Fourier-Motzkin Elimination using Saturation Matrix

Chirantan Mukherjee [1], Marc Moreno Maza [1], Jürgen Gerhard [2], Adam Gale [1], Seyed Abdol Hamid Fathi

[1]University of Western Ontario, [2]Maplesoft

November 11, 2024

Content

Contents

What is the Fourier-Motzkin Elimination?

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

What is the Fourier-Motzkin Elimination?

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

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

What is the Fourier-Motzkin Elimination?

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

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

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

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

- FME has many applications in computer science
  - Scheduling
  - Dependence analysis (automatic parallelization)

Background

- FME has many applications in computer science
  - Scheduling
  - Dependence analysis (automatic parallelization)
- Originally proposed by Fourier (1827) and Motzkin (1936)

Background

- FME has many applications in computer science
  - Scheduling
  - Dependence analysis (automatic parallelization)
- Originally proposed by Fourier (1827) and Motzkin (1936)
  - Complexity: $\mathcal{O}(m^{2^d})$ due to redundant inequalities

Background

- FME has many applications in computer science
  - Scheduling
  - Dependence analysis (automatic parallelization)
- Originally proposed by Fourier (1827) and Motzkin (1936)
  - Complexity: $\mathcal{O}(m^{2^d})$ due to redundant inequalities
  - Optimization: Removing redundant inequalities improves efficiency and output size

Background

- FME has many applications in computer science
  - Scheduling
  - Dependence analysis (automatic parallelization)
- Originally proposed by Fourier (1827) and Motzkin (1936)
  - Complexity: $\mathcal{O}(m^{2^d})$ due to redundant inequalities
  - Optimization: Removing redundant inequalities improves efficiency and output size
  - Using linear programming (LP): Complexity improves to $\mathcal{O}(d^2 m^{2d} LP(d, 2^d h d^2 m^d))$ for a polyhedron with dimension d, m facets and coefficient height h.

Background

- FME has many applications in computer science
  - Scheduling
  - Dependence analysis (automatic parallelization)
- Originally proposed by Fourier (1827) and Motzkin (1936)
  - Complexity: $\mathcal{O}(m^{2^d})$ due to redundant inequalities
  - Optimization: Removing redundant inequalities improves efficiency and output size
  - Using linear programming (LP): Complexity improves to $\mathcal{O}(d^2 m^{2d} LP(d, 2^d hd^2 m^d))$ for a polyhedron with dimension d, 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 by Jing, Moreno-Maza and Talaashrafi [JMT20].

Background

- FME has many applications in computer science
  - Scheduling
  - Dependence analysis (automatic parallelization)
- Originally proposed by Fourier (1827) and Motzkin (1936)
  - Complexity: $\mathcal{O}(m^{2^d})$ due to redundant inequalities
  - Optimization: Removing redundant inequalities improves efficiency and output size
  - Using linear programming (LP): Complexity improves to $\mathcal{O}(d^2 m^{2d} LP(d, 2^d h d^2 m^d))$ for a polyhedron with dimension d, 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 by Jing, Moreno-Maza and Talaashrafi [JMT20].
- Jing, Moreno-Maza, Xie and Yuan [JMXY24] proposed a method using Saturation Matrix.

Maple Package

- FME algorithm is part by `PolyhedralSets` and `RegularChains` library.

Maple Package

- FME algorithm is part by `PolyhedralSets` and `RegularChains` library.
- We have three `redundancycheck` algorithms for doing FME based on,
  1. linear-programming
  2. redundancy cone, which is `Maple`'s default
  3. saturation matrix, which will be added in `Maple 2025` release and will become the default algorithm.

Maple Package

- FME algorithm is part by `PolyhedralSets` and `RegularChains` library.
- We have three `redundancycheck` algorithms for doing FME based on,
  1. linear-programming
  2. redundancy cone, which is `Maple`'s default
  3. saturation matrix, which will be added in `Maple 2025` release and will become the default algorithm.
- Users can access it using the following functions,
  1. `PolyhedralSets:-Project`
  2. `RegularChains:-FMXelim`
  3. `RegularChains:-SemiAlgebraicSetTools:-LinearSolve`

Definition (H-representation)

A polyhedral set P is any $\{\mathbf{x} \mid A\mathbf{x} <= \mathbf{b}\}$, where $A \in \mathbb{Q}^{m\times n}$ and $\mathbf{b} \in \mathbb{Q}^{m}$. Such a linear system is called an H-representation of P.

Definition (H-representation)

A polyhedral set P is any $\{\mathbf{x} \mid A\mathbf{x} <= \mathbf{b}\}$, where $A \in \mathbb{Q}^{m \times n}$ and $\mathbf{b} \in \mathbb{Q}^m$. Such a linear system is called an H-representation of P.

Definition (V-representation)

Let V and R denote the set of vertices and rays of P. Then, the pair $\mathcal{VR}(F) = (V, R)$ is called a V-representation of P.

Definition (H-representation)

A polyhedral set P is any $\{\mathbf{x} \mid A\mathbf{x} <= \mathbf{b}\}$, where $A \in \mathbb{Q}^{m \times n}$ and $\mathbf{b} \in \mathbb{Q}^m$. Such a linear system is called an H-representation of P.

Definition (V-representation)

Let V and R denote the set of vertices and rays of P. Then, the pair $\mathcal{VR}(F) = (V, R)$ is called a V-representation of P.

Definition

- P is full-dimensional whenever $dim(P) = n$.

Definition (H-representation)

A polyhedral set P is any $\{\mathbf{x} \mid A\mathbf{x} <= \mathbf{b}\}$, where $A \in \mathbb{Q}^{m \times n}$ and $\mathbf{b} \in \mathbb{Q}^m$. Such a linear system is called an H-representation of P.

Definition (V-representation)

Let V and R denote the set of vertices and rays of P. Then, the pair $\mathcal{VR}(F) = (V, R)$ is called a V-representation of P.

Definition

- P is full-dimensional whenever $dim(P) = n$.
- P is full-dimensional iff $A\mathbf{x} <= \mathbf{b}$ has no implicit equation.

Definition (H-representation)

A polyhedral set P is any $\{\mathbf{x} \mid A\mathbf{x} <= \mathbf{b}\}$, where $A \in \mathbb{Q}^{m \times n}$ and $\mathbf{b} \in \mathbb{Q}^m$. Such a linear system is called an H-representation of P.

Definition (V-representation)

Let V and R denote the set of vertices and rays of P. Then, the pair $\mathcal{VR}(F) = (V, R)$ is called a V-representation of P.

Definition

- P is full-dimensional whenever $dim(P) = n$.
- P is full-dimensional iff $A\mathbf{x} <= \mathbf{b}$ has no implicit equation.
- P is pointed, if A is full column rank.

NOTE: From now, P is full-dimensional and pointed.

Definition

To eliminate x_1 from two inequalities, $a_1x_1 + \cdots + a_nx_n \leq d_1$ and $b_1x_1 + \cdots + b_nx_n \leq d_2$ where $a_1 > 0$ and $b_1 < 0$, we can multiply the first inequality by $|b_1|$ and the second one by a_1 and add:

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

Definition

To eliminate x_1 from two inequalities, $a_1x_1 + \cdots + a_nx_n \leq d_1$ and $b_1x_1 + \cdots + b_nx_n \leq d_2$ where $a_1 > 0$ and $b_1 < 0$, we can multiply the first inequality by $|b_1|$ and the second one by a_1 and add:

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

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 set formed by the combining each inequality in A^+ with each inequality in A^- and including inequalities in A^0 such that A' does not have x_1 term. 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 points satisfying all inequalities in A.

Idea

Theorem

Let A' be the set formed by the combining each inequality in A^+ with each inequality in A^- and including inequalities in A^0 such that A' does not have x_1 term. 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 points satisfying all inequalities in A.

FME Algorithm

Idea

Theorem

Let A' be the set formed by the combining each inequality in A^+ with each inequality in A^- and including inequalities in A^0 such that A' does not have x_1 term. 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 points satisfying all inequalities in A.

FME Algorithm

- Select each variable in turn.

Idea

Theorem

Let A' be the set formed by the combining each inequality in A^+ with each inequality in A^- and including inequalities in A^0 such that A' does not have x_1 term. 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 points satisfying all inequalities in A.

FME Algorithm

- Select each variable in turn.
- Partition inequalities into A^+, A^- and A^0 based on the variable's coefficient.

Idea

Theorem

Let A' be the set formed by the combining each inequality in A^+ with each inequality in A^- and including inequalities in A^0 such that A' does not have x_1 term. 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 points satisfying all inequalities in A.

FME Algorithm

- Select each variable in turn.
- Partition inequalities into A^+, A^- and A^0 based on the variable's coefficient.
- Combine inequalities in A^+ and A^- to form the resulting union.

Complexity

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

Complexity

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

- Partitioning inequalities by x_1 coefficient is $\mathcal{O}(m)$.

Complexity

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

- Partitioning inequalities by 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 and d variables:

- Partitioning inequalities by 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 and d variables:

- Partitioning inequalities by 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 and d variables:

- Partitioning inequalities by 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 these generated inequalities are redundant.
- Detecting and removing them can significantly improve the complexity.

Definition

For a consistent system of linear inequalities $F : \{A\mathbf{x} \leq \mathbf{b}\}$ representing a polyhedron P:

Definition

For a consistent system of linear inequalities $F : \{A\mathbf{x} \leq \mathbf{b}\}$ representing a polyhedron P:

- An inequality $\ell : \mathbf{a}^t\mathbf{x} \leq \mathbf{b}$ in F is redundant if $F \setminus \{\ell\}$ still represents P.

Definition

For a consistent system of linear inequalities $F : \{A\mathbf{x} \le \mathbf{b}\}$ representing a polyhedron P:

- An inequality $\ell : \mathbf{a}^t\mathbf{x} \le \mathbf{b}$ in F is redundant if $F \setminus \{\ell\}$ still represents P.
- Otherwise, ℓ is irredundant.

Definition

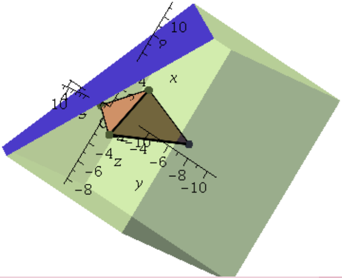
For a consistent system of linear inequalities $F : \{A\mathbf{x} \leq \mathbf{b}\}$ representing a polyhedron P:

- An inequality $\ell : \mathbf{a}^t\mathbf{x} \leq \mathbf{b}$ in F is redundant if $F \setminus \{\ell\}$ still represents P.
- Otherwise, ℓ is irredundant.
- Strongly redundant if $\mathbf{a}^t\mathbf{x} < \mathbf{b}$ for all $\mathbf{x} \in P$.

Definition

For a consistent system of linear inequalities $F : \{A\mathbf{x} \leq \mathbf{b}\}$ representing a polyhedron P:

- An inequality $\ell : \mathbf{a}^t\mathbf{x} \leq \mathbf{b}$ in F is redundant if $F \setminus \{\ell\}$ still represents P.
- Otherwise, ℓ is irredundant.
- Strongly redundant if $\mathbf{a}^t\mathbf{x} < \mathbf{b}$ for all $\mathbf{x} \in P$.
- Weakly redundant if $\mathbf{a}^t\mathbf{x} = \mathbf{b}$ holds for some $\mathbf{x} \in P$.



Contents

- The classical Double Description Method was introduced by Motzkin, Raiffa, Thompson and Thrall in [MRTT53].

- The classical Double Description Method was introduced by Motzkin, Raiffa, Thompson and Thrall in [MRTT53].
- Recall that a polyhedral cone P can be represented as
  1. intersection of finitely many half-spaces, called the H-representation,

     $$P \text{ is an H-cone, i.e., } \exists \text{ matrix } A \text{ such that } P = \{\mathbf{x} \mid A\mathbf{x} >= \mathbf{0}\}$$

  2. by its vertices and rays, called the V-representation,

     $$P \text{ is a V-cone, i.e., } \exists \text{ matrix } R \text{ such that } P = \{\mathbf{x} \mid \mathbf{x} = R\lambda, \lambda >= 0\}.$$

- The classical Double Description Method was introduced by Motzkin, Raiffa, Thompson and Thrall in [MRTT53].
- Recall that a polyhedral cone P can be represented as
  1. intersection of finitely many half-spaces, called the H-representation,

     $$P \text{ is an H-cone, i.e., } \exists \text{ matrix } A \text{ such that } P = \{\mathbf{x} \mid A\mathbf{x} >= \mathbf{0}\}$$

  2. by its vertices and rays, called the V-representation,

     $$P \text{ is a V-cone, i.e., } \exists \text{ matrix } R \text{ such that } P = \{\mathbf{x} \mid \mathbf{x} = R\lambda, \lambda >= 0\}.$$

- The DD algorithm converts one representation to another; the pair (A, R) is called a double description pair,

$$A\mathbf{x} >= \mathbf{0} \iff \mathbf{x} = R\lambda \text{ for some } \lambda >= 0.$$

- The classical Double Description Method was introduced by Motzkin, Raiffa, Thompson and Thrall in [MRTT53].
- Recall that a polyhedral cone P can be represented as
  1. intersection of finitely many half-spaces, called the H-representation,

     $$P \text{ is an H-cone, i.e., } \exists \text{ matrix } A \text{ such that } P = \{\mathbf{x} \mid A\mathbf{x} >= \mathbf{0}\}$$

  2. by its vertices and rays, called the V-representation,

     $$P \text{ is a V-cone, i.e., } \exists \text{ matrix } R \text{ such that } P = \{\mathbf{x} \mid \mathbf{x} = R\lambda, \lambda >= 0\}.$$

- The DD algorithm converts one representation to another; the pair (A, R) is called a double description pair,

  $$A\mathbf{x} >= \mathbf{0} \iff \mathbf{x} = R\lambda \text{ for some } \lambda >= 0.$$

- We focus on the version that takes as input the H-representation of P and returns the V-representation of P.

- The classical Double Description Method was introduced by Motzkin, Raiffa, Thompson and Thrall in [MRTT53].
- Recall that a polyhedral cone P can be represented as
  1. intersection of finitely many half-spaces, called the H-representation,

     $$P \text{ is an H-cone, i.e., } \exists \text{ matrix } A \text{ such that } P = \{\mathbf{x} \mid A\mathbf{x} >= \mathbf{0}\}$$

  2. by its vertices and rays, called the V-representation,

     $$P \text{ is a V-cone, i.e., } \exists \text{ matrix } R \text{ such that } P = \{\mathbf{x} \mid \mathbf{x} = R\lambda, \lambda >= 0\}.$$

- The DD algorithm converts one representation to another; the pair (A, R) is called a double description pair,

$$A\mathbf{x} >= \mathbf{0} \iff \mathbf{x} = R\lambda \text{ for some } \lambda >= 0.$$

- We focus on the version that takes as input the H-representation of P and returns the V-representation of P.
- The most efficient variant, proposed by Fukuda and Prodon in [FP96] is implemented in the CDD library [cdd].

Given an inequality $A_i\mathbf{x} >= 0$, we partition the space $\mathbb{Q}^d$ into three regions,

- $H_i^+ = \{\mathbf{x} \in \mathbb{Q}^d \mid A_i\mathbf{x} > 0\}$.
- $H_i^- = \{\mathbf{x} \in \mathbb{Q}^d \mid A_i\mathbf{x} < 0\}$.
- $H_i^0 = \{\mathbf{x} \in \mathbb{Q}^d \mid A_i\mathbf{x} = 0\}$.

Given an inequality $A_i\mathbf{x} >= 0$, we partition the space $\mathbb{Q}^d$ into three regions,

- $H_i^+ = \{\mathbf{x} \in \mathbb{Q}^d \mid A_i\mathbf{x} > 0\}$.
- $H_i^- = \{\mathbf{x} \in \mathbb{Q}^d \mid A_i\mathbf{x} < 0\}$.
- $H_i^0 = \{\mathbf{x} \in \mathbb{Q}^d \mid A_i\mathbf{x} = 0\}$.

Let J be the set of column indices of R. The rays r_j ($j \in J$) are then partitioned as follows,

- $J^+ = \{j \in J \mid \mathbf{r_j} \in H_i^+\}$.
- $J^- = \{j \in J \mid \mathbf{r_j} \in H_i^-\}$.
- $J^0 = \{j \in J \mid \mathbf{r_j} \in H_i^0\}$.

Given an inequality $A_i\mathbf{x} >= 0$, we partition the space $\mathbb{Q}^d$ into three regions,

- $H_i^+ = \{\mathbf{x} \in \mathbb{Q}^d \mid A_i\mathbf{x} > 0\}$.
- $H_i^- = \{\mathbf{x} \in \mathbb{Q}^d \mid A_i\mathbf{x} < 0\}$.
- $H_i^0 = \{\mathbf{x} \in \mathbb{Q}^d \mid A_i\mathbf{x} = 0\}$.

Let J be the set of column indices of R. The rays r_j $(j \in J)$ are then partitioned as follows,

- $J^+ = \{j \in J \mid \mathbf{r_j} \in H_i^+\}$.
- $J^- = \{j \in J \mid \mathbf{r_j} \in H_i^-\}$.
- $J^0 = \{j \in J \mid \mathbf{r_j} \in H_i^0\}$.

Lemma

Let (A_K, R) be a DD pair and let i be a row index of A not in K. Then (A_{K+i}, R') is a DD pair, where R' is a $d \times |J'|$ matrix with column vectors $\mathbf{r_j}$ $(j \in J')$ defined by,

$$J' = J^+ \cap J^0 \cap (J^+ \times J^-), \text{ and}$$
$$\mathbf{r_{jj'}} = (A_i\mathbf{r_j})\mathbf{r_{j'}} - (A_i\mathbf{r_{j'}})\mathbf{r_j} \text{ for each } (j, j') \in J^+ \times J^-.$$

Let P be an H-cone defined by $P = \{\mathbf{x} \in \mathbb{R}^d \mid A\mathbf{x} >= \mathbf{0}\}$.

Lemma (DD Pair Duality)

For any A and R, (A, R) is a DD pair $\iff$ (R^T, A^T) is a DD pair.

Let P be an H-cone defined by $P = \{\mathbf{x} \in \mathbb{R}^d \mid A\mathbf{x} >= \mathbf{0}\}$.

Lemma (DD Pair Duality)

For any A and R, (A, R) is a DD pair $\iff$ (R^T, A^T) is a DD pair.

Definition

The zero set of $\mathbf{u} \in P$, denoted $Z(A)$ is the set of row indices i such that $A_i\mathbf{u} = 0$, where A_i is the i-th row of A.

Let P be an H-cone defined by $P = \{\mathbf{x} \in \mathbb{R}^d \mid A\mathbf{x} >= \mathbf{0}\}$.

Lemma (DD Pair Duality)

For any A and R, (A, R) is a DD pair $\iff$ (R^T, A^T) is a DD pair.

Definition

The zero set of $\mathbf{u} \in P$, denoted $Z(A)$ is the set of row indices i such that $A_i\mathbf{u} = 0$, where A_i is the i-th row of A.

Lemma (Adjacency Test)

Two distinct rays r and r' of the polyhedral cone P are called adjacent if they span a 2-dimensional face of P, that is,

$$Rank(A_{Z(r)\cap Z(r')}) = d - 2.$$

Algorithm 1 Double Description Method

Require: 1. a matrix $A \in \mathbb{Q}^{m \times n}$ defining the H-representation of a polyhedral cone P;
2. P is full-dimensional and pointed.

Ensure: a matrix R defining the V-representation of P.

Let K be the set of the indices of A's independent rows;
$R' := (A_K)^{-1}$;
Let J be the set of the columns of R';
while $K \neq \{1, \ldots, m\}$ do **do**
    Select a A-row index $i \notin K$;
    Set R' to empty matrix;
    $J^+, J^-, J^0 :=$ **Partition**(J, A_i);
    Append J^+ and J^0 as columns in R';
    for $p \in J^+$ **do**
        for $n \in J^-$ **do**
            if AdjacencyTest$(A_K, \mathbf{r_p}, \mathbf{r_n})$ **then**
                $\mathbf{r_{new}} := (A_i \mathbf{r_p})\mathbf{r_n} - (A_i \mathbf{r_n})\mathbf{r_p}$;
                Append $\mathbf{r_{new}}$ as a column to R';
            end if
        end for
    end for
    Let J be the set of the columns of R';
    $K = K \cup \{i\}$;
end while
Let R be the matrix created by the vectors in J as its columns;
return R;

Contents

Saturation Matrix

Definition

- A vertex $\mathbf{v} \in V$ of P *saturates* an inequality ℓ if $\mathbf{v}$ lies on the hyperplane $\mathcal{H}_\ell$, that is, if $\mathbf{a}^t\mathbf{v} = b$ holds.
- A ray $\mathbf{r} \in R$ of P *saturates* an inequality ℓ if $\mathbf{r}$ is parallel to the hyperplane $\mathcal{H}_\ell$, that is, if $\mathbf{a}^t\mathbf{r} = 0$ holds.

Saturation Matrix

Definition

- A vertex $\mathbf{v} \in V$ of P *saturates* an inequality ℓ if $\mathbf{v}$ lies on the hyperplane $\mathcal{H}_\ell$, that is, if $\mathbf{a}^t\mathbf{v} = b$ holds.
- A ray $\mathbf{r} \in R$ of P *saturates* an inequality ℓ if $\mathbf{r}$ is parallel to the hyperplane $\mathcal{H}_\ell$, that is, if $\mathbf{a}^t\mathbf{r} = 0$ holds.

Notation:

Let ℓ be an inequality and $\mathbf{u}$ be any vertex or ray, we denote

Saturation Matrix

Definition

- A vertex $\mathbf{v} \in V$ of P *saturates* an inequality ℓ if $\mathbf{v}$ lies on the hyperplane $\mathcal{H}_\ell$, that is, if $\mathbf{a}^t\mathbf{v} = b$ holds.
- A ray $\mathbf{r} \in R$ of P *saturates* an inequality ℓ if $\mathbf{r}$ is parallel to the hyperplane $\mathcal{H}_\ell$, that is, if $\mathbf{a}^t\mathbf{r} = 0$ holds.

Notation:

Let ℓ be an inequality and $\mathbf{u}$ be any vertex or ray, we denote

- $\mathcal{S}^{\mathcal{VR}}(\ell)$: all vertices and rays in $\mathcal{VR}(F)$ saturating ℓ.

Saturation Matrix

Definition

- A vertex $\mathbf{v} \in V$ of P *saturates* an inequality ℓ if $\mathbf{v}$ lies on the hyperplane $\mathcal{H}_\ell$, that is, if $\mathbf{a}^t\mathbf{v} = b$ holds.
- A ray $\mathbf{r} \in R$ of P *saturates* an inequality ℓ if $\mathbf{r}$ is parallel to the hyperplane $\mathcal{H}_\ell$, that is, if $\mathbf{a}^t\mathbf{r} = 0$ holds.

Notation:

Let ℓ be an inequality and $\mathbf{u}$ be any vertex or ray, we denote

- $\mathcal{S}^{\mathcal{VR}}(\ell)$: all vertices and rays in $\mathcal{VR}(F)$ saturating ℓ.
- $\mathcal{S}^{\mathcal{H}}(\mathbf{u})$: all hyperplanes that $\mathbf{u}$ saturates.

Saturation Matrix

Definition

- A vertex $\mathbf{v} \in V$ of P *saturates* an inequality ℓ if $\mathbf{v}$ lies on the hyperplane $\mathcal{H}_\ell$, that is, if $\mathbf{a}^t\mathbf{v} = b$ holds.
- A ray $\mathbf{r} \in R$ of P *saturates* an inequality ℓ if $\mathbf{r}$ is parallel to the hyperplane $\mathcal{H}_\ell$, that is, if $\mathbf{a}^t\mathbf{r} = 0$ holds.

Notation:

Let ℓ be an inequality and $\mathbf{u}$ be any vertex or ray, we denote

- $\mathcal{S}^{\mathcal{VR}}(\ell)$: all vertices and rays in $\mathcal{VR}(F)$ saturating ℓ.
- $\mathcal{S}^{\mathcal{H}}(\mathbf{u})$: all hyperplanes that $\mathbf{u}$ saturates.
- $\mathcal{S}^{\mathcal{H}}(\mathcal{S}^{\mathcal{VR}}(\ell)) = \bigcap_{\mathbf{u} \in \mathcal{S}^{\mathcal{VR}}(\ell)} \mathcal{S}^{\mathcal{H}}(\mathbf{u})$: inequalities saturated by all vertices or rays saturating ℓ.

Definition

The *saturation matrix* of F is the boolean matrix $\mathrm{satM}(F) \in \mathbb{Q}^{m \times k}$, where each entry (i, j) is defined as follows:

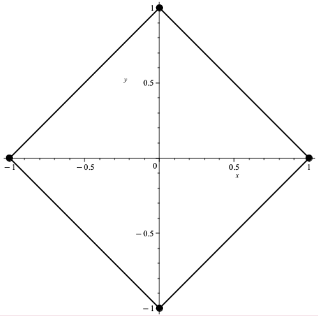
- 1 if the j-th element of $\mathcal{VR}(F)$ saturates the i-th inequality of F.
- 0 otherwise.

Example

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

satM(F)

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



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:

$$
\begin{aligned}
\mathcal{S}^{\mathcal{VR}}(\ell_1) &= \{\mathbf{v_1}, \mathbf{v_2}\} & \mathcal{S}^{\mathcal{H}}(\mathbf{v_1}) &= \{\ell_1, \ell_4\} \\
\mathcal{S}^{\mathcal{VR}}(\ell_2) &= \{\mathbf{v_3}, \mathbf{v_4}\} & \mathcal{S}^{\mathcal{H}}(\mathbf{v_2}) &= \{\ell_1, \ell_3\} \\
\mathcal{S}^{\mathcal{VR}}(\ell_3) &= \{\mathbf{v_2}, \mathbf{v_4}\} & \mathcal{S}^{\mathcal{H}}(\mathbf{v_3}) &= \{\ell_2, \ell_4\} \\
\mathcal{S}^{\mathcal{VR}}(\ell_4) &= \{\mathbf{v_1}, \mathbf{v_3}\} & \mathcal{S}^{\mathcal{H}}(\mathbf{v_4}) &= \{\ell_2, \ell_3\}
\end{aligned}
$$

Updating the Saturation Matrix

The saturation matrix is processed in two ways:

- Row-wise: to compute bit-wise AND.
- Column-wise: to compute bit-wise OR.

Updating the Saturation Matrix

The saturation matrix is processed in two ways:

- Row-wise: to compute bit-wise AND.
- Column-wise: to compute bit-wise OR.
- $\mathcal{S}^{\mathcal{VR}}(\ell)$ is found by taking bit-wise AND of the Boolean vectors satM$(F)[\ell_{pos}]$ and satM$(F)[\ell_{neg}]$.

Updating the Saturation Matrix

The saturation matrix is processed in two ways:

- Row-wise: to compute bit-wise AND.
- Column-wise: to compute bit-wise OR.
- $\mathcal{S}^{\mathcal{VR}}(\ell)$ is found by taking bit-wise AND of the Boolean vectors $\mathrm{satM}(F)[\ell_{pos}]$ and $\mathrm{satM}(F)[\ell_{neg}]$.
- Merging: creates a saturation matrix for the subspace of the remaining coordinates.

Updating the Saturation Matrix

The saturation matrix is processed in two ways:

- Row-wise: to compute bit-wise AND.
- Column-wise: to compute bit-wise OR.
- $\mathcal{S}^{\mathcal{VR}}(\ell)$ is found by taking bit-wise AND of the Boolean vectors $\mathrm{satM}(F)[\ell_{pos}]$ and $\mathrm{satM}(F)[\ell_{neg}]$.
- Merging: creates a saturation matrix for the subspace of the remaining coordinates.
- For any vertices or rays $\{u_1, \ldots, u_e\}$ where $\mathrm{proj}(u_1, \{x\}) = \cdots = \mathrm{proj}(u_e, \{x\})$:

Updating the Saturation Matrix

The saturation matrix is processed in two ways:

- Row-wise: to compute bit-wise AND.
- Column-wise: to compute bit-wise OR.
- $\mathcal{S}^{\mathcal{VR}}(\ell)$ is found by taking bit-wise AND of the Boolean vectors $\mathrm{satM}(F)[\ell_{pos}]$ and $\mathrm{satM}(F)[\ell_{neg}]$.
- Merging: creates a saturation matrix for the subspace of the remaining coordinates.
- For any vertices or rays $\{u_1, \ldots, u_e\}$ where $\mathrm{proj}(u_1, \{x\}) = \cdots = \mathrm{proj}(u_e, \{x\})$:
    - Compute the bit-wise OR of the columns (regarded as bit-vectors) of $\mathrm{satM}(F)$ indexed by $u_1, \ldots, u_e$.

Updating the Saturation Matrix

The saturation matrix is processed in two ways:

- Row-wise: to compute bit-wise AND.
- Column-wise: to compute bit-wise OR.
- $\mathcal{S}^{\mathcal{VR}}(\ell)$ is found by taking bit-wise AND of the Boolean vectors $\mathrm{satM}(F)[\ell_{pos}]$ and $\mathrm{satM}(F)[\ell_{neg}]$.
- Merging: creates a saturation matrix for the subspace of the remaining coordinates.
- For any vertices or rays $\{u_1, \ldots, u_e\}$ where $\mathrm{proj}(u_1, \{x\}) = \cdots = \mathrm{proj}(u_e, \{x\})$:
  - Compute the bit-wise OR of the columns (regarded as bit-vectors) of $\mathrm{satM}(F)$ indexed by $u_1, \ldots, u_e$.
  - Replace these columns with results of this bit-wise OR.

Contents

Algorithm 2 CheckRedundancy

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

Ensure: the minimal system F_{irred} and the corresponding saturation matrix $\mathrm{satM}_{\mathrm{irred}}$.

$\mathit{Irredundant} := \{\mathrm{seq}(i, i = 1..m)\}$;
for i from 1 to m **do**
  if the number of nonzero elements in satM$[i]$ is less than n **then**
    $\mathit{Irredundant} := \mathit{Irredundant} \setminus \{i\}$;
    next;
  end if
  for j in $\mathit{Irredundant} \setminus \{i\}$ **do**
    if $\mathrm{satM}[i] = \mathrm{satM}[i] \mathit{AND} \mathrm{satM}[j]$ **then**
      $\mathit{Irredundant} := \mathit{Irredundant} \setminus \{i\}$;
      break;
    end if
  end for
end for
$F_{\mathrm{irred}} := [\mathrm{seq}(F[i], i \text{ in } \mathit{Irredundant})]$ and $\mathrm{satM}_{\mathrm{irred}} := [\mathrm{seq}(\mathrm{satM}[i], i \text{ in } \mathit{Irredundant})]$;
return F_{irred} and $\mathrm{satM}_{\mathrm{irred}}$;

Algorithm 3 Minimal projected representation

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

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

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

Contents

Complexity

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

Complexity

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

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

Complexity

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

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

Complexity

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

- Compute V-representation [Lemma 9 [JMT20]] : $\mathcal{O}(m^{n+2} n^{\omega+\varepsilon} h^{1+\varepsilon})$.
- Height of V-representation [Lemma 8 of [JMT20]] : $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
- Compute initial satM : $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
  - Computed by multiplying $A \in \mathbb{Q}^{m \times n}$ with $(V, R) \in \mathbb{Q}^{n \times k}$.

Complexity

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

- Compute V-representation [Lemma 9 [JMT20]] : $\mathcal{O}(m^{n+2} n^{\omega+\varepsilon} h^{1+\varepsilon})$.
- Height of V-representation [Lemma 8 of [JMT20]] : $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
- Compute initial satM : $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
    - Computed by multiplying $A \in \mathbb{Q}^{m \times n}$ with $(V, R) \in \mathbb{Q}^{n \times k}$.
    - $\text{height}((V, R))$ is at most $\mathcal{O}(n \log n + nh)$.

Complexity

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

- Compute V-representation [Lemma 9 [JMT20]] : $\mathcal{O}(m^{n+2} n^{\omega+\varepsilon} h^{1+\varepsilon})$.
- Height of V-representation [Lemma 8 of [JMT20]] : $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
- Compute initial satM : $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
  - Computed by multiplying $A \in \mathbb{Q}^{m \times n}$ with $(V, R) \in \mathbb{Q}^{n \times k}$.
  - $\text{height}((V, R))$ is at most $\mathcal{O}(n \log n + nh)$.
  - Multiplication requires $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.

Complexity

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

- Compute V-representation [Lemma 9 [JMT20]] : $\mathcal{O}(m^{n+2} n^{\omega+\varepsilon} h^{1+\varepsilon})$.
- Height of V-representation [Lemma 8 of [JMT20]] : $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
- Compute initial satM : $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
  - Computed by multiplying $A \in \mathbb{Q}^{m \times n}$ with $(V, R) \in \mathbb{Q}^{n \times k}$.
  - $\text{height}((V, R))$ is at most $\mathcal{O}(n \log n + nh)$.
  - Multiplication requires $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
- Redundancy detection in the initial input system: : $\mathcal{O}(m^{n+2})$ bit operations.

Complexity

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

- Compute V-representation [Lemma 9 [JMT20]] : $\mathcal{O}(m^{n+2} n^{\omega+\varepsilon} h^{1+\varepsilon})$.
- Height of V-representation [Lemma 8 of [JMT20]] : $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
- Compute initial satM : $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
  - Computed by multiplying $A \in \mathbb{Q}^{m \times n}$ with $(V, R) \in \mathbb{Q}^{n \times k}$.
  - $\mathrm{height}((V, R))$ is at most $\mathcal{O}(n \log n + nh)$.
  - Multiplication requires $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
- Redundancy detection in the initial input system: : $\mathcal{O}(m^{n+2})$ bit operations.
  - For each inequality ℓ in F, find the index set I of all the 1's in satM[ℓ].

Complexity

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

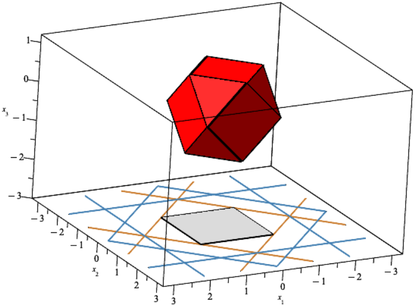
- Compute V-representation [Lemma 9 [JMT20]] : $\mathcal{O}(m^{n+2} n^{\omega+\varepsilon} h^{1+\varepsilon})$.
- Height of V-representation [Lemma 8 of [JMT20]] : $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
- Compute initial satM : $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
  - Computed by multiplying $A \in \mathbb{Q}^{m\times n}$ with $(V, R) \in \mathbb{Q}^{n\times k}$.
  - $\mathrm{height}((V, R))$ is at most $\mathcal{O}(n \log n + nh)$.
  - Multiplication requires $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
- Redundancy detection in the initial input system: : $\mathcal{O}(m^{n+2})$ bit operations.
  - For each inequality ℓ in F, find the index set I of all the 1's in satM$[\ell]$.
  - Apply bit-wise AND on column of satM$[1.. - 1, I]$ requiring $m \cdot |I|$ bit operations, where $|I| < k$.

Complexity

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

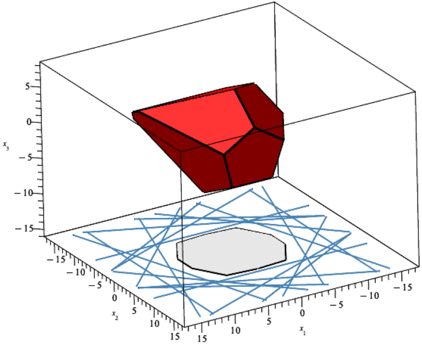
- Compute V-representation [Lemma 9 [JMT20]] : $\mathcal{O}(m^{n+2} n^{\omega+\varepsilon} h^{1+\varepsilon})$.
- Height of V-representation [Lemma 8 of [JMT20]] : $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
- Compute initial satM : $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
  - Computed by multiplying $A \in \mathbb{Q}^{m\times n}$ with $(V, R) \in \mathbb{Q}^{n\times k}$.
  - $\text{height}((V, R))$ is at most $\mathcal{O}(n \log n + nh)$.
  - Multiplication requires $\mathcal{O}(m^{n+1} n^{2+\varepsilon} h)$.
- Redundancy detection in the initial input system: : $\mathcal{O}(m^{n+2})$ bit operations.
  - For each inequality ℓ in F, find the index set I of all the 1's in $\text{satM}[\ell]$.
  - Apply bit-wise AND on column of $\text{satM}[1.. - 1, I]$ requiring $m \cdot |I|$ bit operations, where $|I| < k$.
  - Detectiing redundancy for one inequality requires m^{n+1}; for all inequalities: m^{n+2} bit operations.

Cuboctahedron



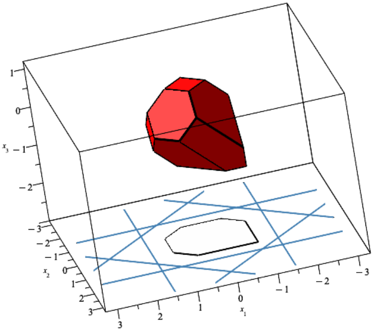
1. strongly redundannt inequalities
2. weakly redundant inequalities eliminated by cardinality
3. weakly redundancies inequalities eliminated by containment

Snub disphenoid (triangular dodecahedron)



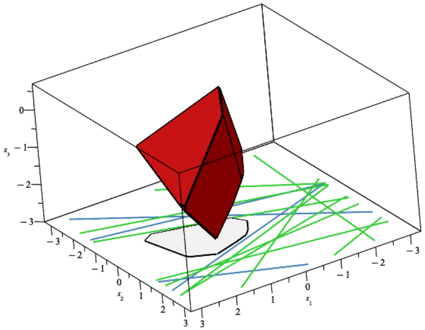
1. strongly redundannt inequalities
2. weakly redundant inequalities eliminated by cardinality
3. weakly redundancies inequalities eliminated by containment

Truncated octahedron



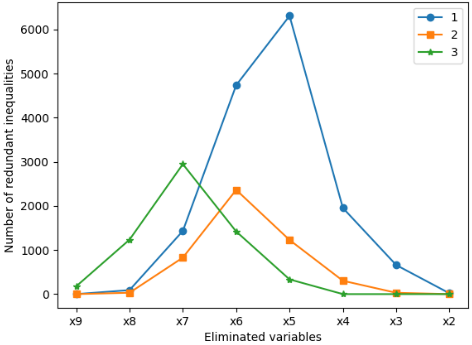
1. strongly redundant inequalities
2. weakly redundant inequalities eliminated by cardinality
3. weakly redundancies inequalities eliminated by containment

Random 3D polyhedron



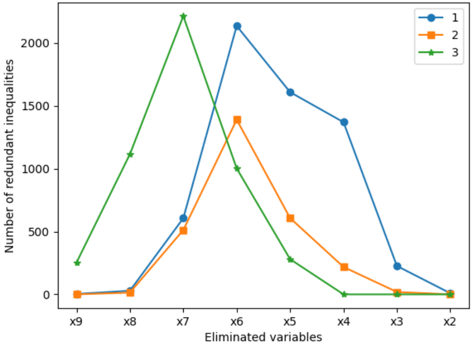
1. strongly redundant inequalities
2. weakly redundant inequalities eliminated by cardinality
3. weakly redundancies inequalities eliminated by containment

Random 10D polyhedron



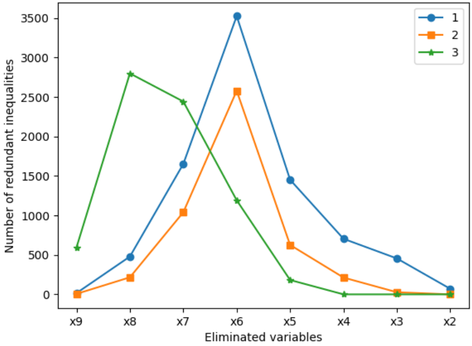
1. strongly redundant inequalities
2. weakly redundant inequalities eliminated by cardinality
3. weakly redundancies inequalities eliminated by containment

Random 10D polyhedron



1. strongly redundant inequalities
2. weakly redundant inequalities eliminated by cardinality
3. weakly redundancies inequalities eliminated by containment

Random 10D polyhedron



1. strongly redundant inequalities
2. weakly redundant inequalities eliminated by cardinality
3. weakly redundancies inequalities eliminated by containment

Comparative experimentation

- FME and mpr (this algorithm by Jing, Moreno-Maza, Xie and Yuan [JMXY24]): Eliminates variables sequentially, uses both the H- and V-representations, with redundancy check via the saturation matrix.

Comparative experimentation

- FME and mpr (this algorithm by Jing, Moreno-Maza, Xie and Yuan [JMXY24]): Eliminates variables sequentially, uses both the H- and V-representations, with redundancy check via the saturation matrix.
- RTC and BPAS (by Jing, Moreno-Maza and Talaashrafi [JMT20]): Eliminates variables sequentially, useing H- and V-representations, with redundancy check via redundancy cones, thus linear algebra over $\mathbb{Q}$.

Comparative experimentation

- FME and mpr (this algorithm by Jing, Moreno-Maza, Xie and Yuan [JMXY24]): Eliminates variables sequentially, uses both the H- and V-representations, with redundancy check via the saturation matrix.
- RTC and BPAS (by Jing, Moreno-Maza and Talaashrafi [JMT20]): Eliminates variables sequentially, useing H- and V-representations, with redundancy check via redundancy cones, thus linear algebra over $\mathbb{Q}$.
- cddlib(by Fukuda [cdd]): Eliminates multiple variables at once, works with H-representation, with redundancy check via Linear Programming (LP).

Comparative experimentation

- FME and mpr (this algorithm by Jing, Moreno-Maza, Xie and Yuan [JMXY24]): Eliminates variables sequentially, uses both the H- and V-representations, with redundancy check via the saturation matrix.
- RTC and BPAS (by Jing, Moreno-Maza and Talaashrafi [JMT20]): Eliminates variables sequentially, useing H- and V-representations, with redundancy check via redundancy cones, thus linear algebra over $\mathbb{Q}$.
- cddlib(by Fukuda [cdd]): Eliminates multiple variables at once, works with H-representation, with redundancy check via Linear Programming (LP).
- polylib(by Loechner [Lo99]): Eliminates multiple variables, works with V-representation, can convert between H- and V-representation as needed.

Each test case is associated with a triple (n, m, k) where n and m are the number of variables and inequalities in the input inequality system, while k is the number of vertices and rays.

test case	(n, m, k)	FME C	RTC	FME DD	FME VR	mpr	BPAS	cdd	polylib
32hedron	(6, 32, 11)	8.00	849.00	363.00	14576.00	6.54	16.80	4183.08	1.92
64hedron	(7,64,13)	25.00	3211.00	764.00	218833.00	13.05	52.42	>5min	1.67
francois	(13,27,2304)	2.00	101.00	48.00	65.00	499.92	253.66	388.36	> 5min
francois2	(13,31,384)	2.00	99.00	57.00	89.00	41.80	140.34	55.17	80.63
xavi	(2,7,5)	1.00	29.00	27.00	31.00	0.92	2.91	2.66	0.57
c6.in	(11,17,31)	6.00	29.00	903.00	>5min	9.85	12.72	84.11	5.56
c9.in	(16,18,140)	8.00	701.00	1685.00	>5min	25.08	65.54	151.17	131.53
c10.in	(18,20,142)	9.00	722.00	2038.00	>5min	22.10	98.68	249.02	16.06
e2.in	(6,9,8)	2.00	96.00	128.00	914.00	2.71	4.60	14.42	1.79
e7.in	(4,7,5)	1.00	28.00	46.00	75.00	1.92	3.12	5.24	1.09
e8.in	(3,6,4)	1.00	14.00	69.00	99.00	2.51	2.30	2.56	0.84
e13.in	(6,9,18)	2.00	91.00	133.00	962.00	4.30	3.74	13.13	1.35
e14.in	(5,7,10)	1.00	32.00	153.00	360.00	2.20	1.58	9.91	1.42
S24	(24, 25,25)	1802.00	1996.00	560.00	>5min	23.50	58.80	748.67	17.47
cube	(10, 20,1024)	1.00	63.00	50.00	70.00	81.33	201.92	125.900	161.06
C56	(5, 6,6)	2.00	216.00	91.00	1399.00	3.67	4.09	11.81	0.79
C68	(6, 16,8)	1.00	132.00	50.00	82.00	4.18	10.13	505.00	1.86
C1011	(10, 11,11)	13.00	>5min	920.00	>5min	24.99	115.68	1716.25	9.99
C510	(5, 42,10)	14.00	314.00	355.00	188.00	12.00	40.01	>5min	4.42
T1	(5, 10,38)	3.00	655.00	216.00	2819.00	5.61	16.44	27.42	8.81
T3	(10,12,29)	18.00	>5min	1943.00	>5min	21.29	141.64	288.07	12.07
T5	(5, 10,36)	4.00	1088.00	411.00	3812.00	8.12	15.62	22.92	4.76
T6	(10,20,390)	44901.00	>5min	>5min	>5min	1142.9	23800.11	14937.61	>5min
T7	(5, 8,26)	2.00	670.00	262.00	3559.00	5.81	10.79	13.96	4.00
T9	(10,12,36)	21.00	>5min	2501.00	>5min	36.56	414.53	479.18	100.34
T10	(6, 8,24)	6.00	1228.00	263.00	7190.00	4.58	13.65	18.39	5.27
T12	(5, 11,42)	9.00	959.00	1144.00	8950.00	8.52	19.03	38.65	8.60
R_15_20	(15, 20,1328)	27800.00	>5min	>5min	>5min	28430.40	336035.00	38037.21	>5min

Contents

References I

T.S. Motzkin, H. Raiffa, GL. Thompson, and R.M. Thrall.
The double description method.
Princeton University Press. Princeton, 1953.

Komei Fukuda and Alain Prodon.
Double description method revisited.
Combinatorics and computer science. Springer, 1996.

Komei Fukuda.
cdd, cddplus and cddlib Homepage.
https://people.inf.ethz.ch/fukudak/cdd_home/

Vincent Loechner.
PolyLib: A library for manipulating parameterized polyhedra, 1999.
https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=f9c9cef7b2d1653057348464e0c2edba185c975

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.
Proceedings of ISSAC. ACM, 2024.

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

References III

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.