

For a non-empty, full-dimensional, and pointed polyhedron $P \subset \mathbb{Q}^n$ as input, given by a system of m linear inequalities of height h, the complexity of eliminating d $(\leq n)$ variables, where $\epsilon > 0$, ω denotes the exponent of matrix multiplication and $LP(d, H)$ is an upper bound for the number of bit operations required for solving a linear program in n variables and with total bit size H.

FME Algorithms	Complexity
Original	$\mathcal{O}(m^{2^d})$
Linear Programming	$\mathcal{O}(d^2 m^{2d} LP(d, 2^d h d^2 m^d))$
Balas and Kohler Check	$\mathcal{O}(m^{\frac{5n}{2}} n^{\omega+1+\epsilon} h^{1+\epsilon})$
Saturation Matrix	$\mathcal{O}(m^{2n} n^{\omega+\epsilon} h^{1+\epsilon})$

References I

Sergei N. Chernikov.
Contraction of systems of linear inequalities.
Dokl. Akad. Nauk SSSR, 1960.

D. A. Kohler.
Projections of convex polyhedral sets.
Technical report, California, University at Berkeley, Operations Research Center, 1967.

Leonid Khachiyan.
Fourier-motzkin elimination method.
Springer, 2009.

Alexander Schrijver.
Theory of linear and integer programming.
John Wiley & Sons, 1986.

References II

Komei Fukuda.
Frequently asked questions in polyhedral computation, 2004.
https://people.inf.ethz.ch/fukudak/Doc_pub/polyfaq040618.pdf

Rui-Juan Jing, Marc Moreno Maza, Yan-Feng Xie and Chun-Ming Yuan.
Efficient detection of redundancies in systems of linear inequalities.
ISSAC (to appear), 2024.

Egon Balas.
Projection with a Minimal System of Inequalities.
Computational Optimization and Applications, 1998.

Rui-Juan Jing, Marc Moreno-Maza, and Delaram Talaashrafi.
Complexity estimates for Fourier-Motzkin elimination.
Proceedings of CASC. Springer, 2020.