Maple Transactions

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

CHIRANTAN MUKHERJEE, The University of Western Ontario, Canada

Recommended Reference Format:
Chirantan Mukherjee. 2024. A New Algorithm for Computing Integer Hulls of 2D Polyhedral Sets. 1, 1 (July 2024), 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

The integer points of rational polyhedral sets are of great interest in various areas of scientific computing. Two such areas are `combinatorial optimization` (in particular integer linear programming) and `compiler optimization` (in particular, the analysis, transformation and scheduling of for-loop nests in computer programs), where a variety of algorithms solve questions related to the points with integer coordinates belonging to a given polyhedron.

Wang and Moreno Maza have developed the existing integer hull algorithm for polyhedral sets in Maple [18, 12]. Their algorithm is applicable to polyhedral sets of any given dimension. However, for the purpose of our discussion, we will focus on the 2D case.

In this paper we propose a novel algorithm 2 for computing the integer hull of 2D polyhedral sets. The existing integer hull algorithm, as referenced in [18, 12], relies on a recursive construction, effectively reducing the computation of integer hulls in arbitrary dimensions to that of dimension 2. However, to enhance performance, we propose improvements for this base case. Our preliminary experiments with the new algorithm demonstrate its efficiency and ability to handle polyhedral sets with a large number of integer point.

There are three main families of integer hull algorithms for polyhedral sets: `cutting plane method`, `branch-and-bound method` and `lattice point counting method`.

The `cutting plane method` was introduced by Gomory [9] to solve integer linear programming and mixed integer linear programming. This method involves solving the linear programming problem to find the optimal solution. It does this by introducing new constraint at each step until an integer solution is found, which in turn reduces the area of the feasible solution. Chvátal [5] and Schrijver [15] provided a geometric description of this method and developed a procedure for computing the integer hull of a polyhedral set.

Land and Doig [11] introduced the `branch-and-bound method` for computing the integer hull of a polyhedral set. This method works by recursively partitioning the polyhedral set into sub-polyhedral sets, then computing the integer vertices of each of the sub-polyhedral set and finally merging them all together.

Pick's theorem [13] can be used for calculating the area of any polygon with integer lattice points. Using this idea Barvinok [3] created an algorithm for counting the number of integer lattice points inside a polyhedron. Building on Barvinok's algorithm, Verdoolaege, Seghir, Beyls, Loechner and Bruynooghe [17] came up with a method for counting the number of integer points in a non-parametric polytope. Meanwhile, Seghir, Loechner and Meister [16] developed a method of

Author's address: Chirantan Mukherjee, The University of Western Ontario, 1151 Richmond St, London, Canada, cmukher@uwo.ca.

https://doi.org/10.1145/nnnnnnn.nnnnnnn

counting the number of images by an affine integer transformation of the lattice points contained in a parametric polytope. In 2004, a software package named LATTE [6] was developed for counting number of integer points in a rational polyhedral set using Barvinok's algorithm.

The paper is organized as follows. Section 2 and 3 is a brief review of polyhedral geometry and the existing integer hull algorithm in Maple. Section 4 and 5 presents the new integer hull algorithm in Maple for 2D cases. Section 6 reports on our preliminary experimentation with the proposed algorithm.

2 Preliminaries

We denote by $\mathbb{Z}$, $\mathbb{Q}$ and $\mathbb{R}$ the ring of integers, the field of rational numbers and the field of real numbers. Unless specified otherwise, all matrices and vectors have their coefficients in $\mathbb{Z}$.

Definition 2.1. A `polyhedral set` P is a set which can be expressed as the intersection of finite number of (closed) half spaces, that is $\{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} \leq \mathbf{b}\}$, where $A \in \mathbb{R}^{m\times n}$ is a matrix and $\mathbf{b} \in \mathbb{R}^m$ is a vector.

Definition 2.2. A subset F of the polyhedron P is called a `face` of P if F equals $\{\mathbf{x} \in P \mid A_{\text{sub}}\mathbf{x} = \mathbf{b}_{\text{sub}}\}$ for a sub-matrix A_{sub} of A and a sub-vector $\mathbf{b}_{\text{sub}}$ of $\mathbf{b}$. A face of P, 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 2.3. Given a polyhedral set P and a vertex v of P, the *vertex cone* of P at v is the intersection of the half-spaces defining P and whose boundaries intersect at v.

In our study of two-dimensional polyhedral sets, we observe that every non-trivial face falls into one of two categories: either it is a `facet` (which can be a segment or a half-line), or it is a `vertex`. Furthermore, each vertex cone exhibits a simplicial structure. Specifically, a vertex cone is defined by two half-lines originating from the same point, that is the vertex of that particular cone.

3 Exisiting Integer Hull Algorithm

The integer points of polyhedral sets hold paramount importance in the context of the delinearization problem [4, 7], scheduling for-loop nests [8], accessing memory location of for-loop nests [10], Barvinok's algorithm for counting integer points in a polyhedral set [3] and many more.

Definition 3.1. The `integer hull` P_I of a convex polyhedral set P is the convex hull of integer points of P.

In this section we focus on implementing the current integer hull algorithm in Maple as described in [18, 12]. It has three main steps:

- `Normalization`: In this step, the polyhedron P is transformed into a rational polyhedron $Q \subseteq \mathbb{Q}^d$. This transformation ensures that each facet of Q has integer points on its supporting hyperplane, while maintaining $P_I = Q_I$.
- `Partitioning`: Here, integer points within Q are identified and used to partition Q into smaller polyhedral sets. Each set's integer hull can then be computed more straightforwardly.
- `Merging`: This step involves merging the integer hulls obtained from the partitioning process using a convex hull algorithm.

Example 3.2. The integer hull of the following triangle can be represented by the shaded pentagonal region in figure 1.

The integer hull P_I of a convex polyhedral set P can be described in terms of its vertices. Since P_I is the smallest polyhedral set containing the integer points of P, the vertices of P_I are necessarily integer points.

For $P = \Delta ABC$ given in the above example 3.2, we have VertexSet(P) = $\{A, B, C\}$, none of which are integer points. We translate the supporting hyperplane of the facet BC to the west until the supporting hyperplane of the facet BC has at least one integer point, call it F . We obtain a new triangle $Q = \Delta AB'C'$, which, clearly has the same integer points as P.

We then repeat the process for the supporting hyperplane of the facet AB' by translating it upwards until we get integer points $\{D, E\}$, obtaining a new facet $A'B''$. We do not need to translate the supporting hyperplane of the facet $A'C'$ since there is already an integer point H on it.

We see that the triangle $\Delta A'B''C'$ can be divided into 4 regions:

(1) the convex hull, say R, of the points $\{D, E, F, H\}$;
(2) three "small" triangles: $\Delta A'DH, \Delta HC'F, \Delta EB''F$.

Since the vertices of R are all integer points, R is its own integer hull. However, R may not be the integer hull of Q. Indeed, each of the small three triangles may still contain integer points. This is actually the case for $\Delta HC'F$.

Because these three triangles are generally small in practice, one can apply a brute force method to search for integer points. One brute force method for computing the integer hull P_I of a polyhedral set P is to use `Fourier-Motzkin elimination` [1, 14] to obtain a parametric representation of P. With such a parametric representation, one can enumerate all the integer points of P.

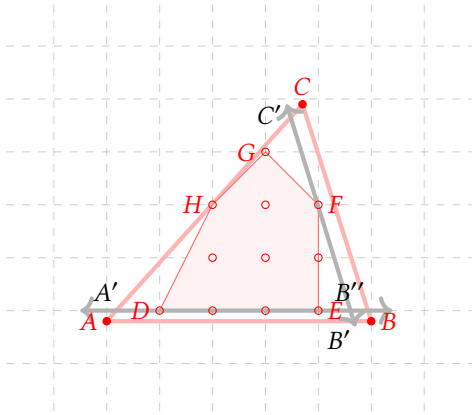

Fig. 1. Integer hull of a triangle.

We want to stress that the computational cost of any brute force algorithm is inherently high, as such it is favorable to apply such a method only if the area is significantly small. In the next section we will discuss our alternative algorithm 2 which is designed to ensure that the area on which the brute force method is applied remains small, thereby optimizing the computational efficiency.

Returning to our example, we complete the construction of the integer hull by putting together:

(1) the integer points found in each small triangle;
(2) the vertices of R;
(3) apply a convex hull algorithm, such as `QuickHull` [2], to all those points.

Let us first state the condition which guarantee the existence of integer points on any line [18, 12].

A (parametric) polyhedral set can be defined as the intersection of vertex cones. That is a polyhedral set $P(\mathbf{b})$ defined by $\begin{pmatrix} a_1 & c_1 \\ a_2 & c_2 \\ \vdots & \vdots \\ a_d & c_d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \leq \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_d \end{pmatrix}$ can be written as intersections of all vertex cones S_i, that is $P_i(\mathbf{b}) = \bigcap_{i=1}^{d} S_i(\mathbf{b})$, where each polyhedral cone S_i is given by $\begin{pmatrix} a_i & c_i \\ a_{i+1} & c_{i+1} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \leq \begin{pmatrix} b_i \\ b_{i+1} \end{pmatrix}$, for all $i \in [1, d]$ such that $d + 1 = 1$.

The following lemma gurantees the existence of an integer vertex on the supporting hyperplane of the facets in the polyhedral set.

Lemma 3.3 ([12] Lemma 1 on p. 256, [18] Lemma 1 on p. 13). *A line $ax + cy = b$, where $a, b, c \in \mathbb{Z}$ and $gcd(a, b, c) = 1$, can have integer vertex (x, y) if one of the following conditions is satisfied:*

(1) *If $a \neq 0$ and $c \neq 0$, then (x, y) is a integer vertex if and only if $gcd(a, c) = 1$, that is, $x \equiv \frac{b}{a}(mod c)$.*
(2) *If $a = 0$ (similarly $c = 0$), then (x, y) is a integer vertex if and only if $c = 1$ (similarly $a = 1$).*

Another crucial observation for constructing the integer hull is the ability to `translate` the supporting hyperplane of each facet. This translation ensures that we can find at least one integer point on it. In other words, the integer hull of the parametric polyhedral set $P_I(\mathbf{b})$ remains unchanged when translating the supporting hyperplane of the facets by an integer T, that is, $P_I(\mathbf{b} + T) = P_I(\mathbf{b})$.

We can combine everything that we have observed so far in this section and construct the integer hull of any parametric 2D polyhedral set as follows.

We translate each supporting hyperplane H of a facet F of the polyhedral set P inwards until H intersects an integer point of the integer hull of the polyhedral set P_I. Indeed,

(1) We can detect whether H has integer points or not by means of Lemma 3.3.
(2) If H has integer points, Lemma 3.3 gives a formula for them, which we can then plug into the system of inequalities defining P, so as to check whether some of those integer points of H are in P_I or not.

4 New Integer Hull Algorithm

In this section, we present our algorithm for computing the integer hull of a 2D polyhedral set. The integer hull algorithm proposed by Wang and Moreno Maza is extremely efficient as evidenced by the data presented in [[18] Table 4.5 on p. 51, Table 4.6 on p. 52]. However, it is crucial to note that when there are integer points present on the supporting hyperplane of the facets of the polyhedral set the computational cost become high. This observation becomes evident when considering the following example.

Example 4.1. Consider the following triangle $P = \Delta ABC$, where none of whose vertices are integer points. Following the algorithm discussed in the last section, we translate the supporting hyperplane of the facet AB upwards until it intersects at least one integer point, call it D. This leads to a new triangle $Q = \Delta A'B'C$, which retains the same integer points as P. We repeat this process for the supporting hyperplane of the facet $A'C$ by translating it to the east until we obtain integer points D, H. No adjustment is needed for the supporting hyperplane of facet $B'C'$ since it already contains an integer point, F.

The resulting triangle $\Delta A''B'C'$ can be partitioned into 3 regions:

- the convex hull, denoted as R, of the points $\{D, H, F\}$
- two triangles $A''B'F$, $HC'F$.

While the vertices of R are all integer points, it may not represent the integer hull of Q. Each of the smaller triangles, $A''B'F$ and $HC'F$, may still contain additional integer points. Therefore, to ensure completeness, a brute force procedure is employed to identify integer points within these triangles. However, it is noteworthy that given the larger area of $\Delta A''B'C'$ the brute force procedure will be computationally expensive. The time for obtaining the number of points using the brute force approach will surpass those obtained using the algorithm. In fact, the situation gets worse if there are integer points in the other edges as well.

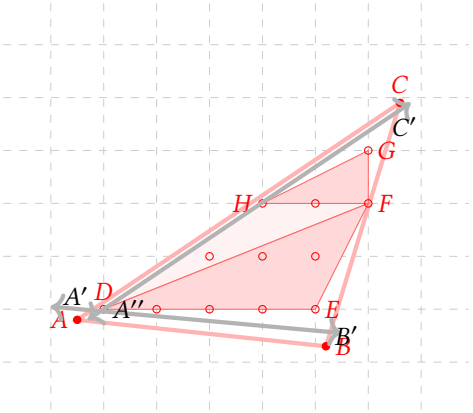

Fig. 2. Wang and Moreno Maza's integer hull algorithm.

We propose a new algorithm to reduce the area on which the brute force method is applied.

We initiate the algorithm by translating the supporting hyperplane of facet AB downward from the vertex C until it intersects at least one integer point, denoted as G. This process is then repeated for the supporting hyperplanes of facets AC and BC by translating them to the west from vertex B and east from vertex A, respectively, until integer points E and D are obtained.

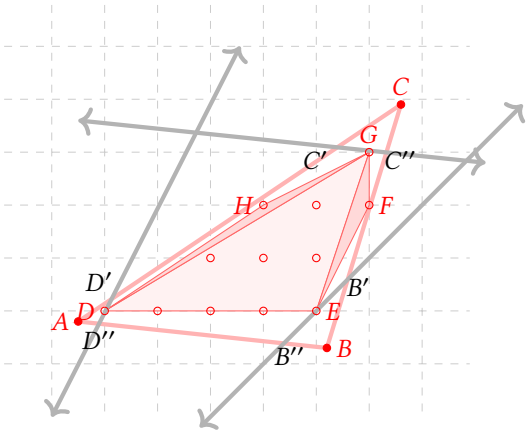

Fig. 3. New integer hull algorithm.

The hexagon $D''B''B'C''C'D'$ resulting from this algorithm can be subdivided into 4 regions:

(1) the convex hull, denoted as R, formed by the points $\{D, E, G\}$
(2) three quadrilaterals $DD''B''E$, $EB'C''G$ and $D'DGC'$.

While the vertices of R are all integer points, R may not represent the integer hull of Q. Indeed, each of the "small" quadrilaterals may still contain integer points, as is the case for $EB'C''G$ and $D'DGC'$.

To ensure that no integer points are missed, a brute force method can now be applied to search for integer points since the quadrilaterals are small. Alternatively, the algorithm can be repeated for quadrilaterals $EB'C''G$ and $D'DGC'$ until integer points H and F are obtained, adding these points to VertexSet(P_I). For quadrilateral $DD''B''E$, there are no integer points.

Hence, VertexSet(P_I) = $\{D, E, F, G, H\}$.

5 Algorithm

We provide an pseudo-code of the algorithm 2 for `Integer Hull Computation` using our procedure. The main steps of this algorithm rely on the function `ReplaceFacets` as described in algorithm 1.

5.1 ReplaceFacets Algorithm

In Line 1 and 2, we initialize the facets F to an empty list, and the integer vertices V' to V. In Line 6 the `getCoeffs` function calculates the equation of the line between vertices va and vb. This forms the facets F of P in line 7.

In Line 10, we choose the vertex v opposite to each facet F. In Line 12 - 16, we find the supporting hyperplane passing through v and is parallel to F.

In Line 17 - 18, we find lines that are interior to the vertex before and after v, and contains integer points using our `nearestLine` function. In Line 19 and 20, we use our `intersection` function to find the point of intersection of these new lines with supporting hyperplane passing through v and parallel to F.

In Line 21, we update the facet with F, and the intersection points to V' in Line 22.

5.2 IntegerHull Algorithm

If the input polyhedral set P is empty at Line 1, then we return an empty list in Line 2. Otherwise, in Line 3, we initialize the vertices and rays of P to V and R respectively using Maple's `VerticesAndRays` function, which is part of the `PolyhedralSets` package.

If the number of vertices of P is atleast 3 then we implement our `SortPoints`, `ReplaceFacets` and `ReplaceVertices` function, otherwise there can not be an integer hull and hence returns empty list in Line 10. The `SortPoints` function sorts the vertices of P based on angle with respect to a randomly selected origin vertex. The vertices obtained from the `ReplaceFacets` might contain rational vertices, which `ReplaceVertices` replaces with integer vertices that are contained within the polyhedral set. This is because the integer hull of a polyhedral set is contained within the polyhedral set itself.

In Line 11, we remove any duplicate vertices using Maple's `MakeUnique` command.

In Line 12 if the number of integer vertices are more than 3, we use Maple's `ConvexHull` function that is part of the `ComputationalGeometry` package, which computes the convex (integer) hull of the integer points. We return the integer hull of P in Line 15.

The integer hull of a polyhedral set is a polyhedral set. Therefore, the output of the algorithm is also a polyhedral set.

Algorithm 1 ReplaceFacets

Require: Vertices V of P.
Ensure: Replace the facets that could not have integer point with the ones that could have.

$F := []$ ▷ Initialize to empty list
$V' := V$ ▷ Initialize to V
for i from 1 to $|V|$ **do**
  $va := V'[i]$
  $vb := V'[\text{if}(i = |V|, 1, i + 1)]$
  $a, b, c := \text{getCoeffs}(va, vb)$ ▷ Coefficients of line $ax + cy = b$
  Append $[a, b, c]$ to F ▷ Facets of P
for i from 1 to $|V|$ **do**
  $a, b, c := F[i]$
  $val := \text{if}(\text{ceil}(|V|, 2) + i > |V|, \text{modp}(\text{ceil}(|V|/2) + i, |V|), \text{ceil}(|V|/2) + i)$
  $v := V[val]$ ▷ Choosing vertex which is opposite to the suppoting hyperplane
  if $c = 0$ **then**
    $b := v[1]$ ▷ Equation of a Parallel line through vertex v, when $ax = b$
  else
    $b' := v[2] + (a/c) \times v[1]$
    $b := c \times new_b$ ▷ Equation of a Parallel line through vertex v
  $a1, b1, c1 := \text{nearestLine}([a, b, c], V[\text{if}(val = 1, |V|, val - 1)])$ ▷ New line closer to v with integer points
  $a2, b2, c2 := \text{nearestLine}([a, b, c], V[\text{if}(val = |V|, 1, val + 1)])$
  $p1 := \text{intersection}([a, b, c], [a1, b1, c1])$ ▷ Intersection of two lines
  $p2 := \text{intersection}([a, b, c], [a2, b2, c2])$
  $F := [F[1], \ldots, F[i - 1], [a, b, c], F[i + 1], \ldots, F[|V|]$
  $V' := [V[1], \ldots, V[i - 1], p1, p2, V'[i + 2], \ldots, V[|V|]]$
return V'

Algorithm 2 NewIntegerHull

Require: P a polyhedral set.
Ensure: Integer hull of the polyhedral set P.

if $P = \phi$ **then**
  return [] ▷ Returns empty set if empty input
V, R := VerticesAndRays(P) ▷ Initialize the vertices and rays of P
if $|V| \geq 3$ **then**
  $\mathscr{V}$:= SortPoints(V, V[1]) ▷ Sorts the vertices of P counter-clockwise starting with the first vertex
  $\mathscr{V}$:= ReplaceFacets($\mathscr{V}$, P)
  $\mathscr{V}$:= ReplaceVertices($\mathscr{V}$) ▷ Replace the vertices with integer vertices (inside of P)
else
  $\mathscr{V}$:= []
$\mathscr{V}$:= MakeUnique($\mathscr{V}$) ▷ Remove the repeated integer vertices from the integer hull of P
if $|\mathscr{V}| > 3$ **then**
  v := ConvexHull($\mathscr{V}$) ▷ Form the convex (integer) hull
  $\mathscr{V}$:= $\mathscr{V}$[v]
return PolyhedralSet($\mathscr{V}$, R) ▷ Returns the integer hull of P

6 Experimentation

In this section, we report on the software implementation of the algorithms proposed in the previous sections. We have implemented the algorithms in Maple programming language. 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			
Number of Vertices	Volume	New Algorithm	Existing Algorithm
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

References

[1] Utpal K. Banerjee. *Loop Transformations for Restructuring Compilers: The Foundations*. Kluwer Academic Publishers, USA, 1993.

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

[3] 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, nov 1994.

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

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

[6] 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. Symbolic Computation in Algebra and Geometry.

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

[8] Paul Feautrier. Automatic parallelization in the polytope model. In *The Data Parallel Programming Model: Foundations, HPF Realization, and Scientific Applications*, page 79–103, Berlin, Heidelberg, 1996. Springer-Verlag.

[9] Ralph E. Gomory. *Outline of an Algorithm for Integer Solutions to Linear Programs and An Algorithm for the Mixed Integer Problem*, pages 77–103. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

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

[11] Ailsa H. Land and Alison G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28:497, 1960.

[12] Marc Moreno Maza and Linxiao Wang. On the pseudo-periodicity of the integer hull of parametric convex polygons. In François Boulier, Matthew England, Timur M. Sadykov, and Evgenii V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing*, pages 252–271, Cham, 2021. Springer International Publishing.

[13] Georg Pick. Geometrisches zur zahlenlehre. *Sitzenber. Lotos (Prague)*, 19:311–319, 1899.

[14] William Pugh. A practical algorithm for exact array dependence analysis. *Commun. ACM*, 35(8):102–114, aug 1992.

[15] A. Schrijver. On cutting planes**research supported by the netherlands organization for the advancement of pure research (z.w.o.). In Peter L. Hammer, editor, *Combinatorics 79*, volume 9 of *Annals of Discrete Mathematics*, pages 291–296. Elsevier, 1980.

[16] 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), jun 2012.

[17] 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.

[18] Linxiao Wang. *Three Contributions to the Theory and Practice of Optimizing Compilers*. PhD thesis, Electronic Thesis and Dissertation Repository, 2022.