algorithms

MDPI

Article

Estimating the Volume of the Solution Space of SMT(LIA) Constraints by a Flat Histogram Method

Wei Gao [1], Hengyi Lv [2], Qiang Zhang [3] and Dunbo Cai [4,*]

1 School of Astronautics, Beihang University, Beijing 100191, China; gaowei_bacc@163.com
2 Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun 130033, China; lv_hengyi@163.com
3 Beijing Aerospace Control Center, Beijing 100094, China; zhangqiang_bacc@163.com
4 Hubei Provincial Key Laboratory of Intelligent Robot, Wuhan Institute of Technology, Wuhan 430205, China
* Correspondence: dunbocai@wit.edu.cn

Received: 12 June 2018; Accepted: 14 September 2018; Published: 18 September 2018

Abstract: The satisfiability modulo theories (SMT) problem is to decide the satisfiability of a logical formula with respect to a given background theory. This work studies the counting version of SMT with respect to linear integer arithmetic (LIA), termed SMT(LIA). Specifically, the purpose of this paper is to count the number of solutions (volume) of a SMT(LIA) formula, which has many important applications and is computationally hard. To solve the counting problem, an approximate method that employs a recent Markov Chain Monte Carlo (MCMC) sampling strategy called "flat histogram" is proposed. Furthermore, two refinement strategies are proposed for the sampling process and result in two algorithms, MCMC-Flat$^{1/2}$ and MCMC-Flat$^{1/t}$, respectively. In MCMC-Flat$^{1/t}$, a pseudo sampling strategy is introduced to evaluate the flatness of histograms. Experimental results show that our MCMC-Flat$^{1/t}$ method can achieve good accuracy on both structured and random instances, and our MCMC-Flat$^{1/2}$ is scalable for instances of convex bodies with up to 7 variables.

Keywords: Markov Chain Monte-Carlo; SAT modulo theories; volume computation; flat histogram

1. Introduction

Satisfiability Modulo Theories (SMT) considers the satisfiability of a formula specified in a fragment of the first order logic with some kind of background theory [1]. One of those important theories is linear integer arithmetic (LIA). An atom in a SMT(LIA) formula is an inequation of LIA. A valuation on variables of the formula that evaluates the formula to true, is called a model (a.k.a. a solution).

Counting the number of models of a SMT(LIA) formula, #SMT(LIA), is theoretically hard. In the simple case, if all the LIA constraints are joined only by conjunctions, #SMT(LIA) reduces to the problem of computing the volume of convex bodies [2] which is shown to be #P-Complete [3]. However, the #SMT(LIA) problem considered in this paper supports formulas with arbitrary Boolean structures.

#SMT(LIA) has valuable applications in various areas, including approximate reasoning [4], program analysis [5] and verification [6,7]. Let us first consider an approximate reasoning problem. Suppose there is a knowledge base specified by a SMT(LIA) formula ϕ, and a formula φ is given. If neither φ nor $\neg\varphi$ is a logical consequence of ϕ, then it is reasonable to assume that the more *models* of ϕ support φ the more likely φ is true. Another important application of #SMT(LIA) is in the static analysis techniques of programs. One such technique is to check the program's properties by processing individual paths in the program's flow graph [8,9]. However, not all paths in the graph correspond to program executions. A path is called *feasible* if there are some values of the variables that can drive the program to be executed along that path. Otherwise, the path is called *infeasible*. There

are quite some works on path feasibility analysis [10]. A basic approach is to collect the conditions along the path to form a SMT(LIA) formula, and decide whether the formula is satisfiable or not. The program path is feasible if and only if the formula is satisfiable. So, the path feasibility analysis problem is reduced to a constraint solving problem [8,11]. Furthermore, we may ask which execution path is a hot/cold path [12]. Identification of such types of paths are valuable for testing case generation and can be translated into #SMT(LIA) problems [13].

In this paper, we propose an approximate method for #SMT(LIA). Specifically, a novel Markov Chain Monte Carlo (MCMC) sampling algorithm based on the flat histogram method is proposed [14]. The flat histogram method has been widely studied in statistical physics [15–17], which is based on the observation that if a random walk in energy space is performed with a probability proportional to the reciprocal of the density of states $1/g(E)$, then a flat histogram is achieved for the energy distribution. Here, E is an energy level and $g(E)$ is the number of states with energy E. For valuations of a SMT(LIA) formula, we define the energies of them, and use the idea of flat histogram method to sample from the space of valuations. The density of the zero energy is used to estimate the volume of the formula. The contribution of this paper is summarized as follows. First, this is the first work of exploiting the flat histogram method to solve the #SMT(LIA) problem. Secondly, a pseudo sampling strategy is introduced for evaluating the flatness of histograms generated in the sampling process, which is shown to be effective for #SMT(LIA) problems. Thirdly, extensive experiments were carried out to compare our methods with four related methods: an exact #SMT(LIA) solver developed in [18], an exact method for counting the number of solutions of convex bodies called VolCE+LattE [9], an approximate counter for convex bodies called SMTApproxMC [19] and another counter using bitblasting called ApproxMC2 [20,21]. Results show that our approaches are competitive to the state-of-the-art.

The organization of this paper is as follows. In the next section, we introduce the basic concepts and applications of #SMT(LIA), and some backgrounds on MCMC. In Section 3, we briefly review related works. In Section 4, a flat histogram based MCMC method for #STM(LIA) is presented. Following that, experimental results are presented. Finally, we conclude the paper.

2. Backgrounds

In this section, we first describe the #SMT(LIA) problem and its application in hot/cold path analysis, and then introduce the flat histogram method.

2.1. SMT(LIA) Formulas and Their Solution Spaces

Definition 1. *A linear integer arithmetic (LIA) constraint is an expression of the form* $a_1x_1 + a_2x_2 + \ldots + a_nx_n \infty a_0$, *where* $\{x_1, x_2, \ldots, x_n\}$ *are numeric variables that take integer values,* $\{a_1, a_2, \ldots, a_n\}$ *are constant coefficients (reals or integers), and the operator* $\infty \in \{\geq, >, =, \neq, <, \leq\}$. *A LIA constraint is also called an SMT(LIA) atom.*

Definition 2. *A SMT(LIA) formula* $\phi(b_1, b_2, \ldots, b_m)$ *is a Boolean formula over a set of SMT(LIA) atoms* $\{b_1, b_2, \ldots, b_m\}$.

Example 1. *A SMT(LIA) formula on a set of SMT(LIA) atoms:* $b_1 \equiv x_1 > 49$, $b_2 \equiv x_2 < 36$, *and* $b_3 \equiv x_3 > 11$, *is:* ϕ: $(x_1 > 49) \wedge (x_2 < 36 \vee x_3 > 11)$.

Definition 3. *The propositional skeleton (PS) of a SMT(LIA) formula* $\phi(b_1, b_2, \ldots, b_m)$, *denoted as* $PS(\phi)$, *is the Boolean structure over* $\{b_1, b_2, \ldots, b_m\}$.

For example, the SMT(LIA) formula ϕ in Example 1 has a propositional skeleton $PS(\phi)$: $b_1 \wedge (b_2 \vee b_3)$, which contains two clauses: b_1, and $b_2 \vee b_3$. To avoid confusion, we call x_i numeric variables, and call b_i Boolean variables.

Definition 4. *The solution (model) of a SMT(LIA) formula ϕ is a valuation on numeric variables in ϕ that evaluates ϕ to true. And the volume of ϕ is the number of its solutions, denoted by Vol(ϕ). The problem of computing the volume of a SMT(LIA) formula is denoted as #SMT(LIA).*

2.2. Hot/Cold Path Analysis

To show the practical importance of #SMT(LIA), we introduce a program analysis problem that can be translated into a #SMT(LIA) problem. In Figure 1, there is a path executing $i = 5$, and the other executing $i = 6$. We are interested in which path is the *hot* path. To solve this problem, we consider two SMT(LIA) formulas.

- The conditions along the first path: $(x > 49) \wedge ((y < 36) \vee (z > 11))$, whose volume is 4,085,328.
- The condition of the second path is the logical complement of the first condition. Its volume is $255 \times 255 \times 255 - 4{,}085{,}328 = 12{,}496{,}047$. As 12,496,047:4,085,328 $\approx$ 3.06:1, we predict that the second path (which executes $i = 6$) is a *hot* path and more test cases should be generated for it. Note that, we assume the ranges of variables x, y and z be $[-128, 127]$.

```
int x, y, z;
int i;
if ((x > 49) && ((y < 36) || (z > 11))) {
    i = 5;
} else {
    i = 6;
}
```

Figure 1. Example program for *Hot/Cold path* analysis.

2.3. Volumes of Convex Bodies and SMT(LIA) Formulas

In this subsection, we explain the differences between the volume computation problem of a convex body and that of a SMT(LIA) formula. A convex body is defined by a conjunction of several LIA constraints. For example, the area with mosaics in Figure 2a is a convex body, and it is defined by:

$$(x_1 \geq -128) \wedge (x_1 \leq 127) \wedge (x_2 \geq -128) \wedge (x_2 \leq 127) \wedge (-x_1 + x_2 < 0) \wedge (x_1 + x_2 > 5).$$

Note that the boundaries of x_1 and x_2 ($[-128, 127]$) are indicated by a square with broken lines, and the volume of the body is the number of value pairs for x_1 and x_2 that satisfy all the LIA constraints.

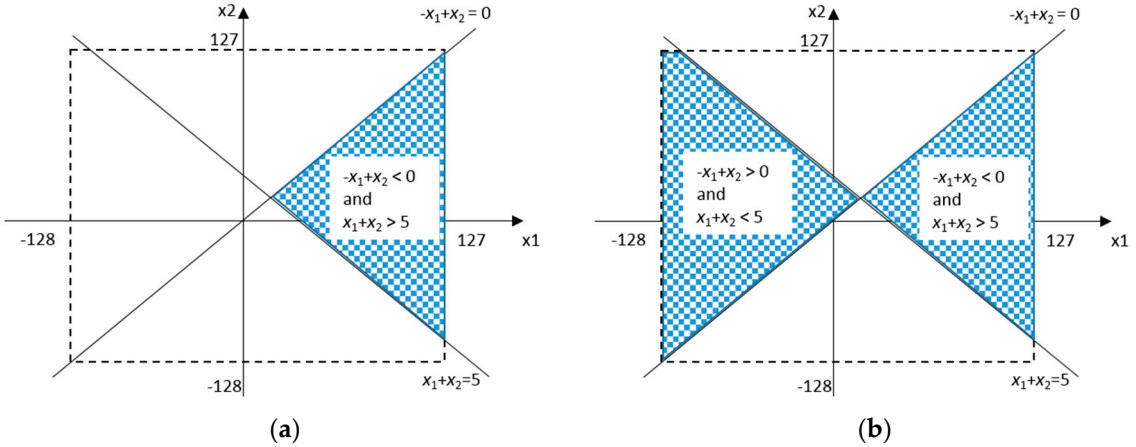


Figure 2. Demonstration of volumes of a convex body and a SMT(LIA) formula: (**a**) Volume of a convex body; (**b**) Volume of a SMT(LIA) formula which implies two convex bodies.

In contrast, a SMT(LIA) formula may imply multiple convex bodies, depending on the underlying logical structure. For example, with the same boundaries for x_1 and x_2, a formula

$$((-x_1 + x_2 < 0) \wedge (x_1 + x_2 > 5)) \vee ((-x_1 + x_2 > 0) \wedge (x_1 + x_2 < 5)).$$

implies two convex bodies as shown in Figure 2b. In general, a SMT(LIA) formula may imply m' convex bodies, and the maximal value of m' is: $2^m - 1$ where m is the number of Boolean variables involved. However, the idea to break a SMT(LIA) formula into a set of convex bodies and then take the sum of the corresponding volumes is not feasible. One reason is that this idea possibly results in an exponential number of convex bodies, and the other reason is there may exist overlaps among the convex bodies [8].

2.4. Markov Chain Monte Carlo and the Flat Histogram Method

Markov Chain Monte Carlo (MCMC) is a sampling method that has been widely used for statistical physics. Sampling from a state (configuration) space S is the process of generating samples from a probability distribution π defined over S. The aim is to estimate the distribution or to compute the states having a maximum (minimum) probability. A MCMC algorithm samples the space in a way that the transition probability from the current state σ to the next state σ', $p_{\sigma \to \sigma'}$, depends only on σ, but not on any previously sampled states. If $\pi(\sigma) > 0$ for any $\sigma \in S$, and $p_{\sigma \to \sigma'}$ satisfies the *detailed balance property*,

$$\pi(\sigma) p_{\sigma \to \sigma'} = \pi(\sigma') p_{\sigma' \to \sigma}. \tag{1}$$

then the sampling process will reach a stationary distribution.

In statistical physics, the flat histogram method [14] was firstly used to estimate the *density of states* (DOS) of a state space S w.r.t. an energy function,

$$g : S \to \mathbb{N}.$$

The DOS of a given energy level E is,

$$n(E) = |\{\sigma : g(\sigma) = E\}|.$$

To estimate $n(\cdot)$, the flat histogram method visits a state σ in the energy level $g(\sigma)$ with a probability $p(\sigma)$ that is inversely proportional to $n(g(\sigma))$. As a result, a flat visit histogram to each energy level is generated. Specifically, the method defines a random walk process with the following transition probability

$$p_{\sigma \to \sigma'} = min\{1, \frac{1/n(E')}{1/n(E)}\} = min\{1, \frac{n(E)}{n(E')}\}. \tag{2}$$

where $g(\sigma) = E$ and $g(\sigma') = E'$. The process satisfies the detailed balance equation

$$P(\sigma) p_{E \to E'} = P(\sigma') p_{E' \to E}. \tag{3}$$

when $P(\sigma)$ is proportional to $1/n(E)$. This strategy guides the process to visit states at every energy level equally often.

3. Related Works

A pioneer work of applying the flat histogram method in model counting is for Boolean formulas [22]. The authors built the connection between the model counting problem of SAT and the DOS computation problem. Their method was shown to outperform the standard Metropolis sampling. In this work, we extend the flat histogram to the field of SMT for solving #SMT(LIA).

A recent work on #SMT(LIA) developed a multiphase Monte-Carlo method that exploits a novel technique for estimating the volume of convex polytopes [9]. Specifically, that work designed a set

of strategies to reduce the number of samplings. On top of their methods, a tool called VolCE that uses LattE [23] was developed (we refer to it as VolCE+LattE) for #SMT(LIA). Experiments in [9] show that VolCE+LattE is superior to a word-level hash-mapping method SMTApproxMC [19] on instances which have more than 4,000,000 solutions.

Hash-mapping-based approximate methods for model counting of SMT formulas have been paid much attention in recent years. The bit-level hash-mapping-based methods ApproxMC [20] and its improved version ApproxMC2 [21] used XOR-based bit-level hash functions to obtain a randomized subset of the solution space by calling a satisfiability solver. However, the approaches have the danger of introducing too many discrete variables when dealing with linear real constraints [9]. On the other hand, meanings of variables are lost in a bit-level encoding, as argued in [9,19]. To advance the techniques of estimating by hash-mapping and to overcome the weakness of bit-level encoding, a word-level hash-mapping method was proposed in [19], where an approximate model counter for linear integer constraints called SMTApproxMC was developed. A great advantage of SMTApproxMC is the ability to exploits the continuously advanced reasoning power of word-level SMT solvers. However, SMTApproxMC is reported to have difficulty in scaling to instances having a large number of solutions [9].

Monte-Carlo based model counting (or volume computation) has a wealth of studies. Nonetheless, few implementations are available for SMT. One recently implemented direct Monte-Carlo method for SMT with linear real constraints was reported in [24]. They showed, by experiments on a set of convex bodies involving increasing number of variables, their direct Monte Carlo method fails on all instances with seven or more variables. In contrast, our experiments (in Section 5: More Results on Convex Bodies) on a subset of their benchmarks will show that one of our techniques can work on many instances with seven or more variables.

We briefly summarize the characteristics of our techniques and the related as follows.

- Our techniques vs. VolCE+LattE [9]: Both techniques are suited for #SMT(LIA). VolCE+LattE is an exact method whereas our methods are approximate. In our experiments, VolCE+LattE fails to scale on many convex body instances (obtained from [24]) that are with more than six variables, whereas one of our techniques succeed in reporting solutions for the test set. Note that both techniques require low memories. In our test, their required memory on all instances is less than 1 GB. So, a failure of VolCE+LattE is due to time out.
- Our techniques vs. SMTApproxMC [19] and ApproxMC2 [21]: The three are approximate techniques. SMTApproxMC and ApproxMC2 provide a guarantee that if given a tolerance $\varepsilon > 0$ and a confidence $1–\delta \in (0, 1)$, its output lies in the interval $[(1 + \varepsilon)^{-1} R_F, (1 + \varepsilon) R_F]$ with probability greater than δ, where R_F is the real count of a formula. Our techniques do not provide such a guarantee. However, experiments will show that our techniques lead to a relative error smaller than 20% in most of the tests. In addition, our methods are capable of working on tested instances with 4,000,000 and more solutions, whereas SMTApproxMC cannot. ApproxMC2 has similar scalability issue as well (see Section 5: More Results on Convex Bodies).
- Our techniques vs. the direct Monte Carlo method [23]: Both techniques are not complicated applications of the Monte Carlo sampling method. The direct Monte Carlo method is reported to fail in sampling from the solution area of convex body instances with seven or more variables, whereas one of our techniques succeeds in obtaining a lot of samplings from the solution areas for many instances extracted from the same test set.

4. The Proposed Flat Histogram Method

To estimate the volume of an SMT(LIA) formula, we propose a flat histogram method that consists of three components.

- A specialized energy function g over the valuations of the SMT(LIA) formula, which satisfies that for any valuation γ, $g(\gamma) = 0$ if γ is a solution and $g(\gamma) > 0$ otherwise.

- A MCMC algorithm based on the flat histogram method to estimate the density of states with respect to g.
- Histogram reweighting strategies and flatness evaluating methods.

After the sampling process, the DOS for energy zero, $n(0)$, is outputted as the volume of the SMT(LIA) formula.

4.1. The Energy Function

For SMT(LIA) formulas, we define an energy function based on their propositional skeletons. Given the propositional skeleton of a formula ϕ: $PS(\phi)$, and the set of clauses of $PS(\phi)$: $C(\phi)$, we define the energy of a valuation σ as

$$g(\sigma) = |\{c : c \in C(\phi) \textit{ that is not satisfied by } \sigma\}|. \tag{4}$$

It can be seen that g maps valuations of numeric variables to integers in $[0, |C(\phi)|]$, which facilitates program implementation. We can use a one-dimensional array with size $|C(\phi)| + 1$, and thus the ith entry is used to record the density of valuations with energy i.

4.2. The Sampling Algorithm

In this subsection, we propose a MCMC method to solve the #SMT(LIA) problem based on the flat histogram method. In the proposed method, we make each random move follow the probability specified in Equation (1). Therefore, our method avoids using Metropolis sampling, and visits states at every energy level equally often. However, the density function $n(\cdot)$ in Equation (1) is not known a priori. To handle this, we use a changing current estimate $n'(\cdot)$ to approximate $n(\cdot)$. Initially, $n'(E)$ is set to 1 for every energy level E. With the initial values of $n'(\cdot)$, our algorithm samples the space with increasing the visit histogram $H(\mathrm{g}(\sigma))$ by 1 and update $n'(g(\sigma))$ with a *modification factor* F, every time a valuation σ is visited. Specifically, the estimate of DOS is updated by

$$n'(g(\sigma)) \leftarrow n'(g(\sigma)) \times F. \tag{5}$$

The sampling process repeats until the visit histogram $H(\cdot)$ is sufficiently flat. If F does not reach its designated value, a new sampling process is started with resetting $H(\cdot)$ and starts with a refinement of F. The role of F will be explained next. An outline of the *MCMC-Flat algorithm* is shown in Algorithm 1. It returns the estimate $n'(0)$.

As is seen at the end of Algorithm 1, the information in $n'(\cdot)$ reflects the relative density of each energy level. To obtain the real density of each level, we normalize $n'(\cdot)$ w.r.t. the volume of the sampling area. For a SMT(LIA) formula ϕ defined on K numeric variables and each has the same range $[lb, ub]$ (lb is the lower bound and ub is the upper bound), the volume of the sampling area is $(ub - lb)^K$. Therefore, the volume of ϕ is obtained by

$$\frac{n'(0)}{\sum\limits_{e \in E} n'(e)} \times (ub - lb)^K. \tag{6}$$

Algorithm 1. *MCMC-Flat for #SMT(LIA)*

$N'(E) \leftarrow 1$ for every energy level E
$F \leftarrow F_0$
Randomly generate a valuation σ
$n'(\sigma) \leftarrow n'(\sigma) \times F$
$H(\sigma) \leftarrow H(\sigma) + 1$
repeat
  repeat
    Generate a new valuation σ'
    if $n'(g(\sigma')) < n'(g(\sigma))$ **then**
      $\sigma \leftarrow \sigma'$
    else
      Move to σ' with probability defined in Equation (2)
    end if
    $n'(\sigma) \leftarrow n'(\sigma) \times F$//reweighting the density
    $H(\sigma) \leftarrow H(\sigma) + 1$//increase the visit histogram
  until *IsFlat*(H)
  $F \leftarrow$ *Update*(F)
  //prepare to a new sweep
  $H(E) \leftarrow 0$ for every E
until F is close enough to F_{final}
Normalize n'
return $n'(0)$ using Equation (6)

4.3. Updating Strategy for the Modification Factor

In Algorithm 1, the modification factor F is used to reweight the density of states. The outer loop updates F at each iteration with the function *Update*(F). Two strategies of updating F were studied.

- MCMC-Flat$^{1/2}$ (the 1/2-strategy): Initially, set F to 1.5, and update F to $F^{1/2}$ when H is flat. Each time F is updated, reset $H(\cdot)$ [14].
- MCMC-Flat$^{1/t}$ (the 1/t-strategy): Initially, set F to 1.0. After a constant MC time (e.g., 1000), check $H(E)$, and if $H(E) \neq 0$ for all E then refine $F \leftarrow F/2$ and reset H$(\cdot)$. If $F \leq 1/t$ ($t = j/N$, j is the number of trails and N is the number of energy levels.) then F $\leftarrow 1/t$ and keep the value in following trails. If $F < F_{final}$, we stop the sampling process [25].

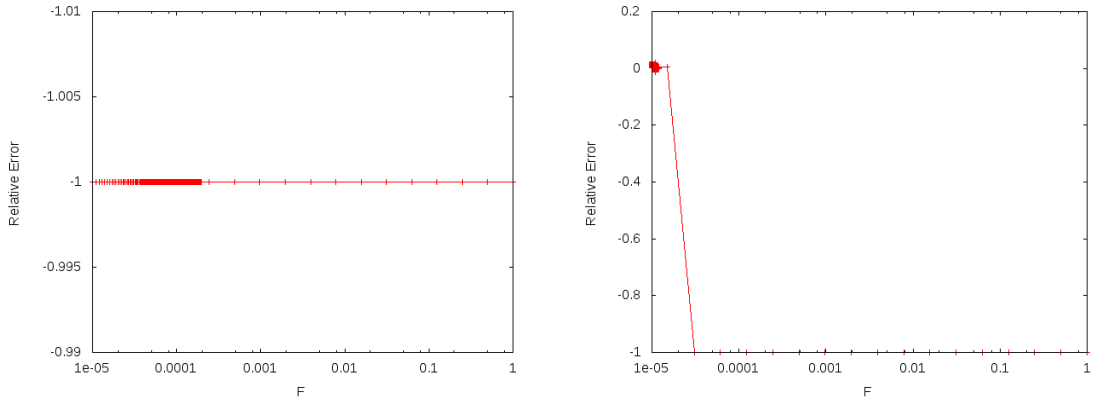
Belardinelli et al. [25] showed that the 1/t strategy can lead to a smaller error than the 1/2 strategy. We also verified this behavior on a small set of test problems. So, we focus on studying the performance of MCMC-Flat$^{1/t}$ on the #SMT(LIA) problem. One of our important findings is that the flatness condition for $H(\cdot)$ is difficult to reach for some test problems. Next, we will examine the notion of flatness of a visit histogram and introduce a new rule to evaluate flatness.

4.4. Evaluating the Flatness of Visit Histogram

One method to evaluate the flatness of the visit histogram H is based on its minimum value *min* and maximum value *max* [14]. Specifically, H is considered to be flat if $max - min < \delta \times max$, where δ is a pre-set constant parameter. Obviously, the more flat H is (i.e., with smaller δ) the more uniformly Algorithm 1 visits each energy level. However, with smaller δ we need more sampling steps. Furthermore, literature [26] pointed out that the flatness is not a good indicator for convergence.

Among other works, Belardinelli and Pereyra [25] introduced a condition to evaluate the visits of each energy level: every energy level must be visited one time. We tested MCMC-Flat$^{1/t}$ with this condition on one of our random problems and found that MCMC-Flat$^{1/t}$ does not converge. In particular, with this condition the relative error of MCMC-Flat$^{1/t}$ stays at -1.0 (see Figure 3a).

One of the underlying reasons, as we conjecture, is that the number of states of a particular energy level is very large for SMT(LIA) formulas with many variables (a.k.a. high dimensions). So, MCMC-Flat$^{1/t}$ needs many trails to visit every energy level and to fulfil the condition for the visit histogram. To overcome the shortcomings, we introduce a new rule to analyze the visit histogram (as shown in Algorithm 2). Algorithm 2 marks one unvisited energy level as visited and then returns, each time it is called (i.e., do a pseudo sampling), while it does not stop the current sampling process. This behavior will force the sampling process do another sweep on energy levels. Note that a sweep will require more Monte Carlo trials. In the worst case, Algorithm 2 will force the sampling process to make N sweeps, where N is the number of energy levels. The amount of sweeps shall be enough to visit each energy level. We call Algorithm 1 with this method MCMC-Flat$^{1/t\text{-pv}}$. The behaviour of MCMC-Flat$^{1/t\text{-pv}}$ is shown in Figure 3b. It can be seen that MCMC-Flat$^{1/t\text{-pv}}$ converges (with a decreasing relative error) when the modification factor F is decreasing. Note that in our method, F initially takes 1.0, and is reduced in the following iterations. What we expect is that the smaller F is, the more accurate our method's estimate of $n(0)$ is.



(**a**) (**b**)

Figure 3. Dynamical behavior of MCMC-Flat$^{1/t}$ and MCMC-Flat$^{1/t\text{-pv}}$ on the same test problem: (**a**) MCMC-Flat$^{1/t}$ does not converge when F is small enough; (**b**) MCMC-Flat$^{1/t\text{-pv}}$ converges when F is small enough.

Algorithm 2. *Pseudo Visit (PV) of Histogram*

Input: *A visit histogram H*
Output: *A Boolean value indicates if every energy is visited*
for each i in H
  if $H[i] = 0$
    $H[i] \leftarrow 1$
    return false
end if
end for
return true

5. Experimental Evaluation

This section consists of two parts. In the first part, we show the accuracy and efficiency of the proposed algorithms on structured and random SMT(LIA) instances. In the second part, we compare our techniques with VolCE+LattE [9], SMTApproxMC [19] and ApproxMC2 [21], on hard convex bodies (where every linear integer constraint is joined by conjunction) instances extracted from the benchmark suit of [24].

In the first part, every structured instance is defined in a two-dimensional space (i.e., involving two numeric variables) containing a single square of varying volumes. The random instances contain different numbers of variables, LIA constraints, and clauses. The Exact Method presented in [18] is used as a reference. Properties of random instances are shown in Table 1.

Table 1. Properties of test problems with random structure.

Problem	Number of Variables	Number of LIA Constraints	Number of Clauses
ran1	5	10	20
ran2	15	20	40
ran3	8	10	20
ran4	10	20	50
ran5	5	20	40
ran6	5	15	40
ran7	8	20	50
ran8	7	20	50
ran9	7	20	80
ran10	8	20	100
ran11	9	20	50
ran12	8	16	80

All the experiments were run on a Linux system with a CPU running at 2.53 GHz, 60 min time limit and 2 GB memory limit. The programs were restricted to run on one core of the CPU by using the Linux command taskset. MCMC-Flat$^{1/2}$ and MCMC-Flat$^{1/\text{t-pv}}$ were run 10 times on each problem; their results were averaged on successful runs. In the following results, Vol. is short for volume, SR is short for success ratio, and RE is short for relative error. Let d' be an estimation of a target value d, we define the relative error d to d' as $\text{RE}(d, d') = (d' - d)/d$. We use MO to indicate the memory out of the Exact Method and use "\" to indicate 'time out'.

With structural instances, we want to see that whether the accuracies of MCMC-Flat methods improve with increasing size of volumes. The detailed results are shown in Table 2. On instance $s1$, the relative errors of MCMC-Flat$^{1/2}$ and MCMC-Flat$^{1/\text{t-pv}}$ are 302.69% and 297.97%, respectively. The errors of the two algorithms decrease consistently when the volumes of the target convex bodies increase. In addition, on the structured instance $s6$ which is with the maximal volume 4096, both of our algorithms reach small relative errors within 10% (9.33% and 5.65% for MCMC-Flat$^{1/2}$ and MCMC-Flat$^{1/\text{t-pv}}$, respectively). We can also see that MCMC-Flat$^{1/\text{t}-\text{pv}}$ leads to smaller relative errors, compared with MCMC-Flat$^{1/2}$, on instances $s5$, $s6$ and $s7$, though the running times of the former are more than the latter. Finally, we note that the Exact Method is the fastest among these three algorithms, with nearly neglectable running times. However, this advantage of the Exact Method will be lost on complex SMT(LIA) formulas contained in our random instances.

Table 2. Comparison of Exact Method and MCMC-Flat methods on structural problems.

Problems	Exact Method		MCMC-Flat$^{1/2}$				MCMC-Flat$^{1/\text{t}-\text{pv}}$			
	Vol.	Time	Vol.	Time	RE (%)	SR	Vol.	Time	RE (%)	SR
$s1$	4	0	16.11	1.31	302.69	1.0	15.92	1.07	297.97	1.0
$s2$	16	0	25.38	1.56	58.61	1.0	36.19	1.07	126.21	1.0
$s3$	64	0	95.71	0.47	49.54	1.0	99.94	1.07	56.15	1.0
$s4$	256	0	322.06	0.14	25.80	1.0	322.71	1.06	26.06	1.0
$s5$	1024	0	1215.48	0.19	18.70	1.0	1137.09	1.05	11.04	1.0
$s6$	4096	0	4478.05	0.17	9.33	1.0	4327.27	1.04	5.65	1.0
$s7$	16129	0	17,020.84	0.06	5.53	1.0	16,396.42	0.98	1.66	1.0

With random instances, the ability of MCMC-Flat methods on hard instances is verified (see Table 3). Table 3 shows that MCMC-Flat$^{1/\text{t-pv}}$ reaches good accuracy very fast, in contrast with

the running time of MCMC-Flat$^{1/2}$. For example, MCMC-Flat$^{1/2}$ takes more than 2000 s on instances *ran2*, *ran5*, and *ran9*, whereas MCMC-Flat$^{1/\text{t-pv}}$ takes at most 205.18 s for *ran9*. Note that the relative errors of MCMC-Flat$^{1/2}$ and MCMC-Flat$^{1/\text{t-pv}}$ on *ran2*, *ran5*, and *ran9* are close. It can also be seen that the volume estimations of MCMC-Flat$^{1/\text{t-pv}}$ on instances *ran2*, *ran4*, *ran7*, and *ran10* are close to those of MCMC-Flat$^{1/2}$. On all the random instances tested, the running times of MCMC-Flat$^{1/\text{t-pv}}$ varies from about ten seconds to a few hundreds of seconds, whereas those of MCMC-Flat$^{1/2}$ varies from 7.59 to 3266.07 s. Therefore, MCMC-Flat$^{1/\text{t-pv}}$ is more scalable than MCMC-Flat$^{1/2}$. Next, let us look at the results of the Exact Method. The Exact Method runs out of memory on *ran2*, *ran4*, *ran7*, *ran10* and *ran11*, which is of dimension 15, 10, 8, 8, and 9 respectively (please refer to Table 1). Seeing that the Exact Method succeeds on other instances with less than 8 dimensions, we can conclude that the Exact Method does not scale well. Overall, we see that MCMC-Flat$^{1/\text{t-pv}}$ has a good scalability and outperforms MCMC-Flat$^{1/2}$ and the Exact Method.

Table 3. Comparison of Exact Method and MCMC-Flat methods on random instances.

Problems	Exact Method		MCMC-Flat$^{1/2}$				MCMC-Flat$^{1/\text{t-pv}}$			
	Vol.	Time	Vol.	Time	RE (%)	SR	Vol.	Time	RE (%)	SR
ran1	5.86×10^{11}	0.15	5.84×10^{11}	7.59	−0.35	1.0	5.84×10^{11}	12.65	−0.51	1.0
ran2	MO	\	4.18×10^{35}	2186.25	\	1.0	4.22×10^{35}	62.82	\	1.0
ran3	2.71×10^{18}	8.36	2.72×10^{18}	4.58	0.04	1.0	2.73×10^{18}	14.23	0.44	1.0
ran4	MO	\	5.22×10^{23}	65.51	\	1.0	5.21×10^{23}	74.92	\	1.0
ran5	4.02×10^{10}	37.51	4.06×10^{10}	2087.45	0.91	1.0	4.07×10^{10}	47.05	1.04	1.0
ran6	3.84×10^{10}	0.73	3.81×10^{10}	31.37	−0.74	1.0	3.82×10^{10}	45.65	−0.59	1.0
ran7	MO	\	4.89×10^{16}	474.35	\	1.0	4.85×10^{16}	76.38	\	1.0
ran8	1.24×10^{14}	4.36	1.29×10^{14}	188.96	4.25	1.0	1.27×10^{14}	73.15	2.07	1.0
ran9	3.14×10^{16}	3625.2	3.13×10^{16}	3266.07	−0.33	0.1	3.15×10^{16}	205.18	0.22	1.0
ran10	MO	\	3.20×10^{17}	476.1	\	0.5	3.25×10^{17}	378.83	\	1.0
ran11	MO	\	2.90×10^{19}	655.12	\	1.0	2.94×10^{19}	76.85	\	1.0
ran12	5.23×10^{18}	709.74	5.21×10^{18}	45.94	−0.35	1.0	5.21×10^{18}	196.62	−0.27	1.0

Another note is that both MCMC-Flat$^{1/2}$ and MCMC-Flat$^{1/\text{t-pv}}$ return greater than zero estimates on all the random instances, especially on those with 8 or more dimensions. The results indicate the ability of our MCMC-Flat algorithms in finding at least one solution. In contrast, the direct Monte Carlo method implemented in [24] failed to find any solution for instances with dimensions greater than 7, which results in a relative error of 100% (please refer to Figure 10b in [24]). We argue that this good ability of our MCMC-Flat algorithms is due to both the transition probability in Equation (2) and a mapping from the valuation space (with possibly exponentially number of states) of a SMT(LIA) formula to our defined energy space that is linear in the number of clauses.

More Results on Convex Bodies

In the second part of experiments, we focus on a restricted form of STM(LIA), the convex body. Specifically, using the volume computation problem of convex bodies, we compare our MCMC-flat methods with some state-of-the-art model counters. Instances of convex bodies were obtained from the benchmarks (http://code.google.com/p/rvc/) used in [24]. All the problems are extracted from the sub-folder named "Exp3_1". A problem with name *m-n-i* involves m variables, and $n+1$ constraints where i is the index of a variant. Each variable has a value bound $[-15, 15]$. Every constraint is an inequality, and coefficients of variables are rational numbers represented by fractions. For example, the first inequality of the problem "7-5-1" is:

$$-1/2 \cdot x_1 - 2/5 \cdot x_2 - 7/5 \cdot x_3 + 7/10 \cdot x_4 + 3/8 \cdot x_5 \leq -3/5. \quad (7)$$

A brief summary on properties of the selected instances are presented in Table 4. "#Variable", "#Constraint", and "Space", in Table 4, stand for the number of variables, the number of LIA constraints,

and the number of valuations in the space, respectively. The "Volume/Space" column is the proportion of the number of models to the size of the space, where "NA" represents an unknown property. Intuitively, a problem with lower Volume/Space is more constrained.

Table 4. Properties of problems used to compare MCMC-Flat$^{1/2}$, Volce+LattE, and SMTApproxMC.

Problems	#Variable	#Constraint	Space	Volume/Space (%)
5-10-1	5	10	2.43×10^7	0.0171276
5-10-2	5	10	2.43×10^7	0.0198066
5-10-3	5	10	2.43×10^7	0.6724568
5-10-4	5	10	2.43×10^7	0.0487407
5-10-5	5	10	2.43×10^7	0.9811399
5-20-1	5	20	2.43×10^7	0.0014280
5-20-2	5	20	2.43×10^7	0.0010206
5-20-3	5	20	2.43×10^7	0.0044444
5-20-4	5	20	2.43×10^7	0.0000329
5-20-5	5	20	2.43×10^7	0.0000494
6-5-1	6	5	7.29×10^8	25.3803016
6-5-2	6	5	7.29×10^9	0.1043014
6-5-3	6	5	7.29×10^{10}	NA
6-5-4	6	5	7.29×10^{11}	0.0017279
6-5-5	6	5	7.29×10^{12}	NA
6-10-1	6	10	7.29×10^{13}	NA
7-5-1	7	5	2.187×10^{10}	NA
7-5-2	7	5	2.187×10^{10}	NA
7-5-3	7	5	2.187×10^{10}	NA
7-5-4	7	5	2.187×10^{10}	NA
7-5-5	7	5	2.187×10^{10}	NA
7-10-1	7	10	2.187×10^{10}	NA

We take an exact model counter Volce+LattE [9] and two approximate model counters SMTApproxMC [19] and ApproxMC2 [21] for comparison. Note that all the three are state-of-the-art model counters. To run them, the benchmark problems were converted to their corresponding formats. The problem format in [24] is similar to that of our methods' accepts. However, the format is quite different from the formats of Volce+LattE, SMTApproxMC, and ApproxMC2. Volce+LattE accepts an input format called the "Enhanced DIMACS CNF Format", whereas SMTApproxMC accepts SMT-LIB2 format of Bit Vector theory. In our implementation, we first translate the instances from [24] into the "Enhanced DIMACS CNF Format" of Volce+LattE. Specifically, denominators of coefficients in a LIA constraint are all multiplied by their least common multiple. For example, the translation result of (7) is,

$$-20 \cdot x_1 - 16 \cdot x_2 - 56 \cdot x_3 + 28 \cdot x_4 + 15 \cdot x_5 \leq -24. \tag{8}$$

In this way, every coefficient in an LIA constraint is a number, instead of a fraction. In addition, every value bound for a variable is translated into a corresponding inequality. Moreover, due to that each problem instance representing a convex body, our converter states that all constraints must be satisfied. For SMTApproxMC, equations in the form of (8) are translated into sentences in the Fixed Size Bit Vector theory of SMT-LIB2, where the size of a bit vector is 32.

To run ApproxMC2 [21], the benchmark problems were bit-blasted by using an embedded function of the SMT solver Yices (http://yices.csl.sri.com/). Specifically, for the problems described in the SMT-LIB2 Bit Vector theory (which is suited for SMTApproxMC), the SMT-LIB2 keywords are replaced by Yices' corresponding ones, for example: "bvsle" in SMT-LIB2 is replaced with "bv-sle" in the Yices language. Furthermore, at the end of the problem, an "export-to-dimacs" command is set. The resulted problem file was then bit-blasted by Yices (with command-line option "–logic = QF_BV").

All the original and translated problems for SMTApproxMC and ApproxMC2 are publicly available (https://github.com/PlanningByCai/MCMCFlat-Count-1-2).

This part of the test was carried out on a 64-bit Linux platform with a 2.6 GHz CPU, 4 GB memory limit. The time limit of each test is 12 h. Volce+LattE was executed once, since it is an exact method. MCMC-Flat$^{1/2}$ was given 5 runs on each instance and the averaged results are reported. A "\" indicates failure due to time out. SMTApproxMC was run with the default configuration (50 iterations, tolerance $\varepsilon = 0.8$ and $\delta = 0.2$, for an instance). Note that these settings for ε and δ are the same as those used by the authors of SMTApproxMC, which make the value v returned by SMTApproxMC satisfy

$$\Pr\left[\frac{|R_F|}{1+\varepsilon} < v < (1+\varepsilon)|R_F|\right] \geq 1-\delta \tag{9}$$

Note that RF is the real volume. ApproxMC2 was run with its default configuration as well (with the same ε and δ). So, return values of ApproxMC2 satisfy (9) as well.

Results regarding Volce+LattE and MCMC-Flat$^{1/2}$ are shown in Table 5. We can observe that Volce+LattE fails on many instances with 6 and 7 variables, among which it only succeeds on 6-5-1, 6-5-2, and 6-5-4 with dozens of minutes. In contrast, our MCMC-Flat$^{1/2}$ method works on all the instances. On the other hand, MCMC-Flat$^{1/2}$ reports a result of 1.74×10^8 on the problem 6-5-1 with a relative error 6.21%. In summary, on the 13 instances on which Volce+LattE succeeds, MCMC-Flat$^{1/2}$ gives 3 results with RE beyond 20% (52.07% for 5-10-2, 48.17% for 5-20-5, and -25.17% for 6-5-4), and gives the other 10 results with RE in $[-20\%, 20\%]$. Therefore, results in Table 5 show that MCMC-Flat$^{1/2}$ is more scalable than Volce+LattE.

Table 6 are results of MCMC-Flat$^{1/2}$, SMTApproxMC and ApproxMC2; the results of MCMC-Flat$^{1/2}$ are the same as those in Table 5. In Table 6, a brief summary of the results is presented at the bottom: "#Solved" is the number of solved instances of an algorithm, "#BestRE" is the number of instances on which an algorithm obtained the best relative error, and "#AvgTime" is the averaged solving time over instances that are solved by all the three algorithms.

Table 5. Comparison of Volce+LattE and MCMC-Flat$^{1/2}$ on convex bodies.

Prob	Volce+LattE		MCMC-Flat$^{1/2}$		
	Vol.	Time	Vol.	Time	RE (%)
5-10-1	4162	13.5	4394.4	625.51	5.6
5-10-2	4813	111.43	7319.1	60.11	52.1
5-10-3	1.63×10^5	218.25	1.94×10^5	9.73	19.0
5-10-4	11844	302.69	13169.6	16.34	11.2
5-10-5	2.38×10^5	95.22	2.43×10^5	56.09	2.1
5-20-1	347	383.69	347.5	75.47	0.1
5-20-2	248	366.6	260.4	1555.6	5.0
5-20-3	1080	194.86	898.3	91.32	−16.8
5-20-4	8	257.82	7.9	162.1	−1.3
5-20-5	12	643.42	17.8	719.27	48.3
6-5-1	1.85×10^8	2248.59	1.74×10^8	28.67	−6.2
6-5-2	7.60×10^6	1059.29	6.52×10^6	5.7	−14.2
6-5-3	\	\	5.37×10^6	796.73	\
6-5-4	1.26×10^7	839.3	9.43×10^6	5.62	−25.2
6-5-5	\	\	7.45×10^6	9.71	\
6-10-1	\	\	1.48×10^5	83.67	\
7-5-1	\	\	4.91×10^8	21.86	\
7-5-2	\	\	1.41×10^8	9.31	\
7-5-3	\	\	7.00×10^8	25.09	\
7-5-4	\	\	6.58×10^8	8.23	\
7-5-5	\	\	1.65×10^8	14.54	\
7-10-1	\	\	6.58×10^4	634.31	\

Table 6. Comparison of MCMC-Flat$^{1/2}$, SMTApproxMC and ApproxMC2 on convex bodies.

Prob.	MCMC-Flat$^{1/2}$			SMTApproxMC			ApproxMC2		
	Vol.	**Time**	**RE (%)**	**Vol.**	**Time**	**RE (%)**	**Vol.**	**Time**	**RE (%)**
5-10-1	4394.4	625.51	**5.6**	16	504.15	−99.6	4416	28.37	6.1
5-10-2	7319.1	60.11	52.1	19.5	552.85	−99.6	4608	34.99	**−4.3**
5-10-3	1.94×10^5	9.73	**19.0**	1.97×10^5	2158.77	20.3	0	4.20×10^3	−100
5-10-4	13169.6	16.34	11.2	45	1276.02	−99.6	12,288	104.77	**3.7**
5-10-5	2.43×10^5	56.09	**2.1**	3	1570.6	−100	2.21×10^5	2.43×10^3	−7.2
5-20-1	347.5	75.47	**0.1**	21	562.4	−93.9	352	24.6	1.4
5-20-2	260.4	1555.6	5.0	239	489.37	−3.6	248	22.75	**0.0**
5-20-3	898.3	91.32	**−16.8**	833	545.84	−22.9	880	28.13	−18.5
5-20-4	7.9	162.1	-1.3	2.5	85.53	−68.8	8	28.94	**0.0**
5-20-5	17.8	719.27	48.3	3.5	310.64	−70.8	12	1.55	**0.0**
6-5-1	1.74×10^8	28.67	**−6.2**	\	\	\	0	5.74×10^3	−100
6-5-2	6.52×10^6	5.7	**−14.2**	2.56×10^6	56,611.85	−66.4	0	6.00×10^3	−100
6-5-3	5.37×10^6	\	796.73	79.5	52,437.11	\	0	6.00×10^3	\
6-5-4	9.43×10^6	5.62	**−25.2**	192.5	65,540.31	−100	0	6.00×10^3	−100
6-5-5	7.45×10^6	9.71	\	1.04×10^7	68,279.69	\	0	6.00×10^3	\
6-10-1	1.48×10^5	83.67	\	1.31×10^5	3660.63	\	1.19×10^5	1.93×10^3	\
7-5-1	4.91×10^8	21.86	\	\	\	\	0	5.29×10^3	\
7-5-2	1.41×10^8	9.31	\	\	\	\	0	6.00×10^3	\
7-5-3	7.00×10^8	25.09	\	\	\	\	0	6.00×10^3	\
7-5-4	6.58×10^8	8.23	\	\	\	\	0	6.00×10^3	\
7-5-5	1.65×10^8	14.54	\	\	\	\	0	6.00×10^3	\
7-10-1	6.58×10^4	634.31	\	\	\	\	5.63×10^4	1.15×10^3	\
#Solved	**22**			15			11		
#BestRE	**8**			0			5		
AvgTime	**344.55**			955.80			463.33		

We make several observations from the results in Table 6, as follows.

- In terms of scalability, our MCMC-Flat$^{1/2}$ method is superior to SMTApproxMC and ApproxMC2. In [9], the word-level hash-mapping-based method SMTApproxMC was shown to handle instances with less than 3.98×10^6 models (around 4,000,000) and failed on others. On our test platform, SMTApproxMC got a result of 1.04×10^7 on instance 6-5-5. However, SMTApproxMC can return a value 15 out of 22 instances, whereas MCMC-Flat$^{1/2}$ returns values on all the 22 instances. Notably, MCMC-Flat$^{1/2}$ reports a result of 1.74×10^8 on the problem 6-5-1 with a low relative error 6.21%. From the results, the scalability of SMTApproxMC is better than ApproxMC2, as the former solved 15 instances whereas the latter solved 11 instances. Therefore, MCMC-Flat$^{1/2}$ is the best, with respect to scalability.
- In terms of accuracy, MCMC-Flat$^{1/2}$ is better than SMTApproxMC and ApproxMC2. Among 13 instances for which the exact volume is known by Volce+LattE (see Table 5), MCMC-Flat$^{1/2}$ returns best results on 8 instances, and ApproxMC2 does the best on the other 5 instances, in terms of relative error. The accuracy of SMTApproxMC is not competitive to either MCMC-Flat$^{1/2}$ or ApproxMC2. The variance in relative errors of ApproxMC2 is better than that of MCMC-Flat$^{1/2}$, as we can observe from Table 6 that on instances where ApproxMC2 returns none-zero results the relative error of ApproxMC2 is better than that of MCMC-Flat$^{1/2}$. On the other hand, ApproxMC2 seems to have big difficulty in working on instances having volume greater than 1.0×10^6, as it takes relatively long time on instances 5-10-1, 6-10-1, and 7-10-1.
- In terms of efficiency, MCMC-Flat$^{1/2}$ is significantly better than SMTApproxMC and ApproxMC2. As can be seen from the bottom row in Table 6, over the ten instances (5-10-1, 5-10-2, 5-10-4, 5-10-5, 5-20-1, 5-20-2, 5-20-3, 5-20-4, 5-20-5, 6-10-1) on which all the three methods return none-zero results, the averaged time costs of MCMC-Flat$^{1/2}$, SMTApproxMC, and ApproxMC2 are 344.5, 955.80, and 463.33, respectively. Obviously, MCMC-Flat$^{1/2}$ is the fastest.

- Regarding the ability of obtaining samples from the solution area for large size instances, MCMC-Flat$^{1/2}$ is superior to the direct Mont Carlo method in [24]. Note that, all the test instances are took from [24], and the direct Mont Carlo method in [24] is reported to give results of RE of 100% on all instances with 7 variables (We failed to run that method due to that its call on Matlab 2010b leads to segment faults). That is, the direct Mont Carlo method failed to sample any solution when the problem size is large. In contrast, MCMC-Flat$^{1/2}$ is able to reach the solution area on all the tested instances with 7 variables.

A final note for this part of experiments is that MCMC-Flat$^{1/\text{t-pv}}$ is not good at these instances. Actually, we also run several tests for MCMC-Flat$^{1/\text{t-pv}}$ on instances with 5 variables. However, its success rate is very low. The underlying reason is to be explored in our future work.

6. Conclusions

We have proposed a flat histogram strategy-based Markov Chain Monte Carlo (MCMC) method for computing the volume of possibly intersected polytopes in a space. Specifically, we have introduced three implementations of the MCMC method, which have different performance in terms of efficiency and accuracy. To analyze the performance of our methods, extensive experiments against two exact and two approximate model counters were carried out on different types of test problems. From our tests on SMT(LIA) instances, MCMC-Flat$^{1/\text{t-pv}}$ is showed to be able to reach good accuracy. From our test on convex bodies (where LIA constraints are joined by conjunctions), MCMC-Flat$^{1/2}$ is competitive with the state-of-the-art and shows a better scalability.

Our future work is to investigate theoretical properties of MCMC-Flat$^{1/2}$ to obtain a guarantee on the quality of the returned values and to improve MCMC-Flat$^{1/\text{t}}$ for the model-counting problem of convex bodies.

Author Contributions: Formal analysis, H.L. and D.C.; Methodology, D.C.; Software, W.G. and Q.Z.; Validation, W.G.; Writing: review and editing, H.L. and Q.Z.

Funding: This work was supported by Open Research Foundation of China's Science and Technology on Aerospace Flight Dynamics Laboratory (Beijing Aerospace Control Center) (2014afdl002), and National Natural Science Foundation of China under grant number [61103136, 61502355].

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Barrett, C.; Sebastiani, R.; Seshia, S.A.; Tinelli, C. Satisfiability modulo theories. In *Handbook of Satisfiability*; Biere, A., Ed.; IOS Press: Amsterdam, The Netherlands, 2009; Volume 185, pp. 825–885, ISBN 978-1-58603-929-5.
2. Dyer, M.E.; Frieze, A.M.; Kannan, R. A random polynomial time algorithm for approximating the volume of convex bodies. In Proceedings of the 21st Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, 14–17 May 1989; pp. 375–381.
3. Dyer, M.E.; Frieze, A.M. On the complexity of computing the volume of a polyhedron. *SIAM J. Comput.* **1988**, *17*, 967–974. [CrossRef]
4. Hahn, E.M.; Hartmanns, A.; Hermanns, H.; Katoen, J. A compositional modelling and analysis framework for stochastic hybrid systems. *Form. Methods Syst. Des.* **2013**, *43*, 191–232. [CrossRef]
5. Immler, F. Verified reachability analysis of continuous systems'. In Proceedings of the Tools and Algorithms for the Construction and Analysis of Systems—21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, 11–18 April 2015; pp. 37–51.
6. Hadarean, L.; Barrett, C.; Reynolds, A.; Tinelli, C.; Deters, M. Fine grained SMT proofs for the theory of fixed-width bit-vectors. In Proceedings of the 20th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning, LPAR-20, Suva, Fiji, 24–28 November 2015; pp. 340–355.

7. Anand, S.; Burke, E.K.; Chen, T.Y.; Clark, J.; Cohen, M.B.; Grieskamp, W.; Harman, M.; Harrold, M.J.; McMinn, P. An orchestrated survey of methodologies for automated software test case generation. *J. Syst. Softw.* **2013**, *86*, 1978–2001. [CrossRef]
8. Zhang, J. Specification analysis and test data generation by solving Boolean combinations of numeric constraints. In Proceedings of the First Asia-Pacific Conference on Quality Software, Hongkong, China, 30–31 October 2000; pp. 267–274.
9. Ge, C.; Ma, F.; Zhang, P.; Zhang, J. Computing and estimating the volume of the solution space of SMT (LA) constraints. *Theor. Comput. Sci.* **2018**, *743*, 110–129. [CrossRef]
10. Yan, J.; Zhang, J. An efficient method to generate feasible paths for basis path testing. *Inform. Process. Lett.* **2008**, *107*, 87–92. [CrossRef]
11. Zhang, J.; Wang, X. A constraint solver and its application to path feasibility analysis. *Int. J. Softw. Eng. Knowl. Eng.* **2001**, *11*, 139–156. [CrossRef]
12. Li, Y.T.S.; Malik, S. Performance analysis of embedded software using implicit path enumeration. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **1997**, *16*, 1477–1487. [CrossRef]
13. Ammons, G.; Larus, J.R. Improving data-flow analysis with path profiles. *ACM Sigplan Not.* **2004**, *39*, 568–582. [CrossRef]
14. Wang, F.; Landau, D.P. Efficient, multiple-range random walk algorithm to calculate the density of states. *Phys. Rev. Lett.* **2001**, *86*. [CrossRef] [PubMed]
15. Whitmer, J.K.; Chiu, C.C.; Joshi, A.A.; De Pablo, J.J. Basis function sampling: A new paradigm for material property computation. *Phys. Rev. Lett.* **2014**, *113*. [CrossRef] [PubMed]
16. Whitmer, J.K.; Fluitt, A.M.; Antony, L.; Qin, J.; McGovern, M.; De Pablo, J.J. Sculpting bespoke mountains: Determining free energies with basis expansions. *J. Chem. Phys.* **2015**, *143*. [CrossRef] [PubMed]
17. Zablotskiy, S.V.; Ivanov, V.A.; Paul, W. Multidimensional stochastic approximation Monte Carlo. *Phys. Rev. E* **2016**, *93*. [CrossRef] [PubMed]
18. Ma, F.; Liu, S.; Zhang, J. Volume computation for boolean combination of linear arithmetic constraints. In Proceedings of the Automated Deduction—CADE-22, 22nd International Conference on Automated Deduction, Montreal, QC, Canada, 2–7 August 2009; pp. 453–468.
19. Chakraborty, S.; Meel, K.S.; Mistry, R.; Vardi, M.Y. Approximate Probabilistic Inference via Word-Level Counting. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; pp. 3218–3224.
20. Meel, K.S.; Vardi, M.Y.; Chakraborty, S.; Fremont, D.J.; Seshia, S.A.; Fried, D.; Ivrii, A.; Malik, S. Constrained Sampling and Counting: Universal Hashing Meets SAT Solving. In Proceedings of the AAAI'16 Workshop: Beyond NP, Phoenix, AZ, USA, 12–13 February 2016; pp. 344–351.
21. Chakraborty, S.; Meel, K.S.; Vardi, M.Y. Algorithmic improvements in approximate counting for probabilistic inference: From linear to logarithmic SAT calls. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, New York, NY, USA, 9–15 July 2016; pp. 3569–3576.
22. Ermon, S.; Gomes, C.P.; Selman, B. Computing the density of states of Boolean formulas. In Proceedings of the Principles and Practice of Constraint Programming—CP 2010—16th International Conference, Scotland, UK, 6–10 September 2010; pp. 38–52.
23. De Loera, J.A.; Hemmecke, R.; Tauzer, J.; Yoshida, R. Effective lattice point counting in rational convex polytopes. *J. Symb. Comput.* **2004**, *38*, 1273–1302. [CrossRef]
24. Zhou, M.; He, F.; Song, X.; He, S.; Chen, G.; Gu, M. Estimating the Volume of Solution Space for Satisfiability Modulo Linear Real Arithmetic. *Theor. Comput. Syst.* **2015**, *56*, 347–371. [CrossRef]
25. Belardinelli, R.E.; Pereyra, V.D. Fast algorithm to calculate density of states. *Phys. Rev. E* **2007**, *75*. [CrossRef] [PubMed]
26. Jacob, P.E.; Ryder, R.J. The Wang-Landau algorithm reaches the flat histogram criterion in finite time. *Ann. Appl. Probab.* **2014**, *24*, 34–53. [CrossRef]