Computing the Integer Points of a Polyhedron

Rui-Juan Jing
Joint work with
Marc Moreno Maza

SCsquare 2017, July 29

Plan

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++)
2:        x = x - a[i][k] * a[i][k];
3:      p[i] = 1.0/sqrt(x);
        for(j = i + 1; j <= n; j++){
4:        x = a[i][j];
          for(k = 1; k < i; k++)
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++)
2:        x = x - a[i][k] * a[i][k];
3:      p[i] = 1.0/sqrt(x);
        for(j = i + 1; j <= n; j++){
4:        x = a[i][j];
          for(k = 1; k < i; k++)
5:          x = x - a[j][k] * a[i][k];
6:        a[j][i] = x * p[i];
        }
    }
```

system 1:

$$\begin{cases} 1 \le i \le n \\ i+1 \le j \le n \\ 1 \le k \le i-1 \\ 1 \le i' \le n \\ i'+1 \le j' \le n \\ j = j', k = i' \\ i < i' \end{cases}$$

system 2:

$$\begin{cases} 1 \le i \le n \\ i+1 \le j \le n \\ 1 \le k \le i-1 \\ 1 \le i' \le n \\ i'+1 \le j' \le n \\ j = j', k = i' \\ i = i', j < j' \end{cases}$$

system 3:

$$\begin{cases} 1 \le i \le n \\ i+1 \le j \le n \\ 1 \le k \le i-1 \\ 1 \le i' \le n \\ i'+1 \le j' \le n \\ j = j', k = i' \\ i = i', j = j' \end{cases}$$

Motivations-Cache line accessed by a for-loop

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

Motivations-Cache line accessed by a for-loop

for $i = 2$ to $N-1$ do

for $j = 2$ to $N-1$ do

$$a(i,j) = 2 * a(i,j) + a(i-1,j) + a(i+1,j) + a(i,j-1) + a(i,j+1)$$

Cache lines touched by this loop:

$$(\Sigma\ x, y : (\exists\ i, j, \triangle i, \triangle j :$$

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

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

$$: 1)$$

Motivations-Cache line accessed by a for-loop

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

Cache lines touched by this loop:

$$(\Sigma\ x, y : (\exists\ i, j, \triangle i, \triangle j :$$
$$x = (i + \triangle i - 1) \div 16 \wedge y = j + \triangle j \wedge 2 \leq i, j \leq N-1$$
$$\wedge -1 \leq \triangle i + \triangle j, \triangle i - \triangle j \leq 1)$$
$$: 1)$$

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

Motivations-Cache line accessed by a for-loop

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

Cache lines touched by this loop:

$$(\Sigma\ x, y : (\exists\ i, j, \triangle i, \triangle j :$$
$$x = (i + \triangle i - 1) \div 16 \wedge y = j + \triangle j \wedge 2 \leq i, j \leq N-1$$
$$\wedge -1 \leq \triangle i + \triangle j, \triangle i - \triangle j \leq 1)$$
$$: 1)$$

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

Simplify using our code

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

Motivations-Cache line accessed by a for-loop

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

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

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

Simplify using our code

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

When $N = 500$, $(\Sigma\ x, y : 0 \leq x \leq 31 \wedge 1 \leq y \leq 500 : 1) = 16000$

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 dimention.
   (A. Barvinok, 1999)
3. Deciding Presburger arithmetic such as
   $$(\forall x \in \mathbb{Z})\,(\exists y \in \mathbb{Z}) : (y + y = x) \vee (y + y + 1 = x)$$
   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

$$O(m^{2d^2} d^{4d^3} L^{4d^3} \mathrm{LP}(d, m^d d^4 (\log d + \log L))) \text{ bit operations,}$$

where $\mathrm{LP}(d, H)$ 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.

Decomposing the integer points of a polyhedron

Example

Input: $K_1 : \begin{cases} 3x_1 - 2x_2 + x_3 \leq 7 \\ -2x_1 + 2x_2 - x_3 \leq 12 \\ -4x_1 + x_2 + 3x_3 \leq 15 \\ -x_2 \leq -25 \end{cases}$, 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:

$$\begin{cases} 3x_1 - 2x_2 + x_3 \leq 7 \\ -2x_1 + 2x_2 - x_3 \leq 12 \\ -4x_1 + x_2 + 3x_3 \leq 15 \\ 2x_2 - x_3 \leq 48 \\ -5x_2 + 13x_3 \leq 67 \\ -x_2 \leq -25 \\ 2 \leq x_3 \leq 17 \end{cases}, \begin{cases} x_1 = 15 \\ x_2 = 27 \\ x_3 = 16 \end{cases}, \begin{cases} x_1 = 18 \\ x_2 = 33 \\ x_3 = 18 \end{cases}, \begin{cases} x_1 = 14 \\ x_2 = 25 \\ x_3 = 15 \end{cases}, \begin{cases} x_1 = 19 \\ x_2 = 50 + t \\ x_3 = 50 + 2t \\ -25 \leq t \leq -16. \end{cases}$$

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:

$$\begin{cases} 3x_1 - 2x_2 + x_3 \leq 7 \\ -2x_1 + 2x_2 - x_3 \leq 12 \\ -4x_1 + x_2 + 3x_3 \leq 15 \\ 2x_2 - x_3 \leq 48 \\ -5x_2 + 13x_3 \leq 67 \\ -x_2 \leq -25 \\ 2 \leq x_3 \leq 17 \end{cases}, \quad \begin{cases} x_1 = 15 \\ x_2 = 27 \\ x_3 = 16 \end{cases}, \quad \begin{cases} x_1 = 18 \\ x_2 = 33 \\ x_3 = 18 \end{cases}, \quad \begin{cases} x_1 = 14 \\ x_2 = 25 \\ x_3 = 15 \end{cases}, \quad \begin{cases} x_1 = 19 \\ x_2 = 50 + t \\ x_3 = 50 + 2t \\ -25 \leq t \leq -16. \end{cases}$$

- 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

Algorithm

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

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

Algorithm-IntegerNormalize

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

1. Solve integer solutions for (implicit) equations:
   - Tools: Hermite normal form;
   - Return $\mathbf{x} = \mathbf{Pt} + \mathbf{q}$, where $\mathbf{t}$ is a new unknown vector with less length than $\mathbf{x}$.
2. Substitute $\mathbf{x} = \mathbf{Pt} + \mathbf{q}$ into $\mathbf{Ax} \leq \mathbf{b}$ and remove redundant inequalities:
   - $\mathbf{cx} \leq d$ is implied by $\mathbf{Ax} \leq \mathbf{b} \iff \sup\{-(\mathbf{cx} - d) | \mathbf{Ax} \leq \mathbf{b}\} = 0$;
   - Return $\mathbf{Mt} \leq \mathbf{v}$.

Algorithm-IntegerNormalize

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

1. Solve integer solutions for (implicit) equations:
   - Tools: Hermite normal form;
   - Return $\mathbf{x} = \mathbf{Pt} + \mathbf{q}$, where $\mathbf{t}$ is a new unknown vector with less length than $\mathbf{x}$.
2. Substitute $\mathbf{x} = \mathbf{Pt} + \mathbf{q}$ into $\mathbf{Ax} \leq \mathbf{b}$ and remove redundant inequalities:
   - $\mathbf{cx} \leq d$ is implied by $\mathbf{Ax} \leq \mathbf{b} \iff \sup\{-(\mathbf{cx} - d) | \mathbf{Ax} \leq \mathbf{b}\} = 0$;
   - Return $\mathbf{Mt} \leq \mathbf{v}$.

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

$$\begin{cases} x = -3t_1 + 2t_2 - 3t_3 + 2 \\ y = 2t_1 + t_3 - 1 \\ z = t_2 \\ w = t_3 \end{cases} \quad \text{and} \quad \begin{cases} 3t_1 - 2t_2 + t_3 \leq 7 \\ -2t_1 + 2t_2 - t_3 \leq 12 \\ -4t_1 + t_2 + 3t_3 \leq 15 \\ -t_2 \leq -25 \end{cases}.$$

Algorithm-DarkShadow

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

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

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

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

Algorithm-DarkShadow

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

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

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

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

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

$$\begin{cases} 2t_2 - t_3 \leq 48 \\ -5t_2 + 13t_3 \leq 67 \\ -t_2 \leq -25 \end{cases}$$

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($\mathbf{Mt} \leq \mathbf{v}$).
dark shadow $D := R \cap d_1 \cap \cdots \cap d_r$
grey shadow: $G := R \setminus D$

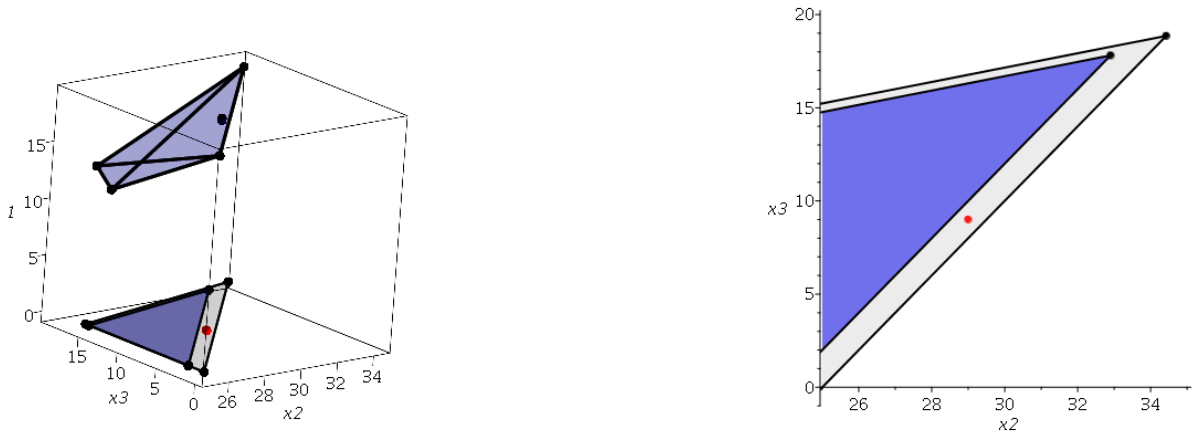


Figure: The real, the dark and the grey shadows of a polyhedron.

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

Algorithm-Greyshadow

Third procedure–GreyShadow($\mathbf{Mt} \leq \mathbf{v}$)
Disjoint decomposition: $R \setminus D = \biguplus_{1 \leq i \leq t} G_i$, 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 $l_1 : \ a_1 t_1 + \mathbf{a}'\mathbf{t}' \leq v_1$ with $a_1 > 0$ and any lower bound $l_2 : \ b_1 t_1 + \mathbf{b}'\mathbf{t}' \leq v_2$ 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 $\{b_1 t_1 + \mathbf{b}'\mathbf{t}' = v_2 - i\} \cap \mathbf{Mt} \leq \mathbf{v} \cap$ $\{-b_1 \mathbf{a}'\mathbf{t}' + a_1 \mathbf{b}'\mathbf{t}' > -b_1 v_1 + a_1 v_2 - (a_1 - 1)(-b_1 - 1)\}$,
3. replace $\mathbf{Mt} \leq \mathbf{v}$ by

$$\mathbf{Mt} \leq \mathbf{v} \cap \{-b_1 \mathbf{a}'\mathbf{t}' + a_1 \mathbf{b}'\mathbf{t}' \leq -b_1 v_1 + a_1 v_2 - (a_1 - 1)(-b_1 - 1)\}.$$

Algorithm-Greyshadow

Returning to our example, combining system $\mathbf{Mt} \leq \mathbf{v}$ with the negation of $2t_2 - t_3 \leq 48$ from Θ, yields

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

$$\text{IntegerNormalize} \Big| \text{new variables } t_4, t_5 \downarrow$$

$$\begin{cases} t_1 = t_4 \\ t_2 = t_5 + 1 \\ t_3 = -2t_4 + 2t_5 + 1 \end{cases}, \quad \text{and} \quad \begin{cases} t_4 \leq 8 \\ -10t_4 + 7t_5 \leq 11 \\ -t_5 \leq -24 \\ -2t_4 - t_5 \leq -48 \end{cases},$$

Algorithm-Illustration

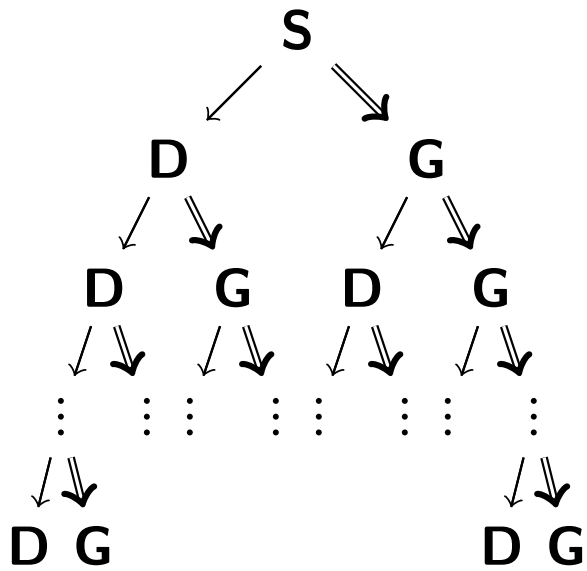


Figure: IntegerSolve

Algorithm

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

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

Plan

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

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

Notation

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

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 $O(d^2\, m^{2d}\, \mathrm{LP}(d, 2^d h d^2 m^d))$ bit operations.

Hypothesis

During the execution of the function call 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 IntegerSolve(K) *runs within $O(m^{2d^2} d^{4d^3} L^{4d^3} \mathrm{LP}(d, m^d d^4 (\log d + \log L)))$ bit operations.*

Plan

Experiments

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

Example	m	d	L	m_o	L_o	?Hyp	t_H	t_P
Tetrahedron	4	3	1	1	1	yes	0.695	0.697
TruncatedTetrahedron	8	3	1	1	1	yes	1.461	1.468
Presburger 4	3	4	5	2	12	yes	0.706	0.871
Presburger 6	4	5	89	6	35	yes	0.893	0.746
Bounded 7	8	3	19	3	190	no	138.448	239.637
Bounded 8	4	3	25	5	67	yes	6.462	3.821
Bounded 9	6	3	18	6	74	no	23.574	16.763
Unbounded 2	3	4	10	61	2255	no	0.547	0.600
Unbounded 5	5	4	8	1	8	no	1.321	1.319
Unbounded 6	10	4	8	1	8	no	1.494	1.479
P91	12	3	96	5	96	no	19.318	15.458
Sys_1	6	3	15	2	67	yes	2.413	1.915
Sys_3	8	3	1	1	1	yes	1.481	1.479
Automatic	8	2	999	1	999	yes	0.552	0.549
Automatic2	6	4	1	1	2	yes	1.115	1.113

Table: Implementation

Plan

Application

Solve integer programming:

$$\begin{gathered}\min_{\text{lex}}(x_1, \ldots, x_d) \\ \mathbf{Ax} \leq \mathbf{b}, \\ \mathbf{x} \in \mathbb{Z}^d\end{gathered}$$

Example

Problem:

$$\begin{gathered}\min_{\text{lex}}(x_3, x_2, x_1) \\ 3x_1 - 2x_2 + x_3 \leq 7 \\ -2x_1 + 2x_2 - x_3 \leq 12 \\ -4x_1 + x_2 + 3x_3 \leq 15 \\ -x_2 \leq -25 \\ x_1, x_2, x_3 \in \mathbb{Z}\end{gathered}$$

Application

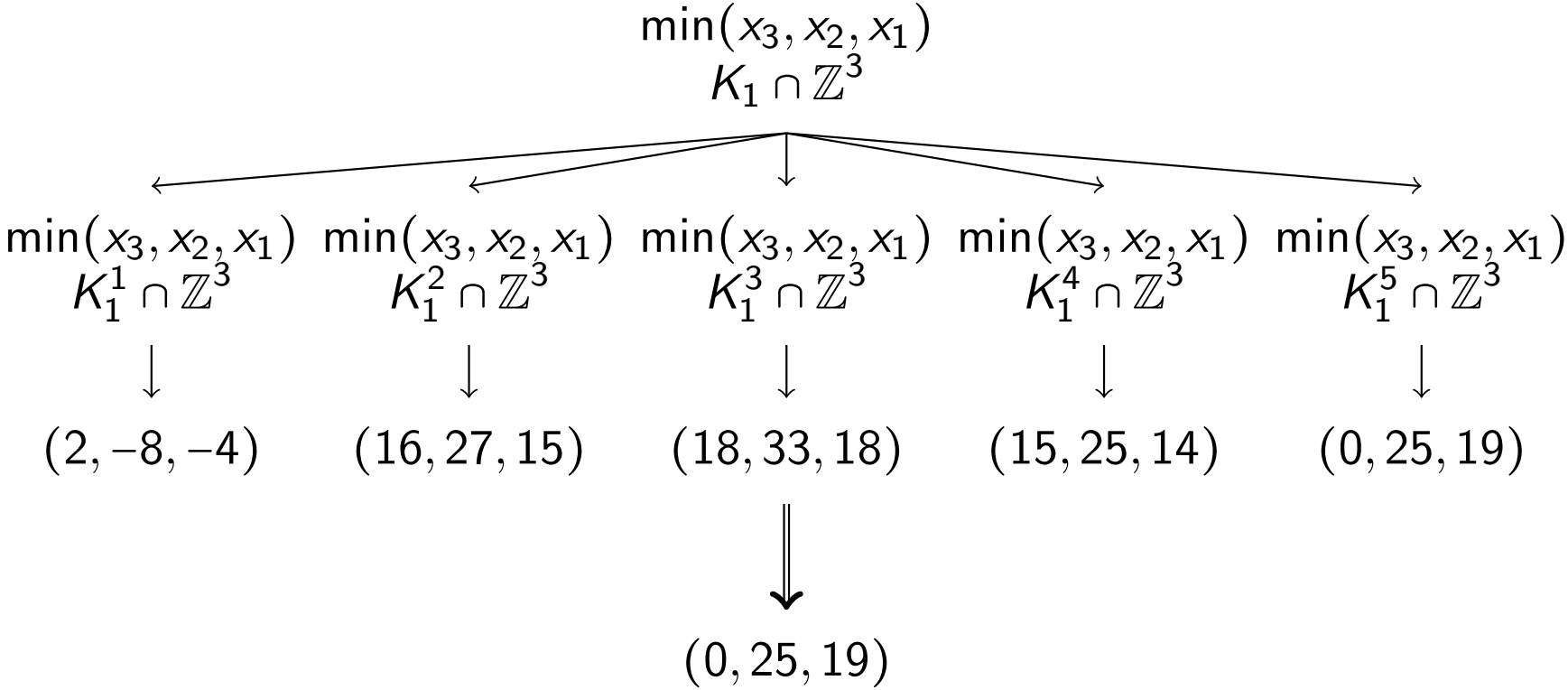
Example

Input: $K_1 : \begin{cases} 3x_1 - 2x_2 + x_3 \leq 7 \\ -2x_1 + 2x_2 - x_3 \leq 12 \\ -4x_1 + x_2 + 3x_3 \leq 15 \\ -x_2 \leq -25 \end{cases}$, 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:

$$\begin{cases} 3x_1 - 2x_2 + x_3 \leq 7 \\ -2x_1 + 2x_2 - x_3 \leq 12 \\ -4x_1 + x_2 + 3x_3 \leq 15 \\ 2x_2 - x_3 \leq 48 \\ -5x_2 + 13x_3 \leq 67 \\ -x_2 \leq -25 \\ 2 \leq x_3 \leq 17 \end{cases}, \begin{cases} x_1 = 15 \\ x_2 = 27 \\ x_3 = 16 \end{cases}, \begin{cases} x_1 = 18 \\ x_2 = 33 \\ x_3 = 18 \end{cases}, \begin{cases} x_1 = 14 \\ x_2 = 25 \\ x_3 = 15 \end{cases}, \begin{cases} x_1 = 19 \\ x_2 = 50 + t \\ x_3 = 50 + 2t \\ -25 \leq t \leq -16. \end{cases}$$

Application



Plan

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.