Uniform Random Sampling in Polyhedra

Benoit Meister, Philippe Clauss

Uniform Random Sampling in Polyhedra

Benoît Meister
Reservoir Labs
Denver, USA
meister@reservoir.com

Philippe Clauss
INRIA CAMUS, ICube lab.
University of Strasbourg
Strasbourg, France
philippe.clauss@inria.fr

Abstract

We propose a method for generating uniform samples among a domain of integer points defined by a polyhedron in a multi-dimensional space. The method extends to domains defined by parametric polyhedra, in which a subset of the variables are symbolic. We motivate this work by a list of applications for the method in computer science. The proposed method relies on polyhedral ranking functions, as well as a recent inversion method for them, named *trahrhe* expressions.

1 Introduction

Some stochastic computations assume a uniform distribution over the domain of valid samples. This paper looks at the problem of obtaining such uniform distribution when the domain is defined as the set of integer points in an arbitrary polyhedron. Examples of this problem occur in compilers, and in stochastic comparison of polyhedra.

The shape of a polyhedron in a multi-dimensional space presents the main difficulty in performing such task. We assemble a set of polyhedral tools, including a recent one, to propose a solution to the problem of sampling uniformly among integer points of a multi-dimensional polyhedron. We also note that the presented technique can be computed symbolically, where constraints of the polyhedron are defined with statically unknown offsets. The computation is then valid for a family of polyhedra defined as a function of these parametric offsets.

The main contributions of this paper are a discussion of the use of stochastic polyhedral relationships, the presentation of a novel application to the so-called *trahrhe* method to uniformly sample integer points within parametric polyhedra, and a qualitative comparison of the novel method with an existing sampling method, which can be adapted to obtain uniform sampling.

After deepening our motivation for this problem in section 2, we provide some technical background about polyhedra, and the enumeration of integer points in polyhedra in section 3. Then, we explain our method for randomly sampling integer points within polyhedra with a uniform distribution in section 4. We spend some time presenting the recent tool used in this technique, the Trahrhe method, in section 5. Finally, we go through related work in section 7, and reconsider our findings in a broader context in section 8.

2 Motivation

The problem of uniformly sampling integer points in a polyhedron is found in various domains that rely on polyhedra. In this section, we present a few of them related to the discipline of polyhedral program optimization. In this so-called "polyhedral model," the execution of the program within repetitive control structures called "loops" (as in FORTRAN's `DO` loops, and C's `for` loops) is modeled in a vector space, as integer-valued points within a multi-dimensional polyhedron. Loops can contain other loops, and the depth of such loop inclusion defines the basic dimension of the polyhedra representing all the iterations, called "iteration space" of the loop nest. Ultimately, computations, data accesses and semantic-preserving orderings (called "dependencies") between operations performed in the modeled loops are modeled with polyhedra.

Uniform sampling within a set (here, the points of polyhedra that have integer coordinates) means that each point in the set has equal chances of being picked by the sampling mechanism. If there are n points in the set, the probability for an element of the set to be sampled is $1/n$.

While we can only imagine that any discipline relying on integer points within polyhedra has a use for random sampling, the ones we could think about within the polyhedral model of program loops include the following.

2.1 Iterative Compilation

The ability to seed the search space with a uniform distribution impacts the quality of several machine learning algorithms, including variants of random search [15], genetic algorithms, hill-climbing, simulated annealing, and particle swarm optimization.

For instance, our method enables uniform sampling among valid program transformations (schedules) in an iterative compilation framework in arbitrary polyhedral domains, a principle used in a form of auto-tuning called "iterative compilation". Bastoul's approach [3] samples schedules uniformly in a simpler superset of the valid solutions and "corrects" the solutions that fall outside of the feasible set. While the correction method was made more general and tuned to the needs of polyhedral model optimization in terms of the difference between the originally-picked schedule and the corrected one [2], it still does not enable a uniform distribution. Intuitively, the set of corrected schedules bias the distribution of randomly sampled points toward the edges of the polyhedron.

Still in the context of iterative compilation, an approach proposed by Pouchet [12] relies on polyhedral scanning (a technique that expresses the integer points of polyhedra back as a loop nest) to enumerate the points. This method can be adapted to generate a uniform sampling, and we compare it with the method presented here in section 7.

2.2 Stochastic polyhedral operators

One powerful aspect of polyhedral tools such as the ones used in polyhedral compilation [13], is that they manipulate parametric polyhedra. We define them in section 3, but basically, the definition of a parametric polyhedron depends upon a set of symbolic variables, called the parameters of the polyhedron. This is powerful, as it enables the manipulation of a whole family of polyhedra at once, or alternatively a polyhedron whose parameters are not instantiated at the time we need to perform computations on it.

Unfortunately, this makes the calculation of some operations and relationships among polyhedra harder. Also, some relationships hold for some parameter values, but not all. We may then wonder *to which degree* the relationship holds, or if it is *mostly* true or false within the parameter domain. Such quantification can be expensive, and sometimes impossible.

An example of impossibility occurs when writing a greedy algorithm that prioritizes the processing of a set of polyhedra heuristically using an ordering relationship. For instance, in existing program optimization algorithms [9], the number of iterations in a loop nest or the number of data reuses between pairs of array references are used to guide the optimization. The issue here is that the integer volume of a parametric polyhedron is given by its Ehrhart polynomial. Ehrhart polynomials are pseudo-polynomials, which are piecewise, periodic polynomial functions. By combining Bernstein polynomials [7] with polynomial bounds of pseudo-polynomials [11], it is possible to detect whether a pseudo-polynomial is *always* greater than another pseudo-polynomial for a useful number of cases. However, there isn't generally such an absolute order between two arbitrary pseudo-polynomials.

To address this problem with parametric polyhedra, we can compute a stochastic relationship, whose output is the expectation that the relationship is true. Unless we know a prior distribution of the parameter values, the relevant sample distribution we should use to compute the expectation is uniform. Let $rel(A, B, N)$ a binary relationship between polyhedra A and B which encodes true as 1 and false as 0 for any value N of their parameters. Its expectation over the parameter domain D can be stochastically approximated as follows:

$$E(rel, A, B) = \frac{1}{|U|} \sum_{N \in U} rel(A, B, N) \quad (1)$$

where U represents the sample set within D.

Let us develop a bit on the particular example of volume comparison mentioned above, where we compute a stochastic comparison of parametric polyhedra in terms of their integer volume. In this case

$$rel(A, B, N) = \begin{cases} 1 & \text{if } intvol(A, N) > intvol(B, N) \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Since the goal is to sort the polyhedra, the approximate expectation is not the most appropriate measure. We would rather go back to a binary relationship such as "rel(A, B, N) is mostly true across D":

$$mt(rel, A, B) = (E(rel, A, B) \geq 0.5) \quad (3)$$

This example is easy to generalize to other relationships for which we only want to know if they will for most parameter values.

Alternatively, we may want to give more importance to cases where the polyhedra are big. In terms of polyhedral compilation, the intuition here is that there is more performance to gain by making the best decisions for larger loop nests, because they represent more computation, and hence more time saved when optimized. One approach toward this is to compute the *mt* relationship as above, but within a domain of parameters reduced to large values. Another approach is to stochastically represent the *difference* of integer volumes by sampling a finite set of parameter instances uniformly among the domain of parameters (itself a polyhedron).

Finally, we may want to give emphasis to how much bigger than the other one a polyhedron becomes as parameters change, irrespective of absolute size. In this case, we would compute a stochastic volume ratio.

The expected value of a relationship can also be computable exactly, for instance if we want to test for inclusion, topological comparison, intersection, etc. A general method there is to compute the sub-domains of D for which the relationship is true, and compare the sum of their integer volumes with the integer volume of D. The key drawback of such precise computation is that it introduces an operation (integer volume) that can be much more expensive than the original type of operation we want to test for (e.g., inclusion, topological comparison, intersection).

Another convenient aspect of stochastic relationships is that they come with a natural trade-off between cost of computation and precision, as precision and cost of computation both increase with the number of samples.

2.3 Random testing

Random testing [8] is a well-known technique to find bugs in libraries and programs. In random testing, inputs are randomly generated and fed to a program or an API through the test harness, to check for bugs. A large number of generated inputs are usually desired. Integer polyhedral random sampling is useful in mostly two cases here. First, when the domain of a tested function is only a small subset of the space of random inputs, we may be able to over-approximate the domain with a polyhedron, and only draw samples from it. This can be integrated in more sophisticated techniques, as for instance concolic testing [**?**], which relies on a mix of dynamic sampling and static program analysis to increase the relevance of the random search space. Second, when the inputs of tested functions are polyhedra, one method to generate a large number of polyhedra is to define a parametric polyhedron and to instantiate a sequence of parameter values to random values within the valid parameter space of such polyhedron. The test harness of the R-Stream compiler [10] includes random polyhedral testing, which is useful to any piece of polyhedral software [1, 13]. The requirement for the sampling to be uniform is problem-specific.

3 Background

3.1 Polyhedra

A polyhedron P in a vector space V is a subset of V, which is the set of solutions to a set of linear inequalities

$$P : Ax + b \geq 0, x \in V \tag{4}$$

Geometrically, (4) defines P as the intersection of half-spaces in V, each of the inequalities (each row of A in (4)) defining one half-space. A finite polyhedron is called a polytope.

When resolving problems on polyhedra – such as measuring their volume for instance – it is possible to consider a subset of the dimensions of V as special variables, which do not get instantiated. Such variables are called the parameters of the polyhedron. Let us consider a parametric polytope example presented in [5], which has two variables (i, j) and two parameters (n, m):

Example 3.1.

$$Q(i,j) = \{(i,j) \in \mathbb{Z}^2 : 0 \leq i \leq \frac{n}{2}; 0 \leq j \leq \frac{m}{2}\} \tag{5}$$

Q is basically a rectangle whose lower left corner is at $(0, 0)$ and top right at $(\frac{n}{2}, \frac{m}{2})$. Since the parameters n and m are (by definition) not instantiated, the solution to the polyhedral problem is expressed as a function of these parameters. The number of integer points (a.k.a. its "integer volume") in Q is

$$\#Q(n.m) = \frac{1}{4}nm + \begin{bmatrix} \frac{1}{4} & \frac{1}{2} \end{bmatrix}_m n + \begin{bmatrix} \frac{1}{4} & \frac{1}{2} \end{bmatrix}_n m + \begin{bmatrix} \frac{1}{4} & \frac{1}{2} \\ \frac{1}{2} & 1 \end{bmatrix}_{n,m} \tag{6}$$

Such an expression is called an *Ehrhart polynomial.* As presented in [4, 5], the square bracket notation in (6) represent periodic numbers. A periodic number is a function of v integer variables, $x \in \mathbb{Z}^v$, represented as a matrix A of size $S \in \mathbb{N}^v$, which is defined as $a(v) = A_{x mod S}$, where *mod* represents the elementwise integer division remainder operator. Their periodic nature come from the need to compute integer points next to the vertices of their input polyhedron. Computing such integer points involve fractional parts of rational affine functions[1], which are periodic functions. For instance, the periodic number $\begin{bmatrix} \frac{1}{4} & \frac{1}{2} \\ \frac{1}{2} & 1 \end{bmatrix}_{n,m}$, a periodic function of n and m, can be written as $(1 - frac(\frac{n+1}{2})) \times (1 - frac(\frac{m+1}{2}))$, where $frac(x)$ is the fractional part of x. The Ehrhart polynomial of a polyhedron whose vertices have integer coordinates is not periodic (it is a regular polynomial).

3.2 Ranking

Ehrhart polynomials were proposed and extended for program analysis by Clauss in [4]. These integer-valued polynomials express the exact number of integer points contained in a finite multi-dimensional convex polyhedron which depends linearly on integer parameters. When considering a d-dimensional polyhedron depending linearly on integer parameters $p_1, p_2, \ldots, p_m$, its Ehrhart polynomial is a polynomial of degree d whose variables are $p_1, p_2, \ldots, p_m$, and whose coefficients are periodic numbers. However, when the vertices of the convex polyhedron have all integer coordinates, then all the coefficients are constant. Ehrhart polynomials can be automatically computed using existing algorithm implementations as the one of the barvinok library [14].

Among their applications, Ehrhart polynomials are used by Clauss and Meister in [6] to reorganize the memory layout of array elements accessed by a loop nest, in order to improve their spatial data locality: array elements are relocated in memory in the same order as they are accessed. In this approach, the new location of an array element is given by the order, or *rank*, of the iteration referencing it. Such a rank of iterations is given by a polynomial, called a *ranking polynomial*, whose variables are the loop iterators, and whose evaluation results in the number of iterations preceding a given iteration.

More formally, the *ranking polynomial* of a d-dimensional polytope whose integer points coordinates are $(i_1, i_2, \ldots, i_d)$, is denoted by $r(i_1, i_2, \ldots, i_d)$. If $(0, 0, \ldots, 0)$ defines the lexicographic minimum point of the polytope, then $r(0, 0, \ldots, 0) = 1$, $r(0, 0, \ldots, 1) = 2$, and so on. If $(N_1, N_2, \ldots, N_d)$ are the indices of the lexicographic maximum point, then the total number of integer points in the polytope is equal to $r(N_1, N_2, \ldots, N_d)$.

The computation of the ranking polynomial of a polytope is detailed in [6]. We recall this technique using the following example.

[1] or equivalently, division remainders of rational affine functions

Example 3.2. Consider the parametric polytope:

$$\{(i,j,k) \in \mathbb{Z}^3 | 0 \le i < N, 0 \le j \le i, 0 \le k < M\}$$

The rank of a given point (i_0, j_0, k_0) is equal to the number of points that are lexicographically less than (i_0, j_0, k_0) (included):

$\forall (i_0, j_0, k_0)$ s.t. $0 \le i_0 < N$ and $0 \le j_0 \le i_0$ and $0 \le k_0 < M$,

$$r(i_0,j_0,k_0) = \#\{(i,j,k) \mid (i,j,k) \trianglelefteq (i_0,j_0,k_0), \\ 0 \le i < N, 0 \le j \le i, 0 \le k < M\}$$

where $\trianglelefteq$ denotes the lexicographic order. Since lexicographic inequalities are not linear, the problem is split as the conjunction of three equivalent sets of linear inequalities, according to the definition of the lexicographic order:

$$(i,j,k) \trianglelefteq (i_0,j_0,k_0) \Leftrightarrow (i < i_0) \text{ or } (i = i_0 \text{ and } j < j_0) \text{ or } \\ (i = i_0 \text{ and } j = j_0 \text{ and } k \le k_0)$$

Therefore, the sets whose integer points must be counted can be defined as the union of three disjoint convex polyhedra, and $r(i_0, j_0, k_0)$ as the sum of three Ehrhart polynomials:

$$\begin{aligned} r(i_0,j_0,k_0) &= \#\{(i,j,k) \mid 0 \le i < i_0, 0 \le j \le i, 0 \le k < M\} \\ &+ \#\{(i,j,k) \mid i = i_0, 0 \le j < j_0, 0 \le k < M\} \\ &+ \#\{(i,j,k) \mid i = i_0, j = j_0, 0 \le k \le k_0\} \\ &= \frac{M\, i_0\, (i_0+1)}{2} + M\, j_0 + k_0 + 1 \\ &= \frac{2\,k_0 + 2\,M\,j_0 + M\,i_0^2 + M\,i_0 + 2}{2} \end{aligned}$$

One can verify that the rank of the first point $(0,0,0)$, $r(0,0,0)$, is equal to 1, the rank of the second point $r(0,0,1) = 2$, the rank of the third point $r(0,0,2) = 3$ and so on. The rank of the triplet when $i = 0, j = 0$ and $k = M-1$, $r(0,0,M-1) = M$, and the rank of the point when $i = 1, j = 0$ and $k = 0$, $r(1,0,0) = M+1$. The total number of points is $r(N-1, N-1, M-1) = \frac{M\,N\,(N+1)}{2}$.

Note that since ranking polynomials are actually particular Ehrhart polynomials, they may also, as Ehrhart polynomials, be defined on several domains of parameter values called *validity domains* [4, 5]. However, their associated validity domains obviously include the variables of the initial polytope, for which the ranking is given by the polynomial. In the previous example, there is only one ranking polynomial defined on one unique validity domain, which is actually equal to the initial polytope.

Such a ranking polynomial associates, to each tuple, a unique integer of a dense interval of integers starting at 1. This interval is the range of integers between 1 and the total number of points. Conversely, each integer value in the interval is associated to one unique tuple. Another important property is that such a ranking polynomial is monotonically increasing over the integers, from 1 to the total number of points, relatively to the lexicographic order of the tuples. Thus, a ranking polynomial defines a *bijection* between the tuples and the interval of successive integers. It implies that in theory, it can be inverted. Such inversions result in expressions called *trahrhe expressions*.

3.3 Inverse ranking

Recent developments brought up a way to invert Ehrhart polynomials as long as they define a ranking. This technique, known as the Trahrhe method, is fairly complex, and it will be explained in detail in Section 5. What matters to us now is it gives us a function U that maps any integer point in [1, intvol(P)] to a distinct n-dimensional integer point within P.

4 Method

Let P a polyhedral domain, as defined earlier. A ranking function of P forms a bijective mapping between integer points of P and an interval of natural numbers.

$$rank_P(I) : I \in P \cap \mathbb{Z}^n \rightarrow [1, intvol(P)] \cap \mathbb{N}$$

$$rank_P(I_1) = rank_P(I_2) \Leftrightarrow I_1 = I_2, \forall I_1 \in P \cap \mathbb{Z}^n, I_2 \in P \cap \mathbb{Z}^n$$

By definition, the inverse $invrank_P(x)$ of $rank_P(I)$ is also a bijective mapping, this time from an interval of natural numbers to the integer points of P, a n-dimensional polyhedral domain.

$$invrank_P(x) : x \in [1, intvol(P)] \cap \mathbb{N} \rightarrow I \in P \cap \mathbb{Z}^n$$

$$invrank_P(x_1) = invrank_P(I_2) \\ \Leftrightarrow x_1 = x_2, \forall (x_1, x_2) \in [1, intvol(P)]^2 \cap \mathbb{N}^2$$

Given a Polyhedron P, we can then derive a bijective function from the interval $[1, intvol(P)]$ to the exact set of integer points included in P, by computing the inverse $invrank_P$ of a ranking function of P.

Theorem 4.1. *$invrank_P$ converts uniform samples within $[1, intvol(P)]$ into uniform samples within the integer points of P.*

Proof. Let $rand_P$ be a uniform sampling function within $[1, intvol(P)]$. This means that each integer $x = rand_P \in [1, intvol(P)]$ has an equal chance of being chosen. Since $invrank_P$ is bijective over $[1, intvol(P)]$, it follows that each integer point $I = invrank_P(rand_P)$ has an equal chance of being chosen as well. In other words, $invrank_P(rand_P)$ is a uniform sampling function within the integer points of P. □

5 Trahrhe expressions

Notations: Let us first introduce some notations. Let N denote a set of identifiers representing integer parameters and variables. Let P and V be two subsets of N such that:

$$P \subset N, V \subset N, P \cap V = \varnothing, P \cup V = N$$

We denote by $D_{P \rightarrow V}$ the polyhedral domain defined by affine constraints on elements of N, where P is considered as the set of parameters and V as the set of variables. If $\#N$, $\#P$ and $\#V$ denote respectively the number of elements of N, P and V, then $\#N = \#P + \#V$. We also denote by $lexmin[D_{P\to V}]$, respectively $lexmax[D_{P\to V}]$, the tuple of $\#V$ symbolic values, which is the lexicographic minimum, respectively maximum, of the $(\#V)$-tuples of $D_{P\to V}$. These values may either depend linearly on elements of P, or be integer constants.

Let $lexmin[D_{P\to V}][k]$, respectively $lexmax[D_{P\to V}][k]$, denote the k^{th} component of the $(\#V)$-tuple $lexmin[D_{P\to V}]$, respectively $lexmax[D_{P\to V}]$, *i.e.*, the value of the k^{th} component of the lexicographic minimum, respectively maximum, $(\#V)$-tuple.

As already mentioned in Subsection 3.2, for a given parametric polytope whose parameters are in P and variables are in V, several associated ranking polynomials r_i may be defined on several validity domains $D^i_{P\to V}$. As it is for general Ehrhart polynomials, these domains are adjacent. Additionally, they are also lexicographically ordered: the $(\#V)$-tuples belonging to a validity domain $D^i_{P\to V}$ are ranked according to the lexicographic order, and their continuous ranking values are given by r_i. These values form a dense interval of integers, whose lower, respectively upper, bound is the rank of the lexicographic minimum, respectively maximum, tuple of D_i. We denote by $pcmin_i$, respectively $pcmax_i$, this lower, respectively upper, bound:

$$pcmin_i = r_i(lexmin[D^i_{P\to V}]), pcmax_i = r_i(lexmax[D^i_{P\to V}])$$

Finally, all the validity domains are also lexicographically ordered. Let n be the total number of validity domains $D^i_{P\to V}$:

$$pcmin_1 = 1 \le pcmax_1 \le pcmin_2 \le pcmax_2 \le \ldots \le pcmax_n$$

Note that $pcmax_n$ is the total number of integer points of the initial polytope, since $D^n_{P\to V}$ contains its lexicographic maximum tuple, while $D^1_{P\to V}$ contains its lexicographic minimum tuple ($pcmin_1 = 1$).

In the following, we consider one of these domains $D^i_{P\to V}$, its associated ranking polynomial r_i and lexicographic minimum tuples, since the same process must be repeated for each ranking polynomial r_i and associated domain $D^i_{P\to V}$.

Computing the trahrhe expressions: For a given rank pc, such that $pcmin_i \le pc \le pcmax_i$, $trahrhe_i(pc)$ is the tuple $(t_1, t_2, \ldots, t_d)$ of V such that $r_i(t_1, t_2, \ldots, t_d) = pc$, *i.e.*, a solution of the latter equation. The definition of function $trahrhe_i(pc)$ is calculated incrementally by first determining t_1, then propagating it to determine t_2, and so on. At each step, a symbolic uni-variate polynomial equation is solved.

Find t_1**:** Let $A = \{t_1\} \cup P$ and $B = V - \{t_1\}$. Solve $r_i(s, lexmin[D^i_{A\to B}]) - pc = 0$. Depending on the degree q of this uni-variate polynomial equation – the greatest power of s in the monomials –, there may be q real or complex solutions $s_1, \ldots, s_q$. Among these solutions, only one solution s_k, $1 \le k \le q$, is such that $t_1 = \lfloor s_k \rfloor = lexmin[D^i_{P\to V}][1]$ when $pc = pcmin_i$,

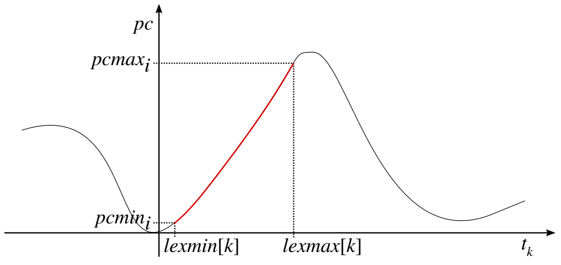

Figure 1. Generic graphical representation of the curve of a ranking polynomial $r(t_1, \ldots., t_{k-1}, t_k, lexmin[D^i_{A\to B}])$, where $A = P \cup \{t_1, \ldots, t_k\}$ and $B = V - \{t_1, \ldots, t_k\}$

which means that for the very first integer point of $D^i_{P\to V}$ following the lexicographic order ($pc = pcmin_i$), the value of the first component t_1 is, as expected, $lexmin[D^i_{P\to V}][1]$. This solution, which is parametrized by pc and P, is propagated in the next equation.

Find t_2**:** Let $A = \{t_1, t_2\} \cup P$ and $B = V - \{t_1, t_2\}$. Solve $r(t_1, s, lexmin[D^i_{A\to B}]) - pc = 0$. Once again, among the solutions, only one solution s_l is such that $t_2 = \lfloor s_l \rfloor = lexmin[D^i_{P\to V}][2]$ when $pc = pcmin_i$. This solution, which is parametrized by pc, t_1 and P, is propagated in the next equation.

Find t_3**:** Let $A = \{t_1, t_2, t_3\} \cup P$ and $B = V - \{t_1, t_2, t_3\}$. Solve $r(t_1, t_2, s, lexmin[D^i_{A\to B}]) - pc = 0$.

...

Find t_d**:** Let $A = \{t_1, \ldots, t_{d-1}\} \cup P$ and $B = V - \{t_1, \ldots, t_{d-1}\}$. Solve $r(t_1, \ldots, t_{d-1}, s) - pc = 0$. This last equation is necessarily linear and its solution is:

$$t_d = pc - r(t_1, t_2, t_3, \ldots, lexmin[D^i_{A\to B}])$$

The typical situation of each step is represented graphically in Figure 1: obviously, the ranking polynomial is monotonically increasing on its definition domain according to the lexicographic order of its variables in V. When instantiated as a uni-variate polynomial, it is monotonically increasing according to one unique variable t_k, whose definition domain is the interval whose lower bound is $lexmin[D^i_{P\to V}][k]$ and upper bound is $lexmax[D^i_{P\to V}][k]$ (noted $lexmin[k]$ and $lexmax[k]$ in the Figure).

As an example, let us compute step by step the trahrhe expressions of the parametric polytope:

$$Q = \{(i, j, k) \in \mathbb{Z}^3 | 0 \le i < L, 0 \le j \le i, 0 \le k < M\}$$

1. $P = \{L, M\}, V = \{i, j, k\}$
2. $r(i, j, k) = \frac{2\,k+2\,M\,j+M\,i^2+M\,i+2}{2}$, $D_{P\to V} = Q$
3. $lexmin(D_{P\to V}) = (0, 0, 0)$, $lexmin(D_{\{i,L,M\}\to\{j,k\}}) = (0, 0)$ and $lexmin(D_{\{i,j,L,M\}\to\{k\}}) = 0$
4. Solve $r(s, 0, 0) - pc = \frac{M\,s^2+M\,s+2}{2} - pc = 0$. This equation has two solutions:

$$s_1 = -\frac{\sqrt{8\,M\,pc + M^2 - 8\,M} + M}{2\,M}, \; s_2 = \frac{\sqrt{8\,M\,pc + M^2 - 8\,M} - M}{2\,M}$$

When $pc = 1$, $s_2 = 0$. Thus $t_1 = \left\lfloor \frac{\sqrt{8\,M\,pc+M^2-8\,M}-M}{2\,M} \right\rfloor$

5. Solve $r(t_1, s, 0) - pc = \frac{M\,t_1^2+M\,t_1+2\,M\,s+2}{2} - pc = 0$. This equation has one solution. Thus $t_2 = \left\lfloor -\frac{M\,t_1^2+M\,t_1-2\,pc+2}{2\,M} \right\rfloor$
6. Finally, $t_3 = pc - r(t_1, t_2, 0) = -\frac{2\,M\,t_2+M\,t_1^2+M\,t_1-2\,pc+2}{2}$

One can verify that for rank $pc = 2$, the trahrhe expressions result in $(0, 0, 1)$ as expected:

- $t_1 = \left\lfloor \frac{\sqrt{M\,(M+8)}-M}{2\,M} \right\rfloor = 0$ for any $M > 1$
- $t_2 = \left\lfloor \frac{2}{2\,M} \right\rfloor = 0$ for any $M > 1$
- $t_3 = \frac{2}{2} = 1$

For rank $pc = M$, the trahrhe expressions result in $(0, 0, M - 1)$ as expected:

- $t_1 = \left\lfloor \frac{\sqrt{M(9\,M-8)}-M}{2\,M} \right\rfloor = 0$ for any $M > 1$
  since $0 < \frac{\sqrt{M(9\,M-8)}-M}{2\,M} < \frac{\sqrt{9\,M^2}-M}{2\,M} = 1$
- $t_2 = \left\lfloor 1 - \frac{1}{M} \right\rfloor = 0$ for any $M \geq 1$
- $t_3 = \frac{2\,M-2}{2} = M - 1$

Remarks: Note that the roots of the solved uni-variate polynomial equations may include radicals (square roots, cubic roots, ...), and such radicals may result in complex numbers. However, the imaginary part of the final entire numerical evaluation of the resulting trahrhe expression is always null. Nevertheless, it means that intermediate evaluations have to be performed using *complex* floating-point arithmetic.

This equation solving process has been implemented in our software computing trahrhe expressions. In addition, several issues regarding arithmetic precision and mathematical generalization had to be fixed.

6 Caveats

With the current method, only polynomial equations whose degree is at most equal to 4 can be solved symbolically, with exact expressions for roots. The handled uni-variate polynomial equations are built from a multi-variate ranking polynomial, where one index t_k is set as the equation unknown, indices $t_1, ..., t_{k-1}$ are set as symbolic parameters, and indices $t_{k+1}, ..., t_d$ are set to their lexicographic minimum values. Thus, to ensure that such a built equation has a degree less than 4, the ranking polynomial must be such that any index t_k, in any of its monomials, has a degree less than 4, *i.e.*, any monomial is of the form: $a\,t_1^{p_1}\,t_2^{p_2}\,...\,t_d^{p_d}$ where a is a rational number, and every power p_k is such that $0 \leq p_k \leq 4$. In practice, this is often not a problem, since it means that the range of at most four variables at a time can depend upon another given variable.

We also want to point out that while we compute the mapping from $\mathbb{N}$ to a point in a parametric polyhedron, we are not aware of a sampling procedure that would sample uniformly within a symbolically-bound interval of $\mathbb{N}$. In fact, we are not even convinced that it would be feasible. Hence we want to underline that what is presented here is the mechanism to compute the mapping from $[1, \#P]$ to points of a parametric polyhedron P, for any instance of the parameters, in such a way that the sampling is maintained uniform within P.

7 Related work

A method for selecting integer points within a polyhedron, with which a uniform distribution can be obtained, was presented by Pouchet *et al.* in the context of iterative compilation [12]. The proposed method is based on polyhedral scanning, i.e., it enumerates all integer points in their lexicographic order, and randomly decides to select the visited point. The complexity of this method is sensitive to the shape of the polyhedral domain (through polyhedral scanning) and to the size of the scanned polyhedron, since every point needs to be scanned. We can easily derive a variant of this technique that guarantees uniformity given a uniform random sampler on an interval of $\mathbb{N}$, by starting the scan at the beginning for each sample. By contrast, the method presented here only depends upon the shape of the polyhedron. It has to be noted, however, that the complexity of computing the ranking and trahrhe expressions is significantly more expensive than polyhedral scanning. Another difference between the presented technique and the scanning method is that the latter does not operate on parametric polyhedra. However, the polyhedral scanning part can be performed parametrically and instantiated at runtime with little overhead.

Finally, the complexity of generating r random points in an *instantiated* polyhedron P is in $O(r)$ with the technique presented here, while it is in $O(r \times \#P)$ with the scanning-based technique. In the context of loop program optimization, execution time of the resulting program is critical, and hence, computations performed at runtime need to be minimized. By contrast, longer optimization times are accepted in the community. Hence, our technique seems more appropriate when the size of the parametric polyhedron P is going to be large in some instances, or even when its size is difficult to predict. When dealing with parametric iteration domains, it is often the case that the values that these parameters will take at runtime is unknown.

8 Conclusion

We presented a novel way to compute a closed-form mapping from the natural numbers to the set of integer points in a polyhedron, based on ranking functions and their inverse, obtained through the trahrhe method. We used this mapping to enable uniform random sampling of integer points within parametric (or non-parametric) polyhedra, with a constant time per random sample. However, there remain limitations in the trahrhe method, which we plan to address in the future. Besides incorporating traditional polynomial separation strategies, we conjecture that using slight over-approximations of the polyhedra, we can still generate near-uniform distributions in constant time.

References

[1] R.o Bagnara, Patricia M. Hill, and Enea Zaffanella. 2004. *Parma Polyhedral Library (version 6.1)*. http://www.cs.unipr.it/ppl/Documentation/ppl-devref-0.6.1-browse.pdf

[2] Cédric Bastoul. 2016. Mapping Deviation: A Technique to Adapt or to Guard Loop Transformation Intuitions for Legality. In *CC'2016 25th International Conference on Compiler Construction*. Barcelone, Spain. https://hal.inria.fr/hal-01271998

[3] Cédric Bastoul and Paul Feautrier. 2005. Adjusting a program transformation for legality. *Parallel processing letters* 15, 1 (March 2005), 3–17. http://icps.u-strasbg.fr/people/bastoul/public_html/research/papers/BF05-PPL.pdf

[4] Philippe Clauss. 1996. Counting Solutions to Linear and Nonlinear Constraints Through Ehrhart Polynomials: Applications to Analyze and Transform Scientific Programs. In *Proceedings of the 10th International Conference on Supercomputing (ICS '96)*. New York, NY, USA, 278–285.

[5] Philippe Clauss and V. Loechner. 1998. Parametric Analysis of Polyhedral Iteration Spaces. *J. VLSI Signal Process. Syst.* 19, 2 (1998), 179–194. https://doi.org/10.1023/A:1008069920230

[6] Ph. Clauss and B. Meister. 2000. Automatic Memory Layout Transformation to Optimize Spatial Locality in Parameterized Loop Nests. *ACM SIGARCH, Computer Architecture News* 28, 1 (2000).

[7] Ph. Clauss and I. Tchoupaeva. 2004. A Symbolic Approach to Bernstein Expansion for Program Analysis and Optimization. In *13th International Conference on Compiler Construction, CC 2004 (LNCS)*, Evelyn Duesterwald (Ed.), Vol. 2985. Springer, 120–133.

[8] Richard Hamlet. 1994. Random Testing. In *Encyclopedia of Software Engineering (1st ed.)*, John J. Marciniak (ed.) (Ed.). John Wiley and Sons.

[9] Benoit Meister, Eric Papenhausen, and Benoit Pradelle. 2019. Polyhedral Tensor Schedulers. In *THe 2019 International Conference on High Performance Computing and Simulation (HPCS 2019)*. Dublin, Ireland.

[10] Benoit Meister, Nicolas Vasilache, David Wohlford, Muthu Baskaran, Allen Leung, and Richard Lethin. 2011. R-Stream Compiler. In *Encyclopedia of Parallel Computing*, David Padua (Ed.). Springer Reference.

[11] B. Meister and S. Verdoolaege. 2008. Polynomial Approximations in the Polytope Model: Bringing the Power of Quasi-Polynomials to the Masses. In *ODES-6: 6th Workshop on Optimizations for DSP and Embedded Systems*. http://www.imec.be/odes/odes6-proceedings.pdf

[12] L.-N. Pouchet, C. Bastoul, A. Cohen, and N. Vasilache. 2007. Iterative optimization in the polyhedral model: Part I, one-dimensional time. In *IEEE/ACM Fifth International Symposium on Code Generation and Optimization (CGO'07)*. IEEE Computer Society press, San Jose, California, 144–156.

[13] Sven Verdoolaege. 2010. isl: an integer set library for the polyhedral model. In *Proceedings of the Third international congress conference on Mathematical software (ICMS'10)*. ACM Press, 299–302.

[14] S. Verdoolaege, R. Seghir, K. Beyls, V. Loechner, and Maurice Bruynooghe. 2007. Counting Integer Points in Parametric Polytopes Using Barvinok's Rational Functions. *Algorithmica* 48, 1 (2007), 37–66. https://doi.org/10.1007/s00453-006-1231-0

[15] Zelda Zabinsky. 2009. *Random search algorithms*. Technical Report. U. Washington. http://courses.washington.edu/inde510/516/AdapRandomSearch4.05.2009.pdf