

SAMPLING LOVÁSZ LOCAL LEMMA FOR GENERAL CONSTRAINT SATISFACTION SOLUTIONS IN NEAR-LINEAR TIME

KUN HE, CHUNYANG WANG, AND YITONG YIN

ABSTRACT. We give a fast algorithm for sampling uniform solutions of *general* constraint satisfaction problems (CSPs) in a local lemma regime. The expected running time of our algorithm is near-linear in n and a fixed polynomial in Δ , where n is the number of variables and Δ is the max degree of constraints. Previously, up to similar conditions, sampling algorithms with running time polynomial in both n and Δ , only existed for the almost atomic case, where each constraint is violated by a small number of forbidden local configurations.

Our sampling approach departs from all previous fast algorithms for sampling LLL, which were based on Markov chains. A crucial step of our algorithm is a recursive marginal sampler that is of independent interests. Within a local lemma regime, this marginal sampler can draw a random value for a variable according to its marginal distribution, at a local cost independent of the size of the CSP.

CONTENTS

1. Introduction	1
2. Notations for CSP	5
3. The Sampling Algorithm	6
4. Preliminary on Lovász Local Lemma	10
5. Correctness of Sampling	11
6. Efficiency of Sampling	15
7. Conclusion and Open Problems	36
Acknowledgement	37
References	37
Appendix A. A Bernoulli Factory for Margin Overflow	39
Appendix B. Monotonicity of Constraint Properties	41
Appendix C. $\{2, 3\}$ -tree Witness for Constraint Properties	43

(Kun He) INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES, NO.6 KEXUEYUAN SOUTH ROAD ZHONGGUANCUN, HAIDIAN DISTRICT, BEIJING, CHINA. E-mail: hekun@ict.ac.cn

(Chunyang Wang, Yitong Yin) STATE KEY LABORATORY FOR NOVEL SOFTWARE TECHNOLOGY, NANJING UNIVERSITY, 163 XIANLIN AVENUE, NANJING, JIANGSU PROVINCE, CHINA. E-mails: wcysai@smail.nju.edu.cn, yinyt@nju.edu.cn

1. INTRODUCTION

Constraint satisfaction problems (CSPs) are one of the most fundamental objects in computer science. A CSP is described by a collection of constraints defined on a set of variables. Formally, an instance of constraint satisfaction problem, called a *CSP formula*, is denoted by $\Phi = (V, Q, C)$. Here, V is a set of $n = |V|$ variables; $Q \triangleq \bigotimes_{v \in V} Q_v$ is a product space of all assignments of variables, where each Q_v is a finite domain of size $q_v \triangleq |Q_v| \geq 2$ over where the variable v ranges; and C gives a collection of local constraints, such that each $c \in C$ is a constraint function $c : \bigotimes_{v \in \text{vbl}(c)} Q_v \rightarrow \{\text{True}, \text{False}\}$ defined on a subset of variables, denoted by $\text{vbl}(c) \subseteq V$. An assignment $\mathbf{x} \in Q$ is called *satisfying* for Φ if

$$\Phi(\mathbf{x}) \triangleq \bigwedge_{c \in C} c(\mathbf{x}_{\text{vbl}(c)}) = \text{True}.$$

The followings are some key parameters of a CSP formula $\Phi = (V, Q, C)$:

- *domain size* $q = q_\Phi \triangleq \max_{v \in V} |Q_v|$ and *width* $k = k_\Phi \triangleq \max_{c \in C} |\text{vbl}(c)|$;
- *constraint degree* $\Delta = \Delta_\Phi \triangleq \max_{c \in C} |\{c' \in C \mid \text{vbl}(c) \cap \text{vbl}(c') \neq \emptyset\}|$;¹
- *violation probability* $p = p_\Phi \triangleq \max_{c \in C} \mathbb{P}[\neg c]$, where \mathbb{P} denotes the law for the uniform assignment, in which each $v \in V$ draws its evaluation from Q_v uniformly and independently at random.

The famous *Lovász Local Lemma (LLL)* [EL75] provides a sufficient criterion for the satisfiability of Φ . Specifically, a satisfying assignment for a CSP formula Φ exists if

$$(1) \quad ep\Delta \leq 1.$$

Due to a lower bound of Shearer [She85], such “LLL condition” for the existence of satisfying solution is essentially tight if only knowing p and Δ . On the other hand, the *algorithmic* or *constructive LLL* seeks to find a solution efficiently. A major breakthrough was the Moser-Tardos algorithm [MT10], which guarantees to find a satisfying assignment efficiently under the LLL condition in (1).

The sampling LLL. We are concerned with the problem of *sampling Lovász Local Lemma*, which has drawn considerable attention in recent years [GJL19, Moi19, GLLZ19, GGGY20, Har20, FGYZ21, FHY21, JPV21a, JPV21b, HSW21, GGW21, FGW22]. In the context of CSP, it seeks to provide an efficient sampling algorithm for (nearly) uniform generation of satisfying assignments for the CSPs in an LLL-like regime. This sampling LLL problem is closely related to the problem of estimating the volume of solution spaces or the partition functions of statistical physics systems, and is motivated by fundamental tasks, including the probabilistic inferences in graphical models [Moi19] and the network reliability problems [GJL19, GJ19, GH20].

This problem of sampling LLL turns out to be computationally more challenging than the traditional algorithmic LLL, which requires constructing an arbitrary satisfactory assignment, not necessarily following the correct distribution. For example, when used as a sampling algorithm, the Moser-Tardos algorithm can only guarantee correct sampling on restrictive classes of CSPs [GJL19]. Due to the computational lower bounds shown in [BGG⁺19, GGW21], a strengthened LLL condition with $c \geq 2$:

$$(2) \quad p\Delta^c \lesssim 1,$$

is necessary for the tractability of sampling LLL, even restricted to typical specific sub-classes of CSPs, such as CNF or hypergraph coloring. Here \lesssim ignores the lower-order terms and the constant factor.

In a seminal work of Moitra [Moi19], a very innovative algorithm was given for sampling almost uniform k -CNF solutions assuming an LLL condition $p\Delta^{60} \lesssim 1$. This sampling algorithm was based on deterministic approximate counting by solving linear programs on properly factorized formulas and has a running time of $n^{\text{poly}(k, \Delta)}$. This LP-based approach was later extended to hypergraph coloring [GLLZ19] and random CNF formulas [GGGY20], and finally in a work of Jain, Pham and Vuong [JPV21b] to all CSPs satisfying a substantially improved LLL condition $p\Delta^7 \lesssim 1$. All these deterministic approximate counting based algorithms suffered from an $n^{\text{poly}(k, \Delta)}$ time cost.

¹The constraint degree Δ should be distinguished from the *dependency degree* D , which is the maximum degree of the dependency graph: $D \triangleq \max_{c \in C} |\{c' \in C \setminus \{c\} \mid \text{vbl}(c) \cap \text{vbl}(c') \neq \emptyset\}|$. Note that $\Delta = D + 1$.

Historically, rapidly mixing Markov chains have been the canonical sampling algorithms, and often have near-linear time efficiency. However, for sampling LLL, there used to be a fundamental barrier for Markov chains. That is, despite the ubiquity of solutions, the solution space of CSPs may be highly disconnected through the transition of local Markov chains.

This barrier of disconnectivity was circumvented in a breakthrough of Feng *et al.* [FGYZ21], in which a rapidly mixing *projected* random walk was simulated efficiently on a subset of variables constructed using the marking/unmarking strategy invented in [Moi19]. Assuming an LLL condition $p\Delta^{20} \lesssim 1$, this new algorithm could generate an almost uniform k -CNF solution using a time cost within $\text{poly}(k, \Delta) \cdot n^{1.0001}$, which is close to linear in the number of variables n . By observing that this marking/unmarking of variables was, in fact a specialization in the Boolean case of compressing variables' states, this Markov chain based fast sampling approach was generalized in [FHY21] to CSPs beyond the Boolean domain, specifically, to all almost *atomic* CSPs (which we will explain later), assuming an LLL condition $p\Delta^{350} \lesssim 1$. This bound was remarkably improved to $p\Delta^{7.04} \lesssim 1$ in another work of Jain, Pham and Vuong [JPV21a] through a very clever witness-tree-like information percolation analysis of the mixing time, which was also used later to support a perfect sampler through the coupling from the past (CFTP) in [HSW21] with a further improved condition $p\Delta^{5.71} \lesssim 1$.

All these fast algorithms for sampling LLL are restricted to the (almost) atomic CSPs, in which each constraint c is violated by exactly one (or very few) forbidden assignment(s) on $\text{vbl}(c)$.

Challenges for general CSP. New techniques are needed for fast sampling LLL for general CSPs. All existing fast algorithms for sampling LLL relied on some projection of the solution space to a much smaller space where the barrier of disconnectivity could be circumvented because the images of the projection might collide and were well connected. In order to efficiently simulate the random walk on the projected space and to recover a random solution from a random image, one would hope that the CSP formula were well “factorized” into small clusters most of the time because many constraints had already been satisfied for sure given the current image, which was indeed the case for fast sampling LLL for atomic CSPs [FGYZ21, FHY21, JPV21a, HSW21]. But for general non-atomic CSPs, it may no longer be the case, because now a bad event (violation of a constraint) may be highly non-elementary, and hence is no longer that easy to avoid cleanly after projection, which breaks the factorization.

It is possible that the non-atomicity of general CSPs might have imposed greater challenges to the sampling LLL than to its constructive counterpart. To see this, note that general CSPs can be simulated by atomic ones: by replacing each general constraint c having N forbidden assignments on $\text{vbl}(c)$, with N atomic constraints on the same $\text{vbl}(c)$ each forbidding one assignment. Such simulation would increase the constraint degree Δ by a factor of at most N and also decrease the violation probability p by a factor of N . For the classic LLL condition (1) where p and Δ are homogeneous, this would not change the LLL condition; but the regime for the sampling LLL captured by (2) would be significantly reduced, since there p and Δ are necessarily not homogeneous due to the lower bounds in [BGG⁺19, GGW21]. This situation seems to suggest that the non-atomicity of general CSPs might impose bigger challenges to the sampling LLL than to the existential/constructive LLL.

Indeed, prior to our work, it was not known for general CSPs with unbounded width k and degree Δ , whether the sampling problem is polynomial-time tractable under an LLL condition like (2).

1.1. Our results. In this paper, we answer the above open question positively. We give a new algorithm that departs from all prior fast samplers based on Markov chains and achieves, for the first time, a fast sampling of almost uniform satisfying solutions for general CSPs in a local lemma regime.

As in the case of algorithmic LLL [MT10, HV15], we assume an abstraction of constraint evaluations, because arbitrary constraint functions defined on a super-constant number of variables can be highly nontrivial to express and evaluate. Specifically, we assume the following evaluation oracle for checking whether a constraint is already satisfied by a partially specified assignment.

Assumption 1 (evaluation oracle). There is an *evaluation oracle* for $\Phi = (V, Q, C)$ such that given any constraint $c \in C$, any assignment $\sigma \in Q_\Lambda \triangleq \bigotimes_{v \in \Lambda} Q_v$ specified on a subset $\Lambda \subseteq \text{vbl}(c)$ of variables, the oracle answers whether c is already satisfied by σ , i.e. $c(\tau) = \text{True}$ for all $\tau \in Q_{\text{vbl}(c)}$ that $\tau_\Lambda = \sigma_\Lambda$.

For specific classes of CSPs, e.g. k -CNF or hypergraph coloring, such an oracle is easy to realize.

Assuming such an oracle for constraint evaluations, we give the following fast, almost uniform sampler for general CSPs in a local lemma regime. Recall the parameters q, k, p, Δ of a CSP formula Φ .

Theorem 1.1 (informal). *There is an algorithm such that given as input any $\varepsilon \in (0, 1)$ and any CSP formula $\Phi = (V, Q, C)$ with n variables satisfying*

$$(3) \quad q^2 \cdot k \cdot p \cdot \Delta^7 \leq \frac{1}{150e^3},$$

the algorithm terminates within $\text{poly}(q, k, \Delta) \cdot n \log(\frac{n}{\varepsilon})$ time in expectation and outputs an almost uniform sample of satisfying assignments for Φ within ε total variation distance.

The formal statement of the theorem is in Theorem 5.1 (for termination and correctness of sampling) and in Theorem 6.3 (for efficiency of sampling).

The condition in (3) becomes $p\Delta^{7+o(1)} \lesssim 1$ when $p \leq (qk)^{-\omega(1)}$, while a typical case is usually given by a much smaller $p \leq q^{-\Omega(k)}$. The previous best bound for sampling general CSP solutions was that $q^3kp\Delta^7 < c$ for a small constant c , achieved by the deterministic approximate counting based algorithm in [JPV21b] whose running time was $(n/\varepsilon)^{\text{poly}(k, \Delta, \log q)}$.

Let Z be the total number of satisfying assignments for Φ . A \hat{Z} is called an ε -approximation of Z if $(1 - \varepsilon)Z \leq \hat{Z} \leq (1 + \varepsilon)Z$. By routinely going through the non-adaptive annealing process in [FGYZ21], the approximate sampler in Theorem 1.1 can be used as a black-box to give for any $\varepsilon \in (0, 1)$ an ε -approximation of Z in time $\text{poly}(q, k, \Delta) \cdot \tilde{O}(n^2\varepsilon^{-2})$ with high probability.

1.1.1. Perfect sampler. The evaluation oracle in Assumption 1 in fact checks the sign of $\mathbb{P}[\neg c \mid \sigma]$, the probability that a constraint c is violated given a partially specified assignment σ . If further this probability can be estimated efficiently, then the sampling in Theorem 1.1 can be made perfect, where the output sample follows exactly the target distribution.

Theorem 1.2 (informal). *For the input class of CSPs, if there is such an FPTAS for violation probability:*

- *for any constraint $c \in C$, any assignment $\sigma \in Q_\Lambda$ specified on a subset $\Lambda \subseteq \text{vbl}(c)$, and $0 < \varepsilon < 1$, an ε -approximation of $\mathbb{P}[\neg c \mid \sigma]$ is returned deterministically within $\text{poly}(q, k, 1/\varepsilon)$ time,*

then the sampling algorithm in Theorem 1.1 returns a perfect sample of uniform satisfying assignment within $\text{poly}(q, k, \Delta) \cdot n$ time in expectation under the same condition (3).

The formal statement of the theorem is in Theorem 5.1 (for termination and correctness of sampling) and in Theorem 6.1 (for efficiency of sampling). In fact, we prove this perfect sampler first, and then realize the FPTAS assumed in Theorem 1.2 using Monte Carlo experiments, which introduces a bounded bias to the sampling and gives us the approximate sampler claimed in Theorem 1.1.

For concrete classes of CSPs defined by simple local constraints, it is no surprise to see that the probability $\mathbb{P}[\neg c \mid \sigma]$ almost always has an easy-to-compute closed-form expression, in which case we have a perfect sampler without assuming the oracles in Assumption 1 and in Theorem 1.2.

The followings are two examples of non-atomic CSPs which admit linear-time perfect samplers.

Example 1.3 (δ -robust k -SAT). *The n variables are Boolean, each clause contains exactly k literals, and a clause is satisfied if and only if at least δk of its literals have the outcome True.*

- *For δ -robust k -SAT with variable degree d (each variable appears in at most d clauses) satisfying*

$$0 < \delta < \frac{1}{2}, \quad k \geq \frac{32 \ln k + 28 \ln d + 40}{(1 - 2\delta)^2},$$

a perfect sample of uniform satisfying solutions is returned within expected time $\text{poly}(k, d) \cdot n$.

Example 1.4 (δ -robust hypergraphs q -coloring). *Each vertex is colored with one of the q colors, each hyperedge is k -uniform and is satisfied if and only if there are no $(1 - \delta)k$ vertices with the same color.*

- *For k -uniform hypergraphs on n vertices with maximum vertex degree d satisfying*

$$(1 - \delta)k \geq 15, \quad q \geq \frac{7d^{\frac{7}{(1-\delta)k-3}} \cdot (6.1)^{\frac{1}{(1-\delta)}}}{(1 - \delta)^{1.25}},$$

a perfect sample of uniform satisfying coloring is returned within expected time $\text{poly}(q, k, d) \cdot n$.

1.1.2. *Marginal sampler.* The core component of our sampling algorithm is a *marginal sampler* for drawing from marginal distributions. Let $\mu = \mu_\Phi$ denote the uniform distribution over all satisfying assignments for Φ , and for each $v \in V$, let μ_v denote the marginal distribution at v induced by μ .

Theorem 1.5 (informal). *There is an algorithm such that given as input any $\varepsilon \in (0, 1)$, any CSP formula $\Phi = (V, Q, C)$ satisfying (3), and any $v \in V$, the algorithm returns a random value $x \in Q_v$ distributed approximately as μ_v within total variation distance ε , within $\text{poly}(q, k, \Delta, \log(1/\varepsilon))$ time in expectation.*

This marginal sampler is also perfect under the same assumption as in Theorem 1.2. Another byproduct of this marginal sampler is the following algorithm for probabilistic inference.

Theorem 1.6 (informal). *There is an algorithm such that given as input any $\varepsilon, \delta \in (0, 1)$, any CSP formula $\Phi = (V, Q, C)$ satisfying (3), and any $v \in V$, the algorithm returns for every $x \in Q_v$ an ε -approximation of the marginal probability $\mu_v(x)$ within $\text{poly}(q, k, \Delta, 1/\varepsilon, \log(1/\delta))$ time with probability at least $1 - \delta$.*

The above two theorems are formally restated and proved in Theorem 6.36.

By a self-reduction, the sampling and inference algorithms in Theorems 1.5 and 1.6 remain to hold for the marginal distributions μ_v^σ conditional on a feasible partially specified assignment σ , as long as the LLL condition (3) is satisfied by the new instance Φ^σ obtained from pinning σ onto Φ .

Both the above algorithms for marginal sampling and probabilistic inference are **local algorithms whose costs are independent of n** . Previously, in order to simulate or estimate the marginal distribution of a variable, it was often necessary to generate a full assignment on all n variables, or at least pay no less than that. One might have asked the following natural question:

Can these locally defined sampling or inference problems be solved at a local cost?

However, decades have passed, and only recently has such a novel local algorithm been discovered for marginal distributions in infinite spin systems [AJ21], which is also our main source of inspiration.

1.2. **Technique overview.** As we have explained before, non-atomicity of constraints causes a barrier for the current Markov chain based algorithms [FGYZ21, FHY21, JPV21a, HSW21, FGW22]. There is another family of sampling algorithms, which we call “resampling based” algorithms [FH00, GJL19, FVY19, Jer21, FGY22]. These algorithms use resampling of variables to fix the assignment until it follows the right distribution, morally like the Moser-Tardos algorithm, and they are not as affected by disconnectivity of solution space as Markov chains, but here a principle to ensure the correct sampling is to resample the variables that the algorithm has observed and conditioned on, which also causes trouble on non-atomic constraints, because to ensure such constraints are satisfied, the algorithm has to observe too many variables, whose resampling would cancel the progress of the algorithm.

We adopt a new idea of sampling, which we call the *recursive marginal sampler*. It is somehow closer to the resampling based algorithms than to the Markov chains, but thanks to its recursive nature, the algorithm avoids excessive resampling. This algorithm is inspired by a recent novel algorithm of Anand and Jerrum [AJ21] for perfectly sampling in infinite spin systems, where a core component is such a marginal sampler that can draw a spin according to its marginal distribution.

Now let us consider the uniform distribution μ over all satisfying assignments of a CSP formula $\Phi = (V, Q, C)$, and its marginal distribution μ_v at a variable $v \in V$, say over domain $Q_v = [q]$. To sample from this μ_v over $[q]$, an idea is to exploit the so-called “local uniformity” property [HSS11], which basically says that μ_v should not be far from a uniform distribution over $[q]$ in total variation distance when Φ satisfies some local lemma condition. Therefore, a uniform sample from $[q]$ already gives a coarse sample of μ_v . It remains to boost such a coarse sampler to a sampler with arbitrary precision.

By the local uniformity, there exists a $\theta < \frac{1}{q}$ close enough to $\frac{1}{q}$, such that

$$(4) \quad \forall x \in [q], \quad \mu_v(x) \geq \theta.$$

The marginal distribution μ_v can then be divided as

$$q\theta \cdot \mathcal{U} + (1 - q\theta) \cdot \mathcal{D},$$

where \mathcal{U} is the uniform distribution over $[q]$ and \mathcal{D} gives a distribution of “overflow” mass such that:

$$\forall x \in [q], \quad \mathcal{D}(x) = \frac{\mu_v(x) - \theta}{1 - q\theta}.$$

Sampling from μ_v then can follow this strategy: with probability $q\theta$, the algorithm falls into the “zone of local uniformity” and returns a uniform sample from \mathcal{U} ; and with probability $1 - q\theta$, the algorithm falls into the “zone of indecision” and has to draw a sample from this overflow distribution \mathcal{D} , which can be done by constructing a Bernoulli factory that accesses μ_v as an oracle. But wait, if we had such an oracle for μ_v in the first place, why would sampling from μ_v even be a problem?

The above “chicken or egg” paradox is somehow resolved by a simple observation: if enough many other variables had already been sampled correctly, say with outcome X , then assuming a strong enough LLL condition, there is a good chance that the resulting formula Φ^X was “factorized” into small clusters, from where a standard rejection sampling on Φ^X would be efficient for sampling from μ_v^X , and overall from μ_v . Therefore, the sampling strategy is now corrected as: after falling into the “zone of indecision” and before trying to draw from the overflow distribution \mathcal{D} , the algorithm picks another variable u whose successful sampling might help factorize Φ , and recursively apply the marginal sampler at u to draw from u ’s current marginal distribution first. The only loose end now is that the LLL condition is not self-reducible, meaning it is not invariant under arbitrary pinning. We adopt the idea of “freezing” constraints used in [JPV21b] to guide the algorithm to pick variables for sampling. The LLL condition is replaced by a more refined invariant condition that guarantees for each variable picked for sampling, the same local uniformity as in (4) to persist throughout the algorithm, and also guarantees a good chance of factorization while there are no other variables to pick.

To bound the fast convergence of the recursive sampler, in [AJ21] the strategy was to show that the branching process given by the recursion tree always has decaying offspring number in expectation given the worst-case boundary condition, which is not true here. Instead, we apply a more average-case style analysis. Interestingly, some of our analysis and the LLL condition resemble those in [JPV21b] for a deterministic approximate counting algorithm with time complexity $n^{\text{poly}(k, \Delta, \log q)}$.

2. NOTATIONS FOR CSP

We recall the definition of CSP formula $\Phi = (V, Q, C)$ in Section 1. We use $\Omega = \Omega_\Phi$ to denote the set of all satisfying assignments of Φ , and use $\mu = \mu_\Phi$ to denote the uniform distribution over Ω . Recall that \mathbb{P} denotes the law for the uniform product distribution over Q . For $C \subseteq \mathcal{C}$, denote $\text{vbl}(C) \triangleq \bigcup_{c \in C} \text{vbl}(c)$; and for $\Lambda \subseteq V$, denote $Q_\Lambda \triangleq \bigotimes_{v \in \Lambda} Q_v$. We introduce a notation for partial assignments.

Definition 2.1 (partial assignment). Given a CSP formula $\Phi = (V, Q, C)$, define:

$$Q^* \triangleq \bigotimes_{v \in V} (Q_v \cup \{\star, \star\}),$$

where \star and \star are two special symbols not in any Q_v . Each $\sigma \in Q^*$ is called a *partial assignments*.

In a partial assignment $\sigma \in Q^*$, each variable $v \in V$ is classified as follows:

- $\sigma(v) \in Q_v$ means that v is *accessed* by the algorithm and *assigned* with the value $\sigma(v) \in Q_v$;
- $\sigma(v) = \star$ means that v is just *accessed* by the algorithm but *unassigned* yet with a value in Q_v ;
- $\sigma(v) = \star$ means that v is *unaccessed* by the algorithm and hence *unassigned* with any value.

Furthermore, we use $\Lambda(\sigma) \subseteq V$ and $\Lambda^+(\sigma) \subseteq V$ to respectively denote the sets of assigned and accessed variables in a partial assignment $\sigma \in Q^*$, that is:

$$(5) \quad \Lambda(\sigma) \triangleq \{v \in V \mid \sigma(v) \in Q_v\} \quad \text{and} \quad \Lambda^+(\sigma) \triangleq \{v \in V \mid \sigma(v) \neq \star\}.$$

Given any partial assignment $\sigma \in Q^*$ and variable $v \in V$, we further denote by $\sigma_{v \leftarrow x}$ the partial assignment obtained from modifying σ by replacing $\sigma(v)$ with $x \in Q_v \cup \{\star, \star\}$.

A partial assignment $\tau \in Q$ is said to *extend* a partial assignment $\sigma \in Q^*$ if $\Lambda(\sigma) \subseteq \Lambda(\tau)$, $\Lambda^+(\sigma) \subseteq \Lambda^+(\tau)$, and σ, τ agree with each other over all variables in $\Lambda(\sigma)$. A partial assignment $\sigma \in Q^*$ is said to satisfy a constraint $c \in C$ if c is satisfied by all full assignments $\tau \in Q$ that extend σ . A partial assignment $\sigma \in Q^*$ is called *feasible* if there is a satisfying assignment $\tau \in \Omega$ that extends σ .

Given any feasible $\sigma \in \mathcal{Q}^*$ and any $S \subseteq V$, we use μ_S^σ to denote the marginal distribution induced by μ on S conditional on σ . For each $\tau \in \mathcal{Q}_S$, we have $\mu_S^\sigma(\tau) = \Pr_{X \sim \mu} [X(S) = \tau \mid \forall v \in \Lambda(\sigma), X(v) = \sigma(v)]$. We further write $\mu_v^\sigma = \mu_{\{v\}}^\sigma$. Similar notation is used for the law \mathbb{P} for the uniform product distribution over \mathcal{Q} . For $\sigma \in \mathcal{Q}^*$ and any event $A \subseteq \mathcal{Q}$, we have $\mathbb{P}[A \mid \sigma] = \Pr_{X \in \mathcal{Q}} [X \in A \mid \forall v \in \Lambda(\sigma), X(v) = \sigma(v)]$.

3. THE SAMPLING ALGORITHM

We give our main algorithm for sampling almost uniform satisfying assignments for a CSP formula. Our presentation uses notations defined in Section 2.

3.1. The main sampling algorithm. Our main sampling algorithm takes as input a CSP formula $\Phi = (V, \mathcal{Q}, C)$ with domain size $q = q_\Phi$, width $k = k_\Phi$, constraint degree $\Delta = \Delta_\Phi$, and violation probability $p = p_\Phi$, where the meaning of these parameters are as defined in Section 1.

We suppose that the $n = |V|$ variables are enumerated as $V = \{v_1, v_2, \dots, v_n\}$ in an arbitrary order. The CSP formula $\Phi = (V, \mathcal{Q}, C)$ is presented to the algorithm by the evaluation oracle in Assumption 1. Also assume that given any $c \in C$ (or $v \in V$), the $\text{vbl}(c)$ (or $\{c \in C \mid v \in \text{vbl}(c)\}$) can be retrieved.

The main algorithm (Algorithm 1) is the same as the main sampling frameworks in [JPV21b, GLLZ19]. A partial assignment $X \in \mathcal{Q}$ is maintained, initially as the empty assignment $X = \star^V$.

- (1) In the 1st phase, at each step it adaptively picks (in a predetermined order) a variable v that has enough “freedom” because it is not involved in any easy-to-violate constraint given the current X , and replaces $X(v)$ with a random value drawn by a subroutine `MarginSample` according to the correct marginal distribution μ_v^X .
- (2) When no such variable with enough freedom remains, the formula is supposed to be “factorized” enough into small clusters and the algorithm enters the 2nd phase, from where the partial assignment constructed in the 1st phase is completed to a uniform random satisfying assignment by a standard `RejectionSampling` subroutine.

A key threshold p' for the violation probability is fixed as below:

$$(6) \quad p' = \left(18e^2 q^2 k \Delta^4\right)^{-1},$$

which satisfies $p' > p = p_\Phi$, assuming the LLL condition in (3).

For the ease of exposition, we assume an oracle for approximately deciding whether a constraint becomes too easy to violate given the current partial assignment.

Assumption 2. There is an oracle such that given any partial assignment $\sigma \in \mathcal{Q}^*$ and any constraint $c \in C$, the oracle distinguishes between the two cases: $\mathbb{P}[\neg c \mid \sigma] > p'$ and $\mathbb{P}[\neg c \mid \sigma] < 0.99p'$, and answers arbitrarily and consistently if otherwise, which means that the answer to the undefined case $\mathbb{P}[\neg c \mid \sigma] \in [0.99p', p']$ can be either “yes” or “no” but remains the same for the same $\sigma_{\text{vbl}(c)}$.

Such an oracle is clearly implied by the FPTAS for violation probability assumed in Theorem 1.2 and will be explicitly realized later in Section 6. For now, with respect to such an oracle, the classes of easy-to-violate constraints and their involved variables are defined as follows.

Definition 3.1 (frozen and fixed). Assume Assumption 2. Let $\sigma \in \mathcal{Q}^*$ be a partial assignment.

- A constraint $c \in C$ is called σ -frozen if it is reported $\mathbb{P}[\neg c \mid \sigma] > p'$ by the oracle in Assumption 2. Denote by C_{frozen}^σ the set of all σ -frozen constraints:

$$C_{\text{frozen}}^\sigma \triangleq \{c \in C \mid c \text{ is reported by the oracle to satisfy } \mathbb{P}[\neg c \mid \sigma] > p'\}.$$

- A variable $v \in V$ is called σ -fixed if v is accessed in σ or is involved in some σ -frozen constraint. Denote by V_{fix}^σ the set of all σ -fixed variables:

$$V_{\text{fix}}^\sigma \triangleq \Lambda^+(\sigma) \cup \bigcup_{c \in C_{\text{frozen}}^\sigma} \text{vbl}(c).$$

Similar ideas of freezing appeared in previous works on sampling and algorithmic LLL [JPV21b, Bec91]

Remark 3.2 (one-sided error for frozen/fixed decision). By the property of the oracle in Assumption 2, any constraint $c \in C$ with $\mathbb{P}[\neg c \mid \sigma] > p'$ must be in C_{frozen}^σ , and any variable $v \in V$ involved in such a constraint must be in V_{fix}^σ ; conversely, any σ -frozen constraint $c \in C_{\text{frozen}}^\sigma$ must have $\mathbb{P}[\neg c \mid \sigma] \geq 0.99p'$ and any unaccessed σ -fixed variable $v \in V_{\text{fix}}^\sigma$ must be involved in at least one of such constraints.

Algorithm 1: The sampling algorithm

Input: a CSP formula $\Phi = (V, Q, C)$;
Output: a uniform random satisfying assignment $X \in \Omega_\Phi$;

```

1  $X \leftarrow \star^V$ ;
2 for  $i = 1$  to  $n$  do
3   if  $v_i$  is not  $X$ -fixed then
4      $X(v_i) \leftarrow \text{MarginSample}(\Phi, X, v_i)$ ;
5  $X_{V \setminus \Lambda(X)} \leftarrow \text{RejectionSampling}(\Phi, X, V \setminus \Lambda(X))$ ;
6 return  $X$ ;
```

The following invariant is satisfied in the **for** loop in Algorithm 1 (formally proved in Lemma 5.4). The correctness of the MarginSample subroutine is guaranteed by this invariant.

Condition 3.3 (invariant for MarginSample). *The followings hold for the input tuple (Φ, σ, v) :*

- $\Phi = (V, Q, C)$ is a CSP formula, $\sigma \in Q^*$ is a feasible partial assignment, and $v \in V$ is a variable;
- v is not σ -fixed and $\sigma(v) = \star$, and for all $u \in V$, $\sigma(u) \in Q_u \cup \{\star\}$;
- $\mathbb{P}[\neg c \mid \sigma] \leq p'q$ for all $c \in C$.

The correctness of Algorithm 1 follows from the correctness of MarginSample and RejectionSampling for sampling from the correct marginal distributions, which is formally proved in Theorem 5.1.

In fact, the sampling in Algorithm 1 is *perfect*. It will only become approximate after the oracle in Assumption 2 realized by a Monte Carlo program that may bias the sampling.

3.2. The rejection sampling. We first introduce the RejectionSampling, which is a standard procedure. Our rejection sampling takes advantages of simplification and decomposition of a CSP formula.

A simplification of $\Phi = (V, Q, C)$ under partial assignment $\sigma \in Q^*$, denoted by $\Phi^\sigma = (V^\sigma, Q^\sigma, C^\sigma)$, is a new CSP formula such that $V^\sigma = V \setminus \Lambda(\sigma)$ and $Q^\sigma = Q_{V \setminus \Lambda(\sigma)}$, and the C^σ is obtained from C by:

- (1) removing all the constraints that have already been satisfied by σ ;
- (2) for the remaining constraints, replacing the variables $v \in \Lambda(\sigma)$ with their values $\sigma(v)$.

It is easy to see that $\mu_{\Phi^\sigma} = \mu_{V \setminus \Lambda(\sigma)}^\sigma$ for the uniform distribution μ_{Φ^σ} over satisfying assignments of Φ^σ .

A CSP formula $\Phi = (V, Q, C)$ can be naturally represented as a (multi-)hypergraph H_Φ , where each variable $v \in V$ corresponds to a vertex in H_Φ and each constraint $c \in C$ corresponds to a hyperedge $\text{vbl}(c)$ in H_Φ . We slightly abuse the notation and write $H_\Phi = (V, C)$.

Let $H_i = (V_i, C_i)$ for $1 \leq i \leq K$ denote all $K \geq 1$ connected components in H_Φ , and $\Phi_i = (V_i, Q_{V_i}, C_i)$ their formulas. Obviously $\Phi = \Phi_1 \wedge \Phi_2 \wedge \dots \wedge \Phi_K$ with disjoint Φ_i , and μ_Φ is the product of all μ_{Φ_i} . Also μ_S on a subset $S \subseteq V$ is determined only by those components with V_i intersecting S .

Our rejection sampling algorithm for drawing from a marginal distribution μ_S^σ is given in Algorithm 2. The correctness of this algorithm is folklore. We state without proof.

Theorem 3.4. *On any input (Φ, σ, S) as specified in Algorithm 2, RejectionSampling terminates with probability 1, and upon termination it returns an assignment $X_S \in Q_S$ distributed as μ_S^σ .*

3.3. The marginal sampler. We now introduce the the core part of our sampling algorithm, the MarginSample subroutine. This procedure is a “marginal sampler”: it can draw a random value for a variable $v \in V$ according to its marginal distribution μ_v^σ . Our marginal sampling algorithm is inspired by a recent novel sampling algorithm of Anand and Jerrum for infinite spin systems [AJ21].

by accessing an oracle for drawing from μ_v^σ , can be resolved using the existing approaches of *Bernoulli factory* [NP05, Hub16, DHKN17].

This sounds silly because if such oracle for μ_v^σ were efficient we would have been using it to output a sample for μ_v^σ in the first place, which is exactly the reason why we ended up trying to draw from \mathcal{D} .

Nevertheless, such Bernoulli factory for sampling from \mathcal{D} may serve as the basis of a recursion, where sufficiently many variables with enough “freedom” would have been sampled successfully in their zones of local uniformity during the recursion, and hence the remaining CSP formula would have been “factorized” into small connected components, in which case an oracle for μ_v^σ would be efficient to realize by the RejectionSampling($\Phi, \sigma, \{v\}$), and the Bernoulli factory for \mathcal{D} could apply.

We define a class of variables that are candidates for sampling with priority in the recursion.

Definition 3.6 (\star -influenced variables). Let $\sigma \in \mathcal{Q}^*$ be a partial assignment. Let $H^\sigma = H_{\Phi^\sigma} = (V^\sigma, C^\sigma)$ be the hypergraph for simplification Φ^σ . Let H_{fix}^σ be the sub-hypergraph of H^σ induced by $V^\sigma \cap V_{\text{fix}}^\sigma$.

- Let $V_\star^\sigma \subseteq V^\sigma \cap V_{\text{fix}}^\sigma$ be the set of vertices belong to the connected components in H_{fix}^σ that contain any v with $\sigma(v) = \star$.
- Let $V_{\star\text{-inf}}^\sigma \triangleq \{u \in V^\sigma \setminus V_\star^\sigma \mid \exists c \in C^\sigma, v \in V_\star^\sigma : u, v \in \text{vbl}(c)\}$ be the vertex boundary of V_\star^σ in H^σ .
- Let $C_{\star\text{-con}}^\sigma$ be the set of constraints $c \in C$ that intersect V_\star^σ .
- Define NextVar(σ) by

$$(9) \quad \text{NextVar}(\sigma) \triangleq \begin{cases} v_i \in V_{\star\text{-inf}}^\sigma \text{ with smallest } i & \text{if } V_{\star\text{-inf}}^\sigma \neq \emptyset, \\ \perp & \text{otherwise.} \end{cases}$$

Remark 3.7. The $C_{\star\text{-con}}^\sigma$ defined above is not used here, but is important in the analysis.

Same as in Definition 3.1, the V_{fix}^σ is defined with respect to the oracle in Assumption 2. In Section 6.5.1, a dynamic data structure is given to efficiently compute NextVar(σ).

With this construction of NextVar(\cdot), the MarginOverflow subroutine is described in Algorithm 4.

Algorithm 4: MarginOverflow(Φ, σ, v)

Input: a CSP formula $\Phi = (V, C)$, a feasible partial assignment $\sigma \in \mathcal{Q}^*$, and a variable $v \in V$;

Output: a random $x \in Q_v$ distributed as $\mathcal{D} \triangleq \frac{1}{(1-q_v \cdot \theta_v)}(\mu_v^\sigma - \theta_v)$;

```

1  $u \leftarrow \text{NextVar}(\sigma)$  where NextVar( $\sigma$ ) is defined as in (9);
2 if  $u \neq \perp$  then
3   choose  $r \in [0, 1)$  uniformly at random;
4   if  $r < q_u \cdot \theta_u$  then //  $r$  falls into the zone of local uniformity
5      $\sigma(u) \leftarrow$  the  $\lceil r/\theta_u \rceil$ -th value in  $Q_u$ ;
6   else //  $r$  falls into the zone of indecision
7      $\sigma(u) \leftarrow \text{MarginOverflow}(\Phi, \sigma_{u \leftarrow \star}, u)$ ;
8   /* Line 4 to Line 7 together draw  $\sigma(u)$  according to  $\mu_u^{\sigma_{u \leftarrow \star}}$  */
9   return MarginOverflow( $\Phi, \sigma, v$ );
9 else // All non- $\sigma$ -fixed variables are d/c. from  $v$  and ancestors.
10  sample a random  $x \in Q_v$  according to  $\mathcal{D} \triangleq \frac{1}{(1-q_v \cdot \theta_v)}(\mu_v^\sigma - \theta_v)$  using the Bernoulli factory in
    Appendix A that accesses RejectionSampling( $\Phi, \sigma, \{v\}$ ) as an oracle;
11 return  $x$ ;
```

Basically, a variable u is a good candidate for sampling if it currently has enough “freedom” (since u is not σ -fixed) and can “influence” the variables that we are trying to sample in the recursion (which are marked by \star) through a chain of constraints in the simplification of Φ under the current σ . Such variables are enumerated by NextVar(σ).

The idea of Algorithm 4 is simple. In order to draw from the overflow distribution \mathcal{D} for a variable $v \in V$: if there is another candidate variable $u = \text{NextVar}(\sigma)$ that still has enough freedom so its sampling might be easy, and also is relevant to the sampling at v or its ancestors, we try to sample u ’s

marginal value first (hopefully within its zone of local uniformity and compensated by a recursive call for drawing from its margin overflow); and if there is no such candidate variable to sample first, we finally draw from \mathcal{D} using the Bernoulli factory.

The following invariant is satisfied by the MarginOverflow subroutine called within the MarginSample subroutine and the MarginOverflow itself (formally proved in Lemma 5.4).

Condition 3.8 (invariant for MarginOverflow). *The followings hold for the input tuple (Φ, σ, v) :*

- $\Phi = (V, Q, C)$ is a CSP formula, $\sigma \in Q^*$ is a feasible partial assignment, and $v \in V$ is a variable;
- $\sigma(v) = \star$;
- $\mathbb{P}[\neg c \mid \sigma] \leq p'q$ for all $c \in C$.

The following marginal lower bound follows from the “local uniformity” property (Corollary 4.3) in the same way as in Proposition 3.5 and is formally proved in Section 4.

Proposition 3.9. *Assuming Condition 3.8 for the input (Φ, σ, v) , it holds that $\min_{x \in Q_v} \mu_v^\sigma(x) \geq \theta_v + \zeta$ and for $u = \text{NextVar}(\sigma)$, if $u \neq \perp$ then it also holds that $\min_{x \in Q_u} \mu_u^\sigma(x) \geq \theta_u + \zeta$.*

The Bernoulli factory used in Algorithm 4 is achieved by a combination of existing constructions (to be specific, the Bernoulli factory for subtraction in [NP05], composed with the linear Bernoulli factory in [Hub16] and the Bernoulli race in [DHKN17]), given access to an oracle for drawing from the marginal distribution μ_v^σ , which is realized by RejectionSampling($\Phi, \sigma, \{v\}$) in Algorithm 2. This is formally stated by the following lemma.

Lemma 3.10 (correctness of Bernoulli factory). *Assuming Condition 3.8 for the input (Φ, σ, v) , there is a Bernoulli factory accessing RejectionSampling($\Phi, \sigma, \{v\}$) as an oracle that terminates with probability 1, and upon termination it returns a random $x \in Q_v$ distributed as the \mathcal{D} defined in (8).*

The construction of the Bernoulli factory stated in above lemma is somehow standard, and is deferred to Appendix A, where Lemma 3.10 is proved and the efficiency of the Bernoulli factory is also analyzed.

4. PRELIMINARY ON LOVÁSZ LOCAL LEMMA

The following is the asymmetric Lovász Local Lemma stated in the context of CSP.

Theorem 4.1 (Erdős and Lovász [EL75]). *Given a CSP formula $\Phi = (V, Q, C)$, if the following holds*

$$(10) \quad \exists x \in (0, 1)^C \quad \text{s.t.} \quad \forall c \in C : \quad \mathbb{P}[\neg c] \leq x(c) \quad \prod_{\substack{c' \in C \\ \text{vbl}(c) \cap \text{vbl}(c') \neq \emptyset}} (1 - x(c')),$$

then

$$\mathbb{P} \left[\bigwedge_{c \in C} c \right] \geq \prod_{c \in C} (1 - x(c)) > 0,$$

When the condition (10) is satisfied, the probability of any event in the uniform distribution μ over all satisfying assignments can be well approximated by the probability of the event in the product distribution. This was observed in [HSS11]:

Theorem 4.2 (Haeupler, Saha, and Srinivasan [HSS11]). *Given a CSP formula $\Phi = (V, Q, C)$, if (10) holds, then for any event A that is determined by the assignment on a subset of variables $\text{vbl}(A) \subseteq V$,*

$$\Pr_{\mu} [A] = \mathbb{P} \left[A \mid \bigwedge_{c \in C} c \right] \leq \mathbb{P}[A] \prod_{\substack{c \in C \\ \text{vbl}(c) \cap \text{vbl}(A) \neq \emptyset}} (1 - x(c))^{-1},$$

where μ denotes the uniform distribution over all satisfying assignments of Φ and \mathbb{P} denotes the law of the uniform product distribution over Q .

The following “local uniformity” property is a straightforward corollary to Theorem 4.2 by setting $x(c) = \epsilon p$ for every $c \in C$ (and the lower bound is calculated by $\mu_v(x) = 1 - \sum_{y \in Q_v \setminus \{x\}} \mu_v(y)$).

Corollary 4.3 (local uniformity). *Given a CSP formula $\Phi = (V, Q, C)$, if $ep\Delta < 1$, then for any variable $v \in V$ and any value $x \in Q_v$, it holds that*

$$\frac{1}{q_v} - \left((1 - ep)^{-\Delta} - 1 \right) \leq \mu_v(x) \leq \frac{1}{q_v} + \left((1 - ep)^{-\Delta} - 1 \right).$$

The following corollary implied by the “local uniformity” property simultaneously proves Proposition 3.5 and Proposition 3.9. Recall p' defined in (6) and θ_v, ζ, η defined in (7).

Corollary 4.4. *For any CSP formula $\Phi = (V, Q, C)$ and any partial assignment $\sigma \in Q^*$, if*

$$\forall c \in C, \quad \mathbb{P}[\neg c \mid \sigma] \leq p'q,$$

then σ is feasible, and for any $v \in V \setminus \Lambda(\sigma)$, and any $x \in Q_v$

$$\theta_v + \zeta \leq \mu_v^\sigma(x) \leq \theta_v + 2\eta + \zeta.$$

Proof. Let $\Phi^\sigma = (V^\sigma, Q^\sigma, C^\sigma)$ be the simplification of Φ over σ , where $V^\sigma = V \setminus \Lambda(\sigma)$. We have

$$\forall c \in C, \quad \mathbb{P}_{\Phi^\sigma}[\neg c] = \mathbb{P}_\Phi[\neg c \mid \sigma] \leq p'q,$$

which means the simplified instance Φ^σ has violation probability $p_{\Phi^\sigma} \leq p'q$. By our choice of p' in (6), we still have $ep_{\Phi^\sigma}\Delta_{\Phi^\sigma} < 1$ where $\Delta_{\Phi^\sigma} \leq \Delta$ is the constraint degree of simplified instance Φ^σ . Then by Theorem 4.1, Φ^σ is satisfiable, i.e. σ is feasible.

Note that the marginal distribution at v induced by the μ_{Φ^σ} over satisfying assignments of Φ^σ is precisely μ_v^σ . By Corollary 4.3, for any $v \in V^\sigma = V \setminus \Lambda(\sigma)$, and any $x \in Q_v$,

$$\theta_v + \zeta = \frac{1}{q_v} - \eta = \frac{1}{q_v} - \left((1 - ep'q)^{-\Delta} - 1 \right) \leq \mu_v^\sigma(x) \leq \frac{1}{q_v} + \left((1 - ep'q)^{-\Delta} - 1 \right) = \frac{1}{q_v} + \eta = \theta_v + 2\eta + \zeta$$

□

5. CORRECTNESS OF SAMPLING

In this section, we prove the correctness of Algorithm 1. All theorems in this section assume the setting of parameters in (6) and (7), and the oracles in Assumption 1 and Assumption 2.

We show that our main sampling algorithm Algorithm 1 is correct.

Theorem 5.1. *On any input CSP formula $\Phi = (V, Q, C)$ satisfying (3), Algorithm 1 terminates with probability 1, and returns a uniform random satisfying assignment of Φ upon termination.*

Remark 5.2 (perfectness of sampling). Note that the sampling in above theorem is *perfect*: Algorithm 1 returns a sample that is distributed exactly as the uniform distribution μ over all satisfying assignments of Φ . Later in Section 6, the oracle assumed in Assumption 2 will be realized by a Monte Carlo routine, which will further generalize the sampling algorithm to assume nothing beyond an evaluation oracle, in a price of a bounded bias introduced to the sampling.

Remark 5.3 (a weaker LLL condition). The LLL condition (3) is assumed mainly to guarantee the efficiency of the algorithm. Theorem 5.1 in fact holds under a much weaker LLL condition:

$$2e \cdot q^2 \cdot p \cdot \Delta < 1.$$

Under this condition, there exists such choices of parameters p' and ζ that satisfy $p < p' < \frac{1}{2eq^2\Delta}$ and $0 < \zeta < \frac{1}{q} - (1 - ep'q)^{-\Delta} + 1$. For any such choice of parameters, the same analysis persists and Algorithm 1 is as correct as claimed in Theorem 5.1.

The following lemma guarantees that the invariants in Condition 3.3 and Condition 3.8 are satisfied respectively by the inputs to Algorithm 3 and Algorithm 4.

Lemma 5.4. *During the execution of Algorithm 1 on a CSP formula $\Phi = (V, Q, C)$ satisfying (3):*

- (1) *whenever $\text{MarginSample}(\Phi, X, v)$ is called, Condition 3.3 is satisfied by its input (Φ, X, v) ;*
- (2) *whenever $\text{MarginOverflow}(\Phi, \sigma, v)$ is called, Condition 3.8 is satisfied by its input (Φ, σ, v) .*

Before proving this lemma, we show that these invariants can already imply the correctness of `MarginSample`, which is critical for the correctness of the main sampling algorithm (Algorithm 1), because the correctness of `RejectionSampling` is standard (Theorem 3.4).

Theorem 5.5. *The followings hold for Algorithm 3 and Algorithm 4:*

- (1) *Assuming Condition 3.3, `MarginSample`(Φ, σ, v) terminates with probability 1, and it returns a random value $x \in Q_v$ distributed as μ_v^σ upon termination.*
- (2) *Assuming Condition 3.8, `MarginOverflow`(Φ, σ, v) terminates with probability 1, and upon termination it returns a random value $x \in Q_v$ distributed as the $\mathcal{D} \triangleq \frac{\mu_v^\sigma - \theta_v}{1 - q_v \cdot \theta_v}$ defined in (8).*

Proof. We verify the correctness of `MarginOverflow` by a structural induction. Then the correctness of `MarginSample` follows straightforwardly.

Suppose that `MarginOverflow` is run on input (Φ, σ, v) satisfying Condition 3.8.

The induction basis is when `NextVar` (σ) = \perp , in which case no further recursive calls to the `MarginOverflow` are incurred. In this case, due to the correctness of the Bernoulli factory stated in Lemma 3.10 under Condition 3.8, `MarginOverflow`(Φ, σ, v) terminates with probability 1 and returns a random value $x \in Q_v$ distributed as \mathcal{D} .

For the induction step, we assume that `NextVar` (σ) = $u \in V$. By the induction hypothesis, the recursive calls to `MarginOverflow` at Line 7 and Line 8 in Algorithm 4 terminate with probability 1. All other computations are finite. By induction, `MarginOverflow`(Φ, σ, v) terminates with probability 1.

We then verify the correctness of sampling. Let W denote the value of $\sigma(u)$ generated in Lines 3-7 of in Algorithm 4. It is easy to verify that for every $a \in Q_u$,

$$(11) \quad \begin{aligned} \Pr[W = a] &= \Pr[r < q_u \cdot \theta_u] \cdot \frac{1}{q_u} + \Pr[r \geq q_u \cdot \theta_u] \cdot \Pr[\text{MarginOverflow}(\Phi, \sigma_{u \leftarrow \star}, u) = a] \\ &= \theta_u + \mu_u^\sigma(a) - \theta_u = \mu_u^\sigma(a), \end{aligned}$$

where the second equality is due to the induction hypothesis (I.H.). Thus, for every $a \in Q_u$,

$$\begin{aligned} \Pr[\text{MarginOverflow}(\Phi, \sigma, v) = a] &= \sum_{b \in Q_u} (\Pr[W = b] \cdot \Pr[\text{MarginOverflow}(\Phi, \sigma_{u \leftarrow b}, v) = a]) \\ (\text{by (11) and I.H.}) \quad &= \sum_{b \in Q_u} \left(\mu_u^\sigma(b) \cdot \frac{\mu_v^{\sigma_{u \leftarrow b}}(a) - \theta_v}{1 - q_v \cdot \theta_v} \right) \\ &= \frac{\mu_v^\sigma(a) - \theta_v}{1 - q_v \cdot \theta_v}, \end{aligned}$$

which means that the value returned by `MarginOverflow`(Φ, σ, v) is distributed as \mathcal{D} . This finishes the induction and proves the correctness of `MarginOverflow` assuming Condition 3.8.

It is then straightforward to verify the correctness of `MarginSample` under Condition 3.3. Let (Φ, σ, v) be an arbitrary input to `MarginSample` satisfying Condition 3.3. It is easy to verify that $(\Phi, \sigma_{v \leftarrow \star}, v)$ satisfies Condition 3.8, and hence the correctness of `MarginOverflow` can apply. Therefore, `MarginOverflow`(Φ, σ, v) terminates with probability 1 and for every $a \in Q_v$,

$$\begin{aligned} &\Pr[\text{MarginSample}(\Phi, \sigma, v) = a] \\ &= \Pr[r < q_v \cdot \theta_v] \cdot \frac{1}{q_v} + \Pr[r \geq q_v \cdot \theta_v] \cdot \Pr[\text{MarginOverflow}(\Phi, \sigma_{v \leftarrow \star}, v) = a] \\ &= \theta_v + \mu_v^\sigma(a) - \theta_v \\ &= \mu_v^\sigma(a). \end{aligned}$$

This shows the termination and correctness of `MarginSample`. □

We then verify the invariant conditions claimed in Lemma 5.4. Before that, we formally define the sequence of partial assignments that evolve in Algorithm 1.

Definition 5.6 (partial assignments in Algorithm 1). Let $X^0, X^1, \dots, X^n \in \mathcal{Q}^*$ denote the sequence of partial assignments, where $X^0 = \star^V$ and for every $1 \leq i \leq n$, X^i is the partial assignments X in Algorithm 1 after the i -th iteration of the **for** loop in Lines 1-4.

Fact 5.7. For each $1 \leq i \leq n$, either v_i is X^{i-1} -fixed, in which case $X^i = X^{i-1}$, or otherwise, in which case X^i extends X^{i-1} by assigning v_i a value in Q_{v_i} . Consequently, for each $1 \leq i \leq n$, if v_i is not X^{i-1} -fixed, then $X^*(v_i) = X^i(v_i)$, where X^* denotes the output of Algorithm 1.

Lemma 5.8. For the X^0, X^1, \dots, X^n in Definition 5.6, it holds for all $0 \leq i \leq n$ that X^i is feasible and

$$(12) \quad \forall c \in C, \quad \mathbb{P}[\neg c \mid X^i] \leq p'q.$$

Proof. We only need to prove (12). Then the feasibility of X^i follows from Corollary 4.4.

Note that if X^i changes from X^{i-1} , then v_i is not σ -fixed, which means that $\mathbb{P}[\neg c \mid X^{i-1}] \leq p'$ for all $c \in C$ that $v_i \in \text{vbl}(c)$. We then claim that for any $0 \leq i < n$ and any $c \in C$,

$$\mathbb{P}[\neg c \mid X^i] \leq p'q.$$

Fix any $c \in C$. Recall that $p' > p$ assuming the LLL condition (3). It holds that

$$\mathbb{P}[\neg c \mid X^0] = \mathbb{P}[\neg c \mid \star^V] = \mathbb{P}[\neg c] \leq p < p'.$$

We then claim that for any $0 \leq i < n$

$$\mathbb{P}[\neg c \mid X^i] > p' \implies \mathbb{P}[\neg c \mid X^{i+1}] > p'.$$

For $X^{i+1} \neq X^i$, as we observed, $\mathbb{P}[\neg c' \mid X^i] \leq p'$ for all $c' \in C$ that $v_{i+1} \in \text{vbl}(c')$, which means $v_{i+1} \notin \text{vbl}(c)$, and hence $\mathbb{P}[\neg c \mid X^{i+1}] = \mathbb{P}[\neg c \mid X^i] > p'$ since X^{i+1} differs from X^i only at v_{i+1} .

Therefore, if $\mathbb{P}[\neg c \mid X^n] \leq p'$ then $\mathbb{P}[\neg c \mid X^i] \leq p' < p'q$ for all $0 \leq i \leq n$.

Now suppose i^* to be the smallest $1 \leq i \leq n$ such that $\mathbb{P}[\neg c \mid X^i] > p'$. Then $\mathbb{P}[\neg c \mid X^{i^*-1}] \leq p'$, which means $X^{i^*} \neq X^{i^*-1}$, thus X^{i^*} extends X^{i^*-1} by assigning v_{i^*} some value in $Q_{v_{i^*}}$. Therefore,

$$(13) \quad \mathbb{P}[\neg c \mid X^{i^*}] \leq \frac{\mathbb{P}[\neg c \mid X^{i^*-1}]}{\min_{x \in Q_{v_{i^*}}} \mathbb{P}[v_{i^*} = x \mid X^{i^*-1}]} = p' |Q_{v_{i^*}}| \leq p'q.$$

And since $\mathbb{P}[\neg c \mid X^{i^*}] > p'$, by the above claim, $\mathbb{P}[\neg c \mid X^i] > p'$ for all $i^* \leq i \leq n$. Therefore, all variables in $\text{vbl}(c)$ are X^i -fixed for all $i^* \leq i \leq n$, and thus the variables in $\text{vbl}(c)$ will stay unchanged in Algorithm 1 for all these X^i , so will that $\mathbb{P}[\neg c \mid X^{i^*}] \leq p'q$. \square

The invariant of Condition 3.3 for MarginSample stated in Lemma 5.4-(1) follows easily from Lemma 5.9. To prove the invariant of Condition 3.8 for MarginOverflow, we show the following.

Lemma 5.9. Assume Condition 3.8 for (Φ, σ, v) . For any $u \in V$, if u is not σ -fixed, then $(\Phi, \sigma_{u \leftarrow a}, v)$ satisfies Condition 3.8 for any $a \in Q_u \cup \{\star\}$.

Proof. It suffices to verify the condition (12) for $\sigma_{u \leftarrow a}$. The feasibility of $\sigma_{u \leftarrow a}$ follows from Corollary 4.4.

If $a = \star$, (12) holds trivially for $\sigma_{u \leftarrow a}$, because as a not σ -fixed variable, u must have $\sigma(u) = \star$, and hence changing σ to $\sigma_{u \leftarrow \star}$ does not change the probability of any event conditional on σ . And for the case that $a \in Q_v$: if c is σ -frozen then $v \notin \text{vbl}(c)$ since u is not σ -fixed, and hence changing the value of u in σ does not affect $\mathbb{P}[\neg c \mid \sigma]$; and if otherwise c is not σ -frozen, which means $\mathbb{P}[\neg c \mid \sigma] \leq p'$, and hence as calculated in (13), we have

$$\mathbb{P}[\neg c \mid \sigma_{u \leftarrow a}] \leq |Q_{v_{i^*}}| \mathbb{P}[\neg c \mid \sigma] \leq p'q.$$

Therefore, the condition (12) remains to hold for $\sigma_{u \leftarrow a}$ for any not σ -fixed u and any $a \in Q_u \cup \{\star\}$. \square

The invariant of Condition 3.8 for MarginOverflow stated in Lemma 5.4-(2) follows from Lemma 5.9, because during the execution, the algorithm will only change an input partial assignment σ to $\sigma_{v \leftarrow a}$ for those vertices v that are not σ -fixed v and for $a \in Q_v \cup \{\star\}$. Lemma 5.4 is proved.

Combining Lemma 5.4 and Theorem 5.5, we prove the correctness of MarginSample (Algorithm 3), assuming the LLL condition in (3) for the input CSP in the main algorithm (Algorithm 1).

The correctness of RejectionSampling (Algorithm 2) has already been established in Theorem 3.4 (which is standard).

The correctness of the main sampling algorithm (Algorithm 1) then follows from the correctness of these two main subroutines. Note that this is not trivial because in Algorithm 1, the variables are

chosen to draw from their marginal distributions adaptive to randomness. We then formally prove that such being adaptive to randomness does not affect the correctness of sampling.

Proof of Theorem 5.1. By Lemmas 3.4 and Lemma 3.10, Algorithm 1 terminates with probability 1.

Let X^0, X^1, \dots, X^n be the sequence of partial assignments defined in Definition 5.6. Let X^* denote the output of Algorithm 1. We then show that, for every $\sigma \in \Omega$, $\Pr[X^* = \sigma] = \mu(\sigma)$.

Fix an arbitrary satisfying assignment $\sigma \in \Omega$. We further define a sequence of partial assignments $\sigma^0, \sigma^1, \dots, \sigma^n$ as follows. Let $\sigma^0 = \star^V$. For each $1 \leq i \leq n$, if v_i is σ^{i-1} -fixed, let $\sigma^i = \sigma^{i-1}$; otherwise, let $\sigma^i = \sigma^{i-1}_{v_i \leftarrow \sigma(v_i)}$. We claim that for every $0 \leq i \leq n$,

$$(14) \quad \prod_{j=1}^i \Pr[X^j = \sigma^j \mid X^{j-1} = \sigma^{j-1}] = \mu_{\Lambda(\sigma^i)}(\sigma_{\Lambda(\sigma^i)}),$$

with convention that both sides equal to 1 for $i = 0$.

We then prove this claim by an induction on i . The basis with $i = 0$ holds by the convention.

For the induction step, we consider $i \geq 1$. By the consistency of the oracle in Assumption 2, if $X^{i-1} = \sigma^{i-1}$, then v_i can only be simultaneously fixed or non-fixed in both X^{i-1} and σ^{i-1} .

- If v_i is σ^{i-1} -fixed, then $\sigma^i = \sigma^{i-1}$, and by Lines 3-4 of Algorithm 1, we have

$$\Pr[X^i = \sigma^i \mid X^{i-1} = \sigma^{i-1}] = 1.$$

Thus,

$$\begin{aligned} \prod_{j=1}^i \Pr[X^j = \sigma^j \mid X^{j-1} = \sigma^{j-1}] &= \prod_{j=1}^{i-1} \Pr[X^j = \sigma^j \mid X^{j-1} = \sigma^{j-1}] \\ &\quad (\text{by I.H.}) = \mu_{\Lambda(\sigma^{i-1})}(\sigma_{\Lambda(\sigma^{i-1})}) \\ &\quad (\text{since } \sigma^i = \sigma^{i-1}) = \mu_{\Lambda(\sigma^i)}(\sigma_{\Lambda(\sigma^i)}). \end{aligned}$$

- If v_i is not σ^{i-1} -fixed, then by the correctness of MarginSample (guaranteed by Lemma 5.4 and Theorem 5.5), we have

$$\Pr[X^i = \sigma^i \mid X^{i-1} = \sigma^{i-1}] = \mu_{v_i}^{\sigma^{i-1}}(\sigma(v_i)).$$

Thus, we have

$$\begin{aligned} \prod_{j=1}^i \Pr[X^j = \sigma^j \mid X^{j-1} = \sigma^{j-1}] &= \mu_{v_i}^{\sigma^{i-1}}(\sigma(v_i)) \cdot \prod_{j=1}^{i-1} \Pr[X^j = \sigma^j \mid X^{j-1} = \sigma^{j-1}] \\ &\quad (\text{by I.H.}) = \mu_{v_i}^{\sigma^{i-1}}(\sigma(v_i)) \cdot \mu_{\Lambda(\sigma^{i-1})}(\sigma_{\Lambda(\sigma^{i-1})}) \\ &\quad (\text{chain rule}) = \mu_{\Lambda(\sigma^i)}(\sigma_{\Lambda(\sigma^i)}). \end{aligned}$$

This finishes the induction. The claim in (14) is proved.

Observe that the sequence $X_0, X_1, \dots, X_n, X^*$ is a Markov chain, where the last step X^* is constructed from X^n by RejectionSampling. Suppose that event $X^* = \sigma$. By Fact 5.7 we have $X^i(v_i) = \sigma(v_i)$ if v_i is not X^{i-1} -fixed. Therefore according to Definition 5.6, we have that if v_i is X^{i-1} -fixed, then $X^i = X^{i-1}$; and if otherwise $X^i = X^{i-1}_{v_i \leftarrow \sigma(v_i)}$. Thus, given that $X^* = \sigma$ occurs, by the consistency of the oracle in Assumption 2, one can verify that $X^i = \sigma^i$ for all $0 \leq i \leq n$. Thus,

$$\begin{aligned} \Pr[X^* = \sigma] &= \Pr\left[(X^* = \sigma) \wedge \left(\bigwedge_{1 \leq i \leq n} (X^i = \sigma^i)\right)\right] \\ (\text{chain rule}) &= \Pr[X^* = \sigma \mid \forall 1 \leq i \leq n, X^i = \sigma^i] \cdot \prod_{i=1}^n \Pr[X^i = \sigma^i \mid \forall 0 \leq j < i, X^j = \sigma^j] \\ (\text{Markov property}) &= \Pr[X^* = \sigma \mid X^n = \sigma^n] \cdot \prod_{i=1}^n \Pr[X^i = \sigma^i \mid X^{i-1} = \sigma^{i-1}] \\ (\text{by (14)}) &= \Pr[X^* = \sigma \mid X^n = \sigma^n] \cdot \mu_{\Lambda(\sigma^n)}(\sigma_{\Lambda(\sigma^n)}) \end{aligned}$$

$$\begin{aligned}
(\text{by Theorem 3.4}) &= \mu_{V \setminus \Lambda(\sigma^n)}^{\sigma^n}(\sigma_{V \setminus \Lambda(\sigma^n)}) \cdot \mu_{\Lambda(\sigma^n)}(\sigma_{\Lambda(\sigma^n)}) \\
(\text{chain rule}) &= \mu(\sigma).
\end{aligned}$$

□

6. EFFICIENCY OF SAMPLING

In this section, we show the efficiency of Algorithm 1 under the LLL condition in (3).

Algorithm 1 assumes accesses to the following oracles for a class of constraints \mathcal{C} , both of which receive as input a constraint $c \in \mathcal{C}$ and a partial assignment $\sigma \in \mathcal{Q}^*$ upon queries:

- $\text{Eval}(c, \sigma)$: the evaluation oracle in Assumption 1, which decides whether $\mathbb{P}[c \mid \sigma] = 1$, that is, whether c is already satisfied by σ ;
- $\text{Frozen}(c, \sigma)$: the oracle for frozen decision in Assumption 2, which distinguishes between the two cases $\mathbb{P}[\neg c \mid \sigma] > p'$ and $\mathbb{P}[\neg c \mid \sigma] < 0.99p'$, where p' is the threshold defined in (6), and answers arbitrarily and consistently if otherwise.

The complexity of our sampling algorithm is measured in terms of the queries to the two oracles $\text{Eval}(\cdot)$ and $\text{Frozen}(\cdot)$, and the computation costs. We prove the following theorem.

Theorem 6.1. *Given as input a CSP formula $\Phi = (V, Q, C)$ satisfying (3), Algorithm 1 in expectation costs $O(q^2 k^2 \Delta^{10} n)$ queries to $\text{Eval}(\cdot)$, $O(k \Delta^7 n)$ queries to $\text{Frozen}(\cdot)$, and $O(q^3 k^3 \Delta^{10} n)$ in computation.*

Together with the correctness of Algorithm 1 stated in Theorem 5.1, this proves the main theorem for perfect sampling (Theorem 1.2), since any query to the oracle $\text{Frozen}(\cdot)$ can be resolved in $\text{poly}(q, k)$ time assuming the FPTAS for violation probability in the condition of Theorem 1.2.

Remark 6.2 (Monte Carlo realization of frozen decision). The oracle $\text{Frozen}(\cdot)$ can be realized probabilistically through the Monte Carlo method. Upon each query on a constraint c and a partial assignment σ , the two extreme cases $\mathbb{P}[\neg c \mid \sigma] > p'$ and $\mathbb{P}[\neg c \mid \sigma] < 0.99p'$ can be distinguished with high probability $(1 - \delta)$ by independently testing for $O(\frac{1}{p'} \log \frac{1}{\delta})$ times whether the constraint c is satisfied by a randomly generated assignment over $\text{vbl}(c)$ consistent with σ . We further apply a memoization to guarantee the consistency of the oracle as required in Assumption 2. The resulting algorithm is called Algorithm 1' and is formally described in Section 6.7.

This Monte Carlo realization of the $\text{Frozen}(\cdot)$ oracle introduces a bounded bias to the result of sampling and turns the perfect sampler in Theorem 6.1 to an approximate sampler Algorithm 1', which no longer assumes any nontrivial machinery beyond evaluating constraints.

Theorem 6.3. *Given as input an $\varepsilon \in (0, 1)$ and a CSP formula Φ satisfying (3), Algorithm 1' in expectation costs $O(q^2 k^2 \Delta^{11} n \log(\frac{\Delta n}{\varepsilon}))$ queries to $\text{Eval}(\cdot)$ and $O(q^3 k^3 \Delta^{11} n \log(\frac{\Delta n}{\varepsilon}))$ in computation, and outputs within ε total variation distance from the output of Algorithm 1 on input Φ .*

Together with the correctness of Algorithm 1 stated in Theorem 5.1, this proves the main theorem (Theorem 1.1) of the paper.

A notation for complexity bound: Throughout the section, we adopt the following abstract notation for any complexity bound. A complexity bound is expressed as a formal bi-variate linear function:

$$(15) \quad t(\underline{x}, \underline{y}) = \alpha \cdot \underline{x} + \beta \cdot \underline{y} + \gamma,$$

where α represents the number of queries to $\text{Eval}(\cdot)$, β represents the number of queries to $\text{Frozen}(\cdot)$, and γ represents the computation costs.

For examples, The complexity bounds in Theorems 6.1 and 6.3 are thus expressed respectively as:

$$O\left((q^2 k^2 \Delta^{10} n) \cdot \underline{x} + (k \Delta^7 n) \cdot \underline{y} + q^3 k^3 \Delta^{10} n\right) \text{ and } O\left(\left(q^2 k^2 \Delta^{11} n \log\left(\frac{\Delta n}{\varepsilon}\right)\right) \cdot \underline{x} + q^3 k^3 \Delta^{11} n \log\left(\frac{\Delta n}{\varepsilon}\right)\right).$$

We remark that such expression is only for notational convenience, because we want to handle three different complexity measures simultaneously in the analyses. Throughout our analyses, only linear calculations will be applied to such functions $t(\underline{x}, \underline{y})$. We further express:

$$\alpha \cdot \underline{x} + \beta \cdot \underline{y} + \gamma \leq \alpha' \cdot \underline{x} + \beta' \cdot \underline{y} + \gamma' \iff \alpha \leq \alpha' \wedge \beta \leq \beta' \wedge \gamma \leq \gamma'.$$

And we write $t(0, 0)$ for the constant term γ in (15), which stands for the computation cost.

6.1. Input model and data structure. Besides being accessed through the two oracles $\text{Eval}(\cdot)$ and $\text{Frozen}(\cdot)$, the input CSP formula $\Phi = (V, Q, C)$ is presented to the algorithm as follows:

- The variables in $V = \{v_1, v_2, \dots, v_n\}$ and constraints $C = \{c_1, c_2, \dots, c_m\}$ can be randomly accessed by their indices $i \in [n]$ and $j \in [m]$.
- Given any $c \in C$, the $\text{vbl}(c)$ can be retrieved in time $O(k)$; given any $v \in V$, the set of constraints c with $v \in \text{vbl}(c)$ can be retrieved in time $O(\Delta)$; given any $c \in C$, the set of dependent $c' \in C$ with $\text{vbl}(c') \cap \text{vbl}(c) \neq \emptyset$ can be retrieved within time $O(\Delta)$.

These requirements can be met by representing the CSP $\Phi = (V, Q, C)$ in its bipartite incidence graph and also the dependency graph, both using the adjacency linked list data structures.

The partial assignment $\sigma \in Q^*$ is maintained by the algorithm in such a way that passing σ to function as its argument takes $O(1)$ time. This can be resolved by storing σ globally as an array of $|V|$ stacks and passing a pointer to this array when σ is passed as a function argument, such that whenever a value x is assigned to $\sigma(v)$, x is pushed into the stack associated to v ; and when a function returns it pops the stacks associated to those variables that it has updated in the current level of recursion.

The partial assignment $\sigma \in Q^*$ also keeps a linked list of the variables currently set as \star .

6.2. The recursive cost tree (RCT). A crucial step for proving Theorem 6.1 is the analysis of the MarginSample (Algorithm 3), which calls to the recursive subroutine MarginOverflow (Algorithm 4).

Consider an input (Φ, σ, v) satisfying Condition 3.3 such that $\text{MarginSample}(\Phi, \sigma, v)$ is well-defined. Our goal is to upper bound the following complexity.

Definition 6.4. Let $\bar{t}_{\text{MS}}(\Phi, \sigma, v)$ denote the expected cost of $\text{MarginSample}(\Phi, \sigma, v)$ (Algorithm 3).

There are two nontrivial tasks involved in computing the $\text{MarginSample}(\Phi, \sigma, v)$: computing of the $\text{NextVar}(\sigma)$ and the Bernoulli factory, both of which are in the MarginOverflow (Algorithm 4).

Definition 6.5. Let $t_{\text{var}}(\sigma)$ denote the cost for deterministically computing $\text{NextVar}(\sigma)$ defined in (9). Let $\bar{t}_{\text{BF}}(\sigma)$ be the expected cost for the Bernoulli factory in Line 10 of Algorithm 4 in the worst case of v such that Condition 3.8 is satisfied, if there exists such a v ; and let $\bar{t}_{\text{BF}}(\sigma) = 0$, if no such v exists.

The $\bar{t}_{\text{BF}}(\sigma)$ upper bounds the expected cost for the Bernoulli factory on well-defined input (Φ, σ, v) .

The above complexity bounds $\bar{t}_{\text{MS}}(\Phi, \sigma, v)$, $t_{\text{var}}(\sigma)$, and $\bar{t}_{\text{BF}}(\sigma)$ are all expressed in the form of (15). The concrete bounds for $t_{\text{var}}(\sigma)$ and $\bar{t}_{\text{BF}}(\sigma)$ are proved respectively in Sections 6.5.1 and 6.5.2.

We first introduce a combinatorial structure that relates $\bar{t}_{\text{MS}}(\Phi, \sigma, v)$ to $t_{\text{var}}(\sigma)$ and $\bar{t}_{\text{BF}}(\sigma)$.

Definition 6.6 (recursive cost tree). For any $\sigma \in Q^*$, let $\mathcal{T}_\sigma = (T_\sigma, \rho_\sigma)$, where T_σ is a rooted tree with nodes $V(T_\sigma) \subseteq Q^*$ and $\rho_\sigma : V(T_\sigma) \rightarrow [0, 1]$ is a labeling of nodes in T_σ , be constructed as:

- (1) The root of T_σ is σ , with $\rho(\sigma) = 1$ and depth of σ being 0;
- (2) for $i = 0, 1, \dots$: for all nodes $\tau \in V(T_\sigma)$ of depth i in the current T_σ ,
 - (a) if $\text{NextVar}(\tau) = \perp$, then leave τ as a leaf node in T_σ ;
 - (b) otherwise, supposed $u = \text{NextVar}(\tau)$, append $\{\tau_{u \leftarrow x} \mid x \in Q_u \cup \{\star\}\}$ as the $q_u + 1$ children to the node τ in T_σ , and label them as:

$$\forall x \in Q_u \cup \{\star\}, \quad \rho_\sigma(\tau_{u \leftarrow x}) = \begin{cases} (1 - q_u \cdot \theta_u) \rho_\sigma(\tau) & \text{if } x = \star, \\ \mu_u^\sigma(x) \cdot \rho_\sigma(\tau) & \text{if } x \in Q_u. \end{cases}$$

The resulting $\mathcal{T}_\sigma = (T_\sigma, \rho_\sigma)$ is called the *recursive cost tree (RCT) rooted at σ* .

Define the following function $\lambda(\cdot)$ on RCTs $\mathcal{T}_\sigma = (T_\sigma, \rho_\sigma)$:

$$(16) \quad \lambda(\mathcal{T}_\sigma) \triangleq \sum_{\tau \in V(T_\sigma)} (\rho_\sigma(\tau) \cdot t_{\text{var}}(\tau)) + \sum_{\text{leaves } \tau \text{ in } T_\sigma} (\rho_\sigma(\tau) \cdot \bar{t}_{\text{BF}}(\tau)).$$

Note that $\lambda(\mathcal{T}_\sigma)$ is expressed in the form of (15) as the t_{var} and \bar{t}_{BF} .

The expected complexity of MarginSample is bounded through this function $\lambda(\mathcal{T}_\sigma)$.

Lemma 6.7. For any input (Φ, σ, v) satisfying Condition 3.3, it holds for $\sigma^* = \sigma_{v \leftarrow \star}$ that

$$\bar{t}_{\text{MS}}(\Phi, \sigma, v) \leq \lambda(\mathcal{T}_{\sigma^*}) + O(\gamma(\sigma^*)),$$

where $\gamma(\sigma^*) = \lambda(\mathcal{T}_{\sigma^*})(0, 0)$ is the constant term in $\lambda(\mathcal{T}_{\sigma^*})$ (standing for the computation cost as in (15)).

In the following, we prove Lemma 6.7. The following recursive relation for RCT is easy to verify.

Proposition 6.8. *Let $\sigma \in Q^*$ and $u = \text{NextVar}(\sigma)$. If $u \neq \perp$, then*

$$\lambda(\mathcal{T}_\sigma) = t_{\text{var}}(\sigma) + (1 - q_u \cdot \theta_u) \lambda(\mathcal{T}_{\sigma_{u \leftarrow \star}}) + \sum_{x \in Q_u} (\mu_u^\sigma(x) \cdot \lambda(\mathcal{T}_{\sigma_{u \leftarrow x}})).$$

Then we show that the complexity upper bound in Lemma 6.7 holds for the MarginOverflow.

Lemma 6.9. *Let (Φ, σ, v) be the input to MarginOverflow (Algorithm 4) satisfying Condition 3.8, and let $\bar{t}_{\text{MO}}(\Phi, \sigma, v)$ denote the expected cost of MarginOverflow(Φ, σ, v). It holds that*

$$\bar{t}_{\text{MO}}(\Phi, \sigma, v) \leq \lambda(\mathcal{T}_\sigma) + O(\gamma(\sigma)),$$

where $\gamma(\sigma) = \lambda(\mathcal{T}_\sigma)(0, 0)$ is the constant term in $\lambda(\mathcal{T}_\sigma)$.

Proof. Let $C > 0$ denote the constant computation cost that dominates the costs for argument passing and all computations in Lines 2-6 of MarginOverflow(Φ, σ, v). It suffices to show that

$$\bar{t}_{\text{MO}}(\Phi, \sigma, v) \leq \lambda(\mathcal{T}_\sigma) + C \cdot \gamma(\sigma).$$

We prove this by an induction on the structure of RCT.

The base case is when T_σ is just a single root, in which case $\text{NextVar}(\sigma) = \perp$, and by Definition 6.6,

$$\lambda(\mathcal{T}_\sigma) = \rho_\sigma(\sigma) \cdot t_{\text{var}}(\sigma) + \rho_\sigma(\sigma) \cdot \bar{t}_{\text{BF}}(\sigma) = t_{\text{var}}(\sigma) + \bar{t}_{\text{BF}}(\sigma).$$

Also if $\text{NextVar}(\sigma) = \perp$, the condition in Line 2 of MarginOverflow(Φ, σ, v) is unsatisfied and

$$\bar{t}_{\text{MO}}(\Phi, \sigma, v) \leq t_{\text{var}}(\sigma) + \mathbb{E}[T_{\text{BFS}}(\Phi, \sigma, v)] \leq t_{\text{var}}(\sigma) + \bar{t}_{\text{BF}}(\sigma) + C,$$

where $T_{\text{BFS}}(\Phi, \sigma, v)$ represents the cost of the Bernoulli factory in Line 10 of Algorithm 4, and by Definition 6.5, it holds that $\bar{t}_{\text{BF}}(\sigma) \geq \mathbb{E}[T_{\text{BFS}}(\Phi, \sigma, v)]$ for all such v that (Φ, σ, v) satisfies Condition 3.8. The base case is proved.

For the induction step, we assume that T_σ is a tree of depth > 0 . Thus by Definition 6.6, $\text{NextVar}(\sigma) = u \neq \perp$ for some $u \in V$. According to Lines 3-7 of MarginOverflow(Φ, σ, v), one can verify that for every $x \in Q_u$, the probability that $\sigma(u) = x$ upon Line 8 is

$$\Pr[r < q_u \cdot \theta_u] \cdot \frac{1}{q_u} + \Pr[r \geq q_u \cdot \theta_u] \cdot \Pr[\text{MarginOverflow}(\Phi, \sigma_{u \leftarrow \star}, u) = x] = \mu_u^\sigma(x),$$

where the first equality is due to the correctness of MarginOverflow guaranteed in Theorem 5.5.

By the law of total expectation,

$$(17) \quad \bar{t}_{\text{MO}}(\Phi, \sigma, v) = t_{\text{var}}(\sigma) + (1 - q_u \cdot \theta_u) \cdot \bar{t}_{\text{MO}}(\Phi, \sigma_{u \leftarrow \star}, u) + \sum_{x \in Q_u} (\mu_u^\sigma(x) \cdot \bar{t}_{\text{MO}}(\Phi, \sigma_{u \leftarrow x}, v)) + C.$$

Note that by Item 2b in Definition 6.6, for each $x \in Q_u \cup \{\star\}$, the subtree in T_σ rooted by $\sigma_{u \leftarrow x}$ is precisely the T_τ in the RCT $\mathcal{T}_\tau = (T_\tau, \rho_\tau)$ rooted at $\tau = \sigma_{u \leftarrow x}$. By Lemma 5.9, Condition 3.8 is still satisfied by $(\Phi, \sigma_{u \leftarrow x}, v)$. Thus, by induction hypothesis,

$$\bar{t}_{\text{MO}}(\Phi, \sigma_{u \leftarrow x}, u) \leq \lambda(\mathcal{T}_{\sigma_{u \leftarrow x}}) + C \cdot \gamma(\sigma_{u \leftarrow x}),$$

where $\gamma(\sigma_{u \leftarrow x}) = \lambda(\mathcal{T}_{\sigma_{u \leftarrow x}})(0, 0)$ represents the constant term in $\lambda(\mathcal{T}_{\sigma_{u \leftarrow x}})$. Combined with (17),

$$\begin{aligned} \bar{t}_{\text{MO}}(\Phi, \sigma, v) &\leq t_{\text{var}}(\sigma) + (1 - q_u \cdot \theta_u) (\lambda(\mathcal{T}_{\sigma_{u \leftarrow \star}}) + C \cdot \gamma(\sigma_{u \leftarrow \star})) \\ &\quad + \sum_{x \in Q_u} (\mu_u^\sigma(x) (\lambda(\mathcal{T}_{\sigma_{u \leftarrow x}}) + C \cdot \gamma(\sigma_{u \leftarrow x}))) + C \\ &= \lambda(\mathcal{T}_\sigma) + C \cdot \gamma(\sigma) - C(\gamma(t_{\text{var}}(\sigma))) + C \\ &\leq \lambda(\mathcal{T}_\sigma) + C \cdot \gamma(\sigma) \end{aligned}$$

where the equation is by Proposition 6.8, and $\gamma(t_{\text{var}}(\sigma)) = t_{\text{var}}(\sigma)(0, 0) \geq 1$ is the constant term in $t_{\text{var}}(\sigma)$ that represents the computation cost for NextVar(σ). \square

For (Φ, σ, v) satisfying Condition 3.3, $(\Phi, \sigma_{v \leftarrow \star}, v)$ satisfies Condition 3.8, hence

$$\bar{t}_{\text{MS}}(\Phi, \sigma, v) = (1 - q_v \theta_v) \bar{t}_{\text{MO}}(\Phi, \sigma_{v \leftarrow \star}, v) + O(1) \leq \lambda(\mathcal{T}_{\sigma_{v \leftarrow \star}}) + O(\gamma(\sigma_{v \leftarrow \star})),$$

where the inequality holds by Lemma 6.9. This proves Lemma 6.7.

6.3. A random path simulating RCT. The recursive cost tree defined in Definition 6.6 inspires the following random process of partial assignments.

Definition 6.10 (the $\text{Path}(\sigma)$ process). For any $\sigma \in \mathcal{Q}^*$, $\text{Path}(\sigma) = (\sigma_0, \sigma_1, \dots, \sigma_\ell)$ is a random sequence of partial assignments generated from the initial $\sigma_0 = \sigma$ as follows: for $i = 0, 1, \dots$,

- (1) if $\text{NextVar}(\sigma_i) = \perp$, the sequence stops at σ_i ;
- (2) if otherwise $u = \text{NextVar}(\sigma_i) \in V$, the partial assignment $\sigma_{i+1} \in \mathcal{Q}^*$ is generated from σ_i by randomly giving $\sigma(u)$ a value $x \in Q_u \cup \{\star\}$, such that

$$\Pr[\sigma_{i+1} = (\sigma_i)_{u \leftarrow \star}] = \frac{1 - q_u \cdot \theta_u}{2 - q_u \cdot \theta_u},$$

$$\forall x \in Q_u, \quad \Pr[\sigma_{i+1} = (\sigma_i)_{u \leftarrow x}] = \frac{\mu_u^{\sigma_i}(x)}{2 - q_u \cdot \theta_u}.$$

The length $\ell(\sigma)$ of $\text{Path}(\sigma) = (\sigma_0, \sigma_1, \dots, \sigma_{\ell(\sigma)})$ is a random variable whose distribution is determined by σ . We simply write $\ell = \ell(\sigma)$ and $\text{Path}(\sigma) = (\sigma_0, \sigma_1, \dots, \sigma_\ell)$ if σ is clear from the context.

It is quite obvious that $\text{Path}(\sigma)$ satisfies the Markov property. In fact, $\text{Path}(\sigma)$ can be seen as a Markov chain on space \mathcal{Q}^* such that any σ with $\text{NextVar}(\sigma) = \perp$ has a self-loop with probability 1.

The significance of the random process $\text{Path}(\sigma)$ is that it is related to the complexity of MarginSample through the following function: for any sequence $P = (\sigma_0, \sigma_1, \dots, \sigma_\ell) \in (\mathcal{Q}^*)^{\ell+1}$ with $\ell \geq 0$,

$$(18) \quad H(P) \triangleq \sum_{i=0}^{\ell} ((2 - q\theta)^i \cdot t_{\text{var}}(\sigma_i)) + (2 - q\theta)^\ell \cdot \bar{t}_{\text{BF}}(\sigma_\ell),$$

where $t_{\text{var}}(\cdot)$ and $\bar{t}_{\text{BF}}(\cdot)$ are defined in Definition 6.5, expressed in form of (15).

Recall the $\lambda(\mathcal{T}_\sigma)$ defined in (16). We have the following lemma.

Lemma 6.11. *Let $\sigma \in \mathcal{Q}^*$ be a partial assignment. The following holds for $P = \text{Path}(\sigma)$:*

$$\mathbb{E}[H(P)] \geq \lambda(\mathcal{T}_\sigma).$$

The following corollary follows immediately by combining Lemma 6.7 and Lemma 6.11.

Corollary 6.12. *For any (Φ, σ, v) satisfying Condition 3.3, supposed for $\sigma^* = \sigma_{u \leftarrow \star}$ that:*

$$\mathbb{E}[H(\text{Path}(\sigma^*))] = \alpha(\sigma^*) \cdot \underline{x} + \beta(\sigma^*) \cdot \underline{y} + \gamma(\sigma^*),$$

which is expressed in the form of (15), it holds that

$$\bar{t}_{\text{MS}}(\Phi, \sigma, v) \leq \alpha(\sigma^*) \cdot \underline{x} + \beta(\sigma^*) \cdot \underline{y} + O(\gamma(\sigma^*)).$$

Proof of Lemma 6.11. We prove this lemma by an induction on the structure of RCT. The base case is when \mathcal{T}_σ is just a single root, in which case $\text{NextVar}(\sigma) = \perp$, and by Definition 6.6,

$$\lambda(\mathcal{T}_\sigma) = \rho_\sigma(\sigma) \cdot t_{\text{var}}(\sigma) + \rho_\sigma(\sigma) \cdot \bar{t}_{\text{BF}}(\sigma) = t_{\text{var}}(\sigma) + \bar{t}_{\text{BF}}(\sigma).$$

Also, by $\text{NextVar}(\sigma) = \perp$ and Definition 6.10 we have $\text{Path}(\sigma) = (\sigma)$ and $\ell(\sigma) = 0$. Hence by (18),

$$\mathbb{E}[H(\text{Path}(\sigma))] = t_{\text{var}}(\sigma) + \bar{t}_{\text{BF}}(\sigma) = \lambda(\mathcal{T}_\sigma),$$

The base case is proved.

For the induction step, we assume that \mathcal{T}_σ is a tree of depth > 0 . Thus by Definition 6.6, $\text{NextVar}(\sigma) = u \neq \perp$ for some $u \in V$ and $\ell(\sigma) \geq 1$. According to Item 2 of Definition 6.10, we have

$$(19) \quad \Pr[\sigma_1 = \sigma_{u \leftarrow \star}] = \frac{1 - q_u \theta_u}{2 - q_u \theta_u}$$

$$(20) \quad \forall x \in Q_u, \quad \Pr[\sigma_1 = \sigma_{u \leftarrow x}] = \frac{\mu_u^\sigma(x)}{2 - q_u \theta_u}$$

Moreover, by the Markov property, given $\sigma_1 = \sigma_{u \leftarrow x}$ for each $x \in Q_u \cup \{\star\}$, the subsequence $(\sigma_1, \sigma_2, \dots, \sigma_{\ell(\sigma)})$ is identically distributed as $\text{Path}(\sigma_{u \leftarrow x})$.

It can be verified that for any sequence of partial assignments $P = (\sigma_0, \sigma_1, \dots, \sigma_\ell)$ with $\ell \geq 1$,

$$(21) \quad H(P) = t_{\text{var}}(\sigma_0) + (2 - q\theta)H((\sigma_1, \dots, \sigma_\ell)).$$

Therefore, conditioning on $\sigma_1 = \sigma_{u \leftarrow x}$ for each $x \in Q_u \cup \{\star\}$, we have

$$(22) \quad \begin{aligned} \mathbb{E}[H(\text{Path}(\sigma)) \mid \sigma_1 = \sigma_{u \leftarrow x}] &= t_{\text{var}}(\sigma) + (2 - q\theta) \mathbb{E}[H((\sigma_1, \dots, \sigma_\ell)) \mid \sigma_1 = \sigma_{u \leftarrow x}] \\ &= t_{\text{var}}(\sigma) + (2 - q\theta) \mathbb{E}[H(\text{Path}(\sigma_{u \leftarrow x}))]. \end{aligned}$$

Therefore by the law of total expectation, we have

$$(23) \quad \begin{aligned} \mathbb{E}[H(\text{Path}(\sigma))] &= \Pr[\sigma_1 = \sigma_{u \leftarrow \star}] \cdot \mathbb{E}[H(\text{Path}(\sigma)) \mid \sigma_1 = \sigma_{u \leftarrow \star}] \\ &\quad + \sum_{x \in Q_u} (\Pr[\sigma_1 = \sigma_{u \leftarrow x}] \cdot \mathbb{E}[H(\text{Path}(\sigma)) \mid \sigma_1 = \sigma_{u \leftarrow x}]) \\ &= \frac{1 - q_u \theta_u}{2 - q_u \theta_u} \cdot (t_{\text{var}}(\sigma) + (2 - q\theta) \mathbb{E}[H(\text{Path}(\sigma_{u \leftarrow \star}))]) \\ &\quad + \sum_{i \in Q_u} \left(\frac{\mu_u^\sigma(x)}{2 - q_u \theta_u} \cdot (t_{\text{var}}(\sigma) + (2 - q\theta) \mathbb{E}[H(\text{Path}(\sigma_{u \leftarrow x}))]) \right) \\ &\geq t_{\text{var}}(\sigma) + (1 - q_u \theta_u) \mathbb{E}[H(\text{Path}(\sigma_{u \leftarrow \star}))] + \sum_{x \in Q_u} (\mu_u^\sigma(x) \cdot \mathbb{E}[H(\text{Path}(\sigma_{u \leftarrow x}))]) \end{aligned}$$

where the second equality is by (19), (20), and (22), the last inequality is by $2 - q\theta \geq 2 - q_u \theta_u$ and

$$\frac{1 - q_u \theta_u}{2 - q_u \theta_u} + \sum_{x \in Q_u} \frac{\mu_u^\sigma(x)}{2 - q_u \theta_u} = 1.$$

Note that by Item 2b in Definition 6.6, for each $x \in Q_u \cup \{\star\}$, the subtree in T_σ rooted by $\sigma_{u \leftarrow x}$ is precisely the T_τ in the RCT $\mathcal{T}_\tau = (T_\tau, \rho_\tau)$ rooted at $\tau = \sigma_{u \leftarrow x}$. By the induction hypothesis,

$$\mathbb{E}[H(\text{Path}(\sigma_{u \leftarrow x}))] \geq \lambda(\mathcal{T}_{\sigma_{u \leftarrow x}}).$$

Combining with (23), we have

$$\mathbb{E}[H(\text{Path}(\sigma))] \geq t_{\text{var}}(\sigma_0) + (1 - q_u \theta_u) \cdot \lambda(\mathcal{T}_{\sigma_{u \leftarrow \star}}) + \sum_{x \in Q_u} (\mu_u^\sigma(x) \cdot \lambda(\mathcal{T}_{\sigma_{u \leftarrow x}})) = \lambda(\mathcal{T}_\sigma).$$

where the equality is by Proposition 6.8. \square

6.4. Refutation of long path. The partial assignment maintained in Algorithm 1 evolves as a random sequence X^0, X^1, \dots, X^n which was formally defined in Definition 5.6. The efficiency of the Margin-Sample (Algorithm 3) called within in Algorithm 1 crucially relies on that its input partial assignments are generated as this random sequence.

We define a procedure $\text{Simulate}(\cdot)$ such that $\text{Simulate}(t)$ generates the prefix (X^0, X^1, \dots, X^t) of the random partial assignments X^0, X^1, \dots, X^n maintained in Algorithm 1 defined in Definition 5.6. This is explicitly described in Algorithm 5, which is defined just to facilitate the analysis.

Algorithm 5: $\text{Simulate}(1 \leq t \leq n)$ with explicit randomness

```

1  $X^0 \leftarrow \star^V$ ;
2 for  $i = 1$  to  $t$  do
3   if  $v_i$  is not  $X^{i-1}$ -fixed then
4     choose  $r_{v_i} \in [0, 1)$  uniformly at random;
5     identify the unique  $b \in Q_{v_i}$  satisfying  $\sum_{a < b} \mu_{v_i}^{X^{i-1}}(a) \leq r_{v_i} < \sum_{a \leq b} \mu_{v_i}^{X^{i-1}}(a)$ ;
6      $X^i \leftarrow X_{v_i \leftarrow b}^{i-1}$ ;
7   else
8      $X^i \leftarrow X^{i-1}$ ;
9 return  $(X^0, X^1, \dots, X^t)$ ;

```

We may consider $\text{Simulate}(t-1) = (X^0, X^1, \dots, X^{t-1})$ and X_0^t constructed as that $X_0^t = X^{t-1}$ if v_t is X^{t-1} -fixed and $X_0^t = X_{v_i \leftarrow \star}^{t-1}$ if otherwise, which simulates what is passed to MarginOverflow and generates $\text{Path}(X_0^t) = (X_0^t, X_1^t, \dots, X_\ell^t)$.

Formally, for $1 \leq t \leq n$, we define the following random process:

$$(24) \quad \begin{aligned} (X^0, X^1, \dots, X^{t-1}) &\leftarrow \text{Simulate}(t-1), \\ X_0^t &\leftarrow \begin{cases} X^{t-1} & \text{if } v_t \text{ is } X^{t-1}\text{-fixed,} \\ X_{v_t \leftarrow \star}^{t-1} & \text{otherwise,} \end{cases} \\ (X_0^t, X_1^t, \dots, X_\ell^t) &\leftarrow \text{Path}(X_0^t). \end{aligned}$$

We want to bound the length ℓ of the resulting random path. But first, we give a witness for long path.

6.4.1. *A witness for long path.* Recall the C_{frozen}^σ defined in Definition 3.1 and the $C_{\star\text{-con}}^\sigma$ in Definition 3.6.

Definition 6.13 (σ -bad constraints). Let $\sigma \in Q^*$ be a partial assignment.

- Define $C_{\star\text{-frozen}}^\sigma \triangleq C_{\text{frozen}}^\sigma \cap C_{\star\text{-con}}^\sigma$.
- Define $C_\star^\sigma \triangleq \{c \in C \mid \exists v \in \text{vbl}(c) \text{ s.t. } \sigma(v) = \star\}$ to be the set of constraints involving \star .
- Define $C_{\star\text{-bad}}^\sigma \triangleq C_\star^\sigma \cup C_{\star\text{-frozen}}^\sigma$.

Intuitively, the purpose of $C_{\star\text{-bad}}^\sigma$ is to provide a witness for the deep recursion of `MarginOverflow`, such that a long `Path` may create many such bad constraints in $C_{\star\text{-bad}}^\sigma$.

The following lemma characterizes a bad event: there are many disjoint “bad” constraints as defined above given the current partial assignment σ . Here we show that within an LLL regime, such bad event rarely occurs if this σ is generated by the `Simulate` and `Path`.

Lemma 6.14. Assume $8\epsilon p \Delta^3 \leq 0.99p'$, where p' is fixed as in (6). Let $1 \leq t \leq n$ and let X_ℓ^t be generated as in (24). For any subset $T \subseteq C$ of disjoint constraints $c \in C$ such that $v_t \notin \text{vbl}(c)$,

$$\Pr \left[T \subseteq C_{\star\text{-bad}}^{X_\ell^t} \right] \leq \left(2e\Delta^3 \right)^{-|T|}.$$

The significance of this technical lemma will become more clear later in Section 6.4.2. In the rest of Section 6.4.1 we prove Lemma 6.14.

Note that the probability of the bad event in Lemma 6.14 is taken over the randomness of $C_{\star\text{-bad}}^{X_\ell^t}$, which comes solely from the randomness of `Simulate` and `Path`. Inspired by the idea of resampling table used in the analysis of the Moser-Tardos Algorithm [MT10], here we explicitly identify all randomness used in `Simulate` and `Path`, and later will couple them with the randomness used within the actual sampling algorithm. In order to do so, we give an implementation of `Path`(σ) with explicit uses of randomness in Algorithm 6.

Algorithm 6: `Path`($\sigma \in Q^*$) with explicit randomness

```

1  $\sigma_0 \leftarrow \sigma, i \leftarrow 0, u \leftarrow \text{NextVar}(\sigma);$ 
2 while  $u \neq \perp$  do
3    $i \leftarrow i + 1;$ 
4   choose  $h_u \in [0, 1)$  uniformly at random;
5   if  $h_u < \frac{1 - q_u \theta_u}{2 - q_u \theta_u}$  then
6      $\sigma(u) \leftarrow \star;$ 
7   else
8     choose  $r_u \in [0, 1)$  uniformly at random;
9     identify the unique  $b \in Q_u$  satisfying  $\sum_{a < b} \mu_u^\sigma(a) \leq r_u < \sum_{a \leq b} \mu_u^\sigma(a);$ 
10     $\sigma(u) \leftarrow b;$ 
11   $u \leftarrow \text{NextVar}(\sigma);$ 
12   $\sigma_i \leftarrow \sigma;$ 
13 return  $\sigma_0, \sigma_1, \dots, \sigma_i;$ 
```

The randomnesses used in Algorithm 5 and Algorithm 6 are: the $r_v \in [0, 1)$ accessed in Line 4 of Algorithm 5 and in Line 8 of Algorithm 6, and the $h_v \in [0, 1)$ accessed in Line 4 of Algorithm 6.

We assume that these random numbers $r_v \in [0, 1)$ and $h_v \in [0, 1)$ for all $v \in V$ are drawn uniformly and independently beforehand, and used in both Algorithm 5 and Algorithm 6 satisfying the following:

- same r_v is used for drawing the value of variable $v \in V$.

Note that this does not actually couple the randomness between the two processes Simulate and Path more than what they should be, and does not change the distribution of X_ℓ^t in Lemma 6.14. This is because within both procedures, a random r_v is ever used only if v is not σ -fixed in the current partial assignment σ , and meanwhile any assigned variable must also be σ -fixed, which means that each r_v can be used in at most one of Algorithm 5 and Algorithm 6.

Indeed, such identification of randomnesses does not affect the behaviors of Simulate and Path. Instead, doing so is for the purpose of coupling these two processes with the sampling algorithm.

The following facts regarding the random path constructed in (24) are easy to observe.

Fact 6.15. *Let $1 \leq t \leq n$ and Let $(X_0^t, X_1^t, \dots, X_\ell^t)$ be constructed as in (24). Then for every $v \neq v_t$:*

- (1) *if $X_\ell^t(v) = \star$, then v is accessed once in $\text{Path}(X_0^t)$, and never assigned in $\text{Path}(X_0^t)$ or $\text{Simulate}(t-1)$;*
- (2) *if $v \in \{v_1, \dots, v_{t-1}\}$ and $X_\ell^t(v) \in Q_v$, then v is assigned once in $\text{Simulate}(t-1)$ (with value $X_\ell^t(v)$) and is never accessed in $\text{Path}(X_0^t)$;*
- (3) *if $v \in \{v_{t+1}, \dots, v_n\}$ and $X_\ell^t(v) \in Q_v$, then v is assigned once in $\text{Path}(X_0^t)$ (with value $X_\ell^t(v)$) and is never assigned in $\text{Simulate}(t-1)$.*

They prove Lemma 6.14, which analyzes $C_{\star\text{-bad}}^{X_\ell^t}$. By Definition 6.13, $C_{\star\text{-bad}}^{X_\ell^t}$ contains two types of constraints: those in $C_\star^{X_\ell^t}$ and those in $C_{\star\text{-frozen}}^{X_\ell^t}$. First, the $C_\star^{X_\ell^t}$ is easy to deal with.

Lemma 6.16. *Under the condition of Lemma 6.14, for any $S \subseteq C$ such that $v_t \notin \text{vbl}(S)$, it holds that $S \subseteq C_\star^{X_\ell^t}$ only if for every $c \in S$, there exists a $v \in \text{vbl}(c)$ such that $h_v < \frac{1-q\theta}{2-q\theta}$.*

Proof. Fix a $c \in S$. By assumption we have $c \in C_\star^{X_\ell^t}$ and $v_t \notin \text{vbl}(c)$. Therefore by the definition of $C_\star^{X_\ell^t}$, we have $X_\ell^t(v) = \star$ for some $v \in \text{vbl}(c)$ with $v \neq v_t$.

Note that for any $v \neq v_t$, $X_\ell^t(v)$ is set to \star only at Line 4 of Algorithm 6, therefore

$$h_v < \frac{1 - q_v\theta_v}{2 - q_v\theta_v} \leq \frac{1 - q\theta}{2 - q\theta},$$

because $q\theta \leq q_v\theta_v$ by (7). □

Lemma 6.17 deals with the more complicated $C_{\star\text{-frozen}}^{X_\ell^t}$.

Lemma 6.17. *Assume $8\text{sep}\Delta^3 \leq 0.99p'$. Let $1 \leq t \leq n$ and let X_ℓ^t be generated as in (24) conditional on an arbitrarily fixed $(h_v)_{v \in V}$. Let $S \subseteq C$ be any subset of disjoint constraints $c \in C$ such that $v_t \notin \text{vbl}(c)$. Then*

$$\Pr \left[S \subseteq C_{\star\text{-frozen}}^{X_\ell^t} \right] \leq \left(4e\Delta^3 \right)^{-|S|}.$$

Let $T \subseteq C$ be any subset of disjoint constraints $c \in C$. Let $(X^0, X^1, \dots, X^n) = \text{Simulate}(n)$. Then

$$\Pr \left[T \subseteq C_{\text{frozen}}^{X^n} \right] \leq \left(4e\Delta^3 \right)^{-|T|}.$$

The following lemma is used in the proof of Lemma 6.17.

Lemma 6.18. *Under the condition of Lemma 6.17, it holds that for any $c \in S$ and $c' \in T$,*

$$\Pr \left[c \in C_{\text{frozen}}^{X_\ell^t} \right] \leq \frac{1}{4e\Delta^3} \quad \text{and} \quad \Pr \left[c' \in C_{\text{frozen}}^{X^n} \right] \leq \frac{1}{4e\Delta^3}.$$

Fix an arbitrary $c \in S$ and $\tau \in Q_{\text{vbl}(c)}$. A key observation for proving Lemma 6.18 is that conditioning on arbitrary assignments of random choices $(h_v)_{v \in V}$ and $(r_v)_{v \in V \setminus \text{vbl}(c)}$, and further conditioning on that $X_\ell^t = \tau$ over $\Lambda(X_\ell^t) \cap \text{vbl}(c)$, the sequences X^0, X^1, \dots, X^{t-1} and $X_0^t, X_1^t, \dots, X_\ell^t$ are fixed. Formally, we have the following lemma.

Lemma 6.19. *Under the condition of Lemma 6.17, fixed any $c \in S$, denoted that $U = \Lambda(X_\ell^t) \cap \text{vbl}(c)$, conditioned on any fixed $(r_v)_{v \in V \setminus \text{vbl}(c)}$, for any $\tau \in \mathcal{Q}_{\text{vbl}(c)}$ with $\Pr[X_\ell^t(U) = \tau(U)] > 0$, we have $X^0, X^1, \dots, X^{t-1}, X_0^t, X_1^t, \dots, X_\ell^t$ fully determined given that $X_\ell^t(U) = \tau(U)$.*

Proof. Fix any $v \in \text{vbl}(c)$. By the definition of S and $c \in S$, we have $v_t \notin \text{vbl}(c)$. Combining with $v \in \text{vbl}(c)$, we have $v \neq v_t$. By $X_\ell^t(U) = \tau(U)$, we have $X_\ell^t(v) \in \{\tau(v), \star, \star\}$. Thus by Fact 6.15, either v is never assigned, or v is assigned as $\tau(v)$ in $\text{Simulate}(t-1)$ and $\text{Path}(X_0^t)$. Note that in these procedures, r_v is only queried to assign v because in $\text{Path}(X_0^t)$ if v is accessed but not assigned, r_v is also not queried. Thus, we have either r_v is never queried or v is assigned as $\tau(v)$. Then, these procedures can be executed properly according to τ , without querying any r_v where $v \in \text{vbl}(c)$. Since h_v for every $v \in V$ and r_v for every $v \in V \setminus \text{vbl}(c)$ are fixed, we have all randomnesses in $\text{Simulate}(t-1)$ and $\text{Path}(X_0^t)$ determined. Thus $X^0, X^1, \dots, X^{t-1}, X_0^t, X_1^t, \dots, X_\ell^t$ are fixed. \square

By the same proof, we can also show the following lemma.

Lemma 6.20. *Under the condition of Lemma 6.17, fixed any $c \in T$, denoted that $U = \Lambda(X^n) \cap \text{vbl}(c)$, conditioned on any fixed $(r_v)_{v \in V \setminus \text{vbl}(c)}$, for any $\tau \in \mathcal{Q}_{\text{vbl}(c)}$ with $\Pr[X^n(U) = \tau(U)] > 0$, we have X^0, X^1, \dots, X^n fully determined given that $X^n(U) = \tau(U)$.*

Now we can prove Lemma 6.18.

Proof of Lemma 6.18. We denote $Y_1 = X_\ell^t$, $R_1 = S$; and $Y_2 = X^n$, $R_2 = T$. Fix any $\xi \in \{1, 2\}$ and $c \in R_\xi$. The lemma is proved if we show:

$$\Pr \left[c \in C_{\text{frozen}}^{Y_\xi} \right] \leq \frac{1}{4e\Delta^3}.$$

We further define two sequences $Z_1 = (X^0, X^1, \dots, X^{t-1}, X_0^t, X_1^t, \dots, X_\ell^t)$ and $Z_2 = (X^0, X^1, \dots, X^n)$. Let $U = \Lambda(Y_\xi) \cap \text{vbl}(c)$ and $\tau \in \mathcal{Q}_{\text{vbl}(c)}$ satisfy $\Pr[Y_\xi(U) = \tau(U)] > 0$. Assume $Y_\xi(U) = \tau(U)$. Then each $v \in U$ must be assigned in $\text{Simulate}(t-1)$ or $\text{Path}(X_0^t)$ if $\xi = 1$, and be assigned in $\text{Simulate}(n)$ if $\xi = 2$. For each $v \in U$, let Y_ξ^v be the partial assignment just before v is assigned in these subroutines. Formally, Y_ξ^v is last partial assignment in the sequence Z_ξ satisfying $Y_\xi^v(v) = \star$. By Lemmas 6.19 and 6.20, we have Y_ξ^v fixed. Thus, we have $Y_\xi(U) = \tau(U)$ only if for each $v \in U$,

$$(25) \quad \sum_{a < \tau(v)} \mu_v^{Y_\xi^v}(a) \leq r_v < \sum_{a \leq \tau(v)} \mu_v^{Y_\xi^v}(a).$$

We first prove (25). If $\xi = 1$, we have $c \in R_1 = S$. Combining with the definition of S , we have $v_t \notin \text{vbl}(c)$. Combining with $U = \Lambda(Y_\xi) \cap \text{vbl}(c)$, we have $v_t \notin U$. Thus, there are only three possibilities for each $v \in U$:

- Case.1: $\xi = 1$, $v \in \{v_1, \dots, v_{t-1}\}$, in which case v is assigned $\tau(v)$ in $\text{Simulate}(t-1)$ because of that $v \in U \subseteq \Lambda(Y_1) = \Lambda(X_\ell^t)$ and Item 2 of Fact 6.15, which according to Line 4 of $\text{Simulate}(t-1)$, means that r_v satisfies (25);
- Case.2: $\xi = 1$, $v \in \{v_{t+1}, \dots, v_n\}$, in which case v is assigned $\tau(v)$ in $\text{Path}(X_0^t)$, because of that $v \in U \subseteq \Lambda(Y_1) = \Lambda(X_\ell^t)$ and Item 3 of Fact 6.15, which according to Line 8 of $\text{Path}(X_0^t)$, means that r_v satisfies (25).
- Case.3: $\xi = 2$, in which case v is assigned as $\tau(v)$ in $\text{Simulate}(n)$ because of that $v \in U \subseteq \Lambda(Y_2) = \Lambda(X^n)$, which according to Line 4 of $\text{Simulate}(n)$, means that r_v satisfies (25).

Let \mathcal{B}_v denote the event in (25). We have $Y_\xi(U) = \tau(U)$ only if \mathcal{B}_v occurs for every $v \in U$. Thus,

$$(26) \quad \Pr[Y_\xi(U) = \tau(U)] \leq \Pr \left[\bigwedge_{v \in U} \mathcal{B}_v \right] = \prod_{v \in U} \Pr[\mathcal{B}_v] = \prod_{v \in U} \mu_v^{Y_\xi^v}(\tau(v)) \leq (1 + \eta)^{|U|} \cdot |Q_U|^{-1},$$

where the first equality is by the independence of r_v , and the last inequality is by Theorem 4.2.

If $Y_\xi(U) = \tau(U)$, then we have Y_ξ fixed by Lemmas 6.19 and 6.20. We use σ_ξ to denote the value of Y_ξ fixed by these lemmas given $Y_\xi(U) = \tau(U)$. Note that σ_ξ is determined by τ , $(h_v)_{v \in V}$ and $(r_v)_{v \in V \setminus \text{vbl}(c)}$

if $\xi = 1$ and by τ and $(r_v)_{v \in V \setminus \text{vbl}(c)}$ if $\xi = 2$. Let $L = \Lambda(\sigma_\xi) \cap \text{vbl}(c)$. If $Y_\xi(U) = \tau(U)$, we have $Y_\xi = \sigma_\xi$, and then

$$U = \Lambda(Y_\xi) \cap \text{vbl}(c) = \Lambda(\sigma_\xi) \cap \text{vbl}(c) = L.$$

Combining with (26), we have

$$\Pr[Y_\xi(U) = \tau(U)] \leq (1 + \eta)^{|L|} \cdot |\mathcal{Q}_L|^{-1}.$$

According to the definition of σ_ξ , we have $\sigma_\xi(L) = \tau(L)$. This is because given $Y_\xi(U) = \tau(U)$, by $Y_\xi = \sigma_\xi$ and $U = L$, it must hold that $\sigma_\xi(L) = Y_\xi(L) = Y_\xi(U) = \tau(U) = \tau(L)$. Thus, we have $Y_\xi(U) = \tau(U)$ if $Y_\xi = \sigma_\xi$, because given $Y_\xi = \sigma_\xi$, we have $\Lambda(Y_\xi) = \Lambda(\sigma_\xi)$ and then $U = L$, which combining with $\sigma_\xi(L) = \tau(L)$, means that $Y_\xi(U) = Y_\xi(L) = \sigma_\xi(L) = \tau(L) = \tau(U)$. Thus, we have

$$(27) \quad \Pr[Y_\xi = \sigma_\xi] \leq \Pr[Y_\xi(U) = \tau(U)] \leq (1 + \eta)^{|L|} \cdot |\mathcal{Q}_L|^{-1}.$$

For each $\sigma'_\xi \in \mathcal{Q}^*$ where $\sigma'_\xi \neq \sigma_\xi$, if $\mathbb{P}[\tau \mid \sigma'_\xi] > 0$ we have $\sigma'_\xi(v) = \tau(v)$ for all $v \in \Lambda(\sigma'_\xi) \cap \text{vbl}(c)$. Thus, $Y_\xi \neq \sigma'_\xi$. Otherwise, suppose $Y_\xi = \sigma'_\xi$. Combining with that $\sigma'_\xi(v) = \tau(v)$ for all $v \in \Lambda(\sigma'_\xi) \cap \text{vbl}(c)$, we have $Y_\xi(U) = \tau(U)$. Thus, we have $Y_\xi = \sigma_\xi$, which is contradictory to that $Y_\xi = \sigma'_\xi$ and $\sigma'_\xi \neq \sigma_\xi$.

In summary, we have either $\mathbb{P}[\tau \mid \sigma'_\xi] = 0$ or $\Pr[Y = \sigma'_\xi] = 0$ for each $\sigma'_\xi \neq \sigma_\xi$. By the law of total expectation,

$$(28) \quad \mathbb{E}[\mathbb{P}[\tau \mid Y_\xi]] = \sum_{\sigma'_\xi \in \mathcal{Q}^*} \mathbb{P}[\tau \mid \sigma'_\xi] \Pr[Y = \sigma'_\xi] = \mathbb{P}[\tau \mid \sigma_\xi] \Pr[Y_\xi = \sigma_\xi].$$

Recall that $L = \Lambda(\sigma_\xi) \cap \text{vbl}(c)$ and $\sigma_\xi(L) = \tau(L)$. Thus we have

$$(29) \quad \mathbb{P}[\tau \mid \sigma_\xi] = |\mathcal{Q}_{\text{vbl}(c) \setminus L}|^{-1}.$$

Thus, combining (28), (29) with (27) we have

$$\mathbb{E}[\mathbb{P}[\tau \mid Y_\xi]] \leq (1 + \eta)^{|L|} \cdot |\mathcal{Q}_{\text{vbl}(c)}|^{-1} \leq (1 + \eta)^k \cdot |\mathcal{Q}_{\text{vbl}(c)}|^{-1}.$$

Let $c^{-1}(\text{False}) \triangleq \{\tau \in \mathcal{Q}_{\text{vbl}(c)} \mid c(\tau) = \text{False}\}$ be the set of assignments on $\text{vbl}(c)$ that violate c . Then we have

$$\mathbb{E}[\mathbb{P}[\neg c \mid Y_\xi]] = \sum_{\tau \in c^{-1}(\text{False})} \mathbb{E}[\mathbb{P}[\tau \mid Y_\xi]] \leq |c^{-1}(\text{False})| \cdot (1 + \eta)^k \cdot |\mathcal{Q}_{\text{vbl}(c)}|^{-1},$$

In addition, by the definition of $c^{-1}(\text{False})$ we have

$$|c^{-1}(\text{False})| = \mathbb{P}[\neg c] \cdot |\mathcal{Q}_{\text{vbl}(c)}| \leq p \cdot |\mathcal{Q}_{\text{vbl}(c)}|$$

Therefore,

$$\mathbb{E}[\mathbb{P}[\neg c \mid Y_\xi]] \leq p(1 + \eta)^k.$$

Then, by Markov's inequality,

$$\Pr[\mathbb{P}[\neg c \mid Y_\xi] \geq 0.99p'] \leq \frac{p}{0.99p'} \cdot (1 + \eta)^k.$$

In addition, by (6) and (7) we have $\eta \leq (2k\Delta^2)^{-1}$. Also by $8ep\Delta^3 \leq 0.99p'$, we have if $\Delta \geq 2$,

$$\frac{p}{0.99p'} \cdot (1 + \eta)^k \leq \frac{1}{4e\Delta^3}.$$

Thus, we have

$$\Pr[\mathbb{P}[\neg c \mid Y_\xi] \geq 0.99p'] \leq \frac{p}{0.99p'} \cdot (1 + \eta)^k \leq \frac{1}{4e\Delta^3}.$$

Moreover, according to Remark 3.2, we have $c \in C_{\text{frozen}}^{Y_\xi}$ only if $\mathbb{P}[\neg c \mid Y_\xi] \geq 0.99p'$. Thus, we have

$$\Pr[c \in C_{\text{frozen}}^{Y_\xi}] \leq \Pr[\mathbb{P}[\neg c \mid Y_\xi] \geq 0.99p'] \leq \frac{1}{4e\Delta^3}. \quad \square$$

Now we can prove Lemma 6.17.

Proof of Lemma 6.17. We denote $Y_1 = X_\ell^t$, $R_1 = S$; and $Y_2 = X^n$, $R_2 = T$. Now we show that for $\xi \in \{1, 2\}$,

$$(30) \quad \Pr \left[R_\xi \subseteq C_{\text{frozen}}^{Y_\xi} \right] \leq \left(4e\Delta^3 \right)^{-|R_\xi|}.$$

Note that this immediately proves the case for $\xi = 2$ with $Y_2 = X^n$ and $R_2 = T$. As for the case $\xi = 1$ with $Y_1 = X_\ell^t$ and $R_1 = S$, observe that $C_{\star\text{-frozen}}^\sigma \subseteq C_{\text{frozen}}^\sigma$ for any σ , therefore the lemma in this case also follows from (30). In the rest, we prove (30).

Fix a $\xi \in [2]$. Let $\tau \in Q^*$. For each $H \subseteq R_\xi$, define

$$\begin{aligned} A_H &\triangleq \{ \mathbf{x} \in Q_{\text{vbl}(H)} \mid \forall v \in \Lambda(\tau) \cap \text{vbl}(H), \mathbf{x}(v) = \tau(v) \}, \\ B_H &\triangleq \{ \mathbf{x} \in A_H \mid \text{all } c \in H \text{ are false under the assignment } \mathbf{x} \}. \end{aligned}$$

Furthermore, for each $c \in C$, we write $A_c = A_{\{c\}}$ and $B_c = B_{\{c\}}$.

By S and T are sets of disjoint constraints, we have $\text{vbl}(c) \cap \text{vbl}(c') = \emptyset$ for any different $c, c' \in R_\xi$. Thus, we have $\bigotimes_{c \in R_\xi} A_c \subseteq \bigotimes_{c \in R_\xi} Q_{\text{vbl}(c)} = Q_{\text{vbl}(R_\xi)}$. Moreover, we can prove that $A_{R_\xi} = \bigotimes_{c \in R_\xi} A_c$ as follows:

- For each $\mathbf{x} \in A_{R_\xi}$, we have $\mathbf{x}(v) = \tau(v)$ for all $v \in \Lambda(\tau) \cap \text{vbl}(R_\xi)$. Thus, we also have $\mathbf{x}(v) = \tau(v)$ for all $c \in R_\xi$ and $v \in \Lambda(\tau) \cap \text{vbl}(c)$. Therefore, we have $\mathbf{x}(\text{vbl}(c)) \in A_c$ for each $c \in R_\xi$. Thus we have $A_{R_\xi} \subseteq \bigotimes_{c \in R_\xi} A_c$.
- For each $\mathbf{x} \in \bigotimes_{c \in R_\xi} A_c$, we have $\mathbf{x}(v) = \tau(v)$ for all $c \in R_\xi$ and $v \in \Lambda(\tau) \cap \text{vbl}(c)$. Thus, we also have $\mathbf{x}(v) = \tau(v)$ for all $v \in \Lambda(\tau) \cap \text{vbl}(R_\xi)$. Therefore, we have $\mathbf{x} \in A_{R_\xi}$. Thus we have $\bigotimes_{c \in R_\xi} A_c \subseteq A_{R_\xi}$.

Similarly, we can also prove that $B_{R_\xi} = \bigotimes_{c \in R_\xi} B_c$ as follows:

- For each $\mathbf{x} \in B_{R_\xi}$, we have $\mathbf{x} \in A_{R_\xi}$. By $A_{R_\xi} = \bigotimes_{c \in R_\xi} A_c$, we have $\mathbf{x} \in \bigotimes_{c \in R_\xi} A_c$. Therefore, $\mathbf{x}(\text{vbl}(c)) \in A_c$ for each $c \in R_\xi$. Moreover, by $\mathbf{x} \in B_{R_\xi}$ we have all $c \in R_\xi$ are false under the assignment \mathbf{x} . Equivalently, each $c \in R_\xi$ is false under $\mathbf{x}(\text{vbl}(c))$. Combining with $\mathbf{x}(\text{vbl}(c)) \in A_c$, we have $\mathbf{x}(\text{vbl}(c)) \in B_c$ for each $c \in R_\xi$. Thus, we have $\mathbf{x} \in \bigotimes_{c \in R_\xi} B_c$. In summary, we have $B_{R_\xi} \subseteq \bigotimes_{c \in R_\xi} B_c$.
- For each $\mathbf{x} \in \bigotimes_{c \in R_\xi} B_c$, we have $\mathbf{x}(\text{vbl}(c)) \in A_c$ and c is false under $\mathbf{x}(\text{vbl}(c))$ for each $c \in R_\xi$. By $\mathbf{x}(\text{vbl}(c)) \in A_c$ for each $c \in R_\xi$ and $A_{R_\xi} = \bigotimes_{c \in R_\xi} A_c$, we have $\mathbf{x} \in A_{R_\xi}$. Combining with that each $c \in R_\xi$ is false under $\mathbf{x}(\text{vbl}(c))$, we have $\mathbf{x} \in B_{R_\xi}$. In summary, we have $\bigotimes_{c \in R_\xi} B_c \subseteq B_{R_\xi}$.

Due to the law \mathbb{P} we have

$$\mathbb{P}[\neg c \mid \tau] = \frac{|B_c \otimes Q_{V \setminus \Lambda(\tau)}|}{|A_c \otimes Q_{V \setminus \Lambda(\tau)}|} = \frac{|B_c|}{|A_c|}.$$

Similarly, we have

$$\mathbb{P} \left[\bigwedge_{c \in R_\xi} (\neg c) \mid \tau \right] = \frac{|B_{R_\xi} \otimes Q_{V \setminus \Lambda(\tau)}|}{|A_{R_\xi} \otimes Q_{V \setminus \Lambda(\tau)}|} = \frac{|B_{R_\xi}|}{|A_{R_\xi}|} = \frac{|\bigotimes_{c \in R_\xi} B_c|}{|\bigotimes_{c \in R_\xi} A_c|} = \prod_{c \in R_\xi} \frac{|B_c|}{|A_c|} = \prod_{c \in R_\xi} \mathbb{P}[\neg c \mid \tau].$$

Thus given Y_ξ , we have

$$\begin{aligned} \mathbb{P} \left[\bigwedge_{c \in R_\xi} (\neg c) \mid Y_\xi \right] &= \frac{\sum_{\tau \in Q^*} \left(\Pr[Y_\xi = \tau] \cdot \mathbb{P} \left[\bigwedge_{c \in R_\xi} (\neg c) \mid \tau \right] \right)}{\sum_{\tau \in Q^*} \Pr[Y_\xi = \tau]} \\ &= \frac{\sum_{\tau \in Q^*} \left(\Pr[Y_\xi = \tau] \cdot \prod_{c \in R_\xi} \mathbb{P}[\neg c \mid \tau] \right)}{\sum_{\tau \in Q^*} \Pr[Y_\xi = \tau]} \\ &= \prod_{c \in R_\xi} \mathbb{P}[\neg c \mid Y_\xi]. \end{aligned}$$

If $\mathbb{P}[\neg c \mid Y_\xi] \geq 0.99p'$ for each $c \in R_\xi$, then

$$\mathbb{P}\left[\bigwedge_{c \in R_\xi} (\neg c) \mid Y_\xi\right] = \prod_{c \in R_\xi} \mathbb{P}[\neg c \mid Y_\xi] \geq (0.99p')^{|R_\xi|}.$$

According to Remark 3.2, we have $R_\xi \subseteq C_{\text{frozen}}^{Y_\xi}$ only if $\mathbb{P}[\neg c \mid Y_\xi] \geq 0.99p'$ for each $c \in R_\xi$. Therefore,

$$\Pr\left[R_\xi \subseteq C_{\text{frozen}}^{Y_\xi}\right] \leq \Pr\left[\mathbb{P}\left[\bigwedge_{c \in R_\xi} (\neg c) \mid Y_\xi\right] \geq (0.99p')^{|R_\xi|}\right].$$

Thus, to (30), it is sufficient to prove the following:

$$(31) \quad \Pr\left[\mathbb{P}\left[\bigwedge_{c \in R_\xi} (\neg c) \mid Y_\xi\right] \geq (0.99p')^{|R_\xi|}\right] \leq (4e\Delta^3)^{-|R_\xi|}.$$

In the following, we prove (31). Assume w.l.o.g. that every r_v with $v \in V \setminus \text{vbl}(R_\xi)$ is fixed. Define

$$R_\xi^{-1}(\text{False}) \triangleq \left\{ \tau \in \mathcal{Q}_{\text{vbl}(R_\xi)} \mid \forall c \in R_\xi, c(\tau_{\text{vbl}(c)}) = \text{False} \right\}.$$

Given $\tau \in R_\xi^{-1}(\text{False})$, by going through the proof of Lemma 6.18, one can verify that

$$\mathbb{E}[\mathbb{P}(\tau \mid Y_\xi)] \leq (1 + \eta)^{k|R_\xi|} \cdot |\mathcal{Q}_{\text{vbl}(R_\xi)}|^{-1}.$$

Thus we have

$$\mathbb{E}\left[\mathbb{P}\left[\bigwedge_{c \in R_\xi} (\neg c) \mid Y_\xi\right]\right] = \sum_{\tau \in R_\xi^{-1}(\text{False})} \mathbb{E}[\mathbb{P}(\tau \mid Y_\xi)] \leq |R_\xi^{-1}(\text{False})| \cdot (1 + \eta)^{k|R_\xi|} \cdot |\mathcal{Q}_{\text{vbl}(R_\xi)}|^{-1}.$$

By definition of $R_\xi^{-1}(\text{False})$ we have

$$|R_\xi^{-1}(\text{False})| = \mathbb{P}\left[\bigwedge_{c \in R_\xi} (\neg c)\right] \cdot |\mathcal{Q}_{\text{vbl}(R_\xi)}| \leq p^{|R_\xi|} \cdot |\mathcal{Q}_{\text{vbl}(R_\xi)}|.$$

Therefore,

$$\mathbb{E}\left[\mathbb{P}\left[\bigwedge_{c \in R_\xi} (\neg c) \mid Y_\xi\right]\right] \leq p^{|R_\xi|} (1 + \eta)^{k|R_\xi|}.$$

Then, by Markov's inequality we have

$$\Pr\left[\mathbb{P}\left[\bigwedge_{c \in R_\xi} (\neg c) \mid Y_\xi\right] \geq (0.99p')^{|R_\xi|}\right] \leq \left(\frac{p}{0.99p'}\right)^{|R_\xi|} \cdot (1 + \eta)^{k|R_\xi|}.$$

In addition, by (6) and (7) we have $\eta \leq (2k\Delta^2)^{-1}$. Also by $8ep\Delta^3 \leq 0.99p'$, we have if $\Delta \geq 2$,

$$\frac{p}{0.99p'} \cdot (1 + \eta)^k \leq \frac{1}{4e\Delta^3}.$$

Then (31) is immediate. □

Now we can prove Lemma 6.14, the main conclusion of Section 6.4.1.

Proof of Lemma 6.14. For each constraint $c \in C$, let \mathcal{B}_c denote the event that there is a $v \in \text{vbl}(c)$ such that $h_v < \frac{1-q\theta}{2-q\theta}$. Let $\mathbf{h} = (h_v)_{v \in V}$ range over $[0, 1]^V$. Let $\vec{0} = (0)_{v \in V}$ and $\vec{1} = (1)_{v \in V}$. For any $S \subseteq T$,

$$\begin{aligned}
\Pr \left[\left(\bigwedge_{c \in S} \mathcal{B}_c \right) \wedge \left((T \setminus S) \subseteq C_{\star\text{-frozen}}^{X_t^t} \right) \right] &\leq \int_{\vec{0}}^{\vec{1}} \Pr \left[\left(\bigwedge_{c \in S} \mathcal{B}_c \right) \wedge \left((T \setminus S) \subseteq C_{\star\text{-frozen}}^{X_t^t} \right) \mid \mathbf{h} \right] d\mathbf{h} \\
&= \int_{\vec{0}}^{\vec{1}} \Pr \left[\bigwedge_{c \in S} \mathcal{B}_c \mid \mathbf{h} \right] \Pr \left[(T \setminus S) \subseteq C_{\star\text{-frozen}}^{X_t^t} \mid \mathbf{h} \right] d\mathbf{h} \\
(32) \quad &\leq (4e\Delta^3)^{-|T \setminus S|} \int_{\vec{0}}^{\vec{1}} \Pr \left[\bigwedge_{c \in S} \mathcal{B}_c \mid \mathbf{h} \right] d\mathbf{h} \\
&= (4e\Delta^3)^{-|T \setminus S|} \Pr \left[\bigwedge_{c \in S} \mathcal{B}_c \right],
\end{aligned}$$

where the integrals are Lebesgue integrals, the first equality is by the independence of $\bigwedge_{c \in S} \mathcal{B}_c$ and $(T \setminus S) \subseteq C_{\star\text{-frozen}}^{X_t^t}$ given \mathbf{h} , and the last inequality is by Lemma 6.17. By union bound we have

$$\Pr [\mathcal{B}_c] \leq \sum_{v \in \text{vbl}(c)} \Pr \left[h_v < \frac{1-q\theta}{2-q\theta} \right] \leq \frac{k(1-q\theta)}{2-q\theta}.$$

By (6) and (7), we have $1 - q\theta \leq (4ek\Delta^3)^{-1}$ and

$$\Pr [\mathcal{B}_c] \leq (4e\Delta^3)^{-1}.$$

By $S \subseteq T$ and that $\text{vbl}(c) \cap \text{vbl}(c') = \emptyset$ for any different $c, c' \in T$, we also have $\text{vbl}(c) \cap \text{vbl}(c') = \emptyset$ for any different $c, c' \in S$. Thus, the events in $\{\mathcal{B}_c : c \in S\}$ are independent. Therefore,

$$\Pr \left[\bigwedge_{c \in S} \mathcal{B}_c \right] = \prod_{c \in S} \Pr [\mathcal{B}_c] \leq (4e\Delta^3)^{-|S|}.$$

Combining with (32), we have

$$\Pr \left[\left(\bigwedge_{c \in S} \mathcal{B}_c \right) \wedge \left((T \setminus S) \subseteq C_{\star\text{-frozen}}^{X_t^t} \right) \right] \leq (4e\Delta^3)^{-|T|}.$$

In addition, by Lemma 6.16 we have $S \subseteq C_{\star}^{X_t^t}$ only if $\bigwedge_{c \in S} \mathcal{B}_c$. Thus we have

$$\Pr \left[\left(S \subseteq C_{\star}^{X_t^t} \right) \wedge \left((T \setminus S) \subseteq C_{\star\text{-frozen}}^{X_t^t} \right) \right] \leq \Pr \left[\left(\bigwedge_{c \in S} \mathcal{B}_c \right) \wedge \left((T \setminus S) \subseteq C_{\star\text{-frozen}}^{X_t^t} \right) \right] \leq (4e\Delta^3)^{-|T|}.$$

If $T \subseteq C_{\star\text{-bad}}^{X_t^t}$, then by $C_{\star\text{-bad}}^{X_t^t} = C_{\star}^{X_t^t} \cup C_{\star\text{-frozen}}^{X_t^t}$, there exists a subset $S \subseteq T$ such that $S \subseteq C_{\star}^{X_t^t}$ and $(T \setminus S) \subseteq C_{\star\text{-frozen}}^{X_t^t}$. Thus, we have

$$\begin{aligned}
\Pr [T \subseteq C_{\star\text{-bad}}^{X_t^t}] &\leq \sum_{S \subseteq T} \Pr \left[\left(S \subseteq C_{\star}^{X_t^t} \right) \wedge \left((T \setminus S) \subseteq C_{\star\text{-frozen}}^{X_t^t} \right) \right] \\
&\leq \sum_{S \subseteq T} (4e\Delta^3)^{-|T|} = 2^{|T|} \cdot (4e\Delta^3)^{-|T|} = (2e\Delta^3)^{-|T|}. \quad \square
\end{aligned}$$

6.4.2. Tail bound for long path via $\{2, 3\}$ -tree. We now show that with high probability the random path generated as in (24) cannot be too long.

Theorem 6.21. Assume $8ep\Delta^3 \leq 0.99p'$. Let $1 \leq t \leq n$ and X_t^t be generated as in (24). For integers $i \geq 0$,

$$\Pr [\ell \geq ik\Delta^2] \leq \Delta \cdot 2^{-i}.$$

A main tool used in the proof of this theorem is the $\{2, 3\}$ -tree, defined by Alon [Alo91].

Definition 6.22 ($\{2, 3\}$ -tree). Let $G = (V, E)$ be a graph and $\text{dist}_G(\cdot, \cdot)$ denote the shortest path distance in G . A $\{2, 3\}$ -tree in G is a subset of vertices $T \subseteq V$ such that:

- for any $u, v \in T$, $\text{dist}_G(u, v) \geq 2$;
- T is connected if an edge is added between every $u, v \in T$ such that $\text{dist}_G(u, v) \in \{2, 3\}$.

We define the following notions related to the dependency graph in the Lovász local lemma.

Definition 6.23 (dependency graphs). Let $\Phi = (V, Q, C)$ be a CSP formula and let $S \subseteq C$.

- $G(S)$ denotes the dependency graph induced by S , which is a graph with vertex set S such that there is an edge between distinct $c, c' \in S$ if and only if $\text{vbl}(c) \cap \text{vbl}(c') \neq \emptyset$.
- $G^2(S)$ denotes the square graph of $G(S)$, in which there is an edge between distinct $c, c' \in S$ if and only if $\text{dist}_{G(C)}(c, c') \leq 2$.

The following two lemmas are known.

Lemma 6.24 ([GLLZ19]). Let $S \subseteq C$ be a set of constraints where $G^2(S)$ is connected. Then for each $c \in S$, there exists a $\{2, 3\}$ -tree $T \subseteq S$ in $G(C)$ such that $c \in T$ and $|T| \geq |S|/\Delta$.

Lemma 6.25 ([Alo91, Lemma 2.1]). Let $G = (V, E)$ be a graph with maximum degree d . Then, for any $v \in V$, the number of $\{2, 3\}$ -trees in G of size t containing v is at most $\frac{(ed^3)^{t-1}}{2}$.

Let $\sigma \in Q^*$ be a partial assignment such that only one variable $v \in V$ has $\sigma(v) = \star$. The following lemma shows that if the path $\text{Path}(\sigma) = (\sigma_0, \sigma_1, \dots, \sigma_\ell)$ generated from such σ is long, then there must be a large $\{2, 3\}$ -tree in $C_{\star\text{-bad}}^{\sigma_\ell}$.

Lemma 6.26. Let $\sigma \in Q^*$ be a partial assignment with exactly one variable $v \in V$ having $\sigma(v) = \star$, and let $\text{Path}(\sigma) = (\sigma_0, \sigma_1, \dots, \sigma_\ell)$. Then it always holds that

$$\ell \leq k\Delta |C_{\star\text{-bad}}^{\sigma_\ell}|.$$

Moreover, there always exists a $\{2, 3\}$ -tree T in $G(C)$ with $v \in \text{vbl}(T)$ such that

$$T \subseteq C_{\star\text{-bad}}^{\sigma_\ell} \quad \text{and} \quad \Delta |T| \geq |C_{\star\text{-bad}}^{\sigma_\ell}|.$$

The proof of Lemma 6.26 relies on a careful verification of monotonicity of several properties along $\text{Path}(\sigma) = (\sigma_0, \dots, \sigma_\ell)$, including C_{frozen}^σ , $C_{\star\text{-con}}^\sigma$, $C_{\star\text{-bad}}^{\sigma_\ell}$, etc. The proof is deferred to Appendix C.1.

With these lemmas, we can prove the tail bound on the length of random path stated in Theorem 6.21.

Proof of Theorem 6.21. The case $i = 0$ is trivial. In the following, we assume $i > 0$. Denote $v = v_t$. Define

$$\mathcal{T}_v^i \triangleq \{\{2, 3\}\text{-tree } T \mid v \in \text{vbl}(T) \wedge |T| = i\}.$$

By Lemma 6.25, for any c with $v \in \text{vbl}(c)$, the number of $\{2, 3\}$ -trees in $G(C)$ of size i containing c is at most $\frac{(e\Delta^3)^{i-1}}{2}$. Note that $|\{c \mid v \in \text{vbl}(c)\}| \leq \Delta$. We then have

$$(33) \quad |\mathcal{T}_v^i| \leq \frac{\Delta}{2} \cdot (e\Delta^3)^{i-1}.$$

For each $\{2, 3\}$ -tree $T \in \mathcal{T}_v^i$, there is $c \in T$ such that $v \in \text{vbl}(c)$ since $v \in \text{vbl}(T)$. And moreover, such $c \in T$ with $v \in \text{vbl}(c)$ must be unique since T is a $\{2, 3\}$ -tree, because if otherwise there is another constraint $c' \in T$ with $v \in \text{vbl}(c')$, then $v \in \text{vbl}(c) \cap \text{vbl}(c')$, which means $\text{dist}_{G(C)}(c, c') = 1$.

Thus, by Lemma 6.14, for each $T \in \mathcal{T}_v^i$,

$$\Pr \left[T \subseteq C_{\star\text{-bad}}^{X_t^i} \right] \leq \Pr \left[(T \setminus \{c\}) \subseteq C_{\star\text{-bad}}^{X_t^i} \right] \leq (2e\Delta^3)^{1-i}.$$

In addition, by Lemma 6.26, we have

$$\Pr \left[\ell \geq ik\Delta^2 \right] \leq \sum_{T \in \mathcal{T}_v^i} \Pr \left[T \subseteq C_{\star\text{-bad}}^{X_t^i} \right] \leq \sum_{T \in \mathcal{T}_v^i} (2e\Delta^3)^{1-i}.$$

Combining with (33), we have

$$\Pr \left[\ell \geq ik\Delta^2 \right] \leq \frac{\Delta}{2} \cdot (e\Delta^3)^{i-1} \cdot (2e\Delta^3)^{1-i} \leq \Delta \cdot 2^{-i}.$$

□

6.5. Efficiency of MarginSample. We now prove the following upper bound on the expected running time of MarginSample (Algorithm 3), which is expressed in the form of (15).

Let $T_{\text{MS}}(\Phi, \sigma, v)$ be the random variable that represents the complexity of MarginSample(Φ, σ, v) when Condition 3.3 is satisfied by (Φ, σ, v) . Note that $\bar{t}_{\text{MS}}(\Phi, \sigma, v) = \mathbb{E}[T_{\text{MS}}(\Phi, \sigma, v)]$ by Definition 6.4.

Theorem 6.27. Assume $8\epsilon p\Delta^3 \leq 0.99p'$. Let X^0, X^1, \dots, X^n be the random sequence in Definition 5.6. Assume the convention that $T_{\text{MS}}(\Phi, X^{t-1}, v_t) = 0$ when v_t is X^{t-1} -fixed. For any $1 \leq t \leq n$,

$$\mathbb{E}[T_{\text{MS}}(\Phi, X^{t-1}, v_t)] \leq O\left(q^2 k^2 \Delta^{10}\right) \cdot \underline{x} + 480k\Delta^7 \cdot \underline{y} + O\left(q^3 k^3 \Delta^{10}\right),$$

where expectation is taken over both X^{t-1} and the randomness of MarginSample algorithm.

The convention is safe to apply since MarginSample(Φ, σ, v) is never called when v is not σ -fixed.

To prove this theorem, one need to give concrete bounds on $t_{\text{var}}(\sigma)$ and $\bar{t}_{\text{BF}}(\sigma)$ (Definition 6.5), respectively for the two nontrivial steps in the algorithm: the NextVar(σ) called at Line 1 and the Bernoulli factory called at Line 10, both in Algorithm 4.

6.5.1. Cost of NextVar(σ). We give an explicit bound on the complexity $t_{\text{var}}(\sigma)$ (Definition 6.5) for computing the NextVar(σ) in Line 1 of Algorithm 4, in terms of the size of $C_{\star\text{-con}}^\sigma$ (Definition 3.6).

Assume the input model and data structures in Section 6.1. We have the following result.

Proposition 6.28. For any $\sigma \in \mathcal{Q}^*$, NextVar(σ) can be computed using at most $|C_{\star\text{-con}}^\sigma|$ queries to Eval(\cdot), $\Delta |C_{\star\text{-con}}^\sigma|$ queries to Frozen(\cdot), and $O\left(k^2 \Delta |C_{\star\text{-con}}^\sigma|^2 + 1\right)$ computation cost, that is

$$t_{\text{var}}(\sigma) \leq |C_{\star\text{-con}}^\sigma| \cdot \underline{x} + \Delta |C_{\star\text{-con}}^\sigma| \cdot \underline{y} + O\left(k^2 \Delta |C_{\star\text{-con}}^\sigma|^2 + 1\right).$$

As assumed in Section 6.1, a linked list is kept alongside with the partial assignment σ for storing the variables set as \star in σ . We use $S_\star \triangleq \{v \in V \mid \sigma(v) = \star\}$ to denote such set of variables. Recall the simplification $\Phi^\sigma = (V^\sigma, Q^\sigma, C^\sigma)$ of Φ under σ and its corresponding hypergraph representation $H^\sigma = H_{\Phi^\sigma} = (V^\sigma, C^\sigma)$, which is formally defined in Section 3.2 and used in the definition of NextVar(σ).

The procedure for computing NextVar(σ) is straightforward on the hypergraph H^σ :

- Perform a depth-first search starting from S_\star on the sub-hypergraph of H^σ induced by $V^\sigma \cap V_{\text{fix}}^\sigma$ to find the connected components $V_\star^\sigma \supseteq S_\star$.
- Construct the vertex boundary of V_\star^σ in H^σ . Return the first boundary vertex v_i with smallest i if such vertex exists, and return \perp if otherwise.

We then verify that this procedure can be implemented within the complexity in Proposition 6.28.

First, observe that the complexity for testing whether a $c \in C$ belongs to the sub-hypergraph of H^σ induced by $V^\sigma \cap V_{\text{fix}}^\sigma$, is bounded by $\underline{x} + \Delta \cdot \underline{y} + O(k\Delta)$, i.e. one query to Eval(\cdot), Δ queries to Frozen(\cdot), and $O(k\Delta)$ computation cost. This is because it is equivalent to check whether c is satisfied by σ and $\text{vbl}(c) \subseteq V_{\text{fix}}^\sigma$: the former takes one query to Eval(\cdot); and the latter can be resolved by enumerating all $c' \in C$ such that $\text{vbl}(c) \cap \text{vbl}(c') \neq \emptyset$ and retrieving $\text{vbl}(c')$ (which costs $O(k\Delta)$ in computation), and checking whether c' is σ -frozen for all such c' (which takes $\leq \Delta$ queries to Frozen(\cdot) in total).

It is not difficult to verify that in above depth-first search, the set of constraints that need to be checked whether belong to the induced sub-hypergraph is in fact just the set $C_{\star\text{-con}}^\sigma$. Then the complexity contributed by testing the membership of constraints in the induced sub-hypergraph is bounded by $|C_{\star\text{-con}}^\sigma| \cdot \underline{x} + \Delta |C_{\star\text{-con}}^\sigma| \cdot \underline{y} + O(k\Delta |C_{\star\text{-con}}^\sigma|)$. And this is the only part that may access the oracles, so we have the respective bounds on the queries to the oracles Eval(\cdot) and Frozen(\cdot).

For other computation costs, in the depth-first search, the sets of variables and constraints that have been visited can be straightforwardly stored using two dynamic arrays, one for variables and the other for constraints. Querying if some variable/constraint has been visited or updating their status, is done by iterating over the entire array, which takes linear time in the current size of the dynamic array each time a query or an update is conducted. Note that the number of visited constraints is at most $|C_{\star\text{-con}}^\sigma|$ and hence the number of visited variables is at most $k |C_{\star\text{-con}}^\sigma|$. Therefore this part costs $O(k^2 |C_{\star\text{-con}}^\sigma|^2)$ in computation in total. Overall, the computation cost is easily dominated by $O(k^2 \Delta |C_{\star\text{-con}}^\sigma|^2 + 1)$, where the additional $O(1)$ is meant to deal with the degenerate case of $|C_{\star\text{-con}}^\sigma| = 0$.

6.5.2. *Cost of Bernoulli factory.* Here we state a complexity bound for the Bernoulli factory in Line 10 of Algorithm 4 that is useful in our analysis of MarginSample.

We have the following bound on the $\bar{t}_{\text{BF}}(\sigma)$ (Definition 6.5) for the Bernoulli factory.

Proposition 6.29. *There exist constants $C_0, C_1 > 0$ such that the following holds. Let $1 \leq t \leq n$ and let $\text{Path}(X_0^t) = (X_0^t, X_1^t, \dots, X_\ell^t)$ be generated as in (24), then*

$$\bar{t}_{\text{BF}}(X_\ell^t) \leq C_1 q^2 k^2 \Delta^6 \left(\left| C_{\star\text{-con}}^{X_\ell^t} \right| + 1 \right) (1 - \text{ep}'q)^{-\left| C_{\star\text{-con}}^{X_\ell^t} \right|} \cdot \underline{x} + C_0 q^3 k^3 \Delta^6 \left(\left| C_{\star\text{-con}}^{X_\ell^t} \right| + 1 \right) (1 - \text{ep}'q)^{-\left| C_{\star\text{-con}}^{X_\ell^t} \right|}.$$

Note that X_ℓ^t is a random variable. And the bound in Proposition 6.29 holds for any possible X_ℓ^t .

Next, we prove Proposition 6.29. In Theorem A.4, the following complexity bound is proved for all such $\sigma \in Q^*$ where Condition 3.8 is satisfied by (Φ, σ, v) for some $v \in V$:

$$(34) \quad \bar{t}_{\text{BF}}(\sigma) = O \left(q^2 k^2 \Delta^6 (|C_v^\sigma| + 1) (1 - \text{ep}'q)^{-|C_v^\sigma|} \cdot \underline{x} + q^3 k^3 \Delta^6 (|C_v^\sigma| + 1) (1 - \text{ep}'q)^{-|C_v^\sigma|} \right),$$

where recall that C_v^σ denotes the set of constraints in the connected component that contain v in Φ^σ , the simplification of Φ under σ .

The following lemma relates the sizes of $C_v^{\sigma_\ell}$, $C_{\star\text{-con}}^{\sigma_\ell}$, $C_{\star\text{-bad}}^{\sigma_\ell}$, where σ_ℓ is generated from the process $\text{Path}(\sigma) = (\sigma_0, \sigma_1, \dots, \sigma_\ell)$ defined in Definition 6.10. It is formally proved in Appendix B.2.

Lemma 6.30. *Let $\sigma \in Q^*$ be a partial assignment with exactly one variable $v \in V$ having $\sigma(v) = \star$, and let $\text{Path}(\sigma) = (\sigma_0, \sigma_1, \dots, \sigma_\ell)$. It always holds that $|C_v^{\sigma_\ell}| \leq |C_{\star\text{-con}}^{\sigma_\ell}| \leq \Delta \cdot |C_{\star\text{-bad}}^{\sigma_\ell}|$.*

Proof of Proposition 6.29. By the construction in (24), X_0^t satisfies one of the two cases:

- (1) (Φ, X_0^t, v_t) satisfies Condition 3.8 and v_t is the only variable with $X_0^t(v_t) = \star$;
- (2) $X_0^t(u) \in Q_u \cup \{\star\}$ for all $u \in V$.

For case (1): By Lemma 6.30, we have $|C_v^{X_\ell^t}| \leq |C_{\star\text{-con}}^{X_\ell^t}|$ because X_ℓ^t is generated by $\text{Path}(X_0^t) = (X_0^t, X_1^t, \dots, X_\ell^t)$ obeying the Markov property. Moreover, by Lemma 5.9 and the construction of Path , it holds for sure that Condition 3.8 is satisfied by (Φ, X_ℓ^t, u) for some $u \in V$. Therefore, the complexity bound in (34) always holds for X_ℓ^t , which combined with the relation $|C_v^{X_\ell^t}| \leq |C_{\star\text{-con}}^{X_\ell^t}|$ that we have just established, proves the case.

For case (2): In this case by the definition of Path , $\ell = 0$ and $\text{Path}(X_0^t) = (X_0^t)$. And Condition 3.8 is no longer satisfied by $(\Phi, X_\ell^t, v) = (\Phi, X_0^t, v)$ for any v . Thus, $\bar{t}_{\text{BF}}(X_\ell^t) = 0$ due to Definition 6.5. \square

By a similar argument, we also have the following corollary for Lemma 6.30.

Corollary 6.31. *Let $1 \leq t \leq n$ and let $\text{Path}(X_0^t) = (X_0^t, X_1^t, \dots, X_\ell^t)$ be generated as in (24), then*

$$\left| C_{\star\text{-con}}^{X_\ell^t} \right| \leq \Delta \cdot \left| C_{\star\text{-bad}}^{X_\ell^t} \right|.$$

Proof. By construction of X_0^t in (24), either (Φ, X_0^t, v_t) satisfies Condition 3.8 and v_t is the only variable with $X_0^t(v_t) = \star$, in which case the corollary follows from Lemma 6.30; or X_0^t satisfies for all $u \in V$, $X_0^t(u) \in Q_u \cup \{\star\}$, in which case by definition of Path , we have $\ell = 0$ and $\left| C_{\star\text{-con}}^{X_\ell^t} \right| = \left| C_{\star\text{-bad}}^{X_\ell^t} \right| = 0$. \square

6.5.3. *Complexity bound for MarginSample.* It remains to bound the complexity of MarginSample.

We need the following monotonicity of several properties along the $\text{Path}(\sigma) = (\sigma_0, \dots, \sigma_\ell)$. The proof is through routine verification of definitions, which is deferred to Appendix B.1.

Lemma 6.32. *Let $\sigma \in Q^*$ and $\text{Path}(\sigma) = (\sigma_0, \sigma_1, \dots, \sigma_\ell)$. For every $0 \leq i \leq j \leq \ell$, it holds that*

$$C_{\mathcal{P}}^{\sigma_i} \subseteq C_{\mathcal{P}}^{\sigma_j},$$

where \mathcal{P} can be any property $\mathcal{P} \in \{\star, \text{frozen}, \star\text{-con}, \star\text{-frozen}, \star\text{-bad}\}$.

Next, we give an upper bound on the weighted function H defined in (18) for a random path.

Lemma 6.33. Assume $8\epsilon p\Delta^3 \leq 0.99p'$, where p' is fixed as in (6). Let $1 \leq t \leq n$ and let $\text{Path}(X_0^t) = (X_0^t, X_1^t, \dots, X_\ell^t)$ be generated as in (24). There exist constants $C, C' > 0$ such that

$$\mathbb{E} [H(\text{Path}(X_0^t))] \leq Cq^2k^2\Delta^{10} \cdot \underline{x} + 480k\Delta^7 \cdot \underline{y} + C'q^3k^3\Delta^{10}.$$

Proof. We will prove that

$$(35) \quad \mathbb{E} \left[\sum_{i=0}^{\ell} ((2-q\theta)^i \cdot t_{\text{var}}(X_i^t)) \right] \leq 480k\Delta^6 \cdot \underline{x} + 480k\Delta^7 \cdot \underline{y} + 8080C_1k^3\Delta^9$$

and

$$(36) \quad \mathbb{E} [(2-q\theta)^\ell \cdot \bar{t}_{\text{BF}}(X_\ell^t)] \leq 100C_2k^2q^2\Delta^{10} \cdot \underline{x} + 100C_3k^3q^3\Delta^{10}.$$

Thus, by (18) we have

$$\begin{aligned} \mathbb{E} [H(\text{Path}(X^t))] &= \mathbb{E} \left[\sum_{i=0}^{\ell} ((2-q\theta)^i \cdot t_{\text{var}}(X_i^t)) + (2-q\theta)^\ell \cdot \bar{t}_{\text{BF}}(X_\ell^t) \right] \\ &= \mathbb{E} \left[\sum_{i=0}^{\ell} ((2-q\theta)^i \cdot t_{\text{var}}(X_i^t)) \right] + \mathbb{E} [(2-q\theta)^\ell \cdot \bar{t}_{\text{BF}}(X_\ell^t)] \\ &\leq 480k\Delta^6 \cdot \underline{x} + 480k\Delta^7 \cdot \underline{y} + 8080C_1k^3\Delta^9 + 100C_2k^2q^2\Delta^{10} \cdot \underline{x} + 100C_3k^3q^3\Delta^{10} \\ &\leq (100C_2 + 480)k^2q^2\Delta^{10} \cdot \underline{x} + 480k\Delta^7 \cdot \underline{y} + (8080C_1 + 100C_3)k^3q^3\Delta^{10} \end{aligned}$$

where the last inequality is by (35) and (36). The lemma is proved by taking $C = 100C_2 + 480$ and $C' = 8080C_1 + 100C_3$.

At first, we prove (35). Let $L \triangleq |C_{\star\text{-bad}}^{X_\ell^t}|$. By Lemma 6.32, $C_{\star\text{-con}}^{X_i^t} \subseteq C_{\star\text{-con}}^{X_\ell^t}$ for every $0 \leq i \leq \ell$. By Corollary 6.31,

$$(37) \quad |C_{\star\text{-con}}^{X_i^t}| \leq |C_{\star\text{-con}}^{X_\ell^t}| \leq \Delta |C_{\star\text{-bad}}^{X_\ell^t}| = \Delta L.$$

Then by Proposition 6.28, there exists a constant $C_1 > 0$ such that for every $0 \leq i \leq \ell$,

$$\begin{aligned} t_{\text{var}}(X_i^t) &\leq |C_{\star\text{-con}}^{X_i^t}| \cdot \underline{x} + \Delta |C_{\star\text{-con}}^{X_i^t}| \cdot \underline{y} + C_1 \cdot k^2\Delta |C_{\star\text{-con}}^{X_i^t}|^2 + C_1 \\ &\leq \Delta L \cdot \underline{x} + \Delta^2 L \cdot \underline{y} + C_1 \cdot k^2\Delta^3 L^2 + C_1. \end{aligned}$$

Thus, we have

$$\mathbb{E} \left[\sum_{i=0}^{\ell} ((2-q\theta)^i \cdot t_{\text{var}}(X_i^t)) \right] \leq \mathbb{E} \left[(\ell+1) \cdot (2-q\theta)^\ell \cdot (\Delta L \cdot \underline{x} + \Delta^2 L \cdot \underline{y} + C_1 \cdot k^2\Delta^3 L^2 + C_1) \right].$$

By Lemma 6.26 and the special case that $\ell = 0$ when X_0^t satisfies that $X_0^t(u) \in Q_u \cup \{\star\}$ for all $u \in V$, we have $\ell \leq kL\Delta$. Thus, the following can be verified in separate cases $\ell > 0$, $(\ell = 0) \wedge (L = 0)$, and $(\ell = 0) \wedge (L > 0)$:

$$\begin{aligned} &(\ell+1) \cdot (2-q\theta)^\ell \cdot (\Delta L \cdot \underline{x} + \Delta^2 L \cdot \underline{y} + C_1 \cdot k^2\Delta^3 L^2 + C_1) \\ &\leq 2kL\Delta \cdot (2-q\theta)^{kL\Delta} \cdot (\Delta L \cdot \underline{x} + \Delta^2 L \cdot \underline{y} + C_1 \cdot k^2\Delta^3 L^2 + C_1). \end{aligned}$$

Therefore, we have

$$\begin{aligned} (38) \quad &\mathbb{E} \left[\sum_{i=0}^{\ell} ((2-q\theta)^i \cdot t_{\text{var}}(X_i^t)) \right] \\ &\leq \mathbb{E} \left[2kL\Delta \cdot (2-q\theta)^{kL\Delta} \cdot (\Delta L \cdot \underline{x} + \Delta^2 L \cdot \underline{y} + C_1 \cdot k^2\Delta^3 L^2 + C_1) \right] \\ &\leq \mathbb{E} \left[(2-q\theta)^{kL\Delta} \cdot (L^2 \cdot 2k\Delta^2 \cdot \underline{x} + L^2 \cdot 2k\Delta^3 \cdot \underline{y} + L^3 \cdot 2C_1k^3\Delta^4 + L \cdot 2C_1k\Delta) \right]. \end{aligned}$$

By (6) and (7) we have $1 - q\theta \leq (4ek\Delta^3)^{-1}$. Thus $(2 - q\theta)^{k\Delta^2} \leq 1.5$ for all $\Delta \geq 2$. Let $j \triangleq \lfloor \frac{i}{\Delta} \rfloor$. Let $\alpha \in \{1, 2, 3\}$. We have:

$$\begin{aligned}
\mathbb{E} \left[L^\alpha \cdot (2 - q\theta)^{kL\Delta} \right] &= \sum_{i \geq 0} \left(\Pr[L = i] \cdot i^\alpha \cdot (2 - q\theta)^{ki\Delta} \right) \\
&\leq \sum_{i \geq 0} \left(\Pr[L = i] \cdot i^\alpha \cdot 1.5^{i/\Delta} \right) \\
(39) \quad &\leq 2\Delta^\alpha \sum_{i \geq 0} \left((j+1)^\alpha \cdot 1.5^j \cdot \Pr[L \geq j\Delta] \right) \\
&\leq 2\Delta^{\alpha+1} \sum_{j \geq 0} \left((j+1)^\alpha \cdot 1.5^j \cdot \Pr[L \geq j\Delta] \right)
\end{aligned}$$

By Theorem 6.21, Lemma 6.26 and the special case that $L = 0$ when X_0^t satisfies that $X_0^t(u) \in Q_u \cup \{\star\}$ for all $u \in V$, we have for every $i \geq 0$,

$$\Pr[L \geq i\Delta] \leq \Pr[\ell \geq ik\Delta^2] \leq \Delta \cdot 2^{-i}.$$

Combining with (39), we have

$$(40) \quad \mathbb{E} \left[L^\alpha \cdot (2 - q\theta)^{kL\Delta} \right] \leq 2\Delta^{\alpha+2} \sum_{i \geq 0} \left((i+1)^\alpha \cdot 0.75^i \right).$$

Note that $\sum_{i \geq 0} ((i+1) \cdot 0.75^i) \leq 20$, $\sum_{i \geq 0} ((i+1)^2 \cdot 0.75^i) \leq 120$ and $\sum_{i \geq 0} ((i+1)^3 \cdot 0.75^i) \leq 2000$. Combining with (40) and (38), (35) is immediate.

In the following, we prove (36). Then the lemma holds. By Proposition 6.29 and (37), there exist some constants $C_2, C_3 > 0$ such that

$$\begin{aligned}
\bar{t}_{\text{BF}}(X_\ell^t) &\leq k^2 q^2 \Delta^6 \left(\left| C_{\star\text{-con}}^{X_\ell^t} \right| + 1 \right) (1 - ep'q)^{-\left| C_{\star\text{-con}}^{X_\ell^t} \right|} (C_2 \cdot \underline{x} + C_3 kq) \\
&\leq k^2 q^2 \Delta^7 \cdot (L+1) (1 - ep'q)^{-L\Delta} \cdot (C_2 \cdot \underline{x} + C_3 kq).
\end{aligned}$$

Thus we have

$$\begin{aligned}
\mathbb{E} \left[(2 - q\theta)^\ell \cdot \bar{t}_{\text{BF}}(X_\ell^t) \right] &\leq \mathbb{E} \left[(2 - q\theta)^\ell k^2 q^2 \Delta^7 \cdot (L+1) (1 - ep'q)^{-L\Delta} \cdot (C_2 \cdot \underline{x} + C_3 kq) \right] \\
&= k^2 q^2 \Delta^7 \cdot \mathbb{E} \left[(2 - q\theta)^\ell (L+1) (1 - ep'q)^{-L\Delta} \right] \cdot (C_2 \cdot \underline{x} + C_3 kq).
\end{aligned}$$

Recall that $\ell \leq kL\Delta$. We have

$$(41) \quad \mathbb{E} \left[(2 - q\theta)^\ell \cdot \bar{t}_{\text{BF}}(X_\ell^t) \right] = k^2 q^2 \Delta^7 \cdot \mathbb{E} \left[(2 - q\theta)^{kL\Delta} (L+1) (1 - ep'q)^{-L\Delta} \right] \cdot (C_2 \cdot \underline{x} + C_3 kq).$$

By (6) and (7) we have $1 - q\theta \leq (4ek\Delta^3)^{-1}$ and $ep'q \leq (2\Delta^3)^{-1}$. Thus $(1 - ep'q)^{-\Delta^2} (2 - q\theta)^{k\Delta^2} \leq 1.5$ for each $\Delta \geq 2$. Let $j \triangleq \lfloor \frac{i}{\Delta} \rfloor$. We have

$$\begin{aligned}
\sum_{i \geq 0} \left(\Pr[L = i] \cdot (i+1) \cdot (1 - ep'q)^{-i\Delta} (2 - q\theta)^{ik\Delta} \right) &\leq \sum_{i \geq 0} \left((i+1) \cdot 1.5^{i/\Delta} \cdot \Pr[L \geq i] \right) \\
&\leq 2\Delta \sum_{i \geq 0} \left((j+1) \cdot 1.5^j \cdot \Pr[L \geq j\Delta] \right) \\
&\leq 2\Delta^2 \sum_{j \geq 0} \left((j+1) \cdot 1.5^j \cdot \Pr[L \geq j\Delta] \right).
\end{aligned}$$

Recall that $\Pr[L \geq i\Delta] \leq \Delta \cdot 2^{-i}$ for each $i \geq 0$. Thus, we have

$$(42) \quad \sum_{i \geq 0} \left(\Pr[L = i] \cdot (i+1) \cdot (1 - ep'q)^{-i\Delta} (2 - q\theta)^{ik\Delta} \right) \leq 2\Delta^3 \sum_{i \geq 0} \left((i+1) \cdot 0.75^i \right) \leq 100\Delta^3.$$

Combining with (41), we have

$$\begin{aligned}
\mathbb{E} \left[(2 - q\theta)^\ell \cdot \bar{t}_{\text{BF}}(X_\ell^t) \right] &\leq k^2 q^2 \Delta^7 \cdot 100\Delta^3 \cdot (C_2 \cdot \underline{x} + C_3 kq) \\
&\leq 100C_2 k^2 q^2 \Delta^{10} \cdot \underline{x} + 100C_3 k^3 q^3 \Delta^{10}.
\end{aligned}$$

Then (36) holds, which finishes the proof of the lemma. \square

Proof of Theorem 6.27. Let

$$U \triangleq \{\sigma \in \mathcal{Q}^* \mid \Pr[X^{t-1} = \sigma] > 0\},$$

$$S \triangleq \{\sigma \in \mathcal{Q}^* \mid \Pr[X^{t-1} = \sigma] > 0 \wedge v_t \notin V_{\text{fix}}^\sigma\}.$$

For any $\sigma \in S$, we have $\Pr[X^{t-1} = \sigma] > 0$ and $v_t \notin V_{\text{fix}}^\sigma$, therefore by the construction of X^0, X^1, \dots, X^n in Definition 5.6, there is a positive probability that $\text{MarginSample}(\Phi, \sigma, v_t)$ is called within Algorithm 1. Hence by Lemma 5.4, Condition 3.3 is satisfied by (Φ, σ, v_t) for every $\sigma \in S$.

For any $\sigma \in \mathcal{Q}^*$, let $\gamma(\sigma) = \mathbb{E}[H(\text{Path}(\sigma))]$ (0, 0) denote the constant term in $\mathbb{E}[H(\text{Path}(\sigma))]$. By Corollary 6.12, there exists a constant $C_1 > 0$ such that for every $\sigma \in S$,

$$(43) \quad \bar{t}_{\text{MS}}(\Phi, \sigma, v_t) \leq \mathbb{E}[H(\text{Path}(\sigma_{v_t \leftarrow \star}))] + C_1 \cdot \gamma(\sigma_{v_t \leftarrow \star}).$$

And for every $\sigma \in U \setminus S$, v_t is σ -fixed, and hence $\bar{t}_{\text{MS}}(\Phi, \sigma, v_t) = \mathbb{E}[T_{\text{MS}}(\Phi, \sigma, v_t)] = 0$ by convention. On the other hand, $\mathbb{E}[H(\text{Path}(\sigma))]$ is always nonnegative. Thus the following holds trivially:

$$(44) \quad \bar{t}_{\text{MS}}(\Phi, \sigma, v_t) \leq \mathbb{E}[H(\text{Path}(\sigma))] + C_1 \cdot \gamma(\sigma).$$

Define

$$X_0^t \triangleq \begin{cases} X^{t-1} & \text{if } v_t \text{ is } X^{t-1}\text{-fixed,} \\ X_{v_t \leftarrow \star}^{t-1} & \text{otherwise.} \end{cases}$$

The above (43) and (44) together show that conditional on any possible $X^{t-1} = \sigma \in U$, the following always holds:

$$\bar{t}_{\text{MS}}(\Phi, X^{t-1}, v_t) \leq \mathbb{E}[H(\text{Path}(X_0^t))] + C_1 \cdot \gamma(X_0^t).$$

Hence by total expectation, we have

$$\mathbb{E}[T_{\text{MS}}(\Phi, X^{t-1}, v_t)] = \mathbb{E}[\bar{t}_{\text{MS}}(\Phi, X^{t-1}, v_t)] \leq \mathbb{E}[H(\text{Path}(X_0^t))] + C_1 \cdot \mathbb{E}[\gamma(X_0^t)].$$

By Lemma 6.33, for such X_0^t constructed as above, there exist constants $C_2, C_3 > 0$ such that

$$\mathbb{E}[H(\text{Path}(X_0^t))] \leq C_2 q^2 k^2 \Delta^{10} \cdot \underline{x} + 480 k \Delta^7 \cdot \underline{y} + C_3 q^3 k^3 \Delta^{10},$$

which means that

$$\mathbb{E}[T_{\text{MS}}(\Phi, X^{t-1}, v_t)] \leq C_2 q^2 k^2 \Delta^{10} \cdot \underline{x} + 480 k \Delta^7 \cdot \underline{y} + (C_1 + 1) C_3 \cdot q^3 k^3 \Delta^{10}. \quad \square$$

6.6. Efficiency of RejectionSampling. Recall the random sequence of partial assignments:

$$X^0, X^1, \dots, X^n,$$

maintained in Algorithm 1, as formally defined in Definition 5.6.

Here, we focus on $X = X^n$, the partial assignment obtained after all n iterations of the **for** loop in Line 2 of Algorithm 1, and passed to the RejectionSampling (Algorithm 2) as input.

For partial assignment $\sigma \in \mathcal{Q}^*$, we let $T_{\text{Rej}}(\sigma)$ be the random variable that represents the complexity of RejectionSampling($\Phi, \sigma, V \setminus \Lambda(\sigma)$), expressed in form of (15). We prove the following bound on the expectation of $T_{\text{Rej}}(X)$ on the random partial assignment $X = X^n$.

Theorem 6.34. Assume $8\text{ep}\Delta^3 \leq 0.99p'$, where p' is fixed as in (6).

$$\mathbb{E}[T_{\text{Rej}}(X)] \leq 10\Delta^5 n \cdot \underline{x} + O(qk\Delta^5 n),$$

where expectation is taken over both X and the randomness of RejectionSampling algorithm.

Recall the following definitions in Section 3.2. For each $\sigma \in \mathcal{Q}^*$ and $v \in V^\sigma$, $H_v^\sigma = (V_v^\sigma, C_v^\sigma)$ denotes the connected component in H^σ that contains the vertex/variable v , where H^σ is the hypergraph representation for the CSP formula Φ^σ obtained from the simplification of Φ under σ . We further stipulate that $H_v^\sigma = (V_v^\sigma, C_v^\sigma) = (\emptyset, \emptyset)$ is the empty hypergraph when $v \in \Lambda(\sigma)$ is assigned in σ .

Recall the definitions of $\{2, 3\}$ -tree in Definition 6.22 and dependency graph $G(C)$ in Definition 6.23. The following lemma is an analogy of Lemma 6.26, the proof of which is deferred to Appendix C.2.

Lemma 6.35. For every $v \in V$, there exists a $\{2, 3\}$ -tree T in $G(C)$ with $v \in \text{vbl}(T)$ such that

$$T \subseteq C_{\text{frozen}}^X \quad \text{and} \quad \Delta^2 |T| \geq |C_v^X|.$$

Proof of Theorem 6.34. Let $\{H_i^X = (V_i^X, C_i^X)\} \mid 1 \leq i \leq K$ be the connected components in H^X constructed in Line 1 Algorithm 2. For each $1 \leq i \leq K$, let $T(H_i^X)$ denote the cost contributed by $H_i^X = (V_i^X, C_i^X)$ during the repeat loop in Algorithm 2. Then the total cost is given by

$$(45) \quad T_{\text{Rej}}(X) \leq \Delta n \cdot \underline{x} + O(\Delta n) + \sum_{i \in [K]} T(H_i^X).$$

This is because:

- the cost of Line 1 is at most $n\Delta \cdot \underline{x} + O(n\Delta)$, because it uses at most $|C| \leq \Delta n$ queries to $\text{Eval}(\cdot)$, which contributes the $\Delta n \cdot \underline{x}$ term, and a depth-first search that visits all variables and constraints to compute $H_1^X, H_2^X, \dots, H_K^X$, which costs $O(\Delta n)$ in computation.
- the cost of Lines 2-5 for each component H_i^X is $T(H_i^X)$.

Alternatively, for each $v \in V$, we use $T(H_v^X)$ to denote the $T(H_i^X)$ for the $1 \leq i \leq K$ with $v \in V_i^X$; and let $T(H_v^X) = 0$ if there is no such $1 \leq i \leq K$, which occurs when $v \in \Lambda(X)$ is assigned in X . Clearly,

$$\sum_{i \in [K]} T(H_i^X) \leq \sum_{v \in V} T(H_v^X).$$

Combining with (45), we have

$$(46) \quad \mathbb{E}[T_{\text{Rej}}(X)] \leq n\Delta \cdot \underline{x} + O(n\Delta) + \sum_{v \in V} \mathbb{E}[T(H_v^X)].$$

We claim that for each $v \in V$,

$$(47) \quad \mathbb{E}[T(H_v^X) \mid X] \leq |C_v^X| \cdot (1 - ep'q)^{-|C_v^X|} \cdot \underline{x} + O(kq \cdot (|C_v^X| + 1) \cdot (1 - ep'q)^{-|C_v^X|}).$$

The degenerate case with $H_v^X = (\emptyset, \emptyset)$ is trivial. We then assume $H_v^X \neq (\emptyset, \emptyset)$. It is well known that the expected number of trials (the iterations of the **repeat** loop in Line 5) taken by the rejection sampling until success is given by $\mathbb{P}_{H_v^X}[\Omega_{H_v^X}]^{-1}$, where $\Omega_{H_v^X}$ denotes the set of satisfying assignments of the CSP of H_v^X and hence $\mathbb{P}_{H_v^X}[\Omega_{H_v^X}]$ gives the probability that a uniform random assignment $\sigma \in Q_{V_v^X}$ is satisfying for Φ_v^X , the CSP formula correspond to the component H_v^X .

By Theorem 4.1 and Lemma 5.8, we have

$$\mathbb{P}_{H_v^X}[\Omega_{H_v^X}] \geq (1 - ep'q)^{|C_v^X|}.$$

Therefore, it takes $(1 - ep'q)^{-|C_v^X|}$ iterations in expectation to successfully sample the assignment on V_v^X . And within each iteration, it is easy to verify that at most $|C_v^X|$ queries to $\text{Eval}(\cdot)$ and $O(k|C_v^X| + q|V_v^X|) = O(k(|C_v^X| + 1))$ computation cost are spent. This proves the claim (47).

We then bound the expectation of (47) over X . This is done by proving a tail bound on $|C_v^X|$.

For integer $t > 0$ and $v \in V$, define

$$\mathcal{T}_v^t \triangleq \{\{2, 3\}\text{-tree } T \text{ in } G(C) \mid v \in \text{vbl}(T) \wedge |T| = t\}.$$

Fix any $v \in V$. By Lemma 6.25, for every $c \in C$ with $v \in \text{vbl}(c)$, the number of $\{2, 3\}$ -trees in $G(C)$ of size t containing c is at most $\frac{(e\Delta^3)^{t-1}}{2}$. Combining with that $|\{c \in C \mid v \in \text{vbl}(c)\}| \leq \Delta$, we have

$$(48) \quad |\mathcal{T}_v^t| \leq \frac{\Delta}{2} \cdot (e\Delta^3)^{t-1}.$$

By Lemma 6.35, we have

$$(49) \quad \Pr[|C_v^X| \geq t\Delta^2] \leq \sum_{T \in \mathcal{T}_v^t} \Pr[T \subseteq C_{\text{frozen}}^X].$$

In addition, for each $\{2, 3\}$ -tree $T \in \mathcal{T}_v^t$, by Lemma 6.17 we have

$$\Pr[T \subseteq C_{\text{frozen}}^X] \leq (4e\Delta^3)^{-t}.$$

Combining with (48) and (49) we have that for every integer $t > 0$,

$$(50) \quad \Pr [|C_v^X| \geq t\Delta^2] \leq \sum_{T \in \mathcal{T}_v^t} (4e\Delta^3)^{-t} \leq \frac{\Delta}{2} \cdot (e\Delta^3)^{t-1} \cdot (4e\Delta^3)^{-t} \leq \Delta \cdot 4^{-t}.$$

Note that (50) holds trivially when $t = 0$. By (6) and (7), we have $ep'q \leq (2\Delta^3)^{-1}$ and $(1 - ep'q)^{-\Delta^2} \leq 2$ for all $\Delta \geq 2$. Let $\ell \triangleq \lfloor \frac{t}{\Delta^2} \rfloor$. We have

$$\begin{aligned} \sum_{t \geq 0} \left((t+1)(1 - ep'q)^{-t} \Pr [|C_v^X| = t] \right) &\leq \Delta^2 \sum_{t \geq 0} \left(\Pr [|C_v^X| \geq \ell\Delta^2] \cdot (\ell+1) \cdot (1 - ep'q)^{-(\ell+1)\Delta^2} \right) \\ &\leq 2\Delta^2 \sum_{t \geq 0} \left((\ell+1) \cdot 2^\ell \cdot \Pr [|C_v^X| \geq \ell\Delta^2] \right) \\ &= 2\Delta^4 \sum_{t \geq 0} \left((\ell+1) \cdot 2^\ell \cdot \Pr [|C_v^X| \geq \ell\Delta^2] \right) \\ (\text{by (50)}) \quad &\leq 2\Delta^5 \sum_{\ell \geq 0} ((\ell+1) \cdot 2^{-\ell}) \\ &\leq 8\Delta^5. \end{aligned}$$

Therefore,

$$\mathbb{E} \left[\left(|C_v^X| + 1 \right) \cdot (1 - ep'q)^{-|C_v^X|} \right] = \sum_{t \geq 0} \left((t+1)(1 - ep'q)^{-t} \Pr [|C_v^X| = t] \right) \leq 8\Delta^5.$$

Thus, there exists a constant $C > 0$ such that the expectation of (47) is bounded by

$$\mathbb{E} [T(H_v^X)] \leq 8\Delta^5 \cdot \underline{x} + 8Ckq\Delta^5.$$

The lemma follows by (46). \square

6.7. Efficiency of the main sampling algorithm. We now prove Theorem 6.1 and Theorem 6.3.

Proof of Theorem 6.1. Assuming the LLL condition in Equation (3), we have $8ep\Delta^3 \leq 0.99p'$ for p' set as in (6), which is the regime of parameters assumed by Theorem 6.27 and Theorem 6.34.

The followings are the nontrivial costs in Algorithm 1:

- The initialization in Line 1 and the testing of being X^{t-1} -fixed for v_t in Line 3 for every $1 \leq t \leq n$, which altogether cost:

$$\Delta n \cdot \underline{y} + O(\Delta n),$$

because each v_t queries $\text{Frozen}(\cdot)$ for at most $|c \in C \mid v_t \in \text{vbl}(c)| \leq \Delta$ times and also costs $O(\Delta)$ in computation to retrieve all such $c \in C$ that $v_t \in \text{vbl}(c)$.

- The calls to $\text{MarginSample}(\Phi, X^{t-1}, v_t)$ at Line 4, which according to Theorem 6.27, costs in total:

$$\sum_{t=1}^n \mathbb{E} [T_{\text{MS}}(\Phi, X^{t-1}, v_t)] \leq O\left(q^2 k^2 \Delta^{10} n\right) \cdot \underline{x} + 480k\Delta^7 n \cdot \underline{y} + O\left(q^3 k^3 \Delta^{10} n\right).$$

- The final call to $\text{RejectionSampling}(\Phi, X^n, V \setminus \Lambda(X^n))$ at Line 5, which by Theorem 6.34, costs:

$$\mathbb{E} [T_{\text{Rej}}(X^n)] \leq 10\Delta^5 n \cdot \underline{x} + O\left(qk\Delta^5 n\right).$$

The overall complexity of Algorithm 1 in expectation is bounded by

$$(51) \quad O\left(q^2 k^2 \Delta^{10} n\right) \cdot \underline{x} + 481k\Delta^7 n \cdot \underline{y} + O\left(q^3 k^3 \Delta^{10} n\right).$$

The theorem is proved. \square

Next, we construct the Algorithm 1' for approximate sampling and prove Theorem 6.3.

Given as input a CSP formula $\Phi = (V, Q, C)$ with $n = |V|$ variables and an error bound $\varepsilon \in (0, 1)$, Algorithm 1' does the followings. Set the parameters as:

$$(52) \quad \delta = 0.005 \quad \text{and} \quad N = \left\lceil \frac{\ln(4 \times 10^3 k \Delta^7 n \varepsilon^{-2})}{0.33 p' \delta^2} \right\rceil.$$

Algorithm 1' simply executes Algorithm 1 on input Φ , with the oracle $\text{Frozen}(\cdot)$ replaced by the following explicitly implemented Monte Carlo subroutine:

- Given as input any constraint $c \in C$ and any partial assignment $\sigma \in Q^*$, repeat for N times:
 - generate an assignment $Y \in Q_{\text{vbl}(c)}$ on $\text{vbl}(c)$ uniformly at random consistent with σ ;
 - check whether $c(Y) = \text{True}$ by querying $\text{Eval}(c, Y)$;
- let Z be the number of times within N trials that $c(Y) = \text{True}$, and return $I[Z/N > 0.995p']$.

We further apply a standard memoization trick to guarantee the consistency of the oracle $\text{Frozen}(\cdot)$ as required in Assumption 2. Each constraint $c \in C$ is associated with a deterministic dynamic dictionary Dic_c which stores key-value pairs in the form of (τ, ans) with $\tau \in \bigotimes_{v \in \text{vbl}(c)} (Q_v \cup \{\star, \star^*\})$ and $\text{ans} \in \{0, 1\}$. Upon each query to $\text{Frozen}(c, \sigma)$, we first lookup in Dic_c for the key $\sigma_{\text{vbl}(c)}$. If an $\text{ans} \in \{0, 1\}$ is retrieved, the query to $\text{Frozen}(c, \sigma)$ is answered with ans ; and if otherwise, an $\text{ans} = I[Z/N > 0.995p'] \in \{0, 1\}$ is computed using the above Monte Carlo subroutine, the key-value pair $(\sigma_{\text{vbl}(c)}, \text{ans})$ is inserted into Dic_c and the query to $\text{Frozen}(c, \sigma)$ is answered with ans . Using a *Trie* data structure for the deterministic dynamic dictionary Dic_c incurs an $O(k)$ computation cost for each query and update. Overall, the resulting algorithm is Algorithm 1'.

Proof of Theorem 6.3. In Algorithm 1', each query to the oracle $\text{Frozen}(\cdot)$ made in Algorithm 1 is implemented using N queries to the evaluation oracle $\text{Eval}(\cdot)$ along with $O(kN)$ computation cost, where N is set as in (52). By Theorem 6.1, the complexity of Algorithm 1 in expectation is bounded as (51). By replacing

$$\underline{y} \leftarrow N \cdot \underline{x} + O(kN),$$

the expected complexity of Algorithm 1' is bounded by

$$O\left(q^2 k^2 \Delta^{11} n \log\left(\frac{\Delta n}{\varepsilon}\right) \cdot \underline{x} + q^3 k^3 \Delta^{11} n \log\left(\frac{\Delta n}{\varepsilon}\right)\right).$$

By Chernoff bound, one can verify that for each query to $\text{Frozen}(c, \sigma)$, the two cases $\mathbb{P}[\neg c \mid \sigma] > p'$ and $\mathbb{P}[\neg c \mid \sigma] < 0.99p'$ are distinguished correctly except for an error probability bounded by:

$$2 \exp\left(-\frac{\delta^2}{3} 0.99p' N\right) \leq \frac{\varepsilon}{2M} \quad \text{where } M \triangleq 10^3 k \Delta^7 n \varepsilon^{-1}.$$

Due to (51), the expected number of queries to $\text{Frozen}(\cdot)$ made in Algorithm 1 is at most $481k\Delta^7 n < \frac{\varepsilon M}{2}$. By Markov's inequality, the probability that the number of queries to $\text{Frozen}(\cdot)$ made in Algorithm 1 exceeds M is at most $\frac{\varepsilon}{2}$. By a union bound, with probability at least $1 - \varepsilon$, no error occurs in any query to $\text{Frozen}(\cdot)$. Then by a coupling between Algorithm 1 and Algorithm 1', the total variation distance between the outputs of the two algorithms on the same input CSP Φ can be bounded within ε . \square

The following is a formal restatement of Theorem 1.5 and Theorem 1.6. Here the $\tilde{O}(\cdot)$ hides poly-logarithmic factors. The algorithm is just the Monte Carlo method that uses $\text{MarginSample}(\Phi, \star^V, v)$ (Algorithm 3) as a subroutine for sampling from the marginal distribution μ_v .

Theorem 6.36. *There are algorithms for marginal sampling and probabilistic inference such that given as input $\varepsilon, \delta \in (0, 1)$, CSP formula $\Phi = (V, Q, C)$ satisfying (3), and $v \in V$, the algorithms perform as follows:*

- *The algorithm for marginal sampling returns a random value $x \in Q_v$ distributed approximately as μ_v within total variation distance ε , in expectation using at most $O(q^2 k^2 \Delta^{11} \log(k\Delta/\varepsilon))$ queries to $\text{Eval}(\cdot)$ and $O(q^3 k^3 \Delta^{11} \log(k\Delta/\varepsilon))$ computation cost.*

- The algorithm for inference returns a $\hat{\mu}_v \in [0, 1]^{Q_v}$ using at most $\tilde{O}(q^3 k^2 \Delta^{11} \varepsilon^{-2} \log(1/\delta))$ queries to $\text{Eval}(\cdot)$ and $\tilde{O}(q^4 k^3 \Delta^{11} \varepsilon^{-2} \log(1/\delta))$ computation cost such that

$$\Pr [\forall x \in Q_v : (1 - \varepsilon)\mu_v(x) \leq \hat{\mu}_v(x) \leq (1 + \varepsilon)\mu_v(x)] \geq 1 - \delta.$$

Proof. When (3) is satisfied, we have $8ep\Delta^3 \leq 0.99p'$ for the p' set as in (6). Then Condition 3.3 is satisfied by (Φ, \star^V, v) . According to Theorem 5.5 and Theorem 6.27, $\text{MarginSample}(\Phi, \star^V, v)$ returns a perfect sample from μ_v with expected cost:

$$O(q^2 k^2 \Delta^{10}) \cdot \underline{x} + 480k\Delta^7 \cdot \underline{y} + O(q^3 k^3 \Delta^{10}).$$

Using the same Monte Carlo simulation of $\text{Frozen}(\cdot)$ as in the proof of Theorem 6.3, but this time with sufficiently large parameter $N = O\left(\frac{1}{p'} \log(k\Delta/\varepsilon)\right) = O(q^2 k \Delta^4 \log(k\Delta/\varepsilon))$, with the substitution $\underline{y} \leftarrow N \cdot \underline{x} + O(kN)$, the $\text{MarginSample}(\Phi, \star^V, v)$ is transformed into an approximate sampler for μ_v with bias at most $\varepsilon/2$ in total variation distance, with the following cost:

$$O(q^2 k^2 \Delta^{11} \log(k\Delta/\varepsilon)) \cdot \underline{x} + O(q^3 k^3 \Delta^{11} \log(k\Delta/\varepsilon)).$$

This proves the marginal sampler part of the algorithm.

By the local uniformity property stated in Corollary 4.3, assuming (3), we have $\mu_v(x) > \frac{1}{2p}$. Thus, by Chernoff bound, each $\mu_v(x)$ for $x \in Q_v$ can be estimated within $(1 \pm \varepsilon)$ -multiplicative precision with probability > 0.9 using $O(q/\varepsilon^2)$ approximate samples, which in expectation costs in total:

$$(53) \quad \tilde{O}(q^3 k^2 \Delta^{11} \varepsilon^{-2}) \cdot \underline{x} + \tilde{O}(q^4 k^3 \Delta^{11} \varepsilon^{-2}).$$

By Markov's inequality, the total cost exceeds this bound with probability < 0.1 . By truncation, this gives us an algorithm with fixed cost bounded as (53) so for each $x \in Q_v$, with probability > 0.8 the algorithm estimates the value of $\mu_v(x)$ within $(1 \pm \varepsilon)$ -multiplicative precision. The algorithm asserted by the theorem is then given by repeating this for $O(\log \frac{2}{\delta})$ times and applying the median trick. \square

7. CONCLUSION AND OPEN PROBLEMS

We give an algorithm for sampling uniform solutions to general constraint satisfaction problems (CSPs) in a local lemma regime. The algorithm runs in an expected near-linear time in the number of variables and polynomial in other local parameters, including: domain size q , width k and degree Δ .

This gives, for the first time, a near-linear time sampling algorithm for general CSPs with constant q, k, Δ , in a local lemma regime; and this also gives, for the first time, a polynomial-time sampling algorithm for general CSPs in a local lemma regime without assuming any degree or width bound.

A crucial step of our sampling algorithm, is a marginal sampler that can draw values of a variable according to the correct marginal distribution. Within a local lemma regime, this marginal sampler is a local algorithm whose cost is independent of the size of the CSP, and is polynomial in the local parameters q, k and Δ . This marginal sampler proves a thought-provoking point: *within a local lemma regime, a locally defined sampling or inference problem can be solved at a local cost.*

There are several open problems:

- An open question is to improve the current LLL condition $p\Delta^7 \lesssim 1$ for sampling general CSPs closer the lower bound $p\Delta^2 \gtrsim 1$. We also believe that removing the extra q, k factors in the current LLL condition may help us better understand the nature of sampling LLL.
- Another fundamental question is to generalize the current bound for CSPs to a general sampling Lovász local lemma with non-uniform distributions and/or asymmetric criteria.
- The recursive marginal sampler of Anand and Jerrum [AJ21] is a refreshingly novel idea for sampling. Here we see that its can solve an otherwise difficult to solve problem. It would be exciting to see what more this new idea can bring to the study of sampling LLL.
- Despite the current technical barrier, Markov chain based algorithms have several advantages, such as their efficient parallelization [LY21]. Therefore, it is still very worthwhile to have Markov chain based algorithms for sampling general CSPs. For this to work, we may have

to develop a way for dynamic projection of solution space, which may be of independent interest.

ACKNOWLEDGEMENT

We thank Weiming Feng and Kewen Wu for helpful discussions. Kun He wants to thank Xiaoming Sun for his support in doing this work. Yitong Yin wants to thank Vishesh Jain for pointing to the notion of robust CSPs.

REFERENCES

- [AJ21] Konrad Anand and Mark Jerrum. Perfect sampling in infinite spin systems via strong spatial mixing. *CoRR*, abs/2106.15992, 2021.
- [Alo91] Noga Alon. A parallel algorithmic version of the local lemma. *Random Struct. Algorithms*, 2(4):367–378, 1991. (Conference version in *FOCS’91*).
- [Bec91] József Beck. An algorithmic approach to the Lovász local lemma. *Random Struct. Algorithms*, 2(4):343–365, 1991.
- [BGG⁺19] Ivona Bezáková, Andreas Galanis, Leslie A. Goldberg, Heng Guo, and Daniel Štefankovič. Approximation via correlation decay when strong spatial mixing fails. *SIAM J. Comput.*, 48(2):279–349, 2019.
- [DHKN17] Shaddin Dughmi, Jason D. Hartline, Robert Kleinberg, and Rad Niazadeh. Bernoulli factories and black-box reductions in mechanism design. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 158–169. ACM, New York, 2017.
- [EL75] Paul Erdős and László Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. *Infinite and finite sets, volume 10 of Colloquia Mathematica Societatis János Bolyai*, pages 609–628, 1975.
- [FGW22] Weiming Feng, Heng Guo, and Jiaheng Wang. Improved bounds for randomly colouring simple hypergraphs. *arXiv preprint arXiv:2202.05554*, 2022.
- [FGY22] Weiming Feng, Heng Guo, and Yitong Yin. Perfect sampling from spatial mixing. *Random Structures and Algorithms*, 2022.
- [FGYZ21] Weiming Feng, Heng Guo, Yitong Yin, and Chihao Zhang. Fast sampling and counting k -sat solutions in the local lemma regime. *Journal of the ACM (JACM)*, 68(6):1–42, 2021.
- [FH00] James A. Fill and Mark Huber. The randomness recycler: a new technique for perfect sampling. In *FOCS*, pages 503–511. IEEE, 2000.
- [FHY21] Weiming Feng, Kun He, and Yitong Yin. Sampling constraint satisfaction solutions in the local lemma regime. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1565–1578, 2021.
- [FVY19] Weiming Feng, Nisheeth K. Vishnoi, and Yitong Yin. Dynamic sampling from graphical models. In *STOC*, pages 1070–1081. ACM, 2019.
- [GGGY20] Andreas Galanis, Leslie Ann Goldberg, Heng Guo, and Kuan Yang. Counting solutions to random CNF formulas. In *ICALP*, volume 168 of *LIPIcs*, pages 53:1–53:14, 2020.
- [GGW21] Andreas Galanis, Heng Guo, and Jiaheng Wang. Inapproximability of counting hypergraph colourings. *arXiv preprint arXiv:2107.05486*, 2021.
- [GH20] Heng Guo and Kun He. Tight bounds for popping algorithms. *Random Structures Algorithms*, 57(2):371–392, 2020.
- [GJ19] Heng Guo and Mark Jerrum. A polynomial-time approximation algorithm for all-terminal network reliability. *SIAM J. Comput.*, 48(3):964–978, 2019.
- [GJL19] Heng Guo, Mark Jerrum, and Jingcheng Liu. Uniform sampling through the Lovász local lemma. *J. ACM*, 66(3):Art. 18, 31, 2019.
- [GLLZ19] Heng Guo, Chao Liao, Pinyan Lu, and Chihao Zhang. Counting hypergraph colorings in the local lemma regime. *SIAM Journal on Computing*, 48(4):1397–1424, 2019.
- [Har20] David G Harris. New bounds for the moser-tardos distribution. *Random Structures & Algorithms*, 57(1):97–131, 2020.

- [HSS11] Bernhard Haeupler, Barna Saha, and Aravind Srinivasan. New constructive aspects of the Lovász local lemma. *J. ACM*, 58(6):28, 2011. (Conference version in *FOCS'10*).
- [HSW21] Kun He, Xiaoming Sun, and Kewen Wu. Perfect sampling for (atomic) lovász local lemma. *CoRR*, abs/2107.03932, 2021.
- [Hub16] Mark Huber. Nearly optimal Bernoulli factories for linear functions. *Combin. Probab. Comput.*, 25(4):577–591, 2016.
- [HV15] Nicholas J. A. Harvey and Jan Vondrák. An algorithmic proof of the Lovász local lemma via resampling oracles. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 1327–1345. IEEE Computer Soc., Los Alamitos, CA, 2015.
- [Jer21] Mark Jerrum. Fundamentals of partial rejection sampling. *arXiv preprint arXiv:2106.07744*, 2021.
- [JPV21a] Vishesh Jain, Huy Tuan Pham, and Thuy Duong Vuong. On the sampling lovász local lemma for atomic constraint satisfaction problems. *CoRR*, abs/2102.08342, 2021.
- [JPV21b] Vishesh Jain, Huy Tuan Pham, and Thuy Duong Vuong. Towards the sampling lovász local lemma. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 173–183. IEEE, 2021.
- [LY21] Hongyang Liu and Yitong Yin. Simple parallel algorithms for single-site dynamics. *arXiv preprint arXiv:2111.04044*. To appear in *STOC'22*, 2021.
- [Moi19] Ankur Moitra. Approximate counting, the Lovász local lemma, and inference in graphical models. *J. ACM*, 66(2):10:1–10:25, 2019. (Conference version in *STOC'17*).
- [MT10] Robin A. Moser and Gábor Tardos. A constructive proof of the general Lovász local lemma. *J. ACM*, 57(2):11, 2010.
- [NP05] Șerban Nacu and Yuval Peres. Fast simulation of new coins from old. *Ann. Appl. Probab.*, 15(1A):93–115, 2005.
- [She85] James B. Shearer. On a problem of Spencer. *Combinatorica*, 5(3):241–245, 1985.
- [vN51] John von Neumann. Various techniques used in connection with random digits. In A. S. Householder, G. E. Forsythe, and H. H. Germond, editors, *Monte Carlo Method*, volume 12 of *National Bureau of Standards Applied Mathematics Series*, chapter 13, pages 36–38. US Government Printing Office, Washington, DC, 1951.

APPENDIX A. A BERNOULLI FACTORY FOR MARGIN OVERFLOW

We present a generic solution to the following problem. Let (Φ, σ, v) be the input to Algorithm 4, where $\Phi = (V, Q, C)$ is a CSP formula, $\sigma \in Q^*$ is a feasible partial assignment, and $v \in V$ is a variable. Assume Condition 3.8 for the (Φ, σ, v) . The marginal distribution μ_v^σ over domain Q_v is well-defined, and by Corollary 4.4, the following is satisfied for the parameters $\theta_v > 0$ and $\zeta > 0$ fixed as in (7):

$$(54) \quad \min_{x \in Q_v} \mu_v^\sigma(x) \geq \theta_v + \zeta.$$

Therefore, the distribution \mathcal{D} in (8) is well-defined. We reiterate its definition here:

$$(55) \quad \forall x \in Q_v, \quad \mathcal{D}(x) \triangleq \frac{\mu_v^\sigma(x) - \theta_v}{1 - q_v \cdot \theta_v}, \quad \text{where } q_v \triangleq |Q_v|.$$

Our goal is to sample from this distribution by accessing an oracle for drawing independent samples from μ_v^σ . Such an oracle for μ_v^σ is realized by `RejectionSampling`($\Phi, \sigma, \{v\}$) defined in Algorithm 2.

Such a problem of simulating a new coin by making black-box accesses to an old coin, while the distribution of the new coin is a function of the old, is known as the Bernoulli factory problem [vN51].

A.1. Construction and correctness of the Bernoulli factory. For $\xi \in [0, 1]$, we denote by O_ξ a coin with probability of heads ξ . Formally, O_ξ is an oracle that, upon each call, independently returns 1 with probability ξ and 0 with probability $1 - \xi$.

We write $\nu = \mu_v^\sigma$ for short. We construct the following two types of basic oracles:

- for each $x \in Q_v$, an $O_{\nu(x)}$ is constructed as $O_{\nu(x)} = \mathbb{1} [\text{RejectionSampling}(\Phi, \sigma, \{v\}) = x]$;
- an O_{θ_v} is constructed as $O_{\theta_v} = \mathbb{1} [r < \theta_v]$ for $r \in [0, 1]$ chosen uniformly at random.

For each $x \in Q_v$, we apply the *Bernoulli factory for subtraction* in [NP05], denoted by `SubtractBF`, such that it constructs a new coin $O_{\nu(x) - \theta_v} = \text{SubtractBF}(O_{\nu(x)}, O_{\theta_v}, \zeta)$ with probability of heads $\nu(x) - \theta_v$.

We then apply the *Bernoulli race* in [DHKN17], denoted by `BernoulliRace`, such that the subroutine `BernoulliRace`($\{O_{\nu(x) - \theta_v}\}_{x \in Q_v}$) returns a random value $I \in Q_v$ satisfying that $I = x$ with probability proportional to $\nu(x) - \theta_v = \mu_v^\sigma(x) - \theta_v$, i.e. I is distributed as \mathcal{D} defined in (55). This achieves our goal.

Although these constructions are not new, for rigorousness, we restate the precise constructions.

The Bernoulli race [DHKN17] subroutine `BernoulliRace`($\{O^1, O^2, \dots, O^q\}$) is given accesses to a list of coins $O^1 = O_{\xi_1}, O^2 = O_{\xi_2}, \dots, O^q = O_{\xi_q}$ with unknown $\xi_1, \xi_2, \dots, \xi_q \in [0, 1]$. Its goal is to return a random $I \in [q]$ such that $I = i$ with probability $\xi_i / \sum_{j=1}^q \xi_j$. This can be achieved by independently repeating the following until a value is returned:

- choose $I \in [q]$ uniformly at random;
- if a draw of O^I returns 1 then return I .

The correctness of this procedure was given in [DHKN17].

Proposition A.1 ([DHKN17, Theorem 3.3]). *Given access to a list of coins $L = \{O^1, O^2, \dots, O^q\}$, where for each $i \in [q]$, the probability of heads for O^i is ξ_i , the `BernoulliRace`(L) defined above terminates with probability 1 and returns a random $I \in [q]$ such that $\Pr[I = i] = \xi_i / \sum_{j=1}^q \xi_j$ for every $i \in [q]$.*

To define the Bernoulli factory for subtraction, we further need to construct a linear Bernoulli factory, which transforms O_ξ to $O_{C\xi}$ for a $C > 1$ with the promise that $C\xi \leq 1$. We adopt the construction of linear Bernoulli factory in [Hub16] described in Algorithm 7. Its correctness is guaranteed as follows.

Proposition A.2 ([Hub16, Theorem 1]). *Given access to a coin O_ξ , given as input $C > 1$ and $\zeta > 0$, with promise that $C\xi \leq 1 - \zeta$, `LinearBF`(O, C, ζ) terminates with probability 1 and returns a draw of $O_{C\xi}$.*

A Bernoulli factory for subtraction, `SubtractBF`($O_{\xi_1}, O_{\xi_2}, \zeta$), is given in [NP05], which transforms two coins O_{ξ_1}, O_{ξ_2} with the promise that $\xi_1 - \xi_2 \geq \zeta > 0$, to a new coin $O_{\xi_1 - \xi_2}$. We implement this procedure using the linear Bernoulli factory defined above:

- `SubtractBF`($O_{\xi_1}, O_{\xi_2}, \zeta$) = $1 - \text{LinearBF}(O_{(1-\xi_1+\xi_2)/2}, 2, \zeta)$,

where the coin $O_{(1-\xi_1+\xi_2)/2}$ is realized with $O_{1/2}, O_{\xi_1}$ and O_{ξ_2} as follows: if $O_{1/2} = 1$, return $1 - O_{\xi_1}$; otherwise, return O_{ξ_2} . The correctness of this procedure is guaranteed as follows.

Algorithm 7: LinearBF(O, C, ζ) [Hub16]

Input: a coin $O = O_\xi$ with unknown ξ , $C > 1$ and a slack $\zeta > 0$, with promise that $C\xi \leq 1 - \zeta$;
Output: a random value Bernoulli($C\xi$);

```
1  $k \leftarrow 4.6/\zeta, \zeta \leftarrow \min\{\zeta, 0.644\}, i \leftarrow 1;$   
2 repeat  
3   repeat  
4     draw  $B \leftarrow O, G \leftarrow \text{Geometric}(\frac{C-1}{C});$   
     //  $G$  is drawn according to geometric distribution with parameter  $\frac{C-1}{C}$   
5      $i \leftarrow i - 1 + (1 - B)G;$   
6   until  $i = 0$  or  $i \geq k;$   
7   if  $i \geq k$  then  
8     draw  $R \leftarrow \text{Bernoulli}((1 + \zeta/2)^{-i});$   
9      $C \leftarrow C(1 + \zeta/2), \zeta \leftarrow \zeta/2, k \leftarrow 2k;$   
10 until  $i = 0$  or  $R = 0;$   
11 return  $\mathbb{1}[i = 0];$ 
```

Proposition A.3 ([NP05, Proposition 14, (iv)]). *Given access to two coins O_{ξ_1} and O_{ξ_2} , and given as input $\zeta > 0$, with promise that $\xi_1 - \xi_2 \geq \zeta$, SubtractBF($O_{\xi_1}, O_{\xi_2}, \zeta$) terminates with probability 1 and returns a draw of $O_{\xi_1 - \xi_2}$.*

Recall the coins O_{θ_v} and $O_{v(x)}$ for $x \in Q_v$ where $v = \mu_v^\sigma$. We further assume that the values in Q_v are enumerated in an arbitrary order as $Q_v = \{x_1, x_2, \dots, x_{q_v}\}$. To draw a sample from the distribution \mathcal{D} defined in (55), we construct the following Bernoulli factory:

- for each $i \in [q_v]$, let $O^i = \text{SubtractBF}(O_{v(x_i)}, O_{\theta_v}, \zeta)$;
- draw $I \leftarrow \text{BernoulliRace}(\{O^1, O^2, \dots, O^{q_v}\})$, and return the I -th value x_I in Q_v .

The parameter $\zeta > 0$ in above is as fixed in (7) and satisfies the promise (54) assuming Condition 3.8 (due to Corollary 4.4). Thus by Propositions A.1, A.2 and A.3, the above procedure terminates with probability 1 and returns an x_I distributed as \mathcal{D} defined as in (55). This proves Lemma 3.10.

A.2. Efficiency of the Bernoulli Factory. We now bound the efficiency of the Bernoulli factory constructed above. In this analysis, we need to explicitly bound the costs for realizations of the basic oracles $O_{v(x)}$ for $x \in Q_v$ through the rejection sampling $O_{v(x)} = \mathbb{1}[\text{RejectionSampling}(\Phi, \sigma, \{v\}) = x]$, whose complexity is measured in terms of both the computation cost and the query complexity for the evaluation oracle in Assumption 1.

Recall the simplification and decomposition of CSP defined in Section 3.2. Let $\Phi^\sigma = (V^\sigma, Q^\sigma, C^\sigma)$ denote the simplification of Φ under partial assignment $\sigma \in Q^*$, and $H^\sigma = H_{\Phi^\sigma} = (V^\sigma, C^\sigma)$ its hypergraph representation. Recall that for each $v \in V^\sigma$, $H_v^\sigma = (V_v^\sigma, C_v^\sigma)$ denotes the connected component in H^σ that contains the vertex/variable v . Let Φ_v^σ be its corresponding formula.

We show the following theorem for upper bound on the complexity of the Bernoulli factory.

Theorem A.4. *Assuming Condition 3.8 for the input (Φ, σ, v) , the Bernoulli factory algorithm constructed in Appendix A.1 costs in expectation:*

- $O\left(q^2 k^2 \Delta^6 (|C_v^\sigma| + 1)(1 - \text{ep}'q)^{-|C_v^\sigma|}\right)$ queries to the evaluation oracle in Assumption 1;
- $O\left(q^3 k^3 \Delta^6 (|C_v^\sigma| + 1)(1 - \text{ep}'q)^{-|C_v^\sigma|}\right)$ in computation.

To prove Theorem A.4, we first bound the cost for realizing the basic oracles $O_{v(x)}$ for $x \in Q_v$.

Lemma A.5. *Assume Condition 3.8 for the input (Φ, σ, v) . It takes at most $\Delta(|C_v^\sigma| + 1)$ queries to the evaluation oracle in Assumption 1 and $O(k\Delta(|C_v^\sigma| + 1))$ computation cost for preprocessing the oracles $O_{v(x)}$ for all $x \in Q_v$. And upon each query, $O_{v(x)}$ returns using at most $|C_v^\sigma|(1 - \text{ep}'q)^{-|C_v^\sigma|}$ queries to the evaluation oracle in expectation and $O\left((qk(|C_v^\sigma| + 1))(1 - \text{ep}'q)^{-|C_v^\sigma|}\right)$ computation cost in expectation.*

Proof. The oracle $O_{v(x)}$ is computed as $O_{v(x)} = \mathbb{1} [\text{RejectionSampling}(\Phi, \sigma, \{v\}) = x]$.

First, observe that $\text{RejectionSampling}(\Phi, \sigma, \{v\})$ is equivalent to $\text{RejectionSampling}(\Phi_v^\sigma, \sigma_{V \setminus \Lambda(\sigma)}, \{v\})$. By using a depth-first search in H^σ , the connected component $\Phi_v^\sigma = (V_v^\sigma, Q_v^\sigma, C_v^\sigma)$ can be constructed using at most $\Delta(|C_v^\sigma| + 1)$ queries to the evaluation oracle and $O(\Delta|C_v^\sigma| + \Delta|V_v^\sigma|) = O(k\Delta(|C_v^\sigma| + 1))$ computation cost, because $|V_v^\sigma| \leq k|C_v^\sigma| + 1$. This is the preprocessing cost.

Then a query to oracle $O_{v(x)}$ is reduced to a calling to $\text{RejectionSampling}(\Phi_v^\sigma, \sigma_{V \setminus \Lambda(\sigma)}, \{v\})$ using Algorithm 2. In fact, Line 1 can be skipped and $K = 1$ in Line 2 since the component Φ_v^σ containing v has been explicitly constructed in the preprocessing. It is well known that the expected number of trials (the **repeat** loop in Line 5) taken by the rejection sampling until success is given by $\mathbb{P}_{\Phi_v^\sigma}[\Omega_{\Phi_v^\sigma}]^{-1}$, where $\Omega_{\Phi_v^\sigma}$ is the set of satisfying assignments of Φ_v^σ and hence $\mathbb{P}_{\Phi_v^\sigma}[\Omega_{\Phi_v^\sigma}]$ gives the probability that a uniform random assignment is satisfying for Φ_v^σ . By Theorem 4.1, assuming Condition 3.8,

$$\mathbb{P}_{\Phi_v^\sigma}[\Omega_{\Phi_v^\sigma}] \geq (1 - ep'q)^{|C_v^\sigma|}.$$

The rejection sampling in $\text{RejectionSampling}(\Phi_v^\sigma, \sigma_{V \setminus \Lambda(\sigma)}, \{v\})$ takes $(1 - ep'q)^{-|C_v^\sigma|}$ trials in expectation. And within each trial, it is easy to verify that it uses at most $|C_v^\sigma|$ queries to the evaluation oracle and $O(k|C_v^\sigma| + q|V_v^\sigma|) = O(qk(|C_v^\sigma| + 1))$ computation cost. This proves the lemma. \square

We then state known results for the efficiency of Bernoulli factories.

Proposition A.6 ([DHKN17, Theorem 3.3]). *Given access to a list of coins $L = \{O^1, O^2, \dots, O^q\}$, where for each $i \in [q]$, the probability of heads for O^i is ξ_i , the expected number of queries to O^1, O^2, \dots, O^q for executing $\text{BernoulliRace}(L)$ is at most $q/(\sum_{i=1}^q \xi_i)$.*

Proposition A.7 ([Hub16, Theorem 1]). *Given access to a coin O_ξ , given as input $C > 1, \zeta > 0$, with the promise $C\xi \leq 1 - \zeta$, the expected number of queries to O_ξ made in $\text{LinearBF}(O, C, \zeta)$ is at most $9.5C/\zeta$.*

By Proposition A.7, we have the following complexity bound for the Bernoulli factory for subtraction.

Corollary A.8. *Given access to two coins O_{ξ_1}, O_{ξ_2} , and given as input $\zeta > 0$, with the promise $\xi_1 - \xi_2 \geq \zeta$, the expected number of queries to O_{ξ_1}, O_{ξ_2} for executing $\text{SubtractBF}(O_{\xi_1}, O_{\xi_2}, \zeta)$ is at most $\frac{39\zeta^{-1}}{1 - (\xi_1 - \xi_2)}$.*

Proof of Theorem A.4. Recall that our Bernoulli factory algorithm is $\text{BernoulliRace}(\{O^1, O^2, \dots, O^{q_v}\})$ where $O^i = \text{SubtractBF}(O_{v(x_i)}, O_{\theta_v}, \zeta)$ for the i -th value $x_i \in Q_v$ and $v = \mu_v^\sigma$. By Proposition A.6 and Corollary A.8, the total number of queries to the basic oracles $O_{v(x)}$ for $x \in Q_v$ is bounded by:

$$\frac{q_v}{\sum_{x \in Q_v} (\mu_v^\sigma(x) - \theta_v)} \cdot \left(\max_{x \in Q_v} \frac{39\zeta^{-1}}{1 - (\mu_v^\sigma(x) - \theta_v)} \right) \leq \frac{39}{\zeta^2(1 - 2\eta - \zeta)} = O(q^2 k^2 \Delta^6),$$

where the inequality is due to $\mu_v^\sigma(x) \leq \theta_v + 2\eta + \zeta$ by Corollary 4.4 assuming Condition 3.8. The theorem then follows by applying Lemma A.5 and observing that the preprocessing costs are paid only once in the beginning. \square

APPENDIX B. MONOTONICITY OF CONSTRAINT PROPERTIES

In this section, we prove two technical lemmas (Lemma 6.32 and Lemma 6.30) regarding the monotonicity of various constraints properties $C_\star^\sigma, C_{\text{frozen}}^\sigma, C_{\star\text{-con}}^\sigma, C_{\star\text{-frozen}}^\sigma, C_{\star\text{-bad}}^\sigma$, where C_{frozen}^σ is defined in Definition 3.1, $C_{\star\text{-con}}^\sigma$ in Definition 3.6, and $C_\star^\sigma, C_{\star\text{-frozen}}^\sigma, C_{\star\text{-bad}}^\sigma$ in Definition 6.13.

Recall in Definition 3.6: for any $\sigma \in Q^*$, H_{fix}^σ denotes the sub-hypergraph of H^σ induced by $V^\sigma \cap V_{\text{fix}}^\sigma$. Recall that for each $v \in V^\sigma$, $H_v^\sigma = (V_v^\sigma, C_v^\sigma)$ denotes the connected component in H^σ that contains the vertex/variable v . Also, for each $c \in C$, we denote the simplified constraint of c under σ as c^σ .

B.1. Proof of Lemma 6.32. The following properties will be used in the proof.

Lemma B.1. *Given $\sigma \in Q^*$ with $\text{NextVar}(\sigma) = u \neq \perp$, it holds that $u \notin V_{\text{fix}}^\sigma$.*

Proof. By $u = \text{NextVar}(\sigma) \neq \perp$ and the definition of $\text{NextVar}(\sigma)$ in Definition 3.6, we have $u \in V_{\star\text{-inf}}^\sigma$. Combining with the definition of $V_{\star\text{-inf}}^\sigma$, we have $u \notin V_\star^\sigma$. Combining with $V_\star^\sigma \subseteq V^\sigma \cap V_{\text{fix}}^\sigma$, we have $u \notin V_{\text{fix}}^\sigma$. \square

The next lemma states a basic monotonicity property when extending some partial assignment σ on $\text{NextVar}(\sigma)$.

Lemma B.2. *Given $\sigma \in Q^*$ with $\text{NextVar}(\sigma) = u \neq \perp$ and $a \in Q_u \cup \{\star\}$, let $\tau = \sigma_{u \leftarrow a}$. It holds that*

$$C_{\mathcal{P}}^\sigma \subseteq C_{\mathcal{P}}^\tau,$$

where \mathcal{P} can be any property $\mathcal{P} \in \{\star, \text{frozen}, \star\text{-con}, \star\text{-frozen}, \star\text{-bad}\}$.

Proof. At first, we prove $C_\star^\sigma \subseteq C_\star^\tau$. By $\text{NextVar}(\sigma) = u \neq \perp$ and Lemma B.1, we have $u \notin V_{\text{fix}}^\sigma$. Combining with the definition of V_{fix}^σ in Definition 3.1, we have $\sigma(u) \neq \star$. Thus,

$$\begin{aligned} C_\star^\sigma &= \{c \in C \mid \exists v \in \text{vbl}(c) \text{ s.t. } \sigma(v) = \star\} \\ &\quad (\text{by } \sigma(u) \neq \star) = \{c \in C \mid \exists v \in \text{vbl}(c) \text{ s.t. } (\sigma(v) = \star) \wedge (v \neq u)\} \\ &\quad (\text{by } \sigma(v) = \tau(v) \text{ if } v \neq u) = \{c \in C \mid \exists v \in \text{vbl}(c) \text{ s.t. } (\tau(v) = \star) \wedge (v \neq u)\} \\ &\subseteq \{c \in C \mid \exists v \in \text{vbl}(c) \text{ s.t. } \tau(v) = \star\} \\ &\quad (\text{by Definition 6.13}) = C_\star^\tau. \end{aligned}$$

Now we prove $C_{\text{frozen}}^\sigma \subseteq C_{\text{frozen}}^\tau$. For each $c \in C_{\text{frozen}}^\sigma$, we have $\text{vbl}(c) \subseteq V_{\text{fix}}^\sigma$. Combining with $u \notin V_{\text{fix}}^\sigma$, we have $u \notin \text{vbl}(c)$, which says $\tau_{\text{vbl}(c)} = \sigma_{\text{vbl}(c)}$, hence $c \in C_{\text{frozen}}^\tau$ by the consistency assumption of frozen oracle in Assumption 2. In summary, we have $C_{\text{frozen}}^\sigma \subseteq C_{\text{frozen}}^\tau$.

In the next, we prove $C_{\star\text{-con}}^\sigma \subseteq C_{\star\text{-con}}^\tau$. For each $c \in C_{\star\text{-con}}^\sigma$, by Definition 3.6, there exists some vertex $v \in \text{vbl}(c) \cap V_\star^\sigma$. By $v \in V_\star^\sigma$, we have there exists some variable v' such that $\sigma(v') = \star$ and v and v' are connected in H_{fix}^σ . We claim that H_{fix}^σ is a sub-hypergraph of H_{fix}^τ . Then we have v and v' are connected in H_{fix}^τ . In addition, recall $u \notin V_{\text{fix}}^\sigma$. Combining with $v' \in V_{\text{fix}}^\sigma$, we have $u \neq v'$ and then $\tau(v') = \sigma(v') = \star$. Combining with v and v' are connected in H_{fix}^τ , we have $v \in V_\star^\tau$. Combining with $v \in \text{vbl}(c)$, we have $c \in C_{\star\text{-con}}^\tau$. In summary, we have $C_{\star\text{-con}}^\sigma \subseteq C_{\star\text{-con}}^\tau$.

Now we prove the claim that H_{fix}^σ is a sub-hypergraph of H_{fix}^τ . At first, we show

$$(56) \quad V^\sigma \cap V_{\text{fix}}^\sigma \subseteq V^\tau \cap V_{\text{fix}}^\tau.$$

Obviously, $\Lambda^+(\sigma) \subseteq \Lambda^+(\tau)$. Combining with $C_{\text{frozen}}^\sigma \subseteq C_{\text{frozen}}^\tau$, we have $V_{\text{fix}}^\sigma \subseteq V_{\text{fix}}^\tau$. In addition, we have

$$V^\sigma \cap V_{\text{fix}}^\sigma = (V^\sigma \setminus \{u\}) \cap V_{\text{fix}}^\sigma \subseteq V^\tau \cap V_{\text{fix}}^\sigma \subseteq V^\tau \cap V_{\text{fix}}^\tau,$$

where the first relation is by $u \notin V_{\text{fix}}^\sigma$, the second is by $V^\sigma \setminus \{u\} \subseteq V^\tau$ and the last is by $V_{\text{fix}}^\sigma \subseteq V_{\text{fix}}^\tau$. Thus, we have (56) holds. In addition, let C_{fix}^σ be the hyperedge set of H_{fix}^σ and C_{fix}^τ be the hyperedge set of H_{fix}^τ . We can show that

$$\begin{aligned} C_{\text{fix}}^\sigma &= \{c^\sigma \mid (\text{vbl}(c^\sigma) \subseteq V^\sigma \cap V_{\text{fix}}^\sigma) \wedge (c \text{ is not satisfied by } \sigma)\} \\ &\quad (\text{by } u \notin V_{\text{fix}}^\sigma) = \{c^\sigma \mid (\text{vbl}(c^\sigma) \subseteq V^\sigma \cap V_{\text{fix}}^\sigma) \wedge (u \notin \text{vbl}(c)) \wedge (c \text{ is not satisfied by } \sigma)\} \\ &\quad (\text{by } c^\sigma = c^\tau \text{ if } u \notin \text{vbl}(c)) \subseteq \{c^\tau \mid (\text{vbl}(c^\tau) \subseteq V^\sigma \cap V_{\text{fix}}^\sigma) \wedge (u \notin \text{vbl}(c)) \wedge (c \text{ is not satisfied by } \tau)\} \\ &\quad (\text{by (56)}) \subseteq \{c^\tau \mid (\text{vbl}(c^\tau) \subseteq V^\tau \cap V_{\text{fix}}^\tau) \wedge (u \notin \text{vbl}(c)) \wedge (c \text{ is not satisfied by } \tau)\} \\ &\subseteq \{c^\tau \mid (\text{vbl}(c^\tau) \subseteq V^\tau \cap V_{\text{fix}}^\tau) \wedge (c \text{ is not satisfied by } \tau)\} \\ &= C_{\text{fix}}^\tau \end{aligned}$$

Combining with (56), we have proven the claim that H_{fix}^σ is a sub-hypergraph of H_{fix}^τ .

By $C_{\text{frozen}}^\sigma \subseteq C_{\text{frozen}}^\tau$ and $C_{\star\text{-con}}^\sigma \subseteq C_{\star\text{-con}}^\tau$, we have

$$C_{\star\text{-frozen}}^\sigma = C_{\text{frozen}}^\sigma \cap C_{\star\text{-con}}^\sigma \subseteq C_{\text{frozen}}^\tau \cap C_{\star\text{-con}}^\tau = C_{\star\text{-frozen}}^\tau.$$

Similarly, by $C_\star^\sigma \subseteq C_\star^\tau$ and $C_{\star\text{-frozen}}^\sigma \subseteq C_{\star\text{-frozen}}^\tau$, we have

$$C_{\star\text{-bad}}^\sigma = C_\star^\sigma \bigcup C_{\star\text{-frozen}}^\sigma \subseteq C_\star^\tau \bigcup C_{\star\text{-frozen}}^\tau = C_{\star\text{-bad}}^\tau.$$

□

By definition of $\text{Path}(\cdot)$ and Lemma B.2, Lemma 6.32 is immediate by induction.

B.2. Proof of Lemma 6.30. The following lemma will be used in the proof.

Lemma B.3. *Assume the condition of Lemma 6.26. For each $c^{\sigma_\ell} \in C_v^{\sigma_\ell}$ and each variable $u \in \text{vbl}(c^{\sigma_\ell})$, it holds that $c \in C_{\star\text{-con}}^{\sigma_\ell}$ and $u \in V_{\text{fix}}^{\sigma_\ell}$.*

Proof. For each simplified constraint $c^{\sigma_\ell} \in C_v^{\sigma_\ell}$, by the definition of $C_v^{\sigma_\ell}$, we have there exists a connected path $c_1^{\sigma_\ell}, c_2^{\sigma_\ell}, \dots, c_t^{\sigma_\ell} \in C_v^{\sigma_\ell}$ such that $v \in \text{vbl}(c_1^{\sigma_\ell})$, $c_t^{\sigma_\ell} = c^{\sigma_\ell}$ and $c_i^{\sigma_\ell}$ intersects $c_{i+1}^{\sigma_\ell}$ for each $1 \leq i < t$. Let $\text{dist}(v, c^{\sigma_\ell})$, or $\text{dist}(v, c^{\sigma_\ell})$ for short, denote the length of the shortest connected path from v to c^{σ_ℓ} in H_ℓ^σ . We prove the lemma by induction on $\text{dist}(v, c^{\sigma_\ell})$.

For the base case when $\text{dist}(v, c^{\sigma_\ell}) = 1$, we have $v \in \text{vbl}(c^{\sigma_\ell})$. In addition, by $\sigma_\ell(v) = \sigma(v) = \star$, we have $v \in V_{\text{fix}}^{\sigma_\ell} \cap V_{\text{fix}}^{\sigma_\ell}$. Thus, we have $v \in V_{\star}^{\sigma_\ell}$. Combining with $v \in \text{vbl}(c^{\sigma_\ell})$, we have $c \in C_{\star\text{-con}}^{\sigma_\ell}$ by Definition 3.6. In addition, we have $u \in V_{\text{fix}}^{\sigma_\ell}$ for each $u \in \text{vbl}(c^{\sigma_\ell})$. Because otherwise, $u \notin V_{\text{fix}}^{\sigma_\ell}$. We have $u \in \text{vbl}(c^{\sigma_\ell}) \setminus V_{\text{fix}}^{\sigma_\ell} \subseteq V^{\sigma_\ell} \setminus V_{\star}^{\sigma_\ell}$ by $V_{\star}^{\sigma_\ell} \subseteq V_{\text{fix}}^{\sigma_\ell}$. In addition, by $u, v \in \text{vbl}(c^{\sigma_\ell})$, $v \in V_{\star}^{\sigma_\ell}$, and $c^{\sigma_\ell} \in C_v^{\sigma_\ell} \subseteq C^{\sigma_\ell}$, we have $u \in V_{\star\text{-inf}}^{\sigma_\ell}$. Therefore, we have $V_{\star\text{-inf}}^{\sigma_\ell} \neq \emptyset$ and $\text{NextVar}(\sigma_\ell) \neq \perp$. Thus, by Definition 6.10, there must be another partial assignment $\sigma_{\ell+1}$ generated from σ_ℓ , which is contradictory with $\text{Path}(\sigma) = (\sigma_0, \dots, \sigma_\ell)$.

For the induction step, we assume $\text{dist}(v, c^{\sigma_\ell}) = t > 1$. Let $c_1^{\sigma_\ell}, c_2^{\sigma_\ell}, \dots, c_t^{\sigma_\ell} \in C_v^{\sigma_\ell}$ be a connected path in $H_v^{\sigma_\ell}$ with $v \in \text{vbl}(c_1^{\sigma_\ell})$, $c_t^{\sigma_\ell} = c^{\sigma_\ell}$ and $c_i^{\sigma_\ell}$ intersects $c_{i+1}^{\sigma_\ell}$ for each $1 \leq i < t$. By the induction hypothesis and $c_i^{\sigma_\ell} \in C_v^{\sigma_\ell}$, we have $c_i \in C_{\star\text{-con}}^{\sigma_\ell}$ for each $i < t$. Choose some $w \in \text{vbl}(c_{t-1}^{\sigma_\ell}) \cap \text{vbl}(c_t^{\sigma_\ell})$. By the induction hypothesis, we have $w \in V_{\text{fix}}^{\sigma_\ell}$. Combining with $w \in V^{\sigma_\ell}$, we have $w \in V_{\text{fix}}^{\sigma_\ell} \cap V^{\sigma_\ell}$. Recall that $\sigma_\ell(v) = \star$. Combining with $w \in V_{\text{fix}}^{\sigma_\ell} \cap V^{\sigma_\ell}$ and that $w \in \text{vbl}(c_{t-1}^{\sigma_\ell})$ is connected to $v \in \text{vbl}(c_1^{\sigma_\ell})$ by the path $c_1^{\sigma_\ell}, c_2^{\sigma_\ell}, \dots, c_{t-1}^{\sigma_\ell}$ in $H_v^{\sigma_\ell}$, we have $w \in V_{\star}^{\sigma_\ell}$. Combining with $w \in \text{vbl}(c_t^{\sigma_\ell})$, we have $c_t \in C_{\star\text{-con}}^{\sigma_\ell}$. In addition, we have $u \in V_{\text{fix}}^{\sigma_\ell}$ for each $u \in \text{vbl}(c^{\sigma_\ell})$. Because otherwise, $u \notin V_{\text{fix}}^{\sigma_\ell}$. We have $u \in \text{vbl}(c^{\sigma_\ell}) \setminus V_{\text{fix}}^{\sigma_\ell} \subseteq V^{\sigma_\ell} \setminus V_{\star}^{\sigma_\ell}$ by $V_{\star}^{\sigma_\ell} \subseteq V_{\text{fix}}^{\sigma_\ell}$. In addition, by $u, w \in \text{vbl}(c^{\sigma_\ell})$, $w \in V_{\star}^{\sigma_\ell}$, and $c^{\sigma_\ell} \in C_v^{\sigma_\ell} \subseteq C^{\sigma_\ell}$, we have $u \in V_{\star\text{-inf}}^{\sigma_\ell}$. Therefore, similar to the base case one can also reach a contradiction. This completes the induction step and the proof of the lemma. □

Now we can prove Lemma 6.30.

Proof of Lemma 6.30. Because $|C_v^{\sigma_\ell}| \leq |C_{\star\text{-con}}^{\sigma_\ell}|$ is immediate by Lemma B.3, it is sufficient to show that $|C_{\star\text{-con}}^{\sigma_\ell}| \leq \Delta \cdot |C_{\star\text{-bad}}^{\sigma_\ell}|$. We show this by proving that for each $c \in C_{\star\text{-con}}^{\sigma_\ell}$, there exists some $c' \in C_{\star\text{-bad}}^{\sigma_\ell}$ such that $\text{vbl}(c) \cap \text{vbl}(c') \neq \emptyset$.

For each $c \in C_{\star\text{-con}}^{\sigma_\ell}$, by Definition 3.6, we have there exists some $u \in V_{\star}^{\sigma_\ell} \cap \text{vbl}(c^{\sigma_\ell})$. By $u \in V_{\star}^{\sigma_\ell}$, we have $u \in V^{\sigma_\ell} \cap V_{\text{fix}}^{\sigma_\ell}$. By $u \in V^{\sigma_\ell}$, we have $u \notin \Lambda(\sigma_\ell)$. Combining with $u \in V_{\text{fix}}^{\sigma_\ell}$, we have either $\sigma_\ell(u) = \star$ or $u \in \widehat{c}$ for some $\widehat{c} \in C_{\text{frozen}}^{\sigma_\ell}$. If $\sigma_\ell(u) = \star$, we have $c \in C_{\star}^{\sigma_\ell} \subseteq C_{\star\text{-bad}}^{\sigma_\ell}$. Otherwise, $u \in \widehat{c}$ for some $\widehat{c} \in C_{\text{frozen}}^{\sigma_\ell}$. In addition, we also have $\widehat{c} \in C_{\star\text{-con}}^{\sigma_\ell}$ by $u \in V_{\star}^{\sigma_\ell}$ and $u \in \text{vbl}(\widehat{c})$. Combining with $\widehat{c} \in C_{\text{frozen}}^{\sigma_\ell}$, we have $\widehat{c} \in C_{\star\text{-frozen}}^{\sigma_\ell} \subseteq C_{\star\text{-bad}}^{\sigma_\ell}$. Let

$$c' = \begin{cases} c & \text{if } \sigma_\ell(u) = \star, \\ \widehat{c} & \text{otherwise.} \end{cases}$$

Obviously, $c' \in C_{\star\text{-bad}}^{\sigma_\ell}$ and $\text{vbl}(c) \cap \text{vbl}(c') \neq \emptyset$. This completes the proof. □

APPENDIX C. $\{2, 3\}$ -TREE WITNESS FOR CONSTRAINT PROPERTIES

In this section, we prove two technical lemmas (Lemma 6.26 and Lemma 6.35), both of which state that for certain constraint properties such as $C_{\star\text{-bad}}^\sigma$ or C_{frozen}^σ , when the class of constraints with that property becomes too large, a large enough $\{2, 3\}$ -tree inevitably appears within the class.

Recall the $\{2, 3\}$ -tree in the dependency graph $G(C)$ in Definition 6.22 and Definition 6.23.

Recall in Definition 3.6: for any $\sigma \in Q^*$, H_{fix}^σ denotes the sub-hypergraph of H^σ induced by $V^\sigma \cap V_{\text{fix}}^\sigma$. Recall that for each $v \in V^\sigma$, $H_v^\sigma = (V_v^\sigma, C_v^\sigma)$ denotes the connected component in H^σ that contains the vertex/variable v . Also, for each $c \in C$, we denote the simplified constraint of c under σ as c^σ .

C.1. Proof of Lemma 6.26. The following lemma will be used in the proof.

Lemma C.1. *Assume the condition of Lemma 6.26. Then $G^2(C_{\star\text{-bad}}^{\sigma_i})$ is connected for each $0 \leq i \leq \ell$.*

Proof. We prove this lemma by induction on i . For simplicity, we say a constraint c is connected to a subset $S \subseteq C$ in $G^2(C)$ if c is connected to some $c' \in S$. The base case is when $i = 0$. By the condition of the lemma, v is the only variable satisfying $\sigma(v) = \star$. Combining with $\sigma_0 = \sigma$, we have v is the only variable satisfying $\sigma_0(v) = \star$. Therefore, $C_{\star}^{\sigma_0} = \{c \in C \mid v \in \text{vbl}(c)\}$. In addition, we have the following claim: each $c \in C_{\star\text{-frozen}}^{\sigma_0}$ is connected to $C_{\star}^{\sigma_0}$ in $G^2(C_{\star\text{-bad}}^{\sigma_0})$. Note that $C_{\star}^{\sigma_0} = \{c \in C \mid v \in \text{vbl}(c)\}$ is connected in $G^2(C)$. Combining with the claim, we have $G^2(C_{\star\text{-bad}}^{\sigma_0})$ is connected.

Now we prove the claim, which completes the proof of the base case. By $c \in C_{\star\text{-frozen}}^{\sigma_0}$, we have $c \in C_{\star\text{-con}}^{\sigma_0} \cap C_{\text{frozen}}^{\sigma_0}$. By $c \in C_{\star\text{-con}}^{\sigma_0}$ and Definition 3.6, we have $V_{\star}^{\sigma_0} \cap \text{vbl}(c) \neq \emptyset$. Combining with the definition of $V_{\star}^{\sigma_0}$, we have there exists a variable w and a connected path $c_1^{\sigma_0}, c_2^{\sigma_0}, \dots, c_t^{\sigma_0} = c^{\sigma_0} \in C^{\sigma_0}$ such that $\sigma_0(w) = \star$, $w \in \text{vbl}(c_1^{\sigma_0})$ and $\text{vbl}(c_j^{\sigma_0}) \subseteq V^{\sigma_0} \cap V_{\text{fix}}^{\sigma_0}$ for each $j < t$. If $c = c_1$, we have $c \in C_{\star}^{\sigma_0}$ by $\sigma_0(w) = \star$ and $w \in \text{vbl}(c_1)$, then the claim is immediate. In the following, we assume $c \neq c_1$. Let $w_j \in (\text{vbl}(c_j^{\sigma_0}) \cap \text{vbl}(c_{j+1}^{\sigma_0}))$ for each $j < t$. Then $w_j \notin \Lambda(\sigma_0)$. By $w_j \in \text{vbl}(c_j^{\sigma_0})$ and $\text{vbl}(c_j^{\sigma_0}) \subseteq V_{\text{fix}}^{\sigma_0}$, we have $w_j \in V_{\text{fix}}^{\sigma_0}$. Combining with $w_j \notin \Lambda(\sigma_0)$, we have either $\sigma_0(w_j) = \star$, then $w_j \in \text{vbl}(\widehat{c}_j)$ for some $\widehat{c}_j \in C_{\star}^{\sigma_0}$, or $w_j \in \text{vbl}(\widehat{c}_j)$ for some $\widehat{c}_j \in C_{\text{frozen}}^{\sigma_0}$. If $\widehat{c}_j \in C_{\star}^{\sigma_0}$, we have $\widehat{c}_j \in C_{\star}^{\sigma_0} \subseteq C_{\star\text{-bad}}^{\sigma_0}$. Otherwise, $\widehat{c}_j \in C_{\text{frozen}}^{\sigma_0}$. By w_j is connected to w in $H_{\text{fix}}^{\sigma_0}$ through the path $c_1^{\sigma_0}, c_2^{\sigma_0}, \dots, c_j^{\sigma_0}$, we have $w_j \in V_{\star}^{\sigma_0}$. Thus, we have $\widehat{c}_j \in C_{\star\text{-con}}^{\sigma_0}$ by Definition 3.6. Combining with $\widehat{c}_j \in C_{\text{frozen}}^{\sigma_0}$, we have $\widehat{c}_j \in C_{\star\text{-frozen}}^{\sigma_0} \subseteq C_{\star\text{-bad}}^{\sigma_0}$. In summary, we always have $\widehat{c}_j \in C_{\star\text{-bad}}^{\sigma_0}$. Moreover, for each $j < t-1$, we have $w_j \in \text{vbl}(c_{j+1}^{\sigma_0}) \cap \text{vbl}(\widehat{c}_{j+1}^{\sigma_0})$ and $w_{j+1} \in \text{vbl}(c_{j+1}^{\sigma_0}) \cap \text{vbl}(\widehat{c}_{j+1}^{\sigma_0})$. Thus, \widehat{c}_j and \widehat{c}_{j+1} are connected in $G^2(C)$. In addition, we have $w_1 \in \text{vbl}(c_1) \cap \text{vbl}(\widehat{c}_1)$ and $w_{t-1} \in \text{vbl}(\widehat{c}_{t-1}) \cap \text{vbl}(c_t)$. Thus, we have $c_1, \widehat{c}_1, \widehat{c}_2, \dots, \widehat{c}_{t-1}, c_t = c$ is a connected path in $G^2(C)$. Combining with $c_1 \in C_{\star}^{\sigma_0} \subseteq C_{\star\text{-bad}}^{\sigma_0}$ and $\widehat{c}_j \in C_{\star\text{-bad}}^{\sigma_0}$ for each $j < t$, the claim is immediate.

For the induction step, we prove this lemma for each $i > 0$. We claim that each $c \in C_{\star}^{\sigma_i}$ is connected to $C_{\star}^{\sigma_{i-1}}$ in $G^2(C_{\star\text{-bad}}^{\sigma_i})$. Moreover, by the induction hypothesis we have $C_{\star}^{\sigma_{i-1}}$ is connected in $G^2(C_{\star\text{-bad}}^{\sigma_{i-1}})$. Combining with $C_{\star\text{-bad}}^{\sigma_{i-1}} \subseteq C_{\star\text{-bad}}^{\sigma_i}$ by Lemma B.2, we have $C_{\star}^{\sigma_{i-1}}$ is connected in $G^2(C_{\star\text{-bad}}^{\sigma_i})$. Combining with the claim that each $c \in C_{\star}^{\sigma_i}$ is connected to $C_{\star}^{\sigma_{i-1}}$ in $G^2(C_{\star\text{-bad}}^{\sigma_i})$, we have $C_{\star}^{\sigma_i}$ is connected in $G^2(C_{\star\text{-bad}}^{\sigma_i})$. In addition, we can also prove that each constraint in $C_{\star\text{-frozen}}^{\sigma_i}$ is connected to $C_{\star}^{\sigma_i}$ in $G^2(C_{\star\text{-bad}}^{\sigma_i})$ by a similar argument to the base case. Combining with that $C_{\star}^{\sigma_i}$ is connected in $G^2(C_{\star\text{-bad}}^{\sigma_i})$, we have $G^2(C_{\star\text{-bad}}^{\sigma_i})$ is connected.

Now we prove the claim that each $c \in C_{\star}^{\sigma_i}$ is connected to $C_{\star}^{\sigma_{i-1}}$ in $G^2(C_{\star\text{-bad}}^{\sigma_i})$, which completes the proof of the lemma. If $c \in C_{\star}^{\sigma_{i-1}}$, the claim is immediate by $C_{\star}^{\sigma_{i-1}} \subseteq C_{\star}^{\sigma_i} \subseteq C_{\star\text{-bad}}^{\sigma_i}$. In the following, we assume $c \in C_{\star}^{\sigma_i} \setminus C_{\star}^{\sigma_{i-1}}$. Let $u = \text{NextVar}(\sigma_{i-1})$. By the definition of $\text{NextVar}(\cdot)$, we have $u \in V_{\star\text{-inf}}^{\sigma_{i-1}}$ and then $u \in \text{vbl}(\widehat{c})$ for some constraint $\widehat{c} \in C_{\star\text{-con}}^{\sigma_{i-1}}$. In addition, by $\widehat{c} \in C_{\star\text{-con}}^{\sigma_{i-1}}$ one can verify that there exists a variable w and a connected path $c_1^{\sigma_{i-1}}, c_2^{\sigma_{i-1}}, \dots, c_t^{\sigma_{i-1}} = \widehat{c}^{\sigma_{i-1}} \in C^{\sigma_{i-1}}$ such that $\sigma_{i-1}(w) = \star$, $w \in \text{vbl}(c_1^{\sigma_{i-1}})$ and $\text{vbl}(c_j^{\sigma_{i-1}}) \subseteq V^{\sigma_{i-1}} \cap V_{\text{fix}}^{\sigma_{i-1}}$ for each $j < t$. Then there are two possibilities for \widehat{c} .

- If $\widehat{c} = c_1$, we have $\widehat{c} \in C_{\star}^{\sigma_{i-1}}$ by $c_1 \in C_{\star}^{\sigma_{i-1}}$. In addition, by $c \in C_{\star}^{\sigma_i} \setminus C_{\star}^{\sigma_{i-1}}$ and $u = \text{NextVar}(\sigma_{i-1})$, we have $u \in \text{vbl}(c)$. Because otherwise $u \notin \text{vbl}(c)$, we have $\sigma_i(w) = \sigma_{i-1}(w)$ for each $w \in \text{vbl}(c)$. Combining with $c \in C_{\star}^{\sigma_i}$, we have $c \in C_{\star}^{\sigma_{i-1}}$, which is contradictory with $c \in C_{\star}^{\sigma_i} \setminus C_{\star}^{\sigma_{i-1}}$.

Recall $u \in \text{vbl}(\widehat{c})$. We have $u \in \text{vbl}(c) \cap \text{vbl}(\widehat{c})$. Therefore, c is connected to $\widehat{c} \in C_{\star}^{\sigma_{i-1}}$ in $G^2(C_{\star}^{\sigma_{i-1}} \cup \{c\})$. In addition, by Lemma B.2 we have

$$C_{\star}^{\sigma_{i-1}} \cup \{c\} \subseteq C_{\star}^{\sigma_i} \cup \{c\} \subseteq C_{\star}^{\sigma_i} \subseteq C_{\star\text{-bad}}^{\sigma_i}.$$

Thus the claim is immediate.

- Otherwise, $\widehat{c} \neq c_1$. Similarly to the base case, one can find a connected path $c_1, \widehat{c}_1, \widehat{c}_2, \dots, \widehat{c}_{t-1}, c_t = \widehat{c}$ in $G^2(C)$, where $c_1 \in C_{\star}^{\sigma_{i-1}}$, $\widehat{c}_j \in C_{\star\text{-bad}}^{\sigma_{i-1}}$ for each $j < t$, and there exists $w_{t-1} \in \text{vbl}(c_t) \cap \text{vbl}(\widehat{c}_{t-1})$. Recall that $u \in \text{vbl}(\widehat{c}) \cap \text{vbl}(c)$. Combining with $\widehat{c} = c_t$, we have \widehat{c}_{t-1} is also connected to c in $G^2(C)$. Thus, $c_1, \widehat{c}_1, \widehat{c}_2, \dots, \widehat{c}_{t-1}, c$ is also a connected path in $G^2(C)$. Combining with $c_1 \in C_{\star}^{\sigma_{i-1}}$, $\widehat{c}_j \in C_{\star\text{-bad}}^{\sigma_{i-1}}$ for each $j < t$, we have c is connected to $C_{\star}^{\sigma_{i-1}}$ in $G^2(C_{\star\text{-bad}}^{\sigma_{i-1}} \cup \{c\})$. In addition, by Lemma B.2 we have

$$C_{\star\text{-bad}}^{\sigma_{i-1}} \cup \{c\} \subseteq C_{\star\text{-bad}}^{\sigma_i} \cup \{c\} \subseteq C_{\star\text{-bad}}^{\sigma_i} \cup C_{\star}^{\sigma_i} \subseteq C_{\star\text{-bad}}^{\sigma_i}.$$

Thus the claim is immediate. □

Now we can prove Lemma 6.26.

Proof of Lemma 6.26. At first, we prove that $\ell \leq k\Delta |C_{\star\text{-bad}}^{\sigma_{\ell}}|$. We claim that for each $0 \leq i < \ell$, there exist some c_i, c'_i such that $\text{NextVar}(\sigma_i) \in \text{vbl}(c_i)$, $c'_i \subseteq C_{\star\text{-bad}}^{\sigma_{\ell}}$, and $\text{vbl}(c_i) \cap \text{vbl}(c'_i) \neq \emptyset$. Therefore, for each $0 \leq i < \ell$, $\text{NextVar}(\sigma_i)$ is in a constraint c where $\text{vbl}(c) \cap \text{vbl}(c') \neq \emptyset$ for some $c' \in C_{\star\text{-bad}}^{\sigma_{\ell}}$. Combining with $|\text{vbl}(c)| \leq k$, we have

$$\begin{aligned} \ell &\leq k \cdot |\{c \in C : \text{vbl}(c) \cap \text{vbl}(c') \neq \emptyset \text{ for some } c' \in C_{\star\text{-bad}}^{\sigma_{\ell}}\}| \\ &\leq k\Delta \cdot |C_{\star\text{-bad}}^{\sigma_{\ell}}|. \end{aligned}$$

Now we prove the claim. Note that by $\text{Path}(\sigma) = (\sigma_0, \dots, \sigma_{\ell})$ and Definition 6.10, we have $\text{NextVar}(\sigma_i) \neq \perp$ for each $0 \leq i < \ell$. Assume that $\text{NextVar}(\sigma_i) = u_i$. By Definition 3.6, we have $u_i \in V_{\star\text{-inf}}^{\sigma_i} \neq \emptyset$. Combining with the definition of $V_{\star\text{-inf}}^{\sigma_i}$, we have there exists some $c_i \in C^{\sigma_i}$, $w_i \in V_{\star}^{\sigma_i}$ such that $u_i, w_i \in \text{vbl}(c_i)$. By $w_i \in V_{\star}^{\sigma_i}$, we have $w_i \in V^{\sigma_i} \cap V_{\text{fix}}^{\sigma_i}$. By $w_i \in V^{\sigma_i}$, we have $w_i \notin \Lambda(\sigma_i)$. Combining with $w_i \in V_{\text{fix}}^{\sigma_i}$, we have either $\sigma_i(w_i) = \star$ or $w_i \in \widehat{c}_i$ for some $\widehat{c}_i \in C_{\text{frozen}}^{\sigma_i}$. If $\sigma_i(w_i) = \star$, we have $c_i \in C_{\star}^{\sigma_i} \subseteq C_{\star\text{-bad}}^{\sigma_i}$. Otherwise, $w_i \in \widehat{c}_i$ for some $\widehat{c}_i \in C_{\text{frozen}}^{\sigma_i}$. In addition, by $w_i \in V_{\star}^{\sigma_i}$ and $w_i \in \widehat{c}_i$, we have $\widehat{c}_i \in C_{\star\text{-con}}^{\sigma_i}$. Combining with $\widehat{c}_i \in C_{\text{frozen}}^{\sigma_i}$, we have $\widehat{c}_i \in C_{\star\text{-bad}}^{\sigma_i}$. Let

$$c'_i = \begin{cases} c_i & \text{if } \sigma_i(w_i) = \star, \\ \widehat{c}_i & \text{otherwise.} \end{cases}$$

Obviously, $c'_i \in C_{\star\text{-bad}}^{\sigma_i}$ and $\text{vbl}(c'_i) \cap \text{vbl}(c_i) \neq \emptyset$. In addition, by Lemma 6.32 we have $C_{\star\text{-bad}}^{\sigma_i} \subseteq C_{\star\text{-bad}}^{\sigma_{\ell}}$. Thus, $c'_i \in C_{\star\text{-bad}}^{\sigma_i} \subseteq C_{\star\text{-bad}}^{\sigma_{\ell}}$. In summary, we have $\text{NextVar}(\sigma_i) \in \text{vbl}(c_i)$, $c'_i \subseteq C_{\star\text{-bad}}^{\sigma_{\ell}}$, and $\text{vbl}(c_i) \cap \text{vbl}(c'_i) \neq \emptyset$. Then the claim is immediate.

In the following, we show that there exists a $\{2, 3\}$ -tree $T \subseteq C_{\star\text{-bad}}^{\sigma_{\ell}}$ in $G(C)$ such that $v \in \text{vbl}(T)$ and $\Delta|T| \geq |C_{\star\text{-bad}}^{\sigma_{\ell}}|$. Then the lemma is immediate. Fix a constraint c where $v \in \text{vbl}(c)$. By the definition of $\text{Path}(\sigma)$ in Definition 6.10, we have $\sigma_0 = \sigma$. Combining with $\sigma(v) = \star$, we have $\sigma_0(v) = \star$. Combining with the definition of $C_{\star}^{\sigma_0}$, we have $c \in C_{\star}^{\sigma_0}$ by $v \in \text{vbl}(c)$. By $c \in C_{\star}^{\sigma_0}$ and $C_{\star}^{\sigma_0} \subseteq C_{\star\text{-bad}}^{\sigma_0}$, we have $c \in C_{\star\text{-bad}}^{\sigma_0}$. In addition, by Lemma 6.32, we have $C_{\star\text{-bad}}^{\sigma_0} \subseteq C_{\star\text{-bad}}^{\sigma_{\ell}}$. Thus we have $c \in C_{\star\text{-bad}}^{\sigma_{\ell}}$. Thus, by Lemmas 6.24 and C.1, we have such a $\{2, 3\}$ -tree T in $G(C)$, where $v \in \text{vbl}(T)$, $T \subseteq C_{\star\text{-bad}}^{\sigma_{\ell}}$ and $\Delta|T| \geq |C_{\star\text{-bad}}^{\sigma_{\ell}}|$, exists. This completes the proof. □

C.2. Proof of Lemma 6.35. Recall that $X = X^n$ where X^0, X^1, \dots, X^n is the partial assignment sequence of Algorithm 1 in Definition 5.6. Then we have the following lemma.

Lemma C.2. $V \setminus \Lambda(X) \subseteq \text{vbl}(C_{\text{frozen}}^X)$.

Proof. Given $v_i \in V \setminus \Lambda(X)$ where $i \in [n]$, by Lines 2-4 of Algorithm 1 and Theorem 5.5, we have $v_i \in V_{\text{fix}}^{X^{i-1}}$, otherwise, v_i will be assigned a value from Q_{v_i} in Line 4. Moreover, again by Lines 2-4 of Algorithm 1, we also have $v_i \notin \Lambda^+(X^{i-1})$. Combining with $v_i \in V_{\text{fix}}^{X^{i-1}}$, we have $v_i \in C_{\text{frozen}}^{X^{i-1}}$. In addition, similar to Lemma B.2, one can also prove that $C_{\text{frozen}}^{X^{j-1}} \subseteq C_{\text{frozen}}^{X^j}$ for each $j \in [n]$. Then we have $C_{\text{frozen}}^{X^{i-1}} \subseteq C_{\text{frozen}}^X$ by induction. Combining with $v_i \in C_{\text{frozen}}^{X^{i-1}}$, we have $v_i \in \text{vbl}(C_{\text{frozen}}^X)$. Then the lemma is immediate. \square

Now we can prove Lemma 6.35.

Proof of Lemma 6.35. Let $\{H_i^X = (V_i^X, C_i^X) \mid 1 \leq i \leq K\}$ be the connected components in H^X constructed in Line 1 Algorithm 2. If $v \notin V_i$ for each $i \in [K]$, we have $C_v^X = \emptyset$ and the lemma is trivial. In the following, we assume w.l.o.g. that $v \in V_i^X$ for some $i \in K$. Then we have $H_v^X = (V_i^X, C_i^X)$. Let

$$S \triangleq \{c \in C_{\text{frozen}}^X \mid c^X \in C_i^X\}.$$

At first, we prove that there exists some $c_v \in S$ such that $v \in \text{vbl}(c_v)$. By $v \in V_i^X$, we have $v \notin \Lambda(X)$. Combining with Lemma C.2, we have there exists some $c_v \in C_{\text{frozen}}^X$ such that $v \in \text{vbl}(c_v)$. In addition, by $v \in \text{vbl}(c_v)$ and $c_v \in C_{\text{frozen}}^X$, we also have $c_v^X \in C_i^X$. Combining with $c_v \in C_{\text{frozen}}^X$, we have $c_v \in S$.

Now we prove $|C_i^X| \leq \Delta |S|$. For each $c^X \in C_i^X$, we have there exists a connected path $c_1^X, c_2^X, \dots, c_t^X = c^X \in C_i^X$ such that $v \in \text{vbl}(c_1^X)$. Let $v' \in \text{vbl}(c^X)$. We have $v' \notin \Lambda(X)$. Combining with Lemma C.2, we have $v' \in \text{vbl}(\widehat{c})$ for some $\widehat{c} \in C_{\text{frozen}}^X$. Then we have $\widehat{c}^X \in C_i^X$ because there exists a connected path $c_1^X, c_2^X, \dots, c_t^X, \widehat{c}^X \in C_i^X$ where $v \in \text{vbl}(c_1^X)$. Combining $\widehat{c} \in C_{\text{frozen}}^X$ with $\widehat{c}^X \in C_i^X$, we have $\widehat{c} \in S$. In summary, for each $c^X \in C_i^X$, there exists some $\widehat{c} \in S$ such that $\text{vbl}(c^X) \cap \text{vbl}(\widehat{c}^X) \neq \emptyset$. Thus, we have $|C_i^X| \leq \Delta |S|$.

In the next, we prove that $G^2(S)$ is connected. It is enough to prove that any two different constraints $c, \widehat{c} \in S$ are connected in $G^2(S)$. Given $c, \widehat{c} \in S$, we have c^X, \widehat{c}^X are in C_i^X . Therefore, we have there exists a connected path $c^X = c_1^X, c_2^X, \dots, c_t^X = \widehat{c}^X \in C_i^X$. If $t \leq 3$, obviously c and \widehat{c} are connected in $G^2(S)$. In the following, we assume that $t > 3$. Let $w_j \in (\text{vbl}(c_j^X) \cap \text{vbl}(c_{j+1}^X))$ for each $j < t$. Then we have $w_j \notin \Lambda(X)$. Combining with Lemma C.2, we have $w_j \in \text{vbl}(\widehat{c}_j^X)$ for some $\widehat{c}_j \in C_{\text{frozen}}^X$. Moreover, we also have $\widehat{c}^X \in C_i^X$, because \widehat{c}^X is connected to c^X through $c_2^X, \dots, c_j^X \in C_i^X$. Thus, we have $\widehat{c}_j \in S$. In addition, for each $\widehat{c}_j, \widehat{c}_{j+1}$ where $j < t-1$, we have \widehat{c}_j and \widehat{c}_{j+1} are connected in $G^2(C)$, because $w_j \in \text{vbl}(\widehat{c}_j) \cap \text{vbl}(c_{j+1})$ and $w_{j+1} \in \text{vbl}(\widehat{c}_{j+1}) \cap \text{vbl}(c_{j+2})$. Thus, the constraints $c = c_1, \widehat{c}_1, \widehat{c}_2, \dots, \widehat{c}_{t-1}, c_t = \widehat{c}$ forms a connected path in $G^2(C)$. Combining with $\widehat{c}_j \in S$ for each $j \leq t-1$ and $c, \widehat{c} \in S$, we have the constraints c, \widehat{c} are connected in $G^2(S)$.

In summary, we have $c_v \in S \subseteq C_{\text{frozen}}^X$, $\Delta |S| \geq |C_i^X|$ and $G^2(S)$ is connected. Combining with Lemma 6.24, we have there exists a $\{2, 3\}$ -tree $T \subseteq S \subseteq C_{\text{frozen}}^X$ such that $c_v \in T$ and

$$|T| \geq |S| / \Delta \geq |C_i^X| / \Delta^2 = |C_v^X| / \Delta^2.$$

Then the lemma is immediate. \square