Computing the Integer Points of a Polyhedron

Rui-Juan Jing
Joint work with
Marc Moreno Maza

SCsquare 2017, July 29
Plan

Overview

Algorithm

Complexity analysis

Experiments

Application

Summary
Motivations-Cholesky’s LU decomposition

Cholesky’s LU decomposition:

1: for \((i = 1; i <= n; i + +)\) {
    
    \(x = a[i][i];\)
    
    for \((k = 1; k < i; k + +)\)
    
    \(x = x - a[i][k] * a[i][k];\)

2: 

3: \(p[i] = 1.0/\text{sqrt}(x);\)

    for \((j = i + 1; j <= n; j + +)\) {

4: 

5: \(x = x - a[j][k] * a[i][k];\)

6: \(a[j][i] = x * p[i];\)

} 

}
Motivations-Cholesky’s LU decomposition

Cholesky’s LU decomposition:

1: for ($i = 1; i <= n; i ++$)
   
   \[ x = a[i][i]; \]
   
   for ($k = 1; k < i; k ++$)
   
   \[ x = x - a[i][k] \times a[i][k]; \]

2: \[ p[i] = 1.0 / \text{sqrt}(x); \]
   
   for ($j = i + 1; j <= n; j ++$)
   
   \[ x = a[i][j]; \]
   
   for ($k = 1; k < i; k ++$)
   
   \[ x = x - a[j][k] \times a[i][k]; \]

3: \[ a[j][i] = x \times p[i]; \]

system 1:

\begin{align*}
1 \leq i & \leq n \\
1 \leq k & \leq i - 1 \\
1 \leq i' & \leq n \\
1 \leq i' & \leq n \\
i' + 1 \leq j' & \leq n \\
n \leq j' \leq n \\
i < i' \\
i < i' \\
\end{align*}

system 2:

\begin{align*}
1 \leq i & \leq n \\
1 \leq k & \leq i - 1 \\
1 \leq i' & \leq n \\
1 \leq i' & \leq n \\
i' + 1 \leq j' & \leq n \\
j = j' \leq n \\
1 \leq i' & \leq n \\
1 \leq i' \leq n \\
i = i', j < j' \\
i = i', j < j' \\
\end{align*}

system 3:

\begin{align*}
1 \leq i & \leq n \\
1 \leq k & \leq i - 1 \\
1 \leq i' & \leq n \\
1 \leq i' & \leq n \\
i' + 1 \leq j' & \leq n \\
j = j' \leq n \\
1 \leq i' & \leq n \\
1 \leq i' \leq n \\
i = i', j = j' \\
i = i', j = j' \\
\end{align*}
Motivations-Cache line accessed by a for-loop

for $i = 2$ to $N - 1$ do
  for $j = 2$ to $N - 1$ do
    $a(i, j) = 2 \times a(i, j) + a(i - 1, j) + a(i + 1, j) + a(i, j - 1) + a(i, j + 1)$
Motivations-Cache line accessed by a for-loop

for $i = 2$ to $N - 1$ do
  for $j = 2$ to $N - 1$ do
    $a(i, j) = 2 \times a(i, j) + a(i - 1, j) + a(i + 1, j) + a(i, j - 1) + a(i, j + 1)$

Cache lines touched by this loop:
$(\Sigma x, y : (\exists i, j, \triangle i, \triangle j :$)
\[
x = (i + \triangle i - 1) \div 16 \land y = j + \triangle j \land 2 \leq i, j \leq N - 1
\land -1 \leq \triangle i + \triangle j, \triangle i - \triangle j \leq 1)
\]
: 1)
Motivations—Cache line accessed by a for-loop

for $i = 2$ to $N - 1$ do
  for $j = 2$ to $N - 1$ do
    $a(i, j) = 2 \times a(i, j) + a(i - 1, j) + a(i + 1, j) + a(i, j - 1) + a(i, j + 1)$

Cache lines touched by this loop:
$(\sum x, y : (\exists i, j, \triangle i, \triangle j :$

$x = (i + \triangle i - 1) \div 16 \land y = j + \triangle j \land 2 \leq i, j \leq N - 1$

$\land -1 \leq \triangle i + \triangle j, \triangle i - \triangle j \leq 1)$

: 1)

\[
\begin{align*}
0 \leq (i + \triangle i - 1)/16 - x < 1 \\
y = j + \triangle j \\
2 \leq i, j \leq N - 1 \\
-1 \leq \triangle i + \triangle j, \triangle i - \triangle j \leq 1
\end{align*}
\]
Motivations-Cache line accessed by a for-loop

for \( i = 2 \) to \( N - 1 \) do
  for \( j = 2 \) to \( N - 1 \) do
    \( a(i, j) = 2 \times a(i, j) + a(i - 1, j) + a(i + 1, j) + a(i, j - 1) + a(i, j + 1) \)

Cache lines touched by this loop:
(\( \Sigma x, y : (\exists i, j, \triangle i, \triangle j : \)
\( x = (i + \triangle i - 1) \div 16 \wedge y = j + \triangle j \wedge 2 \leq i, j \leq N - 1 \)
\( \wedge -1 \leq \triangle i + \triangle j, \triangle i - \triangle j \leq 1 \)
\( : 1 ) \)

\[
\begin{align*}
0 \leq (i + \triangle i - 1)/16 - x < 1 \\
y = j + \triangle j \\
2 \leq i, j \leq N - 1 \\
-1 \leq \triangle i + \triangle j, \triangle i - \triangle j \leq 1
\end{align*}
\]

Simplify using our code

\[
\begin{align*}
-x & \leq 0, \quad 16x - y - N \leq -3 \\
16x - N & \leq -1, \quad 16x + y - 2N \leq -2 \\
1 \leq y - N & \leq 0, \quad -N \leq -3
\end{align*}
\]
Motivations-

Cache line accessed by a for-loop

for $i = 2$ to $N - 1$ do
  for $j = 2$ to $N - 1$ do
    $a(i, j) = 2 \times a(i, j) + a(i - 1, j) + a(i + 1, j) + a(i, j - 1) + a(i, j + 1)$

Cache lines touched by this loop:
($\sum \ x, y : (\exists \ i, j, \triangle i, \triangle j :$

\begin{align*}
  x &= (i + \triangle i - 1) \div 16 \land y = j + \triangle j \land 2 \leq i, j \leq N - 1 \\
  &\land -1 \leq \triangle i + \triangle j, \triangle i - \triangle j \leq 1)
\end{align*}$

: 1)

\begin{align*}
  &\begin{cases}
  0 \leq (i + \triangle i - 1)/16 - x < 1 \\
  y = j + \triangle j \\
  2 \leq i, j \leq N - 1 \\
  -1 \leq \triangle i + \triangle j, \triangle i - \triangle j \leq 1
\end{cases}
\end{align*}

Simplify using our code

\begin{align*}
  &\begin{cases}
  -x \leq 0, \quad 16x - y - N \leq -3 \\
  16x - N \leq -1, \quad 16x + y - 2N \leq -2 \\
  1 \leq y - N \leq 0, \quad -N \leq -3
\end{cases}
\end{align*}

When $N = 500$, ($\sum \ x, y : 0 \leq x \leq 31 \land 1 \leq y \leq 500 : 1) = 16000$
Related Work

1. Fourier-Motzkin elimination: computing the rational points (thus all the points) of a polyhedron in \( \mathbb{R}^d \) given by \( m \) inequalities; Complexity: polynomial in \( m^d \), thus **single exponential in** \( d \) (Fourier-Motzkin algorithm; L. Khachiyan, 2009)

2. Counting the number of integer points of a bounded polyhedron; Complexity: polynomial for fixed dimension. (A. Barvinok, 1999)

3. Deciding Presburger arithmetic such as

\[
(\forall x \in \mathbb{Z}) (\exists y \in \mathbb{Z}) : (y + y = x) \lor (y + y + 1 = x)
\]

Complexity: **doubly exponential in** \( d \)
(Fischer & Rabin, 1974).

4. **Omega test**, can decide Presburger arithmetic; essential in the analysis and transformation of computer programs; (W. Pugh, 1991). Complexity: No complexity estimate known until our work.
Our Contribution

1. Based on the Omega test, we propose an algorithm for decomposing a polyhedron into “simpler” polyhedra, each of them having at least one integer point and good structural properties;

2. Under a mild assumption (almost always verified in practice), this decomposition can be computed within

\[ O(m^2d^2 d^4d^4 L^4d^3 \text{LP}(d, m^d d^4 (\log d + \log L))) \]

bit operations, where \( \text{LP}(d, H) \) is an upper bound for solving a linear program with total bit size \( H \) and \( d \) variables;

3. Implement two versions of our algorithm in Maple:
   One with equations and inequalities as input and intermediate operations;
   Another with matrices as input and intermediate operations.
Example

Input: $K_1$:
\[
\begin{align*}
3x_1 - 2x_2 + x_3 &\leq 7 \\
-2x_1 + 2x_2 - x_3 &\leq 12 \\
-4x_1 + x_2 + 3x_3 &\leq 15 \\
-2x_2 - x_3 &\leq 48
\end{align*}
\]
, assume $x_1 > x_2 > x_3$.

Output: $K_1^1, K_1^2, K_1^3, K_1^4, K_1^5$ given by:
\[
\begin{align*}
3x_1 - 2x_2 + x_3 &\leq 7 \\
-2x_1 + 2x_2 - x_3 &\leq 12 \\
-4x_1 + x_2 + 3x_3 &\leq 15 \\
2x_2 - x_3 &\leq 48 \\
-5x_2 + 13x_3 &\leq 67 \\
x_1 &= 15 \\
x_2 &= 27 \\
x_3 &= 16
\end{align*}
\]
\[
\begin{align*}
3x_1 - 2x_2 + x_3 &\leq 7 \\
-2x_1 + 2x_2 - x_3 &\leq 12 \\
-4x_1 + x_2 + 3x_3 &\leq 15 \\
2x_2 - x_3 &\leq 48 \\
-5x_2 + 13x_3 &\leq 67 \\
x_1 &= 18 \\
x_2 &= 33 \\
x_3 &= 18
\end{align*}
\]
\[
\begin{align*}
3x_1 - 2x_2 + x_3 &\leq 7 \\
-2x_1 + 2x_2 - x_3 &\leq 12 \\
-4x_1 + x_2 + 3x_3 &\leq 15 \\
2x_2 - x_3 &\leq 48 \\
-5x_2 + 13x_3 &\leq 67 \\
x_1 &= 14 \\
x_2 &= 25 \\
x_3 &= 15
\end{align*}
\]
\[
\begin{align*}
3x_1 - 2x_2 + x_3 &\leq 7 \\
-2x_1 + 2x_2 - x_3 &\leq 12 \\
-4x_1 + x_2 + 3x_3 &\leq 15 \\
2x_2 - x_3 &\leq 48 \\
-5x_2 + 13x_3 &\leq 67 \\
x_1 &= 19 \\
x_2 &= 50 + t \\
x_3 &= 50 + 2t
\end{align*}
\]
\[
\begin{align*}
-25 \leq t &\leq -16.
\end{align*}
\]
Decomposing the integer points of a polyhedron

Output: $K_1^1, K_1^2, K_1^3, K_1^4, K_1^5$ given by:

$$
\begin{align*}
3x_1 - 2x_2 + x_3 &\leq 7 \\
-2x_1 + 2x_2 - x_3 &\leq 12 \\
-4x_1 + x_2 + 3x_3 &\leq 15 \\
2x_2 - x_3 &\leq 48 \\
-5x_2 + 13x_3 &\leq 67 \\
-x_2 &\leq -25 \\
2 &\leq x_3 \leq 17
\end{align*}
$$

$\quad$,

$\begin{align*}
x_1 &= 15 \\
x_2 &= 27 \\
x_3 &= 16
\end{align*}$

$\quad$,

$\begin{align*}
x_1 &= 18 \\
x_2 &= 33 \\
x_3 &= 18
\end{align*}$

$\quad$,

$\begin{align*}
x_1 &= 14 \\
x_2 &= 25 \\
x_3 &= 15
\end{align*}$

$\quad$,

$\begin{align*}
x_1 &= 19 \\
x_2 &= 50 + t \\
x_3 &= 50 + 2t
\end{align*}$

$\quad$,

$\begin{align*}
-25 &\leq t \leq -16.
\end{align*}$

$\quad$

- An integer point solves $K_1$ iff it solves either $K_1^1, K_1^2, K_1^3, K_1^4$ or $K_1^5$.
- Each of $K_1^1, K_1^2, K_1^3, K_1^4, K_1^5$ has at least one integer point.
- For each $K_1^i$, each integer point in the projection can be lifted to an integer point in the polyhedron.
Plan

Overview

Algorithm

Complexity analysis

Experiments

Application

Summary
Consider the polyhedron $K$ of $\mathbb{R}^4$ given below ($Ax \leq b$):

$$
\begin{align*}
2x + 3y - 4z + 3w & \leq 1 \\
-2x - 3y + 4z - 3w & \leq -1 \\
-13x - 18y + 24z - 20w & \leq -1 \\
-26x - 40y + 54z - 39w & \leq 0 \\
-24x - 38y + 49z - 31w & \leq 5 \\
54x + 81y - 109z + 81w & \leq 2
\end{align*}
$$
Algorithm-IntegerNormalize

Procedure 1- IntegerNormalize($Ax \leq b$):

1. Solve integer solutions for (implicit) equations:
   
   - Tools: Hermite normal form;
   - Return $x = Pt + q$, where $t$ is a new unknown vector with less length than $x$.

2. Substitute $x = Pt + q$ into $Ax \leq b$ and remove redundant inequalities:
   
   - $cx \leq d$ is implied by $Ax \leq b \iff \sup\{-(cx - d) | Ax \leq b\} = 0$;
   - Return $Mt \leq v$. 

In our example, implicit equation: $2x + 3y - 4z + 3w = 1$.

The systems $x = Pt + q$ and $Mt \leq v$ are given by:

\[
\begin{align*}
  x &= -3t_1 + 2t_2 - 3t_3 + 2y \\
  y &= t_1 + t_3 - 1z \\
  z &= t_2 \\
  w &= t_3 \\
  3t_1 - 2t_2 + t_3 &\leq 7 \\
  -2t_1 + 2t_2 - t_3 &\leq 12 \\
  -4t_1 + t_2 + 3t_3 &\leq 15 \\
  -t_2 &\leq -25 
\end{align*}
\]
Algorithm-IntegerNormalize

**Procedure 1** - IntegerNormalize($Ax \leq b$):

1. Solve integer solutions for (implicit) equations:
   - Tools: Hermite normal form;
   - Return $x = Pt + q$, where $t$ is a new unknown vector with less length than $x$.

2. Substitute $x = Pt + q$ into $Ax \leq b$ and remove redundant inequalities:
   - $cx \leq d$ is implied by $Ax \leq b \iff \sup\{- (cx - d) | Ax \leq b\} = 0$;
   - Return $Mt \leq v$.

In our example, implicit equation: $2x + 3y - 4z + 3w = 1$
the systems $x = Pt + q$ and $Mt \leq v$ are given by:

\[
\begin{align*}
  x &= -3t_1 + 2t_2 - 3t_3 + 2 \\
y &= 2t_1 + t_3 - 1 \\
z &= t_2 \\
w &= t_3
\end{align*}
\]

and

\[
\begin{align*}
  3t_1 - 2t_2 + t_3 &\leq 7 \\
-2t_1 + 2t_2 - t_3 &\leq 12 \\
-4t_1 + t_2 + 3t_3 &\leq 15 \\
-t_2 &\leq -25
\end{align*}
\]
Algorithm-DarkShadow

**Procedure 2—DarkShadow**$(\mathbf{M} \mathbf{t} \leq \mathbf{v})$: considering the variable $t_1$, for any upper bound $l_1 : a_1 t_1 + a't' \leq v_1$ with $a_1 > 0$ and lower bound $l_2 : b_1 t_1 + b't' \leq v_2$ with $b_1 < 0$ do:

$$-b_1 a't' + a_1 b't' \leq -b_1 v_1 + a_1 v_2 - (a_1 - 1)(-b_1 - 1) \leftarrow \text{dark projection}$$

returns a couple $(t', \Theta)$, where

1. $t'$ stands for all $t$-variables except $t_1$,
2. $\Theta$ is a linear system in the $t'$-variables such that any integer point solving of $\Theta$ is the collection of all the dark projections generated by pair of upper and lower bound of $t_1$. 

In our example, $t' = \{t_2, t_3\}$ and $\Theta$ is given by:

$$
\begin{align*}
2t_2 - t_3 & \leq 48 \\
-5t_2 + 13t_3 & \leq 67 \\
-t_2 & \leq -25
\end{align*}
$$
Algorithm-DarkShadow

**Procedure 2—DarkShadow** (*Mt ≤ v*):

considering the variable *t*₁, for any upper bound *l*₁ : \( a₁ t₁ + a' t' \leq v₁ \) with \( a₁ > 0 \) and lower bound *l*₂ : \( b₁ t₁ + b' t' \leq v₂ \) with \( b₁ < 0 \) do:

\[-b₁ a' t' + a₁ b' t' \leq -b₁ v₁ + a₁ v₂ - (a₁ - 1)(-b₁ - 1) \leftarrow \text{dark projection}\]

returns a couple \((t', \Theta)\), where

1. \( t' \) stands for all \( t \)-variables except \( t₁ \),
2. \( \Theta \) is a linear system in the \( t' \)-variables such that any integer point solving of \( \Theta \) is the collection of all the dark projections generated by pair of upper and lower bound of \( t₁ \).

In our example, \( t' = \{t₂, t₃\} \) and \( \Theta \) is given by:

\[
\begin{align*}
2t₂ - t₃ & \leq 48 \\
-5t₂ + 13t₃ & \leq 67 \\
-t₂ & \leq -25
\end{align*}
\]
Algorithm-definitions

**real shadow**: standard projection on \((t_2, \ldots, t_d)\), denoted as \(R\);
Let \(d_1, \ldots, d_r\) be the dark projections computed by DarkShadow\((\mathbf{M}t \leq \mathbf{v})\).

**dark shadow** \(D := R \cap d_1 \cap \cdots \cap d_r\)

**grey shadow**: \(G := R \setminus D\)

![Figure: The real, the dark and the grey shadows of a polyhedron.]

\((t_2, t_3) = (29, 9) \in G\), which can not extend to an integer solution of \(\mathbf{M}t \leq \mathbf{v}\). (Plugging \((t_2, t_3) = (29, 9)\) into \(\mathbf{M}t \leq \mathbf{v}\) yields \(\frac{37}{2} \leq t_1 \leq \frac{56}{3}\), which has no integer solutions.)
**Third procedure**–GreyShadow($\mathbf{M} \mathbf{t} \leq \mathbf{v}$)

Disjoint decomposition: $R \setminus D = \bigcup_{1 \leq i \leq t} G_i$, where

$$G_i = R \cap d_1 \cap \cdots \cap d_{i-1} \cap \neg d_i$$

and $\neg d_i$ is the negation of $d_i$ for $1 \leq i \leq r$.

Considering the variable $t_1$ again, for any upper bound $l_1 : a_1 t_1 + a' t' \leq v_1$ with $a_1 > 0$ and any lower bound $l_2 : b_1 t_1 + b' t' \leq v_2$ with $b_1 < 0$ do:

1. let $B := \lfloor (-a_1 b_1 - a_1 + b_1)/a_1 \rfloor$;
2. for $i$ from 0 to $B$ output what IntegerSolve returns when applied to
   $$\{b_1 t_1 + b' t' = v_2 - i\} \cap \mathbf{M} \mathbf{t} \leq \mathbf{v} \cap$$
   $$\{ -b_1 a' t' + a_1 b' t' > -b_1 v_1 + a_1 v_2 - (a_1 - 1)(-b_1 - 1) \},$$
3. replace $\mathbf{M} \mathbf{t} \leq \mathbf{v}$ by
   $$\mathbf{M} \mathbf{t} \leq \mathbf{v} \cap \{ -b_1 a' t' + a_1 b' t' \leq -b_1 v_1 + a_1 v_2 - (a_1 - 1)(-b_1 - 1) \}.$$
Returning to our example, combining system $\mathbf{M} \mathbf{t} \leq \mathbf{v}$ with the negation of $2t_2 - t_3 \leq 48$ from $\Theta$, yields

$$\begin{cases} -2t_1 + 2t_2 - t_3 = 12 \\ 3t_1 - 2t_2 + t_3 \leq 7 \\ -4t_1 + t_2 + 3t_3 \leq 15 \\ -t_2 \leq -25 \\ -2t_2 + t_3 \leq -49 \end{cases}$$

IntegerNormalize new variables $t_4, t_5$

$$\begin{cases} t_1 = t_4 \\ t_2 = t_5 + 1 \\ t_3 = -2t_4 + 2t_5 + 1 \end{cases}$$

$$\begin{cases} t_4 \leq 8 \\ -10t_4 + 7t_5 \leq 11 \\ -t_5 \leq -24 \\ -2t_4 - t_5 \leq -48 \end{cases}$$
Algorithm-Illustration

Figure: IntegerSolve
Algorithm

Continuing in this manner, the integer points of $\mathbf{M} \mathbf{t} \leq \mathbf{v}$ decomposes into:

$$\begin{cases}
3t_1 - 2t_2 + t_3 \leq 7 \\
-2t_1 + 2t_2 - t_3 \leq 12 \\
-4t_1 + t_2 + 3t_3 \leq 15 \\
2t_2 - t_3 \leq 48 \\
-5t_2 + 13t_3 \leq 67 \\
-t_2 \leq -25 \\
2 \leq t_3 \leq 17
\end{cases}, \begin{cases}
t_1 = 15 \\
t_2 = 27 \\
t_3 = 16
\end{cases}, \begin{cases}
t_1 = 18 \\
t_2 = 33 \\
t_3 = 18
\end{cases}, \begin{cases}
t_1 = 14 \\
t_2 = 25 \\
t_3 = 15
\end{cases} \begin{cases}
t_1 = 19 \\
t_2 = 50 + t_6 \\
t_3 = 50 + 2t_6
\end{cases} \begin{cases}
-25 \leq t_6 \leq -16.
\end{cases}$
Plan

Overview

Algorithm

Complexity analysis

Experiments

Application

Summary
Complexity

Lemma
Let $K$ be a polyhedron in $\mathbb{R}^d$, defined by $m$ inequalities. Let $f_{d,m,k}$ be the number of $k$-dimensional faces of $K$. Then, we have

$$f_{d,m,k} \leq \binom{m}{d-k}.$$ 

Therefore, we have

$$f_{d,m,0} + f_{d,m,1} + \cdots + f_{d,m,d-1} \leq m^d.$$
Lemma
Let $K$ be a polyhedron in $\mathbb{R}^d$, defined by $m$ inequalities. Let $f_{d,m,k}$ be the number of $k$-dimensional faces of $K$. Then, we have

$$f_{d,m,k} \leq \binom{m}{d-k}.$$ 

Therefore, we have $f_{d,m,0} + f_{d,m,1} + \cdots + f_{d,m,d-1} \leq m^d$.

Notation
Given a linear program with total bit size $H$ and with $d$ variables $\text{LP}(d, H)$: the number of bit operations required for solving it. Karmarkar’s algorithm: $\text{LP}(d, H) \in O(d^{3.5} H^2 \log H \cdot \log \log H)$. 
Complexity

Proposition
Given a polyhedron $K$ in $\mathbb{R}^d$, which is defined by $m$ inequalities and with maximum bit size $h$, one can perform Fourier-Motzkin elimination within $O(d^2 m^2 d \text{LP}(d, 2^d hd^2 m^d))$ bit operations.

Hypothesis
During the execution of the function call $\text{IntegerSolve}(K)$, for any polyhedral set $K$, input of a recursive call, each facet of the dark shadow of a polyhedron is parallel to some facet of its real shadow.

Theorem
Under our Hypothesis, the function call $\text{IntegerSolve}(K)$ runs within $O(m^{2d^2} d^{4d^3} L^{4d^3} \text{LP}(d, m^d d^4 (\log d + \log L)))$ bit operations.
Plan

Overview

Algorithm

Complexity analysis

Experiments

Application

Summary
Experiments

IntegerSolve is implemented in the Polyhedra library and available from www.regularchains.org

<table>
<thead>
<tr>
<th>Example</th>
<th>$m$</th>
<th>$d$</th>
<th>$L$</th>
<th>$m_o$</th>
<th>$L_o$</th>
<th>?Hyp</th>
<th>$t_H$</th>
<th>$t_P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tetrahedron</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>yes</td>
<td>0.695</td>
<td>0.697</td>
</tr>
<tr>
<td>Truncated Tetrahedron</td>
<td>8</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>yes</td>
<td>1.461</td>
<td>1.468</td>
</tr>
<tr>
<td>Presburger 4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>12</td>
<td>yes</td>
<td>0.706</td>
<td>0.871</td>
</tr>
<tr>
<td>Presburger 6</td>
<td>4</td>
<td>5</td>
<td>89</td>
<td>6</td>
<td>35</td>
<td>yes</td>
<td>0.893</td>
<td>0.746</td>
</tr>
<tr>
<td>Bounded 7</td>
<td>8</td>
<td>3</td>
<td>19</td>
<td>3</td>
<td>190</td>
<td>no</td>
<td>138.448</td>
<td>239.637</td>
</tr>
<tr>
<td>Bounded 8</td>
<td>4</td>
<td>3</td>
<td>25</td>
<td>5</td>
<td>67</td>
<td>yes</td>
<td>6.462</td>
<td>3.821</td>
</tr>
<tr>
<td>Bounded 9</td>
<td>6</td>
<td>3</td>
<td>18</td>
<td>6</td>
<td>74</td>
<td>no</td>
<td>23.574</td>
<td>16.763</td>
</tr>
<tr>
<td>Unbounded 2</td>
<td>3</td>
<td>4</td>
<td>10</td>
<td>61</td>
<td>2255</td>
<td>no</td>
<td>0.547</td>
<td>0.600</td>
</tr>
<tr>
<td>Unbounded 5</td>
<td>5</td>
<td>4</td>
<td>8</td>
<td>1</td>
<td>8</td>
<td>no</td>
<td>1.321</td>
<td>1.319</td>
</tr>
<tr>
<td>Unbounded 6</td>
<td>10</td>
<td>4</td>
<td>8</td>
<td>1</td>
<td>8</td>
<td>no</td>
<td>1.494</td>
<td>1.479</td>
</tr>
<tr>
<td>P91</td>
<td>12</td>
<td>3</td>
<td>96</td>
<td>5</td>
<td>96</td>
<td>no</td>
<td>19.318</td>
<td>15.458</td>
</tr>
<tr>
<td>Sys1</td>
<td>6</td>
<td>3</td>
<td>15</td>
<td>2</td>
<td>67</td>
<td>yes</td>
<td>2.413</td>
<td>1.915</td>
</tr>
<tr>
<td>Sys3</td>
<td>8</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>yes</td>
<td>1.481</td>
<td>1.479</td>
</tr>
<tr>
<td>Automatic</td>
<td>8</td>
<td>2</td>
<td>999</td>
<td>1</td>
<td>999</td>
<td>yes</td>
<td>0.552</td>
<td>0.549</td>
</tr>
<tr>
<td>Automatic2</td>
<td>6</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>yes</td>
<td>1.115</td>
<td>1.113</td>
</tr>
</tbody>
</table>

**Table:** Implementation
Plan

Overview

Algorithm

Complexity analysis

Experiments

Application

Summary
Application

Solve integer programming:
\[
\min_{\text{lex}} (x_1, \ldots, x_d) \\
Ax \leq b, \\
x \in \mathbb{Z}^d
\]

Example

Problem:
\[
\min_{\text{lex}} (x_3, x_2, x_1) \\
3x_1 - 2x_2 + x_3 \leq 7 \\
-2x_1 + 2x_2 - x_3 \leq 12 \\
-4x_1 + x_2 + 3x_3 \leq 15 \\
-x_2 \leq -25 \\
x_1, x_2, x_3 \in \mathbb{Z}
\]
**Application**

**Example**

Input: $K_1 : \begin{cases} 3x_1 - 2x_2 + x_3 \leq 7 \\ -2x_1 + 2x_2 - x_3 \leq 12 \\ -4x_1 + x_2 + 3x_3 \leq 15 \\ 2x_2 - x_3 \leq 48 \\ -5x_2 + 13x_3 \leq 67 \\ -x_2 \leq -25 \\ 2 \leq x_3 \leq 17 \end{cases}$, assume $x_1 > x_2 > x_3$.

Output: $K_1^1, K_1^2, K_1^3, K_1^4, K_1^5$ given by:

\[
\begin{cases} 3x_1 - 2x_2 + x_3 \leq 7 \\ -2x_1 + 2x_2 - x_3 \leq 12 \\ -4x_1 + x_2 + 3x_3 \leq 15 \\ x_1 = 15 \end{cases}, \begin{cases} x_1 = 15 \\ x_1 = 18 \end{cases}, \begin{cases} x_1 = 14 \\ x_2 = 27 \end{cases}, \begin{cases} x_2 = 33 \end{cases}, \begin{cases} x_2 = 25 \end{cases}, \begin{cases} x_3 = 16 \end{cases}, \begin{cases} x_3 = 18 \end{cases}, \begin{cases} x_3 = 15 \end{cases}, \begin{cases} x_1 = 19 \\ x_2 = 50 + t \end{cases}, \begin{cases} x_3 = 50 + 2t \end{cases}, \begin{cases} -25 \leq t \leq -16. \end{cases}
\]
Application

\[
\min(x_3, x_2, x_1) \\
K_1 \cap \mathbb{Z}^3
\]

\[\downarrow\]

\[
\min(x_3, x_2, x_1) \\
K_1^1 \cap \mathbb{Z}^3
\]
\[
\downarrow
\]

(2, -8, -4)

\[
\min(x_3, x_2, x_1) \\
K_1^2 \cap \mathbb{Z}^3
\]
\[
\downarrow
\]

(16, 27, 15)

\[
\min(x_3, x_2, x_1) \\
K_1^3 \cap \mathbb{Z}^3
\]
\[
\downarrow
\]

(18, 33, 18)

\[
\min(x_3, x_2, x_1) \\
K_1^4 \cap \mathbb{Z}^3
\]
\[
\downarrow
\]

(15, 25, 14)

\[
\min(x_3, x_2, x_1) \\
K_1^5 \cap \mathbb{Z}^3
\]
\[
\downarrow
\]

(0, 25, 19)

\[
(0, 25, 19)
\]
Plan

Overview

Algorithm

Complexity analysis

Experiments

Application

Summary
Summary

- We have presented an algorithm for computing the integer points of a polyhedron, based on the Omega test procedure proposed by W. Pugh.
Summary

- We have presented an algorithm for computing the integer points of a polyhedron, based on the Omega test procedure proposed by W. Pugh.
- This is done by decomposing the input polyhedron into simpler polyhedra, each of them with at least one integer point.
Summary

- We have presented an algorithm for computing the integer points of a polyhedron, based on the Omega test procedure proposed by W. Pugh.
- This is done by decomposing the input polyhedron into simpler polyhedra, each of them with at least one integer point.
- This kind of simpler polyhedra has good structure which will help to solve the lexicographic minimum of some variable orders.
Summary

- We improve it by making use of **Hermite normal form** and controlling the size of the intermediate coefficients.
Summary

- We improve it by making use of Hermite normal form and controlling the size of the intermediate coefficients.
- Assuming that each facet of the dark shadow of a polyhedron is parallel to some facet of its real shadow, we prove that our algorithm runs in time single exponential in the dimension $d$ of the ambient space.
Summary

- We improve it by making use of Hermite normal form and controlling the size of the intermediate coefficients.
- Assuming that each facet of the dark shadow of a polyhedron is parallel to some facet of its real shadow, we prove that our algorithm runs in time single exponential in the dimension $d$ of the ambient space.
- This assumption is almost always verified in practice.
Summary

- We improve it by making use of Hermite normal form and controlling the size of the intermediate coefficients.
- Assuming that each facet of the dark shadow of a polyhedron is parallel to some facet of its real shadow, we prove that our algorithm runs in time single exponential in the dimension $d$ of the ambient space.
- This assumption is almost always verified in practice.

Works in progress

- A CilkPlus version of the Polyhedra library
- Parametric integer programming (PIP) in support of automatic parallelization.