Improved Bounds of Integer Solution Counts via Volume and Extending to Mixed-Integer Linear Constraints

Cunjing Ge
National Key Laboratory for Novel Software Technology, Nanjing University, China
School of Artificial Intelligence, Nanjing University, China

Armin Biere
University of Freiburg, Germany

Abstract

Solution counting and solution space integration over linear constraints are important problems with many applications. Previous works addressed either only counting integer points in polytopes (integer counting) with integer variables or alternatively only computing the volume of polytopes (solution space integration) on variables over the reals, including approximating the integer count via a polytope's volume. We are not aware of a non-trivial algorithm which addresses the mixed case, where linear constraints are over mixed integer and real variables. In this paper, we propose a new randomized algorithm to approximate guarantees (bounds) of integer solution counts. Then we present upper and lower bounds for solution space integration over mixed-integer linear constraints. Thus, proposed algorithms can be extended to mixed-integer cases as well. The experiments show that approximations are often very close to exact results in practice, and bounds approximated by our algorithm are often tight and useful.

2012 ACM Subject Classification Theory of computation → Automated reasoning

Keywords and phrases Integer Solution Counting, Mixed-Integer Linear Constraints, #SMT(LA) Problems, Volume of Polytopes

Digital Object Identifier 10.4230/LIPIcs.CP.2024.11

Supplementary Material Source code of the prototype tool MixIntCount and experimental data including benchmarks can be found at:
Software: `https://github.com/bearben/mixintcount/`
archived at `swh:1:dir:9101f2a1faa2381bc4779dff649915a21ff3cbae`

Acknowledgements Cunjing Ge is supported by the National Natural Science Foundation of China (62202218), and is sponsored by CCF-Huawei Populus Grove Fund (CCF-HuaweiFM202309).

1 Introduction

As one of the most fundamental types of constraints, linear constraints (LCs) have been studied thoroughly in many areas. Counting solutions over LCs has also many applications, such as counting-based search [27, 32], simple temporal planning [16] and probabilistic program analysis [14, 23]. Moreover, it can be incorporated into DPLL (T)-based #SMT (LA) counters [12] as a core subroutine.

Since a set of LCs corresponds to a convex polytope, counting integer solutions over LCs is equivalent to counting integer points inside the polytope. For real solutions, the counting problem is turned into computing the polytope's volume, which is defined by the Lebesgue measure. Naturally, we may be interested in the solution counting problem over mixed-integer variables. In this paper, we will show that it is a problem of computing the integration of solution space, and then study the methods for approximating such integrations. We will call such a problem *solution space integration* for short.

Solution counting problems over LCs were proved to be #P-hard [29]. Barvinok [1, 2] introduced an algorithm for integer counting. Based on it, tools LattE [18] and barvinok [30] were implemented, which are still state-of-the-art. For volume of polytopes, tool Vinci [5] is an implementation and combination of several volume computation algorithms. The Multiphase Monte-Carlo algorithm [9, 21] is a polynomial time volume approximation algorithm for convex bodies. The tool Polyvest [10] is a scalable implementation of the Multiphase Monte-Carlo algorithm over polytopes. A more recent work [11] studied the relation between the count of inner integer points and the volume of a polytope. They proposed an algorithm called Vol2Lat to approximate integer counts via volume. This work inspired us to investigate the relationship among solution space integration, a polytope's volume and inner integer points count. For example, let us consider the following simplified formula F extracted from a scheduling problem:

$$(2a + 0.3b \leq t) \wedge (1 \leq a \leq 32) \wedge (1 \leq b \leq 32) \wedge (0 < t < 50),$$

where a and b are the numbers of scheduled tasks A and B, the coefficient 2 and 0.3 are the time cost of two tasks, t is the time limit, which is a real variable. The solution space integration of F is 14204.5. Therefore, when we uniformly pick an assignment (a_0, b_0, t_0) satisfying the range constraints, the probability that (a_0, b_0, t_0) satisfies F is $14204.5/(32 * 32 * 49) \approx 0.283$. Assuming $a, b, t \in \mathbb{R}$, the volume of its solution space is 14418.9. Assuming $a, b, t \in \mathbb{Z}$, the integer solution count is 15261. From experiments (see Section 5), we observe that the values of volume and integer count are usually close to the solution space integration. Naturally, given a set of mixed-integer LCs, we would like to investigate whether solution space integrations can be approximated via the volume or the integer lattice count of the corresponding polytope. The contributions of this paper are the following:

- We propose a new randomized algorithm to approximate bounds of integer solution counts of a set of linear constraints via a polytope's volume. It returns upper and lower bounds of integer solution counts with respect to a given confidence.
- We introduce and prove bounds for the solution space integration, volume and integer solution count on a set of LCs with mixed-integer, pure real and pure integer variables respectively. The bounds show when the three values are close to each other in theory. Thus, our new algorithm can be directly extended to approximate bounds of solution space integration on mixed-integer cases.
- Experiments show that our approach is promising over mixed-integer cases and also outperforms the existing bound approximation algorithm Vol2Lat, including instances generated from program analysis.

2 Background

2.1 Notations and Preliminaries

A Linear Constraint (LC) can be written in the form $\sum_{i=1}^{n} a_i x_i \ \mathsf{op}\ b$, where x_is are numeric variables, a_is and b are real coefficients, and $\mathsf{op} \in \{<, \leq, >, \geq, =\}$. From a geometric point of view, a LC is an n-dimensional halfspace, and a finite set of LCs is a polytope. Thus a set of LCs F corresponds to a polytope P which is in the form of

$$P = \{\vec{x} \in \mathbb{R}^n : A\vec{x} \leq \vec{b}\}.$$

Naturally, $\mathbb{Z}^n$ represents the set of all integer lattices (points with all integer coordinates). Thus integer models of the linear constraints can be represented by $\{\vec{x} \in \mathbb{Z}^n : A\vec{x} \leq \vec{b}\}$. It is the same as the integer points inside the corresponding polytope, i.e.,

$$\{\vec{x} \in \mathbb{Z}^n : A\vec{x} \leq \vec{b}\} = P \cap \mathbb{Z}^n.$$

Now we consider Mixed-Integer Linear Constraints (MILC) whose variables include not only reals but also integers. We know that changing the sequence of variables of $\vec{x}$ will not affect the size of the solution space of a set of LCs, such as, exchanging two variables x_i, x_j and their coefficients a_i, a_j. So without loss of generality, a set of MILCs F can be written in the form $A\vec{x} = A_1\vec{x_I} + A_2\vec{x_R} \leq \vec{b}$, where $A = [A_1 A_2]$, $\vec{x} = [\vec{x_I}\vec{x_R}]$ and $\vec{x_I}, \vec{x_R}$ are subsets of integer and real variables of $\vec{x}$ respectively.

▶ **Definition 1.** *Given a set of MILCs F, which corresponds to a polytope P.*

- *Let $\texttt{vol}(P)$ denote the volume of P, i.e., $\texttt{vol}(P) = \int_{\vec{x}\in P} 1 \, \mathrm{d}\vec{x}$.*
- *Let $\texttt{lat}(P)$ denote the count of integer lattices in P, i.e., $\texttt{lat}(P) = |P \cap \mathbb{Z}^n|$.*
- *Let $n_I = |\vec{x_I}|$ and $n_R = |\vec{x_R}|$. Obviously, $n = n_I + n_R$.*
- *Let $\mathcal{M}(F) = \{\vec{x} = [\vec{x_I}\vec{x_R}] \in \mathbb{Z}^{n_I} \times \mathbb{R}^{n_R} : A_1\vec{x_I} + A_2\vec{x_R} \leq \vec{b}\} = P \cap \mathbb{Z}^{n_I} \times \mathbb{R}^{n_R}$ denote the solution space of F.*
- *Let $\mathcal{M}_{\mathcal{I}}(F) \subset \mathbb{Z}^{n_I}$ and $\mathcal{M}_{\mathcal{R}}(F) \subset \mathbb{R}^{n_R}$ denote the projection from $\mathcal{M}(F)$ to variables over $\vec{x_I}$ and $\vec{x_R}$ respectively.*
- *Given an integer assignment $\vec{\alpha_I}$ over $\vec{x_I}$. Let $F\{\vec{x_I} = \vec{\alpha_I}\}$ denote the remaining constraints of F by assigning $\vec{\alpha_I}$ to $\vec{x_I}$, i.e., $F\{\vec{x_I} = \vec{\alpha_I}\} = A_1\vec{\alpha_I} + A_2\vec{x_R} \leq \vec{b}$.*
- *Let $\texttt{integral}(F)$ denote the integral on $\mathcal{M}(F)$. In detail,*

$$\texttt{integral}(F) = \sum_{\vec{\alpha_I}\in\mathcal{M}_{\mathcal{I}}(F)} \int_{\vec{x}\in\mathcal{M}(F\{\vec{x_I}=\vec{\alpha_I}\})} 1 \, \mathrm{d}\vec{x} \tag{1}$$

$$= \sum_{\vec{\alpha_I}\in\mathcal{M}_{\mathcal{I}}(F)} \texttt{integral}(F\{\vec{x_I} = \vec{\alpha_I}\}). \tag{2}$$

Note that $\mathcal{M}(F\{\vec{x_I} = \vec{\alpha_I}\})$ is essentially a polytope in n_R-dimensional space. Let $P_{\vec{\alpha_I}}$ represent the corresponding polytope of $F\{\vec{x_I} = \vec{\alpha_I}\}$. We have $\texttt{integral}(F\{\vec{x_I} = \vec{\alpha_I}\}) = \texttt{vol}(P_{\vec{\alpha_I}})$. Consider an assignment $\vec{\beta_I}$ on $\vec{x_I}$ s.t. $\vec{\beta_I} \notin \mathcal{M}_{\mathcal{I}}(F)$, then $\texttt{vol}(P_{\vec{\beta_I}})$ must be zero, otherwise, $\exists\vec{\beta_R} \in \mathcal{M}(F\{\vec{x_I} = \vec{\beta_I}\})$ and $[\vec{\beta_I}\vec{\beta_R}]$ would be a solution of F which contradicts with $\vec{\beta_I} \notin \mathcal{M}_{\mathcal{I}}(F)$. Therefore, Equation (2) is equivalent with

$$\texttt{integral}(F) = \sum_{\vec{\alpha_I}\in S} \texttt{vol}(P_{\vec{\alpha_I}}), \forall \mathcal{M}_{\mathcal{I}}(F) \subseteq S \subset \mathbb{Z}^{n_I}. \tag{3}$$

It also indicates that $\vec{\alpha_I}$s can be enumerated in a looser space S than $\mathcal{M}_{\mathcal{I}}(F)$.

▶ **Definition 2.** *An **integer-cube** is a unit-cube whose center is an integer point. Given an integer point $\vec{\alpha}$ and a polytope P.*

- *Let $\texttt{cube}(\vec{\alpha})$ denote the integer-cube centered at $\vec{\alpha}$, i.e.,*

$$\texttt{cube}(\vec{\alpha}) = \{\vec{x} \in \mathbb{R}^n : \alpha_i - \frac{1}{2} \leq x_i \leq \alpha_i + \frac{1}{2}, i = 1, \ldots, n\}.$$

- *Let $\mathrm{C}(P)$ represent the set of all integer-cubes which intersect with P.*
- *Let $\mathrm{C}(\mathrm{B}(P))$ represent the set of all integer-cubes which intersects with $\mathrm{B}(P)$, where $\mathrm{B}(P)$ is the boundary (facets) of P.*
- *Let $\texttt{union}(C)$ denote the union $\bigcup_{\kappa\in C} \kappa$, where C is a set of integer-cubes.*

Note that each integer-cube corresponds to a unique integer point and its volume is 1. Therefore, the integer-cube is introduced for bridging the gap between the volume and the integer count. Figure 1 is an example of integer-cubes, C(P), C(B(P)), etc.

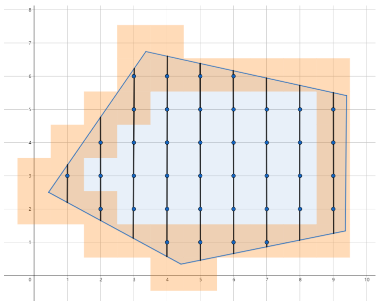


Figure 1 An example over two variables x and y, where x is an integer variable and y is a real variable. The integration `integral`(F) is the sum of lengths of black lines parallel to y-axis. The count `lat`$(P) = 40$ is the number of dots on those black lines. $\mathrm{C}(P)$ is the set of orange and gray squares, $\mathrm{C}(\mathrm{B}(P))$ is the set of orange squares, and thus the set of gray squares can be represented by $\mathrm{C}(P) \setminus \mathrm{C}(\mathrm{B}(P))$. `union`$(\mathrm{C}(P))$ is the union space of orange and grey squares.

2.2 Approximating Lattice Counts via Polytope's Volume

Ge et al. [11] observed that the lattice count and the volume of a given polytope are often close. They also pointed out that there exist cases in which lattice counts and volume are greatly different. For example, a very 'thin' rectangle whose sides are parallel to the coordinates and the short side lies in interval $(0, 1)$. Then there is no integer point in it, but its volume can be arbitrarily large as the long side stretches. Therefore, they focused on the distance between the count and the volume, and further proposed a method to approximate the count by the volume. The following theorems are their main results.

▶ **Lemma 3.** *Both* $\texttt{vol}(P), \texttt{lat}(P)$ *are in the interval* $[|\mathrm{C}(P)| - |\mathrm{C}(\mathrm{B}(P))|, |\mathrm{C}(P)|]$.

▶ **Theorem 4.** $|\texttt{vol}(P) - \texttt{lat}(P)| \leq |\mathrm{C}(B(P))|$.

▶ **Theorem 5.** $|\mathrm{C}(\mathrm{B}(P))| \leq 2\sum_{i=1}^{n}\prod_{i\neq j}(M_j(P) - m_j(P))$*, where* $M_i(P) = \lfloor \max\{x_i | \vec{x} \in P\} + 1\rfloor$ *and* $m_i(P) = \lceil \min\{x_i | \vec{x} \in P\} - 1\rceil$.

Take Figure 1 as the example. We observe that each gray square contains exactly one integer point in the polytope while orange squares maybe not, so it suggests that the differences between $\texttt{vol}(P)$, $\texttt{lat}(P)$ and $|\mathrm{C}(P)|$ are related to those orange squares, i.e., $|\mathrm{C}(\mathrm{B}(P))|$. It is the intuition behind the proof of Lemma 3. Based on Lemma 3, it is easy to obtain Theorem 4 that the difference between $\texttt{vol}(P)$ and $\texttt{lat}(P)$ is bounded by $|\mathrm{C}(\mathrm{B}(P))|$. Then by Theorem 5, a looser bound $2\sum_{i=1}^{n}\prod_{i\neq j}(M_j(P) - m_j(P))$ is proved which is easier to be computed in practice. As a result, the count of lattices in a given polytope P can be approximated via its volume $\texttt{vol}(P)$, i.e.,

$$|\texttt{vol}(P) - \texttt{lat}(P)| \leq 2\sum_{i=1}^{n}\prod_{i\neq j}(M_j(P) - m_j(P)).$$

However, according to experimental results in [11], the bound $2\sum_{i=1}^{n}\prod_{i\neq j}(M_j(P) - m_j(P))$ may be very loose, which prevents some applications. Naturally, we are interested in improving the above bounds. In this paper, we first propose a new method to approximate $|\mathrm{C}(\mathrm{B}(P))|$ which is usually much tighter in experiments. Then we extend Theorem 4 to mixed-integer cases to approximate `integral`(F) by $\texttt{vol}(P)$, where P is the corresponding polytope of F.

2.3 Sampling in Polytopes

The classical algorithm [4, 22, 20, 19, 8, 12] for sampling real points in a polytope P is presented in Algorithm 1. It first calls a rounding method, such as the Shallow-β-Cut Ellipsoid method [15], to find an affine transformation T, s.t., $\texttt{ball}(0, 1) \subset T(P) \subset \texttt{ball}(0, 2n)$, where $\texttt{ball}(0, r)$ is a radius r ball centered at origin. Then it employs a hit-and-run random walk method to generate real points in $T(P)$. It finally returns sample points in P by applying the inverse transformation T^{-1}. Intuitively, T transforms a very "thin" polytope into a well-bounded one. Thus, random walks will mix (converge to limiting distribution) faster on the new polytope $T(P)$. In addition, it guarantees that $T(P)$ contains the origin, which will be used as the start point for the random walks.

Algorithm 1 Real Points Sampling Algorithm

```
1 Function Sampling_Real($P$, $\vec{x_0}$, $w$)
2     $T \leftarrow$ Ellipsoid($P$);
3     $\vec{x} \leftarrow T(\vec{x_0})$;
4     while $w > 0$ do
5         $\vec{x} \leftarrow$ Hit-and-run($T(P)$, $\vec{x}$);
6         $w \leftarrow w - 1$;
7     return $T^{-1}(\vec{x})$;
```

The Hit-and-run random walk method was first introduced in [4], where its limiting distribution was proved to be uniform. It was employed and improved for approximating a polytope's volume by [22, 20]. A variation called Coordinate Directions Hit-and-run is found more efficient by experiments [8, 12]. Thus, we also adopt this variation in our paper, which consists of the following steps:

Step 1. Given a point $\vec{x_0} \in P$, it first selects a line L uniformly over n coordinate directions (parallel to the axes) which passes through the point $\vec{x_0}$.

Step 2. It then chooses the next point $\vec{x_1}$ uniformly on the segment of L in P.

Step 3. Repeat above steps w times, $\vec{x_w}$ is finally obtained and adopted.

Earlier works [20] proved that Hit-and-run method mixes in $w = O(n^2)$ steps for a random initial point and $O(n^3)$ steps for a fixed initial point. However, further numerical studies [19, 12] reported that $w = n$ is sufficient for nearly uniformly sampling in polytopes with dozens of dimensions.

3 Our Approach

In this section, we first introduce our new algorithms for approximating $\texttt{lat}(P)$ via $\texttt{vol}(P)$. Then we extend Theorem 4 and algorithms to mixed-integer cases, i.e., approximating $\texttt{integral}(F)$ via $\texttt{vol}(P)$.

3.1 The Framework of Bounds Approximation Algorithm

To compute bounds of $\texttt{lat}(P)$, i.e., $|\mathrm{C}(P)| - |\mathrm{C}(\mathrm{B}(P))|$ and $|\mathrm{C}(P)|$, we introduce a Monte-Carlo algorithm which samples points in $\texttt{union}(\mathrm{C}(P))$ and then counts the number of points that lie in P and $\texttt{union}(\mathrm{C}(\mathrm{B}(P)))$.

▶ **Theorem 6.** *Suppose X is a set of sample points uniformly generated from* $\texttt{union}(\mathrm{C}(P))$*. Let $\hat{r}_1 = \frac{|X \cap P|}{|X|}$ and $\hat{r}_2 = 1 - \frac{|X \cap \texttt{union}(\mathrm{C}(\mathrm{B}(P)))|}{|X|}$. Then $|\mathrm{C}(P)| - |\mathrm{C}(\mathrm{B}(P))| = \texttt{vol}(P) \cdot \lim_{|X|\to\infty} \frac{\hat{r}_2}{\hat{r}_1}$ and $|\mathrm{C}(P)| = \texttt{vol}(P) \cdot \lim_{|X|\to\infty} \frac{1}{\hat{r}_1}$.*

Proof. Let $r_1 = \frac{\texttt{vol}(P)}{|\mathrm{C}(P)|}$ and $r_2 = 1 - \frac{|\mathrm{C}(\mathrm{B}(P))|}{|\mathrm{C}(P)|}$. Since

$$\texttt{vol}(\texttt{union}(\mathrm{C}(P))) = \sum_{\kappa \in \mathrm{C}(P)} \texttt{vol}(\kappa) = |\mathrm{C}(P)|,$$

then $r_1 = \frac{\texttt{vol}(P)}{\texttt{vol}(\texttt{union}(\mathrm{C}(P)))}$. Note that sampling in $\texttt{union}(\mathrm{C}(P))$ in uniform and then counting the number of points that lie in P is a Bernoulli process. Thus $\hat{r}_1$ is the estimated proportion of successes for r_1. Therefore, $\lim_{|X|\to\infty} \hat{r}_1 = r_1$. Similarly, we could find that $\lim_{|X|\to\infty} \hat{r}_2 = r_2$, and $\lim_{|X|\to\infty} \frac{\hat{r}_2}{\hat{r}_1} = \frac{r_2}{r_1} = \frac{|\mathrm{C}(P)| - |\mathrm{C}(\mathrm{B}(P))|}{\texttt{vol}(P)}$. ◀

From Lemma 3 and Theorem 6, we know that $\frac{\hat{r}_2}{\hat{r}_1} \cdot \texttt{vol}(P)$ and $\frac{1}{\hat{r}_1} \cdot \texttt{vol}(P)$ are the approximations of the lower bound and the upper bound of $\texttt{lat}(P)$ respectively. Therefore, we aim to approximate $\frac{1}{\hat{r}_1}$ and $\frac{\hat{r}_2}{\hat{r}_1}$. Since sampling points is a Bernoulli trial, then $\hat{r}_1 = \frac{|X \cap P|}{|X|} \in [0, 1]$ is an approximation of proportion of a binomial distribution, and $\hat{r}_2$ as well. The confidence interval (CI) of r_1 is thus a binomial CI. The well-known $1 - \delta$ normal approximation CI on r_1 is

$$\hat{r}_1 - z_{1-\delta/2}\sqrt{\frac{\hat{r}_1(1-\hat{r}_1)}{|X|}} \le r_1 \le \hat{r}_1 + z_{1-\delta/2}\sqrt{\frac{\hat{r}_1(1-\hat{r}_1)}{|X|}}, \tag{4}$$

where $z_{1-\delta/2}$ is the $1 - \delta/2$ quantile of a standard normal distribution. Intuitively, CIs of r_1 and r_2 can be used as the algorithm's stopping criterion. Let $\hat{e}_1 \equiv z_{1-\delta/4}\sqrt{\frac{\hat{r}_1(1-\hat{r}_1)}{|X|}}$ and $\hat{e}_2 \equiv z_{1-\delta/4}\sqrt{\frac{\hat{r}_2(1-\hat{r}_2)}{|X|}}$ which are the margin errors. We obtain $1 - \delta/2$ CIs of r_1 and r_2: $\hat{r}_1 - \hat{e}_1 \le r_1 \le \hat{r}_1 + \hat{e}_1$ and $\hat{r}_2 - \hat{e}_2 \le r_2 \le \hat{r}_2 + \hat{e}_2$. Then $r_1 \ge \hat{r}_1 - \hat{e}_1$ and $r_2 \le \hat{r}_2 + \hat{e}_2$ with probability at least $1 - \delta/4$. Thus we have $\frac{\hat{r}_2 - \hat{e}_2}{\hat{r}_1 + \hat{e}_1} \le \frac{r_2}{r_1}$ with probability at least $1 - \delta/2$. By the same way, we obtain the intervals

$$\frac{\hat{r}_2 - \hat{e}_2}{\hat{r}_1 + \hat{e}_1} \le \frac{r_2}{r_1} \le \frac{\hat{r}_2 + \hat{e}_2}{\hat{r}_1 - \hat{e}_1} \text{ and } \frac{1}{\hat{r}_1 + \hat{e}_1} \le \frac{1}{r_1} \le \frac{1}{\hat{r}_1 - \hat{e}_1}, \tag{5}$$

so that $\frac{\hat{r}_2}{\hat{r}_1}$ and $\frac{1}{\hat{r}_1}$ lie in them with probability at least $1 - \delta$.

▶ **Theorem 7.** *In Theorem 6, if $|X| \ge z^2_{1-\delta/4} \cdot (\frac{1}{\epsilon} \cdot \sqrt{\frac{1-r_2}{r_2}} + \frac{1-\epsilon}{\epsilon} \cdot \sqrt{\frac{1-r_1}{r_1}})^2$ and $|X| \ge z^2_{1-\delta/4} \cdot (\frac{1+\epsilon}{\epsilon})^2 \cdot \frac{1-r_1}{r_1}$, then $\mathrm{Prob}(|\frac{\hat{r}_2}{\hat{r}_1} - \frac{r_2}{r_1}| \le \epsilon \cdot \frac{r_2}{r_1}) \ge 1 - \delta$ and $\mathrm{Prob}(|\frac{1}{\hat{r}_1} - \frac{1}{r_1}| \le \epsilon \cdot \frac{1}{r_1}) \ge 1 - \delta$.*

Proof. Note that $\frac{1}{\hat{r}_1 - \hat{e}_1} - \frac{1}{\hat{r}_1} \le \epsilon \cdot \frac{1}{\hat{r}_1} \iff \epsilon\hat{r}_1 \ge (1+\epsilon)\hat{e}_1 \iff |X| \ge z^2_{1-\delta/4} \cdot (\frac{1+\epsilon}{\epsilon})^2 \cdot \frac{1-\hat{r}_1}{\hat{r}_1} \approx z^2_{1-\delta/4} \cdot (\frac{1+\epsilon}{\epsilon})^2 \cdot \frac{1-r_1}{r_1}$. From Equation 5, we have $\mathrm{Prob}(|\frac{1}{\hat{r}_1} - \frac{1}{r_1}| \le \epsilon \cdot \frac{1}{r_1}) \ge \mathrm{Prob}(\frac{1}{\hat{r}_1 - \hat{e}_1} - \frac{1}{\hat{r}_1} \le \epsilon \cdot \frac{1}{\hat{r}_1}) \ge 1 - \delta$. The proof of $|X| \ge z^2_{1-\delta/4} \cdot (\frac{1}{\epsilon} \cdot \sqrt{\frac{1-r_2}{r_2}} + \frac{1-\epsilon}{\epsilon} \cdot \sqrt{\frac{1-r_1}{r_1}})^2$ is similar. ◀

Theorem 7 discusses the relations among the confidence δ, the error ϵ and the number of samples $|X|$. Since $\frac{r_2}{r_1} \in (0, 1]$ and $\frac{1}{r_1} \in [1, \infty)$, the scale of samples $|X|$ may vary a lot with respect to the values of $\frac{r_2}{r_1}$ and $\frac{1}{r_1}$. Therefore, based on the proof of Theorem 7, we introduce a dynamical stopping criterion for our algorithm which checks the quality of the approximation whenever a sample is obtained. The algorithm stops when it found $\frac{\hat{r}_2}{\hat{r}_1} - \frac{\hat{r}_2 - \hat{e}_2}{\hat{r}_1 + \hat{e}_1} \le \epsilon \cdot \frac{\hat{r}_2}{\hat{r}_1}$ and $\frac{1}{\hat{r}_1 - \hat{e}_1} - \frac{1}{\hat{r}_1} \le \frac{\epsilon}{\hat{r}_1}$, which guarantees $\mathrm{Prob}(|\frac{\hat{r}_2}{\hat{r}_1} - \frac{r_2}{r_1}| \le \epsilon \cdot \frac{r_2}{r_1}) \ge 1 - \delta$ and $\mathrm{Prob}(|\frac{1}{\hat{r}_1} - \frac{1}{r_1}| \le \epsilon \cdot \frac{1}{r_1}) \ge 1 - \delta$.

Algorithm 2 MixIntCount

```
1  Input: $P$
2  Parameter: $\epsilon$, $\delta$, $w$, $N$
3  Output: $\mathtt{lb}(P)$, $\mathtt{ub}(P)$
4  $X \leftarrow \emptyset$;
5  Initialize $\vec{x}$ with an arbitrary point in $P$;
6  while $|X| \leq N$ do
7      $\vec{x} \leftarrow$ Sampling($P$, $\vec{x}$, $w$);
8      $X \leftarrow X \cup \{\vec{x}\}$;
9      $\hat{r}_1 \leftarrow \frac{|X \cap P|}{|X|}$;
10     $\hat{r}_2 \leftarrow 1 - \frac{|X \cap \mathtt{union}(\mathrm{C}(\mathrm{B}(P)))|}{|X|}$;
11     if $\frac{\hat{r}_2}{\hat{r}_1} - \frac{\hat{r}_2 - \hat{e}_2}{\hat{r}_1 + \hat{e}_1} \leq \epsilon \cdot \frac{\hat{r}_2}{\hat{r}_1}$ and $\frac{1}{\hat{r}_1 - \hat{e}_1} - \frac{1}{\hat{r}_1} \leq \frac{\epsilon}{\hat{r}_1}$ then break;
12 return $\mathtt{vol}(P) \cdot \frac{\hat{r}_2 - \hat{e}_2}{\hat{r}_1 + \hat{e}_1}$, $\mathtt{vol}(P) \cdot \frac{1}{\hat{r}_1 - \hat{e}_1}$;
```

The pseudocode of the our main framework is presented in Algorithm 2. The parameters ϵ, δ and N determine the accuracy and the confidence of approximations and the maximum number of samples respectively. Note that according to Theorem 7, $|X|$ could be a quite large number when $\frac{r_2}{r_1}$ is close to 0 or $\frac{1}{r_1} \gg 1$. In such cases, the approximations of bounds are meaningless, for example, $\frac{\mathtt{lat}(P)}{\mathtt{vol}(P)} \geq \frac{r_2}{r_1} \approx 0$. So we introduce a sampling limit N. The bounds would be rather meaningless when the algorithm reachs the limit N. The setting of parameters will be further discussed in Section 4. In general, Algorithm 2 returns the bounds of $\mathtt{lat}(P)$, i.e., $\mathtt{lb}(P) \leq \mathtt{lat}(P) \leq \mathtt{ub}(P)$ and $|\mathtt{vol}(P) - \mathtt{lat}(P)| \leq \mathtt{ub}(P) - \mathtt{lb}(P)$ with probability at least $1 - \delta$.

3.2 Sampling in Unions of Integer-cubes

To generate sample points in $\mathtt{union}(\mathrm{C}(P))$ nearly uniformly, we combine Algorithm 1 with rejection sampling. Algorithm 3 presents the new sampling algorithm. It first enlarges P to obtain a new polytope P', such that P' contains all integer-cubes in $\mathrm{C}(P)$. Next it samples real points in P' by Algorithm 1. Then it accepts those in $\mathtt{union}(\mathrm{C}(P))$. Obviously, the larger P', the lower probability of acceptance. Now a question arises:

- How to obtain such a P' that is as small as possible?

Algorithm 3 Sampling in $\mathtt{union}(\mathrm{C}(P))$

```
1  Function Sampling($P$, $\vec{x_0}$, $w$)
2      $P' \leftarrow$ Enlarging($P$);
3      $\vec{x} \leftarrow \vec{x_0}$;
4      while $true$ do
5          $\vec{x} \leftarrow$ Sampling_Real($P'$, $\vec{x}$, $w$);
6          if $\vec{x} \in \mathtt{union}(\mathrm{C}(P))$ then
7              return $\vec{x}$;
```

Intuitively, we can obtain P' by shifting every facet H of P to H', s.t., the distance between H and H' is sufficient to contain an integer-cube. Minimizing this shifting distance is formulated into a linear programming (LP) problem, with constraints $\{-1 \leq x_i \leq 1\}$ and an objective, maximize $\vec{A_k}\vec{x}$, where $\vec{A_k}$ is the kth row of the matrix A. Let v_k be such maximum value for the kth LC. Then we shift it by adding v_k. The pseudocode of constructing P' is shown in Algorithm 4. The following theorem guarantees that P' obtained by Algorithm 4 contains $\texttt{union}(\mathrm{C}(P))$.

▶ **Theorem 8.** *Given the kth LC $H_k \equiv \vec{A_k}\vec{x} \leq b_k$. Let $v_k = \max\{\vec{A_k}\vec{x} | -1 \leq x_i \leq 1, i = 1, \dots, n\}$ and $H'_k \equiv \vec{A_k}\vec{x} \leq b_k + v_k$. Then we have $\kappa \subset H'_k, \forall \kappa \in \mathrm{C}(H_k)$.*

Proof. Let $G_k = \vec{A_k}\vec{x} \leq 0$ and $G'_k = \vec{A_k}\vec{x} \leq v_k$. Then it is equivalent to prove $\kappa \subset G'_k, \forall \kappa \in \mathrm{C}(G_k)$. Assume $\exists\texttt{cube}(\vec{\alpha}) \in C(G_k)$ and $\exists \vec{p_{out}} \in \texttt{cube}(\vec{\alpha})$ s.t., $\vec{p_{out}} \notin G'_k$. Let $c_{max} = \max\{\vec{A_k}\vec{x} | \vec{x} \in \texttt{cube}(\vec{\alpha})\}$ and $c_{min} = \min\{\vec{A_k}\vec{x} | \vec{x} \in \texttt{cube}(\vec{\alpha})\}$. Recall that $\texttt{cube}(\vec{\alpha}) = \{\alpha_i - \frac{1}{2} \leq x_i \leq \alpha_i + \frac{1}{2}\}$, we can find that $v_k = c_{max} - c_{min}$. Since $\texttt{cube}(\vec{\alpha}) \in C(G_k)$, then $\exists \vec{p_{in}} \in \texttt{cube}(\vec{\alpha}) \cap G_k$. Thus the hyperplane $\vec{A_k}\vec{x} = \vec{A_k}\vec{p_{out}}$ is outside of G'_k, and $\vec{A_k}\vec{p_{out}} > v_k$. Similarly, we could find that $\vec{A_k}\vec{p_{in}} \leq 0$, yielding the contradiction $v_k = c_{max} - c_{min} \geq \vec{A_k}\vec{p_{out}} - \vec{A_k}\vec{p_{in}} > v_k$. ◀

Algorithm 4 Enlarging P by shifting hyperplanes

```
1 Function Enlarging($P$)
2     for each $\vec{A_k}\vec{x} \leq b_k$ from $P$ do
3         constraints ← $\{-1 \leq x_i \leq 1\}$;
4         object ← $\vec{A_k}\vec{x}$;
5         $v_i$ ← Simplex(object, constraints);
6     return $\{A\vec{x} \leq \vec{b} + \vec{v}\}$;
```

3.3 Efficient Cube Checking

In Algorithm 2 and 3, we have to frequently check whether a point is in any integer-cube in $\mathrm{C}(P)$ or $\mathrm{C}(\mathrm{B}(P))$. So in this section, we focus on the following question.

- How to efficiently check whether a point $\vec{p}$ is in $\texttt{union}(\mathrm{C}(P))$ or $\texttt{union}(\mathrm{C}(\mathrm{B}(P)))$?

We observe that if p is not on the boundary of an integer-cube, then $\vec{p} \in \texttt{union}(\mathrm{C}(P))$ iff $\texttt{cube}([\vec{p}]) \in \mathrm{C}(P)$, where $[\vec{p}] = ([p_1], \dots, [p_n])$ is obtained by rounding numbers to integers. Since in practice, it is nearly impossible to generate a sample point right on the boundaries of cubes. We assume p does not sit on the boundary of an integer-cube in this section. Similarly, we have if p is not on the boundary of an integer-cube, $\vec{p} \in \texttt{union}(\mathrm{C}(\mathrm{B}(P)))$ iff $\texttt{cube}([\vec{p}]) \in \mathrm{C}(\mathrm{B}(P))$. Furthermore, $\texttt{cube}([\vec{p}]) \in \mathrm{C}(P)$ iff $\vec{p} \in P$ or $\texttt{cube}([\vec{p}]) \in \mathrm{C}(\mathrm{B}(P))$. Since checking whether $\vec{p} \in P$ is trivial, we only have to find an efficient method to check whether $\texttt{cube}(\vec{\alpha}) \in \mathrm{C}(\mathrm{B}(P))$ with a given integer point $\vec{\alpha}$.

Algorithm 5 presents the method for fast cube checking. Note that v_k is the same as in Algorithm 4, that is, $v_k = \max\{\vec{A_k}\vec{x} | -1 \leq x_i \leq 1, i = 1, \dots, n\}$, where $\vec{A_k}$ is the kth row of A. Obviously, $v_k/2 = \max\{\vec{A_k}\vec{x} | -0.5 \leq x_i \leq 0.5, i = 1, \dots, n\}$. Since $\vec{\alpha}$ is the center of $\texttt{cube}(\vec{\alpha})$, we have $\vec{A_k}\vec{\alpha} - v_k/2 = \min\{\vec{A_k}\vec{x} | \vec{x} \in \texttt{cube}(\vec{\alpha})\}$ and $\vec{A_k}\vec{\alpha} + v_k/2 = \max\{\vec{A_k}\vec{x} | \vec{x} \in \texttt{cube}(\vec{\alpha})\}$. If $\exists j$, s.t., $\vec{A_j}\vec{\alpha} - v_j/2 > b_j$, it indicates that $\texttt{cube}(\vec{\alpha})$ is completely outside P. Otherwise, if $\exists j$, s.t., $\vec{A_j}\vec{\alpha} + v_j/2 \geq b_j$, it indicates that $\texttt{cube}(\vec{\alpha})$ intersects with $\mathrm{B}(P)$.

Algorithm 5 Check whether $\texttt{cube}(\vec{\alpha}) \in \mathrm{C}(\mathrm{B}(P))$

```
1 Function CubeOnBound($\vec{\alpha}, P$)
2     flag ← $false$;
3     for each $\vec{A_k}\vec{x} \le b_k$ from $P$ do
4         if $\vec{A_k}\vec{\alpha} - v_k/2 > b_k$ then
5             return $false$;
6         if $\vec{A_k}\vec{\alpha} + v_k/2 \ge b_k$ then
7             flag ← $true$;
8     return flag;
```

3.4 Extending to Mixed-Integer Cases

In this section, we will introduce and prove the theoretical result on mixed-integer cases. Combined with Lemma 3, it not only provides bounds for $\texttt{integral}(F)$, but also shows when $\texttt{integral}(F)$ can be approximated by $\texttt{lat}(P)$ and $\texttt{vol}(P)$.

▶ **Theorem 9.** $|\mathrm{C}(P)| - |\mathrm{C}(\mathrm{B}(P))| \le \texttt{integral}(F) \le |\mathrm{C}(P)|$.

Proof. According to Equation (3), the theorem is equivalent with

$$|\mathrm{C}(P)| - |\mathrm{C}(\mathrm{B}(P))| \le \sum_{\vec{\alpha_I} \in \mathcal{M}_{\mathcal{I}}(F)} \texttt{vol}(P_{\vec{\alpha_I}}) \le |\mathrm{C}(P)|. \tag{6}$$

From Lemma 3, we have $|\mathrm{C}(P_{\vec{\alpha_I}})| - |\mathrm{C}(\mathrm{B}(P_{\vec{\alpha_I}}))| \le \texttt{vol}(P_{\vec{\alpha_I}}) \le |\mathrm{C}(P_{\vec{\alpha_I}})|$. Then

$$\sum_{\vec{\alpha_I}}(|\mathrm{C}(P_{\vec{\alpha_I}})| - |\mathrm{C}(\mathrm{B}(P_{\vec{\alpha_I}}))|) \le \sum_{\vec{\alpha_I}} \texttt{vol}(P_{\vec{\alpha_I}}) \le \sum_{\vec{\alpha_I}} |\mathrm{C}(P_{\vec{\alpha_I}})|. \tag{7}$$

Given an arbitrary $\texttt{cube}(\vec{\alpha_R}) \in \mathrm{C}(P_{\vec{\alpha_I}})$. Obviously, $\exists \vec{\alpha_R}' \in \texttt{cube}(\vec{\alpha_R})$ such that $\vec{\alpha_R}' \in P_{\vec{\alpha_I}}$. Let $\vec{\alpha} = [\vec{\alpha_I}\vec{\alpha_R}]$ and $\vec{\alpha}' = [\vec{\alpha_I}\vec{\alpha_R}']$ which are concatenations of integer and real variables. Then $\vec{\alpha}'$ is an interior point in P and $\vec{\alpha}' \in \texttt{cube}(\vec{\alpha})$. It means $\texttt{cube}(\vec{\alpha}) \in \mathrm{C}(P)$. By this way, we could map $\mathrm{C}(P_{\vec{\alpha_I}})$ to $\mathrm{C}(P \cap \vec{\alpha_I} \times \mathbb{R}^{n_R}) \subset \mathrm{C}(P)$. Note that $\mathrm{C}(P \cap \vec{\alpha_I} \times \mathbb{R}^{n_R}) \cap \mathrm{C}(P \cap \vec{\alpha_I'} \times \mathbb{R}^{n_R}) = \emptyset$, and $\vec{\alpha_I} \ne \vec{\alpha_I'}$, then we have

$$\sum_{\vec{\alpha_I}} |\mathrm{C}(P_{\vec{\alpha_I}})| \le |\mathrm{C}(P)|. \tag{8}$$

Given an arbitrary $\texttt{cube}(\vec{\beta}) \in \mathrm{C}(P) \setminus \mathrm{C}(\mathrm{B}(P))$. Let $\vec{\beta} = [\vec{\beta_I}\vec{\beta_R}]$. Note that $\texttt{cube}(\vec{\beta}) \subset P$, it means $\texttt{cube}(\vec{\beta}) \cap \mathcal{M}(F) \ne \emptyset$. Thus $\vec{\beta_I} \in \mathcal{M}_{\mathcal{I}}(F)$. Then we obtain an unique integer-cube $\texttt{cube}(\vec{\beta_R}) \in \mathrm{C}(P_{\vec{\beta_I}}) \setminus \mathrm{C}(\mathrm{B}(P_{\vec{\beta_I}}))$. Similarly, we have

$$|\mathrm{C}(P)| - |\mathrm{C}(\mathrm{B}(P))| = |\mathrm{C}(P) \setminus \mathrm{C}(\mathrm{B}(P))| \le \sum_{\vec{\alpha_I}}(|\mathrm{C}(P_{\vec{\alpha_I}})| - |\mathrm{C}(\mathrm{B}(P_{\vec{\alpha_I}}))|). \tag{9}$$

Combine Equation (7) (8) and (9), then Equation (6) is obtained. ◀

Theorem 9 indicates that Algorithm 2 can be directly applied for solving the solution space integration problem on mixed-integer constraints, i.e., $\texttt{lb}(P) \le \texttt{integral}(F) \le \texttt{ub}(P)$.

4 Implementation

We implemented a prototype tool called MixIntCount in C++. MixIntCount employs GLPK for linear programming. It calls Vinci [5] and Polyvest [10] for polytopes' volume computation and approximation.

In our implementation, the Ellipsoid method is employed only once, and the affine transformation T will be reused by the real point sampling in Algorithm 3. Moreover, whenever a sample is obtained, it will immediately update the counts of samples that lie in P or $\texttt{union}(\mathrm{C}(\mathrm{B}(P)))$ and check whether the stopping criterion is satisfied.

Wilson score interval.

For convenience, we introduced the normal approximation CI implied by the Central Limit Theorem for describing Algorithm 2 in Section 3. However, the normal CI suffers from problems of overshoot and zero-width intervals, e.g., r_1 and r_2 may sometimes be close to 0 or 1. In such circumstances, the Wilson score interval (or Wilson CI) [31] performs much better than the normal CI. Therefore, we replace Equation 4 by Wilson CI, and let MixIntCount compute Wilson CIs as the stopping criterion.

The setting of parameters.

We choose parameters $\epsilon = 0.1$, $\delta = 0.05$, $w = n$, and $N = 100000$ for MixIntCount, and $\epsilon = 0.2, \delta = 0.05$ for Polyvest. Note that ϵ and δ are different in two tools. In PolyVest, ϵ and δ control the errors of volume approximations. In MixIntCount, ϵ and δ control the errors of lower and upper bounds. Since previous numerical studies [19, 12] reported that $w = n$ is sufficient for nearly uniformly sampling points in polytopes with dozens of dimensions, we also chose $w = n$ in our implementation. Experiments in Section 5 show that $N = 100000$ is sufficiently large, as the lower bounds $\texttt{lb}(P) \leq 0.001 \cdot \texttt{vol}(P)$ are useless when $|X|$ reaches N.

A straightforward method for mixed-integer cases.

Based on the definition, i.e., Equation 2, it is easy to propose an algorithm for computing $\texttt{integral}(F)$. First, it enumerates all assignments $\vec{\alpha_I} \in \mathcal{M}_{\mathcal{I}}(F)$. Then for each $\vec{\alpha_I}$, it computes $\texttt{vol}(P_{\vec{\alpha_I}})$ by volume computation algorithms. Finally, $\texttt{integral}(F)$ is obtained by summing up $\texttt{vol}(P_{\vec{\alpha_I}})$. We implemented this method and called it ExactMI, which could provide exact integration. We adopted it as the baseline for performance comparison in our evaluation (Section 5). Note that in practice, we could try MixIntCount first, and if approximations of bounds are not tight enough, we would then employ ExactMI.

Approximating bounds on SMT(LA) formulas.

We incorporated MixIntCount into the DPLL(T)-based #SMT(LA) counter [12] directly to approximate bounds of solution space integration of SMT(LA) formulas. Without loss of generality, an SMT(LA) formula ϕ with l Boolean variables, n numeric variables and m LCs can be formally represented as a Boolean formula $PS_\phi(b_1, \ldots, b_{m+l})$ together with definitions in the form: $b_i \equiv H_i, i = 1, \ldots, m$, where H_is are LCs. Then $b_{m+1}, \ldots, b_{m+l}$ are the pure Boolean variables of ϕ. An assignment $\vec{\gamma}$ of PS_ϕ is a vector $(\gamma_1, \ldots, \gamma_{m+l}) \in \mathbb{B}^{m+l}$, where γ_i is either 1 or 0. A partial assignment $\vec{\gamma}$ means there are some γ_is not assigned. Let $\texttt{bool}(\vec{\gamma})$ represent the vector $(\gamma_{m+1}, \ldots, \gamma_{m+l}) \in \mathbb{B}^l$ which corresponds to those pure Boolean variables. Let $H_{\vec{\gamma}} = \bigcup_{1 \le i \le m} H_{\vec{\gamma},i}$, where $H_{\vec{\gamma},i}$ is $\{H_i\}$ or $\{\neg H_i\}$ or $\emptyset$ if γ_i is 1 or 0 or not assigned respectively. Note that H_i and $\neg H_i$ are LCs, $H_{\vec{\gamma}}$ is the set of LCs that corresponds to $\vec{\gamma}$. Thus, an assignment $\vec{\mu}$ of ϕ consists of $(\vec{x}, \texttt{bool}(\vec{\gamma}))$, where $\vec{\gamma}$ is an assignment of PS_ϕ and $\vec{x}$ is a point. Let $\mathcal{M}_{\phi,\vec{\gamma}}$ represent $\mathcal{M}(H_{\vec{\gamma}}) \times \texttt{bool}(\vec{\gamma})$. The solution space of ϕ is then the union of sets, formally:

$$\mathcal{M}(\phi) = \bigcup_{\vec{\gamma} \in \mathcal{M}(PS_\phi)} \mathcal{M}(H_{\vec{\gamma}}) \times \texttt{bool}(\vec{\gamma}) = \bigcup_{\vec{\gamma} \in \mathcal{M}(PS_\phi)} \mathcal{M}_{\phi,\vec{\gamma}}. \tag{10}$$

To enumerate $\vec{\gamma} \in \mathcal{M}(PS_\phi)$, a DPLL(T)-based scheme was introduced:

- Step 1. Find a model $\vec{\mu}$ of ϕ by DPLL(T) algorithm. From Equation 10, there exists a partial assignment $\vec{\gamma} \in \mathcal{M}(PS_\phi)$, s.t., $\vec{\mu} \in \mathcal{M}_{\phi,\vec{\gamma}}$.
- Step 2. Conjunct ϕ with the negation formula G of partial assignment $\vec{\gamma}$, which would prevent the DPLL(T) algorithm finding models in $\mathcal{M}_{\phi,\vec{\gamma}}$ again. In detail, $G = \bigvee G_i$, where $G_i \equiv b_i$ if $\gamma_i = 0$, $G_i \equiv \neg b_i$ if $\gamma_i = 1$.
- Step 3. Find the next model $\vec{\mu}' \in \mathcal{M}(\phi')$ and a partial assignment $\vec{\gamma}'$, s.t., $\vec{\mu}' \in \mathcal{M}_{\phi',\vec{\gamma}'}$ like Step 1. Repeat above steps until $\mathcal{M}(\phi') = \emptyset$, i.e., unsatisfiable.

In this way, we could find a set $\Gamma = \{\vec{\gamma}, \vec{\gamma}', \dots\} \subset \mathcal{M}(PS_\phi)$. The above scheme guarantees

$$\mathcal{M}(\phi) = \bigcup_{\vec{\gamma} \in \Gamma} \mathcal{M}_{\phi,\vec{\gamma}} \text{ and } \mathcal{M}_{\phi,\vec{\gamma}_1} \cap \mathcal{M}_{\phi,\vec{\gamma}_2} = \emptyset, \forall \vec{\gamma}_1, \vec{\gamma}_2 \in \Gamma, \vec{\gamma}_1 \neq \vec{\gamma}_2. \tag{11}$$

From Equation 11, we know that $\mathcal{M}_{\phi,\vec{\gamma}}$s are non-overlapping, then

$$\texttt{integral}(\phi) = \sum_{\vec{\gamma} \in \Gamma} \texttt{integral}(\mathcal{M}_{\phi,\vec{\gamma}}) = \sum_{\vec{\gamma} \in \Gamma} \texttt{integral}(H_{\vec{\gamma}}) \cdot 2^{d_{\vec{\gamma}}}, \tag{12}$$

where $d_{\vec{\gamma}}$ is the number of γ_is which are not assigned, $m + 1 \le i \le l$. From Algorithm 2 and Theorem 9, we could approximate bounds for each $H_{\vec{\gamma}}$, i.e., $\texttt{lb}(H_{\vec{\gamma}}) \le \texttt{integral}(H_{\vec{\gamma}}) \le \texttt{ub}(H_{\vec{\gamma}})$. By summing up, we obtain the total bounds for ϕ:

$$\texttt{lb}(\phi) \le \texttt{integral}(\phi) = \sum_{\vec{\gamma} \in \Gamma} \texttt{integral}(H_{\vec{\gamma}}) \cdot 2^{d_{\vec{\gamma}}} \le \texttt{ub}(\phi), \tag{13}$$

where $\texttt{lb}(\phi) = \sum_{\vec{\gamma} \in \Gamma} \texttt{lb}(H_{\vec{\gamma}}) \cdot 2^{d_{\vec{\gamma}}}$ and $\texttt{ub}(\phi) = \sum_{\vec{\gamma} \in \Gamma} \texttt{ub}(H_{\vec{\gamma}}) \cdot 2^{d_{\vec{\gamma}}}$.

5 Evaluation

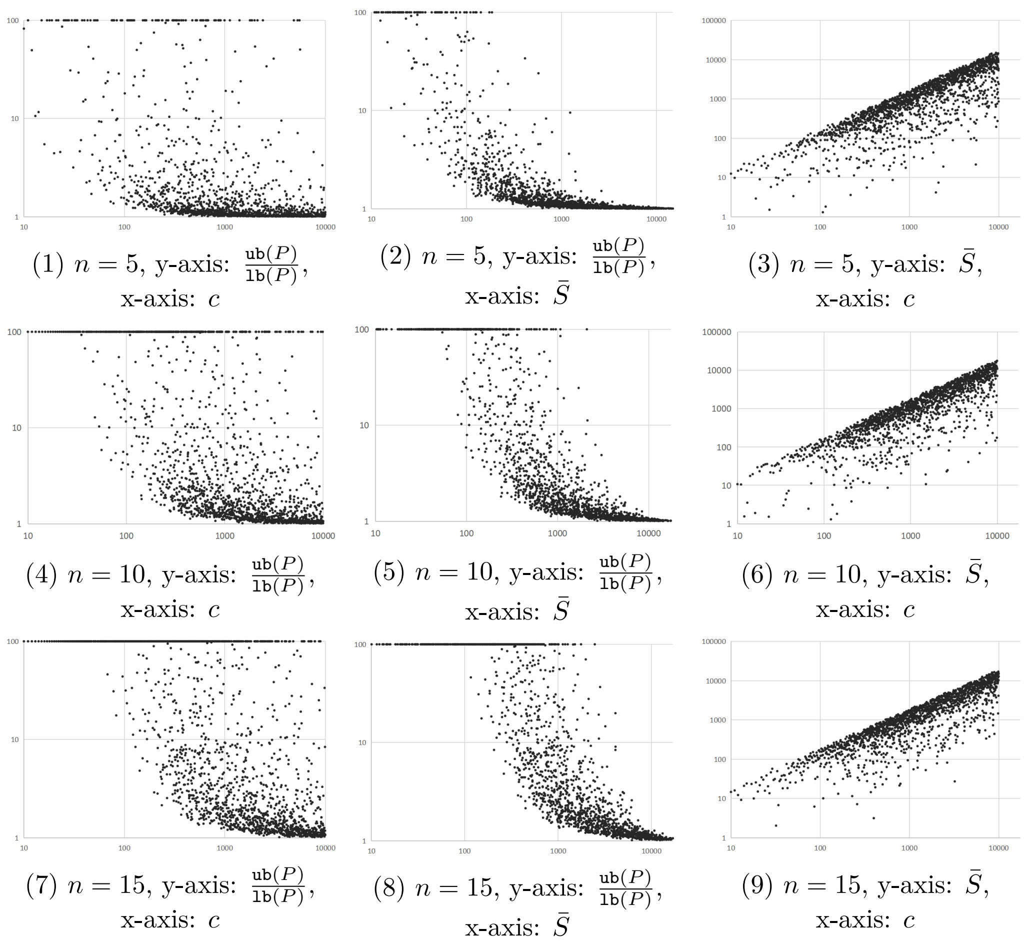
5.1 Experimental settings

Experiments were conducted on a cloud with 48 Core Intel(R) Xeon(R) Gold 6248 CPU @ 2.50GHz and 128GB memory. We used a timeout of 3600 seconds and a memory limit of 16GB. The suite of benchmarks consists of two families:

- **Random Polytopes** $P(m, n, n_I, c)$, where m, n, n_I and c are the number of LCs, the number of all variables, the number of integer variables and the range size of both x_i and b_i, respectively. In detail, a polytope $P(m, n, n_I, c) = \{A\vec{x} \le \vec{b}\}$ is generated by the following steps: (1) randomly choose n_I different integer variables, (2) randomly select $a_{ij} \in A$ from -10 to 10 and b_i from $-c$ to c, (3) if P is unsatisfiable then repeat above steps. In our benchmarks, we chose $m = n \in [3, 15]$, $n_I \in [1, n]$ and $c \in [10, 10000]$.
- **Instances from program analysis:** We adopted the application benchmarks introduced by [11] which are generated by analyzing 7 programs ('cubature', 'gjk', 'http-parser', 'muFFT', 'SimpleXML', 'tcas' and 'timeout') ranging from 0.4k to 7.7k lines of source code via a symbolic execution bug-finding tool. There are 3803 SMT(LIA) (linear integer arithmetic) formulas in total.

Benchmarks			ExactMI		Vinci		PolyVest		Bounds by MixIntCount			
n	n_I	$\bar{S}$	integral(F)	t (s)	vol(P)	t (s)	$\hat{\mathtt{vol}}(P)$	t (s)	$\frac{\mathtt{ub}(P)}{\mathtt{vol}(P)}$	$\frac{\mathtt{lb}(P)}{\mathtt{vol}(P)}$	$\lvert X\rvert$	t (s)
4	1	948	1.44E+10	4.99	1.43E+10	0.007	1.45E+10	0.072	1.055	0.914	281	0.004
	2	874	6.07E+08	508	6.07E+08	0.006	6.15E+08	0.074	1.172	0.820	1177	0.006
	3	1099	—	—	1.47E+11	0.006	1.52E+11	0.067	1.037	0.919	156	0.003
5	1	369	1.64E+09	1.89	1.64E+09	0.012	1.66E+09	0.257	1.479	0.635	2791	0.011
	2	945	—	—	8.70E+12	0.019	8.59E+12	0.256	1.050	0.935	189	0.005
	3	1086	—	—	6.13E+13	0.018	6.17E+13	0.249	1.062	0.923	291	0.005
6	1	715	5.55E+09	18.3	5.55E+09	0.053	5.48E+09	0.755	2.653	0.281	5243	0.023
	2	1438	—	—	1.34E+17	0.067	1.39E+17	0.780	1.034	0.925	113	0.005
	3	229	—	—	5.91E+09	0.063	5.92E+09	0.751	1.795	0.464	3824	0.019
7	1	1101	2.47E+17	113	2.47E+17	0.659	2.50E+17	1.90	1.125	0.848	810	0.008
	2	980	—	—	2.41E+17	0.655	2.44E+17	2.05	1.136	0.867	855	0.010
	3	727	—	—	6.19E+15	0.658	6.13E+15	2.04	1.171	0.813	1180	0.011
8	1	703	6.57E+14	615	6.57E+14	7.64	6.44E+14	4.56	2.036	0.415	4643	0.035
	2	1251	—	—	6.83E+20	7.91	6.83E+20	4.17	1.099	0.873	628	0.009
	3	556	—	—	5.71E+15	8.24	5.68E+15	4.62	1.695	0.527	3706	0.030
9	1	475	—	—	—	—	2.37E+13	9.05	7.560	0.064	49295	0.390
	2	1250	—	—	—	—	3.82E+22	8.46	1.152	0.836	1016	0.015
	3	138	—	—	—	—	1.47E+10	9.71	34.50	0.000	100000	0.792
12	1	252	—	—	—	—	2.56E+15	57.4	43.24	0.001	100000	1.517
	2	439	—	—	—	—	8.18E+19	54.9	7.063	0.070	45424	0.700
	3	1408	—	—	—	—	1.43E+32	52.3	1.123	0.845	842	0.025
15	1	958	—	—	—	—	5.85E+32	235	2.120	0.408	5102	0.157
	2	1598	—	—	—	—	5.10E+38	228	1.292	0.766	1950	0.076
	3	1587	—	—	—	—	4.20E+36	224	1.469	0.635	2902	0.105

Table 1 Running times and approximation results over random polytopes with $c = 1000$.



(1) $n = 5$, y-axis: $\frac{\texttt{ub}(P)}{\texttt{lb}(P)}$, x-axis: c

(2) $n = 5$, y-axis: $\frac{\texttt{ub}(P)}{\texttt{lb}(P)}$, x-axis: $\bar{S}$

(3) $n = 5$, y-axis: $\bar{S}$, x-axis: c

(4) $n = 10$, y-axis: $\frac{\texttt{ub}(P)}{\texttt{lb}(P)}$, x-axis: c

(5) $n = 10$, y-axis: $\frac{\texttt{ub}(P)}{\texttt{lb}(P)}$, x-axis: $\bar{S}$

(6) $n = 10$, y-axis: $\bar{S}$, x-axis: c

(7) $n = 15$, y-axis: $\frac{\texttt{ub}(P)}{\texttt{lb}(P)}$, x-axis: c

(8) $n = 15$, y-axis: $\frac{\texttt{ub}(P)}{\texttt{lb}(P)}$, x-axis: $\bar{S}$

(9) $n = 15$, y-axis: $\bar{S}$, x-axis: c

Figure 2 Experimental results about tightness of bounds $\frac{\texttt{ub}(P)}{\texttt{lb}(P)}$ with different c, $\bar{S}$ and n.

5.2 Experimental results

Table 1 presents experimental results on random polytopes with different values of n and n_I but fixed $c = 1000$. Due to page limit, we could only provide partial results here. In Table 1, $\bar{S} = \frac{1}{n}\sum_{i=1}^{n}(max\{x_i|\vec{x} \in P\} - min\{x_i|\vec{x} \in P\})$ is the average range size of variables in real domain, $|X|$ is the number of sample points generated. ExactMI is the implementation of the straightforward method presented in Section 4, which is the baseline. Vinci and PolyVest are tools for computing or approximating $\texttt{vol}(P)$. MixIntCount approximates the bounds of differences of $\texttt{integral}(F)$ and $\texttt{vol}(P)$. So the closer the bounds $\texttt{lb}(P)/\texttt{vol}(P)$ and $\texttt{ub}(P)/\texttt{vol}(P)$ are to 1, the better.

Table 1 shows that $\texttt{vol}(P)$ is usually very close to the exact $\texttt{integral}(F)$. We observe that ExactMI can only handle instances with only one or two integer variables because the number of integer assignments generated by ExactMI grows exponentially with respect to n_I. The scalability of ExactMI is also limited by Vinci, which runs out of memory in a few seconds when $n \geq 9$. PolyVest is more scalable as it is a polynomial time randomized approximation algorithm. Our tool MixIntCount generates approximate bounds $\texttt{lb}(P) \leq \texttt{integral}(F) \leq \texttt{ub}(P)$. The experimental results show that bounds are mostly useful while the overhead of approximating bounds is negligible compared to the cost of volume computation or approximation. There are two exceptions when $n = 9, n_I = 3$ and

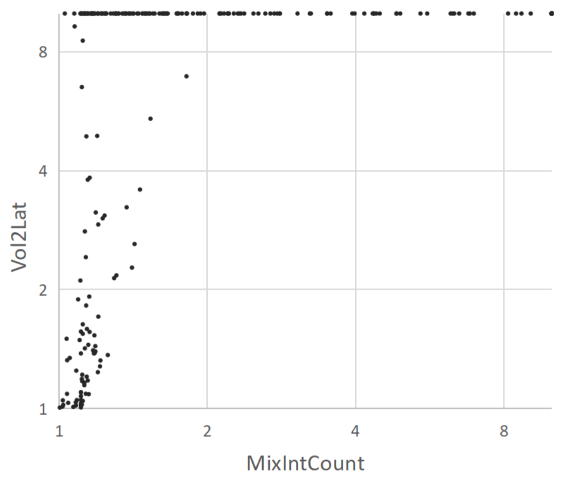


Figure 3 Comparison about the tightness of bounds $\frac{\texttt{ub}(P)}{\texttt{lb}(P)}$ (the smaller the better) between MixIntCount and Vol2Lat on random polytopes with pure-integer variables.

$n = 12, n_I = 1$ in Table 1. In these cases, the algorithm does not stop before reaching the maximum number of sample points, i.e., $|X| = N$, and bounds are also in essence meaningless. Besides, we observe that bounds are sometimes loose while $\texttt{integral}(F)$ and $\texttt{vol}(P)$ are very close, such as, when $n = 6, n_I = 1$ and $n = 8, n_I = 1$. From Table 1, we observe that the tightness of bounds by MixIntCount is related to the size of $\bar{S}$. In addition, the parameter c, which controls the domain size of variables, is fixed to 1000. So we conducted more experiments with different values of c.

Figure 2 presents results about tightness of bounds $\frac{\texttt{ub}(P)}{\texttt{lb}(P)}$ with different settings of c and n. From Figure 2 (3) (6) (9), we observe that $\bar{S}$ is highly correlated with c. Comparing Figure 2 (1) (4) (7) and Figure 2 (2) (5) (8), we find that tightness of bounds is more correlated with $\bar{S}$ than c. Besides, we also find that tightness of bounds is negatively correlated with n by comparing each row.

We compared our tool MixIntCount with Vol2Lat on pure integer instances which can be viewed as the special cases of mixed-integers. Tool Vol2Lat [11] is an approximate integer solution counter via a polytope's volume over pure integer constraints. We conducted experiments on random polytopes where their coefficients are generated in the same way as the mixed-integer cases. The results are presented in Figure 3, whose x-axis and y-axis are the tightness of bounds $\frac{\texttt{ub}(P)}{\texttt{lb}(P)}$ by two tools. Note that we force $\frac{\texttt{ub}(P)}{\texttt{lb}(P)} = 10$ when $\frac{\texttt{ub}(P)}{\texttt{lb}(P)} > 10$, thus the line of points at the top of the figure are cases that $\frac{\texttt{ub}(P)}{\texttt{lb}(P)} > 10$ for Vol2Lat.

Then we conducted experiments on application benchmarks from program analysis which are #SMT(LIA) problems. Note that both MixIntCount and Vol2Lat were incorporated into the DPLL (T)-based #SMT (LA) counter [12]. The results are presented in Table 2. Note that timeout cases by exact counter barvinok (cannot evaluate bounds without exact results) and degenerated cases (volume is zero) are excluded, so there are 3682 instances remaining. In Table 2, $\bar{e}$, $\bar{e}_l$ and $\bar{e}_u$ represent the average values of relative errors $e = \frac{|\texttt{lat}(F) - \texttt{vol}(F)|}{\texttt{lat}(F)}$, $e_l = \frac{\texttt{lb}(F)}{\texttt{lat}(F)}$ and $e_u = \frac{\texttt{ub}(F)}{\texttt{lat}(F)}$. Recall that we chose $\epsilon = 0.1$ for MixIntCount in default, so e_l and e_u will be at most 95% and at least 105%. With a smaller $\epsilon = 0.01$, MixIntCount provides a bit tighter bounds as presented in Table 2. In general, our approach provides much tighter upper bounds than Vol2Lat and consumes more time in exchange.

Benchmarks	Vol2Lat			MIC ($\epsilon = 0.1$)			MIC ($\epsilon = 0.01$)		
Name	$\bar{t}$ (s)	$\bar{e}_l$	$\bar{e}_u$	$\bar{t}$ (s)	$\bar{e}_l$	$\bar{e}_u$	$\bar{t}$ (s)	$\bar{e}_l$	$\bar{e}_u$
cubature	1.65	14.9%	347.4%	6.35	13.8%	215.4%	29.9	14.4%	207.6%
gjk	0.17	<0.1%	806.9%	1.65	<0.1%	389.1%	6.49	<0.1%	375.1%
httpparser	0.44	90.5%	117.5%	1.42	87.8%	111.2%	5.14	90.2%	108.6%
muFFT	0.07	45.6%	500.6%	0.71	43.6%	246.8%	1.65	45.4%	239.2%
SimpleXML	0.21	93.8%	120.3%	0.58	90.6%	109.6%	0.95	94.0%	105.9%
tcas	0.60	>99.9%	<100.1%	0.60	95.4%	104.9%	0.59	99.5%	100.5%
timeout	9.45	97.4%	109.3%	14.0	92.8%	108.0%	19.4	97.3%	103.5%

Table 2 Comparison about errors of bounds (the close to 100% the better) between MixIntCount (MIC) and Vol2Lat on pure integer instances from application instances.

6 Related Works

Ge et al. [11] studied the relation between the count of integer points inside a polytope and a polytope's volume. Then they proposed an algorithm to approximate the upper and lower bounds of integer solution counts via volume. They focused on the pure integer linear constraints. In this paper, we propose new algorithms for bounds approximation via a polytope's volume. The experimental results show that our approach provides tighter bounds on pure-integer cases. In addition, we extend their results and our algorithms to mixed-integer cases using Theorem 9.

The existing works on solving weighted model integration (WMI) problems [3, 17, 25, 26, 28] were focusing on formulas with Boolean and real variables, e.g., in SMT(LRA) (linear real arithmetic) representation. In this paper, we consider the solution space integration problem over mixed-integer linear constraints, which is a theory atom of SMT(LIRA) (linear mixed integer real arithmetic).

Ma et al. [24] first studied #SMT problems and proposed a DPLL(T)-based #SMT framework. Chistikov et al. [7] first extended hashing-based model counting techniques from #SAT to #SMT problems directly. Soon, Chakraborty et al. [6] proposed word-level hash functions for #SMT(BV) (bit-vector) and showed that their approach outperforms Chistikov's method through experiments. After that, Ge et al. [13] compared more recent hashing-based counters with exact integer counters in the scope of linear integer constraints. They observed that exact integer counters combined with the DPLL(T)-based #SMT framework are still more efficient for #LIA and #SMT(LIA) (linear integer arithmetic) problems. So in this paper, we also incorporate our algorithms into DPLL(T)-based #SMT(LA) (linear arithmetic) counters for approximating bounds of solution space integrations of #SMT(LA) formulas.

7 Conclusion

In this paper, we proposed an approximate solution space integration algorithm via a polytope's volume. It is based on theoretical analysis of bounds for solution space integration, volume and integer count. The upper and lower bounds provided by our approach are useful for users to decide whether to trust the approximations. We evaluated our approach's scalability and the tightness of bounds by experiments. Extending the bounds to weighted solution counting problems would be an interesting and challenging direction for future research.

References

1 Alexander I. Barvinok. Computing the volume, counting integral points, and exponential sums. *Discrete & Computational Geometry*, 10:123–141, 1993. doi:10.1007/BF02573970.

2 Alexander I. Barvinok. Computing the ehrhart polynomial of a convex lattice polytope. *Discrete & Computational Geometry*, 12:35–48, 1994. doi:10.1007/BF02574364.

3 V. Belle, A. Passerini, and Guy Van den Broeck. Probabilistic inference in hybrid domains by weighted model integration. In Qiang Yang and Michael J. Wooldridge, editors, *Proc. of IJCAI 2015*, pages 2770–2776. AAAI Press, 2015. URL: http://ijcai.org/Abstract/15/392.

4 H. C. P. Berbee, C. G. E. Boender, A. H. G. Rinnooy Kan, C. L. Scheffer, R. L. Smith, and J. Telgen. Hit-and-run algorithms for the identification of nonredundant linear inequalities. *Math. Program.*, 37(2):184–207, 1987. URL: http://dx.doi.org/10.1007/BF02591694, doi:10.1007/BF02591694.

5 B. Büeler, A. Enge, and K. Fukuda. *Exact Volume Computation for Polytopes: A Practical Study*, pages 131–154. 2000. URL: http://dx.doi.org/10.1007/978-3-0348-8438-9_6, doi:10.1007/978-3-0348-8438-9_6.

6 Supratik Chakraborty, Kuldeep S. Meel, Rakesh Mistry, and Moshe Y. Vardi. Approximate probabilistic inference via word-level counting. In *Proc. of AAAI*, pages 3218–3224, 2016.

7 Dmitry Chistikov, Rayna Dimitrova, and Rupak Majumdar. Approximate counting in SMT and value estimation for probabilistic programs. In *Proc. of TACAS*, pages 320–334, 2015.

8 Ben Cousins and Santosh S. Vempala. Gaussian cooling and o*(n3) algorithms for volume and gaussian volume. *SIAM J. Comput.*, 47(3):1237–1273, 2018. doi:10.1137/15M1054250.

9 M. E. Dyer, A. M. Frieze, and R. Kannan. A random polynomial time algorithm for approximating the volume of convex bodies. In *Proc. of STOC*, pages 375–381, 1989. URL: http://doi.acm.org/10.1145/73007.73043, doi:10.1145/73007.73043.

10 C. Ge and F. Ma. A fast and practical method to estimate volumes of convex polytopes. In Jianxin Wang and Chee-Keng Yap, editors, *Proc. of FAW*, volume 9130 of *Lecture Notes in Computer Science*, pages 52–65. Springer, 2015. doi:10.1007/978-3-319-19647-3_6.

11 C. Ge, F. Ma, X. Ma, F. Zhang, P. Huang, and J. Zhang. Approximating integer solution counting via space quantification for linear constraints. In Sarit Kraus, editor, *Proc. of IJCAI*, pages 1697–1703. ijcai.org, 2019. doi:10.24963/ijcai.2019/235.

12 C. Ge, F. Ma, P. Zhang, and J. Zhang. Computing and estimating the volume of the solution space of SMT(LA) constraints. *Theor. Comput. Sci.*, 743:110–129, 2018. doi:10.1016/j.tcs.2016.10.019.

13 Cunjing Ge and Armin Biere. Decomposition strategies to count integer solutions over linear constraints. In Zhi-Hua Zhou, editor, *Proc. of IJCAI*, pages 1389–1395. ijcai.org, 2021. doi:10.24963/ijcai.2021/192.

14 Jaco Geldenhuys, Matthew B. Dwyer, and Willem Visser. Probabilistic symbolic execution. In *Proc. of ISSTA*, pages 166–176, 2012. URL: http://doi.acm.org/10.1145/2338965.2336773, doi:10.1145/2338965.2336773.

15 M. Grötschel, L. Lovász, and A. Schrijver. Geometric algorithms and combinatorial optimization. *Combinatorica*, 1988.

16 Amy Huang, Liam Lloyd, Mohamed Omar, and James C. Boerkoel. New perspectives on flexibility in simple temporal planning. In *Proc. of ICAPS*, pages 123–131, 2018. URL: https://aaai.org/ocs/index.php/ICAPS/ICAPS18/paper/view/17775.

17 Samuel Kolb, Pedro Zuidberg Dos Martires, and Luc De Raedt. How to exploit structure while solving weighted model integration problems. In Amir Globerson and Ricardo Silva, editors, *Proc. of UAI*, volume 115 of *Proceedings of Machine Learning Research*, pages 744–754. AUAI Press, 2019. URL: http://proceedings.mlr.press/v115/kolb20a.html.

18 Jesús A. De Loera, Raymond Hemmecke, Jeremiah Tauzer, and Ruriko Yoshida. Effective lattice point counting in rational convex polytopes. *J. Symb. Comput.*, 38(4):1273–1302, 2004. URL: http://dx.doi.org/10.1016/j.jsc.2003.04.003, doi:10.1016/j.jsc.2003.04.003.

19 L. Lovász and I. Deák. Computational results of an O$^*(n^4)$ volume algorithm. *European Journal of Operational Research*, 216(1):152–161, 2012. URL: `http://dx.doi.org/10.1016/j.ejor.2011.06.024`, `doi:10.1016/j.ejor.2011.06.024`.

20 L. Lovász and S. Vempala. Hit-and-run from a corner. *SIAM J. Comput.*, 35(4):985–1005, 2006. URL: `http://dx.doi.org/10.1137/S009753970544727X`, `doi:10.1137/S009753970544727X`.

21 L. Lovász and S. Vempala. Simulated annealing in convex bodies and an O$^*(n^4)$ volume algorithm. *J. Comput. Syst. Sci.*, 72(2):392–417, 2006. URL: `http://dx.doi.org/10.1016/j.jcss.2005.08.004`, `doi:10.1016/j.jcss.2005.08.004`.

22 László Lovász. Hit-and-run mixes fast. *Math. Program.*, 86(3):443–461, 1999. `doi:10.1007/s101070050099`.

23 Kasper Søe Luckow, Corina S. Pasareanu, Matthew B. Dwyer, Antonio Filieri, and Willem Visser. Exact and approximate probabilistic symbolic execution for nondeterministic programs. In *Proc. of ASE*, pages 575–586, 2014. URL: `http://doi.acm.org/10.1145/2642937.2643011`, `doi:10.1145/2642937.2643011`.

24 F. Ma, S. Liu, and J. Zhang. Volume computation for boolean combination of linear arithmetic constraints. In *Proc. of CADE*, pages 453–468, 2009. `doi:10.1007/978-3-642-02959-2_33`.

25 P. Z. Dos Martires, A. Dries, and Luc De Raedt. Exact and approximate weighted model integration with probability density functions using knowledge compilation. In *Proc. of AAAI, 2019*, pages 7825–7833. AAAI Press, 2019. `doi:10.1609/aaai.v33i01.33017825`.

26 P. Morettin, A. Passerini, and R. Sebastiani. Advanced SMT techniques for weighted model integration. *Artif. Intell.*, 275:1–27, 2019. `doi:10.1016/j.artint.2019.04.003`.

27 Gilles Pesant. Counting-based search for constraint optimization problems. In *Proc. of AAAI*, pages 3441–3448, 2016. URL: `http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12065`.

28 Giuseppe Spallitta, Gabriele Masina, Paolo Morettin, Andrea Passerini, and Roberto Sebastiani. Smt-based weighted model integration with structure awareness. In James Cussens and Kun Zhang, editors, *Proc. of UAI*, volume 180 of *Proceedings of Machine Learning Research*, pages 1876–1885. PMLR, 2022. URL: `https://proceedings.mlr.press/v180/spallitta22a.html`.

29 Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM J. Comput.*, 8(3):410–421, 1979. `doi:10.1137/0208032`.

30 Sven Verdoolaege, Rachid Seghir, Kristof Beyls, Vincent Loechner, and Maurice Bruynooghe. Counting integer points in parametric polytopes using barvinok's rational functions. *Algorithmica*, 48(1):37–66, 2007. `doi:10.1007/s00453-006-1231-0`.

31 E. B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927. `doi:10.1080/01621459.1927.10502953`.

32 Alessandro Zanarini and Gilles Pesant. Solution counting algorithms for constraint-centered search heuristics. In *Proc. of CP*, pages 743–757, 2007. `doi:10.1007/978-3-540-74970-7\_52`.