SAT-Metropolis: Combining Markov Chain Monte Carlo with SAT/SMT Sampling

Maja Aaslyng Dall
IT University of Copenhagen, Denmark

Raúl Pardo
IT University of Copenhagen, Denmark

Thomas Lumley
University of Auckland, New Zealand

Andrzej Wąsowski
IT University of Copenhagen, Denmark

Abstract

Probabilistic inference via Markov Chain Monte Carlo (MCMC) is at the core of statistical analysis and has a myriad of applications. However, probabilistic inference in the presence of hard constraints, so constraints that must hold with probability one, remains a difficult task. The reason is that hard constraints make the state space of the target distribution sparse, and may even divide it into disjoint areas separated by probability-zero states. As a consequence, the random walk performed by MCMC algorithms fails to effectively sample from the complete set of states in the target distribution. In this paper, we propose the use of SAT/SMT sampling to adapt a classic and widely used MCMC algorithm, namely Metropolis sampling. We use SAT/SMT samplers as proposal distributions. In this way, the algorithm ignores probability-zero states. Our method, SAT-METROPOLIS, effectively works in problems with multivariate polynomial hard constraints where regular Metropolis fails. We evaluate the convergence and scalability of SAT-METROPOLIS using three different state-of-the-art SAT/SMT samplers: SPUR, CMSGen, and MegaSampler. The evaluation shows how different features of the SAT/SMT sampling tools affect the effectiveness of probabilistic inference. We conclude that SAT/SMT sampling is a viable and promising technology for probabilistic inference under hard constraints.

2012 ACM Subject Classification Mathematics of computing

Keywords and phrases SAT/SMT sampling, Probabilistic inference, Markov Chain Monte Carlo

Digital Object Identifier 10.4230/LIPIcs.SAT.2025.12

Supplementary Material *Software*: `https://github.com/itu-square/sat-metropolis` [5]
archived at `swh:1:dir:bfc2169353301af051d1af4488050fcfba06a849`

Funding This work was partially funded by DIREC (Digital Research Centre Denmark), a collaboration between the eight Danish universities and the Alexandra Institute supported by the Innovation Fund Denmark.

1 Introduction

Probabilistic inference with Markov Chain Monte Carlo (MCMC) is at the core of statistical analysis with a myriad of applications in domains such as biology, medicine and physics. Probabilistic inference for problems with soft constraints has seen notable progress [14], but inference in the presence of hard constraints remains a difficult task [12]. Yet hard constraints appear in a wide range of real-world applications such as road routing, genetics or computer security; just to mention a few of them.

Probabilistic inference via MCMC is a technique to sample data or compute expected values from distributions whose explicit density functions are unknown, but we can evaluate them up to a multiplicative factor. For instance, in probabilistic Bayesian inference $P(x|y) =$

$P(y|x)P(x)/P(y)$, the term $P(y|x)P(x)$ (the likelihood and the prior) can be evaluated at any point of the state space of the target distribution, but computing the normalizing factor $1/P(y)$ is infeasible in practice. MCMC algorithms tackle this problem by following a random walk on the state space guided by the term $P(y|x)P(x)$ [4].

Unfortunately, MCMC algorithms often fail in the presence of hard constraints that make the state space sparse or even partition it into disjoint parts of positive probability separated by areas of zero probability. Since MCMC algorithms reject zero probability samples, they struggle to explore such distributions completely, even though they are correct in the limit. Classic algorithms like Metropolis-Hastings [19, 11] and Gibbs sampling [9] cannot effectively reach all positive probability states under these conditions [12]. They fail to converge or must be run for a prohibitively long time.

Existing work addressing this problem focuses on the design of sampling algorithms for specific types of probabilistic inference problems [12, 13, 22]. For instance, Hazelton and coauthors propose a method to sample from a $\mathbb{Z}$-polytope defined by inverse linear counting problems – i.e., problems of the form $\mathbf{y} = \mathbf{A}\mathbf{x}$ where $\mathbf{y}$ is a vector of counts, $\mathbf{A}$ is a contingency matrix and $\mathbf{x}$ the vector of variables of interest [12]. Although efficient, their method is limited to problems of this particular form. Poon and Domingos pioneered using SAT samplers in probabilistic inference. They propose an inference method for problems with propositional constraints that encode relational properties [22]. They create a slice sampler using SAT sampling to perform probabilistic inference over Markov Logic Networks [23]. Unfortunately, this requires encoding probabilistic models as Markov Logic Networks, a model not used commonly in statistics and limited when continuous variables are needed.

In this work, we extend a classic MCMC method with SAT/SMT sampling to support probabilistic inference under mixed soft and hard constraints. In particular, we use SAT/SMT samplers as proposal distributions for the Metropolis algorithm [19]. Our method works on standard probabilistic models widely used in statistics. The theory of discrete integer linear arithmetic with multiplication allows us to handle multivariate polynomial hard constraints. We evaluate it using multiple modern SAT/SMT samplers that provide an excellent basis to "jump" between the set of states with positive probability and avoid probability-zero regions of the state space by design. Finally, we discuss what properties of SAT/SMT samplers help to effectively address probabilistic inference. In summary, our contributions are:

- A MCMC algorithm for probabilistic inference problems under mixed soft and hard constraints, supporting multivariate polynomial hard constraints,
- An implementation of the algorithm using three different SAT/SMT samplers,
- An evaluation of convergence of the inference for different encodings and samplers,
- An evaluation of the scalability of the algorithm in realistic case studies.

2 SAT/SMT sampling

Let T be a theory over a set of variables with some concrete domain, logical symbols, predicate symbols, function symbols and a fixed interpretation. Let φ denote a formula in T and $\mathbf{x}$ be a vector with the variables appearing in φ. A satisfying assignment is an assignment of domain values to the variables in $\mathbf{x}$ such that φ evaluates to true. We use $\{\mathbf{x} \mid \mathbf{x} \models \varphi\}$ to denote the set of the satisfying assignments of φ.

A Satisfiability Modulo Theory (SMT) sampler is an algorithm that generates samples from the set of solutions of a formula φ, denoted $\mathbf{x}_i \xleftarrow{\$} \mathcal{D}(\{\mathbf{x} \mid \mathbf{x} \models \varphi\})$, according to a probability distribution $\mathcal{D}$. When φ is a Boolean formula, we call it a SAT sampler. A SAT/SMT sampler is uniform if the probability of obtaining any satisfiable assignment is

$1/|\{\mathbf{x} \mid \mathbf{x} \models \varphi\}|$, denoted $\mathcal{U}(\{\mathbf{x} \mid \mathbf{x} \models \varphi\})$. A SAT/SMT sampler is *almost* uniform if the probability of sampling any satisfiable assignments is at least $1 - \epsilon/|\{\mathbf{x} \mid \mathbf{x} \models \varphi\}|$ and at most $1 + \epsilon/|\{\mathbf{x} \mid \mathbf{x} \models \varphi\}|$ for some $\epsilon > 0$.

In this paper, we use the theory T_{MIA} of integer linear arithmetic extended with multiplication to handle multivariate polynomial constraints. A T_{MIA} formula includes logical connectives $\{\neg, \wedge\}$ (and derived), predicates $\{>, =\}$ (and derived) and arithmetic operations $\{+, -, *\}$, but no division. The interpretation of the symbols is standard. Variables have finite integer domains. We use bit-blasting to encode formulae in Boolean logics for SAT-samplers.

3 Metropolis Algorithm for Bayesian Probabilistic Inference

The Metropolis algorithm samples from a *target* distribution $P(\mathbf{y}) = P^*(\mathbf{y})/Z$ by using evaluations of $P^*(\mathbf{y})$, a function different by an unknown constant factor $1/Z$. The vector $\mathbf{y}$ denotes the list of parameters for the target distribution. When applied to Bayesian inference, the target distribution is the posterior defined by the Bayes rule $P(\mathbf{x} \mid D) = P(D \mid \mathbf{x})P(\mathbf{x})/Z$. The term $P^*(\mathbf{y})$ is instantiated as $P(D \mid \mathbf{x})P(\mathbf{x})$, i.e., the *likelihood* times the *prior*. When no ambiguity arises, we write $P(\cdot)$ to denote a probability measure on a measurable space that is compatible with its parameters.

The likelihood function $P(D \mid \mathbf{x})$ is used to model soft and hard constraints in the underlying probabilistic model. Let $D = \{c_1, \ldots, c_n\}$ be a set of events that model constraint parameters, e.g., observations from real-world data or logical formulae. The set D is commonly known as a *set of observations*. The likelihood is defined as $P(D \mid \mathbf{x}) = \prod_i P(c_i \mid \mathbf{x})$. A hard constraint is a constraint with 0–1 likelihood. For example, for a model with parameter x, we can impose $x > 0$, i.e., $c_i = \{x \mid x > 0\}$, by defining $P(x > 0 \mid \mathbf{x}) = \mathbf{1}_{x>0}(x)$ where $\mathbf{1}_{x>0}(x)$ is the indicator function; its value is 1 if $x > 0$ and 0 otherwise. Note that the posterior probability of values of x that do not satisfy the constraint is zero, $P(x < 0 \mid \mathbf{x}) = 0$. Soft constraints, on the other hand, allow for specifying noisy properties in the model. For example, consider that value $c_i = v$ has been collected using a recording device with a known measurement error of $\sigma_v \in \mathbb{R}_{>0}$. It is possible to model the measurement error using a soft constraint $P(v \mid x) = \mathcal{N}(v; \mu = x, \sigma = \sigma_v)$. In this example, we use a Gaussian distribution ($\mathcal{N}$) to model that values of x close to v have high likelihood, and other values, though less likely, are also possible; due to the measurement error of the recording device.

To sample values from the posterior, the Metropolis algorithm takes a random walk over the parameter space starting in a random point $\mathbf{x}$ with a positive posterior probability, $P(\mathbf{x} \mid D) > 0$. Given a state $\mathbf{x}_i$, a *proposal distribution*, $q(\mathbf{x}^* | \mathbf{x})$, is used to sample a candidate next state in the random walk. The Metropolis algorithm requires $q(\cdot)$ to be symmetric, i.e., $q(\mathbf{x}^* | \mathbf{x}) = q(\mathbf{x} | \mathbf{x}^*)$. An extension of the algorithm, Metropolis-Hastings, relaxes this constraint [11, 16]. The probabilistic inference presented in Section 5 does not require asymmetric proposal distributions. Given a state $\mathbf{x}_i$ and proposal $\mathbf{x}^*$, the next state $\mathbf{x}_{i+1}$ is selected according to the following *acceptance ratio*:

$$r(\mathbf{x}, \mathbf{x}^*) = \frac{P(\mathbf{x}^* \mid D)}{P(\mathbf{x} \mid D)} = \frac{P(D \mid \mathbf{x}^*)P(\mathbf{x}^*)/Z}{P(D \mid \mathbf{x})P(\mathbf{x})/Z} = \frac{P(D \mid \mathbf{x}^*)P(\mathbf{x}^*)}{P(D \mid \mathbf{x})P(\mathbf{x})} \tag{1}$$

In particular, $\mathbf{x}_{i+1}$ is selected as

$$\mathbf{x}_{i+1} = \begin{cases} \mathbf{x}^* & \text{with probability } \min(1, r(\mathbf{x}, \mathbf{x}^*)) \\ \mathbf{x} & \text{otherwise.} \end{cases} \tag{2}$$

That is, the proposed state is accepted with probability $\min(1, r(\mathbf{x}, \mathbf{x}^*))$. Otherwise, the proposal is rejected and the random walk stays in the current state.

If the proposal distribution is positive, i.e., $q(\mathbf{x}^* \mid \mathbf{x}) > 0$ for all $\mathbf{x}, \mathbf{x}^*$, the probability distribution induced by a collection of n samples $\{\mathbf{x}_i\}_{i=1..n}$ is known to converge to $P(\mathbf{x}|D)$ as $n \to \infty$ [11, 16]. The theoretical guarantees aside, the choice of proposal distribution drastically affects the speed of convergence in practice. Much of the work on MCMC addresses designing efficient (in terms of convergence speed) proposal distributions such as Gibbs sampling or Hamiltonian Monte Carlo (HMC) [4], predominantly for continuous variables. Despite these advances, it remains a challenge to sample from posterior distributions with hard discrete constraints.

4 The Problem with Hard Constraints in Markov Chain Monte Carlo

The popular MCMC algorithms, such as Metropolis, Metropolis-Hastings, Gibbs sampling and HMC, are not suitable for sampling from distributions with hard constraints. The problem is that the proposal distribution is unaware of the hard constraints and, consequently, proposes states violating them. Such proposals are promptly rejected – for any $\mathbf{x}^*$ violating hard constraints, we have $P(\mathbf{x}^* \mid D) = 0$, the acceptance ratio $r(\mathbf{x}, \mathbf{x}^*) = 0/P(D|\mathbf{x})P(\mathbf{x}) = 0$, and the probability of accepting $\mathbf{x}^*$ is $\min(1, 0) = 0$. Thus, the random walk gets stuck in its current state ($\mathbf{x}$).

▶ **Example 1.** Consider a simple reconstruction problem (cf. Section 6). Two integers (x_1, x_2) in the closed interval $[0, 10]$ are known to sum to eleven. The task is to determine the most likely values of x_1 and x_2 given the sum. We pose this as a Bayesian inference problem.

$$
\begin{aligned}
x_1 &\sim \text{DiscUni}(0, 10) \\
x_2 &\sim \text{DiscUni}(0, 10) \\
y &:= x_1 + x_2 \\
y &= 11
\end{aligned}
$$

The first two lines model the prior belief regarding valuations of x_1 and x_2. As we use a discrete uniform distribution in the interval $[0, 10]$, all the values from 0 to 10 are equally likely *a priori*. The third line models the sum of the variables. The last line states the hard constraint on the sum. The posterior $P(x_1, x_2 \mid y = 11)$ gives the answer to the reconstruction problem.

To use Metropolis to estimate this posterior, we need a proposal distribution. A common choice is a standard normal distribution $\mathcal{N}(0, 1)$. This is, e.g., the default proposal distribution in the PyMC library [1]. For discrete random variables, the proposal values from the normal distribution are discretized to the closest integer. Thus, given a state for x_i, the next state is computed as $[x_i + \nu]$ where $\nu_i \sim \mathcal{N}(0, 1)$ and $[\cdot]$ denotes rounding to the closest integer. For a state (x_1, x_2) we write $q(x_1^*, x_2^*|x_1, x_2)$ for the probability of proposing state (x_1^*, x_2^*).

To appreciate the detrimental effect of the hard constraints on Metropolis, consider the probability of transitioning to a state satisfying $y = 11$ from $x_1 = 1, x_2 = 10$. The most likely proposal is to remain in the current state, as the mode of the normal distribution equals its mean. Considering the rounding to integers, the probability of sampling the current state is $q((x_1, x_2) \mid (x_1, x_2)) \approx 0.14$. The next most likely states, those within L1 distance of one from the current state $||(x_1', x_2') - (x_1, x_2)||_1 = 1$, are proposed with $q((x_1', x_2')|(x_1, x_2)) \approx 0.09$. New states within distance of two are proposed with probability < 0.06. In general, new states satisfying $y = 11$ distinct from the current state $(1, 10)$ *at any distance* are proposed with probability just above 0.06. Consequently, the probability of staying in the current state is Q is $\approx 1 - 0.06 = 0.94$. It is approximately 16 times more likely to remain in the

current state than to move to a new state satisfying the hard constraint. In practice, this leads to very low sampling speed and slow convergence; as the algorithm spends most time being stuck in the same state. ⌟

This problem gets much worse for sparse constraints when the state space increases. It is intractable to randomly sample a satisfying solution to a hard constraint. As we show in Section 6, problems do not have to get very complex for the sampler to get stuck completely.

5 SAT/SMT-Powered Metropolis

The key idea to address the above problems is to use SAT/SMT sampling as the proposal distribution. We present, SAT-METROPOLIS, our modification of the Metropolis sampler as Algorithm 1 below. The algorithm takes four arguments: (i) a possibly not normalized probability density P to approximate, (ii) a set of hard constraints Φ, (iii) an initial state for the random walk, and (iv) the number of samples n to generate. As common for the sampling algorithms, the initial state must have positive probability $P(\mathbf{x}_{\text{init}}) > 0$, which in this case means that it must also satisfy the hard constraints. The output is a set of samples $\{x_i\}_{i=1}^{n}$. When used for Bayesian inference, P is instantiated as likelihood times prior, i.e., the unnormalized posterior distribution (cf. Section 3).

We begin in the provided initial state (line 2). The main novelty is found in line 4. The proposal distribution, $\mathcal{U}(\{\mathbf{x} \mid \mathbf{x} \models \Phi\})$ is a uniform distribution over all states that satisfy the hard constraints. A uniform proposal distribution over non-zero probability states reduces autocorrelation among samples, which helps reducing bias and improving speed of convergence [17]. Furthermore, it is compatible with the inner workings of existing SAT/SMT samplers, which work by producing batches of uniform samples. However, the proposal distribution in Metropolis can be non-uniform and conditional on a state. This type of proposal distributions can also achieve low autocorrelation, if they are properly configured. Since SAT/SMT sampling from conditional and non-uniform is not supported by existing tools, we do not consider it in this version of SAT-METROPOLIS. The remaining steps in Algorithm 1 proceed as usual in Metropolis (cf. Section 3). Lines 5 and 6 compute the acceptance probability. Lines 7 to 11 select the new proposal $\mathbf{x}^*$ with probability α. Otherwise we remain in the current state. The procedure is repeated n times.

The proposal distribution in SAT-METROPOLIS is symmetric and positive; as it is independent of the current state and it assigns non-zero probability to any pair of states. Therefore, it trivially follows that SAT-METROPOLIS converges in the limit (cf. Section 3). However, the practical use of this algorithm will depend on how efficiently and uniformly we can sample from the proposal distribution.

We implemented SAT-METROPOLIS in Python, using three engines for sampling solutions to hard constraints: SPUR [2], CMSGen [10], and MegaSampler [21]. We selected these engines as they vary in: the uniformity of generated samples, speed of sampling and amount of pre-processing. These features are the main factors that affect the speed of convergence and scalability of our algorithm (cf. section 6). Furthermore, these engines are at the forefront of the research in SAT/SMT sampling. The implementation is publicly available at `https://github.com/itu-square/sat-metropolis`.

Figure 1 gives a high level overview of the implementation. The implementation takes the target distribution as a Python function from the domain of the parameter space ($\mathbf{x} \in \mathcal{X}^m$) to reals $\mathbb{R}$. This distribution specifies the soft constraints. To specify hard constraints, the user should provide an SMT problem in abstract syntax constructed using the Z3 API [6].

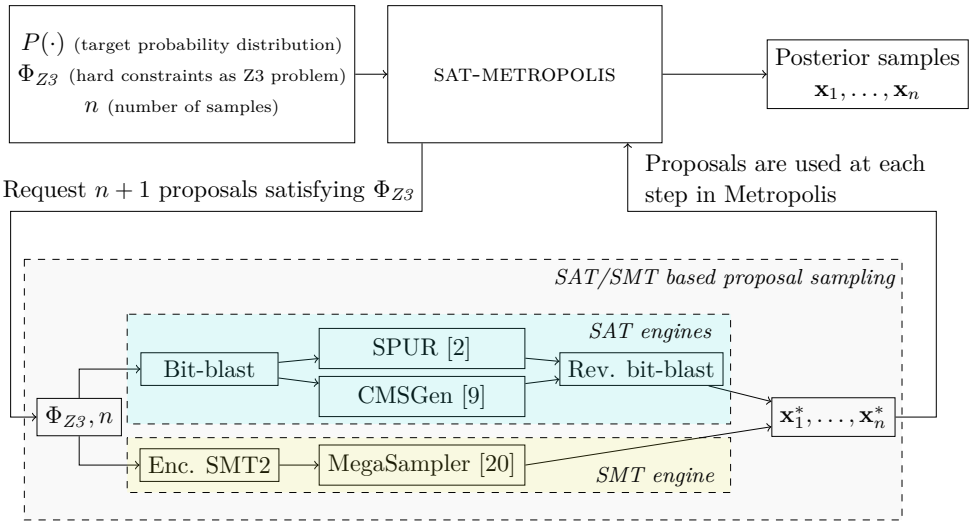


Figure 1 An overview of the SAT-METROPOLIS implementation.

The implementation starts by requesting $n+1$ proposal samples from an SMT/SAT sampling engine. For simplicity, we use the first sample as the initial state $\mathbf{x}_{\text{init}}$. This assumes that the target distribution has positive density for all states satisfying the hard constraints, which is the case for the problems we consider. Then, it proceeds as shown in Algorithm 1 with the only difference that the sampling step on line 4 retrieves a sample from the received proposals. This is equivalent to the behavior specified by Algorithm 1, as proposals are sampled independently of the current state. Depending on the type of engine used for generating samples, there are requirements on the input Z3 problem and pre-processing steps. We discuss them below. However all the engines used allow us to handle problems with polynomial hard constraints that was not possible for previous methods for probabilistic inference.

Algorithm 1 Metropolis algorithm with SAT/SMT uniform sampling as proposal distribution.

Input: Target probability density P (not normalized)
Set of hard constraints Φ
Initial state $\mathbf{x}_{\text{init}}$ such that $P(\mathbf{x}_{\text{init}}) > 0$
Number of samples n
Output: Set of samples $\mathbf{x}_1, \ldots, \mathbf{x}_n$

1: **procedure** SAT-METROPOLIS($P, \Phi, \mathbf{x}_{\text{init}}, n$)
2: $\quad\mathbf{x}_1 \leftarrow \mathbf{x}_{\text{init}}$
3: $\quad$**for** $i \leftarrow 1$ to n **do**
4: $\quad\quad\mathbf{x}^* \xleftarrow{\$} \mathcal{U}(\{\mathbf{x} \mid \mathbf{x} \models \Phi\})$
5: $\quad\quad r \leftarrow P(\mathbf{x}^*)/P(\mathbf{x}_i)$
6: $\quad\quad\alpha \leftarrow \min(1, r)$
7: $\quad\quad u \xleftarrow{\$} \mathcal{U}(0, 1)$
8: $\quad\quad$**if** $u \leq \alpha$ **then**
9: $\quad\quad\quad\mathbf{x}_{i+1} \leftarrow \mathbf{x}^*$
10: $\quad\quad$**else**
11: $\quad\quad\quad\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i$
12: $\quad\quad$**end if**
13: $\quad$**end for**
14: **end procedure**

SMT-sampling engine. For MegaSampler, the Z3 problem is specified in T_{MIA} over integers (`Int` in Z3). This allows for modeling discrete random variables with infinite support. We convert the problem into the SMT2 format using the Z3 built-in facility and feed it into MegaSampler, requesting to generate n samples. As before, we parse the output file with the samples and convert it into a format that can be used by SAT-METROPOLIS.

SAT-sampling engines. To use the SAT engines SPUR or CMSGen, we require that the constraint is specified in T_{MIA} (Section 2) using bit-vector variables. The user provides the bit-lengths for the variables and we use Z3 to translate the constraints into the Boolean space to obtain a CNF problem [15] and eventually DIMACS. We combine three Z3 tactics to carry out this process: `'simplify'`, `'bit-blast'`, and `'tseitin-cnf'`. The first tactic, `'simplify'`, produces a equisatisfiable formula with the goal of making it easier to tackle by the solver. The second tactic, `'bit-blast'`, performs standard bit-blasting. The last tactic, `'tseitin-cnf'`, performs a Tseiting's transformation that results in a CNF formula. We use our own script to turn the output of Z3 into DIMACS format. Then, we instruct SPUR or CMSGen to generate n samples. The output files with the samples are parsed, reverse bit-blasted, and converted into a format that our SAT-METROPOLIS algorithm can use.

Due to the bit-vector encoding, SAT engines can only handle discrete distributions with finite support. However, this is does not affect the results and conclusions of our evaluation (cf. Section 6), as we consider problems with finite support and bit-vectors of sufficient length.

6 Evaluation

We evaluate SAT-METROPOLIS by answering the following research questions.

RQ1 How does the SAT/SMT sampling method affect the convergence of SAT-METROPOLIS?

RQ2 How big problems can be handled by SAT-METROPOLIS with the contemporary SAT/SMT samplers?

Table 1 lists the probabilistic inference problems used in the evaluation. To answer **RQ1**, we use two database reconstruction problems, DB CACM and NZ Stats. DB CACM was previously introduced as an example of database reconstruction using SMT solvers [8]. NZ Stats uses data published by Statistics New Zealand. We use the data and the information on the anonymization algorithms used in these cases, to reason about beliefs for the possible reconstructions. These problems are simple enough to solve analytically, and we use the analytical solution as the ground truth for evaluation of convergence. Note that evaluating the convergence of the estimation on large problems without ground truth is not possible.

To answer **RQ2**, we use the last two problems in the table (Books and Roads). Books and Roads are examples of problems with realistic sizes regarding the number of variables and constraints and domain sizes. They had been used to evaluate specialized methods to solve probabilistic inference under linear constraints [12] (Section 7). Determining how SAT-METROPOLIS performs on them is a good indicator on whether modern SAT/SMT-sampling can handle realistic tasks. We consider multiple versions of these two problems by systematically adjusting their size; with the objective of finding the limits of scalability.

In general, we consider problems that range from 10 to 72 variables and 2 to 141 constraints. The formulae include polynomial and linear constraints (cf. Table 1). The structure of the formulae is problem dependent and we describe it the sections below. The SAT version of these problems ranges from 148 to 10252 variables and from 499 to 47299 clauses. Clauses contain from 2 to 4 literals and the clause-to-var ratio ranges from 3.06 to 5.08. Table 4 in Appendix A contains the complete list of problems that we consider in this paper.

Table 1 Inference problems used for evaluation.

Problem	#vars	#constraints	Hard constraints type
Database reconstruction (DB CACM)	10	20	Polynomial
Database reconstruction (NZ Stats)	38	141	Linear
Highway traffic flow (Roads)	28	7	Linear
Book ratings (Books)	72	42	Linear

The experiments were run on Ubuntu 24.04 with an Intel Core i7-1355U CPU and 16GB RAM. The experiment package is available at `https://github.com/itu-square/sat-metropolis`.

6.1 Convergence

We evaluate the convergence of SAT-METROPOLIS using the SAT/SMT sampling engines: SPUR, CMSGen and MegaSampler. The goal of the evaluation is to determine how the different features of these engines affect convergence. SPUR ensures uniformity by design. CMSGen does not guarantee uniformity, but it has been shown empirically to produce near uniform samples. Technically, MegaSampler is not a sampler in the statistical sense, but a solution enumerator. It never produces the same state twice. We use it to evaluate the strategy of reweighing the set of unique samples using the unormalized target distribution. To evaluate convergence, we use two database reconstruction problems, DB CACM and NZ Stats. The former includes polynomial constraints and the latter only linear constraints. This allows us to observe the effect of different types of constraints on the algorithm.

We compute the mean absolute difference between the ground truth and the SAT-METROPOLIS estimation of probability for each outcome for each variable. Let X be the set of variables in a problem and $x \in X$ be a problem variable with domain $\mathrm{dom}(x)$. Let $P(x)$ the target distribution and $\hat{P}(x)$ its approximation by SAT-METROPOLIS. The absolute mean difference is defined as:

$$\mu_{\mathrm{diff}}(X) = |X|^{-1} \sum_{x \in X} |\mathrm{dom}(x)|^{-1} \sum_{v \in \mathrm{dom}(x)} |P(x = v) - \hat{P}(x = v)| \tag{3}$$

where $|\cdot|$ is overloaded to mean cardinality for sets and absolute value for reals. When μ_{diff} tends to 0 as the number of samples increases, the SAT-METROPOLIS converges to the right histogram. When μ_{diff} does not approach zero, we conclude no convergence or a biased convergence (convergence to a wrong value). We generate 20000 samples for each experiment. We executed every experiment 3 times and report the mean.

DB CACM. A database contains records of five individuals with ages $(x_0, \ldots, x_4)$ and whether they identify as male $(m_0, \ldots, m_4)$. Ages x_i are integers in the interval $[0, 125]$, and variables m_i are Booleans. A data analyst releases the following information about the database: the median age is 30, the mean age is 38, the number of males is 3, and the mean age of males is 44. The database reconstruction problem is to determine the values of variables x_i and m_i that are consistent with the released data. This problem can be addressed using probabilistic inference [20]. We assign prior distributions to variables x_i and m_i modeling attacker knowledge before observing the published data. Then, using probabilistic inference, we compute the posterior distribution given the released data encoded as hard constraints, which models the attacker knowledge after observing the released data. We model an

Table 2 A simplified database over South Head, NZ demographics.

Age range	0-4	5-9	10-14	15-19	20-24	25-29	30-34	35-39	40-44	45-49	50-54	55-59	60-64	65-69	70-74	75-79	80-84	85-89	90+
Male	3	6	6	12	9	3	9	9	6	6	15	6	3	6	3	3	0	0	0
Female	6	0	3	6	3	6	3	9	3	6	9	6	9	3	3	3	0	0	0
Total	15	3	12	15	15	12	9	18	9	12	21	12	12	12	6	0	0	0	0

attacker who considers the all ages equally possible for all individuals, and also assumes that individuals are male with probability 0.5. Thus, we use uniform distributions over all possible values as prior distributions, i.e., $x_i \sim \text{DiscUni}(0, 125)$ and $m_i \sim \text{DiscUni}(0, 1)$, where $0, 1$ denote false and true respectively.

$$0 \leq x_0 \leq x_1 \leq x_2 = 30 \leq x_3 \leq x_4 \leq 125 \qquad \textit{Age ranges and median} \qquad (4)$$

$$\textstyle\sum_i x_i = 38 \cdot 5 \qquad \textit{Mean age of all individuals} \qquad (5)$$

$$\textstyle\sum_i m_i = 3 \qquad \textit{Num. individuals who identify as male} \qquad (6)$$

$$\textstyle\sum_i m_i \cdot x_i = 44 \cdot 3 \qquad \textit{Mean age of 3 males} \qquad (7)$$

We use SAT-METROPOLIS to generate samples from $P(x_0, \ldots, x_4, m_0, \ldots, m_4 | (4), (5), (6), (7))$.

NZ Stats. This is a case study similar to the one above, but based on data from the statistics agency in New Zealand, NZ Stats. Table 2 shows a set of anonymized counts of the number of males and females in the South Head area in New Zealand. All values in the table are multiples of three, as NZ Stats uses a method known as "round to base 3" to anonymize counts. Given a count value v, if v is a multiple of three, release v. Otherwise consider the multiples of three below and above v and return the closer one with probability p and the further one with probability $1 - p$.

Let x_i be the real counts and y_i be the corresponding anonymized values shown in Table 2. We model an attacker that assumes all values from 0 to 31 to be equally likely for all x_i, so $x_i \sim \text{DiscUni}(0, 31)$. This range is chosen to ensure a sufficient number of bits to represent the largest value in Table 2 (i.e., 23). To model the anonymization method, we impose the constraint that $y_i - 2 \leq x_i \leq y_i + 2$, i.e., each y_i value must have been computed from a value x_i that was at most two units away. If $y_i = 0$, we impose $x_i \leq y_i + 2$, as negative counts are not possible. For the total counts (last row in Table 2), the real count is the sum $x_i = x_i^{\text{m}} + x_i^{\text{f}}$ of real counts of males and females. The total counts are also anonymized.

Results. Figures 2 and 3 summarize the convergence results. SAT-METROPOLIS with SPUR converges to the correct values in both case studies: μ_{diff} is below 0.011 for DB CACM and below 0.008 for NZ Stats after 5000 samples. After 15000+ samples, it falls below 0.004 for both problems. The convergence for the linear problem (NZ Stats) is only minimally faster than for the polynomial problem (DB CACM). These are excellent results, showing that SAT-METROPOLIS with SPUR solves these problems effectively.

SAT-METROPOLIS with CMSGen converges, but with an observable bias. The value of $\mu_{\text{diff}}(\{x_i\})$ for marginals $x_0 \,..\, x_4$ and x_9 is below 0.014 for DB CACM after 5000 samples. However, for the binary variables $x_5 \,..\, x_8$ the error increases; notably $\mu_{\text{diff}}(\{x_5\}) \approx 0.19$. The value of μ_{diff} is below 0.076 for all variables in the NZ Stats case study after 5000 samples. The plots show that μ_{diff} does not notably reduce as the number of samples increases, which indicates that results are biased. The bias is rather small for integer variables, but

problematic for binary variables. The results are acceptable, but, an order of magnitude away from the results with the SPUR backend, suggesting that handling variables with binary domains might be a limitation of the CMSGen backend in this application.

SAT-METROPOLIS with MegaSampler converges as well, but, like with CMSGen, it exhibits bias for select variables. The value of $\mu_{\text{diff}}(\{x_i\})$ for variables $x_0 .. x_4$ and x_9 is below 0.022 in DB CACM after 5000+ samples. However, for binary variables $x_5 .. x_8$ the value increases up to 0.23. For NZ Stats and 5000+ samples, the value of μ_{diff} is between 0.06 and 0.15 for most variables. We do observe a larger absolute mean difference for integer variables in the NZ Stats problem. At the same time the MegaSampler backend shows the slowest speed of convergence, with values of μ_{diff} above 0.3 for some variables in all problems. Increasing the number of samples above ≈ 5000 does not reduce the error significantly, which suggests bias. These results also indicate that the reweighing method that we used for MegaSampler is not effective in removing bias.

Discussion. The results of the convergence experiments indicate that uniformity helps un-biased convergence, and in this regard SPUR performs very well. This might be because a uniform proposal distribution, is not only positive, but also symmetric. It would be interesting to try more robust versions of the Metropolis algorithms, to see whether they would be able to overcome the non-uniformity of CMSGen and MegaSampler.

The type of constraints did not significantly impact the speed of convergence or bias. It is also noteworthy that all backends converged to their lowest absolute mean difference value within 5000 samples. This indicates that the samples are indeed obtained in a (practically) independent fashion. Generating un-correlated samples is one of the key factors in the efficiency of MCMC inference algorithms. If proposals are correlated, convergence is slow.

We executed an additional experiment that combines hard and soft constraints. We evaluated the convergence of SAT-METROPOLIS for the DB CACM and NZ Stats problems using a Poisson distribution for count variables and bias Bernoulli for binary variables. We assigned to each variable x_i a distribution Poisson(λ) and each variable m_i a distribution Bernoulli(p). The Poisson distribution assigns the highest posterior probability to the value λ and probability decreases as values move away from λ. Poisson is a common distribution for count variables, as it has a positive support – i.e., it assigns probability 0 to negative values. The Bernoulli distribution assigns a probability of p to m_i equals true and $1-p$ to the opposite. The hard constraints are the same as in the case studies, but this distributions add soft constraints showing higher preference for values close to λ for count variables and bias p for binary variables. Our results indicated no significant changes in convergence behavior. SPUR converges to the right value, but with slower speed of convergence. CMSGen and Metropolis are bias, but μ_{diff} decreases with respect to the results without soft constraints. These changes are due to the regularizing effect of the soft constraints. This shows that SAT-METROPOLIS can handle a mix of soft and hard constraints well, even though the Metropolis part does not know the hard constraints, and the SAT/SMT sampler does not know about the soft constraints.

6.2 Scalability

We evaluate the runtime performance of SAT-METROPOLIS for increasing number of constraints and domain size for variables. The goal is to determine whether the performance of SAT/SMT samplers is sufficient to generate samples from realistic problems. We have selected two probabilistic inference problems that were used as benchmarks by Hazelton et al. [12]. Their work is one of most recent ones on MCMC algorithms for probabilistic inference with (linear) hard constraints, and, as such, these benchmarks demonstrate whether SAT-METROPOLIS using

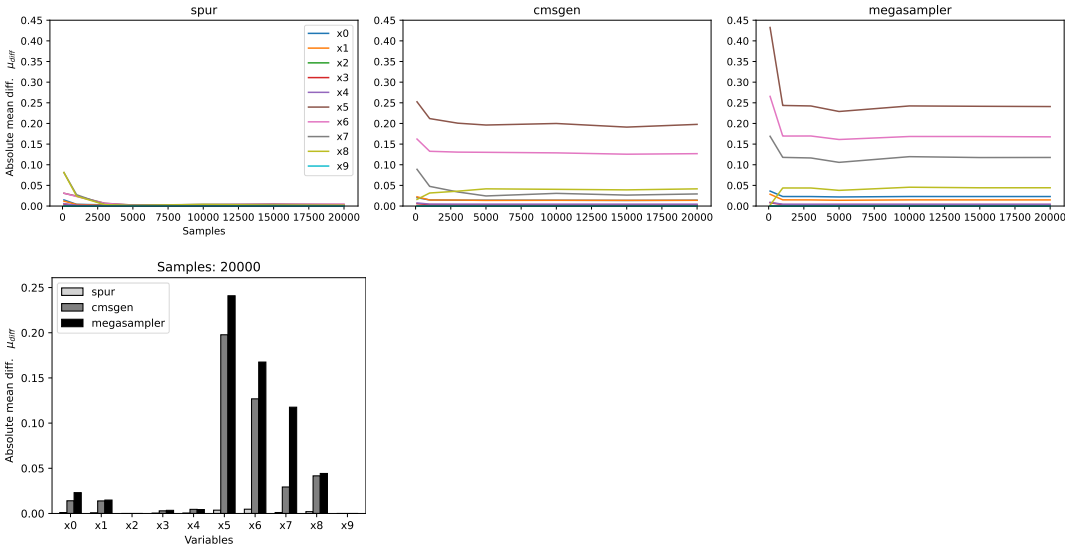


Figure 2 Convergence with different SAT/SMT samplers in the DB CACM case study. Variables $x_0 .. x_4$ represent age and $x_5 .. x_9$ are the binary variables $m_0 .. m_4$ representing gender.

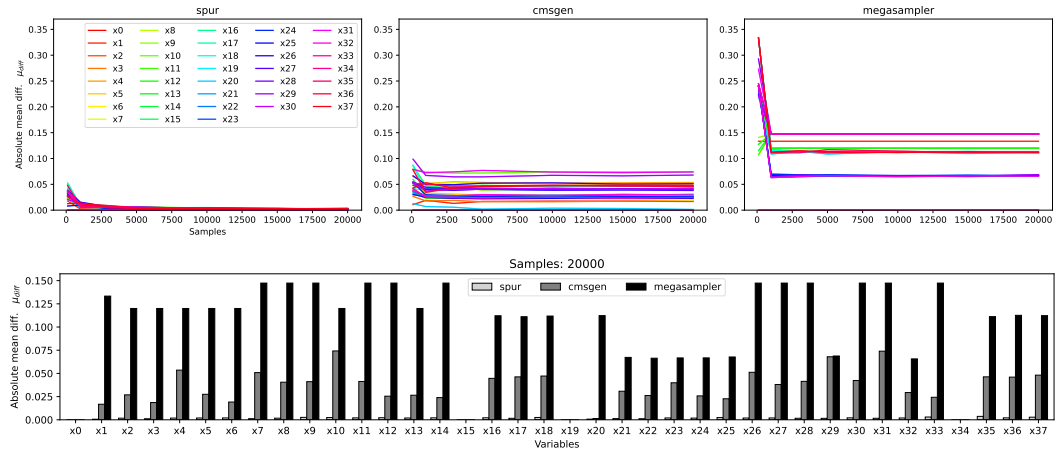
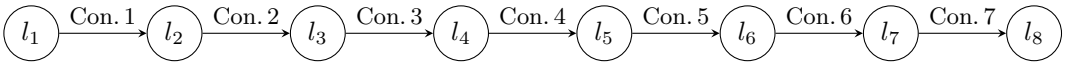


Figure 3 Convergence with different SAT/SMT samplers in the NZ stats case study. Variables $x_0 .. x_{18}$ represents the male counts from (0-4) to (90+), and variables $x_{19} .. x_{37}$ represents the female counts from (0-4) to (90+).

state-of-the-art SAT/SMT samplers is capable of tackling realistic problems. Furthermore, one can easily control the number of constraints, number of variables, and domain size for each variable for their problems, which allows us to find the limits of each sampling backend when they cannot sample from the problem as originally defined.

We perform experiments for an increasing number of samples $\{100, 500, 1000, 3000, 7000\}$, with the range matching the convergence speed in the experiments of Section 6.1. We set a timeout of ten minutes for each sampling run. This timeout is set to be twice the time the method of Hazelton and colleagues takes for the hardest cases. Note, that we expect SAT-METROPOLIS to be slower, as it is solves a more general class of problems than their method, and admits a general constrained-based formulation of these problems. We executed every experiment 3 times and report the mean.

Roads. We are estimating the traffic flow over a highway in UK. Nodes l_i represent locations in the highway. The directed edges are road segments connecting locations (connections).



To travel from l_i to l_j one must pass through all intermediate connections $i..j-1$. The traffic flows in the direction of the edges, so it is only possible to travel from l_i to l_j if $i < j$.

The task is to estimate the flow of traffic for each connection defined as a pair of nodes based on measurement of flow at each connection between adjacent nodes. Hazelton et al. use a probabilistic model for this [12]. For instance, trips between (l_1, l_2), (l_1, l_3), (l_1, l_4), ..., (l_7, l_8). For each possible trip (l_i, l_j) the probabilistic model contains a variable x_{ij} that models the number of trips from i to j. The hard constraints are given by the observed flow v_c on each segment between adjacent nodes; flow is measured in terms of vehicle counts. Given a connection c, let $T_c = \{x_{ij} \mid i \leq c < j\}$ be the set of trip variables that pass through c. Then, hard constraints are defined as $v_c = \sum_{x \in T_c} x$. The set of observed vehicle counts is $\{1087, 1008, 1068, 1204, 1158, 1151, 1143\}$, corresponding to $\{v_1, \ldots, v_7\}$. We assign a prior DiscUni$(0, m)$ for each variable x_{ij}, where m is a positive integer modeling the maximum number of trips. We set $m = 2^n - 1$ where n is the lowest integer such that hard constraints are satisfiable. For SAT backends, we set the bit-vector size to $s = \max(\lceil \log_2(v_c) \rceil) + 1$ to avoid overflow errors. Note that $s > n$ must hold, as the result of $\sum_{x \in T_c} x$ must not overflow (this constraint ensures that we do not get accidental solutions due to overflow).

The full model contains 28 variables (x_{ij}) and 7 connections. To adjust the complexity of the problem we will apply two modifications. First, we consider problems from two to seven connections. Second, we reduce the vehicle counts for each connection by a *reduction factor* r, i.e., for a reduction factor r the vehicle counts for a connection is updated as v_c/r and rounded to the closest integer. Note that this reduction also decreases domain size and bit-vector size of all variables x_{ij}. We consider reduction factors of 1, 2 and 4.

Books. We are estimating the number ratings from dataset of book rating counts with three dimensions: six different books (labeled A–F), four countries (Australia, Canada, UK and USA) and three ranges of age (0-25, 26-50 and +51). The data is shown in Table 3. The task is to estimate book ratings for all book/country/age combinations by only observing two-way interactions (marginals), i.e., number of book ratings: for all books given country and age, for all countries given age and book, and for all ages given country and book. More precisely, the set of two-way interaction counts is

$$
\begin{aligned}
B = \{&c_{\text{Aus},0-25}, c_{\text{Aus},26-50}, c_{\text{Aus},51+}, c_{\text{Can},0-25} \ldots, c_{\text{USA},51+}\} \cup && \textit{(Country,Age)} \\
&\{c_{0-25,\text{A}}, c_{26-50,\text{A}}, \ldots, c_{51+,\text{F}}\} \cup && \textit{(Age,Book)} \\
&\{c_{\text{Aus},\text{A}}, c_{\text{Can},\text{A}}, \ldots, c_{\text{USA},\text{F}}\} && \textit{(Country,Book)}
\end{aligned}
$$

where, for instance $c_{\text{Aus},0-25} = 3$ is computed by summing the counts of the first column in Table 3. The probabilistic model for this problem consists of variables of the form x_{bca} where b, c, a are indexes for book labels, countries and ages, respectively. Thus, hard constraints are defined as: $c_{ca} = \sum_{i \in \{A\,..\,F\}} x_{ica}$ (Country, Age), $c_{ab} = \sum_{i \in \{\text{Aus}\,..\,\text{USA}\}} x_{bia}$ (Age, Book), and $c_{cb} = \sum_{i \in \{0-25\,..\,51+\}} x_{bci}$ (Country, Book). Similar to the previous problem, each variable x_{bca} has a prior DiscUni$(0, m)$ where $m = 2^n - 1$, n is the lowest integer such that hard constraints are satisfiable, and, for SAT backends, bit-vectors have size $\max(\lceil \log_2(c_{ij}) \rceil) + 1$.

The model has 72 variables. We control its complexity by increasing the number of two-way interactions. We start with (Country, Age), add (Age, Book), and finally include all two-way interactions. We also reduce the book rating counts to c_{ij}/r by a reduction factor r, to adapt the domain size of the variables and thus the bit-vectors. We use factors 1, 2 and 4.

Table 3 The dataset for book rating counts per title (A–F), Country (Australia, Canada, U.K. and U.S.A) and ages (0-25, 26-50 and +51). Source: Hazelton et al. [12]

	Ages 25 and under				Ages 26 to 50				Ages 51 and over			
Book	Aus.	Can.	UK	USA	Aus.	Can.	UK	USA	Aus.	Can.	UK	USA
A	0	0	1	39	1	3	4	94	0	0	1	12
B	1	7	1	49	7	25	4	217	0	4	0	34
C	0	2	0	16	1	10	1	92	0	2	1	37
D	0	2	0	17	1	8	12	126	0	0	1	24
E	1	8	0	43	2	29	3	186	0	6	1	28
F	1	2	0	20	0	11	0	107	1	4	1	28

Results. Figures 4 and 5 summarize the average runtime of SAT-METROPOLIS for the different SAT/SMT samplers. SAT-METROPOLIS with SPUR performs successfully in eight out of 35 benchmarks. It is able to successfully complete, so generate up to 7000 samples without timing out, the Roads experiments for up to three connections – with the exception of three connections and no reduction ($r = 1$). Increasing the reduction for three connections reduces the running time notably. For Books, the SPUR variant can only solve the case with all constraints and the reduction by four. SPUR scales only to problems with low number of variables or variables with small domains, but it is noteworthy that when it does not timeout, the runtime is essentially independent of the number of samples.

The CMSGen variant is by far the most successful one, handling 18 out of 35 benchmarks. SAT-METROPOLIS with CMSGen scales up to five connections for Roads. For Books it can handle all cases with more than 27 constraints. There are two exceptions: i) the roads problem with 5 connections, reductions by 1 and 2 with 1000+ samples; and ii) the books problem with 27 constraints and no reduction. Moreover, when it does not timeout, it always outperforms the SPUR and MegaSampler backends. As expected, for all successful experiments, increasing the reduction always results in a decrease in running time. The running time seems to increase linearly with the number of samples to generate; although the increase is steeper for lower reduction factors. The CMSGen backend exhibits a similar scalability tendency to that of SPUR, but CMSGen can handle larger number of variables, constraints and domain sizes. Notably, SAT-METROPOLIS with CMSGen can solve Books as originally posed by Hazelton et al. [12].

SAT-METROPOLIS with MegaSampler performs successfully in seven out of 35 benchmarks. It solves the roads problem for five to seven connections and the books problem for all amounts of constraints – with the exception of generating 7000 samples for 27 and 42 constraints. Unfortunately, it only works for reductions by four, which for these benchmarks means a domain size of 64. MegaSampler-based SAT-METROPOLIS is adequate only for problems with small domain size and large number of constraints.

Discussion. We did consider using a vanilla Metropolis algorithm, as a baseline for the experiments above, using the implementation of the widely used probabilistic programming library PyMC [1]. However, this algorithm is unable to generate more than one sample in any of the benchmark above. It fails them all. Furthermore, to work with it, we had to use Z3 to provide a satisfying initial state, as otherwise the PyMC implementation of Metropolis cannot start the execution. In this respect, SAT-METROPOLIS is a major improvement compared to using MCMC with hard constraints, as anticipated in Section 4.

It is remarkable that SAT-METROPOLIS with CMSGen is able to solve Books as originally defined by Hazelton et al. [12], bias issues notwithstanding. Their method, known as Dynamic Lattice Basis (DLB), takes about ten seconds to produce 7000 effective samples. The number

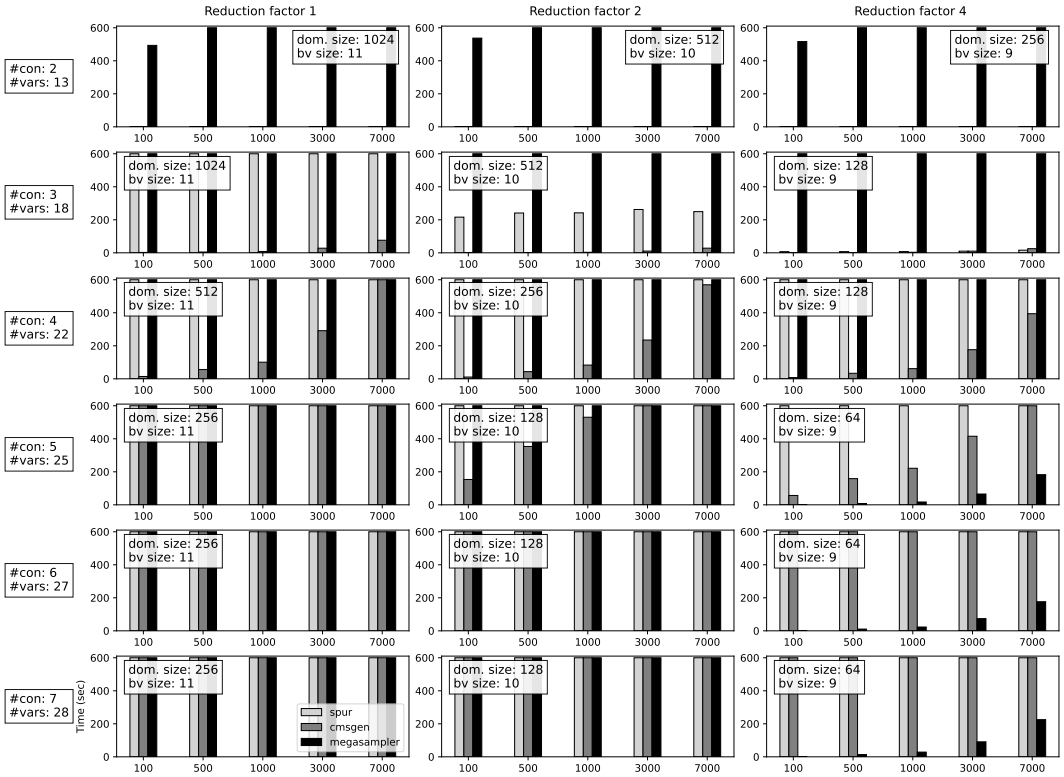


Figure 4 Running times for Roads with increasing number of connections (top to bottom), and decreasing variable domains (left to right). The original problem [12] is in the bottom-left corner.

of effective samples measures the number of samples with low autocorrelation, and it is a common measure of to compare MCMC methods of different nature [17]. This means DLB is 28x faster than SAT-METROPOLIS. However, DLB was specifically designed to solve problems with linear hard constraints in counting problems, like the case studies above, whereas SAT-METROPOLIS uses generic SAT sampling technology and can handle higher-order polynomial constraints. Also DLB requires presenting the problem as a linear program of a specific kind, while SAT-METROPOLIS does not have this limitation.

Our measurements split the running time of the SAT/SMT engine, and the rest of the algorithm, i.e., computing the acceptance ratio and deciding whether to move to the proposed state. We observed an average runtime for the latter of 0.01 and 0.05 seconds for Roads and Books, respectively. This indicates that the runtime is heavily dominated by the SAT/SMT sampler, and the effect soft constraints does significantly impact the overall runtime.

7 Related work

Recent work on MCMC based probabilistic inference with hard constraints targets discrete inverse linear problems [12, 18, 7, 3]. These are problems of the form $\mathbf{y} = \mathbf{A}\mathbf{x}$ where $\mathbf{y}$ is an observed counts vector and $\mathbf{A}$ is a contingency matrix that defines the constraints on the variables in the vector $\mathbf{x}$ w.r.t. the observed counts. A major advance in tackling this type of problems was the use of Markov (sub-)basis (see [3] for an overview). Let $\mathcal{F}_{\mathbf{y}} = \{\mathbf{x}|\mathbf{y} = \mathbf{A}\mathbf{x}\}$ be the set of count vectors satisfying the hard constraints – this set is known as the $\mathbf{y}$-*fiber*.

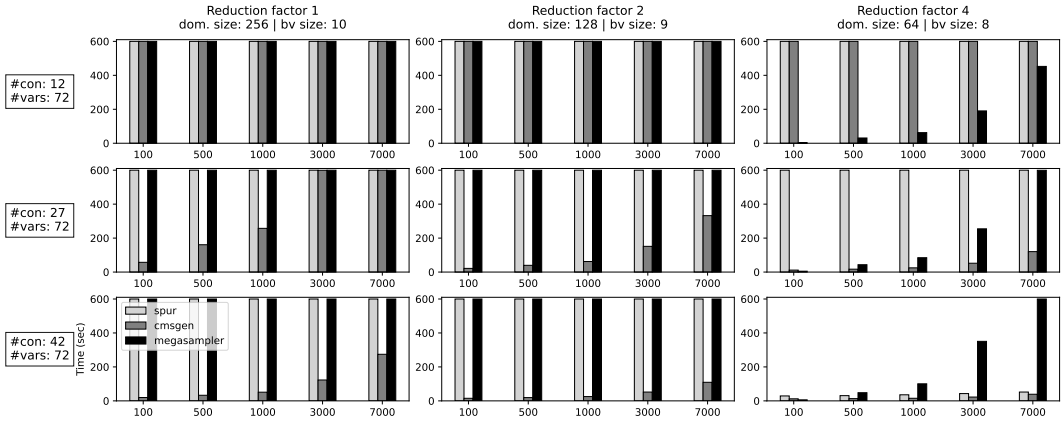


Figure 5 Running times for Books with increasing number of constraints (top to bottom) and decreasing variable domains (left to right). The original problem [12] is in the bottom-left corner.

Markov basis methods produce a Markov chain such that all points within the $\mathbf{y}$-*fiber* are reachable from any state – recall that this is a requirement for convergence of MCMC. However, computing the Markov basis for a given problem can be computationally expensive. This has motivated works suppressing the need for computing the full Markov basis. Dobra [7] designed a method to generate sampling directions dynamically while ensuring that the union of all directions is within the $\mathbf{y}$-*fiber*. More recently, the work by Hazelton et al. [12] introduces a method based on dynamic lattice basis which allows for sampling from the $\mathbf{y}$-*fiber* without computing the full Markov basis, which notably improves sampling performance. Our method does not achieve the same level of performance as these specialized methods. However, SAT-METROPOLIS can tackle a larger class of problems. Our method can handle problems with polynomial hard constraints whereas discrete inverse linear problems only feature linear constraints. Nevertheless, despite being able to tackle a more general class of problems, our evaluation showed that SAT-METROPOLIS with CMSGen can solve the book ratings problem of Hazelton.

We are not the first to propose the use of SAT samplers for probabilistic inference. Poon and Domingos [22] define a slice sampler to tackle probabilistic inference problems specified as Markov Logic Networks [23]. Markov Logic Networks are useful to express statistical problems with relational constraints, but they require uncommon encodings for other problems such as linear or polynomial constraints. Instead of designing a sampler for a specific type of probabilistic models, namely Markov Logic Networks, we adapt Metropolis, which a standard and widely used algorithm in statistics. SAT-METROPOLIS takes probabilistic inference problems expressed as an unnormalized density function (typically the numerator in Bayes rule: prior $\times$ likelihood), which is common place in the statistics literature. The slice sampler uses SampleSAT [24] to sample solutions for the relational constraints (expressed as propositional formulae). SampleSAT does not have uniformity guarantees and works on SAT problems. As a consequence, it cannot be used to analyze the effect of uniformity or bit-blasting, as we did in our work; with the backends SPUR (perfect uniformity) or MegaSampler (which works directly on SMT problems and can be compared to SPUR/CMSGen that require bit-blasting). Furthermore, the most scalable backend of SAT-METROPOLIS (cf. Section 6.2), CMSGen, has been recently shown to be one of the most scalable SAT sampling tools [10]. Thus, our work provides a more detailed and updated perspective on the use of SAT/SMT sampling methods for probabilistic inference than Poon and Domingos [22].

8 Conclusion

We have demonstrated that SAT/SMT sampling technology can effectively be used to tackle probabilistic inference problems with multivariate polynomial hard constraints. SAT/SMT sampling presents itself as a promising technology for tackling the problem of probabilistic inference via MCMC, where hard and soft constraints are mixed. To this end, we have adapted the Metropolis algorithm to use SAT/SMT samplers as proposal distributions. This prevents proposing zero-probability states, which lead the standard Metropolis algorithm to fail to explore the target distribution well. We have implemented SAT-METROPOLIS with three SAT/SMT samplers; SPUR, CMSGen and MegaSampler, and have evaluated convergence and scalability for these three variants. The results indicate that uniformity impacts convergence of SAT-METROPOLIS positively; as SPUR is the backend that exhibits the best convergence. At the same time, CMSGen is the most scalable sampler in this application. Even though SAT-METROPOLIS is generic with respect to the constraint class supported, it can handle problems used to benchmark specialized methods.

References

1 Oriol Abril-Pla, Virgile Andreani, Colin Carroll, Larry Dong, Christopher J. Fonnesbeck, Maxim Kochurov, Ravin Kumar, Junpeng Lao, Christian C. Luhmann, Osvaldo A. Martin, Michael Osthege, Ricardo Vieira, Thomas Wiecki, and Robert Zinkov. PyMC: a modern, and comprehensive probabilistic programming framework in Python. *PeerJ Comput. Sci.*, 9, 2023. doi:10.7717/peerj-cs.1516.

2 Dimitris Achlioptas, Zayd Hammoudeh, and Panos Theodoropoulos. Fast sampling of perfectly uniform satisfying assignments. In *Proceedings of Theory and Applications of Satisfiability Testing (SAT'18)*, volume 10929 of *LNCS*, pages 135–147. Springer, 2018. doi:10.1007/978-3-319-94144-8_9.

3 Satoshi Aoki, Hisayuki Hara, and Akimichi Takemura. *Markov bases in algebraic statistics*, volume 199. Springer Science & Business Media, 2012.

4 Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. *Handbook of markov chain monte carlo*. CRC press, 2011.

5 Maja Aaslyng Dall, Raúl Pardo, Thomas Lumley, and Andrzej Wąsowski. SAT-Metropolis. Software, swhId: swh:1:dir:bfc2169353301af051d1af4488050fcfba06a849 (visited on 2025-07-28). URL: https://github.com/itu-square/sat-metropolis, doi:10.4230/artifacts.24207.

6 Leonardo Mendonça de Moura and Nikolaj S. Bjørner. Z3: an efficient SMT solver. In *Proceedings of International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'08)*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008. doi:10.1007/978-3-540-78800-3_24.

7 Adrian Dobra. Dynamic markov bases. *Journal of Computational and Graphical Statistics*, 21(2):496–517, 2012.

8 Simson Garfinkel, John Abowd, and Christian Martindale. Understanding database reconstruction attacks on public data: These attacks on statistical databases are no longer a theoretical danger. *Commun. ACM*, 62:46–53, 2019. doi:10.1145/3287287.

9 Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, 1984. doi:10.1109/TPAMI.1984.4767596.

10 Priyanka Golia, Mate Soos, Sourav Chakraborty, and Kuldeep S. Meel. Designing samplers is easy: The boon of testers. In *Proceedings of Formal Methods in Computer Aided Design, (FMCAD'21)*, pages 222–230. IEEE, 2021. doi:10.34727/2021/isbn.978-3-85448-046-4_31.

11 W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

12 Martin Hazelton, M Mcveagh, and Bruce van Brunt. Geometrically Aware Dynamic Markov Bases for Statistical Linear Inverse Problems. *Biometrika*, 108:609–626, October 2020.

13 Martin Hazelton, Michael McVeagh, Christopher Tuffley, and Bruce van Brunt. Some rapidly mixing hit-and-run samplers for latent counts in linear inverse problems. *Bernoulli*, 30(4):2676–2699, 2024.

14 Matthew D Hoffman, Andrew Gelman, et al. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623, 2014. doi:10.5555/2627435.2638586.

15 Daniel Kroening and Ofer Strichman. *Decision Procedures: An Algorithmic Point of View*. Springer, 2008. doi:10.1007/978-3-540-74105-3.

16 David J.C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

17 R. McElreath. *Statistical Rethinking: A Bayesian Course with Examples in R and Stan*. A Chapman & Hall book. CRC Press, 2020.

18 Michael McVeagh. *Efficient Markov bases for Z-polytope sampling: a thesis presented in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Mathematics at Massey University, Manawatū, New Zealand*. PhD thesis, Massey University, 2021.

19 Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

20 Raúl Pardo, Willard Rafnsson, Christian W. Probst, and Andrzej Wąsowski. Privug: Using probabilistic programming for quantifying leakage in privacy risk analysis. In *Proceedings of European Symposium on Research in Computer Security (ESORICS'21)*, volume 12973 of *LNCS*, pages 417–438. Springer, 2021. doi:10.1007/978-3-030-88428-4_21.

21 Matan I. Peled, Bat-Chen Rothenberg, and Shachar Itzhaky. SMT sampling via model-guided approximation. In *Proceedings of International Symposium on Formal Methods (FM'23)*, volume 14000 of *LNCS*, pages 74–91. Springer, 2023. doi:10.1007/978-3-031-27481-7_6.

22 Hoifung Poon and Pedro Domingos. Sound and efficient inference with probabilistic and deterministic dependencies. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'06)*, volume 1, pages 458–463. AAAI Press, 2006. URL: http://www.aaai.org/Library/AAAI/2006/aaai06-073.php.

23 Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62:107–136, 2006. doi:10.1007/s10994-006-5833-1.

24 Wei Wei, Jordan Erenrich, and Bart Selman. Towards efficient sampling: Exploiting random walk strategies. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI'04)*, pages 670–676. AAAI Press / The MIT Press, 2004. URL: http://www.aaai.org/Library/AAAI/2004/aaai04-106.php.

A Experiments Details Summary

Table 4 show the details of each of the experiments that we considered in the paper. In total there are 37 benchmarks. We include the number of variables and clauses (both in the original SMT problem and after bit-blasting), the bit-vector size and domain size for each experiment. Additionally, we include the clause-to-variable ratio of the bit-blasted version of each experiment. The last three columns indicate whether the corresponding backend was able to generate at least 7000 within 10 minutes (✔) or not (✘).

Table 4 Qualitative summary of experiments results. The symbol (✔) denotes that running SAT-METROPOLIS with the backend indicated in the column was able to generate 7000 samples or more within the 10min timeout; which we consider a success. If not, we mark the experiment with the symbol (✘).

Problem	#vars	#constraints	#vars (bit-blasted)	#clauses (bit-blasted)	reduction	bv size	dom. size	clause-to-var ratio	spur	cmsgen	megasampler
DB CACM	10	20	1605	7507	NA	8	126	4.67	✔	✔	✔
NZ Stats	38	141	661	2025	NA	5	24	3.06	✔	✔	✔
Roads	13	2	203	713	1	11	1023	3.51	✔	✔	✘
Roads	13	2	183	641	2	10	511	3.50	✔	✔	✘
Roads	13	2	148	499	3	9	255	3.37	✔	✔	✘
Roads	13	2	162	565	4	9	255	3.48	✔	✔	✘
Roads	18	3	646	2722	1	11	1023	4.21	✘	✔	✘
Roads	18	3	586	2464	2	10	511	4.20	✔	✔	✘
Roads	18	3	512	2154	3	9	255	4.20	✔	✔	✘
Roads	18	3	516	2168	4	9	127	4.20	✔	✔	✘
Roads	22	4	1409	6491	1	11	511	4.60	✘	✘	✘
Roads	22	4	1267	5827	2	10	255	4.59	✘	✔	✘
Roads	22	4	1112	5118	3	9	127	4.60	✘	✔	✘
Roads	22	4	1128	5180	4	9	127	4.59	✘	✔	✘
Roads	25	5	2537	12279	1	11	255	4.83	✘	✘	✘
Roads	25	5	2288	11055	2	10	127	4.83	✘	✘	✘
Roads	25	5	2029	9776	3	9	127	4.81	✘	✘	✘
Roads	25	5	1998	9655	4	9	63	4.83	✘	✘	✔
Roads	27	6	4146	20654	1	11	255	4.98	✘	✘	✘
Roads	27	6	3742	18605	2	10	127	4.97	✘	✘	✘
Roads	27	6	3239	16122	3	9	127	4.97	✘	✘	✘
Roads	27	6	3298	16383	4	9	63	4.96	✘	✘	✔
Roads	28	7	6266	31854	1	11	255	5.08	✘	✘	✘
Roads	28	7	5683	28798	2	10	127	5.06	✘	✘	✘
Roads	28	7	4966	25201	3	9	63	5.07	✘	✘	✔
Roads	28	7	4962	25174	4	9	63	5.07	✘	✘	✔
Books	72	12	5895	24629	1	10	255	4.17	✘	✘	✘
Books	72	12	5229	21830	2	9	127	4.17	✘	✘	✘
Books	72	12	5237	21955	3	9	63	4.19	✘	✔	✔
Books	72	12	4565	19045	4	8	63	4.17	✘	✘	✔
Books	72	27	8622	38995	1	10	255	4.52	✘	✘	✘
Books	72	27	7632	34482	2	9	127	4.51	✘	✔	✘
Books	72	27	7657	34671	3	9	63	4.52	✘	✔	✘
Books	72	27	6627	29941	4	8	63	4.51	✘	✔	✔
Books	72	42	10252	47299	1	10	255	4.61	✘	✔	✘
Books	72	42	8987	41502	2	9	127	4.61	✘	✔	✘
Books	72	42	7811	36101	4	8	63	4.62	✔	✔	✘
								Success/timeout experiment counts:	10/37	20/37	9/37