Counting problems with parametric polyhedral Sets

Rui-Juan Jing[1] Yuzhuo Lei[2] Christopher Maligec
Marc Moreno Maza[2]

[1]School of Mathematical Sciences, Jiangsu University

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

CASC 2024
September 2, 2024

Plan

Plan

Counting memory accesses in a for-loop nest (1/2)

Consider the well-known example SOR (Successive-Over Relaxation) from the numerical solving of PDEs (Partial differential Equations).

```
for (i=2, i<N, i++)
    for (j=2, j <N, j++)
        a[i][j] = (2*a[i][j] + a[i-1][j] + a[i+1][j] +
                   a[i][j-1] + a[i][j+1])/6;
```

Counting memory accesses in a for-loop nest (1/2)

Consider the well-known example SOR (Successive-Over Relaxation) from the numerical solving of PDEs (Partial differential Equations).

```
for (i=2, i<N, i++)
    for (j=2, j <N, j++)
        a[i][j] = (2*a[i][j] + a[i-1][j] + a[i+1][j] +
                   a[i][j-1] + a[i][j+1])/6;
```

- The memory slots accessed by the for-loop nest are given by:

$$\{(i + \Delta i, j + \Delta j) \quad | \quad -1 \leq \Delta i - \Delta j, \Delta i + \Delta j, \leq 1, 2 \leq i, j \leq N - 1\}$$

Counting memory accesses in a for-loop nest (1/2)

Consider the well-known example SOR (Successive-Over Relaxation) from the numerical solving of PDEs (Partial differential Equations).

```
for (i=2, i<N, i++)
    for (j=2, j <N, j++)
        a[i][j] = (2*a[i][j] + a[i-1][j] + a[i+1][j] +
                   a[i][j-1] + a[i][j+1])/6;
```

- The memory slots accessed by the for-loop nest are given by:

$$\{(i + \Delta i, j + \Delta j) \mid -1 \leq \Delta i - \Delta j, \Delta i + \Delta j, \leq 1, 2 \leq i, j \leq N - 1\}$$

- Using standard techniques from Linear Algebra, namely Fourier-Motzkin elimination (FME), we can rewrite the above set as:

$$\left\{ (x, y) \mid \left\{ \begin{array}{l} 1 \leq x, y \leq N \\ 3 \leq x + y \leq 2N - 1 \\ 2 - N \leq x - y \leq N - 2 \end{array} \right. \right\}, \text{ for } N \geq 3.$$

Counting memory accesses in a for-loop nest (1/2)

Consider the well-known example SOR (Successive-Over Relaxation) from the numerical solving of PDEs (Partial differential Equations).

```
for (i=2, i<N, i++)
    for (j=2, j <N, j++)
        a[i][j] = (2*a[i][j] + a[i-1][j] + a[i+1][j] +
                           a[i][j-1] + a[i][j+1])/6;
```

- The memory slots accessed by the for-loop nest are given by:

  $$\{(i+\Delta i, j+\Delta j) \quad | \quad -1 \leq \Delta i - \Delta j, \Delta i + \Delta j, \leq 1, 2 \leq i, j \leq N-1\}$$

- Using standard techniques from Linear Algebra, namely Fourier-Motzkin elimination (FME), we can rewrite the above set as:

  $$\left\{ (x, y) \quad \middle| \quad \left\{ \begin{array}{l} 1 \leq x, y \leq N \\ 3 \leq x+y \leq 2N-1 \\ 2-N \leq x-y \leq N-2 \end{array} \right. \right\}, \text{ for } N \geq 3.$$

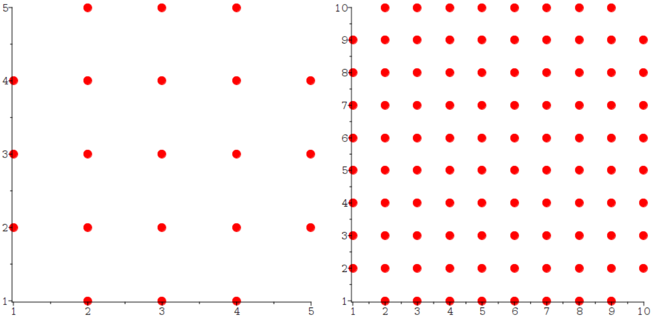
- Hence the problem becomes counting the number of integer points of a parametric polyhedral set P_N.

Counting memory accesses in a for-loop nest (2/2)

```
for (i=2, i<N, i++)
    for (j=2, j <N, j++)
        a[i][j] = (2*a[i][j] + a[i-1][j] + a[i+1][j] +
                           a[i][j-1] + a[i][j+1])/6;
```

Counting memory accesses in a for-loop nest (2/2)

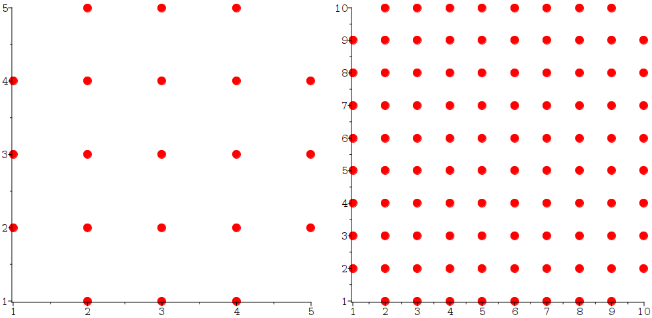
```
for (i=2, i<N, i++)
    for (j=2, j <N, j++)
        a[i][j] = (2*a[i][j] + a[i-1][j] + a[i+1][j] +
                   a[i][j-1] + a[i][j+1])/6;
```



The integer points of the parametric polyhedron P_N for $N = 5$ and $N = 10$.

Counting memory accesses in a for-loop nest (2/2)

```
for (i=2, i<N, i++)
    for (j=2, j <N, j++)
        a[i][j] = (2*a[i][j] + a[i-1][j] + a[i+1][j] +
                   a[i][j-1] + a[i][j+1])/6;
```



The integer points of the parametric polyhedron P_N for $N = 5$ and $N = 10$.
We will see later that $\mid P \cap \mathbb{Z}^2 \mid \; = \; N^2 - 4$.

Objective and challenges (1/2)

- Parametric polyhedra are used in various applications (optimization of computer programs, combinatorial optimization).

Objective and challenges (1/2)

- Parametric polyhedra are used in various applications (optimization of computer programs, combinatorial optimization).
- A *polyhedron* P is the solution set of a system of linear inequalities:

$$\mathbf{A}\vec{x} \leq \vec{b},$$

  where:

  1. $\mathbf{A}$ is an $m \times d$ matrix of rational numbers,
  2. $\vec{x}$ is a column vector of n unknowns $x_1, \ldots, x_d$ and
  3. $\vec{b}$ is a column vector of n coefficients $b_1, \ldots, b_m$.

Objective and challenges (1/2)

- Parametric polyhedra are used in various applications (optimization of computer programs, combinatorial optimization).
- A *polyhedron* P is the solution set of a system of linear inequalities:

$$\mathbf{A}\vec{x} \leq \vec{b},$$

  where:

  1. $\mathbf{A}$ is an $m \times d$ matrix of rational numbers,
  2. $\vec{x}$ is a column vector of n unknowns $x_1, \ldots, x_d$ and
  3. $\vec{b}$ is a column vector of n coefficients $b_1, \ldots, b_m$.

- If all $b_1, \ldots, b_m$ are rational numbers, then P is non-parametric.

Objective and challenges (1/2)

- Parametric polyhedra are used in various applications (optimization of computer programs, combinatorial optimization).
- A *polyhedron* P is the solution set of a system of linear inequalities:

$$\mathbf{A}\vec{x} \leq \vec{b},$$

where:

1. $\mathbf{A}$ is an $m \times d$ matrix of rational numbers,
2. $\vec{x}$ is a column vector of n unknowns $x_1, \ldots, x_d$ and
3. $\vec{b}$ is a column vector of n coefficients $b_1, \ldots, b_m$.

- If all $b_1, \ldots, b_m$ are rational numbers, then P is non-parametric.
- If at least one b_i is an affine expression in some parameters (e.g. $N - 1$) then P is said parametric.

Objective and challenges (1/2)

- Parametric polyhedra are used in various applications (optimization of computer programs, combinatorial optimization).
- A *polyhedron* P is the solution set of a system of linear inequalities:

$$\mathbf{A}\vec{x} \leq \vec{b},$$

  where:

  1. $\mathbf{A}$ is an $m \times d$ matrix of rational numbers,
  2. $\vec{x}$ is a column vector of n unknowns $x_1, \ldots, x_d$ and
  3. $\vec{b}$ is a column vector of n coefficients $b_1, \ldots, b_m$.

- If all $b_1, \ldots, b_m$ are rational numbers, then P is non-parametric.
- If at least one b_i is an affine expression in some parameters (e.g. $N - 1$) then P is said parametric.
- For simplicity, in the sequel, we will see the vector $\vec{b}$ as the parameter of a parametric polyhedron $P(\vec{b})$.

Objective and challenges (1/2)

- Parametric polyhedra are used in various applications (optimization of computer programs, combinatorial optimization).
- A *polyhedron* P is the solution set of a system of linear inequalities:

$$\mathbf{A}\vec{x} \leq \vec{b},$$

  where:

  1. $\mathbf{A}$ is an $m \times d$ matrix of rational numbers,
  2. $\vec{x}$ is a column vector of n unknowns $x_1, \ldots, x_d$ and
  3. $\vec{b}$ is a column vector of n coefficients $b_1, \ldots, b_m$.

- If all $b_1, \ldots, b_m$ are rational numbers, then P is non-parametric.
- If at least one b_i is an affine expression in some parameters (e.g. $N - 1$) then P is said parametric.
- For simplicity, in the sequel, we will see the vector $\vec{b}$ as the parameter of a parametric polyhedron $P(\vec{b})$.

Objective

Our goal is, given a parametric polyhedron $P(\vec{b})$, to count the number of its integer points as a function $c(\vec{b})$ of the parameter $\vec{b}$.

Objective and challenges (2/2)

- One challenge is that the shape (vertices, facets, etc.) of the integer hull of $P(\vec{b})$, that is, $P(\vec{b}) \cap \mathbb{Z}^d$, may vary with the values of $\vec{b}$.

Objective and challenges (2/2)

- One challenge is that the shape (vertices, facets, etc.) of the integer hull of $P(\vec{b})$, that is, $P(\vec{b}) \cap \mathbb{Z}^d$, may vary with the values of $\vec{b}$.
- Consider the parametric polyhedron P_N given by:

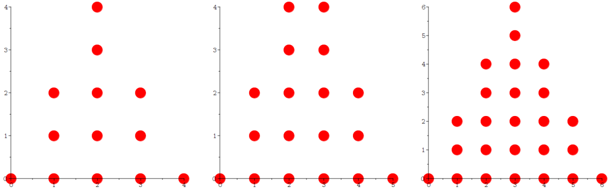
$$\begin{cases} 0 \leq i, j \\ j \leq 2i \\ 2i + j \leq N \end{cases}$$

Objective and challenges (2/2)

- One challenge is that the shape (vertices, facets, etc.) of the integer hull of $P(\vec{b})$, that is, $P(\vec{b}) \cap \mathbb{Z}^d$, may vary with the values of $\vec{b}$.
- Consider the parametric polyhedron P_N given by:

$$\begin{cases} 0 \le i, j \\ j \le 2i \\ 2i + j \le N \end{cases}$$

- The plots below show P_N for $N = 8, 10, 12$.



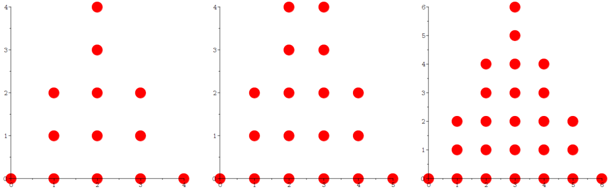
- Fortunately, Ehrhart Theory tells us that these variations are periodic

Objective and challenges (2/2)

- One challenge is that the shape (vertices, facets, etc.) of the integer hull of $P(\vec{b})$, that is, $P(\vec{b}) \cap \mathbb{Z}^d$, may vary with the values of $\vec{b}$.
- Consider the parametric polyhedron P_N given by:

$$\begin{cases} 0 \leq i, j \\ j \leq 2i \\ 2i + j \leq N \end{cases}$$

- The plots below show P_N for $N = 8, 10, 12$.



- Fortunately, Ehrhart Theory tells us that these variations are periodic
- Hence, the function $c(\vec{b})$ is computable as a piece-wise function.

Plan

Related works

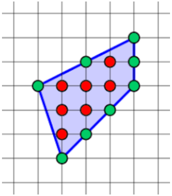
- Given a 2D polyhedral set (= polytope) P, whose vertices are integer points, Pick's theorem relates the area A of P, the number b of integer points on the border of P, and the number i in the interior of P:

$$A = i + \frac{b}{2} - 1$$

Related works

- Given a 2D polyhedral set (= polytope) P, whose vertices are integer points, Pick's theorem relates the area A of P, the number b of integer points on the border of P, and the number i in the interior of P:

$$A = i + \frac{b}{2} - 1$$

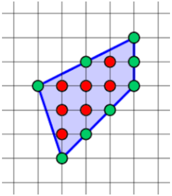


Related works

- Given a 2D polyhedral set (= polytope) P, whose vertices are integer points, Pick's theorem relates the area A of P, the number b of integer points on the border of P, and the number i in the interior of P:

$$A = i + \frac{b}{2} - 1$$

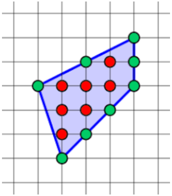
- No generalization of Pick's theorem to higher dimension.



Related works

- Given a 2D polyhedral set (= polytope) P, whose vertices are integer points, Pick's theorem relates the area A of P, the number b of integer points on the border of P, and the number i in the interior of P:

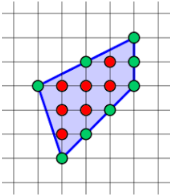
$$A = i + \frac{b}{2} - 1$$



- No generalization of Pick's theorem to higher dimension.
- By studying the dilation of polyhedral sets, Eugène Ehrhart discovered and studied the *periodic behaviour* of parametric polyhedral sets.

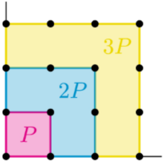
Related works

- Given a 2D polyhedral set (= polytope) P, whose vertices are integer points, Pick's theorem relates the area A of P, the number b of integer points on the border of P, and the number i in the interior of P:

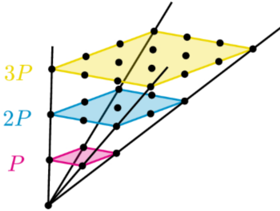


$$A = i + \frac{b}{2} - 1$$

- No generalization of Pick's theorem to higher dimension.
- By studying the dilation of polyhedral sets, Eugène Ehrhart discovered and studied the *periodic behaviour* of parametric polyhedral sets.



(a)

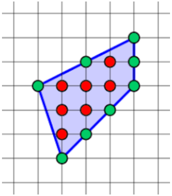


(b)

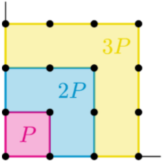
Related works

- Given a 2D polyhedral set (= polytope) P, whose vertices are integer points, Pick's theorem relates the area A of P, the number b of integer points on the border of P, and the number i in the interior of P:

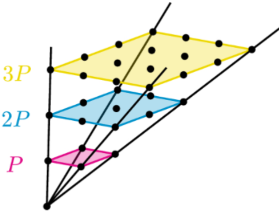
$$A = i + \frac{b}{2} - 1$$



- No generalization of Pick's theorem to higher dimension.
- By studying the dilation of polyhedral sets, Eugène Ehrhart discovered and studied the *periodic behaviour* of parametric polyhedral sets.
- See Ehrhart polynomial.
- Images are from Wikipedia (fair use category).



(a)



(b)

Plan

Generating function of a polyhedral set (1/4)

- Consider a polyhedral set $P \subseteq \mathbb{Q}^d$.

Generating function of a polyhedral set (1/4)

- Consider a polyhedral set $P \subseteq \mathbb{Q}^d$.
- Each integer point $\mathbf{e} = (e_1, \ldots, e_d)$ of P is mapped to the monomial $\mathbf{x}^{\mathbf{e}} = {x_1}^{e_1} \cdots {x_d}^{e_d}$

Generating function of a polyhedral set (1/4)

- Consider a polyhedral set $P \subseteq \mathbb{Q}^d$.
- Each integer point $\mathbf{e} = (e_1, \ldots, e_d)$ of P is mapped to the monomial $\mathbf{x}^{\mathbf{e}} = {x_1}^{e_1} \cdots {x_d}^{e_d}$
- When $d = 2$, we write (x, y) instead of (x_1, x_2).

Generating function of a polyhedral set (1/4)

- Consider a polyhedral set $P \subseteq \mathbb{Q}^d$.
- Each integer point $\mathbf{e} = (e_1, \ldots, e_d)$ of P is mapped to the monomial $\mathbf{x}^{\mathbf{e}} = {x_1}^{e_1} \cdots {x_d}^{e_d}$
- When $d = 2$, we write (x, y) instead of (x_1, x_2).

Definition

The *generating function* of P is the formal power series:

$$G(P, \mathbf{x}) = \sum_{\mathbf{e} \in P \cap \mathbb{Z}^d} \mathbf{x}^{\mathbf{e}}.$$

Generating function of a polyhedral set (1/4)

- Consider a polyhedral set $P \subseteq \mathbb{Q}^d$.
- Each integer point $\mathbf{e} = (e_1, \ldots, e_d)$ of P is mapped to the monomial $\mathbf{x}^{\mathbf{e}} = {x_1}^{e_1} \cdots {x_d}^{e_d}$
- When $d = 2$, we write (x, y) instead of (x_1, x_2).

Definition

The *generating function* of P is the formal power series:

$$G(P, \mathbf{x}) = \sum_{\mathbf{e} \in P \cap \mathbb{Z}^d} \mathbf{x}^{\mathbf{e}}.$$

- If P is bounded, then $G(P, (1, \ldots, 1))$ counts the number of its integer points.

Generating function of a polyhedral set (1/4)

- Consider a polyhedral set $P \subseteq \mathbb{Q}^d$.
- Each integer point $\mathbf{e} = (e_1, \ldots, e_d)$ of P is mapped to the monomial $\mathbf{x}^{\mathbf{e}} = {x_1}^{e_1} \cdots {x_d}^{e_d}$
- When $d = 2$, we write (x, y) instead of (x_1, x_2).

Definition

The *generating function* of P is the formal power series:

$$G(P, \mathbf{x}) = \sum_{\mathbf{e} \in P \cap \mathbb{Z}^d} \mathbf{x}^{\mathbf{e}}.$$

- If P is bounded, then $G(P, (1, \ldots, 1))$ counts the number of its integer points.
- If P is not bounded, then $G(P, \mathbf{x})$ is a formal power series and can still be manipulated algorithmically.

Generating function of a polyhedral set (1/4)

- Consider a polyhedral set $P \subseteq \mathbb{Q}^d$.
- Each integer point $\mathbf{e} = (e_1, \ldots, e_d)$ of P is mapped to the monomial $\mathbf{x}^{\mathbf{e}} = {x_1}^{e_1} \cdots {x_d}^{e_d}$
- When $d = 2$, we write (x, y) instead of (x_1, x_2).

Definition

The *generating function* of P is the formal power series:

$$G(P, \mathbf{x}) = \sum_{\mathbf{e} \in P \cap \mathbb{Z}^d} \mathbf{x}^{\mathbf{e}}.$$

- If P is bounded, then $G(P, (1, \ldots, 1))$ counts the number of its integer points.
- If P is not bounded, then $G(P, \mathbf{x})$ is a formal power series and can still be manipulated algorithmically.
- For $d = 2$, suppose P is the ray corresponding to $y = 0$ and $x \geq 0$, then:

$$G(P, \mathbf{x}) =$$

Generating function of a polyhedral set (1/4)

- Consider a polyhedral set $P \subseteq \mathbb{Q}^d$.
- Each integer point $\mathbf{e} = (e_1, \ldots, e_d)$ of P is mapped to the monomial $\mathbf{x}^{\mathbf{e}} = {x_1}^{e_1} \cdots {x_d}^{e_d}$
- When $d = 2$, we write (x, y) instead of (x_1, x_2).

Definition

The *generating function* of P is the formal power series:

$$G(P, \mathbf{x}) = \sum_{\mathbf{e} \in P \cap \mathbb{Z}^d} \mathbf{x}^{\mathbf{e}}.$$

- If P is bounded, then $G(P, (1, \ldots, 1))$ counts the number of its integer points.
- If P is not bounded, then $G(P, \mathbf{x})$ is a formal power series and can still be manipulated algorithmically.
- For $d = 2$, suppose P is the ray corresponding to $y = 0$ and $x \geq 0$, then:

$$G(P, \mathbf{x}) = \sum_{n=0}^{n=\infty} (x, y)^{(n,0)} =$$

Generating function of a polyhedral set (1/4)

- Consider a polyhedral set $P \subseteq \mathbb{Q}^d$.
- Each integer point $\mathbf{e} = (e_1, \ldots, e_d)$ of P is mapped to the monomial $\mathbf{x}^{\mathbf{e}} = {x_1}^{e_1} \cdots {x_d}^{e_d}$
- When $d = 2$, we write (x, y) instead of (x_1, x_2).

Definition

The *generating function* of P is the formal power series:

$$G(P, \mathbf{x}) = \sum_{\mathbf{e} \in P \cap \mathbb{Z}^d} \mathbf{x}^{\mathbf{e}}.$$

- If P is bounded, then $G(P, (1, \ldots, 1))$ counts the number of its integer points.
- If P is not bounded, then $G(P, \mathbf{x})$ is a formal power series and can still be manipulated algorithmically.
- For $d = 2$, suppose P is the ray corresponding to $y = 0$ and $x \geq 0$, then:

$$G(P, \mathbf{x}) = \sum_{n=0}^{n=\infty} (x, y)^{(n,0)} = \sum_{n=0}^{n=\infty} x^n y^0 =$$

Generating function of a polyhedral set (1/4)

- Consider a polyhedral set $P \subseteq \mathbb{Q}^d$.
- Each integer point $\mathbf{e} = (e_1, \ldots, e_d)$ of P is mapped to the monomial $\mathbf{x}^{\mathbf{e}} = {x_1}^{e_1} \cdots {x_d}^{e_d}$
- When $d = 2$, we write (x, y) instead of (x_1, x_2).

Definition

The *generating function* of P is the formal power series:

$$G(P, \mathbf{x}) = \sum_{\mathbf{e} \in P \cap \mathbb{Z}^d} \mathbf{x}^{\mathbf{e}}.$$

- If P is bounded, then $G(P, (1, \ldots, 1))$ counts the number of its integer points.
- If P is not bounded, then $G(P, \mathbf{x})$ is a formal power series and can still be manipulated algorithmically.
- For $d = 2$, suppose P is the ray corresponding to $y = 0$ and $x \geq 0$, then:

$$G(P, \mathbf{x}) = \sum_{n=0}^{n=\infty} (x, y)^{(n,0)} = \sum_{n=0}^{n=\infty} x^n y^0 = \sum_{n=0}^{n=\infty} x^n =$$

Generating function of a polyhedral set (1/4)

- Consider a polyhedral set $P \subseteq \mathbb{Q}^d$.
- Each integer point $\mathbf{e} = (e_1, \ldots, e_d)$ of P is mapped to the monomial $\mathbf{x}^{\mathbf{e}} = {x_1}^{e_1} \cdots {x_d}^{e_d}$
- When $d = 2$, we write (x, y) instead of (x_1, x_2).

Definition

The *generating function* of P is the formal power series:

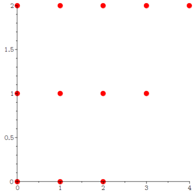
$$G(P, \mathbf{x}) = \sum_{\mathbf{e} \in P \cap \mathbb{Z}^d} \mathbf{x}^{\mathbf{e}}.$$

- If P is bounded, then $G(P, (1, \ldots, 1))$ counts the number of its integer points.
- If P is not bounded, then $G(P, \mathbf{x})$ is a formal power series and can still be manipulated algorithmically.
- For $d = 2$, suppose P is the ray corresponding to $y = 0$ and $x \geq 0$, then:

$$G(P, \mathbf{x}) = \sum_{n=0}^{n=\infty} (x, y)^{(n,0)} = \sum_{n=0}^{n=\infty} x^n y^0 = \sum_{n=0}^{n=\infty} x^n = \frac{1}{1 - x}.$$

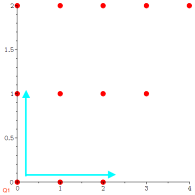
Generating function of a polyhedral set (2/4)

With $d = 2$, we will compute $G(P, \mathbf{x})$ for the polyhedron P given as the convex hull of the 12 points on the figure below.



Generating function of a polyhedral set (2/4)

With $d = 2$, we will compute $G(P, \mathbf{x})$ for the polyhedron P given as the convex hull of the 12 points on the figure below.

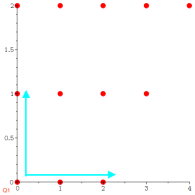


Consider the bottom-left of P, that is, the first quadrant Q_1, that is, the points $(x, , y)$ with $x, y \geq 0$. Then, we have:

$$G(Q_1, \mathbf{x}) =$$

Generating function of a polyhedral set (2/4)

With $d = 2$, we will compute $G(P, \mathbf{x})$ for the polyhedron P given as the convex hull of the 12 points on the figure below.

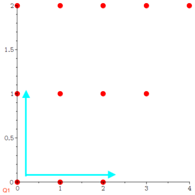


Consider the bottom-left of P, that is, the first quadrant Q_1, that is, the points $(x, , y)$ with $x, y \geq 0$. Then, we have:

$$G(Q_1, \mathbf{x}) = \sum_{m,n \geq 0} x^m y^n =$$

Generating function of a polyhedral set (2/4)

With $d = 2$, we will compute $G(P, \mathbf{x})$ for the polyhedron P given as the convex hull of the 12 points on the figure below.

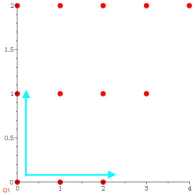


Consider the bottom-left of P, that is, the first quadrant Q_1, that is, the points $(x, , y)$ with $x, y \geq 0$. Then, we have:

$$G(Q_1, \mathbf{x}) = \sum_{m,n \geq 0} x^m y^n = \left(\sum_{m=0}^{n=\infty} x^m \right) \left(\sum_{n=0}^{n=\infty} y^n \right) =$$

Generating function of a polyhedral set (2/4)

With $d = 2$, we will compute $G(P, \mathbf{x})$ for the polyhedron P given as the convex hull of the 12 points on the figure below.

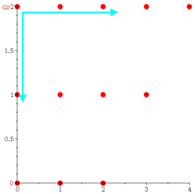


Consider the bottom-left of P, that is, the first quadrant Q_1, that is, the points $(x, , y)$ with $x, y \geq 0$. Then, we have:

$$G(Q_1, \mathbf{x}) = \sum_{m,n \geq 0} x^m y^n = \left(\sum_{m=0}^{n=\infty} x^m\right)\left(\sum_{n=0}^{n=\infty} y^n\right) = \frac{1}{1-x}\frac{1}{1-y}.$$

Generating function of a polyhedral set (2/4)

With $d = 2$, we will compute $G(P, \mathbf{x})$ for the polyhedron P given as the convex hull of the 12 points on the figure below.



Consider the bottom-left of P, that is, the first quadrant Q_1, that is, the points $(x,, y)$ with $x, y \geq 0$. Then, we have:

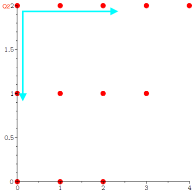
$$G(Q_1, \mathbf{x}) = \sum_{m,n \geq 0} x^m y^n = \left(\sum_{m=0}^{n=\infty} x^m\right)\left(\sum_{n=0}^{n=\infty} y^n\right) = \frac{1}{1-x}\frac{1}{1-y}.$$

Consider the top-left corner of P, that is, the vertex cone Q_2 rooted at $(0, 2)$ and with rays $(0, 1)$ and $(1, 0)$.

$G(Q_2, \mathbf{x}) =$

Generating function of a polyhedral set (2/4)

With $d = 2$, we will compute $G(P, \mathbf{x})$ for the polyhedron P given as the convex hull of the 12 points on the figure below.



Consider the bottom-left of P, that is, the first quadrant Q_1, that is, the points $(x, , y)$ with $x, y \geq 0$. Then, we have:

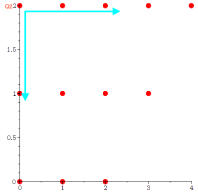
$$G(Q_1, \mathbf{x}) = \sum_{m,n \geq 0} x^m y^n = \left(\sum_{m=0}^{n=\infty} x^m \right) \left(\sum_{n=0}^{n=\infty} y^n \right) = \frac{1}{1-x} \frac{1}{1-y}.$$

Consider the top-left corner of P, that is, the vertex cone Q_2 rooted at $(0, 2)$ and with rays $(0, 1)$ and $(1, 0)$.

$$G(Q_2, \mathbf{x}) = \left(\sum_{m \geq 0} x^m \right) \left(\sum_{n \leq 2} y^n \right) =$$

Generating function of a polyhedral set (2/4)

With $d = 2$, we will compute $G(P, \mathbf{x})$ for the polyhedron P given as the convex hull of the 12 points on the figure below.



Consider the bottom-left of P, that is, the first quadrant Q_1, that is, the points $(x, , y)$ with $x, y \geq 0$. Then, we have:

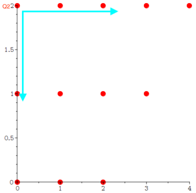
$$G(Q_1, \mathbf{x}) = \sum_{m,n \geq 0} x^m y^n = \left(\sum_{m=0}^{n=\infty} x^m \right) \left(\sum_{n=0}^{n=\infty} y^n \right) = \frac{1}{1-x} \frac{1}{1-y}.$$

Consider the top-left corner of P, that is, the vertex cone Q_2 rooted at $(0, 2)$ and with rays $(0, 1)$ and $(1, 0)$.

$$G(Q_2, \mathbf{x}) = \left(\sum_{m \geq 0} x^m \right) \left(\sum_{n \leq 2} y^n \right) = \left(\sum_{m \geq 0} x^m \right) y^2 \left(\sum_{n \geq 0} (y^{-1})^n \right) =$$

Generating function of a polyhedral set (2/4)

With $d = 2$, we will compute $G(P, \mathbf{x})$ for the polyhedron P given as the convex hull of the 12 points on the figure below.



Consider the bottom-left of P, that is, the first quadrant Q_1, that is, the points $(x, , y)$ with $x, y \geq 0$. Then, we have:

$$G(Q_1, \mathbf{x}) = \sum_{m,n \geq 0} x^m y^n = \left(\sum_{m=0}^{n=\infty} x^m \right) \left(\sum_{n=0}^{n=\infty} y^n \right) = \frac{1}{1-x} \frac{1}{1-y}.$$

Consider the top-left corner of P, that is, the vertex cone Q_2 rooted at $(0, 2)$ and with rays $(0, 1)$ and $(1, 0)$.

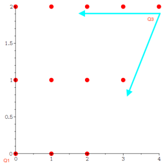
$$G(Q_2, \mathbf{x}) = \left(\sum_{m \geq 0} x^m \right) \left(\sum_{n \leq 2} y^n \right) = \left(\sum_{m \geq 0} x^m \right) y^2 \left(\sum_{n \geq 0} (y^{-1})^n \right) = \frac{1}{1-x} \frac{y^2}{1-y^{-1}}$$

Generating function of a polyhedral set (2/4)

Continuing with the other corners Q_3 and Q_4 of the polytope P

Generating function of a polyhedral set (2/4)

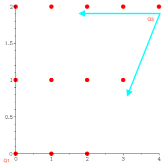
Continuing with the other corners Q_3 and Q_4 of the polytope P



$$G(Q_3, \mathbf{x}) =$$

Generating function of a polyhedral set (2/4)

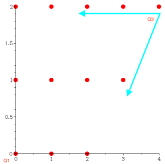
Continuing with the other corners Q_3 and Q_4 of the polytope P



$$G(Q_3, \mathbf{x}) = x^4 y^2 \left(\sum_{n \le m \le 0} x^m y^n \right) =$$

Generating function of a polyhedral set (2/4)

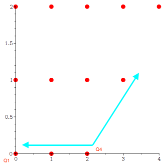
Continuing with the other corners Q_3 and Q_4 of the polytope P



$$G(Q_3, \mathbf{x}) = x^4 y^2 \left(\sum_{n \le m \le 0} x^m y^n \right) = \frac{x^4 y^2}{(1 - x^{-1})(1 - x^{-1} y^{-1})}$$

Generating function of a polyhedral set (2/4)

Continuing with the other corners Q_3 and Q_4 of the polytope P

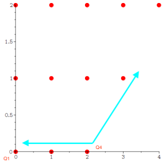


$$G(Q_3, \mathbf{x}) = x^4 y^2 \left(\sum_{n \le m \le 0} x^m y^n \right) = \frac{x^4 y^2}{(1 - x^{-1})(1 - x^{-1} y^{-1})}$$

$$G(Q_4, \mathbf{x}) =$$

Generating function of a polyhedral set (2/4)

Continuing with the other corners Q_3 and Q_4 of the polytope P

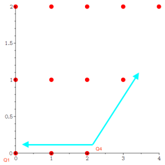


$$G(Q_3, \mathbf{x}) = x^4 y^2 \left(\sum_{n \le m \le 0} x^m y^n \right) = \frac{x^4 y^2}{(1 - x^{-1})(1 - x^{-1} y^{-1})}$$

$$G(Q_4, \mathbf{x}) = x^4 y^0 \left(\sum_{0 \le n, m \le n} x^m y^n \right) =$$

Generating function of a polyhedral set (2/4)

Continuing with the other corners Q_3 and Q_4 of the polytope P

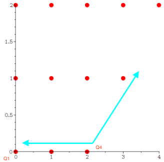


$$G(Q_3, \mathbf{x}) = x^4 y^2 \left(\sum_{n \le m \le 0} x^m y^n \right) = \frac{x^4 y^2}{(1 - x^{-1})(1 - x^{-1} y^{-1})}$$

$$G(Q_4, \mathbf{x}) = x^4 y^0 \left(\sum_{0 \le n, m \le n} x^m y^n \right) = \frac{x^2 y^0}{(1 - xy)(1 - x^{-1})}$$

Generating function of a polyhedral set (2/4)

Continuing with the other corners Q_3 and Q_4 of the polytope P



$$G(Q_3, \mathbf{x}) = x^4 y^2 \left(\sum_{n \le m \le 0} x^m y^n \right) = \frac{x^4 y^2}{(1 - x^{-1})(1 - x^{-1} y^{-1})}$$

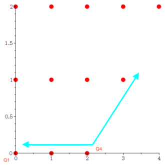
$$G(Q_4, \mathbf{x}) = x^4 y^0 \left(\sum_{0 \le n, m \le n} x^m y^n \right) = \frac{x^2 y^0}{(1 - xy)(1 - x^{-1})}$$

Applying a theorem of Michel Brion (1988) we have:

$$G(P, \mathbf{x}) \quad =$$

Generating function of a polyhedral set (2/4)

Continuing with the other corners Q_3 and Q_4 of the polytope P



$$G(Q_3, \mathbf{x}) = x^4 y^2 \left(\sum_{n \le m \le 0} x^m y^n \right) = \frac{x^4 y^2}{(1 - x^{-1})(1 - x^{-1} y^{-1})}$$

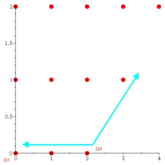
$$G(Q_4, \mathbf{x}) = x^4 y^0 \left(\sum_{0 \le n, m \le n} x^m y^n \right) = \frac{x^2 y^0}{(1 - xy)(1 - x^{-1})}$$

Applying a theorem of Michel Brion (1988) we have:

$$G(P, \mathbf{x}) \quad = \quad G(Q_1, \mathbf{x}) + G(Q_2, \mathbf{x}) + G(Q_3, \mathbf{x}) + G(Q_4, \mathbf{x})$$

Generating function of a polyhedral set (2/4)

Continuing with the other corners Q_3 and Q_4 of the polytope P



$$G(Q_3, \mathbf{x}) = x^4 y^2 \left(\sum_{n \le m \le 0} x^m y^n \right) = \frac{x^4 y^2}{(1 - x^{-1})(1 - x^{-1} y^{-1})}$$

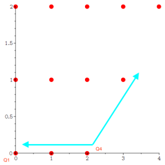
$$G(Q_4, \mathbf{x}) = x^4 y^0 \left(\sum_{0 \le n, m \le n} x^m y^n \right) = \frac{x^2 y^0}{(1 - xy)(1 - x^{-1})}$$

Applying a theorem of Michel Brion (1988) we have:

$$\begin{aligned} G(P, \mathbf{x}) &= G(Q_1, \mathbf{x}) + G(Q_2, \mathbf{x}) + G(Q_3, \mathbf{x}) + G(Q_4, \mathbf{x}) \\ &= \frac{1}{1-x}\frac{1}{1-y} + \frac{1}{1-x}\frac{y^2}{1-y^{-1}} + \frac{x^4 y^2}{(1-x^{-1})(1-x^{-1}y^{-1})} + \frac{x^2 y^0}{(1-xy)(1-x^{-1})} \end{aligned}$$

Generating function of a polyhedral set (2/4)

Continuing with the other corners Q_3 and Q_4 of the polytope P



$$G(Q_3, \mathbf{x}) = x^4 y^2 \left(\sum_{n \le m \le 0} x^m y^n \right) = \frac{x^4 y^2}{(1 - x^{-1})(1 - x^{-1} y^{-1})}$$

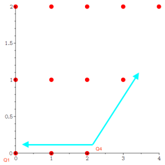
$$G(Q_4, \mathbf{x}) = x^4 y^0 \left(\sum_{0 \le n, m \le n} x^m y^n \right) = \frac{x^2 y^0}{(1 - xy)(1 - x^{-1})}$$

Applying a theorem of Michel Brion (1988) we have:

$$\begin{aligned} G(P, \mathbf{x}) &= G(Q_1, \mathbf{x}) + G(Q_2, \mathbf{x}) + G(Q_3, \mathbf{x}) + G(Q_4, \mathbf{x}) \\ &= \frac{1}{1-x}\frac{1}{1-y} + \frac{1}{1-x}\frac{y^2}{1-y^{-1}} + \frac{x^4 y^2}{(1-x^{-1})(1-x^{-1}y^{-1})} + \frac{x^2 y^0}{(1-xy)(1-x^{-1})} \\ &= y^2 + xy^2 + x^2y^2 + x^3y^2 + x^4y^2 + y + xy + x^2y + x^3y + 1 + x + x^2. \end{aligned}$$

Generating function of a polyhedral set (2/4)

Continuing with the other corners Q_3 and Q_4 of the polytope P



$$G(Q_3, \mathbf{x}) = x^4 y^2 \left(\sum_{n \le m \le 0} x^m y^n \right) = \frac{x^4 y^2}{(1 - x^{-1})(1 - x^{-1} y^{-1})}$$

$$G(Q_4, \mathbf{x}) = x^4 y^0 \left(\sum_{0 \le n, m \le n} x^m y^n \right) = \frac{x^2 y^0}{(1 - xy)(1 - x^{-1})}$$

Applying a theorem of Michel Brion (1988) we have:

$$\begin{array}{rcl} G(P, \mathbf{x}) & = & G(Q_1, \mathbf{x}) + G(Q_2, \mathbf{x}) + G(Q_3, \mathbf{x}) + G(Q_4, \mathbf{x}) \\ & = & \frac{1}{1-x} \frac{1}{1-y} + \frac{1}{1-x} \frac{y^2}{1-y^{-1}} + \frac{x^4 y^2}{(1-x^{-1})(1-x^{-1} y^{-1})} + \frac{x^2 y^0}{(1-xy)(1-x^{-1})} \\ & = & y^2 + xy^2 + x^2 y^2 + x^3 y^2 + x^4 y^2 + y + xy + x^2 y + x^3 y + 1 + x + x^2. \end{array}$$

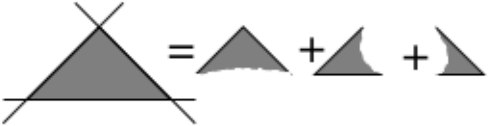
Consequently

$$\begin{array}{rcl} \mid P \cap \mathbb{Z}^2 \mid & = & G(P, (1, 1))) \\ & & = 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 \\ & & = 12. \end{array}$$

Plan

Recall Brion's formula

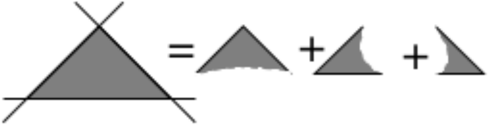
- This formula asserts that for a polytope $P \subseteq \mathbb{Q}^d$ its generating function is the sum of the generating functions of its corners (= vertex cones)



$$G(P, \mathbf{x}) \;=\; G(Q_1, \mathbf{x}) + G(Q_2, \mathbf{x}) + G(Q_3, \mathbf{x})$$

Recall Brion's formula

- This formula asserts that for a polytope $P \subseteq \mathbb{Q}^d$ its generating function is the sum of the generating functions of its corners (= vertex cones)

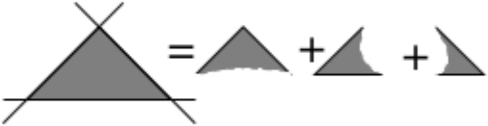


$$G(P, \mathbf{x}) = G(Q_1, \mathbf{x}) + G(Q_2, \mathbf{x}) + G(Q_3, \mathbf{x})$$

- Our previous calculations used two facts

Recall Brion's formula

- This formula asserts that for a polytope $P \subseteq \mathbb{Q}^d$ its generating function is the sum of the generating functions of its corners (= vertex cones)

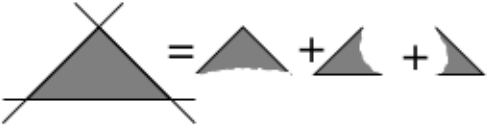


$$G(P, \mathbf{x}) = G(Q_1, \mathbf{x}) + G(Q_2, \mathbf{x}) + G(Q_3, \mathbf{x})$$

- Our previous calculations used two facts
    1. In dimension $d = 2$, every cone is **simplicial** that is, can be generated by d rays,

Recall Brion's formula

- This formula asserts that for a polytope $P \subseteq \mathbb{Q}^d$ its generating function is the sum of the generating functions of its corners (= vertex cones)

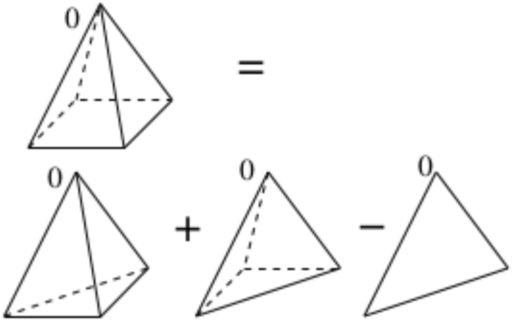


$$G(P, \mathbf{x}) = G(Q_1, \mathbf{x}) + G(Q_2, \mathbf{x}) + G(Q_3, \mathbf{x})$$

- Our previous calculations used two facts
  1. In dimension $d = 2$, every cone is **simplicial** that is, can be generated by d rays,
  2. The cones $Q2, Q_3, Q_4$ are **unimodular**, that is, the sums of the power series $G(Q_2, \mathbf{x})$, $G(Q_3, \mathbf{x})$, $G(Q_4, \mathbf{x})$ can be deduced from that of $G(Q_1, \mathbf{x})$ (the first quadrant) by means of unimodular transformations (that is, mapping integer vectors to integer vectors).

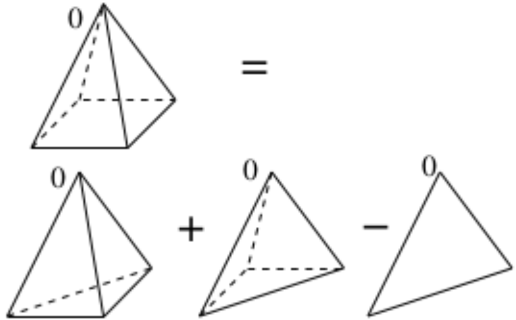
Barvinok's algorithm

- In dimension d, one can decompose any cone into **simplicial** cones ($=$ cones generated by d rays),



Barvinok's algorithm

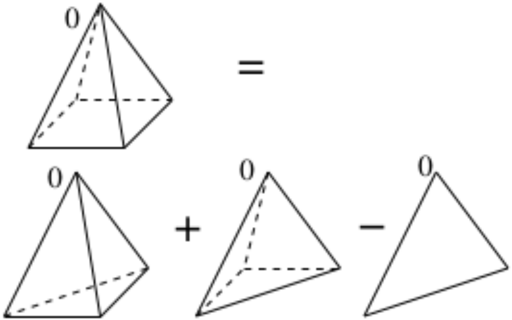
- In dimension d, one can decompose any cone into **simplicial** cones (= cones generated by d rays),



- Alexander Barvinok (1994) proposed an algorithm to decompose any simplicial cones into **unimodular** cones,

Barvinok's algorithm

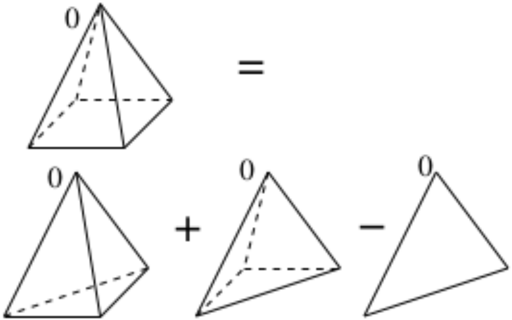
- In dimension d, one can decompose any cone into **simplicial** cones (= cones generated by d rays),



- Alexander Barvinok (1994) proposed an algorithm to decompose any simplicial cones into **unimodular** cones,
- consequently, Barvinok has found the first algorithm to compute $G(P, \mathbf{x})$,

Barvinok's algorithm

- In dimension d, one can decompose any cone into **simplicial** cones ($=$ cones generated by d rays),



- Alexander Barvinok (1994) proposed an algorithm to decompose any simplicial cones into **unimodular** cones,
- consequently, Barvinok has found the first algorithm to compute $G(P, \mathbf{x})$,
- Moreover, Barvinok's algorithm runs in **polynomial time for a fixed d.**

Plan

Sanity-check examples

Example (1)

Input:

$$\{1 \leq i, 1 \leq j, i \leq n, j \leq n\}$$

Output:

$$[[\{n^2\}, [0 \leq n - 1]]]$$

Sanity-check examples

Example (1)

Input:

$$\{1 \leq i, 1 \leq j, i \leq n, j \leq n\}$$

Output:

$$[[\{n^2\}, [0 \leq n-1]]]$$

Example (3)

Input:

$$\{1 \leq i, 1 \leq j, i+j \leq n, 0 \leq n\}$$

Output:

$$[[\{\frac{n^2}{2} - \frac{n}{2}\}, [0 \leq n-2]]]$$

Examples with several parameters

Example (4)

Input:

$$\{1 \leq i, i \leq n, i \leq m, 1 \leq j, j \leq i\}$$

Output:

$$\begin{gathered}[[\{1\},[m-1=0, 0 \leq n-2]],\\ [\{\tfrac{n^2}{2}+\tfrac{n}{2}\},[0 \leq m-n, 0 \leq n-1]],\\ [\{\tfrac{m^2}{2}+\tfrac{m}{2}\},[0 \leq m-2, 0 \leq n-3, 0 \leq -m+n-1]]]\end{gathered}$$

Examples with several parameters

Example (4)

Input:

$$\{1 \leq i, i \leq n, i \leq m, 1 \leq j, j \leq i\}$$

Output:

$$\begin{gathered}[[\{1\},[m-1=0,0 \leq n-2]],\\ [\{\tfrac{n^2}{2}+\tfrac{n}{2}\},[0 \leq m-n,0 \leq n-1]],\\ [\{\tfrac{m^2}{2}+\tfrac{m}{2}\},[0 \leq m-2,0 \leq n-3,0 \leq -m+n-1]]]\end{gathered}$$

Example (5)

Input:

$$\{1 \leq i, i \leq n, i \leq m, 1 \leq j, j \leq p\}$$

Output:

$$\begin{gathered}[[\{pm\},[n-m \geq 1, p-2 \geq 0, m-1 \geq 0]],\\ [\{pn\},[m-n \geq 0, n-2 \geq 0, p-1 \geq 0]],\\ [\{1\},[n-1=0, p-1=0, 0 \leq m-1]],\\ [\{p\},[m-1=0, 0 \leq -2+n, 0 \leq p-1]]]\end{gathered}$$

Examples with quasi-polynomials

Example (6)

Input:

$$\{1 \leq i, j \leq n, 2i \leq 3j\}$$

Output:

$$[[\{Q([n, 2, [\tfrac{3n^2}{4} + \tfrac{n}{2}, -1/4 + \tfrac{3n^2}{4} + \tfrac{n}{2}]])\}, [1 \leq n]]]$$

Examples with quasi-polynomials

Example (6)

Input:

$$\{1 \leq i, j \leq n, 2i \leq 3j\}$$

Output:

$$[[\{Q([n, 2, [\tfrac{3n^2}{4} + \tfrac{n}{2}, -1/4 + \tfrac{3n^2}{4} + \tfrac{n}{2}]])\}, [1 \leq n]]]$$

Example (7)

Input:

$$\{0 \leq i, 0 \leq j, j \leq 2i, 2i + j \leq n\}$$

Output:

$$\begin{gathered}[[\{Q([n, 4, [1 + \tfrac{n}{2} + \tfrac{n^2}{8}, 3/8 + \tfrac{n}{2} + \tfrac{n^2}{8}, 1/2 + \tfrac{n}{2} + \tfrac{n^2}{8}, 3/8 + \tfrac{n}{2} + \tfrac{n^2}{8}]])\}, \\ [0 \leq n - 1]], \\ [\{1\}, [n = 0]]]\end{gathered}$$

Plan

Integer point counting for parametric polyhedra

Given a parametric polyhedron $P(\vec{b})$, the procedures:

1. Vertices($P(\vec{b})$) determines the vertices of $P(\vec{b})$

Integer point counting for parametric polyhedra

Given a parametric polyhedron $P(\vec{b})$, the procedures:

1. Vertices($P(\vec{b})$) determines the vertices of $P(\vec{b})$
   1.1 Yields to solve a (large) number of parametric linear systems, which are **independent** problems

Integer point counting for parametric polyhedra

Given a parametric polyhedron $P(\vec{b})$, the procedures:

1. Vertices($P(\vec{b})$) determines the vertices of $P(\vec{b})$
   1.1 Yields to solve a (large) number of parametric linear systems, which are **independent** problems
   1.2 Their results need to be merged into a **single case discussion**

Integer point counting for parametric polyhedra

Given a parametric polyhedron $P(\vec{b})$, the procedures:

1. Vertices($P(\vec{b})$) determines the vertices of $P(\vec{b})$
   1.1 Yields to solve a (large) number of parametric linear systems, which are **independent** problems
   1.2 Their results need to be merged into a **single case discussion**
2. Cones($P(\vec{b})$) determines the vertex cones ($=$ corners) of $P(\vec{b})$

Integer point counting for parametric polyhedra

Given a parametric polyhedron $P(\vec{b})$, the procedures:

1. Vertices$(P(\vec{b}))$ determines the vertices of $P(\vec{b})$
   1.1 Yields to solve a (large) number of parametric linear systems, which are **independent** problems
   1.2 Their results need to be merged into a **single case discussion**
2. Cones$(P(\vec{b}))$ determines the vertex cones ($=$ corners) of $P(\vec{b})$
   2.1 **Same challenges!**

Integer point counting for parametric polyhedra

Given a parametric polyhedron $P(\vec{b})$, the procedures:

1. Vertices$(P(\vec{b}))$ determines the vertices of $P(\vec{b})$
   1.1 Yields to solve a (large) number of parametric linear systems, which are **independent** problems
   1.2 Their results need to be merged into a **single case discussion**
2. Cones$(P(\vec{b}))$ determines the vertex cones ($=$ corners) of $P(\vec{b})$
   2.1 **Same challenges!**
   2.2 And, at the end, many sets of cases of the case discussion can be replaced by a single case, that is, doing **recombination**.

Integer point counting for parametric polyhedra

Given a parametric polyhedron $P(\vec{b})$, the procedures:

1. Vertices($P(\vec{b})$) determines the vertices of $P(\vec{b})$
   1.1 Yields to solve a (large) number of parametric linear systems, which are **independent** problems
   1.2 Their results need to be merged into a **single case discussion**
2. Cones($P(\vec{b})$) determines the vertex cones (= corners) of $P(\vec{b})$
   2.1 **Same challenges!**
   2.2 And, at the end, many sets of cases of the case discussion can be replaced by a single case, that is, doing **recombination**.
3. GeneratingFunction($P(\vec{b})$) determines the generating functions of each cone Cones($P(\vec{b})$)

Integer point counting for parametric polyhedra

Given a parametric polyhedron $P(\vec{b})$, the procedures:

1. Vertices($P(\vec{b})$) determines the vertices of $P(\vec{b})$
   1.1 Yields to solve a (large) number of parametric linear systems, which are **independent** problems
   1.2 Their results need to be merged into a **single case discussion**
2. Cones($P(\vec{b})$) determines the vertex cones ($=$ corners) of $P(\vec{b})$
   2.1 **Same challenges!**
   2.2 And, at the end, many sets of cases of the case discussion can be replaced by a single case, that is, doing **recombination**.
3. GeneratingFunction($P(\vec{b})$) determines the generating functions of each cone Cones($P(\vec{b})$)
   3.1 since the linear changes of coordinates involve the vertices, the **parameters appear in the exponents** of the generating functions,

Integer point counting for parametric polyhedra

Given a parametric polyhedron $P(\vec{b})$, the procedures:

1. Vertices($P(\vec{b})$) determines the vertices of $P(\vec{b})$
   1.1 Yields to solve a (large) number of parametric linear systems, which are **independent** problems
   1.2 Their results need to be merged into a **single case discussion**
2. Cones($P(\vec{b})$) determines the vertex cones ($=$ corners) of $P(\vec{b})$
   2.1 **Same challenges!**
   2.2 And, at the end, many sets of cases of the case discussion can be replaced by a single case, that is, doing **recombination**.
3. GeneratingFunction($P(\vec{b})$) determines the generating functions of each cone Cones($P(\vec{b})$)
   3.1 since the linear changes of coordinates involve the vertices, the **parameters appear in the exponents** of the generating functions,
   3.2 thanks the periodicity of things, **quasi-polynomials** solve the issue.

Integer point counting for parametric polyhedra

Given a parametric polyhedron $P(\vec{b})$, the procedures:

1. Vertices($P(\vec{b})$) determines the vertices of $P(\vec{b})$
   1.1 Yields to solve a (large) number of parametric linear systems, which are **independent** problems
   1.2 Their results need to be merged into a **single case discussion**
2. Cones($P(\vec{b})$) determines the vertex cones ($=$ corners) of $P(\vec{b})$
   2.1 **Same challenges!**
   2.2 And, at the end, many sets of cases of the case discussion can be replaced by a single case, that is, doing **recombination**.
3. GeneratingFunction($P(\vec{b})$) determines the generating functions of each cone Cones($P(\vec{b})$)
   3.1 since the linear changes of coordinates involve the vertices, the **parameters appear in the exponents** of the generating functions,
   3.2 thanks the periodicity of things, **quasi-polynomials** solve the issue.
4. NumberOfIntegerPoints($P(\vec{b})$)

Integer point counting for parametric polyhedra

Given a parametric polyhedron $P(\vec{b})$, the procedures:

1. Vertices($P(\vec{b})$) determines the vertices of $P(\vec{b})$
   1.1 Yields to solve a (large) number of parametric linear systems, which are **independent** problems
   1.2 Their results need to be merged into a **single case discussion**
2. Cones($P(\vec{b})$) determines the vertex cones (= corners) of $P(\vec{b})$
   2.1 **Same challenges!**
   2.2 And, at the end, many sets of cases of the case discussion can be replaced by a single case, that is, doing **recombination**.
3. GeneratingFunction($P(\vec{b})$) determines the generating functions of each cone Cones($P(\vec{b})$)
   3.1 since the linear changes of coordinates involve the vertices, the **parameters appear in the exponents** of the generating functions,
   3.2 thanks the periodicity of things, **quasi-polynomials** solve the issue.
4. NumberOfIntegerPoints($P(\vec{b})$)
   4.1 Putting everything together requires computing with **multivariate quasi-polynomials.**

Generic case discussion (1/3)

1. Let $\mathcal{A}, \mathcal{B}, \mathcal{V}$ be 3 non-empty sets

Generic case discussion (1/3)

1. Let $\mathcal{A}, \mathcal{B}, \mathcal{V}$ be 3 non-empty sets
2. Let $\mathcal{F}$ be a non-empty set of functions from $\mathcal{A}$ to $\mathcal{B}$.

Generic case discussion (1/3)

1. Let $\mathcal{A}, \mathcal{B}, \mathcal{V}$ be 3 non-empty sets
2. Let $\mathcal{F}$ be a non-empty set of functions from $\mathcal{A}$ to $\mathcal{B}$.
3. Let $\mathcal{P}$ be a non-empty set of predicates on $\mathcal{B}$. closed under negation.

Generic case discussion (1/3)

1. Let $\mathcal{A}, \mathcal{B}, \mathcal{V}$ be 3 non-empty sets
2. Let $\mathcal{F}$ be a non-empty set of functions from $\mathcal{A}$ to $\mathcal{B}$.
3. Let $\mathcal{P}$ be a non-empty set of predicates on $\mathcal{B}$. closed under negation.
4. A **constraint** is any pair $c = (f, p)$ where $f \in \mathcal{F}$ and $p \in \mathcal{P}$ and its zero set is
$$Z(c) = \{a \in \mathcal{A} \mid p(f(a))\} \tag{1}$$
while its negation is $\neg c := (f, \neg p)$.

Generic case discussion (1/3)

1. Let $\mathcal{A}, \mathcal{B}, \mathcal{V}$ be 3 non-empty sets
2. Let $\mathcal{F}$ be a non-empty set of functions from $\mathcal{A}$ to $\mathcal{B}$.
3. Let $\mathcal{P}$ be a non-empty set of predicates on $\mathcal{B}$. closed under negation.
4. A **constraint** is any pair $c = (f, p)$ where $f \in \mathcal{F}$ and $p \in \mathcal{P}$ and its zero set is
$$Z(c) \ = \ \{a \in \mathcal{A} \ | \ p(f(a))\} \tag{1}$$
while its negation is $\neg c := (f, \neg p)$.
5. The constraint $c = (f, p)$ is **consistent** whenever $Z(C) \neq \varnothing$ holds.

Generic case discussion (1/3)

1. Let $\mathcal{A}, \mathcal{B}, \mathcal{V}$ be 3 non-empty sets
2. Let $\mathcal{F}$ be a non-empty set of functions from $\mathcal{A}$ to $\mathcal{B}$.
3. Let $\mathcal{P}$ be a non-empty set of predicates on $\mathcal{B}$. closed under negation.
4. A **constraint** is any pair $c = (f, p)$ where $f \in \mathcal{F}$ and $p \in \mathcal{P}$ and its zero set is
$$Z(c) \;=\; \{a \in \mathcal{A} \;\mid\; p(f(a))\} \tag{1}$$
while its negation is $\neg c := (f, \neg p)$.
5. The constraint $c = (f, p)$ is **consistent** whenever $Z(C) \neq \varnothing$ holds.
6. A **system of constraints** is any finite set C of constraints and its zero set is
$$Z(C) \;=\; \bigcap_{c \in C} Z(c). \tag{2}$$

Generic case discussion (1/3)

1. Let $\mathcal{A}, \mathcal{B}, \mathcal{V}$ be 3 non-empty sets
2. Let $\mathcal{F}$ be a non-empty set of functions from $\mathcal{A}$ to $\mathcal{B}$.
3. Let $\mathcal{P}$ be a non-empty set of predicates on $\mathcal{B}$. closed under negation.
4. A **constraint** is any pair $c = (f, p)$ where $f \in \mathcal{F}$ and $p \in \mathcal{P}$ and its zero set is
$$Z(c) \;=\; \{a \in \mathcal{A} \;\mid\; p(f(a))\} \tag{1}$$
while its negation is $\neg c := (f, \neg p)$.
5. The constraint $c = (f, p)$ is **consistent** whenever $Z(C) \neq \varnothing$ holds.
6. A **system of constraints** is any finite set C of constraints and its zero set is
$$Z(C) \;=\; \bigcap_{c \in C} Z(c). \tag{2}$$
7. A constraint $\gamma \notin C$ is **redundant** w.r.t. C, whenever we have $Z(C \cup \{\gamma\}) \;=\; Z(C)$.

Generic case discussion (1/3)

1. Let $\mathcal{A}, \mathcal{B}, \mathcal{V}$ be 3 non-empty sets
2. Let $\mathcal{F}$ be a non-empty set of functions from $\mathcal{A}$ to $\mathcal{B}$.
3. Let $\mathcal{P}$ be a non-empty set of predicates on $\mathcal{B}$. closed under negation.
4. A **constraint** is any pair $c = (f, p)$ where $f \in \mathcal{F}$ and $p \in \mathcal{P}$ and its zero set is
$$Z(c) \; = \; \{a \in \mathcal{A} \mid p(f(a))\} \tag{1}$$
while its negation is $\neg c := (f, \neg p)$.
5. The constraint $c = (f, p)$ is **consistent** whenever $Z(C) \neq \varnothing$ holds.
6. A **system of constraints** is any finite set C of constraints and its zero set is
$$Z(C) \; = \; \bigcap_{c \in C} Z(c). \tag{2}$$
7. A constraint $\gamma \notin C$ is **redundant** w.r.t. C, whenever we have $Z(C \cup \{\gamma\}) \; = \; Z(C)$.
8. A **value-constraints pair** is any pair (V, C) where $V \subseteq \mathcal{V}$ and C is a system of constraints.

Generic case discussion (2/3)

1. Let $S = (V_1, C_1), \ldots, (V_e, C_e)$ be a sequence of val.-constr. pairs.

Generic case discussion (2/3)

1. Let $S = (V_1, C_1), \ldots, (V_e, C_e)$ be a sequence of val.-constr. pairs.
2. S is **irredundant**, if, for all $1 \leq i, j \leq e$, we have
   $i \neq j \implies Z(C_i) \not\subseteq Z(C_j)$.

Generic case discussion (2/3)

1. Let $S = (V_1, C_1), \ldots, (V_e, C_e)$ be a sequence of val.-constr. pairs.
2. S is **irredundant**, if, for all $1 \leq i, j \leq e$, we have
   $i \neq j \implies Z(C_i) \nsubseteq Z(C_j)$.
3. S is **non-overlapping**, if, for all $1 \leq i < j \leq e$, we have
   $Z(C_i) \cap Z(C_j) = \varnothing$.

Generic case discussion (2/3)

1. Let $S = (V_1, C_1), \ldots, (V_e, C_e)$ be a sequence of val.-constr. pairs.
2. S is **irredundant**, if, for all $1 \leq i, j \leq e$, we have
   $i \neq j \implies Z(C_i) \nsubseteq Z(C_j)$.
3. S is **non-overlapping**, if, for all $1 \leq i < j \leq e$, we have
   $Z(C_i) \cap Z(C_j) = \emptyset$.
4. Let $T = (W_1, D_1), \ldots, (W_f, D_f)$ be a second sequence of value-constraint pairs.
5. We say that T **refines** S whenever the following 3 properties all hold:

Generic case discussion (2/3)

1. Let $S = (V_1, C_1), \ldots, (V_e, C_e)$ be a sequence of val.-constr. pairs.
2. S is **irredundant**, if, for all $1 \leq i, j \leq e$, we have $i \neq j \implies Z(C_i) \nsubseteq Z(C_j)$.
3. S is **non-overlapping**, if, for all $1 \leq i < j \leq e$, we have $Z(C_i) \cap Z(C_j) = \emptyset$.
4. Let $T = (W_1, D_1), \ldots, (W_f, D_f)$ be a second sequence of value-constraint pairs.
5. We say that T **refines** S whenever the following 3 properties all hold:
   5.1 we have: $\bigcup_{i=1}^{e} Z(C_i) = \bigcup_{i=1}^{f} Z(D_i)$,

Generic case discussion (2/3)

1. Let $S = (V_1, C_1), \ldots, (V_e, C_e)$ be a sequence of val.-constr. pairs.
2. S is **irredundant**, if, for all $1 \leq i, j \leq e$, we have
   $i \neq j \implies Z(C_i) \not\subseteq Z(C_j)$.
3. S is **non-overlapping**, if, for all $1 \leq i < j \leq e$, we have
   $Z(C_i) \cap Z(C_j) = \varnothing$.
4. Let $T = (W_1, D_1), \ldots, (W_f, D_f)$ be a second sequence of value-constraint pairs.
5. We say that T **refines** S whenever the following 3 properties all hold:
   5.1 we have: $\bigcup_{i=1}^{e} Z(C_i) \; = \; \bigcup_{i=1}^{f} Z(D_i)$,
   5.2 we have: $\bigcup_{i=1}^{e} V_i \; = \; \bigcup_{i=1}^{f} W_i$,

Generic case discussion (2/3)

1. Let $S = (V_1, C_1), \ldots, (V_e, C_e)$ be a sequence of val.-constr. pairs.
2. S is **irredundant**, if, for all $1 \leq i, j \leq e$, we have
   $i \neq j \implies Z(C_i) \not\subseteq Z(C_j)$.
3. S is **non-overlapping**, if, for all $1 \leq i < j \leq e$, we have
   $Z(C_i) \cap Z(C_j) = \emptyset$.
4. Let $T = (W_1, D_1), \ldots, (W_f, D_f)$ be a second sequence of value-constraint pairs.
5. We say that T **refines** S whenever the following 3 properties all hold:
   5.1 we have: $\bigcup_{i=1}^{e} Z(C_i) = \bigcup_{i=1}^{f} Z(D_i)$,
   5.2 we have: $\bigcup_{i=1}^{e} V_i = \bigcup_{i=1}^{f} W_i$,
   5.3 $(\forall i,\ 1 \leq i \leq f)\ (\exists j, 1 \leq j \leq e) \quad Z(D_i) \subseteq Z(C_j)$ and $V_j \subseteq W_i$.

Generic case discussion (2/3)

1. Let $S = (V_1, C_1), \ldots, (V_e, C_e)$ be a sequence of val.-constr. pairs.
2. S is **irredundant**, if, for all $1 \leq i, j \leq e$, we have $i \neq j \implies Z(C_i) \not\subseteq Z(C_j)$.
3. S is **non-overlapping**, if, for all $1 \leq i < j \leq e$, we have $Z(C_i) \cap Z(C_j) = \emptyset$.
4. Let $T = (W_1, D_1), \ldots, (W_f, D_f)$ be a second sequence of value-constraint pairs.
5. We say that T **refines** S whenever the following 3 properties all hold:
   5.1 we have: $\bigcup_{i=1}^{e} Z(C_i) = \bigcup_{i=1}^{f} Z(D_i)$,
   5.2 we have: $\bigcup_{i=1}^{e} V_i = \bigcup_{i=1}^{f} W_i$,
   5.3 $(\forall i,\ 1 \leq i \leq f)\ (\exists j, 1 \leq j \leq e)\quad Z(D_i) \subseteq Z(C_j)$ and $V_j \subseteq W_i$.
6. We assume that we have a procedure that, for any system of constraints C, decides whether C is consistent or not.

Generic case discussion (2/3)

1. Let $S = (V_1, C_1), \ldots, (V_e, C_e)$ be a sequence of val.-constr. pairs.
2. S is **irredundant**, if, for all $1 \leq i, j \leq e$, we have $i \neq j \implies Z(C_i) \nsubseteq Z(C_j)$.
3. S is **non-overlapping**, if, for all $1 \leq i < j \leq e$, we have $Z(C_i) \cap Z(C_j) = \emptyset$.
4. Let $T = (W_1, D_1), \ldots, (W_f, D_f)$ be a second sequence of value-constraint pairs.
5. We say that T **refines** S whenever the following 3 properties all hold:
   - 5.1 we have: $\bigcup_{i=1}^{e} Z(C_i) \;=\; \bigcup_{i=1}^{f} Z(D_i)$,
   - 5.2 we have: $\bigcup_{i=1}^{e} V_i \;=\; \bigcup_{i=1}^{f} W_i$,
   - 5.3 $(\forall i,\ 1 \leq i \leq f)\ (\exists j, 1 \leq j \leq e) \quad Z(D_i) \subseteq Z(C_j)$ and $V_j \subseteq W_i$.
6. We assume that we have a procedure that, for any system of constraints C, decides whether C is consistent or not.
7. Then, there exists an algorithm that, for the sequence S computes a non-overlapping sequence T refining S.

“Generic” case discussion (3/3)

1. Assume $\mathcal{A} = \mathcal{B} = \mathbb{Z}$ and $\mathcal{P} = \{\leq, \geq, \leq, \geq, =, \neq\}$.

“Generic” case discussion (3/3)

1. Assume $\mathcal{A} = \mathcal{B} = \mathbb{Z}$ and $\mathcal{P} = \{\leq, \geq, \leq, \geq, =, \neq\}$.
2. Because $\mathcal{A} = \mathcal{B} = \mathbb{Z}$, we can normalize systems of constraints to use $\geq$ only.

“Generic” case discussion (3/3)

1. Assume $\mathcal{A} = \mathcal{B} = \mathbb{Z}$ and $\mathcal{P} = \{\leq, \geq, \leq, \geq, =, \neq\}$.
2. Because $\mathcal{A} = \mathcal{B} = \mathbb{Z}$, we can normalize systems of constraints to use $\geq$ only.
3. Consider two systems of constraints C_1 and C_2

“Generic” case discussion (3/3)

1. Assume $\mathcal{A} = \mathcal{B} = \mathbb{Z}$ and $\mathcal{P} = \{\leq, \geq, \leq, \geq, =, \neq\}$.
2. Because $\mathcal{A} = \mathcal{B} = \mathbb{Z}$, we can normalize systems of constraints to use $\geq$ only.
3. Consider two systems of constraints C_1 and C_2
4. For each constraint $\gamma : p(\mathbf{x}) \geq 0$ of C_1

“Generic” case discussion (3/3)

1. Assume $\mathcal{A} = \mathcal{B} = \mathbb{Z}$ and $\mathcal{P} = \{\leq, \geq, \leq, \geq, =, \neq\}$.
2. Because $\mathcal{A} = \mathcal{B} = \mathbb{Z}$, we can normalize systems of constraints to use $\geq$ only.
3. Consider two systems of constraints C_1 and C_2
4. For each constraint $\gamma : p(\mathbf{x}) \geq 0$ of C_1
   4.1 γ is **valid over** C_2 if $p(\mathbf{x}) \geq 0$ for all $x \in Z(C_2)$

"Generic" case discussion (3/3)

1. Assume $\mathcal{A} = \mathcal{B} = \mathbb{Z}$ and $\mathcal{P} = \{\leq, \geq, \leq, \geq, =, \neq\}$.
2. Because $\mathcal{A} = \mathcal{B} = \mathbb{Z}$, we can normalize systems of constraints to use $\geq$ only.
3. Consider two systems of constraints C_1 and C_2
4. For each constraint $\gamma : p(\mathbf{x}) \geq 0$ of C_1
   - 4.1 γ is **valid over** C_2 if $p(\mathbf{x}) \geq 0$ for all $x \in Z(C_2)$
   - 4.2 γ is **separating over** C_2 if $p(\mathbf{x}) \leq -1$ for all $x \in Z(C_2)$

"Generic" case discussion (3/3)

1. Assume $\mathcal{A} = \mathcal{B} = \mathbb{Z}$ and $\mathcal{P} = \{\leq, \geq, \leq, \geq, =, \neq\}$.
2. Because $\mathcal{A} = \mathcal{B} = \mathbb{Z}$, we can normalize systems of constraints to use $\geq$ only.
3. Consider two systems of constraints C_1 and C_2
4. For each constraint $\gamma : p(\mathbf{x}) \geq 0$ of C_1
   4.1 γ is **valid over** C_2 if $p(\mathbf{x}) \geq 0$ for all $x \in Z(C_2)$
   4.2 γ is **separating over** C_2 if $p(\mathbf{x}) \leq -1$ for all $x \in Z(C_2)$
   4.3 γ is **cut over** C_2 if γ neither valid nor separating over C_2.

“Generic” case discussion (3/3)

1. Assume $\mathcal{A} = \mathcal{B} = \mathbb{Z}$ and $\mathcal{P} = \{\leq, \geq, \leq, \geq, =, \neq\}$.
2. Because $\mathcal{A} = \mathcal{B} = \mathbb{Z}$, we can normalize systems of constraints to use $\geq$ only.
3. Consider two systems of constraints C_1 and C_2
4. For each constraint $\gamma : p(\mathbf{x}) \geq 0$ of C_1
   4.1 γ is **valid over** C_2 if $p(\mathbf{x}) \geq 0$ for all $x \in Z(C_2)$
   4.2 γ is **separating over** C_2 if $p(\mathbf{x}) \leq -1$ for all $x \in Z(C_2)$
   4.3 γ is **cut over** C_2 if γ neither valid nor separating over C_2.
   4.4 If for $\gamma : p(\mathbf{x}) \geq 0$ of C_1 we have $p(\mathbf{x}) = -1 - u(\mathbf{x})$ and $u(\mathbf{x}) \geq 0$ is a constraint of C_2, then (p, u) is a pair of **adjacent** inequalities.

“Generic” case discussion (3/3)

1. Assume $\mathcal{A} = \mathcal{B} = \mathbb{Z}$ and $\mathcal{P} = \{\leq, \geq, \leq, \geq, =, \neq\}$.
2. Because $\mathcal{A} = \mathcal{B} = \mathbb{Z}$, we can normalize systems of constraints to use $\geq$ only.
3. Consider two systems of constraints C_1 and C_2
4. For each constraint $\gamma : p(\mathbf{x}) \geq 0$ of C_1
   - 4.1 γ is **valid over** C_2 if $p(\mathbf{x}) \geq 0$ for all $x \in Z(C_2)$
   - 4.2 γ is **separating over** C_2 if $p(\mathbf{x}) \leq -1$ for all $x \in Z(C_2)$
   - 4.3 γ is **cut over** C_2 if γ neither valid nor separating over C_2.
   - 4.4 If for $\gamma : p(\mathbf{x}) \geq 0$ of C_1 we have $p(\mathbf{x}) = -1 - u(\mathbf{x})$ and $u(\mathbf{x}) \geq 0$ is a constraint of C_2, then (p, u) is a pair of **adjacent** inequalities.
5. Theorem: If (p, u) is a pair of adjacent inequalities, and if all other constraints of C_1 (resp. C_2) are valid on C_2 (resp. C_1) then the system of constraints C_3 consisting of all those valid constraints satisfies $Z(C_3) = Z(C_1) \cup Z(C_2)$.

Plan

Concluding remarks

Summary and notes

1. We have presented Brion's formula and Barvinok's algorithm for computing the number of integer points of a polytope.
2. We have discussed our adaptation of those works to the case of parametric polyhedra and its implementation in MAPLE.
3. Another adaptation to this parametric case, tailored to compiler optimization, was led by Sven Verdoolaege and is part of a C library called `barvinok`.

Work in progress

1. Our MAPLE implementation aims at supporting Presburger arithmetic
2. This implementation is designed to extend to parametric polyhedra $\mathbf{A}\vec{x} \leq \vec{b}$ where parameters appear not only in $\vec{b}$ but also in $\mathbf{A}$.
3. Our current work focuses on minimizing the number of cases in the discussion and controlling expression swell.

References

[1] A. Barvinok. *A course in convexity*. Vol. 54. American Mathematical Soc., 2002.

[2] A. Barvinok and J. E. Pommersheim. "An algorithmic theory of lattice points in polyhedra". In: *New perspectives in algebraic combinatorics* 38 (1999), pp. 91–147.

[3] A. I. Barvinok. "A polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed". In: *Mathematics of Operations Research* 19.4 (1994), pp. 769–779.

[4] M. Beck, C. Haase, and F. Sottile. "Formulas of Brion, Lawrence, and Varchenko on rational generating functions for cones". In: *Math. Intelligencer* 31.1 (2009), pp. 9–17.

[5] M. Beck, S. V. Sam, and K. M. Woods. "Maximal periods of Ehrhart quasi-polynomials". In: *Journal of Combinatorial Theory, Series A* 115.3 (2008), pp. 517–525.

[6] A. Bemporad, K. Fukuda, and F. D. Torrisi. "Convexity recognition of the union of polyhedra". In: *Computational Geometry* 18.3 (2001), pp. 141–154. ISSN: 0925-7721.

[7] M. Brion. "Points entiers dans les polyedres convexes". In: *Annales scientifiques de l'École normale supérieure*. Vol. 21. 4. 1988, pp. 653–663.

[8] R. Jing and M. Moreno Maza. "Computing the Integer Points of a Polyhedron, I: Algorithm". In: *CASC 2017, Proceedings*. Vol. 10490. LNCS. Springer, 2017, pp. 225–241.

[9] V. Loechner and D. K. Wilde. "Parameterized polyhedra and their vertices". In: *International Journal of Parallel Programming* 25 (1997), pp. 525–549.

[10] J. A. D. Loera, R. Hemmecke, J. Tauzer, and R. Yoshida. "Effective lattice point counting in rational convex polytopes". In: *J. Symb. Comput.* 38.4 (2004), pp. 1273–1302.

[11] S. Verdoolaege. "Integer set coalescing". In: *International Workshop on Polyhedral Compilation Techniques, Date: 2015/01/19-2015/01/19, Location: Amsterdam, The Netherlands*. 2015.

[12] K. Woods. "The unreasonable ubiquitousness of quasi-polynomials". In: *The Electronic Journal of Combinatorics* 21.1 (2014), P1–44.