A New Algorithm for Computing Integer Hulls of 2D Polyhedral Sets

Chirantan Mukherjee

University of Western Ontario

July 24, 2024

Ontario Research Centre for Computer Algebra

Content

1. Motivation
2. Introduction
3. Existing Algorithm
4. New Algorithm
5. Future Work
6. References

One-dimensional array

```
for(int i = 0; i < n; i++){
  for(int j = i + 1; j < n; j ++)
    A[i * n + j] = A[(n * j - n + j - i - 1];
    }
```

One-dimensional array

```
for(int i = 0; i < n; i++){
  for(int j = i + 1; j < n; j ++)
    A[i * n + j] = A[(n * j - n + j - i - 1];
    }
```

1. Can we parallelize the two `for`-loops?

One-dimensional array

```
for(int i = 0; i < n; i++){
  for(int j = i + 1; j < n; j ++)
    A[i * n + j] = A[(n * j - n + j - i - 1];
    }
```

1. Can we parallelize the two `for-loops`?
2. Is there data dependence between them?

One-dimensional array

```
for(int i = 0; i < n; i++){
  for(int j = i + 1; j < n; j ++)
    A[i * n + j] = A[(n * j - n + j - i - 1];
    }
```

1. Can we parallelize the two `for-loops`?
2. Is there data dependence between them?
3. Are there integer solutions to the system of linear inequalities?

$$\begin{cases} 0 \leq i_1 < n \\ i_1 + 1 \leq j_1 < n \\ 0 \leq i_2 < n \\ i_2 + 1 \leq j_2 < n \\ i_1 \times n + j_1 = n \times j_2 - n + j_2 - i_2 - 1 \end{cases}$$

Delinearize the array accesses

Linearized one-dimensional array

```
for(int i = 0; i < n; i++)
  for(int j = i + 1; j < n; j ++)
    A[i * n + j] =
        A[(n * j - n + j - i - 1];
```

Delinearize the array accesses

Linearized one-dimensional array

```
for(int i = 0; i < n; i++)
  for(int j = i + 1; j < n; j ++)
    A[i * n + j] =
        A[(n * j - n + j - i - 1];
```

Delinearized multi-dimensional array

```
for(int i = 0; i < n; i++)
  for(int j = i + 1; j < n; j ++)
    B[i][j] = B[j - 1][j - i - 1];
```

Delinearize the array accesses

Linearized one-dimensional array

```
for(int i = 0; i < n; i++)
  for(int j = i + 1; j < n; j ++)
    A[i * n + j] =
        A[(n * j - n + j - i - 1];
```

$$\begin{cases} 0 \leq i_1 < n \\ i_1 + 1 \leq j_1 < n \\ 0 \leq i_2 < n \\ i_2 + 1 \leq j_2 < n \\ i_1 \times n + j_1 = n \times j_2 - n + j_2 - i_2 - 1 \end{cases}$$

Delinearized multi-dimensional array

```
for(int i = 0; i < n; i++)
  for(int j = i + 1; j < n; j ++)
    B[i][j] = B[j - 1][j - i - 1];
```

Delinearize the array accesses

Linearized one-dimensional array

```
for(int i = 0; i < n; i++)
  for(int j = i + 1; j < n; j ++)
    A[i * n + j] =
        A[(n * j - n + j - i - 1];
```

$$\begin{cases} 0 \leq i_1 < n \\ i_1 + 1 \leq j_1 < n \\ 0 \leq i_2 < n \\ i_2 + 1 \leq j_2 < n \\ i_1 \times n + j_1 = n \times j_2 - n + j_2 - i_2 - 1 \end{cases}$$

Delinearized multi-dimensional array

```
for(int i = 0; i < n; i++)
  for(int j = i + 1; j < n; j ++)
    B[i][j] = B[j - 1][j - i - 1];
```

$$\begin{cases} 0 \leq i_1 < n \\ i_1 + 1 \leq j_1 < n \\ 0 \leq i_2 < n \\ i_2 + 1 \leq j_2 < n \\ i_1 = j_2 - 1 \\ j_1 = j_2 - i_2 - 1 \end{cases}$$

Delinearize the array accesses

Linearized one-dimensional array

```
for(int i = 0; i < n; i++)
  for(int j = i + 1; j < n; j ++)
    A[i * n + j] =
        A[(n * j - n + j - i - 1];
```

$$\begin{cases} 0 \le i_1 < n \\ i_1 + 1 \le j_1 < n \\ 0 \le i_2 < n \\ i_2 + 1 \le j_2 < n \\ i_1 \times n + j_1 = n \times j_2 - n + j_2 - i_2 - 1 \end{cases}$$

Delinearized multi-dimensional array

```
for(int i = 0; i < n; i++)
  for(int j = i + 1; j < n; j ++)
    B[i][j] = B[j - 1][j - i - 1];
```

$$\begin{cases} 0 \le i_1 < n \\ i_1 + 1 \le j_1 < n \\ 0 \le i_2 < n \\ i_2 + 1 \le j_2 < n \\ i_1 = j_2 - 1 \\ j_1 = j_2 - i_2 - 1 \end{cases}$$

There is no integer solution, therefore, no dependence.

ORCCA
Ontario Research Centre for Computer Algebra

1. Loop counter can only be integers.

1. Loop counter can only be integers.
2. This leads to the problem of finding the integer points of a polyhedral set, called the iteration space.

1. Loop counter can only be integers.
2. This leads to the problem of finding the integer points of a polyhedral set, called the iteration space.
3. Often this space is parametric (for example the loop bounds).

1. Loop counter can only be integers.
2. This leads to the problem of finding the integer points of a polyhedral set, called the iteration space.
3. Often this space is parametric (for example the loop bounds).
4. Wang and Moreno Maza developed the integer hull algorithm for polyhedral sets in Maple [[MW21], [Wa22]].

1. Loop counter can only be integers.
2. This leads to the problem of finding the integer points of a polyhedral set, called the iteration space.
3. Often this space is parametric (for example the loop bounds).
4. Wang and Moreno Maza developed the integer hull algorithm for polyhedral sets in Maple [[MW21], [Wa22]].
5. Their algorithm is applicable to polyhedral sets of any given dimension.

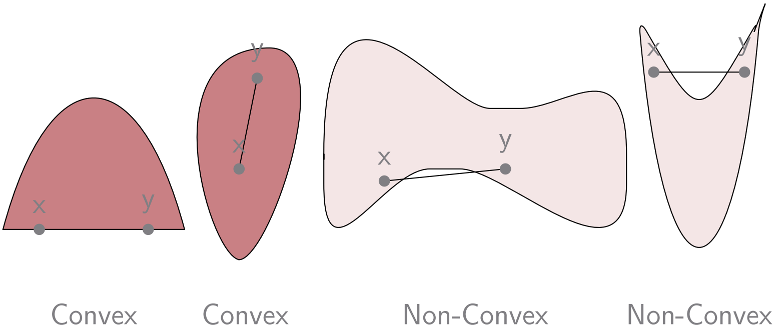
1. Loop counter can only be integers.
2. This leads to the problem of finding the integer points of a polyhedral set, called the iteration space.
3. Often this space is parametric (for example the loop bounds).
4. Wang and Moreno Maza developed the integer hull algorithm for polyhedral sets in Maple [[MW21], [Wa22]].
5. Their algorithm is applicable to polyhedral sets of any given dimension.
6. This algorithm relies on a recursive construction, effectively reducing the computation of integer hulls in an arbitrary dimensions to that of dimension 2.

1. Loop counter can only be integers.
2. This leads to the problem of finding the integer points of a polyhedral set, called the iteration space.
3. Often this space is parametric (for example the loop bounds).
4. Wang and Moreno Maza developed the integer hull algorithm for polyhedral sets in Maple [[MW21], [Wa22]].
5. Their algorithm is applicable to polyhedral sets of any given dimension.
6. This algorithm relies on a recursive construction, effectively reducing the computation of integer hulls in an arbitrary dimensions to that of dimension 2.
7. Hence, we will focus on the 2D case.

Definition

A set S is called convex if the line joining any two points in S is in S, i.e.,

$$\forall x, y \in S, \forall \lambda \in [0,1], \lambda x + (1-\lambda) y \in S.$$



Definition

A convex polyhedral set (or simply a polyhedral set) P is a set $\{\mathrm{x} \in \mathbb{R}^n \mid A\mathrm{x} \leq \mathrm{b}\}$, where $A \in \mathbb{R}^{m \times n}$ is a matrix and $\mathrm{b} \in \mathbb{R}^m$ is a vector.

Definition

A convex polyhedral set (or simply a polyhedral set) P is a set $\{\mathrm{x} \in \mathbb{R}^n \mid A\mathrm{x} \leq \mathrm{b}\}$, where $A \in \mathbb{R}^{m \times n}$ is a matrix and $\mathrm{b} \in \mathbb{R}^m$ is a vector.

Definition

A face F of P is a subset $\{\mathrm{x} \in P \mid A_{\mathrm{sub}}\mathrm{x} = \mathrm{b}_{\mathrm{sub}}\}$ for a sub-matrix A_{sub} of A and a sub-vector $\mathrm{b}_{\mathrm{sub}}$ of b.

Definition

A convex polyhedral set (or simply a polyhedral set) P is a set $\{\mathrm{x} \in \mathbb{R}^n \mid A\mathrm{x} \leq \mathrm{b}\}$, where $A \in \mathbb{R}^{m \times n}$ is a matrix and $\mathrm{b} \in \mathbb{R}^m$ is a vector.

Definition

A face F of P is a subset $\{\mathrm{x} \in P \mid A_{\mathrm{sub}}\mathrm{x} = \mathrm{b}_{\mathrm{sub}}\}$ for a sub-matrix A_{sub} of A and a sub-vector $\mathrm{b}_{\mathrm{sub}}$ of b.

Definition

A face distinct from P and of maximum dimension, is called a facet of P.
A face of dimension 0 is called a vertex of P.

Definition

A convex polyhedral set (or simply a polyhedral set) P is a set $\{x \in \mathbb{R}^n \mid Ax \leq b\}$, where $A \in \mathbb{R}^{m \times n}$ is a matrix and $b \in \mathbb{R}^m$ is a vector.

Definition

A face F of P is a subset $\{x \in P \mid A_{\text{sub}}x = b_{\text{sub}}\}$ for a sub-matrix A_{sub} of A and a sub-vector b_{sub} of b.

Definition

A face distinct from P and of maximum dimension, is called a facet of P. A face of dimension 0 is called a vertex of P.

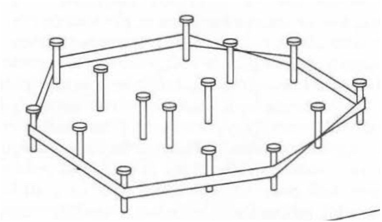
Definition

A vertex cone of P at v is the intersection of the half-spaces defining P and whose boundaries intersect at v.

Definition

The convex hull of a polyhedral set S is the set of all convex combination of S, given by,

$$conv(S) := \{\Sigma_{i=1}^{n} \lambda_i x_i \mid \Sigma_{i=1}^{n} \lambda_i = 1, \lambda_i \in [0, 1]\}.$$

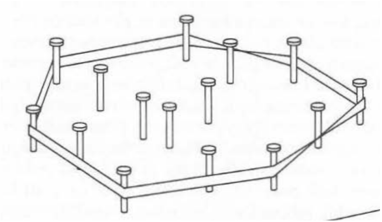


1984 – Dewdney's Analog Gadgets

Definition

The convex hull of a polyhedral set S is the set of all convex combination of S, given by,

$$conv(S) := \{\Sigma_{i=1}^{n} \lambda_i x_i \mid \Sigma_{i=1}^{n} \lambda_i = 1, \lambda_i \in [0, 1]\}.$$



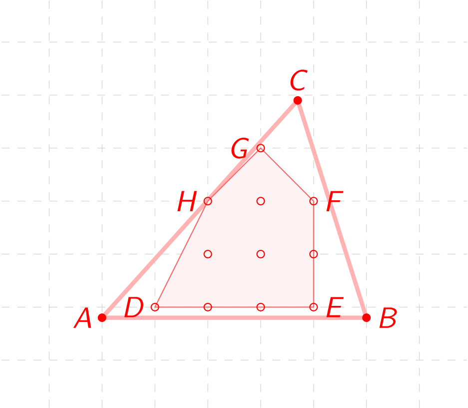
1984 – Dewdney's Analog Gadgets

The $conv(S)$ is the smallest convex set containing S, i.e. it is the intersection of all convex sets containing S.

Definition

The integer hull P_I of a convex polyhedral set P is the convex hull of integer points of P.

Example



Background

1. The cutting plane method by Gomory [Go10] to solve integer linear programming (ILP),
   - ILP is solved by introducing constraints at each step until an integer solution is found.
   - Chvátal [Ch73] and Schrijver [Sc80] provided a geometric description.

Background

1. The cutting plane method by Gomory [Go10] to solve integer linear programming (ILP),
   - ILP is solved by introducing constraints at each step until an integer solution is found.
   - Chvátal [Ch73] and Schrijver [Sc80] provided a geometric description.
2. The branch and bound method by Land and Doig [LD60],
   - Recursively divides the polyhedral set into sub-polyhedra.
   - The vertices of the integer hull of each part of the partition are then computed.

Background

1. The cutting plane method by Gomory [Go10] to solve integer linear programming (ILP),
   - ILP is solved by introducing constraints at each step until an integer solution is found.
   - Chvátal [Ch73] and Schrijver [Sc80] provided a geometric description.
2. The branch and bound method by Land and Doig [LD60],
   - Recursively divides the polyhedral set into sub-polyhedra.
   - The vertices of the integer hull of each part of the partition are then computed.
3. Pick's theorem [Pi99] can be used for calculating the area of any polygon with integer vertices.

Background

1. The cutting plane method by Gomory [Go10] to solve integer linear programming (ILP),
   - ILP is solved by introducing constraints at each step until an integer solution is found.
   - Chvátal [Ch73] and Schrijver [Sc80] provided a geometric description.
2. The branch and bound method by Land and Doig [LD60],
   - Recursively divides the polyhedral set into sub-polyhedra.
   - The vertices of the integer hull of each part of the partition are then computed.
3. Pick's theorem [Pi99] can be used for calculating the area of any polygon with integer vertices.
   - Barvinok [Ba94] created an algorithm for counting the number of integer vertices inside a polyhedron.

ORCCA
Ontario Research Centre for Computer Algebra

Background

1. The cutting plane method by Gomory [Go10] to solve integer linear programming (ILP),
   - ILP is solved by introducing constraints at each step until an integer solution is found.
   - Chvátal [Ch73] and Schrijver [Sc80] provided a geometric description.
2. The branch and bound method by Land and Doig [LD60],
   - Recursively divides the polyhedral set into sub-polyhedra.
   - The vertices of the integer hull of each part of the partition are then computed.
3. Pick's theorem [Pi99] can be used for calculating the area of any polygon with integer vertices.
   - Barvinok [Ba94] created an algorithm for counting the number of integer vertices inside a polyhedron.
   - Verdoolaege, Seghir, Beyls, Loechner and Bruynooghe [VSBLB07] came up with a method for counting the number of integer points in a non-parametric polytope.

ORCCA
Ontario Research Centre for Computer Algebra

Background

1. The cutting plane method by Gomory [Go10] to solve integer linear programming (ILP),
   - ILP is solved by introducing constraints at each step until an integer solution is found.
   - Chvátal [Ch73] and Schrijver [Sc80] provided a geometric description.
2. The branch and bound method by Land and Doig [LD60],
   - Recursively divides the polyhedral set into sub-polyhedra.
   - The vertices of the integer hull of each part of the partition are then computed.
3. Pick's theorem [Pi99] can be used for calculating the area of any polygon with integer vertices.
   - Barvinok [Ba94] created an algorithm for counting the number of integer vertices inside a polyhedron.
   - Verdoolaege, Seghir, Beyls, Loechner and Bruynooghe [VSBLB07] came up with a method for counting the number of integer points in a non-parametric polytope.
   - Seghir, Loechner and Meister [SLM12] developed a method for the parametric polytope case.

$\boxed{\text{Integer Hull Algorithm in Maple}}$

Three main steps of the algorithm:

- Normalization: construct a new polyhedral set Q from P as follows. Consider in turn each facet F of P:

Three main steps of the algorithm:

- Normalization: construct a new polyhedral set Q from P as follows. Consider in turn each facet F of P:
    1. if the hyperplane H supporting F contains an integer point, then H is a hyperplane supporting a facet of Q,

Three main steps of the algorithm:

- Normalization: construct a new polyhedral set Q from P as follows. Consider in turn each facet F of P:
  1. if the hyperplane H supporting F contains an integer point, then H is a hyperplane supporting a facet of Q,
  2. otherwise we slide H towards the center of P along the normal vector of F, stopping as soon as we hit a hyperplane H' containing an integer point, then making H' a hyperplane supporting a facet of Q.

  Clearly $Q_I = P_I$.

Three main steps of the algorithm:

- Normalization: construct a new polyhedral set Q from P as follows. Consider in turn each facet F of P:
  1. if the hyperplane H supporting F contains an integer point, then H is a hyperplane supporting a facet of Q,
  2. otherwise we slide H towards the center of P along the normal vector of F, stopping as soon as we hit a hyperplane H' containing an integer point, then making H' a hyperplane supporting a facet of Q.

  Clearly $Q_I = P_I$.
- Partitioning: make each part of the partition a polyhedron R which:

Three main steps of the algorithm:

- Normalization: construct a new polyhedral set Q from P as follows. Consider in turn each facet F of P:
  1. if the hyperplane H supporting F contains an integer point, then H is a hyperplane supporting a facet of Q,
  2. otherwise we slide H towards the center of P along the normal vector of F, stopping as soon as we hit a hyperplane H' containing an integer point, then making H' a hyperplane supporting a facet of Q.

  Clearly $Q_I = P_I$.
- Partitioning: make each part of the partition a polyhedron R which:
  1. either has integer points as vertices so that $R_I = R$,

Three main steps of the algorithm:

- Normalization: construct a new polyhedral set Q from P as follows. Consider in turn each facet F of P:
  1. if the hyperplane H supporting F contains an integer point, then H is a hyperplane supporting a facet of Q,
  2. otherwise we slide H towards the center of P along the normal vector of F, stopping as soon as we hit a hyperplane H' containing an integer point, then making H' a hyperplane supporting a facet of Q.

  Clearly $Q_I = P_I$.
- Partitioning: make each part of the partition a polyhedron R which:
  1. either has integer points as vertices so that $R_I = R$,
  2. or any brute force algorithm can be applied to compute R_I.

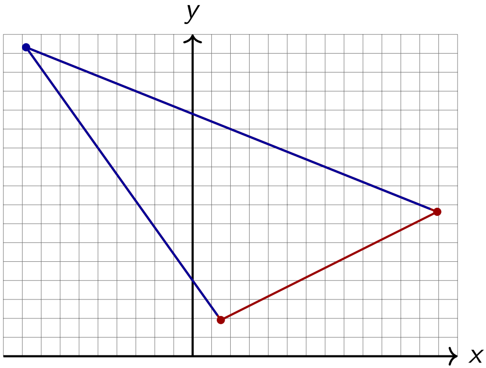
Three main steps of the algorithm:

- Normalization: construct a new polyhedral set Q from P as follows. Consider in turn each facet F of P:
  1. if the hyperplane H supporting F contains an integer point, then H is a hyperplane supporting a facet of Q,
  2. otherwise we slide H towards the center of P along the normal vector of F, stopping as soon as we hit a hyperplane H' containing an integer point, then making H' a hyperplane supporting a facet of Q.

  Clearly $Q_I = P_I$.
- Partitioning: make each part of the partition a polyhedron R which:
  1. either has integer points as vertices so that $R_I = R$,
  2. or any brute force algorithm can be applied to compute R_I.
- Merging: Once the integer hull of each part of the partition is computed and given by the list of its vertices, an algorithm for computing the convex hull of a set points, such as QuickHull, can be applied to deduce P_I.

Consider the triangle given by $(-\frac{44}{5}, \frac{408}{25}), (\frac{349}{27}, \frac{206}{27})$ and $(\frac{85}{57}, \frac{109}{57})$. We want to find the integer hull.

$$\begin{cases} 2x + 5y \leq & 64 \\ 7x + 5y \geq & 20 \\ 3x - 6y \leq & -7 \end{cases}$$



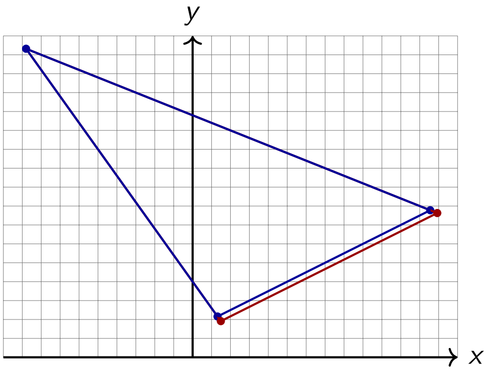
Normalization

The integer hull of the normalized polyhedral set should be the same as that of the input.

$$\begin{cases} 2x + 5y \leq & 64 \\ 7x + 5y \geq & 20 \\ 3x - 6y \leq & -7 \end{cases}$$

$$\downarrow$$

$$\begin{cases} 2x + 5y \leq & 64 \\ 7x + 5y \geq & 20 \\ 3x - 6y \leq & -9 \end{cases}$$



Normalization

The integer hull of the normalized polyhedral set should be the same as that of the input.

$with(PolyhedralSets):$
$inset1 := PolyhedralSet([2\cdot x+5\cdot y \le 64, 7\cdot x+5\cdot y \ge 20, 3\cdot x-6\cdot y \le -9], [x, y]):$
$IntegerHull(inset1);$

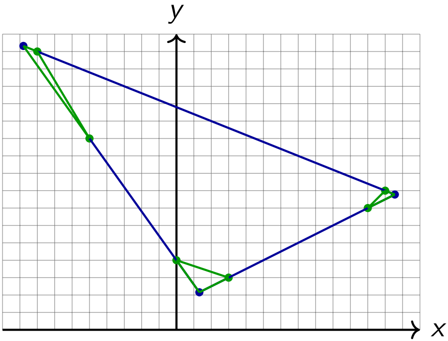
$$[[[12, 8], [-8, 16], [-7, 14], [-5, 11], [0, 4], [1, 3], [3, 3], [11, 7]], [\,]]$$

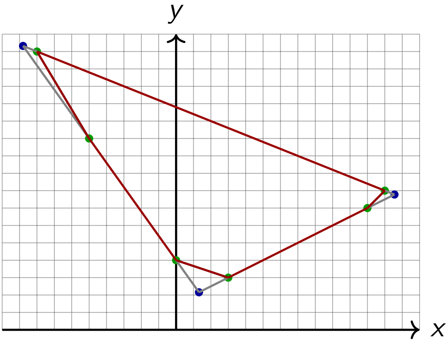
$inset2 := PolyhedralSet([2\cdot x+5\cdot y \le 64, 7\cdot x+5\cdot y \ge 20, 3\cdot x-6\cdot y \le -7], [x, y]):$
$IntegerHull(inset2);$

$$[[[12, 8], [-8, 16], [-7, 14], [-5, 11], [0, 4], [1, 3], [3, 3], [11, 7]], [\,]]$$

Partitioning

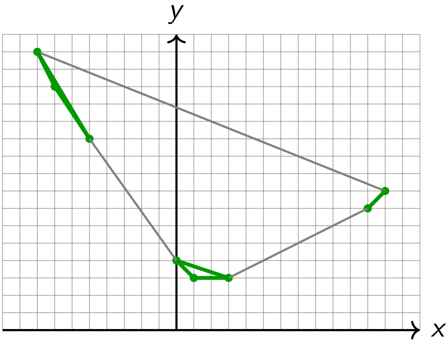
Find the closest integer points to each vertex on its adjacent facets. Apply brute force method to compute the integer hull of each corner.





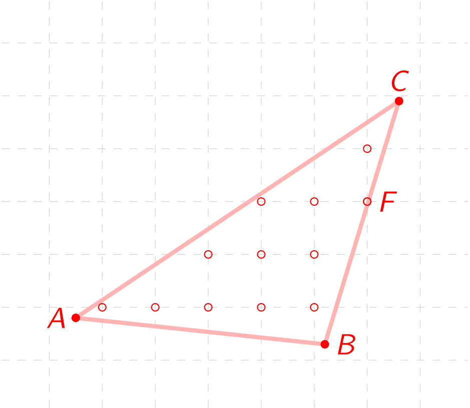
Merging

Compute the convex hull of all the integer hulls.



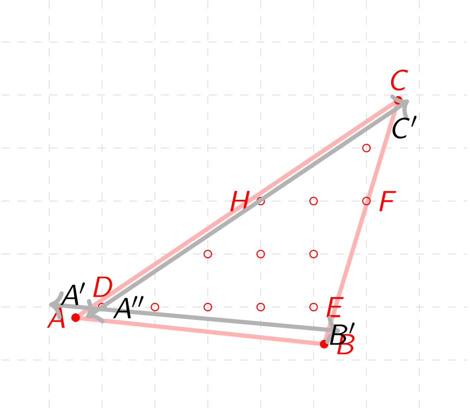
Example

Consider the $\triangle ABC$, where F is an integer point. We want to find the integer hull of $\triangle ABC$.



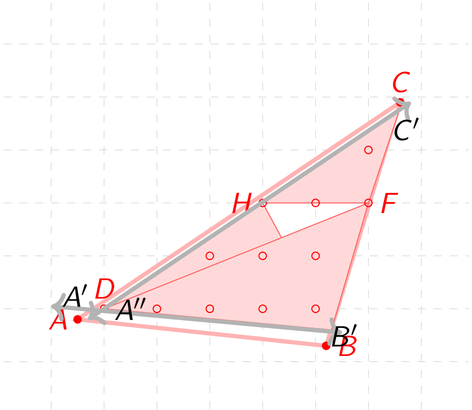
Example

To do so, we translate AB, AC till it touches D and H respectively. BC already has F. The new hyperplanes are $A'B'$ and $A''C'$ respectively.



Example

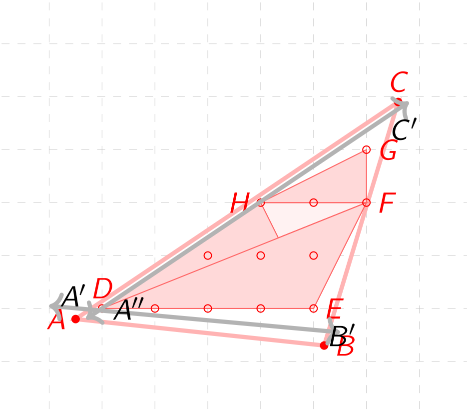
We apply brute force algorithm to compute integer hull of the corner.



The computational cost of brute force algorithm is inherently high, as such it is favorable to apply such a method only if the area is significantly small.

We apply `QuickHull` to all the integer hulls. The integer hull of $\triangle ABC$ is the pentagon $DEFGH$.

Example

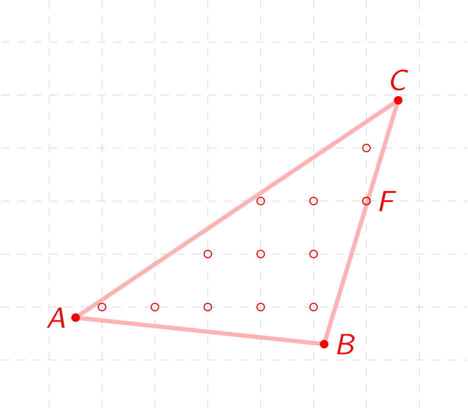


The compiler spends more time on the brute force algorithm than the "real" algorithm.

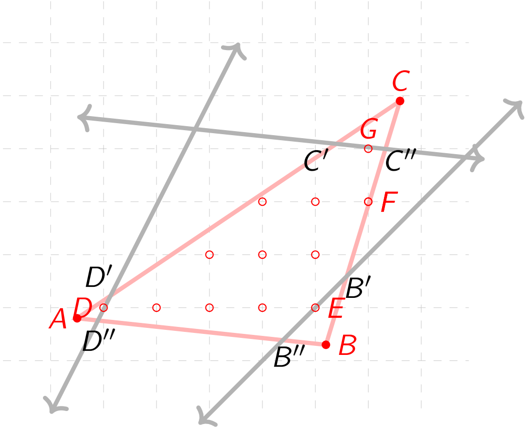
$\boxed{\text{New method}}$

Example

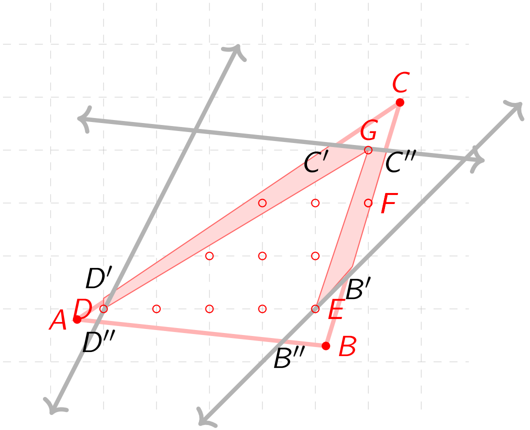
Consider the same $\triangle ABC$, where F is an integer point. We want to find the integer hull of $\triangle ABC$.



To do so, we translate AB, BC and AC to G, D and E respectively. The new hyperplanes are $C'C''D''D'$, and $B''B'$ respectively.

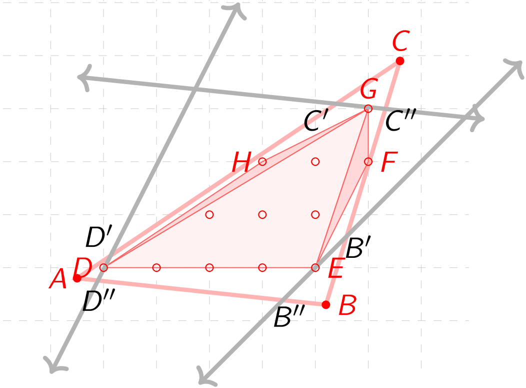


We aply brute force algorithm to compute integer hull of the corner.



The area on which brute force algorithm is applied is significantly small.

We apply `QuickHull` to all the integer hulls. The integer hull of $\triangle ABC$ is the pentagon $DEFGH$.



Preliminary Benchmark

The Maple version used is the 2024 release of Maple. All the benchmarks are done on an Intel Core i7-7700T with Clockspeed: 2.9 GHz and Turbo Speed: 3.8 GHz. It has 4 cores and 8 threads.

Preliminary Comparison Test			
Vertices	Volume	New Algo.	Existing Algo.
10	29.9	663.00ms	823.00ms
13	35.08	906.00ms	1.06ms
13	40.56	996.00ms	1.10s
12	40.63	898.00ms	928.00ms
1000	69829.26	855ms	952ms
15	263124.06	1.65s	1.73s
1000	6.54×10^6	1.98s	2.44s
1500	3.13×10^9	74.11s	89.72s

1. In this implementation we have only reduced the area on which the brute force algorithm is applied during partitioning to search for integer vertices in the corner.

1. In this implementation we have only reduced the area on which the brute force algorithm is applied during partitioning to search for integer vertices in the corner.
2. In the existing algorithm the integer vertices are sort recursively, that is for a given facet, the integer vertices on its adjacent facets are computed.

1. In this implementation we have only reduced the area on which the brute force algorithm is applied during partitioning to search for integer vertices in the corner.
2. In the existing algorithm the integer vertices are sort recursively, that is for a given facet, the integer vertices on its adjacent facets are computed.
3. Since, we are translate our hyperplane to its corresponding opposite vertex, we can find integer points simultaneously on both the adjacent facets and the "newly" introduced facet.

1. In this implementation we have only reduced the area on which the brute force algorithm is applied during partitioning to search for integer vertices in the corner.
2. In the existing algorithm the integer vertices are sort recursively, that is for a given facet, the integer vertices on its adjacent facets are computed.
3. Since, we are translate our hyperplane to its corresponding opposite vertex, we can find integer points simultaneously on both the adjacent facets and the "newly" introduced facet.
4. This will significantly cut down the time for computing integer vertices in the hyperplanes.

1. In this implementation we have only reduced the area on which the brute force algorithm is applied during partitioning to search for integer vertices in the corner.
2. In the existing algorithm the integer vertices are sort recursively, that is for a given facet, the integer vertices on its adjacent facets are computed.
3. Since, we are translate our hyperplane to its corresponding opposite vertex, we can find integer points simultaneously on both the adjacent facets and the "newly" introduced facet.
4. This will significantly cut down the time for computing integer vertices in the hyperplanes.
5. Brute force algorithm gets worse with increase in dimension. We expect to see imporvements in complexity for the general integer hull algorithm.

ORCCA Ontario Research Centre for Computer Algebra

References I

Utpal K. Banerjee.
Loop Transformations for Restructuring Compilers: The Foundations.
Kluwer Academic Publishers, 1960.

C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa.
The quickhull algorithm for convex hulls.
ACM Trans. Math. Softw., 22(4):469–483, 1996.

Alexander I. Barvinok.
A polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed.
Math. Oper. Res., 19(4):769–779, 1994.

Mohamed-Walid Benabderrahmane, Louis-Noël Pouchet, Albert Cohen, and Cédric Bastoul.
The polyhedral model is more widely applicable than you think.
International Conference on Compiler Construction, 2010.

References II

V. Chvátal.
Edmonds polytopes and a hierarchy of combinatorial problems.
Discrete Mathematics, 4(4):305–337, 1973.

Jesús A. De Loera, Raymond Hemmecke, Jeremiah Tauzer, and Ruriko Yoshida.
Effective lattice point counting in rational convex polytopes.
Journal of Symbolic Computation, 38(4):1273–1302, 2004.

Paul Feautrier.
Parametric integer programming.
RAIRO - Operations Research - Recherche Opérationnelle, 22(3):243–268, 1988.

Paul Feautrier.
Automatic parallelization in the polytope model.
Springer, 1996.

References III

Ralph E. Gomory.
Outline of an Algorithm for Integer Solutions to Linear Programs and An Algorithm for the Mixed Integer Problem.
Springer, 2010.

Matthias Köppe and Sven Verdoolaege.
Computing parametric rational generating functions with a primal barvinok algorithm.
arXiv preprint arXiv:0705.3651, 2007.

Ailsa H. Land and Alison G. Doig.
An automatic method of solving discrete programming problem.
Econometrica, 28:497, 1960.

Marc Moreno Maza and Linxiao Wang.
On the pseudo-periodicity of the integer hull of parametric convex polygons.
CASC, 2021.

References IV

- Georg Pick.
  Geometrisches zur zahlenlehre.
  Sitzenber. Lotos (Prague), 19:311–319, 1899.

- William Pugh.
  A practical algorithm for exact array dependence analysis.
  Commun. ACM, 35(8):102–114, 1992.

- A. Schrijver.
  On cutting planes.
  Combinatorics 79, volume 9 of Annals of Discrete Mathematics, pages 291–296. Elsevier, 1, 1980.

- Rachid Seghir, Vincent Loechner, and Benoît Meister.
  Integer affine transformations of parametric $\mathbb{Z}$-polytopes and applications to loop nest optimization
  ACM Trans. Archit. Code Optim., 9(2), 2012.

References V

Sven Verdoolaege, Rachid Seghir, Kristof Beyls, Vincent Loechner, and Maurice Bruynooghe.
Counting integer points in parametric polytopes using barvinok's rational functions
Algorithmica, 48:37–66, 2007.

Linxiao Wang.
Three Contributions to the Theory and Practice of Optimizing Compilers
PhD thesis, 2022.