Exploiting Sparsity in Polyhedral Analysis

Axel Simon and Andy King

Computing Laboratory, University of Kent,
Canterbury, CT2 7NF, UK
{A.Simon,A.King}@kent.ac.uk

September 7, 2005

Closed, Convex Polyhedra

Let P be a finite set of (non-strict) inequalities over n variables.
Let $soln(P) \subseteq \mathbb{R}^n$ denote the solution set of P.
Operations on polyhedra:

Closed, Convex Polyhedra

Let P be a finite set of (non-strict) inequalities over n variables.
Let $soln(P) \subseteq \mathbb{R}^n$ denote the solution set of P.
Operations on polyhedra:

$P_1 \models P_2$ entailment check, i.e. $soln(P_1) \subseteq soln(P_2)$.

Closed, Convex Polyhedra

Let P be a finite set of (non-strict) inequalities over n variables.
Let $soln(P) \subseteq \mathbb{R}^n$ denote the solution set of P.
Operations on polyhedra:

$P_1 \models P_2$ entailment check, i.e. $soln(P_1) \subseteq soln(P_2)$.

$compress(P)$ redundancy removal, i.e. smallest $P' \subseteq P$ with $soln(P') = soln(P)$.

Closed, Convex Polyhedra

Let P be a finite set of (non-strict) inequalities over n variables.
Let $soln(P) \subseteq \mathbb{R}^n$ denote the solution set of P.
Operations on polyhedra:

$P_1 \models P_2$	entailment check, i.e. $soln(P_1) \subseteq soln(P_2)$.
$compress(P)$	redundancy removal, i.e. smallest $P' \subseteq P$ with $soln(P') = soln(P)$.
$P_1 \sqcap P_2$	intersection, i.e. $soln(P_1) \cap soln(P_2)$.

Closed, Convex Polyhedra

Let P be a finite set of (non-strict) inequalities over n variables.
Let $soln(P) \subseteq \mathbb{R}^n$ denote the solution set of P.
Operations on polyhedra:

$P_1 \models P_2$	entailment check, i.e. $soln(P_1) \subseteq soln(P_2)$.
$compress(P)$	redundancy removal, i.e. smallest $P' \subseteq P$ with $soln(P') = soln(P)$.
$P_1 \sqcap P_2$	intersection, i.e. $soln(P_1) \cap soln(P_2)$.
$P_1 \sqcup P_2$	convex hull
$\exists_Y P$	projection

Frame Representation

Classically: $P_1 \sqcup P_2$ and $\exists_Y$ implemented on frame representation.

Frame representation:
Calculate vertices V_i, rays R_i and lines L_i of $soln(P_i)$.

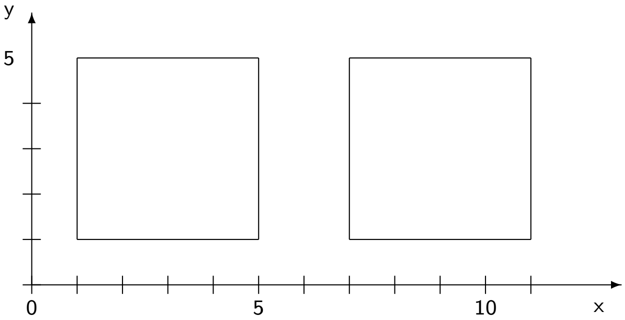
- $P_1 \sqcup P_2$ Convex hull defined by $V_1 \cup V_2$, $R_1 \cup R_2$ and $L_1 \cup L_2$.
- $\exists_Y$ Remove the components corresponding to Y from V_i, R_i and L_i

Fundamental problem: Conversion to and from frame representation can incur exponential growth.

Convex Hull

$P = P_1 \sqcup P_2$ is the a set of inequalities P such that $soln(P)$ is the smallest set with $soln(P_1) \cup soln(P_2) \subseteq soln(P)$.

Example: $P_1 = \{x \geq 1, x \leq 5, y \geq 1, y \leq 5\}$
$P_2 = \{x \geq 7, x \leq 11, y \geq 1, y \leq 5\}$

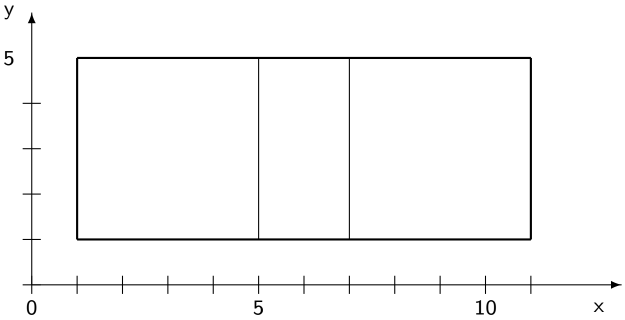


Convex Hull

$P = P_1 \sqcup P_2$ is the a set of inequalities P such that $soln(P)$ is the smallest set with $soln(P_1) \cup soln(P_2) \subseteq soln(P)$.

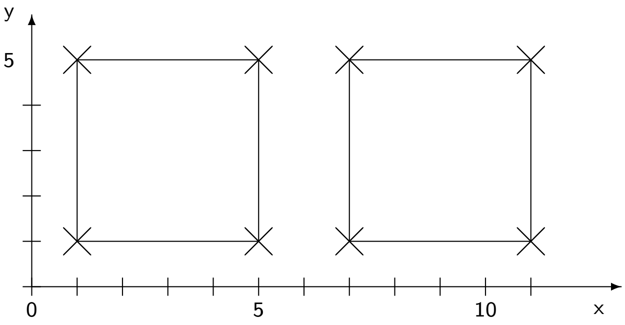
Example: $P_1 = \{x \geq 1, x \leq 5, y \geq 1, y \leq 5\}$
$P_2 = \{x \geq 7, x \leq 11, y \geq 1, y \leq 5\}$

Solution: $P = \{x \geq 1, x \leq 11, y \geq 1, y \leq 5\}$.



Convex Hull

Frame representation: P_1, P_2 can be represented with 4 vertices. In general, let P_1, P_2 be two n-dimensional hypercubes. Then $|P_1| = |P_2| = |P| = 2n$, but each hypercube contains 2^n vertices.



Alternatives to General Polyhedra

Domains proposed to circumvent exponential cost:

Alternatives to General Polyhedra

Domains proposed to circumvent exponential cost:

Difference-Bound Matrices $x - y \leq c$, $c \in \mathbb{R}$

Alternatives to General Polyhedra

Domains proposed to circumvent exponential cost:

Difference-Bound Matrices	$x - y \leq c$, $c \in \mathbb{R}$
Octagon	$\pm x \pm y \leq c$, $c \in \mathbb{R}$

Alternatives to General Polyhedra

Domains proposed to circumvent exponential cost:

Difference-Bound Matrices	$x - y \leq c,\ c \in \mathbb{R}$
Octagon	$\pm x \pm y \leq c,\ c \in \mathbb{R}$
Octahedron	$\pm x_1 \cdots \pm x_n \leq c,\ c \in \mathbb{R}$

Alternatives to General Polyhedra

Domains proposed to circumvent exponential cost:

Difference-Bound Matrices	$x - y \leq c,\ c \in \mathbb{R}$
Octagon	$\pm x \pm y \leq c,\ c \in \mathbb{R}$
Octahedron	$\pm x_1 \cdots \pm x_n \leq c,\ c \in \mathbb{R}$
Two-Variables-Per-Inequality	$ax_1 + bx_2 \leq c,\ a, b, c \in \mathbb{N}$

Alternatives to General Polyhedra

Domains proposed to circumvent exponential cost:

Difference-Bound Matrices	$x - y \leq c,\ c \in \mathbb{R}$
Octagon	$\pm x \pm y \leq c,\ c \in \mathbb{R}$
Octahedron	$\pm x_1 \cdots \pm x_n \leq c,\ c \in \mathbb{R}$
Two-Variables-Per-Inequality	$ax_1 + bx_2 \leq c,\ a, b, c \in \mathbb{N}$

Question: Which one?

Choosing one domain commits to a limited degree of precision.

Aim:

Stop generating inequalities when system becomes too large.

Problem:

Frame representation is inherently all-or-nothing.

Convex Hull as Convex Combination

Given: Two input polyhedra $A_1\vec{x} \leq \vec{c_1}$ and $A_2\vec{x} \leq \vec{c_2}$.

- Smallest convex combination of entailed points:

$$P = \left\{ \vec{x} \;\middle|\; \begin{array}{l} \vec{x} = \lambda_1\vec{x_1} + \lambda_2\vec{x_2}\wedge \\ A_1\vec{x_1} \leq \vec{c_1} \wedge A_2\vec{x_2} \leq \vec{c_2} \wedge \\ \lambda_1 + \lambda_2 = 1 \wedge \lambda_1 \geq 0 \wedge \lambda_2 \geq 0 \end{array} \right\}$$

Convex Hull as Convex Combination

Given: Two input polyhedra $A_1\vec{x} \leq \vec{c_1}$ and $A_2\vec{x} \leq \vec{c_2}$.

- Smallest convex combination of entailed points:

$$P = \left\{ \vec{x} \;\middle|\; \begin{array}{l} \vec{x} = \lambda_1\vec{x_1} + \lambda_2\vec{x_2} \wedge \\ A_1\vec{x_1} \leq \vec{c_1} \wedge A_2\vec{x_2} \leq \vec{c_2} \wedge \\ \lambda_1 + \lambda_2 = 1 \wedge \lambda_1 \geq 0 \wedge \lambda_2 \geq 0 \end{array} \right\}$$

- $\lambda_1\vec{x_1}$ is not linear

Convex Hull as Convex Combination

Given: Two input polyhedra $A_1\vec{x} \leq \vec{c_1}$ and $A_2\vec{x} \leq \vec{c_2}$.

- Smallest convex combination of entailed points:

$$P = \left\{ \vec{x} \;\middle|\; \begin{array}{l} \vec{x} = \lambda_1\vec{x_1} + \lambda_2\vec{x_2} \wedge \\ A_1\vec{x_1} \leq \vec{c_1} \wedge A_2\vec{x_2} \leq \vec{c_2} \wedge \\ \lambda_1 + \lambda_2 = 1 \wedge \lambda_1 \geq 0 \wedge \lambda_2 \geq 0 \end{array} \right\}$$

- Substitute $\vec{y_1} = \lambda_1\vec{x_1}$ and $\vec{y_2} = \lambda_2\vec{x_2}$:

$$P' = \left\{ \vec{x} \;\middle|\; \begin{array}{l} \vec{x} = \vec{y_1} + \vec{y_2} \wedge \\ A_1\vec{y_1} \leq \lambda_1\vec{c_1} \wedge A_2\vec{y_2} \leq \lambda_2\vec{c_2} \wedge \\ \lambda_1 + \lambda_2 = 1 \wedge \lambda_1 \geq 0 \wedge \lambda_2 \geq 0 \end{array} \right\}$$

Convex Hull as Convex Combination

Given: Two input polyhedra $A_1\vec{x} \leq \vec{c_1}$ and $A_2\vec{x} \leq \vec{c_2}$.

- Smallest convex combination of entailed points:

$$P = \left\{ \vec{x} \;\middle|\; \begin{array}{l} \vec{x} = \lambda_1\vec{x_1} + \lambda_2\vec{x_2} \wedge \\ A_1\vec{x_1} \leq \vec{c_1} \wedge A_2\vec{x_2} \leq \vec{c_2} \wedge \\ \lambda_1 + \lambda_2 = 1 \wedge \lambda_1 \geq 0 \wedge \lambda_2 \geq 0 \end{array} \right\}$$

- Substitute $\vec{y_1} = \lambda_1\vec{x_1}$ and $\vec{y_2} = \lambda_2\vec{x_2}$:

$$P' = \left\{ \vec{x} \;\middle|\; \begin{array}{l} \vec{x} = \vec{y_1} + \vec{y_2} \wedge \\ A_1\vec{y_1} \leq \lambda_1\vec{c_1} \wedge A_2\vec{y_2} \leq \lambda_2\vec{c_2} \wedge \\ \lambda_1 + \lambda_2 = 1 \wedge \lambda_1 \geq 0 \wedge \lambda_2 \geq 0 \end{array} \right\}$$

- Project out $\vec{y_1}$, $\vec{y_2}$, λ_1 and λ_2.
  $\rightsquigarrow$ Need an efficient projection algorithm.

Fourier-Motzkin Algorithm

Consider the following system E:

$$\begin{array}{rcrcrcr}
2x_1 & + & x_2 & + & 2x_3 & \leq & 7 \\
-1x_1 & + & 2x_2 & + & x_3 & \leq & -5 \\
1x_1 & - & x_2 & + & 1x_3 & \leq & -3 \\
3x_1 & - & 2x_2 & - & 1x_3 & \leq & 6 \\
2x_1 & & & + & 2x_3 & \leq & 7 \\
-1x_1 & & & + & x_3 & \leq & -5
\end{array}$$

Fourier-Motzkin Algorithm

Task: Eliminate x_2.

$$
\begin{array}{rcrcrcr}
2x_1 & + & x_2 & + & 2x_3 & \leq & 7 \\
-1x_1 & + & 2x_2 & + & x_3 & \leq & -5 \\
1x_1 & - & x_2 & + & 1x_3 & \leq & -3 \\
3x_1 & - & 2x_2 & - & 1x_3 & \leq & 6 \\
2x_1 & & & + & 2x_3 & \leq & 7 \\
-1x_1 & & & + & x_3 & \leq & -5
\end{array}
$$

Fourier-Motzkin Algorithm

Partition system:

$$
\begin{array}{lrcrcrcr}
E^+ = \{ & 2x_1 & + & \color{red}{x_2} & + & 2x_3 & \leq & 7, \\
 & -1x_1 & + & \color{red}{2x_2} & + & x_3 & \leq & -5\} \\
E^- = \{ & 1x_1 & - & \color{red}{x_2} & + & 1x_3 & \leq & -3, \\
 & 3x_1 & - & \color{red}{2x_2} & - & 1x_3 & \leq & 6\} \\
E^{res} = \{ & 2x_1 & & & + & 2x_3 & \leq & 7, \\
 & -1x_1 & & & + & x_3 & \leq & -5\}
\end{array}
$$

Fourier-Motzkin Algorithm

$$
\begin{array}{lrcrcrcr}
E^{+} = \{ & 2x_1 & + & x_2 & + & 2x_3 & \leq & 7, \\
 & -1x_1 & + & 2x_2 & + & x_3 & \leq & -5\} \\
E^{-} = \{ & 1x_1 & - & x_2 & + & 1x_3 & \leq & -3, \\
 & 3x_1 & - & 2x_2 & - & 1x_3 & \leq & 6\} \\
E^{res} = \{ & 2x_1 & & & + & 2x_3 & \leq & 7, \\
 & -1x_1 & & & + & x_3 & \leq & -5, \\
 & 3x_1 & & & + & 3x_3 & \leq & 4\}
\end{array}
$$

Fourier-Motzkin Algorithm

$$
\begin{array}{lrcrcrcr}
E^{+} = \{ & 2x_1 & + & \color{red}{x_2} & + & 2x_3 & \leq & 7, \\
 & -1x_1 & + & 2x_2 & + & x_3 & \leq & -5\} \\
E^{-} = \{ & 1x_1 & - & x_2 & + & 1x_3 & \leq & -3, \\
 & 3x_1 & - & \color{red}{2x_2} & - & 1x_3 & \leq & 6\} \\
E^{res} = \{ & 2x_1 & & & + & 2x_3 & \leq & 7, \\
 & -1x_1 & & & + & x_3 & \leq & -5, \\
 & 3x_1 & & & + & 3x_3 & \leq & 4, \\
 & 7x_1 & & & + & 5x_3 & \leq & 10\}
\end{array}
$$

Fourier-Motzkin Algorithm

$$
\begin{array}{lrcrcrcr}
E^+ = \{ & 2x_1 & + & x_2 & + & 2x_3 & \leq & 7, \\
 & -1x_1 & + & 2x_2 & + & x_3 & \leq & -5\} \\
E^- = \{ & 1x_1 & - & x_2 & + & 1x_3 & \leq & -3, \\
 & 3x_1 & - & 2x_2 & - & 1x_3 & \leq & 6\} \\
E^{res} = \{ & 2x_1 & & & + & 2x_3 & \leq & 7, \\
 & -1x_1 & & & + & x_3 & \leq & -5, \\
 & 3x_1 & & & + & 3x_3 & \leq & 4, \\
 & 7x_1 & & & + & 5x_3 & \leq & 10, \\
 & 1x_1 & & & + & 3x_3 & \leq & -11\}
\end{array}
$$

Fourier-Motzkin Algorithm

$$
\begin{array}{lrcrcrcr}
E^{+} = \{ & 2x_1 & + & x_2 & + & 2x_3 & \leq & 7, \\
 & -1x_1 & + & 2x_2 & + & x_3 & \leq & -5\} \\
E^{-} = \{ & 1x_1 & - & x_2 & + & 1x_3 & \leq & -3, \\
 & 3x_1 & - & 2x_2 & - & 1x_3 & \leq & 6\} \\
E^{res} = \{ & 2x_1 & & & + & 2x_3 & \leq & 7, \\
 & -1x_1 & & & + & x_3 & \leq & -5, \\
 & 3x_1 & & & + & 3x_3 & \leq & 4, \\
 & 7x_1 & & & + & 5x_3 & \leq & 10, \\
 & 1x_1 & & & + & 3x_3 & \leq & -11, \\
 & 2x_1 & & & + & & \leq & 1\}
\end{array}
$$

Fourier-Motzkin Algorithm

$$E^{res} = \left\{ \begin{array}{rcrcr} 2x_1 & + & 2x_3 & \leq & 7, \\ -1x_1 & + & x_3 & \leq & -5, \\ 3x_1 & + & 3x_3 & \leq & 4, \\ 7x_1 & + & 5x_3 & \leq & 10, \\ 1x_1 & + & 3x_3 & \leq & -11, \\ 2x_1 & + & & \leq & 1 \end{array} \right\}$$

$\leadsto$ E^{res} is projection onto x_1, x_3 - plane

Variable Selection

Input system: $|E| = |E^+| + |E^-| + |E^{res}|$
Output system: $|E^+| \times |E^-| + |E^{res}|$

Variable Selection

Input system: $|E| = |E^+| + |E^-| + |E^{res}|$
Output system: $|E^+| \times |E^-| + |E^{res}|$

- Growth is $\Delta = |E^+| \times |E^-| - (|E^+| + |E^-|)$

Variable Selection

Input system: $|E| = |E^+| + |E^-| + |E^{res}|$
Output system: $|E^+| \times |E^-| + |E^{res}|$

- Growth is $\Delta = |E^+| \times |E^-| - (|E^+| + |E^-|)$
- For sparse systems Δ is often zero or even -1

Variable Selection

Input system: $|E| = |E^+| + |E^-| + |E^{res}|$
Output system: $|E^+| \times |E^-| + |E^{res}|$

- Growth is $\Delta = |E^+| \times |E^-| - (|E^+| + |E^-|)$
- For sparse systems Δ is often zero or even -1
- To delay growth, select variables with lowest Δ first

Variable Selection

Input system: $|E| = |E^+| + |E^-| + |E^{res}|$
Output system: $|E^+| \times |E^-| + |E^{res}|$

- Growth is $\Delta = |E^+| \times |E^-| - (|E^+| + |E^-|)$
- For sparse systems Δ is often zero or even -1
- To delay growth, select variables with lowest Δ first
- Remove redundant inequalities:

  quasi-syntactic after each step

  compress if system grows beyond initial size

Variable Selection

Input system: $|E| = |E^+| + |E^-| + |E^{res}|$
Output system: $|E^+| \times |E^-| + |E^{res}|$

- Growth is $\Delta = |E^+| \times |E^-| - (|E^+| + |E^-|)$
- For sparse systems Δ is often zero or even -1
- To delay growth, select variables with lowest Δ first
- Remove redundant inequalities:

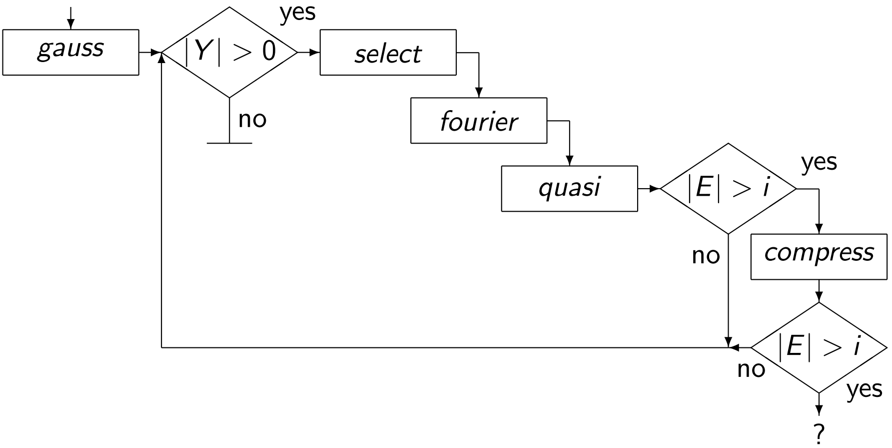
  quasi-syntactic after each step

  compress if system grows beyond initial size

Fourier-Motzkin eliminates most variable without growth.

Projection Algorithm

Eliminating Y from E producing at most i inequalities:



Eliminating Several Variables At Once

Eliminate $Y = \{x_1, \ldots x_4\}$ from the following system:

$$\begin{array}{rcrcrcrcrcrcrcl}
1x_1 & + & 2x_2 & - & 3x_3 & + & 4x_4 & - & 2x_5 & & & + & 3x_7 & \leq & -9 \\
4x_1 & + & 4x_2 & + & 2x_3 & - & x_4 & - & 3x_5 & - & 2x_6 & + & 6x_7 & \leq & 3 \\
-2x_1 & - & 2x_2 & + & 7x_3 & + & 2x_4 & + & x_5 & + & 8x_6 & + & 2x_7 & \leq & 4 \\
7x_1 & + & 5x_2 & & & - & 4x_4 & & & + & 4x_6 & + & 10x_7 & \leq & -2 \\
 & & 2x_2 & + & 3x_3 & + & 8x_4 & - & 3x_5 & - & 2x_6 & + & 3x_7 & \leq & 12 \\
8x_1 & + & 2x_2 & - & 2x_3 & & & + & 2x_5 & - & 9x_6 & + & x_7 & \leq & 0 \\
-8x_1 & & & - & x_3 & - & x_4 & - & 4x_5 & - & x_6 & + & 6x_7 & \leq & -9
\end{array}$$

Eliminating Several Variables At Once

Eliminate $Y = \{x_1, \ldots x_4\}$ from the following system:

$$\begin{array}{rcrcrcrcrcrcrcl}
1x_1 & + & 2x_2 & - & 3x_3 & + & 4x_4 & - & 2x_5 & & & + & 3x_7 & \leq & -9 \\
4x_1 & + & 4x_2 & + & 2x_3 & - & x_4 & - & 3x_5 & - & 2x_6 & + & 6x_7 & \leq & 3 \\
-2x_1 & - & 2x_2 & + & 7x_3 & + & 2x_4 & + & x_5 & + & 8x_6 & + & 2x_7 & \leq & 4 \\
7x_1 & + & 5x_2 & & & - & 4x_4 & & & + & 4x_6 & + & 10x_7 & \leq & -2 \\
 & & 2x_2 & + & 3x_3 & + & 8x_4 & - & 3x_5 & - & 2x_6 & + & 3x_7 & \leq & 12 \\
8x_1 & + & 2x_2 & - & 2x_3 & & & + & 2x_5 & - & 9x_6 & + & x_7 & \leq & 0 \\
-8x_1 & & & - & x_3 & - & x_4 & - & 4x_5 & - & x_6 & + & 6x_7 & \leq & -9
\end{array}$$

Rewrite:

$$\begin{pmatrix} 1 & 2 & -3 & 4 \\ 4 & 4 & 2 & -1 \\ -2 & -2 & 7 & 2 \\ 7 & 5 & 0 & -4 \\ 0 & 2 & 3 & 8 \\ 8 & 2 & -2 & 0 \\ -8 & 0 & -1 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} -2 & 0 & 3 \\ -3 & -2 & 6 \\ 1 & 8 & 2 \\ 0 & 4 & 10 \\ -3 & -2 & 3 \\ 2 & -9 & 1 \\ -4 & -1 & 6 \end{pmatrix} \begin{pmatrix} x_5 \\ x_6 \\ x_7 \end{pmatrix} \leq \begin{pmatrix} -9 \\ 3 \\ 4 \\ -2 \\ 12 \\ 0 \\ -9 \end{pmatrix}$$

Eliminating Several Variables At Once

Eliminate $Y = \{x_1, \dots x_4\}$ from the following system:

$$
\begin{array}{rcrcrcrcrcrcrl}
1x_1 & + & 2x_2 & - & 3x_3 & + & 4x_4 & - & 2x_5 & & & + & 3x_7 & \leq -9 \\
4x_1 & + & 4x_2 & + & 2x_3 & - & x_4 & - & 3x_5 & - & 2x_6 & + & 6x_7 & \leq 3 \\
-2x_1 & - & 2x_2 & + & 7x_3 & + & 2x_4 & + & x_5 & + & 8x_6 & + & 2x_7 & \leq 4 \\
7x_1 & + & 5x_2 & & & - & 4x_4 & & & + & 4x_6 & + & 10x_7 & \leq -2 \\
 & & 2x_2 & + & 3x_3 & + & 8x_4 & - & 3x_5 & - & 2x_6 & + & 3x_7 & \leq 12 \\
8x_1 & + & 2x_2 & - & 2x_3 & & & + & 2x_5 & - & 9x_6 & + & x_7 & \leq 0 \\
-8x_1 & & & - & x_3 & - & x_4 & - & 4x_5 & - & x_6 & + & 6x_7 & \leq -9
\end{array}
$$

$$
\overbrace{\begin{pmatrix} 1 & 2 & -3 & 4 \\ 4 & 4 & 2 & -1 \\ -2 & -2 & 7 & 2 \\ 7 & 5 & 0 & -4 \\ 0 & 2 & 3 & 8 \\ 8 & 2 & -2 & 0 \\ -8 & 0 & -1 & -1 \end{pmatrix}}^{A}
\overbrace{\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}}^{\vec{y}}
+
\overbrace{\begin{pmatrix} -2 & 0 & 3 \\ -3 & -2 & 6 \\ 1 & 8 & 2 \\ 0 & 4 & 10 \\ -3 & -2 & 3 \\ 2 & -9 & 1 \\ -4 & -1 & 6 \end{pmatrix}}^{B}
\overbrace{\begin{pmatrix} x_5 \\ x_6 \\ x_7 \end{pmatrix}}^{\vec{z}}
\leq
\overbrace{\begin{pmatrix} -9 \\ 3 \\ 4 \\ -2 \\ 12 \\ 0 \\ -9 \end{pmatrix}}^{\vec{c}}
$$

Finding Inequalities in the Projection Space

Project out $\vec{y}$ from the n inequalities rewritten as:

$$A\vec{y} + B\vec{z} \leq \vec{c}$$

Finding Inequalities in the Projection Space

Project out $\vec{y}$ from the n inequalities rewritten as:

$$A\vec{y} + B\vec{z} \leq \vec{c}$$

- Let $\vec{\lambda} = \langle \lambda_1, \ldots \lambda_n \rangle$ combine rows in A such that $\vec{\lambda}A = 0$.

Finding Inequalities in the Projection Space

Project out $\vec{y}$ from the n inequalities rewritten as:

$$A\vec{y} + B\vec{z} \leq \vec{c}$$

- Let $\vec{\lambda} = \langle \lambda_1, \ldots \lambda_n \rangle$ combine rows in A such that $\vec{\lambda} A = 0$.
- Require that $\lambda_i \geq 0$, $i = 1, \ldots n$.

Finding Inequalities in the Projection Space

Project out $\vec{y}$ from the n inequalities rewritten as:

$$A\vec{y} + B\vec{z} \leq \vec{c}$$

- Let $\vec{\lambda} = \langle \lambda_1, \ldots \lambda_n \rangle$ combine rows in A such that $\vec{\lambda}A = 0$.
- Require that $\lambda_i \geq 0$, $i = 1, \ldots n$.
- Given a $\vec{\lambda}$, it follows that $\vec{\lambda}(A\vec{y} + B\vec{z}) \leq \vec{\lambda}\vec{c}$, hence $\vec{\lambda}B\vec{z} \leq \vec{\lambda}\vec{c}$, which is a single inequality in the projection space.

Finding Inequalities in the Projection Space

Project out $\vec{y}$ from the n inequalities rewritten as:

$$A\vec{y} + B\vec{z} \leq \vec{c}$$

- Let $\vec{\lambda} = \langle \lambda_1, \dots \lambda_n \rangle$ combine rows in A such that $\vec{\lambda}A = 0$.
- Require that $\lambda_i \geq 0$, $i = 1, \dots n$.
- Given a $\vec{\lambda}$, it follows that $\vec{\lambda}(A\vec{y} + B\vec{z}) \leq \vec{\lambda}\vec{c}$, hence $\vec{\lambda}B\vec{z} \leq \vec{\lambda}\vec{c}$, which is a single inequality in the projection space.
- If $\vec{\lambda}$ is a solution, so is $s\vec{\lambda}$, $s > 0$, hence require $\lambda_1 + \dots + \lambda_n = 1$.

Finding Inequalities in the Projection Space

Project out $\vec{y}$ from the n inequalities rewritten as:

$$A\vec{y} + B\vec{z} \leq \vec{c}$$

- Let $\vec{\lambda} = \langle \lambda_1, \ldots \lambda_n \rangle$ combine rows in A such that $\vec{\lambda} A = 0$.
- Require that $\lambda_i \geq 0$, $i = 1, \ldots n$.
- Given a $\vec{\lambda}$, it follows that $\vec{\lambda}(A\vec{y} + B\vec{z}) \leq \vec{\lambda}\vec{c}$, hence $\vec{\lambda} B\vec{z} \leq \vec{\lambda}\vec{c}$, which is a single inequality in the projection space.
- If $\vec{\lambda}$ is a solution, so is $s\vec{\lambda}$, $s > 0$, hence require $\lambda_1 + \ldots + \lambda_n = 1$.
- Find vertices of the polytope $\vec{\lambda} A = 0, \lambda_i \geq 0, \lambda_1 + \ldots + \lambda_n = 1$. The set of all vertices $\vec{\lambda_1}, \ldots \vec{\lambda_m}$ define projection space.

Generating Useful Inequalities

Use Simplex to find vertices $\vec{\lambda}$ of $\vec{\lambda}A = 0, \lambda_i \geq 0, \lambda_1 + \ldots + \lambda_n = 1$.

Need a goal function!

Observation:

Generating Useful Inequalities

Use Simplex to find vertices $\vec{\lambda}$ of $\vec{\lambda}A = 0, \lambda_i \geq 0, \lambda_1 + \ldots + \lambda_n = 1$.

Need a goal function!

Observation:

- [Kohler] Given $\vec{\lambda^a}, \vec{\lambda^b}$, if $\{i \mid \lambda_i^a = 0\} \supset \{i \mid \lambda_i^b = 0\}$ then $\lambda^b B \leq \lambda^b \vec{c}$ will be redundant.

Generating Useful Inequalities

Use Simplex to find vertices $\vec{\lambda}$ of $\vec{\lambda}A = 0, \lambda_i \geq 0, \lambda_1 + \ldots + \lambda_n = 1$.

Need a goal function!

Observation:

- [Kohler] Given $\vec{\lambda^a}, \vec{\lambda^b}$, if $\{i \mid \lambda^a_i = 0\} \supset \{i \mid \lambda^b_i = 0\}$ then $\lambda^b B \leq \lambda^b \vec{c}$ will be redundant.

Idea: Run Simplex for $\langle 1, 0, \ldots 0\rangle$, $\langle 0, 1, 0, \ldots 0\rangle$, $\ldots \langle 0, \ldots 0, 1\rangle$.

Generating Useful Inequalities

Use Simplex to find vertices $\vec{\lambda}$ of $\vec{\lambda}A = 0, \lambda_i \geq 0, \lambda_1 + \ldots + \lambda_n = 1$.

Need a goal function!

Observation:

- [Kohler] Given $\vec{\lambda^a}, \vec{\lambda^b}$, if $\{i \mid \lambda_i^a = 0\} \supset \{i \mid \lambda_i^b = 0\}$ then $\lambda^b B \leq \lambda^b \vec{c}$ will be redundant.

Idea: Run Simplex for $\langle 1, 0, \ldots 0\rangle$, $\langle 0, 1, 0, \ldots 0\rangle$, $\ldots \langle 0, \ldots 0, 1\rangle$.

- Creates at most n inequalities in the projection space.

Argument-Size Analysis for Prolog

Analysis times on a 2.4GHz, 512MB RAM PC using classic widening if SCCs are not stable after two iterations. Inequalities with excessive coefficients are removed.

		vars approx'ed		sparsity			
benchmark	LOC	ratio	%	dim	ineq	vars	time
sim	1071	0/2412	0.0	12.0	20.1	1.3	0.61
rubik	1229	0/1062	0.0	5.7	9.4	1.5	0.20
chat	4698	105/7917	1.3	9.7	19.1	1.5	4.58
pl2wam	4775	96/4078	2.3	8.0	13.4	1.5	3.20
lptp	7419	213/12525	1.7	8.2	15.2	1.4	9.97
aqua_c	15026	493/32340	1.5	10.3	19.5	1.5	27.59

$\leadsto$ Results seem comparable to classic polyhedra (cTI).

Conclusion

- Program analyses often generate sparse inequalities.
- Fourier-Motzkin projection works well on sparse systems.
- Projection can be approximated if output becomes large.
- Calculating convex hull without reverting to frame representation yields incremental algorithm.

Future Work:

- More optimisations possible (e.g. Kohler's rule).