Efficient detection of redundancies in systems of linear inequalities

Rui-Juan Jing[1] Marc Moreno Maza[2] Yan-Feng Xie[3]
Chun-Ming Yuan[3]

[1]School of Mathematical Sciences, Jiangsu University

[2]Ontario Research Center for Computer Algebra, UWO, London, Ontario

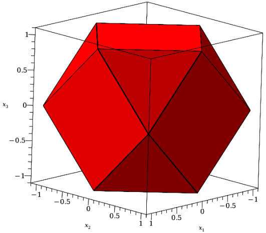
[3]KLMM, Academy of Mathematics and Systems Sciences, Chinese Academy of Sciences, Beijing

July 18, 2024

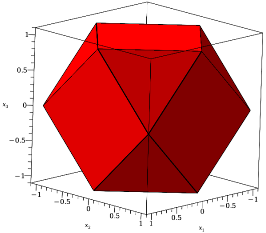
Plan

$$
\begin{cases}
-x_3 \leq 1 \\
-x_1 - x_2 - x_3 \leq 2 \\
-x_1 + x_2 - x_3 \leq 2 \\
x_1 - x_2 - x_3 \leq 2 \\
x_1 + x_2 - x_3 \leq 2 \\
x_3 0 \leq 1 \\
-x_1 - x_2 + x_3 \leq 2 \\
-x_1 + x_2 + x_3 \leq 2 \\
x_1 - x_2 + x_3 \leq 2 \\
x_1 + x_2 + x_3 \leq 2 \\
-x_2 0 \leq 1 \\
x_2 \leq 1 \\
-x_1 \leq 1 \\
x_1 0 \leq 1
\end{cases}
$$

$$
\begin{cases}
-x_3 \le 1 \\
-x_1 - x_2 - x_3 \le 2 \\
-x_1 + x_2 - x_3 \le 2 \\
x_1 - x_2 - x_3 \le 2 \\
x_1 + x_2 - x_3 \le 2 \\
x_3 0 \le 1 \\
-x_1 - x_2 + x_3 \le 2 \\
-x_1 + x_2 + x_3 \le 2 \\
x_1 - x_2 + x_3 \le 2 \\
x_1 + x_2 + x_3 \le 2 \\
-x_2 0 \le 1 \\
x_2 \le 1 \\
-x_1 \le 1 \\
x_1 0 \le 1
\end{cases}
$$

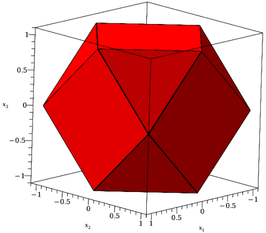


$$
\begin{cases}
-x_3 \leq 1 \\
-x_1 - x_2 - x_3 \leq 2 \\
-x_1 + x_2 - x_3 \leq 2 \\
x_1 - x_2 - x_3 \leq 2 \\
x_1 + x_2 - x_3 \leq 2 \\
x_3 0 \leq 1 \\
-x_1 - x_2 + x_3 \leq 2 \\
-x_1 + x_2 + x_3 \leq 2 \\
x_1 - x_2 + x_3 \leq 2 \\
x_1 + x_2 + x_3 \leq 2 \\
-x_2 0 \leq 1 \\
x_2 \leq 1 \\
-x_1 \leq 1 \\
x_1 0 \leq 1
\end{cases}
$$

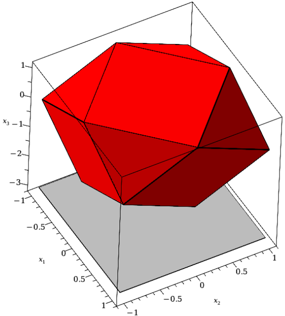


$$
\begin{cases}
0 \leq 1 + x_2 \\
0 \leq 1 - x_2 \\
0 \leq x_1 + 1 \\
0 \leq 1 - x_1
\end{cases}
$$

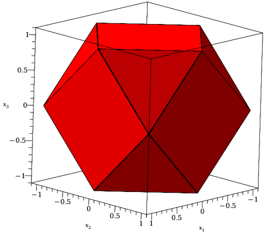
$$
\begin{cases}
-x_3 \leq 1 \\
-x_1 - x_2 - x_3 \leq 2 \\
-x_1 + x_2 - x_3 \leq 2 \\
x_1 - x_2 - x_3 \leq 2 \\
x_1 + x_2 - x_3 \leq 2 \\
x_3 0 \leq 1 \\
-x_1 - x_2 + x_3 \leq 2 \\
-x_1 + x_2 + x_3 \leq 2 \\
x_1 - x_2 + x_3 \leq 2 \\
x_1 + x_2 + x_3 \leq 2 \\
-x_2 0 \leq 1 \\
x_2 \leq 1 \\
-x_1 \leq 1 \\
x_1 0 \leq 1
\end{cases}
$$



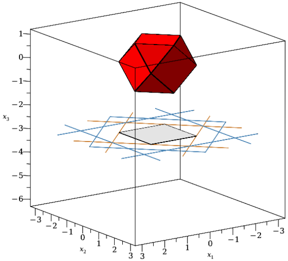
$$
\begin{cases}
0 \leq 1 + x_2 \\
0 \leq 1 - x_2 \\
0 \leq x_1 + 1 \\
0 \leq 1 - x_1
\end{cases}
$$



$$
\begin{cases}
-x_3 \leq 1 \\
-x_1 - x_2 - x_3 \leq 2 \\
-x_1 + x_2 - x_3 \leq 2 \\
x_1 - x_2 - x_3 \leq 2 \\
x_1 + x_2 - x_3 \leq 2 \\
x_3 0 \leq 1 \\
-x_1 - x_2 + x_3 \leq 2 \\
-x_1 + x_2 + x_3 \leq 2 \\
x_1 - x_2 + x_3 \leq 2 \\
x_1 + x_2 + x_3 \leq 2 \\
-x_2 0 \leq 1 \\
x_2 \leq 1 \\
-x_1 \leq 1 \\
x_1 0 \leq 1
\end{cases}
$$



$$
\begin{cases}
0 \leq 1 + x_2 \\
0 \leq 1 - x_2 \\
0 \leq x_1 + 1 \\
0 \leq 1 - x_1
\end{cases}
$$



Application of FME: code generation

```
for(i=0; i<=n; i++){
  c[i] = 0; c[i+n] = 0;
  for(j=0; j<=n; j++)
    c[i+j] += a[i]*b[j];
}
```

skip slide

Application of FME: code generation

```
for(i=0; i<=n; i++){
   c[i] = 0; c[i+n] = 0;
   for(j=0; j<=n; j++)
      c[i+j] += a[i]*b[j];
}
```

Dependence analysis yields:
$(t, p) := (n - j, i + j)$.

skip slide

Application of FME: code generation

```
for(i=0; i<=n; i++){
  c[i] = 0; c[i+n] = 0;
  for(j=0; j<=n; j++)
    c[i+j] += a[i]*b[j];
}
```

Dependence analysis yields:
$(t, p) := (n - j, i + j)$.

$$\begin{cases} 0 \leq & i \\ i \leq & n \\ 0 \leq & j \\ j \leq & n \\ t = & n - j \\ p = & i + j \end{cases}$$

skip slide

Application of FME: code generation

```
for(i=0; i<=n; i++){
  c[i] = 0; c[i+n] = 0;
  for(j=0; j<=n; j++)
    c[i+j] += a[i]*b[j];
}
```

Dependence analysis yields:
$(t, p) := (n - j, i + j)$.

$$\begin{cases} 0 \le & i \\ i \le & n \\ 0 \le & j \\ j \le & n \\ t = & n - j \\ p = & i + j \end{cases}$$

FME reorders $p > t > i > j > n$ to $i > j > t > p > n$, thus eliminating i, j.

skip slide

Application of FME: code generation

```
for(i=0; i<=n; i++){
  c[i] = 0; c[i+n] = 0;
  for(j=0; j<=n; j++)
    c[i+j] += a[i]*b[j];
}
```

Dependence analysis yields:
$(t, p) := (n - j, i + j)$.

$$\begin{cases} 0 \le & i \\ i \le & n \\ 0 \le & j \\ j \le & n \\ t = & n - j \\ p = & i + j \end{cases}$$

$$\begin{cases} i = & p + t - n \\ j = & -t + n \\ t \ge & \max(0, -p + n) \\ t \le & \min(n, -p + 2n) \\ 0 \le & p \\ p \le & 2n \\ 0 \le & n. \end{cases}$$

FME reorders $p > t > i > j > n$ to $i > j > t > p > n$, thus eliminating i, j.

skip slide

Application of FME: code generation

```
for(i=0; i<=n; i++){
  c[i] = 0; c[i+n] = 0;
  for(j=0; j<=n; j++)
    c[i+j] += a[i]*b[j];
}
```

Dependence analysis yields: $(t,p) := (n-j, i+j)$.

$$\begin{cases} 0 \leq & i \\ i \leq & n \\ 0 \leq & j \\ j \leq & n \\ t = & n-j \\ p = & i+j \end{cases}$$

The new representation allows us to generate the multithreaded code.

$$\begin{cases} i = & p+t-n \\ j = & -t+n \\ t \geq & \max(0, -p+n) \\ t \leq & \min(n, -p+2n) \\ 0 \leq & p \\ p \leq & 2n \\ 0 \leq & n. \end{cases}$$

FME reorders $p > t > i > j > n$ to $i > j > t > p > n$, thus eliminating i, j.

skip slide

Application of FME: code generation

```
for(i=0; i<=n; i++){
  c[i] = 0; c[i+n] = 0;
  for(j=0; j<=n; j++)
    c[i+j] += a[i]*b[j];
}
```

```
parallel_for (p=0; p<=2*n; p++){
  c[p] = 0;
  for (t=max(0,n-p);
       t<=min(n,2*n-p);t++)
    c[p] += A[t+p-n] * B[n-t];
}
```

Dependence analysis yields: $(t, p) := (n - j, i + j)$.

The new representation allows us to generate the multithreaded code.

$$\begin{cases} 0 \leq & i \\ i \leq & n \\ 0 \leq & j \\ j \leq & n \\ t = & n - j \\ p = & i + j \end{cases}$$

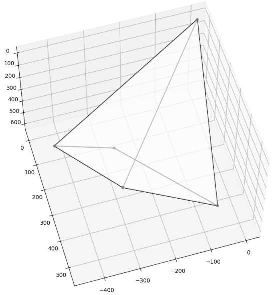
$$\begin{cases} i = & p + t - n \\ j = & -t + n \\ t \geq & \max(0, -p + n) \\ t \leq & \min(n, -p + 2n) \\ 0 \leq & p \\ p \leq & 2n \\ 0 \leq & n. \end{cases}$$

FME reorders $p > t > i > j > n$ to $i > j > t > p > n$, thus eliminating i, j.

Application of FME: computing integer hulls (1/3)

The input polyhedral set:

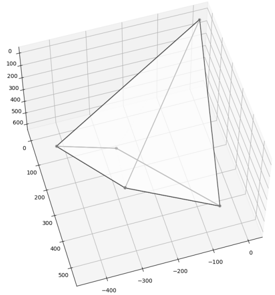
$$\left\{\begin{array}{ccc} -98877x_1 - 189663x_2 - 1798x_3 & \leq & 705915 \\ -10109x_1 - 5958x_2 - 14601x_3 & \leq & 31333 \\ -5405x_1 + 4965x_2 + 3870x_3 & \leq & 4303504 \\ 729x_1 - 117x_2 + 350x_3 & \leq & 4561 \\ 677x_1 + 465x_2 - 540x_3 & \leq & 3489 \end{array}\right.$$



Application of FME: computing integer hulls (1/3)

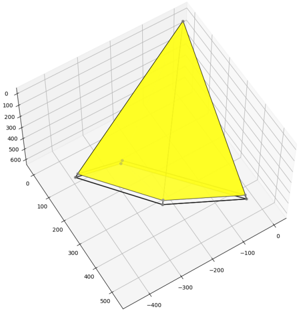
The input polyhedral set:

$$\left\{ \begin{array}{rcl} -98877x_1 - 189663x_2 - 1798x_3 & \leq & 705915 \\ -10109x_1 - 5958x_2 - 14601x_3 & \leq & 31333 \\ -5405x_1 + 4965x_2 + 3870x_3 & \leq & 4303504 \\ 729x_1 - 117x_2 + 350x_3 & \leq & 4561 \\ 677x_1 + 465x_2 - 540x_3 & \leq & 3489 \end{array} \right.$$

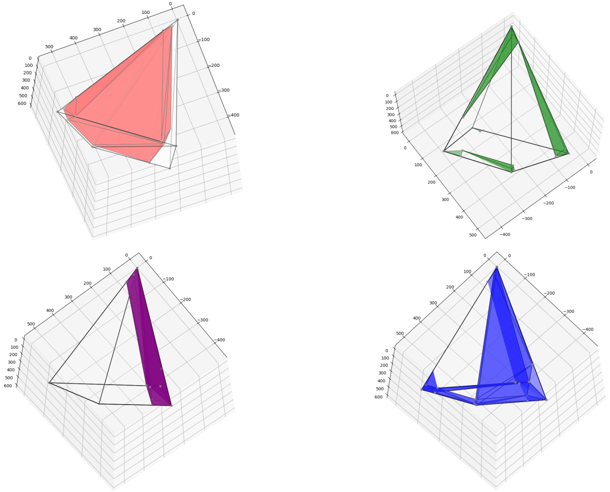


Normalization (leaves the integer hull unchanged):

$$\left\{ \begin{array}{rcl} -98877x_1 - 189663x_2 - 1798x_3 & \leq & 705915 \\ -10109x_1 - 5958x_2 - 14601x_3 & \leq & 31333 \\ -1081x_1 + 993x_2 + 774x_3 & \leq & 860700 \\ 729x_1 - 117x_2 + 350x_3 & \leq & 4561 \\ 677x_1 + 465x_2 - 540x_3 & \leq & 3489 \end{array} \right.$$



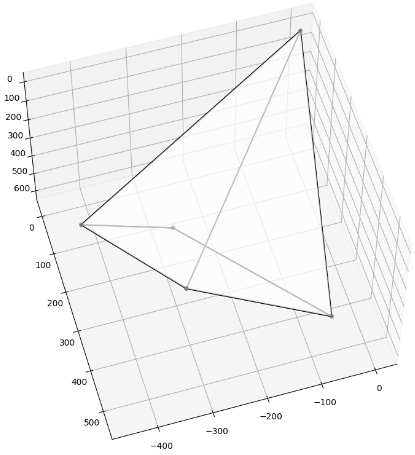
Application of FME: computing integer hulls (2/3)



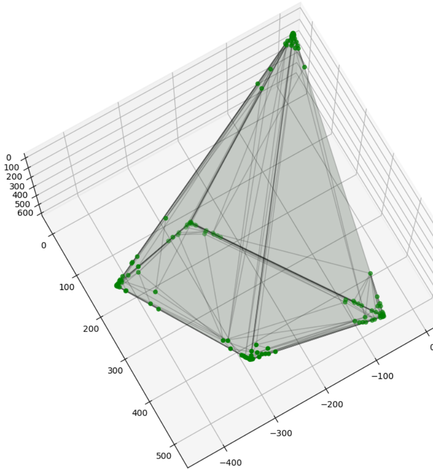
1. The **red** is an approximation of the integer hull of the input.
2. The integer hulls of border regions (green, blue, purple) are brute-force computed via FME.
3. Then `QuickHull` is applied to obtain the integer hull of the input.

Application of FME: computing integer hulls (3/3)

The input has only 5 vertices.



Its integer hull has 139 vertices.



All details are in https://ir.lib.uwo.ca/etd/8985/ and in https://doi.org/10.1007/978-3-031-14788-3_14

Plan

Polyhedral sets

1. A *polyhedral set* P is any $\{\mathbf{x} \mid A\mathbf{x} \le \mathbf{b}\}$, 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.

Polyhedral sets

1. A *polyhedral set* P is any $\{\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}\}$, 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.
2. P is *full-dimensional* whenever $\dim(P) = n$

Polyhedral sets

1. A *polyhedral set* P is any $\{\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}\}$, 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.
2. P is *full-dimensional* whenever $\dim(P) = n$
3. An inequality ℓ of $A\mathbf{x} \leq \mathbf{b}$ is an *implicit equation* if $\mathbf{a}^t\mathbf{x} = b$ holds for all $\mathbf{x} \in P$.

Polyhedral sets

1. A *polyhedral set* P is any $\{\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}\}$, 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.
2. P is *full-dimensional* whenever $\dim(P) = n$
3. An inequality ℓ of $A\mathbf{x} \leq \mathbf{b}$ is an *implicit equation* if $\mathbf{a}^t\mathbf{x} = b$ holds for all $\mathbf{x} \in P$.
4. Thus, P is full-dimensional iff $A\mathbf{x} \leq \mathbf{b}$ has no implicit equation.

Polyhedral sets

1. A *polyhedral set* P is any $\{\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}\}$, 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.
2. P is *full-dimensional* whenever $\dim(P) = n$
3. An inequality ℓ of $A\mathbf{x} \leq \mathbf{b}$ is an *implicit equation* if $\mathbf{a}^t\mathbf{x} = b$ holds for all $\mathbf{x} \in P$.
4. Thus, P is full-dimensional iff $A\mathbf{x} \leq \mathbf{b}$ has no implicit equation.
5. The polyhedron P is said *pointed*, if A is full column rank.

Polyhedral sets

1. A *polyhedral set* P is any $\{\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}\}$, 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.
2. P is *full-dimensional* whenever $\dim(P) = n$
3. An inequality ℓ of $A\mathbf{x} \leq \mathbf{b}$ is an *implicit equation* if $\mathbf{a}^t\mathbf{x} = b$ holds for all $\mathbf{x} \in P$.
4. Thus, P is full-dimensional iff $A\mathbf{x} \leq \mathbf{b}$ has no implicit equation.
5. The polyhedron P is said *pointed*, if A is full column rank.
6. From now, P is full-dimensional and pointed.

Polyhedral sets

1. A *polyhedral set* P is any $\{\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}\}$, 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.
2. P is *full-dimensional* whenever $\dim(P) = n$
3. An inequality ℓ of $A\mathbf{x} \leq \mathbf{b}$ is an *implicit equation* if $\mathbf{a}^t\mathbf{x} = b$ holds for all $\mathbf{x} \in P$.
4. Thus, P is full-dimensional iff $A\mathbf{x} \leq \mathbf{b}$ has no implicit equation.
5. The polyhedron P is said *pointed*, if A is full column rank.
6. From now, P is full-dimensional and pointed.
7. Fixing $F : A\mathbf{x} \leq \mathbf{b}$ an H-representation of P, a *face* of P is any intersection of P with the solution set of sub-system of F.

Polyhedral sets

1. A *polyhedral set* P is any $\{\mathbf{x} \mid A\mathbf{x} \le \mathbf{b}\}$, 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.
2. P is *full-dimensional* whenever $\dim(P) = n$
3. An inequality ℓ of $A\mathbf{x} \le \mathbf{b}$ is an *implicit equation* if $\mathbf{a}^t\mathbf{x} = b$ holds for all $\mathbf{x} \in P$.
4. Thus, P is full-dimensional iff $A\mathbf{x} \le \mathbf{b}$ has no implicit equation.
5. The polyhedron P is said *pointed*, if A is full column rank.
6. From now, P is full-dimensional and pointed.
7. Fixing $F : A\mathbf{x} \le \mathbf{b}$ an H-representation of P, a *face* of P is any intersection of P with the solution set of sub-system of F.
8. A *vertex* (resp. *facet*) is a face of dimension 0 (resp. $n - 1$).

Polyhedral sets

1. A *polyhedral set* P is any $\{\mathbf{x} \mid A\mathbf{x} \le \mathbf{b}\}$, 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.
2. P is *full-dimensional* whenever $\dim(P) = n$
3. An inequality ℓ of $A\mathbf{x} \le \mathbf{b}$ is an *implicit equation* if $\mathbf{a}^t\mathbf{x} = b$ holds for all $\mathbf{x} \in P$.
4. Thus, P is full-dimensional iff $A\mathbf{x} \le \mathbf{b}$ has no implicit equation.
5. The polyhedron P is said *pointed*, if A is full column rank.
6. From now, P is full-dimensional and pointed.
7. Fixing $F : A\mathbf{x} \le \mathbf{b}$ an H-representation of P, a *face* of P is any intersection of P with the solution set of sub-system of F.
8. A *vertex* (resp. *facet*) is a face of dimension 0 (resp. $n - 1$).
9. The *characteristic cone* of P is the polyhedral cone $\mathrm{CharCone}(P)$ represented by $\{A\mathbf{x} \le \mathbf{0}\}$.

Polyhedral sets

1. A *polyhedral set* P is any $\{\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}\}$, 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.
2. P is *full-dimensional* whenever $\dim(P) = n$
3. An inequality ℓ of $A\mathbf{x} \leq \mathbf{b}$ is an *implicit equation* if $\mathbf{a}^t\mathbf{x} = b$ holds for all $\mathbf{x} \in P$.
4. Thus, P is full-dimensional iff $A\mathbf{x} \leq \mathbf{b}$ has no implicit equation.
5. The polyhedron P is said *pointed*, if A is full column rank.
6. From now, P is full-dimensional and pointed.
7. Fixing $F : A\mathbf{x} \leq \mathbf{b}$ an H-representation of P, a *face* of P is any intersection of P with the solution set of sub-system of F.
8. A *vertex* (resp. *facet*) is a face of dimension 0 (resp. $n - 1$).
9. The *characteristic cone* of P is the polyhedral cone $\mathrm{CharCone}(P)$ represented by $\{A\mathbf{x} \leq \mathbf{0}\}$.
10. Every polyhedral cone has a unique representation as a conical hull of its extremal generators, called the *extreme rays* of P.

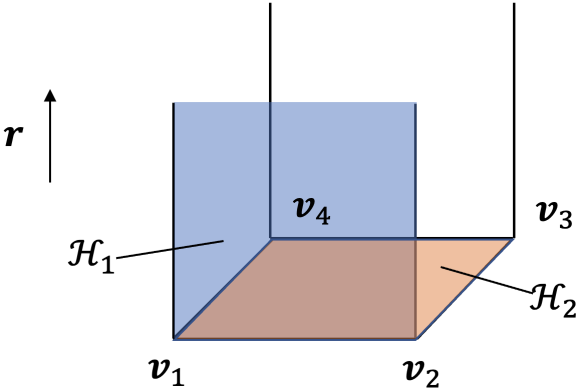
Polyhedral sets

1. A *polyhedral set* P is any $\{\mathbf{x} \mid A\mathbf{x} \le \mathbf{b}\}$, 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.
2. P is *full-dimensional* whenever $\dim(P) = n$
3. An inequality ℓ of $A\mathbf{x} \le \mathbf{b}$ is an *implicit equation* if $\mathbf{a}^t\mathbf{x} = b$ holds for all $\mathbf{x} \in P$.
4. Thus, P is full-dimensional iff $A\mathbf{x} \le \mathbf{b}$ has no implicit equation.
5. The polyhedron P is said *pointed*, if A is full column rank.
6. From now, P is full-dimensional and pointed.
7. Fixing $F : A\mathbf{x} \le \mathbf{b}$ an H-representation of P, a *face* of P is any intersection of P with the solution set of sub-system of F.
8. A *vertex* (resp. *facet*) is a face of dimension 0 (resp. $n-1$).
9. The *characteristic cone* of P is the polyhedral cone $\mathrm{CharCone}(P)$ represented by $\{A\mathbf{x} \le \mathbf{0}\}$.
10. Every polyhedral cone has a unique representation as a conical hull of its extremal generators, called the *extreme rays* of P.
11. Since P is pointed, an extreme ray of P is a one-dimensional face of $\mathrm{CharCone}(P)$.

Polyhedral sets

1. A *polyhedral set* P is any $\{\mathbf{x} \mid A\mathbf{x} \leq \mathbf{b}\}$, 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.
2. P is *full-dimensional* whenever $\dim(P) = n$
3. An inequality ℓ of $A\mathbf{x} \leq \mathbf{b}$ is an *implicit equation* if $\mathbf{a}^t\mathbf{x} = b$ holds for all $\mathbf{x} \in P$.
4. Thus, P is full-dimensional iff $A\mathbf{x} \leq \mathbf{b}$ has no implicit equation.
5. The polyhedron P is said *pointed*, if A is full column rank.
6. From now, P is full-dimensional and pointed.
7. Fixing $F : A\mathbf{x} \leq \mathbf{b}$ an H-representation of P, a *face* of P is any intersection of P with the solution set of sub-system of F.
8. A *vertex* (resp. *facet*) is a face of dimension 0 (resp. $n-1$).
9. The *characteristic cone* of P is the polyhedral cone $\mathrm{CharCone}(P)$ represented by $\{A\mathbf{x} \leq \mathbf{0}\}$.
10. Every polyhedral cone has a unique representation as a conical hull of its extremal generators, called the *extreme rays* of P.
11. Since P is pointed, an extreme ray of P is a one-dimensional face of $\mathrm{CharCone}(P)$.
12. Let V and R denote the set of vertices and extreme rays of P. Then, the pair $\mathcal{VR}(F) := (V, R)$ is called a *V-representation* of P.

An unbounded polyhedral set and its representations



The open cube $P := \{(x, y, z) \mid -z \leq 1, 0 \leq x \leq 1, 0 \leq y \leq 1\}$ shown above has 4 vertices $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4$ and extreme ray $\mathbf{r}$.

Redundant inequalities, the saturation matrix

1. Recall $F : A\mathbf{x} \leq \mathbf{b}$ is an H-representation of our polyhedral set P.

Redundant inequalities, the saturation matrix

1. Recall $F : A\mathbf{x} \leq \mathbf{b}$ is an H-representation of our polyhedral set P.
2. Fix an inequality $\ell : \mathbf{a}^t\mathbf{x} \leq b$ of F

Redundant inequalities, the saturation matrix

1. Recall $F : A\mathbf{x} \leq \mathbf{b}$ is an H-representation of our polyhedral set P.
2. Fix an inequality $\ell : \mathbf{a}^t\mathbf{x} \leq b$ of F
3. Denote by $\mathcal{H}_\ell$ the hyperplane $\mathbf{a}^t\mathbf{x} = b$.

Redundant inequalities, the saturation matrix

1. Recall $F : A\mathbf{x} \leq \mathbf{b}$ is an H-representation of our polyhedral set P.
2. Fix an inequality $\ell : \mathbf{a}^t\mathbf{x} \leq b$ of F
3. Denote by $\mathcal{H}_\ell$ the hyperplane $\mathbf{a}^t\mathbf{x} = b$.
4. Recall V and R are the vertices and rays of P. Let $k := \#\mathcal{VR}(F)$.

Redundant inequalities, the saturation matrix

1. Recall $F : A\mathbf{x} \leq \mathbf{b}$ is an H-representation of our polyhedral set P.
2. Fix an inequality $\ell : \mathbf{a}^t\mathbf{x} \leq b$ of F
3. Denote by $\mathcal{H}_\ell$ the hyperplane $\mathbf{a}^t\mathbf{x} = b$.
4. Recall V and R are the vertices and rays of P. Let $k := \#\mathcal{VR}(F)$.

Definition

The inequality ℓ of F is

Redundant inequalities, the saturation matrix

1. Recall $F : A\mathbf{x} \leq \mathbf{b}$ is an H-representation of our polyhedral set P.
2. Fix an inequality $\ell : \mathbf{a}^t\mathbf{x} \leq b$ of F
3. Denote by $\mathcal{H}_\ell$ the hyperplane $\mathbf{a}^t\mathbf{x} = b$.
4. Recall V and R are the vertices and rays of P. Let $k := \#\mathcal{VR}(F)$.

Definition

The inequality ℓ of F is

- *redundant* in F, if $F \setminus \{\ell\}$ still defines P,

Redundant inequalities, the saturation matrix

1. Recall $F : A\mathbf{x} \leq \mathbf{b}$ is an H-representation of our polyhedral set P.
2. Fix an inequality $\ell : \mathbf{a}^t\mathbf{x} \leq b$ of F
3. Denote by $\mathcal{H}_\ell$ the hyperplane $\mathbf{a}^t\mathbf{x} = b$.
4. Recall V and R are the vertices and rays of P. Let $k := \#\mathcal{VR}(F)$.

Definition

The inequality ℓ of F is

- *redundant* in F, if $F \setminus \{\ell\}$ still defines P,
- *strongly redundant* in F, if $\mathbf{a}^t\mathbf{x} < b$ holds for all $\mathbf{x} \in P$,

Redundant inequalities, the saturation matrix

1. Recall $F : A\mathbf{x} \le \mathbf{b}$ is an H-representation of our polyhedral set P.
2. Fix an inequality $\ell : \mathbf{a}^t\mathbf{x} \le b$ of F
3. Denote by $\mathcal{H}_\ell$ the hyperplane $\mathbf{a}^t\mathbf{x} = b$.
4. Recall V and R are the vertices and rays of P. Let $k := \#\mathcal{VR}(F)$.

Definition

The inequality ℓ of F is

- *redundant* in F, if $F \smallsetminus \{\ell\}$ still defines P,
- *strongly redundant* in F, if $\mathbf{a}^t\mathbf{x} < b$ holds for all $\mathbf{x} \in P$,
- *weakly redundant* if it is redundant and $\mathbf{a}^t\mathbf{x} = b$ holds for some $\mathbf{x} \in P$.

Redundant inequalities, the saturation matrix

1. Recall $F : A\mathbf{x} \leq \mathbf{b}$ is an H-representation of our polyhedral set P.
2. Fix an inequality $\ell : \mathbf{a}^t\mathbf{x} \leq b$ of F
3. Denote by $\mathcal{H}_\ell$ the hyperplane $\mathbf{a}^t\mathbf{x} = b$.
4. Recall V and R are the vertices and rays of P. Let $k := \#\mathcal{VR}(F)$.

Definition

The inequality ℓ of F is

- *redundant* in F, if $F \setminus \{\ell\}$ still defines P,
- *strongly redundant* in F, if $\mathbf{a}^t\mathbf{x} < b$ holds for all $\mathbf{x} \in P$,
- *weakly redundant* if it is redundant and $\mathbf{a}^t\mathbf{x} = b$ holds for some $\mathbf{x} \in P$.

Definition

Redundant inequalities, the saturation matrix

1. Recall $F : A\mathbf{x} \leq \mathbf{b}$ is an H-representation of our polyhedral set P.
2. Fix an inequality $\ell : \mathbf{a}^t\mathbf{x} \leq b$ of F
3. Denote by $\mathcal{H}_\ell$ the hyperplane $\mathbf{a}^t\mathbf{x} = b$.
4. Recall V and R are the vertices and rays of P. Let $k := \#\mathcal{VR}(F)$.

Definition

The inequality ℓ of F is

- *redundant* in F, if $F \setminus \{\ell\}$ still defines P,
- *strongly redundant* in F, if $\mathbf{a}^t\mathbf{x} < b$ holds for all $\mathbf{x} \in P$,
- *weakly redundant* if it is redundant and $\mathbf{a}^t\mathbf{x} = b$ holds for some $\mathbf{x} \in P$.

Definition

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

Redundant inequalities, the saturation matrix

1. Recall $F : A\mathbf{x} \leq \mathbf{b}$ is an H-representation of our polyhedral set P.
2. Fix an inequality $\ell : \mathbf{a}^t\mathbf{x} \leq b$ of F
3. Denote by $\mathcal{H}_\ell$ the hyperplane $\mathbf{a}^t\mathbf{x} = b$.
4. Recall V and R are the vertices and rays of P. Let $k := \#\mathcal{VR}(F)$.

Definition

The inequality ℓ of F is

- *redundant* in F, if $F \setminus \{\ell\}$ still defines P,
- *strongly redundant* in F, if $\mathbf{a}^t\mathbf{x} < b$ holds for all $\mathbf{x} \in P$,
- *weakly redundant* if it is redundant and $\mathbf{a}^t\mathbf{x} = b$ holds for some $\mathbf{x} \in P$.

Definition

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

Redundant inequalities, the saturation matrix

1. Recall $F : A\mathbf{x} \le \mathbf{b}$ is an H-representation of our polyhedral set P.
2. Fix an inequality $\ell : \mathbf{a}^t\mathbf{x} \le b$ of F
3. Denote by $\mathcal{H}_\ell$ the hyperplane $\mathbf{a}^t\mathbf{x} = b$.
4. Recall V and R are the vertices and rays of P. Let $k := \#\mathcal{VR}(F)$.

Definition

The inequality ℓ of F is

- *redundant* in F, if $F \setminus \{\ell\}$ still defines P,
- *strongly redundant* in F, if $\mathbf{a}^t\mathbf{x} < b$ holds for all $\mathbf{x} \in P$,
- *weakly redundant* if it is redundant and $\mathbf{a}^t\mathbf{x} = b$ holds for some $\mathbf{x} \in P$.

Definition

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

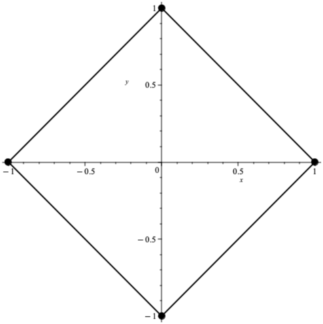
The *saturation matrix* of F is the $0-1$ matrix $S \in \mathbb{Q}^{m\times k}$, where $S_{i,j} = 1$ iff the j-th element of $\mathcal{VR}(F)$ saturates the i-th inequality of F.

A bounded polyhedral set and its the saturation matrix

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



Basic redundancy check

1. Denote by $\mathcal{VR}(F) \cap \mathcal{H}_\ell$ the vertices and rays in $\mathcal{VR}(F)$ saturating the hyperplane $\mathcal{H}_\ell$.

Basic redundancy check

1. Denote by $\mathcal{VR}(F) \cap \mathcal{H}_\ell$ the vertices and rays in $\mathcal{VR}(F)$ saturating the hyperplane $\mathcal{H}_\ell$.
2. Write $\mathcal{VR}(F) \cap \mathcal{H}_\ell := (\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_t\}, \{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_s\})$, where $\mathbf{v}_i$'s are vertices and $\mathbf{r}_j$'s are rays.

Basic redundancy check

1. Denote by $\mathcal{VR}(F) \cap \mathcal{H}_\ell$ the vertices and rays in $\mathcal{VR}(F)$ saturating the hyperplane $\mathcal{H}_\ell$.
2. Write $\mathcal{VR}(F) \cap \mathcal{H}_\ell := (\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_t\}, \{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_s\})$, where $\mathbf{v}_i$'s are vertices and $\mathbf{r}_j$'s are rays.
3. The *affine rank* of $\mathcal{VR}(F) \cap \mathcal{H}_\ell$ is the rank of the matrix

$$[\mathbf{v}_2 - \mathbf{v}_1, \mathbf{v}_3 - \mathbf{v}_1, \ldots, \mathbf{v}_t - \mathbf{v}_1, \mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_s].$$

Basic redundancy check

1. Denote by $\mathcal{VR}(F) \cap \mathcal{H}_\ell$ the vertices and rays in $\mathcal{VR}(F)$ saturating the hyperplane $\mathcal{H}_\ell$.
2. Write $\mathcal{VR}(F) \cap \mathcal{H}_\ell := (\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_t\}, \{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_s\})$, where $\mathbf{v}_i$'s are vertices and $\mathbf{r}_j$'s are rays.
3. The *affine rank* of $\mathcal{VR}(F) \cap \mathcal{H}_\ell$ is the rank of the matrix
$$[\mathbf{v}_2 - \mathbf{v}_1, \mathbf{v}_3 - \mathbf{v}_1, \ldots, \mathbf{v}_t - \mathbf{v}_1, \mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_s].$$

With these notations, we have the following lemma. We note that any permutation $(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_t)$ would leave this result unchanged.

Basic redundancy check

1. Denote by $\mathcal{VR}(F) \cap \mathcal{H}_\ell$ the vertices and rays in $\mathcal{VR}(F)$ saturating the hyperplane $\mathcal{H}_\ell$.
2. Write $\mathcal{VR}(F) \cap \mathcal{H}_\ell := (\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_t\}, \{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_s\})$, where $\mathbf{v}_i$'s are vertices and $\mathbf{r}_j$'s are rays.
3. The *affine rank* of $\mathcal{VR}(F) \cap \mathcal{H}_\ell$ is the rank of the matrix

$$[\mathbf{v}_2 - \mathbf{v}_1, \mathbf{v}_3 - \mathbf{v}_1, \ldots, \mathbf{v}_t - \mathbf{v}_1, \mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_s].$$

With these notations, we have the following lemma. We note that any permutation $(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_t)$ would leave this result unchanged.

Lemma

Assume the inequalities of F define hyperplanes that are pairwise different. Then, the following conditions are equivalent:

Basic redundancy check

1. Denote by $\mathcal{VR}(F) \cap \mathcal{H}_\ell$ the vertices and rays in $\mathcal{VR}(F)$ saturating the hyperplane $\mathcal{H}_\ell$.
2. Write $\mathcal{VR}(F) \cap \mathcal{H}_\ell := (\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_t\}, \{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_s\})$, where $\mathbf{v}_i$'s are vertices and $\mathbf{r}_j$'s are rays.
3. The *affine rank* of $\mathcal{VR}(F) \cap \mathcal{H}_\ell$ is the rank of the matrix

$$[\mathbf{v}_2 - \mathbf{v}_1, \mathbf{v}_3 - \mathbf{v}_1, \ldots, \mathbf{v}_t - \mathbf{v}_1, \mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_s].$$

With these notations, we have the following lemma. We note that any permutation $(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_t)$ would leave this result unchanged.

Lemma

Assume the inequalities of F define hyperplanes that are pairwise different. Then, the following conditions are equivalent:

1. The inequality $\ell \in F$ is irredundant,

Basic redundancy check

1. Denote by $\mathcal{VR}(F) \cap \mathcal{H}_\ell$ the vertices and rays in $\mathcal{VR}(F)$ saturating the hyperplane $\mathcal{H}_\ell$.
2. Write $\mathcal{VR}(F) \cap \mathcal{H}_\ell := (\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_t\}, \{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_s\})$, where $\mathbf{v}_i$'s are vertices and $\mathbf{r}_j$'s are rays.
3. The *affine rank* of $\mathcal{VR}(F) \cap \mathcal{H}_\ell$ is the rank of the matrix

$$[\mathbf{v}_2 - \mathbf{v}_1, \mathbf{v}_3 - \mathbf{v}_1, \ldots, \mathbf{v}_t - \mathbf{v}_1, \mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_s].$$

With these notations, we have the following lemma. We note that any permutation $(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_t)$ would leave this result unchanged.

Lemma

Assume the inequalities of F define hyperplanes that are pairwise different. Then, the following conditions are equivalent:

1. The inequality $\ell \in F$ is irredundant,
2. $\mathcal{H}_\ell \cap P$ is a facet of the polyhedron P.

Basic redundancy check

1. Denote by $\mathcal{VR}(F) \cap \mathcal{H}_\ell$ the vertices and rays in $\mathcal{VR}(F)$ saturating the hyperplane $\mathcal{H}_\ell$.
2. Write $\mathcal{VR}(F) \cap \mathcal{H}_\ell := (\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_t\}, \{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_s\})$, where $\mathbf{v}_i$'s are vertices and $\mathbf{r}_j$'s are rays.
3. The *affine rank* of $\mathcal{VR}(F) \cap \mathcal{H}_\ell$ is the rank of the matrix

$$[\mathbf{v}_2 - \mathbf{v}_1, \mathbf{v}_3 - \mathbf{v}_1, \ldots, \mathbf{v}_t - \mathbf{v}_1, \mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_s].$$

With these notations, we have the following lemma. We note that any permutation $(\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_t)$ would leave this result unchanged.

Lemma

Assume the inequalities of F define hyperplanes that are pairwise different. Then, the following conditions are equivalent:

1. The inequality $\ell \in F$ is irredundant,
2. $\mathcal{H}_\ell \cap P$ is a facet of the polyhedron P.
3. The affine rank of $\mathcal{VR}(F) \cap \mathcal{H}_\ell$ equals to $n - 1$.

Plan

Redundancy tests (1/2)

1. For any inequality ℓ, the set $\mathcal{S}^{\mathcal{VR}}(\ell)$ collects all the vertices and rays saturating ℓ.

Redundancy tests (1/2)

1. For any inequality ℓ, the set $\mathcal{S}^{\mathcal{VR}}(\ell)$ collects all the vertices and rays saturating ℓ.
2. For any ray or vertex $\mathbf{u}$, the set $\mathcal{S}^{\mathcal{H}}(\mathbf{u})$ collects all the hyperplanes saturated by $\mathbf{u}$.

Redundancy tests (1/2)

1. For any inequality ℓ, the set $\mathcal{S}^{\mathcal{VR}}(\ell)$ collects all the vertices and rays saturating ℓ.
2. For any ray or vertex $\mathbf{u}$, the set $\mathcal{S}^{\mathcal{H}}(\mathbf{u})$ collects all the hyperplanes saturated by $\mathbf{u}$.
3. Fix an inequality ℓ of F.

Redundancy tests (1/2)

1. For any inequality ℓ, the set $\mathcal{S}^{\mathcal{VR}}(\ell)$ collects all the vertices and rays saturating ℓ.
2. For any ray or vertex $\mathbf{u}$, the set $\mathcal{S}^{\mathcal{H}}(\mathbf{u})$ collects all the hyperplanes saturated by $\mathbf{u}$.
3. Fix an inequality ℓ of F.
4. Hence, the set

$$\mathcal{S}^{\mathcal{H}}(\mathcal{S}^{\mathcal{VR}}(\ell)) := \bigcap_{\mathbf{u} \in \mathcal{S}^{\mathcal{VR}}(\ell)} \mathcal{S}^{\mathcal{H}}(\mathbf{u}),$$

   is the set of all inequalities saturated by all the vertices or rays saturating ℓ.

Redundancy tests (1/2)

1. For any inequality ℓ, the set $\mathcal{S}^{\mathcal{VR}}(\ell)$ collects all the vertices and rays saturating ℓ.
2. For any ray or vertex $\mathbf{u}$, the set $\mathcal{S}^{\mathcal{H}}(\mathbf{u})$ collects all the hyperplanes saturated by $\mathbf{u}$.
3. Fix an inequality ℓ of F.
4. Hence, the set

$$\mathcal{S}^{\mathcal{H}}(\mathcal{S}^{\mathcal{VR}}(\ell)) := \bigcap_{\mathbf{u} \in \mathcal{S}^{\mathcal{VR}}(\ell)} \mathcal{S}^{\mathcal{H}}(\mathbf{u}),$$

   is the set of all inequalities saturated by all the vertices or rays saturating ℓ.

Theorem

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

Redundancy tests (1/2)

1. For any inequality ℓ, the set $\mathcal{S}^{\mathcal{VR}}(\ell)$ collects all the vertices and rays saturating ℓ.
2. For any ray or vertex $\mathbf{u}$, the set $\mathcal{S}^{\mathcal{H}}(\mathbf{u})$ collects all the hyperplanes saturated by $\mathbf{u}$.
3. Fix an inequality ℓ of F.
4. Hence, the set

$$\mathcal{S}^{\mathcal{H}}(\mathcal{S}^{\mathcal{VR}}(\ell)) := \bigcap_{\mathbf{u} \in \mathcal{S}^{\mathcal{VR}}(\ell)} \mathcal{S}^{\mathcal{H}}(\mathbf{u}),$$

   is the set of all inequalities saturated by all the vertices or rays saturating ℓ.

Theorem

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

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

Redundancy tests (1/2)

1. For any inequality ℓ, the set $\mathcal{S}^{\mathcal{VR}}(\ell)$ collects all the vertices and rays saturating ℓ.
2. For any ray or vertex $\mathbf{u}$, the set $\mathcal{S}^{\mathcal{H}}(\mathbf{u})$ collects all the hyperplanes saturated by $\mathbf{u}$.
3. Fix an inequality ℓ of F.
4. Hence, the set

$$\mathcal{S}^{\mathcal{H}}(\mathcal{S}^{\mathcal{VR}}(\ell)) := \bigcap_{\mathbf{u} \in \mathcal{S}^{\mathcal{VR}}(\ell)} \mathcal{S}^{\mathcal{H}}(\mathbf{u}),$$

is the set of all inequalities saturated by all the vertices or rays saturating ℓ.

Theorem

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

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

Redundancy tests (1/2)

1. For any inequality ℓ, the set $\mathcal{S}^{\mathcal{VR}}(\ell)$ collects all the vertices and rays saturating ℓ.
2. For any ray or vertex $\mathbf{u}$, the set $\mathcal{S}^{\mathcal{H}}(\mathbf{u})$ collects all the hyperplanes saturated by $\mathbf{u}$.
3. Fix an inequality ℓ of F.
4. Hence, the set

$$\mathcal{S}^{\mathcal{H}}(\mathcal{S}^{\mathcal{VR}}(\ell)) := \bigcap_{\mathbf{u} \in \mathcal{S}^{\mathcal{VR}}(\ell)} \mathcal{S}^{\mathcal{H}}(\mathbf{u}),$$

is the set of all inequalities saturated by all the vertices or rays saturating ℓ.

Theorem

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

1. *The inequality ℓ is strongly redundant in F iff $\mathcal{S}^{\mathcal{VR}}(\ell)$ 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 iff the set $\mathcal{S}^{\mathcal{H}}(\mathcal{S}^{\mathcal{VR}}(\ell)) \setminus \{\ell\}$ is not empty.*

Redundancy tests (2/2)

Theorem (Recall from previous slide)

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

1. *The inequality ℓ is strongly redundant in F iff $\mathcal{S}^{\mathcal{VR}}(\ell)$ 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 iff the set $\mathcal{S}^{\mathcal{H}}(\mathcal{S}^{\mathcal{VR}}(\ell)) \setminus \{\ell\}$ is not empty.*

Redundancy tests (2/2)

Theorem (Recall from previous slide)

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

1. *The inequality ℓ is strongly redundant in F iff $\mathcal{S}^{\mathcal{VR}}(\ell)$ 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 iff the set $\mathcal{S}^{\mathcal{H}}(\mathcal{S}^{\mathcal{VR}}(\ell)) \setminus \{\ell\}$ is not empty.*

- Denote by satM(F) the saturation matrix of F.

Redundancy tests (2/2)

Theorem (Recall from previous slide)

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

1. *The inequality ℓ is strongly redundant in F iff $\mathcal{S}^{\mathcal{VR}}(\ell)$ 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 iff the set $\mathcal{S}^{\mathcal{H}}(\mathcal{S}^{\mathcal{VR}}(\ell)) \setminus \{\ell\}$ is not empty.*

- Denote by $\mathrm{satM}(F)$ the saturation matrix of F.
- $\mathrm{satM}(F)[\ell]$ is the row in $\mathrm{satM}(F)$ corresponding to ℓ, for $\ell \in F$.

Redundancy tests (2/2)

Theorem (Recall from previous slide)

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

1. *The inequality ℓ is strongly redundant in F iff $\mathcal{S}^{\mathcal{VR}}(\ell)$ 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 iff the set $\mathcal{S}^{\mathcal{H}}(\mathcal{S}^{\mathcal{VR}}(\ell)) \smallsetminus \{\ell\}$ is not empty.*

- Denote by $\mathrm{satM}(F)$ the saturation matrix of F.
- $\mathrm{satM}(F)[\ell]$ is the row in $\mathrm{satM}(F)$ corresponding to ℓ, for $\ell \in F$.

Corollary

The following properties hold:

Redundancy tests (2/2)

Theorem (Recall from previous slide)

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

1. *The inequality ℓ is strongly redundant in F iff $\mathcal{S}^{\mathcal{VR}}(\ell)$ 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 iff the set $\mathcal{S}^{\mathcal{H}}(\mathcal{S}^{\mathcal{VR}}(\ell)) \setminus \{\ell\}$ is not empty.*

- Denote by $\mathrm{satM}(F)$ the saturation matrix of F.
- $\mathrm{satM}(F)[\ell]$ is the row in $\mathrm{satM}(F)$ corresponding to ℓ, for $\ell \in F$.

Corollary

The following properties hold:

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

Redundancy tests (2/2)

Theorem (Recall from previous slide)

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

1. *The inequality ℓ is strongly redundant in F iff $\mathcal{S}^{\mathcal{VR}}(\ell)$ 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 iff the set $\mathcal{S}^{\mathcal{H}}(\mathcal{S}^{\mathcal{VR}}(\ell)) \setminus \{\ell\}$ is not empty.*

- Denote by $\mathrm{satM}(F)$ the saturation matrix of F.
- $\mathrm{satM}(F)[\ell]$ is the row in $\mathrm{satM}(F)$ corresponding to ℓ, for $\ell \in F$.

Corollary

The following properties hold:

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

Redundancy tests (2/2)

Theorem (Recall from previous slide)

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

1. *The inequality ℓ is strongly redundant in F iff $\mathcal{S}^{\mathcal{VR}}(\ell)$ 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 iff the set $\mathcal{S}^{\mathcal{H}}(\mathcal{S}^{\mathcal{VR}}(\ell)) \setminus \{\ell\}$ is not empty.*

- Denote by $\mathrm{satM}(F)$ the saturation matrix of F.
- $\mathrm{satM}(F)[\ell]$ is the row in $\mathrm{satM}(F)$ corresponding to ℓ, for $\ell \in F$.

Corollary

The following properties hold:

1. If $\mathrm{satM}(F)[\ell]$ contains zeros only, then ℓ is strongly redundant.
2. If the number of nonzeros 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.

Updating satM(F) after eliminating one variable

- Consider the elimination of a variable, say x, during FME.

Updating satM(F) after eliminating one variable

- Consider the elimination of a variable, say x, during FME.
- Let $\ell_{pos} : a_1 x + \mathbf{c}_1^t \mathbf{y} \leq b_1$ and $\ell_{neg} : a_2 x + \mathbf{c}_2^t \mathbf{y} \leq b_2$, be two inequalities in x, where:

Updating satM(F) after eliminating one variable

- Consider the elimination of a variable, say x, during FME.
- Let $\ell_{pos} : a_1 x + \mathbf{c}_1^t \mathbf{y} \leq b_1$ and $\ell_{neg} : a_2 x + \mathbf{c}_2^t \mathbf{y} \leq b_2$, be two inequalities in x, where:
  1. we have $a_1 > 0$ and $a_2 < 0$,

skip slide

Updating satM(F) after eliminating one variable

- Consider the elimination of a variable, say x, during FME.
- Let $\ell_{pos} : a_1 x + \mathbf{c}_1^t \mathbf{y} \le b_1$ and $\ell_{neg} : a_2 x + \mathbf{c}_2^t \mathbf{y} \le b_2$, be two inequalities in x, where:
  1. we have $a_1 > 0$ and $a_2 < 0$,
  2. $\mathbf{y}$ is the vector of the remaining $(n-1)$ variables, and

skip slide

Updating satM(F) after eliminating one variable

- Consider the elimination of a variable, say x, during FME.
- Let $\ell_{pos} : a_1 x + \mathbf{c}_1^t \mathbf{y} \leq b_1$ and $\ell_{neg} : a_2 x + \mathbf{c}_2^t \mathbf{y} \leq b_2$, be two inequalities in x, where:
  1. we have $a_1 > 0$ and $a_2 < 0$,
  2. $\mathbf{y}$ is the vector of the remaining $(n-1)$ variables, and
  3. $\mathbf{c}_1, \mathbf{c}_2$ are the corresponding coefficient vectors.

Updating satM(F) after eliminating one variable

- Consider the elimination of a variable, say x, during FME.
- Let $\ell_{pos} : a_1 x + \mathbf{c}_1^t \mathbf{y} \leq b_1$ and $\ell_{neg} : a_2 x + \mathbf{c}_2^t \mathbf{y} \leq b_2$, be two inequalities in x, where:
  1. we have $a_1 > 0$ and $a_2 < 0$,
  2. $\mathbf{y}$ is the vector of the remaining $(n-1)$ variables, and
  3. $\mathbf{c}_1, \mathbf{c}_2$ are the corresponding coefficient vectors.
- Then, we have

$$\mathrm{proj}(\{\ell_{pos}, \ell_{neg}\}, \{x\}) = \{-a_2 \mathbf{c}_1^t \mathbf{y} + a_1 \mathbf{c}_2^t \mathbf{y} \leq -a_2 b_1 + a_1 b_2\}.$$

Updating satM(F) after eliminating one variable

- Consider the elimination of a variable, say x, during FME.
- Let $\ell_{pos} : a_1 x + \mathbf{c}_1^t \mathbf{y} \leq b_1$ and $\ell_{neg} : a_2 x + \mathbf{c}_2^t \mathbf{y} \leq b_2$, be two inequalities in x, where:
  1. we have $a_1 > 0$ and $a_2 < 0$,
  2. $\mathbf{y}$ is the vector of the remaining $(n-1)$ variables, and
  3. $\mathbf{c}_1, \mathbf{c}_2$ are the corresponding coefficient vectors.
- Then, we have

$$\mathrm{proj}(\{\ell_{pos}, \ell_{neg}\}, \{x\}) = \{-a_2 \mathbf{c}_1^t \mathbf{y} + a_1 \mathbf{c}_2^t \mathbf{y} \leq -a_2 b_1 + a_1 b_2\}.$$

After computing all $\mathrm{proj}(\{\ell_{pos}, \ell_{neg}\}, \{x\})$'s and eliminating the redundant such inequalities, how to update the saturation matrix and prepare for the next variable elimination?

Updating satM(F) after eliminating one variable

- Consider the elimination of a variable, say x, during FME.
- Let $\ell_{pos} : a_1 x + \mathbf{c}_1^t \mathbf{y} \leq b_1$ and $\ell_{neg} : a_2 x + \mathbf{c}_2^t \mathbf{y} \leq b_2$, be two inequalities in x, where:
  1. we have $a_1 > 0$ and $a_2 < 0$,
  2. $\mathbf{y}$ is the vector of the remaining $(n-1)$ variables, and
  3. $\mathbf{c}_1, \mathbf{c}_2$ are the corresponding coefficient vectors.
- Then, we have

$$\mathrm{proj}(\{\ell_{pos}, \ell_{neg}\}, \{x\}) = \{-a_2 \mathbf{c}_1^t \mathbf{y} + a_1 \mathbf{c}_2^t \mathbf{y} \leq -a_2 b_1 + a_1 b_2\}.$$

After computing all $\mathrm{proj}(\{\ell_{pos}, \ell_{neg}\}, \{x\})$'s and eliminating the redundant such inequalities, how to update the saturation matrix and prepare for the next variable elimination?

Theorem

We have:

$$\mathcal{S}^{\mathcal{VR}}(\mathrm{proj}(\{\ell_{pos}, \ell_{neg}\}, \{x\})) = \mathrm{proj}(\mathcal{S}^{\mathcal{VR}}(\ell_{pos}) \cap \mathcal{S}^{\mathcal{VR}}(\ell_{neg}), \{x\}).$$

Plan

Algorithm 1: CheckRedundancy

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

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

```
Irredundant := {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
        Irredundant := Irredundant ∖ {i}.
        next.
    for j in Irredundant ∖ {i} do
        if satM[i] = satM[i]&satM[j] then
            Irredundant := Irredundant ∖ {i}.
            break.
F_irred := [seq(F[i], i in Irredundant)] and
 satM_irred := [seq(satM[i], i in Irredundant)].
return F_irred and satM_irred.
```

Algorithm 2: Minimal projected representation

Input: 1. an inequality system F;

2. a variable order $x_1 > x_2 > \ldots > x_n$.

Output: 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 lexico 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 $\text{satM}_{new} := \text{satM}[F^0]$.
    foreach $f_p \in F^p$ *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}.
    $F, \text{satM} := \mathrm{CheckRedundancy}(F_{new}, \text{satM}_{new})$.
    $V := V_{new}$, $res[x_{i+1}] := F^{x_{i+1}}$.
return *res*.

Plan

Implementation techniques

1. Clearly, $\mathrm{satM}(F)$ should be encoded with bit vectors (aka bit-arrays).

Implementation techniques

1. Clearly, satM(F) should be encoded with bit vectors (aka bit-arrays).
2. We use bitarray, the bitarray library by Michael Dipperstein.

Implementation techniques

1. Clearly, satM(F) should be encoded with bit vectors (aka bit-arrays).
2. We use bitarray, the bitarray library by Michael Dipperstein.
3. satM(F) is traversed both

Implementation techniques

1. Clearly, satM(F) should be encoded with bit vectors (aka bit-arrays).
2. We use bitarray, the bitarray library by Michael Dipperstein.
3. satM(F) is traversed both
   - row-wise (to compute bit-wise AND) Line 7 in Algorithm 1, and

Implementation techniques

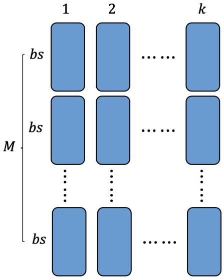
1. Clearly, satM(F) should be encoded with bit vectors (aka bit-arrays).
2. We use bitarray, the bitarray library by Michael Dipperstein.
3. satM(F) is traversed both
   - row-wise (to compute bit-wise AND) Line 7 in Algorithm 1, and
   - column-wise (to compute bit-wise OR) Line 10 in Algorithm 2.

Implementation techniques

1. Clearly, satM(F) should be encoded with bit vectors (aka bit-arrays).
2. We use bitarray, the bitarray library by Michael Dipperstein.
3. satM(F) is traversed both
    - row-wise (to compute bit-wise AND) Line 7 in Algorithm 1, and
    - column-wise (to compute bit-wise OR) Line 10 in Algorithm 2.
4. For cache complexity reasons, we maintain both satM(F) and satM$(F)^t$.

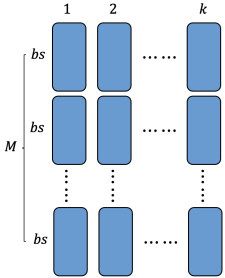
Implementation techniques

1. Clearly, satM(F) should be encoded with bit vectors (aka bit-arrays).
2. We use bitarray, the bitarray library by Michael Dipperstein.
3. satM(F) is traversed both
   - row-wise (to compute bit-wise AND) Line 7 in Algorithm 1, and
   - column-wise (to compute bit-wise OR) Line 10 in Algorithm 2.
4. For cache complexity reasons, we maintain both satM(F) and satM$(F)^t$.
5. Moreover, these matrices should be represented by blocks.



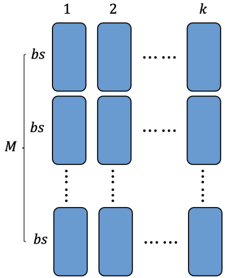
Implementation techniques

1. Clearly, satM(F) should be encoded with bit vectors (aka bit-arrays).
2. We use bitarray, the bitarray library by Michael Dipperstein.
3. satM(F) is traversed both
   - row-wise (to compute bit-wise AND) Line 7 in Algorithm 1, and
   - column-wise (to compute bit-wise OR) Line 10 in Algorithm 2.
4. For cache complexity reasons, we maintain both satM(F) and satM$(F)^t$.
5. Moreover, these matrices should be represented by blocks.
6. Other key tasks Algorithm 2 are



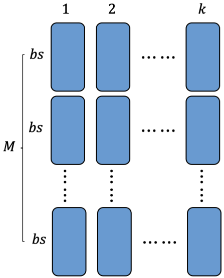
Implementation techniques

1. Clearly, satM(F) should be encoded with bit vectors (aka bit-arrays).
2. We use bitarray, the bitarray library by Michael Dipperstein.
3. satM(F) is traversed both
    - row-wise (to compute bit-wise AND) Line 7 in Algorithm 1, and
    - column-wise (to compute bit-wise OR) Line 10 in Algorithm 2.
4. For cache complexity reasons, we maintain both satM(F) and satM$(F)^t$.
5. Moreover, these matrices should be represented by blocks.
6. Other key tasks Algorithm 2 are
    - computing the V-representation of each successive projection



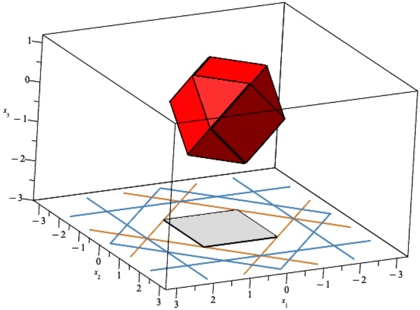
Implementation techniques

1. Clearly, satM(F) should be encoded with bit vectors (aka bit-arrays).
2. We use bitarray, the bitarray library by Michael Dipperstein.
3. satM(F) is traversed both
   - row-wise (to compute bit-wise AND) Line 7 in Algorithm 1, and
   - column-wise (to compute bit-wise OR) Line 10 in Algorithm 2.
4. For cache complexity reasons, we maintain both satM(F) and satM$(F)^t$.
5. Moreover, these matrices should be represented by blocks.
6. Other key tasks Algorithm 2 are
   - computing the V-representation of each successive projection
   - updating the saturation matrix.



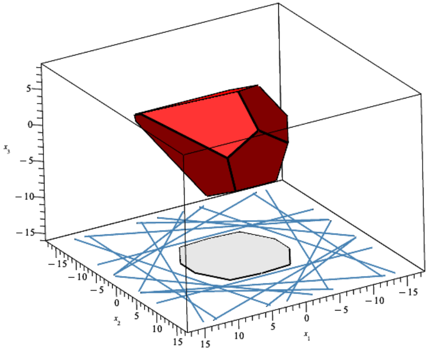
Plan

Cuboctahedron



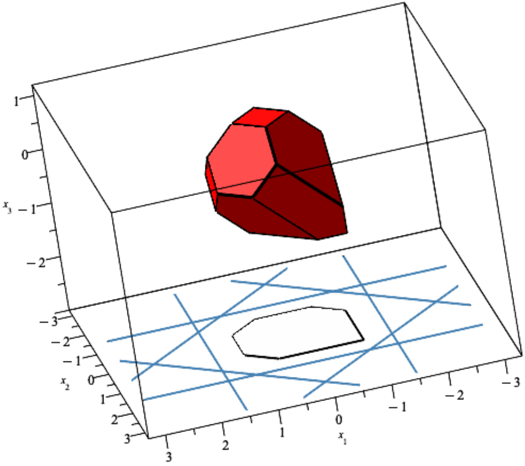
1. **strongly redundannt inequalities**
2. **weakly redundant inequalities eliminated by cardinality**
3. **weakly redundancies inequalities eliminated by containment**

Snub disphenoid (triangular dodecahedron)

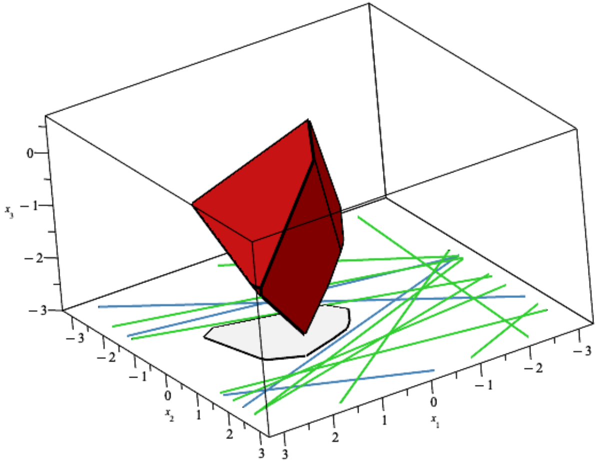


1. **strongly redundannt inequalities**
2. **weakly redundant inequalities eliminated by cardinality**
3. **weakly redundancies inequalities eliminated by containment**

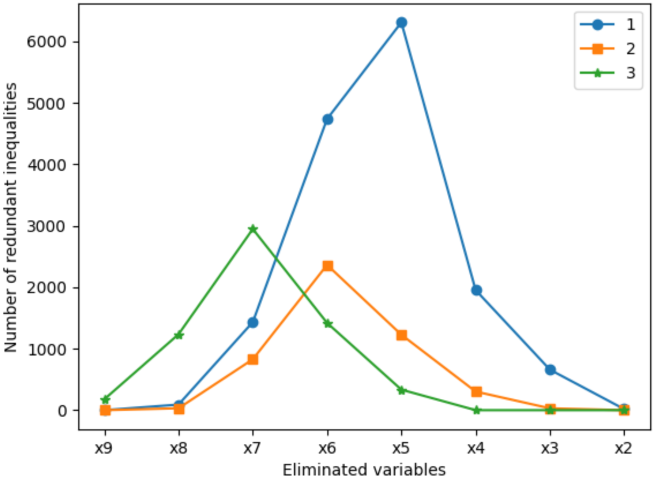
Truncated octahedron



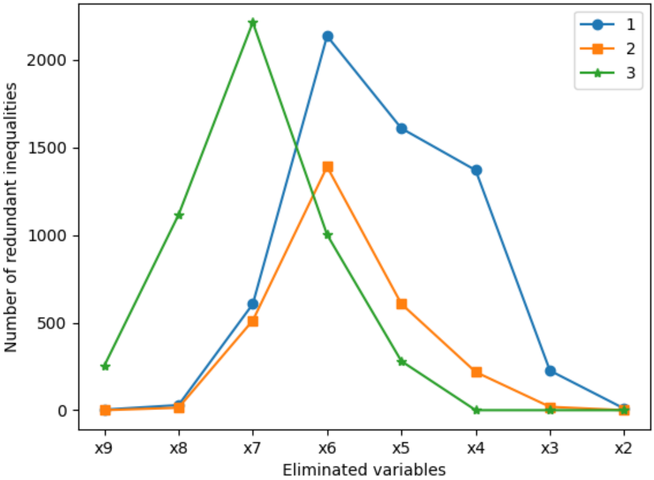
Random 3D polyhedron



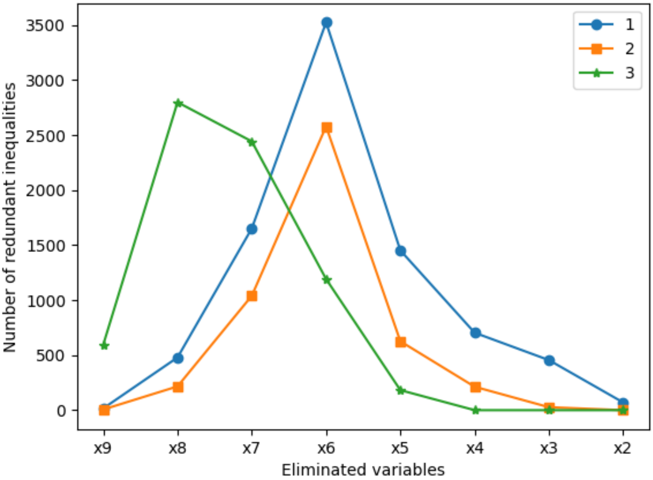
Random 10D polyhedron



Random 10D polyhedron



Random 10D polyhedron



Comparative experimentation (1/3)

Four ways of eliminating all variables:

- MPR (this paper): one variable after another, uses both the H-representation and V-representations, redundancy test via saturation matrices

Comparative experimentation (1/3)

Four ways of eliminating all variables:

- MPR (this paper): one variable after another, uses both the H-representation and V-representations, redundancy test via saturation matrices
- BPAS ([3] by Authors 1 and 2, with Delaram Talaashrafi): one variable after another, uses both the H-representation and V-representations, redundancy test via redundancy test cones, thus linear algebra over $\mathbb{Q}$.

Comparative experimentation (1/3)

Four ways of eliminating all variables:

- MPR (this paper): one variable after another, uses both the H-representation and V-representations, redundancy test via saturation matrices
- BPAS ([3] by Authors 1 and 2, with Delaram Talaashrafi): one variable after another, uses both the H-representation and V-representations, redundancy test via redundancy test cones, thus linear algebra over $\mathbb{Q}$.
- cddlib [1] by Komei Fukuda: can eliminate several variables in one step, can work with the H-representation only, redundancy test via Linear Programming (LP).

Comparative experimentation (1/3)

Four ways of eliminating all variables:

- MPR (this paper): one variable after another, uses both the H-representation and V-representations, redundancy test via saturation matrices
- BPAS ([3] by Authors 1 and 2, with Delaram Talaashrafi): one variable after another, uses both the H-representation and V-representations, redundancy test via redundancy test cones, thus linear algebra over $\mathbb{Q}$.
- cddlib [1] by Komei Fukuda: can eliminate several variables in one step, can work with the H-representation only, redundancy test via Linear Programming (LP).
- polylib [5] by Vincent Loechner and Doran K. Wilde: can eliminate several variables in one step, can work with the V-representation only, convert between H-rep and V-rep as needed.

Comparative experimentation (1/3)

Four ways of eliminating all variables:

- MPR (this paper): one variable after another, uses both the H-representation and V-representations, redundancy test via saturation matrices
- BPAS ([3] by Authors 1 and 2, with Delaram Talaashrafi): one variable after another, uses both the H-representation and V-representations, redundancy test via redundancy test cones, thus linear algebra over $\mathbb{Q}$.
- cddlib [1] by Komei Fukuda: can eliminate several variables in one step, can work with the H-representation only, redundancy test via Linear Programming (LP).
- polylib [5] by Vincent Loechner and Doran K. Wilde: can eliminate several variables in one step, can work with the V-representation only, convert between H-rep and V-rep as needed.

We used the following sources for our test cases:

1. random non-empty polyhedra with n variables and m inequalities. The coefficients rang in the interval $[-10, 10]$.
2. polyhedra coming from libraries polylib and BPAS.

Comparative experimentation (1/3)

Four ways of eliminating all variables:

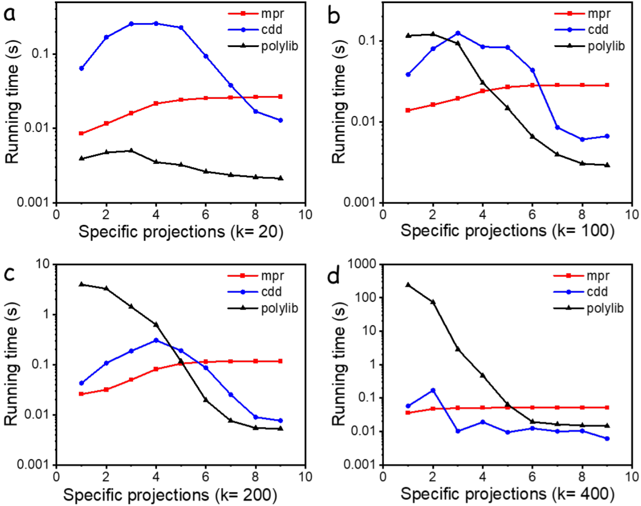
- MPR (this paper): one variable after another, uses both the H-representation and V-representations, redundancy test via saturation matrices
- BPAS ([3] by Authors 1 and 2, with Delaram Talaashrafi): one variable after another, uses both the H-representation and V-representations, redundancy test via redundancy test cones, thus linear algebra over $\mathbb{Q}$.
- cddlib [1] by Komei Fukuda: can eliminate several variables in one step, can work with the H-representation only, redundancy test via Linear Programming (LP).
- polylib [5] by Vincent Loechner and Doran K. Wilde: can eliminate several variables in one step, can work with the V-representation only, convert between H-rep and V-rep as needed.

We used the following sources for our test cases:

1. random non-empty polyhedra with n variables and m inequalities. The coefficients rang in the interval $[-10, 10]$.
2. polyhedra coming from libraries polylib and BPAS.

All the experimental results were collected on a PC (Intel(R) Xeon(R) Gold 6258R CPU 2.70GHz, 503G RAM, Ubuntu 20.04.3).

Comparative experimentation (2/3)



1. Four different random polyhedra with $m = 15$ and $n = 10$.
2. For $1 \le i \le 9$, in the hor. axiss, the first i variables are eliminated.
3. The vert. axis in each figure shows the running time (in seconds).

test case	(n, m, k)	mpr	BPAS	cdd	polylib
32hedron	(6, 32, 11)	6.54	16.80	4183.08	**1.92**
64hedron	(7,64,13)	13.05	52.42	>5min	**1.67**
francois	(13,27,2304)	499.92	**253.66**	388.36	> 5min
francois2	(13,31,384)	**41.80**	140.34	55.17	80.63
herve.in	(14,25,262)	34.42	140.34	294.01	**30.08**
c6.in	(11,17,31)	**9.85**	12.72	84.11	5.56
c9.in	(16,18,140)	**25.08**	65.54	151.17	131.53
c10.in	(18,20,142)	22.10	98.68	249.02	**16.06**
S24	(24, 25,25)	23.50	58.80	748.67	**17.47**
S35	(35, 36,36)	46.55	182.14	3575.00	**46.007**
cube	(10, 20,1024)	**81.33**	201.92	125.900	161.06
C56	(5, 6,6)	3.67	4.09	11.81	**0.79**
C1011	(10, 11,11)	24.99	115.68	1716.25	**9.99**
C510	(5, 42,10)	12.00	40.01	>5min	**4.42**
T1	(5, 10,38)	**5.61**	16.44	27.42	8.81
T3	(10,12,29)	21.29	141.64	288.07	**12.07**
T5	(5, 10,36)	8.12	15.62	22.92	**4.76**
T6	(10,20,390)	**1142.9**	23800.11	14937.61	>5min
T7	(5, 8,26)	5.81	10.79	13.96	**4.00**
T9	(10,12,36)	**36.56**	414.53	479.18	100.34
T10	(6, 8,24)	**4.58**	13.65	18.39	5.27
T12	(5, 11,42)	**8.52**	19.03	38.65	8.60
R_15_20	(15, 20,1328)	**28430.40**	336035.00	38037.21	>5min

Plan

Complexity estimates (1/2)

Recall the notations

1. m is the number of inequalities and n is the dimension of the ambient space. If the input H-representation is irredundant, the m is also the number of facets of P.
2. Let $h := \text{height}([A, \mathbf{b}])$, let θ be the coefficient of linear algebra and ω the bit-size of a machine word.

Well-known bounds

1. The size k of the V-representation (V, R) is at most $\binom{m}{n} + \binom{m}{n-1} \leq \frac{m^n}{n!}$.
2. From [2], for $1 \leq i < n$, after eliminating i variables during the process of FME, the number of irredundant inequalities defining the projection is at most $\binom{m}{n-i-1} \leq m^n$.

Theorem

The costs for computing all the inequalities (redundant and irredundant) and generating the initial saturation matrix are within $O(m^{2n} n^{\theta+\varepsilon} h^{1+\varepsilon})$ *bit operations, while the costs for updating and operating on the saturation matrices are bounded over by* $\frac{3m^{3n-4}}{\omega}$ *word operations.*

Complexity estimates (1/2)

Recall the notations

1. m is the number of inequalities and n is the dimension of the ambient space. If the input H-representation is irredundant, the m is also the number of facets of P.
2. Let $h := \mathrm{height}([A, \mathbf{b}])$, let θ be the coefficient of linear algebra and ω the bit-size of a machine word.

Bounds for FME

1. FME based on LP: $O(n^2\, m^{2n}\, \mathrm{LP}(n, 2^n h n^2 m^n))$ bit operations, where $\mathrm{LP}(d, H)$ is an upper bound for the number of bit operations required for solving a linear program in d variables and with total bit size H. For instance, in the case of Karmarkar's algorithm [4], we have $\mathrm{LP}(d, H) \in O(d^{3.5} H^2 \cdot \log H \cdot \log\log H)$.
2. FME based on redundancy test cone: $O(m^{\frac{5n}{2}}\, n^{\theta+1+\epsilon}\, h^{1+\epsilon})$ bit operations, for any $\epsilon > 0$.
3. This paper: $O(m^{2n} n^{\theta+\varepsilon} h^{1+\varepsilon})$ bit operations and $\frac{3m^{3n-4}}{\omega}$ word operations.

Plan

Concluding remarks

Summary and notes

1. We proposed a technique for removing redundant inequalities in linear systems.
2. It relies on the analysis of 3 different types of redundancies
3. Our redundancy tests allow for efficient implementation based on bit-vector arithmetic.
4. From the experimental results, our method works best on hard problems.
5. This is promising to solve large scale problems in areas like information theory, SMT and optimizing compilers.

Work in progress

1. Our implementation has room for improvements.
2. Indeed, our algorithms have opportunities for both multithreaded parallelism and instruction-level parallelism.
3. The third criterion (redundancy test based on containment) needs further study to discover the container.

References

[1] K. Fukuda. *The CDD and CDDplus Homepage*. https://www.inf.ethz.ch/personal/fukudak/cdd_home/.

[2] R. J. Jing and M. Moreno Maza. "Computing the integer points of a polyhedron, II: complexity estimates". In: *Proceedings of CASC*. Springer. 2017, pp. 242–256.

[3] R. J. Jing, M. Moreno-Maza, and D. Talaashrafi. "Complexity estimates for Fourier-Motzkin elimination". In: *Proceedings of CASC*. Springer. 2020, pp. 282–306.

[4] N. Karmarkar. "A new polynomial-time algorithm for linear programming". In: *Proceedings of the sixteenth annual ACM symposium on Theory of computing*. STOC '84. New York, NY, USA: ACM, 1984, pp. 302–311. ISBN: 0-89791-133-4. DOI: 10.1145/800057.808695. URL: http://doi.acm.org/10.1145/800057.808695.

[5] V. Loechner. *PolyLib: A library for manipulating parameterized polyhedra*. 1999.