

# Random Local Access for Sampling $k$ -SAT Solutions

Dingding Dong  

Harvard University, Cambridge, MA, USA

Nitya Mani  

Massachusetts Institute of Technology, Cambridge, MA, USA

---

## Abstract

We present a *sublinear time* algorithm that gives *random local access* to the uniform distribution over satisfying assignments to an arbitrary  $k$ -SAT formula  $\Phi$ , at exponential clause density. Our algorithm provides *memory-less* query access to variable assignments, such that the output variable assignments consistently emulate a single global satisfying assignment whose law is close to the uniform distribution over satisfying assignments to  $\Phi$ . Random local access and related models have been studied for a wide variety of natural Gibbs distributions and random graphical processes. Here, we establish feasibility of random local access models for one of the most canonical such sample spaces, the set of satisfying assignments to a  $k$ -SAT formula.

Our algorithm proceeds by leveraging the local uniformity of the uniform distribution over satisfying assignments to  $\Phi$ . We randomly partition the variables into two subsets, so that each clause has sufficiently many variables from each set to preserve local uniformity. We then sample some variables by simulating a systematic scan Glauber dynamics backward in time, greedily constructing the necessary intermediate steps. We sample the other variables by first conducting a search for a polylogarithmic-sized local component, which we iteratively grow to identify a small subformula from which we can efficiently sample using the appropriate marginal distribution. This two-pronged approach enables us to sample individual variable assignments without constructing a full solution.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Streaming, sublinear and near linear time algorithms

**Keywords and phrases** sublinear time algorithms, random generation,  $k$ -SAT, local computation

**Digital Object Identifier** 10.4230/LIPIcs.SAT.2025.13

**Related Version** *Full Version*: <https://arxiv.org/abs/2409.03951>

**Funding** NM was supported by the Hertz Graduate Fellowship and the NSF Graduate Research Fellowship Program #2141064. This material includes work supported by the NSF Grant DMS-1928930, while the authors were in residence at SLMath during the Spring 2025 semester.

**Acknowledgements** We would like to thank Kuikui Liu and Ronitt Rubinfeld for helpful discussions, suggestions, and feedback on a draft. We would like to thank the anonymous referees for identifying an issue with a previous version of this article.

## 1 Introduction

Efficiently sampling from an exponentially sized solution space is a fundamental problem in computation. The quintessential such problem is sampling a uniformly random satisfying assignment to a given  $k$ -SAT formula, the setting of this work. Throughout, we let  $\Phi$  denote an  $n$ -variable  $(k, d)$ -formula, which is a  $k$ -SAT formula in *conjunctive normal form* (which we also call a  $k$ -CNF) such that each variable appears in at most  $d$  clauses. We let  $\Omega = \Omega_\Phi$  denote the set of satisfying assignments to  $\Phi$ , and let  $\mu = \mu_\Phi$  denote the uniform distribution over  $\Omega$ . We consider the regime where  $k$  is a (large) constant and take  $n \rightarrow \infty$ .



© Dingding Dong and Nitya Mani;

licensed under Creative Commons License CC-BY 4.0

28th International Conference on Theory and Applications of Satisfiability Testing (SAT 2025).

Editors: Jeremias Berg and Jakob Nordström; Article No. 13; pp. 13:1–13:24

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

For  $d \leq (2ek)^{-1}2^k$ , it has long been known (e.g., by the Lovász local lemma introduced in [8]) that any such  $k$ -SAT formula  $\Phi$  has at least one satisfying assignment. Moreover, seminal work of Moser and Tardos [27] gave a simple algorithm that computes such a satisfying assignment in expected linear time. However, a similarly basic question in the sampling regime is far more poorly understood: when is it possible to efficiently sample an (approximately uniformly) random satisfying assignment to  $\Phi$ ?

The first breakthrough in this direction came from Moitra [25] who gave a deterministic, fixed-parameter tractable algorithm for approximately sampling from  $\mu$  provided that  $d \lesssim 2^{ck}$  for  $c \approx 1/60$ . This was followed up by work of Feng, Guo, Yin and Zhang [10], who used a Markov chain approach to give a polynomial time algorithm for sampling from  $\mu$  provided that  $d \lesssim 2^{k/20}$ . Recent works [18, 31] have given state of the art results for sampling from  $k$ -CNFs, when  $d \lesssim 2^{k/4.82}$  (in fact for large domain sizes, [31] gives essentially tight bounds on sampling from atomic constraint satisfaction problems). There has also been a tremendous amount of work sampling from random  $k$ -CNFs, recently essentially resolved by work of [6], building on previous progress of [13, 7, 12, 19].

A natural question about such algorithms to sample from  $k$ -CNFs is whether one can adapt them to more efficiently answer *local* questions about a random satisfying assignment in *sublinear time*. For variable  $v$ , let  $\mu_v$  denote the marginal distribution on  $v$  induced by  $\mu$ . When  $n$  is large, one might wish to sample a small handful of variables  $S$  from their marginal distributions  $\mu_S$  in  $o(n)$  time, without computing an entire  $\Omega(n)$ -sized satisfying assignment  $\sigma \sim \mu$ . Further, many traditional algorithms for counting the number of satisfying assignments to a given  $k$ -SAT formula proceed by computing marginal probabilities of variable assignments, a task that can be completed given local access to a random assignment. Therefore, sublinear time algorithms for answering local questions can also yield speedups in more general counting algorithms.

Ideally, a *random local access* model should provide query access to variable assignments such that the output enjoys the following properties:

- (a) the model is *consistent*: queries for different variable assignments consistently emulate those of a single random satisfying assignment  $\sigma \sim \mu$ ;
- (b) the model is *sublinear*: sampling variable assignments takes  $o(n)$  time in expectation;
- (c) the model is *memory-less*: given the same initialization of randomness, answers to multiple, possibly independent queries for variable assignments are consistent with each other.

We give a more precise definition of random local access models in Section 2. Random local access models were formally introduced in work of Biswas, Rubinfeld, and Yodpinyanee [5], motivated by a long line of work over the past decades studying sublinear time algorithms for problems in theoretical computer science. The authors of [5] studied a different natural Gibbs distribution, the uniform distribution over proper  $q$ -colorings of a given  $n$ -vertex graph with maximum degree  $\Delta$ . By adapting previously studied classical and parallel sampling algorithms for  $q$ -colorings, they were able to construct a local access algorithm to a random proper  $q$ -coloring when  $q \geq 9\Delta$ . The related problem of sampling partial information about huge random objects was pioneered much earlier in [15, 16]; further work in [28] considers the implementation of different random graph models. Random local access and related (e.g., parallel computation) models have also been recently studied for several other random graphical processes and Gibbs distributions (cf. [5, 4, 26, 23]).

The above model for local access to random samples extends a rich line of work studying *local computation algorithms* (LCAs), originally introduced in works of Rubinfeld, Tamir, Vardi, and Xie [29] and Alon, Rubinfeld, Vardi, and Xie [2]. Local computation algorithms

are widely used in parallel and distributed computing, with notable practical success in areas such as graph sparsification [1], load balancing [24], and sublinear-time coloring [2]. A primary initial application of LCAs in [29] was locally constructing a satisfying assignment to a given  $(k, d)$ -formula  $\Phi$  when  $d \lesssim 2^{k/10}$ . Similar to the desired properties of random local access, LCA implements query access to a large solution space in sublinear time using local probes. However, rather than sampling from a given distribution, LCA only provides local access to *some* consistent valid solution in the desired solution space that may otherwise be completely arbitrary in nature.

In this work, we present a polylogarithmic time algorithm that gives random local access to the uniform distribution over satisfying assignments of an arbitrary  $k$ -SAT formula  $\Phi$  at exponential clause density (i.e., the number of occurrences of each variable is bounded by an exponential function of  $k$ ). The following is an informal version of our main result; a more formal version can be found at Theorem 24.

► **Theorem 1** (Main theorem: informal). *There exists a random local access algorithm  $\mathcal{A}$  that satisfies the following conditions. For any  $(k, d)$ -formula  $\Phi = (V, \mathcal{C})$  with maximum degree  $d \lesssim 2^{k/400}$  and any variable  $v \in V$ , we can use  $\mathcal{A}$  to sample a single assignment for  $v$ ,  $\mathcal{A}(v) \in \{0, 1\}$  in expected polylogarithmic time (in  $|V|$ ), such that the distribution of  $\mathcal{A}(v)$  is the same as the marginal distribution of  $\sigma(v)$ , where  $\sigma$  is a uniformly random satisfying assignment to  $\Phi$ .*

Similar to [5], the proof of Theorem 1 adapts some of the algorithmic tools used to study parallel and distributed sampling. The proof also builds upon the work of Moitra [25] and Feng–Guo–Yin–Zhang [10] on sampling from bounded degree  $k$ -CNFs in polynomial time. The authors of [25, 10] both critically took advantage of a global variable *marking* (see Definition 10) to better condition the marginal distributions of variables. Such an approach allows for a subset of the variable assignments to be sampled with ease; the resulting *shattering* of the solution space conditioned on such a partial assignment then allows one to efficiently complete the random satisfying assignment. These initial approaches have been extended and strengthened to nearly linear time algorithms that succeed for larger maximum variable degree in a flurry of recent work (c.f. [25, 10, 20, 17, 13, 7, 12, 31, 6]).

Recently, Feng, Guo, Wang, Wang, and Yin [9] used a recursive sampling scheme to simulate the systematic scan projected Glauber dynamics via a strategy termed *coupling towards the past*, which they used to derandomise several Markov chain Monte Carlo (MCMC) algorithms for sampling from CSPs. Additionally, recent work of He, Wang and Yin [18] used a *recursive sampling* scheme to sample  $k$ -SAT solutions. Their work immediately gives a sublinear (in fact, expected constant time) algorithm for sampling the assignment of a single variable one time; however, this work does not immediately extend out of the box to a random local access model that enjoys the desired *consistency* and *memory-less* properties if multiple variables are sampled. Recursive sampling schemes have also been recently used to make progress on analyzing and designing fast sampling algorithms for a variety of Gibbs distributions (cf. [18, 7, 3]). As noted earlier, such schemes have been particularly useful for *derandomising* and constructing *parallel* and *distributed* versions of many popular MCMC algorithms for sampling solutions to CSPs [9, 11, 17, 18].

An immediate roadblock to adapting such global or parallel sampling strategies to random local access to  $k$ -SAT solutions is that the vast majority of the aforementioned existing algorithms critically either use some *global* information – like a variable marking, particular ordering of variables, or other global state compression – as an input to make algorithmic decisions or postpone sampling of certain problematic variable assignments until a *linear* number of other choices are made. Both these issues necessitate a substantive

departure from these approaches for any hope of local access adaptation. In this work, we overcome these obstacles by adapting the *coupling towards the past* strategy used in [9] to derandomise MCMC in conjunction with a local implementation of the variable marking strategy introduced in [25].

We use these algorithms to carefully select and sample a small number of auxiliary variable assignments on an as-needed basis, showing that bad cases can be reduced to small calculations after a sublinear number of steps. Our proof of Theorem 1 requires a novel adaptation of sampling techniques to avoid requiring global information or complete passes of the variable set; we show that we can perform analogous operations locally or leverage local uniformity properties to circumvent them altogether in the course of locally sampling a variable assignment. Importantly, we demonstrate how to execute local sampling in an *oblivious, consistent* fashion, so that the algorithm need not retain any memory and that variables need not be sampled in any particular order.

## 2 Preliminaries

### Notation

Throughout, let  $\Phi = (V, \mathcal{C})$  be a  $k$ -CNF on variable set  $V = \{v_1, \dots, v_n\}$  with associated collection of clauses  $\mathcal{C}$ . In this work we do not let the same variable appear multiple times in any clause of  $\Phi$ , although our algorithm could be adapted to the more general scenario. We further assume that  $\Phi$  is a  $(k, d)$ -formula; that is, each variable  $v_i$  appears in at most  $d$  clauses. For every clause  $C \in \mathcal{C}$ , let  $\text{vbl}(C)$  denote the collection of variables in the clause  $C$ . We further define  $\Delta := \max_{C \in \mathcal{C}} |\{C' \in \mathcal{C} : \text{vbl}(C) \cap \text{vbl}(C') \neq \emptyset\}|$ , so in particular  $\Delta \leq kd$ .

In the regime we work in, we assume  $k$  is a large fixed constant and view  $n \rightarrow \infty$ . We use  $f \lesssim g$  to denote that there is some constant  $C$  (not depending on  $k$ ) such that  $f \leq Cg$ . We also use the standard asymptotic notation ( $O, o, \Omega, \omega, \Theta$ ), where when not specified, we assume these are in the  $n \rightarrow \infty$  limit. We use  $\text{Law}(X)$  to denote the underlying distribution of a random variable  $X$ .

We let  $\Omega = \Omega_\Phi \subseteq \{0, 1\}^n$  denote the set of satisfying assignments to  $\Phi$  and let  $\mu = \mu_\Phi$  denote the uniform distribution over  $\Omega$ . We suppress the  $\Phi$  subscripts when the formula is clear from context. We introduce a few more concepts associated to a  $k$ -SAT instance that will be used later.

► **Definition 2.** Given probability distributions  $\nu_1, \nu_2$  over  $\Omega$ , the total variation distance is

$$d_{\text{TV}}(\nu_1, \nu_2) := \frac{1}{2} \sum_{\omega \in \Omega} |\nu_1(\omega) - \nu_2(\omega)|.$$

If we have a random variable  $X$  with  $\text{Law}(X) = \nu$ , we may write  $d_{\text{TV}}(\mu, X)$  instead of  $d_{\text{TV}}(\mu, \nu)$  in a slight abuse of notation.

► **Definition 3** (Dependency hypergraph). Given a  $(k, d)$ -formula  $\Phi$ , let  $H_\Phi = (V, \mathcal{E})$  be the dependency hypergraph (with multiple edges allowed), where  $V$  is the set of variables and  $\mathcal{E} = \{\text{vbl}(C) : C \in \mathcal{C}\}$  is the collection of variable sets of clauses of  $\Phi$ .

► **Definition 4** (Partial assignment). Given  $(k, d)$ -formula  $\Phi = (V, \mathcal{C})$ , let

$$\mathcal{Q}^* := \bigcup_{\Lambda \subseteq V} \{0, 1, \perp\}^\Lambda$$

denote the space of partial assignments with the following symbology. Given a partial assignment  $\sigma \in \{0, 1, \perp\}^\Lambda$  on some  $\Lambda \subseteq V$ , each variable  $v \in \Lambda$  is classified as follows:

- $\sigma(v) \in \{0, 1\}$  means that  $v$  is accessed by the algorithm and assigned with  $\sigma(v) \in \{0, 1\}$ ;
  - $\sigma(v) = \perp$  means that  $v$  is accessed by the algorithm but unassigned yet with any value.
- We sometimes use  $\emptyset$  to denote the empty assignment (i.e.,  $\Lambda$  is the empty set). We say that  $\sigma$  is feasible if it can be extended to a satisfying assignment to  $\Phi$ .

► **Definition 5** (Reduced formula on partial assignment). Let  $\Phi$  be a  $(k, d)$ -formula. Given a partial assignment  $\sigma$  on  $\Lambda \subseteq V$ , let  $\Phi^\sigma$  be the result of simplifying  $\Phi$  under  $\sigma$ . That is, define  $\Phi^\sigma := (V^\sigma, \mathcal{C}^\sigma)$  where

- $V^\sigma = V \setminus \{v \in \Lambda : \sigma(v) \in \{0, 1\}\}$ ,
- $\mathcal{C}^\sigma$  is obtained from  $\mathcal{C}$  by removing all clauses that have been satisfied under  $\sigma$ , and removing any appearance of variables that are assigned 0 or 1 by  $\sigma$ .

Let  $H_\Phi^\sigma$  be the associated (not necessarily  $k$ -uniform) hypergraph to  $\Phi^\sigma$ . For variable  $v \in V \setminus \Lambda$ , let  $\Phi_v^\sigma$  denote the maximal connected component of  $\Phi^\sigma$  to which  $v$  belongs.

► **Definition 6** (Marginal distribution). For an arbitrary set of variables  $S \subseteq V$ , let  $\mu_S$  be the marginal distribution on  $S$  induced by  $\mu$ , so that

$$\mu_S(\tau) = \sum_{\tau' \in \{0,1\}^n : \tau'|_S = \tau} \mu(\tau') \quad \forall \tau \in \{0,1\}^S.$$

When  $S = \{v\}$  is a single vertex, we let  $\mu_v = \mu_{\{v\}}$ . Furthermore, given some partial assignment  $\sigma \in \{0, 1, \perp\}^\Lambda$  for  $\Lambda \subseteq V$ , if  $S \cap \Lambda = \emptyset$ , we let  $\mu_S^\sigma(\cdot) := \mu_S(\cdot \mid \sigma)$  be the marginal distribution on  $S$  conditioned on the partial assignment  $\sigma$ . When  $S = V \setminus \Lambda$ , we simply use  $\mu^\sigma$  to denote  $\mu_{V \setminus \Lambda}^\sigma$ .

## 2.1 The random local access model and local computation algorithms

One of the most widely studied models of local computation are local computation algorithms (LCAs) introduced in [2, 29]. Given a computation problem  $F$ , an LCA (in an *online fashion*) provides the  $i$ -th bit to *some* consistent solution  $F$  in sublinear time. As originally defined, local computation algorithms need not be *query-oblivious*; in other words, the output solution can depend on the order of queried bits. However, several follow-up works have given query-oblivious analogues of LCAs for a variety of natural problems. Such models are the non-random-sampling version of the random local access models we study here.

In this work, we construct a query-oblivious LCA for an intermediate step in our construction of the random local access model (described in more detail in Section 2.2). We thus precisely define both LCAs and random local access models below.

► **Definition 7** (Local computation algorithm). Given an object family  $(\Pi, \Omega)$  with input  $\Pi$  and sample space  $\Omega \subseteq \Sigma^n$  (for some alphabet  $\Sigma$ ), a  $(t(n), \delta(n))$ -local computation algorithm (LCA) provides an oracle  $\mathcal{A}$  that implements query access to some arbitrary  $\sigma \in \Omega$  that satisfies the following conditions:

- $\mathcal{A}$  has query access to the input description  $\Pi$  and to a tape of public random bits  $\mathbf{R}$ .
  - $\mathcal{A}$  gets a sequence of queries  $i_1, \dots, i_q$  for any  $q > 0$ , and after each query  $i_j$ , it produces an output  $\sigma_{i_j}$  such that the outputs  $\sigma_{i_1}, \dots, \sigma_{i_q}$  are consistent with some  $\sigma \in \Omega$ .
  - The probability of success over all  $q$  queries is at least  $1 - \delta(n)$  (where  $\delta(n) < 1/3$ ).
  - The expected running time of  $\mathcal{A}$  on any query is at most  $t(n)$ , which is sublinear in  $n$ .
- We further say that  $\mathcal{A}$  is query-oblivious if the outputs of  $\mathcal{A}$  do not depend on the order of the queries but depend only on  $\Pi$  and  $\mathbf{R}$ .

Motivated by the above, we give a formal definition of the random local access model introduced in [5].

► **Definition 8** (Random local access). *Given a random object family  $(\Pi, \Omega, \mu)$  with input  $\Pi$ , sample space  $\Omega \subseteq \Sigma^n$  (with alphabet  $\Sigma$ ) and distribution  $\mu$  supported on  $\Omega$ , a  $(t(n), \delta(n))$ -random local access implementation of a family of query functions  $\{F_1, F_2, \dots\}$  provides an oracle  $\mathcal{A}$  with the following properties:*

- $\mathcal{A}$  has query access to the input description  $\Pi$  and to a tape of public random bits  $\mathbf{R}$ .
- For a given input  $\Pi$ , upon being queried with  $F_i$ , the oracle with probability  $1 - o(1)$  uses at most  $t(n)$  resources (where  $t(n)$  is sublinear in  $n$ ) to return the value  $\mathcal{A}(\Pi, \mathbf{R}, F_i(Y))$  for some specific  $Y \in \Omega$ .
- The choice of  $Y \in \Omega$  only depends on  $\Pi$  and  $\mathbf{R}$ .
- The distribution of  $Y$  over the randomness  $\mathbf{R}$  satisfies

$$d_{\text{TV}}(Y, \mu) = \frac{1}{2} \sum_{\omega \in \Omega} |\mathbb{P}(Y = \omega) - \mu(\omega)| < \delta(n),$$

where  $\delta(n) \lesssim \frac{1}{n^c}$  for constant  $c > 1$ .

In other words, if  $\mathcal{A}$  is a random local access oracle for a set of queries, then when provided the same input  $\Pi$  and the same random bits  $\mathbf{R}$ , it must provide outputs that are consistent with a single choice of  $Y$  regardless of the order and content of these queries.

► **Remark 9.** *In this work, we do not study or seek to optimize the memory usage of our algorithms. However, there is also a rich literature studying space-efficient and parallelizable local models (e.g., [2, 14, 21]).*

## 2.2 Marking and a query-oblivious LCA

As noted in the introduction, the Lovász local lemma [30] guarantees the existence of a satisfying assignment to any  $(k, d)$ -formula  $\Phi$  if  $d \leq (2ek)^{-1}2^k$ . Furthermore, the Moser-Tardos algorithm [27] gives a simple linear-time algorithm to construct such a satisfying assignment in the above  $(k, d)$ -regime:

- Sample  $v_1, \dots, v_n \stackrel{\mathbf{R}}{\leftarrow} \{0, 1\}$  uniformly at random;
- While there exists a clause  $C \in \mathcal{C}$  that is violated by the current assignment, resample variables in  $C$  uniformly at random.

The Lovász local lemma can be made *quantitative*, showing that not only is there some satisfying assignment to  $\Phi$  if  $d \leq (2ek)^{-1}2^k$ , but both that there are exponentially many such satisfying assignments and that the marginal distributions  $\mu_v$  are approximately uniform with  $\mu_v(1) \leq \frac{1}{2} \exp(1/k)$  (see [25, 10]). Such *local uniformity* is critical to the success of algorithms that approximately sample from  $\mu$ . In his breakthrough work, Moitra [25] noted that this local uniformity continues to hold for conditional distributions  $\mu^\sigma$  provided that each clause has a sufficiently large number of free variables under partial assignment  $\sigma$ . This motivates the idea of a *marking*, as introduced in [25], which is a careful selection (via the Lovász local lemma) of a linear sized subset of variables  $\mathcal{M} \subseteq V$  that has the following properties:

- For every clause  $C$ ,  $|\text{vbl}(C) \cap \mathcal{M}| \gtrsim k$ . Having a large number of marked variables in each clause would hopefully result in a desired *shattering condition*; namely, we can sample a partial assignment  $\sigma \in \{0, 1\}^{\mathcal{M}}$  on this marking that partitions the original formula into sufficiently small connected components.
- For every clause  $C$ ,  $|\text{vbl}(C) \setminus \mathcal{M}| \gtrsim k$ . This large non-intersection preserves the *local uniformity* of the marginal distributions of the as yet-unsampled variables in  $\mathcal{M}$ .



The general strategy of arguments following the marking approach is to show that it is “easy” to sample a partial assignment  $\sigma \in \{0, 1\}^{\mathcal{M}}$ , and moreover, conditioned on any such assignment, the reduced formula  $\Phi^\sigma$  is very likely to shatter into sufficiently small connected components such that the remaining variable assignments can be efficiently sampled from the conditional distribution by solving a smaller instance of the original problem. We now make this notion precise.

► **Definition 10** ( $\alpha$ -marking). *Given  $(k, d)$ -formula  $\Phi$  and constant  $\alpha \in (0, 1/2)$ , we say that  $\mathcal{M} \subseteq V$  is an  $\alpha$ -marking if for every  $C \in \mathcal{C}$ ,  $|\text{vbl}(C) \cap \mathcal{M}| \geq \alpha k$  and  $|\text{vbl}(C) \setminus \mathcal{M}| \geq \alpha k$ .*

In this work, we locally, greedily construct an  $\alpha$ -marking  $\mathcal{M}$  using a local implementation of Moser-Tardos. We will further argue that because of the shattering property, we can locally compute the connected component of  $\Phi_v^\sigma$  for some  $\sigma \sim \mu_{\mathcal{M}}$  and a given vertex  $v$ , without having to actually assign linearly many vertices.

Precisely, we construct a query-oblivious LCA,  $\text{IsMarked}(\cdot)$ , where for a  $(k, d)$ -formula  $\Phi$  and a given range of  $\alpha \in (0, 1)$ ,  $\text{IsMarked}(\cdot)$  can take in any variable  $v \in V$  and output either 0 or 1 indicating whether  $v$  is in some  $\alpha$ -marking of  $V$ . Moreover,  $\text{IsMarked}(\cdot)$  takes sublinear time and when queried on all of  $V$ , gives a consistent  $\alpha$ -marking of  $V$ .

► **Theorem 11.** *Let  $\Phi = (V, \mathcal{C})$  be a  $(k, d)$ -formula. Suppose there exist constants  $1/2 < \beta_1 < \beta_2 < 1 - \alpha$  that satisfy the following conditions:*

$$\begin{aligned} 4\alpha &< 2(1 - \beta_2) < 1 - \beta_1, \\ 16k^4 d^5 &\leq 2^{(\beta_1 - h(1 - \beta_1))k}, \\ 16k^4 d^5 &\leq 2^{(\beta_2 - \beta_1)k - h\left(\frac{\beta_2 - \beta_1}{1 - \beta_1}\right)(1 - \beta_1)k}, \\ \delta \mapsto (\beta_2 - \delta) - h\left(\frac{\beta_2 - \delta}{1 - \delta}\right)(1 - \delta) &\text{ is decreasing on } 0 \leq \delta \leq \beta_1, \\ 2e(kd + 1) &< 2^{(1 - h\left(\frac{\alpha}{1 - \beta_2}\right))(1 - \beta_2)k}. \end{aligned} \tag{1}$$

(Here  $h(x) := -x \log_2(x) - (1 - x) \log_2(1 - x)$  is the binary entropy.)

Fix constant  $c > 0$ . Then there exists a  $\text{polylog}(n)$  time local computation algorithm  $\text{IsMarked}(\cdot)$  which, given any variable  $v \in V$ , returns an assignment  $s_v \in \{0, 1\}$  denoting whether  $v$  is contained in an  $\alpha$ -marking of  $\Phi$ . Moreover, with probability at least  $1 - n^{-c}$ , the responses for all  $v \in V$  yield a consistent  $\alpha$ -marking of  $\Phi$ .

The construction of  $\text{IsMarked}(\cdot)$  and the verification that it is a query-oblivious LCA draws inspiration from the approach in [2] to obtain an oblivious LCA for hypergraph 2-coloring. From a high level,  $\text{IsMarked}(\cdot)$  determines  $s_v$  by randomly and greedily including variables in the marking and subsequently determining those that must/must not be in the marking, and finally (if needed) performing a rejection sampling on a smaller connected component that contains  $v$ . The formulae in Equation (1) are some technical conditions that guarantee this process to go through. We refer the readers to Appendix E of the full version (arxiv:2409.03951) for a proof of Theorem 11.

### 3 A random local access algorithm for $k$ -SAT solutions

In this section, we introduce the main algorithm (Algorithm 1) that locally accesses the assignment of a variable  $v \in V$  in a uniformly random satisfying assignment  $\sigma$  to  $\Phi$ .

### 13:8 Random Local Access for Sampling $k$ -SAT Solutions

Recall from Theorem 1 that given a  $(k, d)$ -formula  $\Phi = (V, \mathcal{C})$ , variable  $v \in V$ , and any constant  $c > 0$ , we wish to come up with a random local access algorithm  $\mathcal{A}$  such that (1) the expected running time of  $\mathcal{A}$  is  $\text{poly log}(n)$ , and (2) the output  $\hat{\mu}_v$  of  $\mathcal{A}$  for every  $v \in V$  satisfies  $d_{\text{TV}}(\text{Law}(\hat{\mu}_v), \mu_v) \leq \frac{1}{n^c}$ . As a high level description, given the above inputs, Algorithm 1 performs the following:

1. Locally decides whether  $v$  lies in an  $\alpha$ -marking of  $\Phi$  using  $\text{IsMarked}(v)$  (Theorem 11), such that the responses for all  $v \in V$  yield a consistent  $\alpha$ -marking  $\mathcal{M} \subseteq V$ .
2. Suppose  $\sigma \sim \mu$  is a uniformly random satisfying assignment to  $\Phi$ . If  $v$  is marked, then we sample  $\sigma(v)$  by computing  $\text{MarginSample}(v)$  (adapted from [9, Algorithm 8]), which may make further recursive calls to sample other *marked* variables.
3. If  $v$  is not marked, then we perform a depth-first search starting from  $v$  to compute  $\sigma$  restricted to  $\mathcal{M}$ . We start from  $\sigma = \emptyset$ ; for every  $w \in \mathcal{M}$  we encounter that we have not yet sampled, we compute  $\text{MarginSample}(w)$  and update  $\sigma(w)$ , to eventually obtain a (w.h.p. polylogarithmic in size) connected component  $\Phi_v := \Phi_v^\sigma$  that contains  $v$ . This part is carried out by the algorithm  $\text{Conn}(v)$ .  
After obtaining the connected component  $\Phi_v^\sigma$ , we call  $\text{UniformSample}$  (Theorem 13) to sample a uniformly random satisfying assignment to  $\Phi_v^\sigma$  and extend  $\sigma$ . We then output  $\sigma(v)$ .

■ **Algorithm 1** The sampling algorithm.

---

**Input:**  $k$ -CNF formula  $\Phi = (V, \mathcal{C})$  and variable  $v \in V$   
**Output:** random assignment  $\sigma(v) \in \{0, 1\}$

- 1: **if**  $\text{IsMarked}(v)$  **then**
- 2:     **return**  $\text{MarginSample}(v)$
- 3: **else**
- 4:      $\Phi_v \leftarrow \text{Conn}(v)$
- 5:      $\sigma \leftarrow \text{UniformSample}(\Phi_v)$
- 6:     **return**  $\sigma(v)$

---

To illustrate the workflow of Algorithm 1, we present a toy example on a small  $k$ -CNF formula, omitting some subroutine details for brevity.

► **Example 12.** Suppose  $k = 3$ ,  $d = 2$  and  $\alpha = 1/3$ . Consider the following  $(k, d)$ -formula  $\Phi = (V, \mathcal{C})$  with variables  $V = \{x_1, x_2, x_3, x_4, x_5\}$  and clauses  $\mathcal{C} = \{C_1, C_2, C_3\}$ , where

$$C_1 = (x_1 \vee x_2 \vee \neg x_3), \quad C_2 = (\neg x_2 \vee x_3 \vee x_4), \quad C_3 = (\neg x_4 \vee x_5 \vee \neg x_1).$$

Suppose we wish to approximately sample  $\sigma(x_1)$  using the local access algorithm  $\mathcal{A}$ . We run  $\text{IsMarked}$  on each variable; suppose the resulting marking is  $\mathcal{M} = \{x_2, x_5\}$ , so that  $x_1$  is not marked.

Since  $x_1 \notin \mathcal{M}$ , we call  $\text{Conn}(x_1)$  to explore the connected component of  $\Phi^{\mathcal{M}}$  containing  $x_1$ . We initialize  $\mathcal{S} := \{C_1, C_3\}$  and partial assignment  $\sigma = \emptyset$ .

- Process clause  $C_1$ . Since  $x_2 \in \mathcal{M}$ , we call  $\text{MarginSample}(x_2)$ , which returns (say)  $\sigma(x_2) = 0$ . The clause becomes  $(x_1 \vee 0 \vee \neg x_3) = (x_1 \vee \neg x_3)$ , which is unsatisfied with the current  $\sigma$ . We add adjacent clause  $C_2$  (via  $x_2$  and  $x_3$ ) to  $\mathcal{S}$ .
- Process clause  $C_3$ . Since  $x_5 \in \mathcal{M}$ , we call  $\text{MarginSample}(x_5)$ , which returns (say)  $\sigma(x_5) = 1$ . The clause becomes  $(\neg x_4 \vee 1 \vee \neg x_1)$ , which is satisfied.
- Process clause  $C_2$ . We already have  $\sigma(x_2) = 0$ , so  $\neg x_2 = 1$ , and the clause is satisfied. Remove  $C_2$  from  $\mathcal{S}$ .



Now the discovered component  $\Phi_{x_1}^\sigma$  includes (1) clauses  $C_1$ , (2) marked variable assignments  $\sigma(x_2) = 0$ ,  $\sigma(x_5) = 1$ , (3) free variables:  $x_1, x_3, x_4$ . We now run `UniformSample` on the reduced subformula  $C'_1 = (x_1 \vee \neg x_3)$ . The satisfying assignments  $(x_1, x_3)$  are  $\{(1, 0), (1, 1), (0, 0)\}$ . We pick one uniformly at random, say  $(x_1, x_3) = (1, 0)$ . Then we return  $\hat{\mu}_{x_1} = 1$ .

Algorithm 1 is our main routine; it has several subroutines `IsMarked`, `MarginSample`, `Conn`, and `UniformSample` that we now describe. Recall that `IsMarked` has been discussed in Theorem 11. We now introduce the algorithm `UniformSample` given by the work of He–Wang–Yin [18].

► **Theorem 13** ([18, Theorems 5.1 and 6.1]). *Suppose  $\Phi$  is a  $(k, d)$ -CNF formula on  $n$  variables with  $k \cdot 2^{-k} \cdot (dk)^5 \cdot 4 \leq \frac{1}{150e^3}$ . Then there exists an algorithm `UniformSample` that terminates in  $O(k^3(dk)^9n)$  time in expectation, and outputs a uniformly random satisfying assignment to  $\Phi$ .*

As seen in Theorem 13, for an  $n$ -variable  $(k, d)$ -CNF formula, the algorithm has running time  $O(n)$ . However, as we will only apply `UniformSample` to connected components of size  $\text{poly} \log(n)$  (that are shattered by the partial assignment on the marked variables), the execution of `UniformSample` in Line 6 of Algorithm 1 will only take polylogarithmic time.

We will define subroutines `MarginSample` and `Conn` below and show that they satisfy the desired correctness and efficiency properties, beginning by verifying `MarginSample` has the desired properties in Section 4.

► **Theorem 14.** *Let  $\Phi = (V, \mathcal{C})$  be a  $(k, d)$ -formula,  $\alpha > 0$ , and  $\mathcal{M} \subseteq V$  be an  $\alpha$ -marking as in Definition 10. Fix positive constant  $c > 0$ . Suppose  $\theta := 1 - \frac{1}{2} \exp\left(\frac{2edk}{2^{\alpha k}}\right) \geq 0.4$ ,  $36ed^3k^4 \cdot 0.6^{\alpha k} \leq 1/2$ , and  $2^{-\frac{1}{48dk^4}} \cdot e^{\frac{2d^2/\alpha}{2^{\alpha k}}} \leq 0.9$ . Then there exists a random local access algorithm `MarginSample`( $\cdot$ ) such that for every  $u \in \mathcal{M}$ , `MarginSample`( $u$ ) returns a random value in  $\{0, 1\}$  that satisfies the following:*

1. *Let  $\nu := \mu_{\mathcal{M}}$  and  $\hat{\nu}$  be the joint distribution of the outputs  $(\text{MarginSample}(u))_{u \in \mathcal{M}}$ . Then we have  $d_{\text{TV}}(\hat{\nu}, \nu) < n^{-c}$ .*
2. *For every  $u \in \mathcal{M}$ , the expected cost of `MarginSample`( $u$ ) is  $\text{poly} \log(n)$ .*

We also require `Conn` to be correct and have low expected cost.

► **Theorem 15.** *Let  $\Phi = (V, \mathcal{C})$  be a  $(k, d)$ -formula,  $\alpha > 0$ , and  $\mathcal{M} \subseteq V$  be an  $\alpha$ -marking as in Definition 10. Suppose  $\theta := 1 - \frac{1}{2} \exp\left(\frac{2edk}{2^{\alpha k}}\right) \geq 0.4$  and  $d \leq 2^{\alpha k/4}$ . Then there exists a random local access algorithm `Conn`( $\cdot$ ) such that for every  $v \in V \setminus \mathcal{M}$ , `Conn`( $v$ ) returns the connected component in  $\Phi^\sigma$  that contains  $v$ , where  $\sigma$  is the partial assignment given by  $(\text{MarginSample}(u))_{u \in \mathcal{M}}$ . Moreover, for every  $v \in V \setminus \mathcal{M}$ , the expected cost of `Conn`( $v$ ) is  $\text{poly} \log(n)$ .*

From a high level, the algorithm `Conn`( $v$ ) explores the clauses and marked variables in the CNF formula that are reachable from  $v$ , greedily sampling the marked variables and expanding through unsatisfied clauses. It iteratively builds a partial assignment  $\sigma$  that “shatters” the formula into disconnected components, isolating the one containing  $v$ . We will verify Theorem 15 in Appendix D of the full version (arxiv:2409.03951).

## 4 Proof of Theorem 14

In this section we show Theorem 14. Throughout, let  $\Phi = (V, \mathcal{C})$  be a  $(k, d)$ -formula,  $\alpha > 0$ , and  $\mathcal{M} \subseteq V$  be an  $\alpha$ -marking with  $|\mathcal{M}| = m$ . We introduce a local access algorithm `MarginSample` that satisfies the key property that the joint distribution of outputs

## 13:10 Random Local Access for Sampling $k$ -SAT Solutions

$(\text{MarginSample}(u))_{u \in \mathcal{M}}$  consistently follows the marginal distribution  $\mu_{\mathcal{M}}$ . In particular,  $(\text{MarginSample}(u))_{u \in \mathcal{M}}$  will simulate the output of a *systematic scan Glauber dynamics* on the marked variables.

- **Definition 16.** Let  $\mathcal{M} = \{u_1, \dots, u_m\}$  denote the marked variables (so  $|\mathcal{M}| = m$ ).
- For every time  $t \in \mathbb{Z}$ , define  $i(t) := (t \bmod m) + 1$ .
  - For every time  $t \in \mathbb{Z}$  and  $u_i \in \mathcal{M}$ , define  $\text{pred}_{u_i}(t) := \max\{s \leq t : i(s) = i\}$ .

In the systematic scan Glauber dynamics, we always resample vertex  $u_{i(t)}$  at time  $t$  (as opposed to randomly choosing a variable to resample at each step). For every  $u \in \mathcal{M}$  and time  $t$ ,  $\text{pred}_u(t)$  denotes the most recent time up to time  $t$  at which  $u$  got resampled. Observe that for all  $t \in \mathbb{Z}$  and  $u \in \mathcal{M}$ , we have  $t - m < \text{pred}_u(t) \leq t$ . Moreover, for all  $w \neq u_{i(t)}$ , we have  $\text{pred}_w(t) < t$ .

■ **Algorithm 2** Systematic scan projected Glauber dynamics.

---

**Input:** a  $k$ -CNF formula  $\Phi = (V, \mathcal{C})$ , a set of marked variables  $\mathcal{M} \subseteq V$ , time  $T$ , and an ordering  $\mathcal{M} = \{u_1, \dots, u_m\}$

**Output:** a random assignment  $X_* \in \{0, 1\}^{\mathcal{M}}$

- 1: Sample  $X_0(u) \xleftarrow{\mathbf{R}} \{0, 1\}$  uniformly and independently for each  $u \in \mathcal{M}$
  - 2: **for**  $t = 1, \dots, T$  **do**
  - 3:   Let  $u := u_{i(t)}$
  - 4:   For all  $u' \in \mathcal{M} \setminus \{u\}$ ,  $X_t(u') \leftarrow X_{t-1}(u')$
  - 5:   Sample  $X_t(u) \xleftarrow{\mathbf{R}} \mu_u(\cdot \mid X_{t-1}(\mathcal{M} \setminus \{u\}))$
  - 6: **Return**  $X_* = X_T$
- 

We refer the readers to Appendix A of the full version (arxiv:2409.03951) for more introduction on the systematic scan Glauber dynamics and its comparison with the (original) projected Glauber dynamics Markov chain for sampling  $k$ -SAT solutions.

We have the following non-quantitative convergence for systematic scan Glauber dynamics.

► **Theorem 17** ([22]). Let  $(X_t)_{t=0}^{\infty}$  denote the Glauber dynamics or the systematic scan Glauber dynamics with stationary distribution  $\pi$ . If  $(X_t)_{t=0}^{\infty}$  is irreducible over  $\Omega \subseteq \{0, 1\}^{\mathcal{M}}$ , then for every  $X_0 \in \Omega$ , we have  $\lim_{t \rightarrow \infty} d_{\text{TV}}(X_t, \pi) = 0$ .

Let  $\mathcal{M} \subseteq V$  be a fixed set of marked variables for a given  $k$ -CNF  $\Phi$ . Let  $\nu := \mu_{\mathcal{M}}$  be the marginal distribution of  $\mu$  on  $\mathcal{M}$ . We will crucially use the following *local uniformity* of Algorithm 2 (the proof in the systematic scan setting follows essentially identically to [10, Lemma 5.3]):

► **Lemma 18.** Suppose  $\Phi$  is a  $k$ -CNF with  $1 < s \leq \frac{2^{\alpha k}}{2\epsilon d k}$ . Let  $X \subseteq \{0, 1\}^{\Lambda}$  be either  $X_0$  or  $X_t(\mathcal{M} \setminus \{u_{i(t)}\})$  for some  $t > 0$  (so correspondingly,  $\Lambda$  is either  $\mathcal{M}$  or  $\mathcal{M} \setminus \{u_{i(t)}\}$  for some  $t > 0$ ). Then for any  $S \subseteq \Lambda$  and  $\sigma : S \rightarrow \{0, 1\}$ , we have

$$\mathbb{P}(X(S) = \sigma) \leq \left(\frac{1}{2}\right)^{|S|} \exp\left(\frac{|S|}{s}\right).$$

Specifically, for any  $v \in \mathcal{M}$  and  $c \in \{0, 1\}$ , we have  $\mathbb{P}(X(v) = c) \geq 1 - \frac{1}{2} \exp(\frac{1}{s}) \geq \frac{1}{2} - \frac{1}{s}$ .

► **Definition 19.** Let  $\pi$  be a distribution on  $\{0, 1\}^{\mathcal{M}}$ . We say that  $\pi$  is  $b$ -marginally lower bounded if for all  $v \in \mathcal{M}$ ,  $\Lambda \subseteq \mathcal{M} \setminus \{v\}$  and feasible partial assignment  $\sigma_{\Lambda} \in \{0, 1, \perp\}^{\Lambda}$ , we have

$$\pi_v^{\sigma_{\Lambda}}(0), \pi_v^{\sigma_{\Lambda}}(1) \geq b.$$

Let  $\pi$  be a  $b$ -marginally lower bounded distribution over  $\{0, 1\}^{\mathcal{M}}$ . For every  $v \in \mathcal{M}$ , we define the following distributions:

1. Lower bound distribution  $\pi_v^{\text{LB}}$  over  $\{0, 1, \perp\}$ : we define  $\pi_v^{\text{LB}} := \pi^{\text{LB}}$ , with

$$\pi^{\text{LB}}(0) = \pi^{\text{LB}}(1) = b, \quad \pi^{\text{LB}}(\perp) = 1 - 2b.$$

2. Padding distribution  $\pi_v^{\text{pad}, \sigma_\Lambda}$  over  $\{0, 1\}$ : for  $\Lambda \subseteq \mathcal{M} \setminus \{v\}$  and feasible partial assignment  $\sigma_\Lambda \in \{0, 1, \perp\}^\Lambda$ , we define

$$\pi_v^{\text{pad}, \sigma_\Lambda}(\cdot) := \frac{\pi_v^{\sigma_\Lambda}(\cdot) - b}{1 - 2b}.$$

Per Lemma 18, we have that  $\nu = \mu_{\mathcal{M}}$  is  $\theta$ -lower bounded for

$$\theta := 1 - \frac{1}{2} \exp\left(\frac{2edk}{2^{\alpha k}}\right) \geq \frac{1}{2} - \frac{1}{2^{(\alpha-2\log_2 d)k}}. \quad (2)$$

#### 4.1 Systematic scan Glauber dynamics on marked variables

We adapt the approach of [9] to a local sampling algorithm by simulating the systematic scan projected Glauber dynamics on  $\mathcal{M}$  from time  $-T$  to 0, which is an aperiodic and irreducible Markov chain by results in [10, 9].

Let  $(X_t)_{-T \leq t \leq 0}$  be the output of Algorithm 2, where we relabel  $X_0, \dots, X_T$  by  $X_{-T}, \dots, X_0$ . We know from Theorem 17 that, as  $T \rightarrow \infty$ , we have  $d_{\text{TV}}((X_0(v))_{v \in \mathcal{M}}, \nu) \rightarrow 0$  where  $\nu = \mu_{\mathcal{M}}$  is the marginal distribution of  $\mu$  on the marked variables. In particular, for every fixed  $n$  and  $\gamma > 0$ , there exists  $T_\gamma \in \mathbb{N}$  such that for all  $T > T_\gamma$ , the Markov chain  $(X_t)_{-T \leq t \leq 0}$  satisfies  $d_{\text{TV}}((X_0(v))_{v \in \mathcal{M}}, \nu) < \gamma$ .

We know from Lemma 18 that  $\nu$  is  $\theta$ -lower bounded, with the lower bound distribution  $\nu^{\text{LB}}$  defined by  $\nu^{\text{LB}}(0) = \nu^{\text{LB}}(1) = \theta$  and  $\nu^{\text{LB}}(\perp) = 1 - 2\theta$ . Thus, for every  $-T < t \leq 0$  and  $u = u_{i(t)}$ , sampling  $X_t(u) \sim \nu_u^{X_{t-1}(\mathcal{M} \setminus \{u\})}$  can be decomposed into the following process:

1. With probability  $2\theta$ , set  $X_t(u)$  to 0 or 1 uniformly at random;
2. With probability  $1 - 2\theta$ , sample  $X_t(u)$  from the padding distribution  $\nu_u^{\text{pad}, X_{t-1}(\mathcal{M} \setminus \{u\})}$ .

Our goal is to obtain  $X_0(u)$  for  $u \in \mathcal{M}$ , which by Theorem 17 will closely follow the marginal distribution  $\nu_u$  for  $T$  sufficiently large. It suffices to simulate the last update for  $u$ . The key observation here is that updates of Glauber dynamics may depend only on a very small amount of extra information. When  $\theta$  is close to  $1/2$ , it is reasonably likely that we can determine  $X_0(u) = X_{\text{pred}_u(0)}(u)$  without any other information. Thus, we can deduce  $X_0(u)$  by recursively revealing only the necessary randomness backwards in time. This method was termed *coupling towards the past* and studied for a variety of constraint satisfaction problems in [9].

We now give a general algorithm  $\text{Glauber}_{T,Y}(t, M, R)$  in Algorithm 5, whose output *simulates* the effects of Algorithm 2 at any particular time  $t$ , by looking *backwards* in time at what happened over the course of running  $(X_t)_{-T \leq t \leq 0}$ . The eventual algorithm  $\text{MarginSample}(u)$  we give in Theorem 14 will retrieve the most recent update of each variable  $u$ , i.e., retrieve the coordinate  $X_0(u)$ .

The algorithm  $\text{Glauber}_{T,Y}(t, M, R)$  contains another subroutine  $\text{LB-Sample}_{T,Y}(t, R)$  that is defined in Algorithm 4. For every time  $t$ , the output of  $\text{LB-Sample}_{T,Y}(t, R)$  follows the distribution  $\nu^{\text{LB}}$  (see Definition 19). In other words,  $\text{LB-Sample}_{T,Y}(t, R)$  preliminarily decides which of the above two regimes we fell into while resampling  $X_{u_{i(t)}}$  at time  $t$ .

Throughout Algorithm 5, we maintain two global data structures.

## 13:12 Random Local Access for Sampling $k$ -SAT Solutions

### ■ Algorithm 3 MarginSample( $u$ ).

**Input:** a  $k$ -CNF formula  $\Phi = (V, \mathcal{C})$ , a set of marked variables  $\mathcal{M} = \{u_1, \dots, u_m\} \subseteq V$ , and a marked variable  $u \in \mathcal{M}$

**Output:** a random value in  $\{0, 1\}$

- 1:  $T \leftarrow T_{n^{-(2+c)}}$
- 2:  $Y \xleftarrow{\mathbf{R}} \{0, 1\}^{\mathcal{M}}$
- 3: return  $\text{Glauber}_{T,Y}(\text{pred}_u(0), M = \perp^{\mathbb{Z}}, R = \emptyset)$

- We let  $M : \mathbb{Z} \rightarrow \{0, 1, \perp\}$  record the previously revealed outcomes of Algorithm 5. That is, for every  $t' \leq 0$  such that  $\text{Glauber}_{T,Y}(t', M, R)$  has already been executed, we set  $M(t')$  to be the outcome of  $\text{Glauber}_{T,Y}(t', M, R)$ .
- We let  $R = \{(s, r_s)\} \subseteq \mathbb{Z} \times \{0, 1, \perp\}$  record the previously revealed outcomes of Algorithm 4. That is, for every  $t' \leq 0$  such that  $\text{LB-Sample}_{T,Y}(t', M, R)$  has already been executed and returned  $r_{t'} \in \{0, 1, \perp\}$ , we add  $\{(t', r_{t'})\}$  to  $R$ .

Since  $T, Y$  remain constant throughout Algorithms 4 and 5, and all recursive calls access and update the same  $M$  and  $R$ , we sometimes write  $\text{Glauber}(t) = \text{Glauber}_{T,Y}(t, M, R)$  and  $\text{LB-Sample}(t) = \text{LB-Sample}_{T,Y}(t, R)$  for short.

At the beginning of  $\text{Glauber}(t)$ , we first check a few early stopping conditions:

- (Lines 1–2) If variable  $u_{i(t)}$  remains its initial assignment  $Y(u_{i(t)})$  at the end of time  $t$  (i.e., is never resampled), we terminate and return  $Y(u_{i(t)})$ .
- (Lines 3–4) If  $|R|$ , the number of stored outcomes of  $\text{LB-Sample}$ , already reaches  $80dk^4 \log n$ , we terminate and return 1.
- (Lines 5–6) If previous iterations have already computed  $\text{Glauber}(t)$  and stored  $M(t) \in \{0, 1\}$ , we terminate and return  $M(t)$ .

If none of the above conditions occurs, we then resample, first by applying  $\text{LB-Sample}(t)$  (Algorithm 4) in Lines 7–8. If  $\text{LB-Sample}(t) \in \{0, 1\}$  (which occurs with probability  $2\theta$ ), we can update  $u = u_{i(t)}$  by choosing an assignment from  $\{0, 1\}$  uniformly at random without investigating the assignments of any other variables at earlier time steps (i.e., we fall into the *zone of local uniformity*).

If  $\text{LB-Sample}(t) = \perp$  (which occurs with probability  $1 - 2\theta$ ), then we fall into a *zone of indecision* and must resample  $u = u_{i(t)}$  from the padding distribution  $\nu_u^{\text{pad}, X_{t-1}(\mathcal{M} \setminus \{u\})}$ . To resample its spin, we slowly proceed backwards in time, lazily computing previously resampled assignments, until we have determined enough information to compute the assignment of  $u$  at the desired time step. Verifying accuracy is somewhat involved, given our lazy computation strategy and partitioning of  $\nu$  into a locally uniform piece and an associated padding distribution. Thus, in Section A we show that Lines 9–19 correctly complete the desired task, proving the following bound on  $d_{\text{TV}}((\text{MarginSample}(u))_{u \in \mathcal{M}}, \nu)$ .

► **Proposition 20.** *For any  $c > 0$  and  $n$  sufficiently large, we have*

$$d_{\text{TV}}((\text{MarginSample}(u))_{u \in \mathcal{M}}, \nu) < n^{-c}.$$

We next require that Algorithm 5 has expected polylogarithmic cost. This is largely a consequence of the local uniformity of  $\mu_{\mathcal{M}}$  and our lazy recursive computation of variable assignments in Algorithm 5.

► **Lemma 21.** *Suppose  $2^{-\frac{1}{48dk^4}} \cdot e^{\frac{2d^2/\alpha}{2\alpha k}} \leq 0.9$ . For every  $t \leq 0$ , the expected cost of  $\text{Glauber}(t)$  is  $O(k^{17} d^{10} \log^2 n/\alpha)$ .*

---

**Algorithm 4**  $\text{LB-Sample}_{T,Y}(t, R)$ .

---

**Input:** An integer  $T \geq 0$  and assignment  $Y \in \{0, 1\}^{\mathcal{M}}$ ; an integer  $t \leq 0$   
**Global variables:** a set  $R \subseteq \mathbb{Z} \times \{0, 1, \perp\}$  and  $\alpha$ -marking  $\mathcal{M} = \{u_1, \dots, u_m\}$   
**Output:** a random value in  $\{0, 1, \perp\}$  distributed as  $\nu^{\text{LB}}$

- 1: **if**  $\text{pred}_{u_{i(t)}}(t) \leq -T$  **then**
- 2:     **return**  $Y(u_{i(t)})$
- 3: **if**  $(t, r) \in R$  **then**
- 4:     **return**  $r$
- 5: Draw  $x \in [0, 1]$  uniformly at random
- 6: **if**  $x < 2\theta$  **then**
- 7:      $R \leftarrow R \cup \{(t, \lfloor x/\theta \rfloor)\}$  and **return**  $\lfloor x/\theta \rfloor$
- 8:  $R \leftarrow R \cup \{(t, \perp)\}$  and **return**  $\perp$

---



---

**Algorithm 5**  $\text{Glauber}_{T,Y}(t, M, R)$ .

---

**Input:** An integer  $T \geq 0$  and assignment  $Y \in \{0, 1\}^{\mathcal{M}}$ ; an integer  $t \leq 0$   
**Global variables:**  $(k, d)$ -CNF  $\Phi = (V, \mathcal{C})$ ,  $\alpha$ -marking  $\mathcal{M} = \{u_1, \dots, u_m\}$ ,  $M : \mathbb{Z} \rightarrow \{0, 1, \perp\}$ , and  $R \subseteq \mathbb{Z} \times \{0, 1, \perp\}$   
**Output:** a random value in  $\{0, 1\}$

- 1: **if**  $\text{pred}_{u_{i(t)}}(t) \leq -T$  **then**
- 2:     **return**  $Y(u_{i(t)})$
- 3: **if**  $|R| \geq 80dk^4 \log n$  **then**
- 4:     **return** 1
- 5: **if**  $M(t) \neq \perp$  **then**
- 6:     **return**  $M(t)$
- 7: **if**  $\text{LB-Sample}_{T,Y}(t, R) \neq \perp$  **then**
- 8:      $M(t) \leftarrow \text{LB-Sample}_{T,Y}(t, R)$  and **return**  $M(t)$
- 9:  $u \leftarrow u_{i(t)}$ ,  $\Lambda \leftarrow \emptyset$ ,  $\sigma_\Lambda \leftarrow \emptyset$ ,  $V' \leftarrow \{u\}$
- 10: **while**  $\exists C \in \mathcal{C}$  such that  $\text{vbl}(C) \cap V' \neq \emptyset$ ,  $\text{vbl}(C) \cap (V \setminus V') \neq \emptyset$  and  $C$  is not satisfied by  $\sigma_\Lambda$  **do**
- 11:     choose  $C$  with the lowest index
- 12:     **if** for all marked  $w \in \text{vbl}(C) \setminus \{u\}$ ,  $\text{LB-Sample}_{T,Y}(\text{pred}_w(t), R)$  does not satisfy  $C$  **then**
- 13:         **for all** marked  $w \in \text{vbl}(C) \setminus \{u\}$  **do**
- 14:              $\Lambda \leftarrow \Lambda \cup \{w\}$  and  $\sigma_\Lambda(w) \leftarrow \text{Glauber}_{T,Y}(\text{pred}_w(t), M, R)$
- 15:          $V' \leftarrow V' \cup \text{vbl}(C)$
- 16:     **else**
- 17:         **for all** marked  $w \in \text{vbl}(C) \setminus \{u\}$  **do**
- 18:              $\Lambda \leftarrow \Lambda \cup \{w\}$  and  $\sigma_\Lambda(w) \leftarrow \text{LB-Sample}_{T,Y}(\text{pred}_w(t), R)$
- 19: Let  $\Psi$  be the connected component in  $\Phi^{\sigma_\Lambda}$  with  $u$ , and sample  $c \sim (\nu_\Psi)_u^{\text{pad}, \sigma_\Lambda}$
- 20:  $M(t) \leftarrow c$  and **return**  $M(t)$

---

## 13:14 Random Local Access for Sampling $k$ -SAT Solutions

We prove Lemma 21 in Section B. These two results together allow us to prove Theorem 14.

**Proof of Theorem 14.** The theorem directly follows from combining Proposition 20 and Lemma 21. By Proposition 20, we know that the joint distribution  $\hat{\nu} := (\text{MarginSample}(u))_{u \in \mathcal{M}}$  satisfies  $d_{\text{TV}}(\hat{\nu}, \nu) < n^{-c}$ . By Lemma 21, we know that for every  $u \in \mathcal{M}$ , the expected cost of  $\text{MarginSample}(u) = \text{Glauber}_{T,Y}(\text{pred}_u(0), M = \perp^{\mathbb{Z}}, R = \emptyset)$  is of order  $\text{poly} \log(n)$ .  $\blacktriangleleft$

### 5 Proof of the main theorem

We are finally able to state and prove the formal version of Theorem 1. Before picking the relevant parameters, we first collect the list of conditions required to apply Theorems 13–15.

► **Condition 22.**

$$\begin{aligned} k \cdot 2^{-\alpha k} \cdot (dk)^5 \cdot 4 &\leq \frac{1}{150e^3}, \\ \theta := 1 - \frac{1}{2} \exp\left(\frac{2edk}{2^{\alpha k}}\right) &\geq 0.4, \\ 36ed^3k^4 \cdot 0.6^{\alpha k} &\leq 1/2, \\ 2^{-\frac{1}{48dk^4}} \cdot e^{\frac{2d^2/\alpha}{2^{\alpha k}}} &\leq 0.9, \\ d &\leq 2^{\alpha k/4}. \end{aligned} \tag{3}$$

Recall that we also need conditions Equation (1) to apply Theorem 11. One can verify that for  $d \leq 2^{k/400}$  and  $k$  sufficiently large, we can choose all the parameters appropriately so that Equations (1) and (3) are satisfied.

► **Lemma 23.** *Let*

$$\alpha = 1/75, \quad \beta_1 = 0.778, \quad \beta_2 = 0.96.$$

*If  $k$  is sufficiently large, and  $d = d(k) \leq 2^{k/400}$ , then Conditions Equations (1) and (3) are satisfied.*

We defer the proof of Lemma 23 to Appendix F of the full version (arxiv:2409.03951). We can now state and prove Theorem 24, the formal version of our main result.

► **Theorem 24.** *Suppose  $\Phi = (V, \mathcal{C})$  is a  $(k, d)$ -formula with  $d \leq 2^{k/400}$  and  $k$  sufficiently large. Let  $\mu$  be the uniform distribution over satisfying assignments to  $\Phi$ , with marginal distributions  $\mu_v$  for  $v \in V$ . Then for all  $c > 0$ , there is a  $(\text{poly} \log(n), n^{-c})$ -random local access algorithm  $\mathcal{A}$  for sampling the variable assignment of  $v \in V$  as  $\hat{\mu}_v$ , such that*

$$d_{\text{TV}}((\hat{\mu}_v)_{v \in V}, \mu) \leq \frac{1}{n^c}.$$

Here we remark that  $c > 0$  is any fixed constant, and the runtime of  $\mathcal{A}$  depends on it. As written, both the algorithmic runtime and correctness are random, since we give expected running time and bounds on the marginal distribution in total variation distance. However, our algorithm allows derandomising either correctness or running time at the expense of worse bounds on the other.



**Proof.** Suppose  $\Phi = (V, \mathcal{C})$  is a  $(k, d)$ -formula with  $d \leq 2^{k/400}$  and  $k$  sufficiently large, and  $c > 0$  is any constant. Choose parameters

$$\alpha = 1/75, \quad \beta_1 = 0.778, \quad \beta_2 = 0.96.$$

By Lemma 23, we know that with these parameters, conditions Equations (1) and (3) are satisfied. Thus, by Theorem 11, there exists a  $\text{poly log}(n)$  time oblivious local computation algorithm  $\text{IsMarked}(\cdot)$  that with probability at least  $1 - n^{-2c}$  gives a consistent  $\alpha$ -marking  $\mathcal{M} \subseteq V$ .

Suppose  $\text{IsMarked}(\cdot)$  gives a consistent  $\alpha$ -marking  $\mathcal{M} \subseteq V$ . By Theorem 14, we know that there is a random local access algorithm  $\text{MarginSample}(\cdot)$  with expected cost  $\text{poly log}(n)$  such that the distribution of  $(\text{MarginSample}(u))_{u \in \mathcal{M}} \sim \hat{\nu}$  satisfies  $d_{\text{TV}}(\hat{\nu}, \mu_{\mathcal{M}}) < n^{-2c}$ .

Let  $\tau = (\text{MarginSample}(u))_{u \in \mathcal{M}}$ . By the proof of Theorem 15, we know that for every unmarked variable  $v \in V \setminus \mathcal{M}$ , with probability  $1 - n^{-0.1 \log n}$ , the number of clauses in  $\Phi_v^\tau$  is at most  $kd \log^2 n$ . We already proved in Theorem 15 that for every  $v \in V \setminus \mathcal{M}$ , the expected cost of  $\text{Conn}(v)$  is at most  $\text{poly log}(n)$ .

Furthermore, since the reduced formula  $\Phi_v^\tau$  has at least  $\alpha k$  variables and at most  $k$  variables in each clause, and every variable lies in at most  $d$  clauses with  $d \leq 2^{\alpha k/5.4}$ , by Theorem 13, the expected cost of  $\text{UniformSample}(\Phi_v^\tau)$  is asymptotically at most

$$k^3(dk^9)(kd \log^2 n + n^{-0.1 \log n} \cdot n) = \text{poly log}(n).$$

Since both  $\text{Conn}$  and  $\text{UniformSample}$  succeed with probability 1, we get that  $\hat{\mu}$ , the joint distribution of outputs of Algorithm 1 for all  $v \in V$ , satisfies  $d_{\text{TV}}(\hat{\mu}, \mu) < n^{-c}$  for all  $c > 0$ .

By construction, Algorithm 1 is memory-less as it samples all necessary variable assignments in order to compute the assignment of a queried variable  $v$ . Furthermore, Algorithm 1 queried on different variables  $v \in V$  collectively returns an assignment  $\sigma \sim \hat{\mu}$  that has  $d_{\text{TV}}(\hat{\mu}, \mu) < n^{-c}$ . Since this holds for any  $c > 0$  constant, we obtain the desired result.  $\blacktriangleleft$

## 6 Concluding remarks

With more involved refinements and optimizations of the arguments in this work, the density constraint  $d \lesssim 2^{k/400}$  of Theorem 1 can be substantially improved (to something closer to  $d \lesssim 2^{k/50}$ ). We omit these additional calculations in favor of expositional clarity to highlight our main result: random local access models exist for arbitrary bounded degree  $k$ -CNFs at exponential clause density. Furthermore, these arguments can also be adapted (in a similar fashion to [13, 19, 7]) to obtain a random local access model for *random*  $k$ -CNFs in a comparable density regime.

Nonetheless, the limit of the approaches in this work would still fall well short of obtaining random local access for, e.g., approximately  $d \lesssim 2^{k/4.82}$ , the maximum density at which we currently know how to efficiently sample solutions to an arbitrary bounded degree  $k$ -CNF in nearly-linear time [18, 31]. This is because of our reliance on a query-oblivious LCA to construct a local marking and our application of weaker sampling results to a correspondingly reduced CNF.

The approach we take in this work is only one of many possible schema to reduce from existing classical, parallel, and/or distributed algorithms to more local algorithms. Our approach involved using ideas and techniques from a variety of previous works (notably [25, 9, 10]), many of which were in the non-local setting, and adapting them in a novel way to obtain a sublinear sampler. Our approach bears some resemblance to work of [5] where authors adapted a parallel Glauber dynamics algorithm to obtain random local access to

proper  $q$ -colorings, and to work of [3] that used a recursive strategy to give perfect sampling algorithms from certain spin systems in amenable graphs. We expect that many other existing algorithms (including [18, 17, 19, 31]) can be adapted with some work to give random local access algorithms.

---

## References

- 1 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '12, pages 5–14, New York, NY, USA, 2012. doi:10.1145/2213556.2213560.
- 2 Noga Alon, Ronitt Rubinfeld, Shai Vardi, and Ning Xie. Space-efficient local computation algorithms. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1132–1139. SIAM, 2012. doi:10.1137/1.9781611973099.89.
- 3 Konrad Anand and Mark Jerrum. Perfect sampling in infinite spin systems via strong spatial mixing. *SIAM Journal on Computing*, 51(4):1280–1295, 2022. doi:10.1137/21M1437433.
- 4 Amartya Shankha Biswas, Edward Pyne, and Ronitt Rubinfeld. Local access to random walks. In *13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ITCS.2022.24.
- 5 Amartya Shankha Biswas, Ronitt Rubinfeld, and Anak Yodpinyanee. Local access to huge random objects through partial sampling. *11th Innovations in Theoretical Computer Science (ITCS 2020)*, 2020. doi:10.4230/LIPIcs.ITCS.2020.27.
- 6 Zongchen Chen, Aditya Lonkar, Chunyang Wang, Kuan Yang, and Yitong Yin. Counting random  $k$ -sat near the satisfiability threshold. *arXiv preprint*, 2024. doi:10.48550/arXiv.2411.02980.
- 7 Zongchen Chen, Nitya Mani, and Ankur Moitra. From algorithms to connectivity and back: finding a giant component in random  $k$ -SAT. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3437–3470. SIAM, 2023.
- 8 Paul Erdos and László Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. *Infinite and finite sets*, 10(2):609–627, 1975.
- 9 Weiming Feng, Heng Guo, Chunyang Wang, Jiaheng Wang, and Yitong Yin. Towards derandomising markov chain monte carlo. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1963–1990, 2023. doi:10.1109/FOCS57990.2023.00120.
- 10 Weiming Feng, Heng Guo, Yitong Yin, and Chihao Zhang. Fast sampling and counting  $k$ -SAT solutions in the local lemma regime. *J. ACM*, 68(6):Art. 40, 42, 2021. doi:10.1145/3469832.
- 11 Weiming Feng and Yitong Yin. On local distributed sampling and counting. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 189–198, 2018. doi:10.1145/3212734.3212757.
- 12 Andreas Galanis, Leslie Ann Goldberg, Heng Guo, and Andrés Herrera-Poyatos. Fast sampling of satisfying assignments from random  $k$ -SAT. *arXiv*, 2022. arXiv:2206.15308.
- 13 Andreas Galanis, Leslie Ann Goldberg, Heng Guo, and Kuan Yang. Counting solutions to random CNF formulas. *SIAM Journal on Computing*, 50(6):1701–1738, 2021. doi:10.1137/20M1351527.
- 14 Mohsen Ghaffari and Jara Uitto. Sparsifying distributed algorithms with ramifications in massively parallel computation and centralized local computation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1636–1653. SIAM, 2019. doi:10.1137/1.9781611975482.99.
- 15 Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, August 1986. doi:10.1145/6490.6503.
- 16 Oded Goldreich, Shafi Goldwasser, and Asaf Nussboim. On the implementation of huge random objects. *SIAM Journal on Computing*, 39(7):2761–2822, 2010. doi:10.1137/080722771.

- 17 Kun He, Xiaoming Sun, and Kewen Wu. Perfect sampling for (atomic) Lovász local lemma. *arXiv*, 2021. [arXiv:2107.03932](#).
- 18 Kun He, Chunyang Wang, and Yitong Yin. Sampling Lovász local lemma for general constraint satisfaction solutions in near-linear time. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 147–158. IEEE, 2022. [doi:10.1109/FOCS54457.2022.00021](#).
- 19 Kun He, Kewen Wu, and Kuan Yang. Improved bounds for sampling solutions of random CNF formulas. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3330–3361. SIAM, 2023. [doi:10.1137/1.9781611977554.CH128](#).
- 20 Vishesh Jain, Huy Tuan Pham, and Thuy-Duong Vuong. On the sampling Lovász local lemma for atomic constraint satisfaction problems. *arXiv*, 2021. [arXiv:2102.08342](#).
- 21 Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Local computation: Lower and upper bounds. *Journal of the ACM (JACM)*, 63(2):1–44, 2016. [doi:10.1145/2742012](#).
- 22 David A. Levin and Yuval Peres. *Markov chains and mixing times*. American Mathematical Society, Providence, RI, second edition, 2017. With contributions by Elizabeth L. Wilmer, With a chapter on “Coupling from the past” by James G. Propp and David B. Wilson. [doi:10.1090/mbk/107](#).
- 23 Hongyang Liu and Yitong Yin. Simple parallel algorithms for single-site dynamics. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1431–1444, 2022. [doi:10.1145/3519935.3519999](#).
- 24 Yishay Mansour, Aviad Rubinfeld, Shai Vardi, and Ning Xie. Converting online algorithms to local computation algorithms. In *Automata, Languages, and Programming*, pages 653–664, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. [doi:10.1007/978-3-642-31594-7\\_55](#).
- 25 Ankur Moitra. Approximate counting, the Lovász local lemma, and inference in graphical models. *J. ACM*, 66(2):Art. 10, 25, 2019. [doi:10.1145/3268930](#).
- 26 Peter Mörters, Christian Sohler, and Stefan Walzer. A sublinear local access implementation for the chinese restaurant process. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022)*, 2022. [doi:10.4230/LIPIcs.APPROX/RANDOM.2022.28](#).
- 27 Robin A. Moser and Gábor Tardos. A constructive proof of the general Lovász local lemma. *J. ACM*, 57(2):Art. 11, 15, 2010. [doi:10.1145/1667053.1667060](#).
- 28 Moni Naor and Asaf Nussboim. Implementing huge sparse random graphs. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 596–608, 2007. [doi:10.1007/978-3-540-74208-1\\_43](#).
- 29 Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast local computation algorithms. *arXiv*, 2011. [arXiv:1104.1377](#).
- 30 Joel Spencer. Asymptotic lower bounds for ramsey functions. *Discrete Mathematics*, 20:69–76, 1977. [doi:10.1016/0012-365X\(77\)90044-9](#).
- 31 Chunyang Wang and Yitong Yin. A sampling lovász local lemma for large domain sizes. In *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 129–150. IEEE, 2024. [doi:10.1109/FOCS61266.2024.00019](#).

## A Correctness of Algorithm 5

In this section, we show Proposition 20 that with high probability,  $(\text{Glauber}(\text{pred}_u(0)))_{u \in \mathcal{M}}$  faithfully returns the final outcome  $(X_0(u))_{u \in \mathcal{M}}$  of the systematic scan Glauber dynamics  $(X_t)_{-T \leq t \leq 0}$  initialized at  $X_{-T} = Y$ . We will later use Theorem 17 to show that therefore, when  $T$  is set to be sufficiently large, the distribution of  $(\text{MarginSample}(u))_{u \in \mathcal{M}} = (\text{Glauber}(\text{pred}_u(0)))_{u \in \mathcal{M}}$  is close to the marginal distribution  $\nu = \mu_{\mathcal{M}}$ .

► **Proposition 25.** Fix  $t \leq 0$  and let  $u = u_{i(t)}$ . Suppose  $|R| < 80dk^4 \log n$  after the execution of  $\text{Glauber}(t)$  (i.e., Line 4 of Algorithm 5 is never triggered). Then  $\text{Glauber}(t)$  faithfully returns  $X_t(u)$ , where  $(X_t)_{-T \leq t \leq 0}$  is the systematic scan Glauber dynamics started at  $X_{-T} = Y$ .

**Proof.** The statement clearly holds for  $t = -T$  and  $u = u_{i(-T)}$ . Since  $\text{pred}_u(-T) = -T$ , by Lines 1–2 of Algorithm 5, we have  $\text{Glauber}(-T) = Y(u) = X_{-T}(u)$ .

Inductively, for  $-T < t \leq 0$ , assume the proposition for all  $-T \leq t' < t$ . Let  $u = u_{i(t)}$ . Suppose  $|R| < 80dk^4 \log n$  after the execution of  $\text{Glauber}(t)$ . Observe that, since  $\text{pred}_w(t) < t$  for all  $w \neq u$ ,  $\text{Glauber}(t)$  only makes further calls to  $\text{Glauber}(t')$  with  $t' < t$ . Thus, by the inductive hypothesis, all further calls of  $\text{Glauber}(t')$  have correctly returned the outcomes  $X_{t'}(u_{i(t')})$ .

We wish to show that the resampled outcome  $\text{Glauber}(t)$  follows the marginal distribution  $\nu_u^{X_{t-1}(\mathcal{M} \setminus \{u\})}$ . Per Lines 5–6, we may assume that  $\text{Glauber}(t)$  has never been called before, in which case we directly go to Line 7 of Algorithm 5. Lines 7–8 guarantee that with probability  $2\theta$ , we assign  $X_t(u)$  to be 0 or 1 uniformly at random. It remains to show that in Lines 9–19, we are able to resample  $X_t(u)$  from the padding distribution  $\nu_u^{\text{pad}, X_{t-1}(\mathcal{M} \setminus \{u\})}$ .

To show this, we first verify that the sets  $\Lambda$ ,  $V'$  and the partial assignment  $\sigma_\Lambda \in \{0, 1, \perp\}^\Lambda$  obtained in Line 19 satisfy the following four conditions:

- (1)  $u \in V'$ ;
- (2)  $(V' \cap \mathcal{M}) \subseteq \Lambda \cup \{u\}$ ;
- (3) for all  $C \in \mathcal{C}$  such that  $\text{vbl}(C) \cap V' \neq \emptyset$  and  $\text{vbl}(C) \cap (V \setminus V') \neq \emptyset$ ,  $C$  is satisfied by  $\sigma_\Lambda$ ;
- (4) for all marked variables  $w \in V' \setminus \{u\}$ , we have  $\sigma_\Lambda(w) = X_{t-1}(w) \in \{0, 1\}$ ; for all variables  $w \in \Lambda \setminus V'$ , we either have  $\sigma_\Lambda(w) = X_{t-1}(w) \in \{0, 1\}$ , or have  $\sigma_\Lambda(w) = \perp$ .

Here, property (1) holds because  $u$  is added to  $V'$  in the initialization, and  $V'$  never removes variables. Property (2) holds because if variables in some clause  $C$  are added to  $V'$  in Line 15, then all marked variables in  $C$  are added to  $\Lambda$  in Line 14. As the while loop terminates, the opposite condition of Line 10 holds, which is exactly property (3).

We now show property (4). For every  $w \in V' \setminus \{u\}$ , we know that  $w$  is added to  $V'$  in Line 15 due to some clause  $C$ ; if  $w$  is marked, then by Line 14 and the inductive hypothesis, we know that we have assigned  $\sigma_\Lambda(w) \leftarrow \text{Glauber}(\text{pred}_w(t)) = X_{t-1}(w) \in \{0, 1\}$ . For every  $w \in \Lambda \setminus V'$ , by Line 18, we have assigned  $\sigma_\Lambda(w) \leftarrow \text{LB-Sample}(\text{pred}_w(t))$ . If  $\text{LB-Sample}(\text{pred}_w(t)) \neq \perp$ , then we have  $\text{LB-Sample}(\text{pred}_w(t)) = X_{\text{pred}_w(t)}(w) = X_{t-1}(w) \in \{0, 1\}$ .

Let  $\Psi$  denote the connected component in  $\Phi^{\sigma_\Lambda}$  that contains  $u$ . Let  $\mu_\Psi$  denote the distribution of a uniformly random satisfying assignment to  $\Psi$ . By property (3), we know that the connected component in  $\Phi^{\sigma_\Lambda}$  that contains  $u$  is supported on  $V'$ . By property (4), we know that  $X_{t-1}(\mathcal{M} \setminus \{u\})$  is an extension of  $\sigma_\Lambda$ , which means that the connected component in  $\Phi^{X_{t-1}(\mathcal{M} \setminus \{u\})}$  that contains  $u$  is also supported on  $V'$ . Moreover, property (4) implies that  $\sigma_\Lambda(V' \setminus \{u\}) = X_{t-1}(V' \setminus \{u\})$ , which means that the two marginal distributions  $(\mu_\Psi)_u^{\sigma_\Lambda}$  and  $(\mu_\Psi)_u^{X_{t-1}(V' \setminus \{u\})}$  are the same. Altogether, we get that

$$\mu_u^{\sigma_\Lambda} = (\mu_\Psi)_u^{\sigma_\Lambda} = (\mu_\Psi)_u^{X_{t-1}(\mathcal{M} \setminus \{u\})} = \mu_u^{X_{t-1}(\mathcal{M} \setminus \{u\})}.$$

Recall that  $\nu$  is the marginal distribution of  $\mu$  on  $\mathcal{M}$ . Let  $\nu_\Psi$  denote the marginal distribution of  $\mu_\Psi$  on  $\mathcal{M}$ . Since  $X_{t-1}(\mathcal{M} \setminus \{u\})$  and  $\sigma_\Lambda$  are both supported on subsets of  $\mathcal{M}$ , the above gives

$$\nu_u^{\sigma_\Lambda} = (\nu_\Psi)_u^{\sigma_\Lambda} = (\nu_\Psi)_u^{X_{t-1}(\mathcal{M} \setminus \{u\})} = \nu_u^{X_{t-1}(\mathcal{M} \setminus \{u\})}.$$

Recall that we wish to sample from  $\nu_u^{\text{pad}, X_{t-1}(\mathcal{M} \setminus \{u\})}$ . Observe that for any partial assignment  $\sigma$ ,  $\nu_u^{\text{pad}, \sigma}$  is a deterministic function of  $\nu_u^\sigma$  (see Definition 19). Since  $\nu_u^{X_{t-1}(\mathcal{M} \setminus \{u\})} = (\nu_\Psi)_u^{\sigma_\Lambda}$ , we have  $\nu_u^{\text{pad}, X_{t-1}(\mathcal{M} \setminus \{u\})} = (\nu_\Psi)_v^{\text{pad}, \sigma_\Lambda}$  as well. Thus, it suffices to

sample  $c \sim (\nu_\Psi)_u^{\text{pad}, \sigma_\Lambda} = \nu_u^{\text{pad}, X_{t-1}(\mathcal{M} \setminus \{u\})}$  which was performed in Line 19. This shows that Lines 9–19 draws  $X_t(u)$  from the padding distribution  $\nu_u^{\text{pad}, X_{t-1}(\mathcal{M} \setminus \{u\})}$ , and finishes the proof.  $\blacktriangleleft$

We now show that the condition  $|R| < 80dk^4 \log n$  happens with high probability. To do this, we show that in a single instance of  $\text{Glauber}(t)$ , the “chain” of further recursions  $\text{Glauber}(t')$  is unlikely to be large. We build the following graph  $G$  to track these recursions.

► **Definition 26.** For every  $C \in \mathcal{C}$  and  $t \leq 0$ , let

$$\phi(C, t) := (\text{vbl}(C), \{\text{pred}_w(t) : w \in \text{vbl}(C) \cap \mathcal{M}\}),$$

i.e.,  $\phi(C, t)$  is the pair comprising the variables of  $C$  and the most recent times that any marked variable in  $C$  was resampled up until time  $t$ . Consider an associated graph  $G_t$  defined by

$$V(G_t) = \{\phi(C, t') : C \in \mathcal{C}, -T \leq t' \leq t\},$$

such that  $\phi(C_1, t_1) \sim \phi(C_2, t_2)$  in  $G_t$  if and only if the following holds:

1.  $\text{vbl}(C_1) \cap \text{vbl}(C_2) \neq \emptyset$ ,
2. For  $\mathcal{T} := \{\text{pred}_w(t_1) : w \in \text{vbl}(C_1) \cap \mathcal{M}\} \cup \{\text{pred}_w(t_2) : w \in \text{vbl}(C_2) \cap \mathcal{M}\}$ , we have  $\max \mathcal{T} - \min \mathcal{T} < 2m$  (recall that  $m = |\mathcal{M}|$ ).

We also recall the notion of a 2-tree in a graph or hypergraph.

► **Definition 27.** Let  $G = (V(G), E(G))$  be a graph or hypergraph. We say that  $Z \subseteq V(G)$  is a 2-tree if  $Z$  satisfies the following conditions:

1. for all  $u, v \in Z$ ,  $\text{dist}_G(u, v) \geq 2$ ;
2. the associated graph with vertex set  $Z$  and edge set  $\{\{u, v\} \subseteq Z : \text{dist}_G(u, v) = 2\}$  is connected.

There are not many 2-trees containing a fixed vertex in a graph of bounded maximum degree. We recall one example upper bound below that will be sufficient for our purposes.

► **Observation 28** (see [10, Corollary 5.7]). Let  $G = (V(G), E(G))$  be a graph or hypergraph with maximum degree  $D$ . Then for every  $v \in V$ , the number of 2-trees  $Z \subseteq V(G)$  containing  $v$  with  $|Z| = \ell$  is at most  $\frac{(eD^2)^{\ell-1}}{2}$ .

The following proposition shows that the size of  $R$  is unlikely to be large when we terminate  $\text{Glauber}(t)$  for any  $t \leq 0$ .

► **Proposition 29.** Fix  $t \leq 0$ . Suppose  $\theta := 1 - \frac{1}{2} \exp\left(-\frac{2edk}{2\alpha k}\right) \geq 0.4$  and  $36ed^3k^4 \cdot 0.6^{\alpha k} \leq 1/2$ . Upon the termination of  $\text{Glauber}(t)$ , for every  $\eta \geq 1$ , the size of  $R$  satisfies

$$\mathbb{P}[|R| \geq 24dk^4(\eta + 1)] \leq 2^{-\eta}.$$

**Proof.** Fix  $t \leq 0$  and consider some instance of  $\text{Glauber}(t)$ . For  $t_1 < t_0 \leq t$ , we say that  $\text{Glauber}(t_1)$  is triggered by  $x := \phi(C, t_0)$  if  $\text{Glauber}(t_1)$  is called in Line 14 of  $\text{Glauber}(t_0)$  with clause  $C$ . Let  $W = \{x \in V(G_t) : x \text{ triggers recursive calls}\}$ . We begin by verifying a few basic properties that  $R$  and  $G := G_t$  enjoy.

▷ **Claim 30.** Upon the termination of  $\text{Glauber}(t)$ , we have  $|R| \leq 2k^2|W| + 2k$ .

## 13:20 Random Local Access for Sampling $k$ -SAT Solutions

Proof. Observe that for every  $t_0 \leq t$ ,  $\text{Glauber}(t_0)$  calls  $\text{LB-Sample}$  at most  $k+1$  times (in Line 7 and Line 12 of Algorithm 5) before possibly going into another subroutine  $\text{Glauber}(t_1)$ . Let  $A$  denote the set of timestamps  $t_0$  such that  $\text{Glauber}(t_0)$  was called at least once. Then we have  $|R| \leq (k+1)|A|$ .

Observe that every  $\text{Glauber}(t_0)$  with  $t_0 < t$  is triggered by some  $x \in W$ . Moreover, every  $x \in W$  triggers at most  $k$  subroutines  $\text{Glauber}(t_0)$  in Line 14. Thus, we get that  $|A| \leq k|W| + 1$ , which gives  $|R| \leq (k+1)|A| \leq (k^2 + k)|W| + k + 1 \leq 2k^2|W| + 2k$ .  $\triangleleft$

$\triangleright$  **Claim 31.** The maximum degree of  $G$  is at most  $6k^2d - 1$ .

Proof. Fix  $\phi(C_1, t_1) \in W$ . There are at most  $kd$  clauses  $C_2 \in \mathcal{C}$  such that  $\text{vbl}(C_1) \cap \text{vbl}(C_2) \neq \emptyset$ . For any such  $C_2$ , we count the number of possible  $\phi(C_2, t_2)$  so that  $\mathcal{T} = \{\text{pred}_w(t_1) : w \in \text{vbl}(C_1) \cap \mathcal{M}\} \cup \{\text{pred}_w(t_2) : w \in \text{vbl}(C_2) \cap \mathcal{M}\}$  satisfies  $\max \mathcal{T} - \min \mathcal{T} < 2m$ .

Suppose

$$\begin{aligned} \{\text{pred}_w(t_1) : w \in \text{vbl}(C_1) \cap \mathcal{M}\} &= \{s_1, \dots, s_{k_1}\} \text{ with } s_1 < \dots < s_{k_1}, \\ \{\text{pred}_w(t_2) : w \in \text{vbl}(C_2) \cap \mathcal{M}\} &= \{s'_1, \dots, s'_{k_2}\} \text{ with } s'_1 < \dots < s'_{k_2}. \end{aligned}$$

Observe that  $s_{k_1} \leq t_1 < s_1 + m$  and  $s'_{k_2} \leq t_2 < s'_1 + m$ . If  $\max \mathcal{T} - \min \mathcal{T} < 2m$ , then we have

$$s_{k_1} - 2m < s'_1 < s'_{k_2} < s_1 + 2m,$$

which gives  $t_2 < s'_1 + m < s_1 + 3m$  and  $t_2 \geq s'_{k_2} > s_{k_1} - 2m$ . Thus, we have

$$s_{k_1} - 2m < t_2 < s_1 + 3m.$$

Let  $S := \{s_{k_1} - 2m + 1, s_{k_1} - 2m + 2, \dots, s_1 + 3m - 1\}$ . In particular,  $S$  is an interval of size  $\leq 5m$  given by  $\phi(C_1, t_1)$ .

Observe that as  $t_2$  increments from  $s_{k_1} - 2m + 1$  to  $s_1 + 3m - 1$ , we have  $\{\text{pred}_w(t_2) : w \in \text{vbl}(C_2) \cap \mathcal{M}\} \neq \{\text{pred}_w(t_2 - 1) : w \in \text{vbl}(C_2) \cap \mathcal{M}\}$  only if  $\text{pred}_w(t_2) > \text{pred}_w(t_2 - 1)$  for some  $w \in \text{vbl}(C_2) \cap \mathcal{M}$ . Moreover, since  $|\text{vbl}(C_2) \cap \mathcal{M}| \leq k$  and  $|S| \leq 5m$ , we know that there are at most  $5k$  such numbers  $t_2$  in  $S$ , and these numbers have been completely determined by  $\text{vbl}(C_2)$  and  $S$  (i.e., by  $\phi(C_1, t_1)$  and  $C_2$ ). These numbers partition  $S$  into at most  $5k + 1$  intervals such that  $\{\text{pred}_w(t_2) : w \in \text{vbl}(C_2) \cap \mathcal{M}\}$  is the same for all  $t_2$  on each interval. Thus, for every fixed  $\phi(C_1, t_1)$  and  $C_2$ , the set  $\{\{\text{pred}_w(t_2) : w \in \text{vbl}(C_2) \cap \mathcal{M}\} : t_2 \in S\}$  has size at most  $5k + 1$ .

Therefore, given any  $\phi(C_1, t_1)$ , we can pick a neighbor  $\phi(C_2, t_2) \sim \phi(C_1, t_1)$  by first picking  $C_2$  (which has  $\leq kd$  choices) and then picking an element in  $\{\{\text{pred}_w(t_2) : w \in \text{vbl}(C_2) \cap \mathcal{M}\} : t_2 \in S\}$  (which has  $\leq 5k + 1$  choices). So the number of possible  $\phi(C_2, t_2) \sim \phi(C_1, t_1)$  in  $G$  is at most  $kd(5k + 1) \leq 6k^2d - 1$ .  $\triangleleft$

$\triangleright$  **Claim 32.** Let  $u = u_{i(t)}$  and  $W' = \{\phi(C, t) \in W : u \in \text{vbl}(C)\}$ . Then  $G[W']$  is a clique.

Proof. Consider any two clauses  $C, C'$  such that  $u \in \text{vbl}(C) \cap \text{vbl}(C')$ . Suppose  $\phi(C, t), \phi(C', t) \in W'$ . Clearly  $\text{vbl}(C) \cap \text{vbl}(C') \neq \emptyset$ . Moreover, since the timestamps  $\text{pred}_w(t)$  over all marked variables  $w \in \mathcal{M}$  lie in the range  $\{t - m + 1, \dots, t\}$ , we get that the maximum and minimum of  $\{\text{pred}_w(t) : w \in \text{vbl}(C) \cap \mathcal{M}\} \cup \{\text{pred}_w(t) : w \in \text{vbl}(C') \cap \mathcal{M}\}$  differ by at most  $m - 1 < 2m$ . Thus we have  $\phi(C, t) \sim \phi(C', t)$  in  $G$ .  $\triangleleft$

$\triangleright$  **Claim 33.** Let  $W'$  be as in Claim 32. For every  $x \in W$ , there exists a path  $p_0 \dots p_\ell$  in  $G[W]$  such that  $p_0 \in W'$  and  $p_\ell = x$ .



Proof. Let  $x = \phi(C, t_1)$  be any element in  $W$ . We perform a double induction, the outside on  $t_1$  and the inside on  $C$ .

■ **Base case:  $t_1 = t$ .**

Suppose first that  $t_1 = t$ , so  $x = \phi(C, t)$  for some  $C$ . Let  $u = u_{i(t)}$ . If  $u \in \text{vbl}(C)$ , then  $x \in W'$  and we are done. Inductively, suppose Claim 33 holds for all  $x' = \phi(C', t)$  that triggers a recursive call earlier than  $x$  in Algorithm 5. If  $u \notin \text{vbl}(C)$ , then by the while loop condition Line 10, there must exist some  $x' = \phi(C', t) \in W$  such that  $x'$  triggers a recursive call earlier than  $x$ , and  $\text{vbl}(C) \cap \text{vbl}(C') \neq \emptyset$ . By the inductive hypothesis, there exists a path  $p_0 \dots p_\ell$  in  $G[W]$  such that  $p_0 \in W'$  and  $p_\ell = x'$ . Since the maximum and minimum of  $\{\text{pred}_w(t) : w \in \text{vbl}(C) \cap \mathcal{M}\} \cup \{\text{pred}_w(t) : w \in \text{vbl}(C') \cap \mathcal{M}\}$  differ by at most  $m - 1 < 2m$ , we get that  $x' \sim x$  in  $G$ . Therefore we can extend the path  $p_0 \dots p_\ell$  with  $p_{\ell+1} = x$ . This finishes the inductive step.

■ **Inductive step:  $t_1 < t$ .**

Now suppose  $x = \phi(C, t_1)$  with  $t_1 < t$ . By the inductive hypothesis, we can assume Claim 33 for all  $t_0 \in \{t_1 + 1, \dots, t\}$ . Let  $u_1 = u_{i(t_1)}$ .

Suppose first that  $u_1 \in \text{vbl}(C)$ . Since  $t_1 < t$ , there must exist  $t_0 \in \{t_1 + 1, \dots, t\}$  and  $C' \in \mathcal{C}$  such that  $\phi(C', t_0)$  triggers  $\text{Glauber}(t_1)$ , with  $u_1 \in \text{vbl}(C')$  and  $t_1 = \text{pred}_{u_1}(t_0)$ . Let  $y = \phi(C', t_0)$ . Clearly  $\text{vbl}(C) \cap \text{vbl}(C') \neq \emptyset$ . Since  $t_0 - m < t_1 < t_0$ , we also know that the maximum and minimum of  $\{\text{pred}_w(t_1) : w \in \text{vbl}(C) \cap \mathcal{M}\} \cup \{\text{pred}_w(t_0) : w \in \text{vbl}(C') \cap \mathcal{M}\}$  differ by at most  $2m - 1 < 2m$ . Thus we have  $x \sim y$  in  $G$ . By the inductive hypothesis for  $t_0$ , there exists a path  $p_0 \dots p_\ell$  in  $G[W]$  such that  $p_0 \in W'$  and  $p_\ell = y$ . Since  $x \sim y$  in  $G$ , we can extend the path by  $p_{\ell+1} = x$ .

Inductively, suppose  $u_1 \notin \text{vbl}(C_1)$ , and Claim 33 holds for all  $t_0 \in \{t_1 + 1, \dots, t\}$  and for all  $x' = \phi(C', t_1)$  that triggers a recursive call earlier than  $x$ . Then there must exist some  $x' = \phi(C', t_1) \in W$  such that  $x'$  triggers a recursive call earlier than  $x$ , and  $\text{vbl}(C) \cap \text{vbl}(C') \neq \emptyset$ . By the inductive hypothesis, there exists a path  $p_0 \dots p_\ell \in G[W]$  such that  $p_0 \in W'$  and  $p_\ell = x'$ . Since the maximum and minimum of  $\{\text{pred}_w(t_1) : w \in \text{vbl}(C) \cap \mathcal{M}\} \cup \{\text{pred}_w(t_1) : w \in \text{vbl}(C') \cap \mathcal{M}\}$  differ by at most  $m - 1 < 2m$ , we get that  $x' \sim x$  in  $G$ . Thus we can extend the path by  $p_{\ell+1} = x$ . This finishes the inductive step.  $\triangleleft$

▷ **Claim 34.** For all  $x = \phi(C, t') \in W$  and  $w \in \text{vbl}(C) \cap \mathcal{M}$ , the function  $\text{LB-Sample}(\text{pred}_w(t'))$  does not satisfy  $C$ .

Proof. This directly follows from Lines 12–14 of Algorithm 5. Since  $\phi(C, t')$  triggers a recursion in Line 14, by the condition in Line 12, for all marked variables  $w \in \text{vbl}(C) \cap \mathcal{M}$ , the function  $\text{LB-Sample}(\text{pred}_w(t'))$  does not satisfy  $C$ .  $\triangleleft$

With these claims, we can now prove the proposition. Fix  $t \leq 0$  and  $\eta \geq 1$ . Assume  $|R| \geq 24dk^4(\eta + 1)$ , which by Claim 30 implies that  $|W| \geq 6dk^2(\eta + 1)$ . By Claim 33, we know that  $W \cap W' \neq \emptyset$ ; by Claim 31,  $G$  has maximum degree  $\leq 6dk^2 - 1$ . Thus, by a greedy selection, we can find a 2-tree  $Z \subseteq W$  containing some element in  $W'$  such that  $|Z| = \eta + 1$ .

Fix any 2-tree  $Z \subseteq V(G)$  such that  $|Z| = \eta + 1$ . For every  $x = \phi(C, t') \in Z$ , if  $x \in W$ , then we know from Claim 34 that  $\text{LB-Sample}(\text{pred}_w(t'))$  does not satisfy  $C$  for all  $w \in \text{vbl}(C) \cap \mathcal{M}$ . Since the latter happens with probability at most  $(1 - \theta)^{\alpha k}$ , we have

$$\mathbb{P}[x \in W] \leq (1 - \theta)^{\alpha k}.$$

## 13:22 Random Local Access for Sampling $k$ -SAT Solutions

Since  $Z$  is a 2-tree, we know that for every two  $x = \phi(C_1, t_1), y = \phi(C_2, t_2) \in Z$ , we have  $\{\text{pred}_w(t_1) : w \in \text{vbl}(C_1) \cap \mathcal{M}\} \cap \{\text{pred}_w(t_2) : w \in \text{vbl}(C_2) \cap \mathcal{M}\} = \emptyset$  (as otherwise  $C_1$  and  $C_2$  would share a variable, and the union of these two sets would span an interval of size at most  $2(m-1) + 1 = 2m-1$ , meaning that  $x \sim y$  in  $G$ , which contradicts the fact that  $Z$  is an independent set in  $G$ ). In particular, the sets

$$\{\{\text{pred}_w(t') : w \in \text{vbl}(C) \cap \mathcal{M}\} : \phi(C, t') \in Z\}$$

are disjoint. Thus, for every fixed 2-tree  $|Z| = \eta + 1$  in  $V(G)$ , we have

$$\mathbb{P}[Z \subseteq W] \leq \mathbb{P}[x \in W \text{ for all } x \in Z] \leq (1 - \theta)^{\alpha k \eta}.$$

Let  $\mathcal{T}$  denote the set of 2-trees of  $V(G)$  of size  $\eta + 1$  that intersects with  $W'$ . Since  $|W'| \leq d$  and  $G$  has maximum degree at most  $6k^2d - 1$ , by Observation 28, we have

$$|\mathcal{T}| \leq d \cdot \frac{(e(6k^2d)^2)^\eta}{2} \leq (36ed^3k^4)^\eta.$$

Thus, we have

$$\mathbb{P}[|W| \geq 6dk^2(\eta + 1)] \leq \sum_{Z \in \mathcal{T}} \mathbb{P}[Z \subseteq W] \leq (36ed^3k^4)^\eta (1 - \theta)^{\alpha k \eta} \leq 2^{-\eta},$$

where the last step used the assumption that  $\theta \geq 0.4$  and  $36ed^3k^4 \cdot 0.6^{\alpha k} \leq 1/2$ . This implies that

$$\mathbb{P}[|R| \geq 24dk^4(\eta + 1)] \leq \mathbb{P}[|W| \geq 6dk^2(\eta + 1)] \leq 2^{-\eta}. \quad \blacktriangleleft$$

Setting  $\eta = (3 + c) \log n$ , we get the following correctness statement on Algorithm 5.

► **Corollary 35.** *For every  $t \leq 0$ , we have*

$$\mathbb{P}[\text{Glauber}(t) \neq X_t(u)] \stackrel{25}{\leq} \mathbb{P}[|R| > 80dk^4 \log n] \leq \mathbb{P}[|R| > 24dk^4((3+c) \log n + 1)] \stackrel{29}{\leq} n^{-(3+c)}.$$

*In particular, for every  $u \in \mathcal{M}$ , we have*

$$\mathbb{P}[\text{Glauber}(\text{pred}_u(0)) \neq X_0(u)] = \mathbb{P}[\text{Glauber}(\text{pred}_u(0)) \neq X_{\text{pred}_u(0)}(u)] \leq n^{-(3+c)}.$$

*Taking a union bound over all  $u \in \mathcal{M}$ , we get that*

$$\mathbb{P}[(\text{MarginSample}(u))_{u \in \mathcal{M}} \neq X_0] \leq n^{-(2+c)}.$$

Since we have picked  $T$  in Algorithm 3 sufficiently large so that  $d_{\text{TV}}(X_0, \nu) \leq n^{-(2+c)}$ , we get that the joint output  $(\text{MarginSample}(u))_{u \in \mathcal{M}}$  satisfies  $d_{\text{TV}}((\text{MarginSample}(u))_{u \in \mathcal{M}}, \nu) \leq n^{-c}$ , proving Proposition 20.

## B Efficiency of Algorithm 5

We now move on to show the efficiency of  $\text{Glauber}(t)$  for all  $t \leq 0$ . Observe that for every  $r \geq 48dk^4$ , by Proposition 29, we have

$$\mathbb{P}[|R| \geq r] \leq 2^{-\frac{r}{48dk^4}}. \quad (4)$$

Moreover, we terminate  $\text{Glauber}(t)$  once we reach  $|R| \geq 80dk^4 \log n$ . We will use these information to give an upper bound on the expected cost of  $\text{Glauber}(t)$ .

We start by upper bounding the size of the final sets  $V'$  and  $\Lambda$  in Algorithm 5 in terms of  $|R|$ .

► **Lemma 36.** *The  $V'$  and  $\Lambda$  in Line 19 of Algorithm 5 satisfy*

$$|\Lambda| \leq kd|V'| \leq 2kd^2|R|/\alpha.$$

**Proof.** By Line 10 of Algorithm 5, we know that for all  $u \in \Lambda$ , there exists clause  $C \in \mathcal{C}$  such that  $u \in \text{vbl}(C)$  and  $\text{vbl}(C) \cap V' \neq \emptyset$ . This shows that  $|\Lambda| \leq kd|V'|$ . Observe that  $V'$  contains at least  $|V'|/(\Delta + 1) \geq |V'|/(2dk)$  clauses with disjoint variable sets. Since every marked variable in the clauses added to  $V'$  have gone through at least one round of LB-Sample (per Line 5 and Line 12), we get that  $|R| \geq \alpha k \cdot |V'|/(2dk) = \alpha|V'|/(2d)$ . ◀

To upper bound the expected cost of Line 19, we further need the result of He–Wang–Yin [18] on the existence of a “Bernoulli factory” algorithm  $\text{BF}(\cdot)$  such that for every locally uniform CNF and variable  $v$ , the Bernoulli factory efficiently draws a  $\{0, 1\}$ -random variable according to the padding distribution of  $v$ .

► **Proposition 37** ([18, Lemma 3.10 and Appendix A]). *Let  $\Psi = (V_\Psi, \mathcal{C}_\Psi)$  be a  $k$ -CNF,  $\sigma$  be a feasible partial assignment of  $\Psi$ , and  $\Psi^\sigma = (V_\Psi^\sigma, \mathcal{C}_\Psi^\sigma)$  be the reduced CNF (see Definition 5). Suppose we have*

$$\mathbb{P}[\neg C \mid \sigma] \leq \zeta \quad \text{for every } C \in \mathcal{C}_\Psi^\sigma.$$

*Then there exists an algorithm  $\text{BF}(\cdot)$  such that for every  $v \in V_\Psi^\sigma$ ,*

- $\text{BF}(v)$  with probability 1 returns  $\xi \sim (\mu_\Psi)_v^{\text{pad}, \sigma}$ ;
- $\text{BF}(v)$  has expected cost  $O(k^9 d^6 |\mathcal{C}_\Psi^\sigma| (1 - e\zeta)^{-|\mathcal{C}_\Psi^\sigma|})$ .

We remark that in Algorithm 5, our partial assignment  $\sigma_\Lambda$  is always supported on the marked variables. Since every clause has at least  $\alpha k$  unmarked variables that are not assigned by  $\sigma_\Lambda$ , we get that  $\mathbb{P}[\neg C \mid \sigma_\Lambda] \leq 2^{-\alpha k}$  for every  $C$ . Thus when applying Proposition 37, we will take  $\zeta = 2^{-\alpha k}$ .

We can now give an upper bound on the expected cost of Algorithm 5, proving Lemma 21.

**Proof of Lemma 21.** Let  $A$  be the set of all  $t'$  such that  $\text{Glauber}(t')$  is executed at least once as subroutine of  $\text{Glauber}(t)$ . Since we run  $\text{LB-Sample}(t')$  at the beginning (Line 7) of each round  $\text{Glauber}(t')$ , we know that  $|A| \leq |R|$ .

Observe that for every  $t'$ , if  $\text{Glauber}(t')$  has been computed once, then we will permanently assign  $M(t') = \text{Glauber}(t') \in \{0, 1\}$ , and later executions of  $\text{Glauber}(t')$  will terminate at Line 4. Thus, it suffices to upper bound the cost of every first execution of  $\text{Glauber}(t')$  before entering another  $\text{Glauber}(t'')$ . Multiplying this by  $|A|$  would give an upper bound on the cost of  $\text{Glauber}(t)$ .

Suppose we are at the first time of executing  $\text{Glauber}(t')$  for some  $t'$ . We first estimate the cost of the while loop in Lines 10–18, which is a constant multiple of the number of executions of Line 14 and Line 18. Note that every time we execute Line 14 or Line 18, some  $w$  is added to  $\Lambda$  due to a clause  $C$  chosen in Line 11, with  $w \in \text{vbl}(C)$ . Moreover, each clause  $C$  can be chosen in Line 11 at most once. Thus, the number of pairs  $(w, C)$  that correspond to an execution of Line 14 or Line 18 is at most  $d|\Lambda|$ . Consequently, the cost of the while loop in Lines 10–18 is  $O(d|\Lambda|) = O(kd^3|R|/\alpha)$ .

We now estimate the cost of Line 19. Observe that in Line 19, we can apply the Bernoulli factory in Proposition 37 to compute the padding distribution  $(\nu_\Psi)_u^{\text{pad}, \sigma_\Lambda}$ . Since the connected component  $\Psi$  in Line 19 has at most  $d|V'| \leq 2d^2|R|/\alpha$  clauses, by Proposition 37, the expected cost of Line 19 is at most

$$O(k^9 d^8 |R|/\alpha (1 - e2^{-\alpha k})^{-2d^2|R|/\alpha}).$$

### 13:24 Random Local Access for Sampling $k$ -SAT Solutions

Let  $R_{\max} = 80dk^4 \log n$ . Combining the above and applying Equation (4), we get that the expected cost of  $\text{Glauber}(t)$  is at most

$$\begin{aligned}
& \mathbb{E}[|A| \cdot O(kd^3|R|/\alpha + k^9 d^8|R|/\alpha(1 - e2^{-\alpha k})^{-2d^2|R|/\alpha})] \\
& \leq \mathbb{E}[|R| \cdot O(kd^3|R|/\alpha + k^9 d^8|R|/\alpha(1 - e2^{-\alpha k})^{-2d^2|R|/\alpha})] \\
& \leq \sum_{r=0}^{R_{\max}} \mathbb{P}[|R| \geq r] \cdot r \cdot O(kd^3 r/\alpha + k^9 d^8 r/\alpha(1 - e2^{-\alpha k})^{-2d^2 r/\alpha}) \\
& \leq \sum_{r=0}^{R_{\max}} O(2^{-\frac{r}{48dk^4}} \cdot r \cdot kd^3 r/\alpha + k^9 d^8 r/\alpha(1 - e2^{-\alpha k})^{-2d^2 r/\alpha}) \\
& \leq \sum_{r=0}^{R_{\max}} O\left(2^{-\frac{r}{48dk^4}} \cdot r \cdot \left(\frac{kd^3 r}{\alpha} + \frac{k^9 d^8 r}{\alpha} \cdot e^{\frac{2d^2 r/\alpha}{2\alpha k}}\right)\right) = O(k^9 d^8 R_{\max}^2/\alpha) = O(k^{17} d^{10} \log^2 n/\alpha),
\end{aligned}$$

where in the last step we used the information that

$$2^{-\frac{1}{48dk^4}} \cdot e^{\frac{2d^2/\alpha}{2(1-\alpha)k}} \leq 0.9 \quad \implies \quad \sum_{r=0}^{R_{\max}} \left(2^{-\frac{1}{48dk^4}} \cdot e^{\frac{2d^2/\alpha}{2\alpha k}}\right)^r = O(1). \quad \blacktriangleleft$$