Generating from Discrete Distributions Using Diffusions: Insights from Random Constraint Satisfaction Problems

Alankrita Bhatt[1] *, **Mukur Gupta**[1] *, **Germain Kolossov**[1] *, **and Andrea Montanari**[1,2] *
[1]Granica Computing Inc. – granica.ai
[2]Stanford University

Abstract

Generating data from discrete distributions is important for applications to text, tabular, and genomic data. Several groups have adopted random k-satisfiability (k-SAT) as a synthetic benchmark for generative techniques. We show that fundamental insights from the theory of random constraint satisfaction problems (CSPs) have observable (sometimes counterintuitive) implications on the behavior of generative techniques on such benchmarks. More precisely, we study the problem of generating a uniformly random solution of a given (random) CSP. We observe that: (i) Continuous diffusions outperform masked discrete diffusions; (ii) Learned diffusions can match the predicted 'ideal' accuracy; (iii) Smart ordering of the variables can significantly improve accuracy, although not using popular heuristics.

1 Introduction

Since the introduction of denoising diffusions [1, 2, 3, 4], an impressive number of variants have emerged, in particular to generate data over discrete domains, most notably language and tabular data [5, 6, 7]. Despite this, the relative merits of different techniques for generating discrete data remain poorly understood. This is particularly true for tabular data generation [8, 9, 10], an application domain that lacks a universally representative set of benchmarks.

Several groups have recently used random k-satisfiability (k-SAT) as a synthetic benchmark for discrete data generation [11, 12, 13, 14]. However, the choice of the setting and metrics in these works is somewhat arbitrary, and not guided by fundamental principles. In particular, there is no notion of what is the target accuracy the generative model should strive for, and no guidance about the parameters of the k-SAT distribution to be used.

The objective of this paper is twofold. *First,* we will show that the theory of random constraint satisfaction problems (CSPs) can provide guidance about the specific setting (specific CSP and parameters) as well as target accuracy for benchmarking generative techniques. *Second,* we will show that insights from that theory are predictive of the behavior of diffusion sampling techniques, leading to conclusions that can contradict naive intuition.

The present work builds upon the wealth of mathematical understanding of random CSPs that has been developed over the last twenty years, through an interdisciplinary effort across computer science, statistical physics, and probability theory, see [15, 16, 17, 18, 19] for a few pointers. Recently, methods from this literature were used for construct algorithms that provably sample from high-dimensional and highly multi-modal Gibbs measures arising from spin glass models or random CSPs [20, 21, 22, 23, 24] (see also [25] for an earlier example). These algorithms implement denoising diffusions or discrete diffusions, with the difference that the denoiser is constructed explicitly by

*Authors are listed in alphabetical order. Correspondence to {first_name.second_name}@granica.ai

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

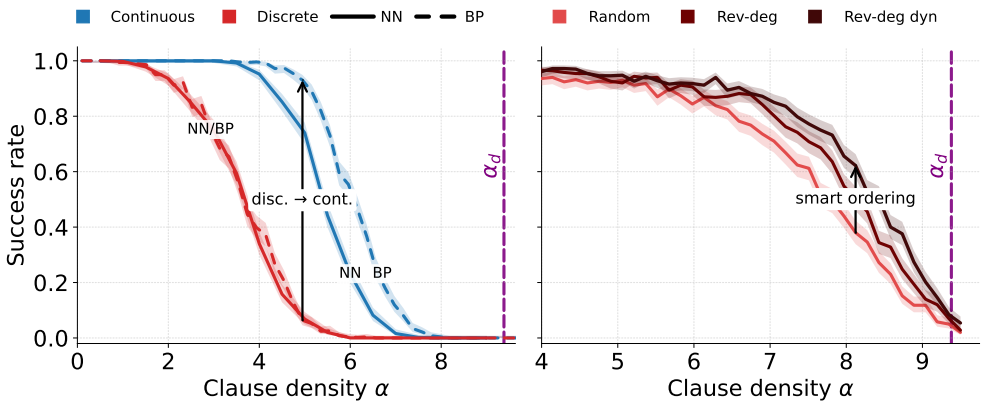


Figure 1: **4-SAT** ($N = 300$) Left: The learned NN denoiser (solid) matches the BP denoiser (dashed), and continuous diffusion (blue) outperforms discrete diffusion (red) for both denoisers at locality radius $r = 3$. Right: Smart variable ordering improves discrete diffusion with $r = 100$ BP denoiser. We compare random ordering with reverse-degree and dynamic reverse-degree orderings. We report success rate (probability of producing a satisfying assignment) versus clause density α, averaged over 500 random formulas. Shaded regions: 95% confidence interval. Vertical line: dynamical phase transition $\alpha_{\rm d}(k = 4) \approx 9.38$.

using (variants of) loopy belief propagation (BP). For large random instances, loopy BP is expected to provide a near-optimal denoiser among polynomial-time algorithms.

Here, we take this line of work one step forward, by investigating learned diffusions and comparing them with the behavior predicted from the theory for BP-based diffusions.

An instance of a binary CSP is given by a set of constraints over the vector $\boldsymbol{x} \in \{+1, -1\}^N$. A solution of a specific CSP instance is a vector $\boldsymbol{x}$ that satisfies all the constraints. We attempt to generate samples from the uniform measure over solutions of a given CSP, using both standard denoising diffusions and masked discrete diffusions. For each of these two approaches and each of four families of random CSPs (k-SAT, k-XORSAT, k-NAESAT and hypergraph bicoloring), we study the behavior of generative methods based on several learned and BP-based denoisers. (We use the term 'denoiser' broadly: for masked discrete diffusions this is the conditional probability model.)

We succinctly summarize some of our findings below, and refer to Figure 1 for an illustration:

1. Continuous diffusions outperform masked discrete diffusions. One of the main motivation for the introduction of discrete diffusions was the idea that they are better adapted to discrete data [6, 5]. We show that this is not the case for random CSPs, in agreement with asymptotic theory [25, 23].

2. Ideal target accuracy from random CSP theory. We mostly focus on 'local' denoisers, i.e. denoisers that estimate each variable using a bounded radius neighborhood of that variable in the graph G which represent the CSP instance. We derive a prediction for the ideal 'local' denoiser, i.e. the denoiser that minimizes the score matching objective in the large size, large sample size limit.

3. Learned diffusion match theoretical target. We show empirically that learned local denoisers can match the behavior of the ideal denoisers at the previous point.

4. Accuracy is improved by smart ordering (continuous and discrete). We study a more general class of generative methods which treats different coordinates x_i in a different way. In the case of discrete diffusions, this corresponds to unmasking variables in a smart order. For continuous diffusions, this corresponds to adding noise with different variances to different variables. We construct schemes that significantly improve the success rate of both discrete and continuous diffusions.

2 Setting and background

We consider four families of random CSPs with binary decision $\boldsymbol{x} = (x_1, \dots, x_N) \in \{+1, -1\}^N$: k-SAT, k-XORSAT, k-NAE-SAT, and k-hypergraph bicoloring. An instance is defined by a set M constraints over the N variables. Each constraint ('clause') comprises k variables (or their negation).

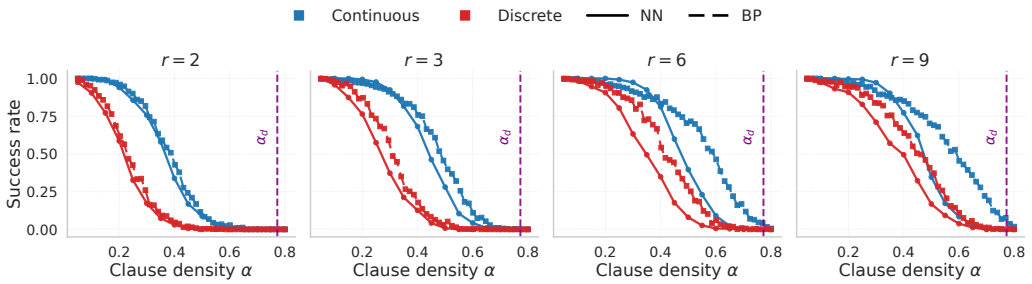


Figure 2: **4-XORSAT ($N = 100$): learned NN denoiser (solid lines) vs. BP denoiser (dashed lines).** Success rate as a function of clause density α for continuous (blue) and discrete (red) diffusions. Each panel corresponds to a different locality radius for the denoisers $r \in \{2, 3, 6, 9\}$. Success rates are computed over 500 random formulas. The vertical line corresponds to the dynamical phase transitoion at $\alpha_{\rm d}(k = 4) \approx 0.77228$.

k-SAT. Constraint a takes the form $(z_{i_a(1)} \vee z_{i_a(2)} \vee \cdots \vee z_{i_a(k)}) = +1$ where $i_a(1), \dots, i_a(k) \subseteq [N]$ are distinct, $z_{i_a(j)} = s_{a,j} x_{i_a(j)}$ with $s_{a,j} \in \{+1, -1\}$ and $x \vee x' = \max(x, x')$ (logical OR in the True/False encoding).

k-XORSAT. Constraint a takes the form $x_{i_a(1)} \cdot x_{i_a(2)} \cdots x_{i_a(k)} = s_a$, where $s_a \in \{+1, -1\}$, and $\cdot$ denotes simple product.

Because of space limitations, we defer definitions and results for **NAE-SAT**, and **hypergraph bicoloring** to the appendices. (Our results for SAT and XORSAT are more complete, and confirmed by the additional examples in the appendices.) A random k-SAT/XORSAT formula is generated by drawing each of the M constraints independently and uniformly at random among the $2^k\binom{N}{k}$ possible ones (for k-SAT) or $2\binom{N}{k}$ possible ones (for k-XORSAT). We will denote by $\partial a := \{i_a(1), \dots, i_a(k)\}$ the set of variable indices appearing in constraint a.

k-SAT and k-XORSAT belong to different complexity classes. Because of the linear-algebraic structure, a XORSAT solution can be found in polynomial time, and indeed it is easy to sample a uniformly random solution. In contrast, even simply finding a k-SAT is NP complete [26]. We emphasize that XORSAT is a challenging testing ground nevertheless because: (i) We limit ourselves to local algorithms for which SAT and XORSAT are much more similar; (ii) Denoising diffusions require solving a denoising problem that is worst-case hard also for XORSAT [27] .

An instance of k-SAT or k-XORSAT can be encoded in a factor graph $G = (V, F, E)$, with variable nodes $V = [N]$ (corresponding to the N variables), factor nodes F, $|F| = M$ (corresponding to the M constraint), and edges (i, a), $i \in V$, $a \in F$, every time variable x_i appears in constraint a. G is a marked graph, with marks on the edges carrying information of the signs $s_{a,j}$, s_a. Given an instance G, we are interested in the uniform measure over solutions

$$\mu_G(\boldsymbol{x}) = \frac{1}{Z_G}\mathbf{1}_{\{\boldsymbol{x} \text{ solves } G\}} = \frac{1}{Z_G}\prod_{a\in F}\psi_a(\boldsymbol{x}_{\partial a})\,, \tag{1}$$

where Z_G is the number of solution of formula G, and $\psi_a(\boldsymbol{x}_{\partial a}) = 1$ if $\boldsymbol{x}$ satisfies the a-th constraint, while $\psi_a(\boldsymbol{x}_{\partial a}) = 0$ otherwise. We consider two sampling schemes.

Denoising diffusions. We update a vector $\boldsymbol{Y}_\ell \in \mathbb{R}^N$, $\ell \in \{0, 1, \dots, L\}$, with initialization $\boldsymbol{Y}_0 \sim \mathsf{N}(0, \boldsymbol{I}_N)$, and proceeding according to the Euler discretization [3, 2]

$$\boldsymbol{Y}_{\ell+1} = \gamma_\ell \boldsymbol{Y}_\ell + \delta_\ell \boldsymbol{W}\,\boldsymbol{m}\big(\boldsymbol{Y}_\ell; t_\ell\big) + \sqrt{\beta_\ell}\boldsymbol{g}_\ell\,, \tag{2}$$

where $(\boldsymbol{g}_\ell)_{\ell\ge 0} \sim_{iid} \mathsf{N}(\boldsymbol{0}, \boldsymbol{I}_N)$, $\beta_\ell \in (0, 1)$ are stepsizes, $t_\ell = \prod_{i=\ell}^{L-1}(1-\beta_i)$, $\gamma_\ell = (1 - \beta_\ell/(1 - t_\ell))/\sqrt{1-\beta_\ell}$, $\delta_\ell = \beta_\ell\sqrt{t_\ell}/((1-t_\ell)\sqrt{1-\beta_\ell})$. (Here ℓ increases along the generation process.)

Further, $\boldsymbol{W}$ is a matrix which we will take diagonal and non-negative, and can be interpreted as a weighting scheme determining the order in which information about variables is revealed. Finally, $\boldsymbol{m}(\,\cdot\,)$ is the denoiser. The ideal denoiser $\boldsymbol{m}(\boldsymbol{y}; t)$ is the posterior mean given observations

$\boldsymbol{y} = \sqrt{t}\boldsymbol{W}\boldsymbol{x} + \sqrt{1-t}\,\boldsymbol{g}$, $\boldsymbol{g} \sim \mathsf{N}(0, \boldsymbol{I}_N)$. This coincides with the mean of the tilted measure

$$\mu_G^{\boldsymbol{y},t}(\boldsymbol{x}) = \frac{1}{Z_G(t)} \prod_{a\in F} \psi_a(\boldsymbol{x}_{\partial a})\, e^{\sqrt{t}\langle \boldsymbol{y}, \boldsymbol{W}\boldsymbol{x}\rangle/(1-t)}\,, \tag{3}$$

for a suitable normalizing constant $Z_G(t)$. A key observation is that the tilted measure (3) factors according to the same graph $G = (V, F, E)$ as the original measure μ_G. In particular, we can employ BP to attempt to approximate the marginals of $\mu_G^{\boldsymbol{y},t}$ and hence the denoiser $\boldsymbol{m}(\,\cdot\,)$.

Discrete masked diffusions. It is useful to introduce the notation $\boldsymbol{x}_{a:b} = (x_a, x_{a+1}, \dots, x_b)$ for $a \le b$ (and $\boldsymbol{x}_{:b} = \boldsymbol{x}_{1:b}$, $\boldsymbol{x}_{a:} = \boldsymbol{x}_{a:N}$). Given an arbitrary permutation $\sigma : [N] \to [N]$ of the variables, we let $\boldsymbol{x}^\sigma = (x_{\sigma(1)}, \dots, x_{\sigma(N)})$ be the variables ordered according to σ, and

$$\mu_G^{\sigma,\ell}(\boldsymbol{x}|\overline{\boldsymbol{x}}^\sigma_{1:\ell}) = \mathbb{P}_{\boldsymbol{X}\sim\mu_G}\big(\boldsymbol{X} = \boldsymbol{x}\big|\boldsymbol{X}^\sigma_{1:\ell} = \overline{\boldsymbol{x}}^\sigma_{1:\ell}\big)\,, \tag{4}$$

the conditional distribution given assignments of the first ℓ variables to $\overline{\boldsymbol{x}}^\sigma_{1:\ell}$. With a slight abuse of notation, we also write $\mu_G^{\sigma,\ell}(x^\sigma_{\ell+1}|\boldsymbol{x}^\sigma_{1:\ell})$ for the conditional distribution of $x^\sigma_{\ell+1}$ given $\overline{\boldsymbol{x}}^\sigma_{1:\ell}$. We then sequentially sample variables by repeating for $\ell \in \{0, \dots, N-1\}$, i.e. $\overline{x}_{\sigma(\ell+1)} \sim \mu_G^{\sigma,\ell}(\,\cdot\,|\overline{\boldsymbol{x}}^\sigma_{1:\ell})$. In the following, the order of variables σ will be chosen to be purely as a function of the instance G.

We note that the conditional distribution (4) also factors according to the graph G, namely

$$\mu_G^{\sigma,\ell}(\boldsymbol{x}|\overline{\boldsymbol{x}}^\sigma_{1:\ell}) = \frac{1}{Z_{G,\ell}} \prod_{a\in F} \psi_a(\boldsymbol{x}_{\partial a}) \prod_{i=1}^{\ell} \mathbf{1}_{x_{\sigma(i)}=\overline{x}_{\sigma(i)}}\,. \tag{5}$$

Local denoisers. Implementing either continuous or discrete diffusions requires to compute marginals of probability distributions that factor according to the graph G, respectively, the distribution $\nu(\boldsymbol{x}) = \mu_G^{\boldsymbol{y},t}(\boldsymbol{x})$ of Eq. (3) or $\nu(\boldsymbol{x}) = \mu_G^{\sigma,\ell}(\,\cdot\,|\overline{\boldsymbol{x}}^\sigma_{1:\ell})$ of Eq. (5). We refer to any any function that take as input $(G, \boldsymbol{y}, t)$ (in the first case) or $(G, \sigma, \overline{\boldsymbol{x}}^\sigma_{1:\ell})$ (in the second) and computes the marginal $\nu(x_i \in \,\cdot\,)$ of this distribution as to a 'denoiser' (hence this term refers both to continuous and discrete diffusions). We will focus on denoisers that are r-*local* with respect to G, in the sense that the marginal at $i \in V$ is estimated by a function of the radius-r neighborhood of i.

Namely, let $\mathsf{B}_G(i, r)$ be the subgraph of G induced by vertices $j \in V$ such that $d_G(i, j) \le r$ (with d_G the graph distance, s.t. if $i, j \in V$ belong to the same constraint $d_G(i, j) = 1$). A denoiser is r-local if the marginal computed at $i \in V$ only depends on $(G, \boldsymbol{y})$ or $(G, \overline{\boldsymbol{x}}^\sigma_{1:\ell})$ via their restriction to $\mathsf{B}_G(i, r)$. We denote the information in such a neighborhood by $\boldsymbol{Y}_{\mathsf{B}_G(i,r)}$, e.g., for continuous diffusions, $\boldsymbol{Y}_{\mathsf{B}_G(i,r)} := (i, \mathsf{B}_G(i, r), \boldsymbol{y}|_{\mathsf{B}_G(i,r)})$. We will use two classes of local denoisers:

(i) BP denoiser. Since the probability distributions $\mu_G^{\boldsymbol{y},t}(\boldsymbol{x})$ and $\mu_G^{\sigma,\ell}(\,\cdot\,|\overline{\boldsymbol{x}}^\sigma_{1:\ell})$ factorize according to the graph G, we can use (loopy) BP to approximate their marginals, see Appendix C for details. Provided we use an initialization that is independent of G, r iterations of BP result in an r-local denoiser. We will consider two initialization 'uniform' and 'cavity,' and denote the corresponding denoisers by $\mathrm{BP}_{\mathsf{u}}(r)$, $\mathrm{BP}_{\mathsf{c}}(r)$. In the first one, we initialize all beliefs to be uniform on $\{+1, -1\}$, while the second one will be defined in Section 3.2.

(ii) Neural Network (NN) denoiser. We construct a neural network architecture which takes as input the radius r neighborhood of an arbitrary vertex i and information revealed within such a neighborhood, and returns the predicted logit of variable x_i: $f(\,\cdot\,;\boldsymbol{\theta}) : \boldsymbol{Y}_{\mathsf{B}_G(i,r)} \mapsto f(\boldsymbol{Y}_{\mathsf{B}_G(i,r)};\boldsymbol{\theta}) \in \mathbb{R}$.

Such a model also induces a predicted conditional expectation for variable i given information $\boldsymbol{Y}_{\mathsf{B}_G(i,r)}$, via $\hat{m}(\boldsymbol{Y};\boldsymbol{\theta}) = \tanh(f(\boldsymbol{Y};\boldsymbol{\theta})/2)$. This corresponds to a unique predicted probability distribution ν_i on $\{+1, -1\}$. Our architecture has parameters $\boldsymbol{\theta}$ and is –by construction– equivariant with respect to relabeling of the vertices. We refer to Appendix A for a description of this architecture.

Satisfiability and dynamical phase transitions. As $N, M \to \infty$, with constant number of constraints per variable $M/N = \alpha$, the space of solutions of random k-SAT/XORSAT undergoes certain phase transitions[2]. Two phase transitions at locations $\alpha_{\mathsf{d}} < \alpha_{\mathsf{s}}$ are particularly relevant (their location differ between SAT and XORSAT, but we omit this dependence unless useful):

[2]This picture is rigorously established for k-XORSAT [28, 18] and, in a sequence of recent mathematical breakthroughs [17, 29], for k-SAT at k above a certain constant. It is still conjectural for –say– $k = 3, 4$, see [19] the conjectured picture.

(1) The *satisfiability phase transition* at $\alpha_{\rm s}(k)$: for $\alpha < \alpha_{\rm s}(k)$ a random instance G has solutions with high probability (whp, i.e. with probability converging to one as $N, M \to \infty$); for $\alpha > \alpha_{\rm s}(k)$ it does not, whp. (For $\alpha < \alpha_{\rm s}(k)$, the number of solutions is exponentially large in N.)

(2) The *dynamical (or, shattering) phase transition* at $\alpha_{\rm d}(k)$: for $\alpha < \alpha_{\rm d}(k)$ the set of solutions $\mathcal{S} \subseteq \{\pm 1\}^N$ is whp 'well connected' while for $\alpha > \alpha_{\rm d}(k)$ it comprises an exponential number of components $\mathcal{S} = \cup_{j=1}^{L} \mathcal{S}_j$. Any two distinct components have Hamming distance $d(\mathcal{S}_i, \mathcal{S}_j) \geq cN$.

The dynamical phase transition is particularly important because it is expected that no polynomial time algorithm can sample k-SAT solutions for $\alpha \in (\alpha_{\rm d}(k), \alpha_{\rm s}(k))$ [16, 30]. For k-XORSAT, sampling is always tractable exploiting the linear-algebraic structure, but local algorithms are believed to fail for $\alpha \in (\alpha_{\rm d}(k), \alpha_{\rm s}(k))$. Our objective will be to sample solutions for $\alpha \in (0, \alpha_{\rm d}(k))$.

Phase transitions for diffusions. For a diffusion-based sampler to be successful, it needs to have access to an oracle that compute marginals with respect to tilted measure $\mu_G^{\boldsymbol{y},t}(\,\cdot\,)$ of Eq. (3) (for continuous diffusions) or $\mu_G^{\sigma,\ell}(\,\cdot\,)$ of Eq. (5) (for discrete diffusions). As a consequence, if marginal computation (or –equivalently– sampling) is intractable for these measures, then diffusion sampling fails, see [31] for a formal statement.

The measure $\mu_G^{\sigma,\ell}(\,\cdot\,)$ is just the uniform measures over the set of solutions of a modified random CSP obtained by constraining some of the variabled to be equal to the 'observed' values. This modified CSP undergoes a dynamical phase transition $\alpha_{\rm d}(k;t)$. It was conjectured in [25] that discrete diffusions will succeed only if $\alpha \leq \alpha_{\rm mask}(k) := \inf_{t\in[0,1]} \alpha_{\rm d}(k;t)$. This threshold was computed in [25] for SAT, while for XORSAT the computation of $\alpha_{\rm mask}(k)$ can be carried out similarly: we have $\alpha_{\rm mask}^{\rm XOR}(4) \approx 0.5625$, $\alpha_{\rm mask}^{\rm SAT}(4) \approx 8.05$. A similar threshold can be computed for continuous diffusions with $\boldsymbol{W} = \boldsymbol{I}_N$, yielding in particular $\alpha_{\rm diff}^{\rm XOR}(4) \approx 0.63$ (Appendix C). While a complete proof of this conjecture is still lacking, the positive direction was proven in a related spin-glass model in [22].

3 Main results

We will compare generative methods both in theory and practice across three different axes: learned vs BP denoisers; discrete vs continuous diffusions; uniform vs smart variable ordering. We will mostly report the probability that they generate solutions ('success rate'). This is a relatively strict criterion since a single incorrect variable can result in a failure (part of previous work considered the fraction of satisfied constraints, a weaker metric.) Since high success rate does not guarantee the generated solutions to be uniformly distributed, we will test approximate uniformity in Section 3.4.

We emphasize that BP denoisers do not require training. For NN denoisers, we train on solutions/noisy observation pairs $\boldsymbol{x}^{(s)}, \boldsymbol{y}^{(s,\ell)}$. In the case of XORSAT it is easy to generate uniform solutions using linear-algebraic methods. For other CSPs, we experimented with several heuristics to generate solutions, as discussed in Appendix B. As already mentioned, we defer a presentation of results for NAE-SAT and hypergraph bicoloring to Appendix G.

We train r-local NN models $f(\,\cdot\,;\boldsymbol{\theta})$, by minimizing $\hat{R}_n(\boldsymbol{\theta}) = \frac{1}{NL}\sum_{\ell\leq L, i\leq N} \hat{\mathbb{E}}_{n,\ell} \mathcal{L}\big(x_i, f(\boldsymbol{Y}_{\mathsf{B}_G(i,r)};\boldsymbol{\theta})\big)$. For continuous diffusions we use the square loss $\mathcal{L}(x,f) = (x - \tanh(f/2))^2$ (recall that the conditional mean is related to the log-likelihood via $\hat{m} = \tanh(f/2)$), while for discrete masked diffusions we use the cross-entropy $\mathcal{L}(x,f) = -xf/2 + \log\cosh(f/2)$. Here $\boldsymbol{Y}_{\mathsf{B}_G(i,r)} = (i, \mathsf{B}_G(i,r), \boldsymbol{y}|_{\mathsf{B}_G(i,r)})$ is the entire local information available to the denoiser, $\hat{\mathbb{E}}_{n,\ell}$ denotes empirical expectation with respect to the training samples $(G^{(s)}, \boldsymbol{x}^{(s)}, \boldsymbol{y}^{(s,\ell)})$, $s \in \{1, \dots, n_{\rm s}\}$, where $G^{(s)}$ is a random k-SAT/XORSAT instance, $\boldsymbol{x}^{(s)}$ is a random solution of $G^{(s)}$, and $\boldsymbol{y}^{(s,\ell)} = \sqrt{t_\ell}\boldsymbol{x}^{(s)} + \sqrt{1-t_\ell}\boldsymbol{g}^{(s,\ell)}$, $\boldsymbol{g}^{(s,\ell)} \sim \mathsf{N}(0, \boldsymbol{I}_N)$ (for continuous diffusions) or $\overline{\boldsymbol{x}}_{1:\ell}^{\sigma}$ (for discrete diffusions).

3.1 Continuous outperform discrete diffusions; NN matches BP

In Figures 1 (left frame) and 2 we compare the success rate of continuous and discrete diffusions, both using BP and NN denoisers, respectively for 4-SAT (with $N = 300$) and 4-XORSAT (with $N = 100$). In Figure 2, each panel corresponding to a different locality radius (see Appendix for the

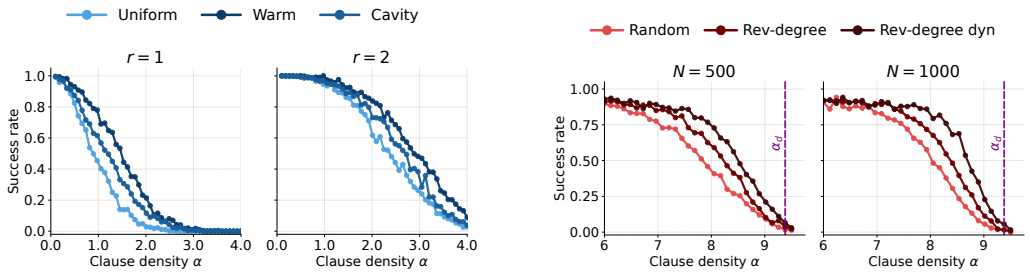


Figure 3: **4-SAT. Two left panels:** Effect of BP message initialization in discrete diffusion. We compare zero, warm-start, and cavity-based initializations for $N = 300$ using a BP denoiser with radius $r = 1$ (*left*) and $r = 2$ (*right*). **Two right panels:** Effect of variable ordering in discrete diffusion with a BP denoiser. We compare random, reverse min-degree, and dynamic reverse min-degree orderings for $N = 500$ (*left*) and $N = 1000$ (*right*) with $r = 100$. Each panel reports success rate versus clause density α, averaged over 500 random formulas; the vertical line marks the dynamical phase transition $\alpha_{\rm d}(k = 4) \approx 9.38$.

analogous comparison for SAT). We observe that continuous diffusions systematically outperforms discrete diffusions.

The non-rigorous theoretical predictions of [25, 23] imply sharp thresholds for discrete/continuous diffusion sampling at $\alpha^{\rm SAT}_{\rm mask}(4) < \alpha^{\rm SAT}_{\rm diff}(4)$ for 4-SAT and $\alpha^{\rm XOR}_{\rm mask}(4) < \alpha^{\rm XOR}_{\rm diff}(4)$ for 4-XOR. This theory applies to the BP denoiser under the asymptotics $N, M \to \infty$ as well as $r \to \infty$. Our results demonstrate that: (a) The asymptotic theory is relevant at moderate N, M, r; (b) The predictions for BP denoisers also capture the behavior of learned NN denoisers; (c) The success rate of learned NN denoisers nearly matches the one achieved with BP denoisers.

The last point is higly non-trivial, given that the NN architecture is only aware of the permutation invariance and locality structure of the formula G. We observe a bigger gap between BP and NN denoisers in the case of XORSAT than SAT. This appears to be related to the fact that it is harder to train a (see Appendix B.1 for more details.)

3.2 Ideal target accuracy from random CSP theory

Random CSP formulas are locally tree-like. Namely, for any fixed r, the neighborhood $\mathsf{B}_G(i, r)$ centered at a uniformly random vertex $i \sim \mathsf{Unif}([N])$ is a tree with probability converging to one, as $N, M \to \infty$, $N/M = \alpha$. Since BP computes correct marginals on trees, the output of $\mathrm{BP}_{\mathsf{u}}(r)$ at node i, is exactly the marginal of the modification of measures Eqs. (3), (4) obtained by keeping only terms in the neighborhood $\mathsf{B} := \mathsf{B}_G(i, r)$.

One might expect that therefore $\mathrm{BP}_{\mathsf{u}}(r)$ is asymptotically optimal among r-local denoisers.

In Figure 3 we compare the success rate of $\mathrm{BP}_{\mathsf{u}}(r)$ with the one of $\mathrm{BP}_{\mathsf{c}}(r)$, which differs from $\mathrm{BP}_{\mathsf{u}}(r)$ uniquely in the initialization. Recall that one BP message is associated to each edge of the factor graph (V, F, E), each message being a probability distribution over $\{+1, -1\}$ (equivalently, a likelihood ratio). $\mathrm{BP}_{\mathsf{c}}(u)$ initializes message to be i.i.d. from the asymptotic distribution of the root marginal of an equivalent tree model. This distribution be approximately sampled numerically by using a sampled distributional recursion method (a.k.a. 'population dynamics' [32]). We refer to Appendix H for explicit definitions and algorithms.

The next theorem establishes that this 'cavity' initialization is optimal (among local algorithms) with respect to the score matching objective of Eq. (3). We refer to the appendix for background on the local convergence assumption: suffices to say that this is expected to hold for $\alpha \in [0, \alpha_{\rm c}(k, t))$, where the 'condensation phase transition' satisfies $\alpha_{\rm c}(k, t) \geq \alpha_{\rm d}(k, t)$ [17].

Theorem 3.1. *For any $r \in \mathbb{N}$, $\varepsilon > 0$, $t \in (0, 1)$, let $\hat{m}_{\mathrm{BP}_{\mathsf{c}}(r)}(\boldsymbol{Y}_{\mathsf{B}_G(i,r)})$ be $\mathrm{BP}_{\mathsf{c}}(r)$ estimate of the conditional expectation at node $i \in V$, under the tilted distribution of Eq. (3). If the measure $\mu_G^{\boldsymbol{y},t}$ converges locally to the free boundary Gibbs measure under $N, M \to \infty$, $N/M = \alpha$, then we have*

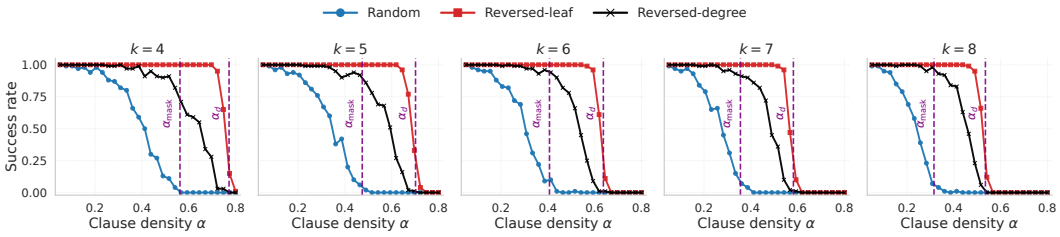


Figure 4: k**-XORSAT** ($N = 300$)**:** reversed-leaf (red) vs reversed-degree (black) vs random (blue) decoding ordering. Success rate as a function of clause density α for $k \in \{4, \dots, 8\}$ obtained using discrete diffusion with BP denoiser with $r = 300$. The two vertical lines are the theoretical thresholds for random (α_{mask}) and optimal (α_{d}) decoding ordering at every value of k. Success rates are computed over 500 random formulas.

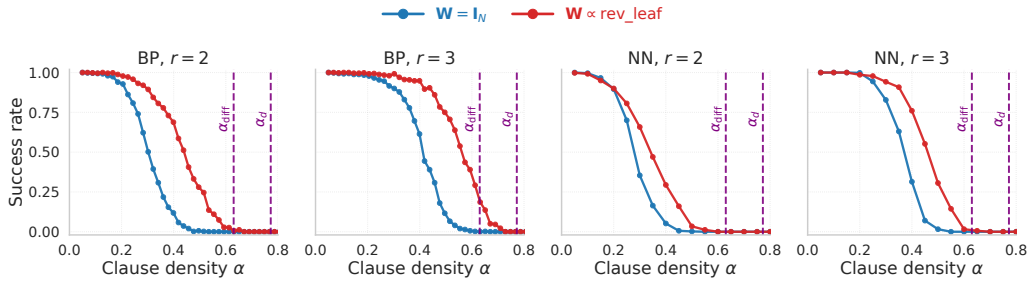


Figure 5: **4-XORSAT** ($N = 300$)**:** reversed-leaf (red) vs random (blue) ordering in continuous diffusion. Success rate as a function of clause density α obtained using $\boldsymbol{W}$ in Eq. (2), with a BP and NN denoiser. The vertical line indicates the theoretical dynamical phase-transition threshold.

(expectation is with respect to $G, \boldsymbol{x}, \boldsymbol{y}$ as well as $I \sim \mathsf{Unif}([N])$)

$$\mathbb{E}\big\{\big(x_I - \hat{m}_{\mathrm{BP}_{\mathsf{c}}(r)}(\boldsymbol{Y}_{\mathsf{B}_G(I,r)})\big)^2\big\} \le \inf_{\hat{m}(\cdot)} \mathbb{E}\big\{\big(x_I - \hat{m}(\boldsymbol{Y}_{\mathsf{B}_G(I,r)})\big)^2\big\} + o_N(1)\,.$$

This asymptotic prediction is in agreement the experiments of Figure 3. For comparison we also report the results obtained with a 'warm-start' denoiser where, at each diffusion step, we initialize BP with the outcome of the previous step. Notice that this is not a local denoiser (as information propagates across longer distances over multiple diffusion steps).

3.3 Accuracy is improved by smart ordering (continuous and discrete)

So far we studied the behavior of denoising diffusions and discrete masked diffusions in their canonical implementation, respectively with weight matrix $\boldsymbol{W} = \boldsymbol{I}_N$ in Eq. (2), and random ordering σ in Eq. (4). The idea that generating variables in a special order can improve the effectiveness of diffusion models is not new. Various heuristics have been developed: the most important are based on 'model confidence' [33, 34] (exploiting the estimated marginals) or 'coarse-to-fine' ordering [35, 36] (exploiting prior information about the variables' relations). Despite the importance of these efforts, it is fair to say that a fundamental understanding of variable ordering is still missing.

We obtain a satifactory picture for discrete diffusions in XORSAT. We begin with a negative result.

Proposition 3.2. *Consider discrete masked diffusion sampling for XORSAT. Then any ordering σ that is causal and confidence based (i.e., for each $\ell \le N$, $\sigma(\ell+1)$ only depends on on the BP marginals of $\mu_G^{\sigma,\ell}$) has the same success probability as random ordering.*

We also observe that 'coarse-to-fine' ordering does not have a clear interpretation for random CSPs.

In contrast, Figure 4 demonstrates that a principled variable ordering –which we refer to as 'reversed leaf'– dramatically improves over random ordering in generating XORSAT solutions via discrete masked diffusions.

The reversed-leaf ordering is constructed as follows. Given a XORSAT formula, with factor graph $G = (V, F, E)$, recursively select one variable that appears only in one constraint (a degree-one variable node in the factor graph), remove that variable and the corresponding constraint. After j steps, the original formula (N variable with M constraints) is reduced to a formula with $N - j$ variables and $M - j$ constraints, call it G_j (whereby $G_0 = G$): if the formula G_j can be solved, then any solution can also be extended to a solution of the formula G_{j-1} and hence (iterating) to a solution of G. The procedure can end after M, steps, hence resulting in a 'trivial' formula with $N - M$ variables and no constraints, or halt before, if each variable appears in two or more constraints.

It is known that the first event occurs with high probability for $\alpha < \alpha_{\rm d}(k)$ and the latter for $\alpha > \alpha_{\rm d}(k)$ [37, 38]. The reversed-leaf ordering is the reverse of the order in which variables are removed by the above process. This ordering is not unique: we assume that ties are broken arbitrarily.

The next statement establishes that the threshold for discrete diffusions increases from $\alpha_{\rm mask}(k)$ to $\alpha_{\rm d}(k)$ when replacing random ordering by reversed-leaf ordering.

Theorem 3.3. *Let $G = (V, F, E)$ be a k-XORSAT instance with $|V| = N$, $|F| = M$, and $N \geq M$. If the leaf-removal on G terminates with the empty graph, then discrete diffusions with reversed-leaf ordering produce samples from the target distributions.*

For random k-XORSAT, the success probability converges to one as $N \to \infty$, $N/M = \alpha < \alpha_{\rm d}(k)$.

We expect the reversed-leaf ordering to be optimal for random k-XORSAT (in the large N limit). We note that low degree vertices tend to be removed early in the leaf-removal process (in particular degree-one vertices can be removed before all the others). Motivated by this remark, in Figure 4 we consider a simple heuristic where we order variables by decreasing degree. We observe that this heuristic captures a significant portion of the improvements due to reversed-leaf.

In Figure 5 we test whether the leaf removal order is also relevant for continuous diffusions in XORSAT. One possibility would be to use the scheme (2) in which $\boldsymbol{W}$ depends on ℓ as to approximate with arbitrary precision the masked diffusion process. Such a scheme successfully samples for all $\alpha < \alpha_{\rm d}(k)$. We try something much simpler, whereby the matrix $\boldsymbol{W}$ is independent of ℓ and diagonal with $W_{ii} = 1 + c_0(S_i - {\rm ave}(S))/{\rm std}(S)$, where S_i is the rank of variable i in the leaf removal order, and c_0 is a constant that we choose to be $c_0 = 0.55$ for BP denoiser and $c_0 = 0.25$ for the learned NN denoiser. The results of Figure 5 show that this theory-motivated approach yields a substantial improvement over the standard $\boldsymbol{W} = \boldsymbol{I}_N$.

Finally, in Figs. 1 (right panel) and 3 (two right panels), we extend these experiment to SAT. While no simple equivalent of the reversed-leaf ordering applies to this case, there is a natural generalization of the reversed-degree heuristics of Figure 4. In the case of SAT, each variable i can be assigned two degrees $\deg^+(i)$, $\deg^-(i)$, counting the number of constraints in which it appears directed and negated. A highly constrained variable is a variable with large $\deg_{\rm min}(i) = \min(\deg^+(i), \deg^-(i))$: we therefore generate variables in order of decreasing $\deg_{\rm min}$. As shown in Figs. 1 and 3, the resulting procedure has a higher success rate than random ordering masked diffusions.

As the sampling proceeds, certain clauses are satisfied by the variables assigned so far and can be removed from the SAT instance. We denote by $\deg_{\rm min}(i; \ell)$ the min-degree with respect to such a reduced formula after ℓ steps. We experiment with a procedure that selects te next variable to be among the ones with largest value of $\deg_{\rm min}(i; \ell)$. We break the ties by sampling first the variables with the larger absolute value of the logit. We observe that this procedure ('dynamical reverse-degree') yields a significant increase in success rate.

3.4 Approximate uniformity test

Under discrete masked diffusions, let $\widehat{\mu}_G^{\sigma,\ell}(\,\cdot\,|\overline{\boldsymbol{x}}^{\sigma,\ell})$ be the estimated conditional marginal after sampling ℓ variable. It is then easy to estimate the probability of sampling $\overline{\boldsymbol{x}}$ as $-\log \widehat{\mu}_G(\overline{\boldsymbol{x}}) = -\sum_{\ell=0}^{N-1} \log \widehat{\mu}_G^{\sigma,\ell}(\overline{x}_{\sigma(\ell+1)}|\overline{\boldsymbol{x}}^{\sigma}_{1:\ell})$. The expectation of this quantity over $\overline{\boldsymbol{x}}$ generated according the proposed diffusion is the entropy $H(\widehat{\mu}_G)$. Since the target distribution μ_G is uniform over solutions, their KL distance is $\mathsf{KL}(\widehat{\mu}_G \| \mu_G) = \log Z_G + \mathbb{E}_{\overline{\boldsymbol{x}}\sim\widehat{\mu}_G} \log \widehat{\mu}_G(\overline{\boldsymbol{x}})$. It is elementary to compute $\mathbb{E}_G Z_G = e^{N\phi(\alpha)}$, $\phi(\alpha) := \log 2 + \alpha \log(1 - 2^{-k})$, whence we get the upper bound

$$N^{-1}\mathbb{E}_G \mathsf{KL}(\widehat{\mu}_G \| \mu_G) \leq \phi(\alpha) - N^{-1}\mathbb{E}_G \mathbb{E}_{\overline{\boldsymbol{x}}\sim\widehat{\mu}_G} \log \frac{1}{\widehat{\mu}_G(\overline{\boldsymbol{x}})} \,. \tag{6}$$

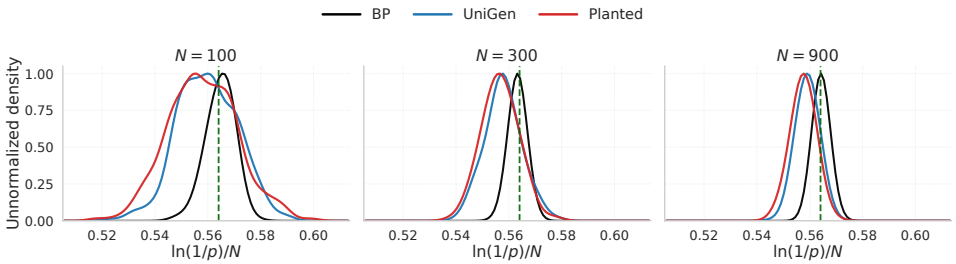


Figure 6: **4-SAT** ($\alpha = 2, r = 3$) : Distribution of probabilities of sampled solutions using BP and NN (trained on either planted on UniGen formulas) discrete diffusion samplers. The verticle line corresponds to $\phi(\alpha) = \log(2) + \alpha \log(1 - 2^{-k})$. Averaged over 3 different formulas and 100 simulations per formula.

In Figure 6, we test whether the solutions generated by our discrete diffusion algorithms (both BP and learned) are approximately uniform in the sense of KL distance. We plot the distribution of $N^{-1} \log \frac{1}{\widehat{\mu}_G(\overline{\boldsymbol{x}})}$ computed over multiple formulas and generates solutions $\overline{\boldsymbol{x}}$. We compare this histogram with the upper bound $\phi(\alpha)$. We observe that the mean of the generated distributions is slightly smaller but quite close to $\phi(\alpha)$, indicating rough uniformity. We also observe that the distribution becomes more concentrated with a mean closer to $\phi(\alpha)$ as N increases, indicating increasing uniformity.

4 Conclusion

This paper demonstrated that the uniform measure over solutions of random CSP formulas can be a useful benchmark for generative methods.

We can summarize our findings (with supporting empirical and rigorous evidence):

1. The asymptotic theory (derived for BP denoisers) is relevant at moderate N, M, r, and for learned NN denoisers (Figs. 1, 2).
2. The success rate of learned NN denoisers nearly matches the one achieved with BP denoisers (Figs. 1, 2).
3. Continuous diffusions outperform discrete diffusions (Figs. 1, 2).
4. The optimal initialization for BP denoisers is not the 'uniform' one (Fig. 3, Theorem 3.1).
5. Smart ordering of the unmasking or denoising process leads significant improvement in success rate (Figs. 1, 3, 4, 5). We achieve the optimal phase transition $\alpha_{\rm d}(k)$ for XORSAT (4, Theorem 3.3). In contrast, 'confidence-based' unmasking does not yield gains for XORSAT (Proposition 3.2).

We also emphasize that our results provides guidance as to which regimes and metrics can be challenging for generative methods: (i) Large formula close or beyond the dynamical phase transitions $\alpha_{\rm d}(k)$ (see Fig. 1); (ii) CSPs with special symmetry properties such as XORSAT, which lead to difficulty in training (Fig. 2); (iii) Measuring uniformity of the generated solutions (e.g., the KL bound of Fig. 6) can yield a very stringent test.

An important feature of using random CSPs as benchmark is that theory-motivated methods such as BP provide a very strong baseline to compare to.

Acknowledgements

We are grateful to Neeraja Abhyankar, Zachary Frangella, Ayush Jain, Marc Laugharn, Rahul Ponnala, Sahasrajit Sarmasarkar, Andreas Santucci and Pulkit Tandon, for several conversations about this work.

References

[1] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015. 1

[2] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019. 1, 3

[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 1, 3

[4] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021. 1, 17

[5] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021. 1, 2

[6] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in Neural Information Processing Systems*, 34:12454–12465, 2021. 1, 2

[7] Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37:103131–103167, 2024. 1

[8] Tennison Liu, Zhaozhi Qian, Jeroen Berrevoets, and Mihaela van der Schaar. Goggle: Generative modelling for tabular data by learning relational structure. In *The Eleventh International Conference on Learning Representations*, 2023. 1

[9] Hengrui Zhang, Jiani Zhang, Zhengyuan Shen, Balasubramaniam Srinivasan, Xiao Qin, Christos Faloutsos, Huzefa Rangwala, and George Karypis. Mixed-type tabular data synthesis with score-based diffusion in latent space. In *The Twelfth International Conference on Learning Representations*, 2024. 1

[10] Juntong Shi, Minkai Xu, Harper Hua, Hengrui Zhang, Stefano Ermon, and Jure Leskovec. TabDiff: a unified diffusion model for multi-modal tabular data generation. In *NeurIPS 2024 Third Table Representation Learning Workshop*, 2024. 1

[11] Daniel Selsam, Matthew Lamm, B Benedikt, Percy Liang, Leonardo de Moura, David L Dill, et al. Learning a sat solver from single-bit supervision. In *International Conference on Learning Representations*, 2019. 1

[12] Karlis Freivalds and Sergejs Kozlovics. Denoising diffusion for sampling sat solutions. *arXiv preprint arXiv:2212.00121*, 2022. 1

[13] Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Beyond autoregression: Discrete diffusion for complex reasoning and planning. In *The Thirteenth International Conference on Learning Representations*, 2025. 1

[14] Gautham Govind Anil, Sachin Yadav, Dheeraj Mysore Nagaraj, Karthikeyan Shanmugam, and Prateek Jain. Interleaved gibbs diffusion for constrained generation. In *ICLR 2025 Workshop on Deep Generative Model in Machine Learning: Theory, Principle and Efficacy*, 2025. 1, 17

[15] Marc Mézard, Giorgio Parisi, and Riccardo Zecchina. Analytic and algorithmic solution of random satisfiability problems. *Science*, 297(5582):812–815, 2002. 1

[16] Florent Krzakała, Andrea Montanari, Federico Ricci-Tersenghi, Guilhem Semerjian, and Lenka Zdeborová. Gibbs states and the set of solutions of random constraint satisfaction problems. *Proceedings of the National Academy of Sciences*, 104(25):10318–10323, 2007. 1, 5, 25

[17] Jian Ding, Allan Sly, and Nike Sun. Proof of the satisfiability conjecture for large k. *Annals of Mathematics*, 196(1):1–388, 2022. 1, 4, 6

[18] Peter Ayre, Amin Coja-Oghlan, Pu Gao, and Noëla Müller. The satisfiability threshold for random linear equations. *Combinatorica*, 40(2):179–235, 2020. 1, 4

[19] Marc Mézard and Andrea Montanari. *Information, Physics and Computation*. Oxford, 2009. 1, 4, 20, 25, 33

[20] Ahmed El Alaoui, Andrea Montanari, and Mark Sellke. Sampling from the Sherrington-Kirkpatrick Gibbs measure via algorithmic stochastic localization. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 323–334. IEEE, 2022. 1

[21] Andrea Montanari and Yuchen Wu. Posterior sampling in high dimension via diffusion processes. *arXiv preprint arXiv:2304.11449*, 2023. 1

[22] Brice Huang, Andrea Montanari, and Huy Tuan Pham. Sampling from spherical spin glasses in total variation via algorithmic stochastic localization. *arXiv preprint arXiv:2404.15651*, 2024. 1, 5, 24

[23] Davide Ghio, Yatin Dandi, Florent Krzakala, and Lenka Zdeborová. Sampling with flows, diffusion, and autoregressive neural networks from a spin-glass perspective. *Proceedings of the National Academy of Sciences*, 121(27):e2311810121, 2024. 1, 2, 6, 20

[24] Song Mei and Yuchen Wu. Deep networks as denoising algorithms: Sample-efficient learning of diffusion models in high-dimensional graphical models. *IEEE Transactions on Information Theory*, 2025. 1

[25] Andrea Montanari, Federico Ricci-Tersenghi, and Guilhem Semerjian. Solving constraint satisfaction problems through belief propagation-guided decimation. *arXiv:0709.1667*, 2007. 1, 2, 5, 6, 20, 24

[26] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009. 3

[27] Badih Ghazi and Euiwoong Lee. Lp/sdp hierarchy lower bounds for decoding random ldpc codes. *IEEE Transactions on Information Theory*, 64(6):4423–4437, 2017. 3

[28] Morteza Ibrahimi, Yash Kanoria, Matt Kraning, and Andrea Montanari. The set of solutions of random xorsat formulae. *The Annals of Applied Probability*, pages 2743–2808, 2015. 4, 24, 25

[29] Amin Coja-Oghlan, Allan Sly, and Nike Sun. Rigorous results: Random constraint satisfaction problems. *Spin Glass Theory and Far Beyond: Replica Symmetry Breaking After 40 Years*, pages 679–695, 2023. 4

[30] Guy Bresler and Brice Huang. The algorithmic phase transition of random k-sat for low degree polynomials. In *2021 IEEE 62nd annual symposium on foundations of computer science (FOCS)*, pages 298–309. IEEE, 2022. 5

[31] Andrea Montanari and Viet Vu. Computational bottlenecks for denoising diffusions. *arXiv:2503.08028*, 2025. 5

[32] Marc Mézard and Giorgio Parisi. The bethe lattice spin glass revisited. *The European Physical Journal B-Condensed Matter and Complex Systems*, 20(2):217–233, 2001. 6, 26

[33] Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6112–6121, 2019. 7

[34] Jungo Kasai, James Cross, Marjan Ghazvininejad, and Jiatao Gu. Non-autoregressive machine translation with disentangled context transformer. In *International conference on machine learning*, pages 5144–5155. PMLR, 2020. 7

[35] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 23(47):1–33, 2022. 7

[36] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022. 7

[37] Olivier Dubois and Jacques Mandler. The 3-XORSAT threshold. *Comptes Rendus Mathematique*, 335(11):963–966, 2002. 8, 24

[38] Amir Dembo and Andrea Montanari. Finite size scaling for the core of large random hypergraphs. *The Annals of Applied Probability*, pages 1993–2040, 2008. 8

[39] Supratik Chakraborty, Kuldeep S. Meel, and Moshe Y. Vardi. Balancing scalability and uniformity in sat-witness generator. In *Proceedings of Design Automation Conference (DAC)*, pages 60:1–60:6, 6 2014. 15

[40] Supratik Chakraborty, Daniel J. Fremont, Kuldeep S. Meel, Sanjit A. Seshia, and Moshe Y. Vardi. On parallel scalable uniform sat witness generator. In *Proceedings of Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 304–319, 4 2015. 15

[41] Mate Soos, Stephan Gocht, and Kuldeep S. Meel. Tinted, detached, and lazy cnf-xor solving and its applications to counting and sampling. In *Proceedings of International Conference on Computer-Aided Verification (CAV)*, 7 2020. 15

[42] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021. 16

[43] Marc Mézard and Andrea Montanari. Reconstruction on trees and spin glass transition. *Journal of statistical physics*, 124(6):1317–1350, 2006. 25, 26

[44] Peter Ayre, Amin Coja-Oghlan, and Catherine Greenhill. Hypergraph coloring up to condensation. *Random Structures & Algorithms*, 54(4):615–652, 2019. 25

[45] Danny Nam, Allan Sly, and Youngtak Sohn. One-step replica symmetry breaking of random regular nae-sat. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 310–318. IEEE, 2022. 25

[46] Danny Nam, Allan Sly, and Youngtak Sohn. One-step replica symmetry breaking of random regular nae-sat ii. *Communications in Mathematical Physics*, 405(3):61, 2024. 25

[47] Amir Dembo and Andrea Montanari. Gibbs measures and phase transitions on sparse random graphs. *Brazilian Journal of Probability and Statistics*, 24(2):137–211, 2010. 32

[48] T.J. Richardson and R. Urbanke. *Modern Coding Theory*. Cambridge University Press, Cambridge, 2008. 33

A Model architecture

In this section, we formally define the architecture of the neural network employed to learn a denoiser from data. Consider a factor graph $G = (V, F, E)$ characterizing a CSP instance, and the noisy information about a solution of that instance, $\boldsymbol{Y}$. In continuous denoising diffusions, $\boldsymbol{Y}$ is obtained by adding Gaussian noise to the true solution $\boldsymbol{x}$. In masked discrete diffusions it is obtained by masking some randomly selected variables in $\boldsymbol{x}$ (more details on the diffusion are provided in Appendix B.1). The neural network for a radius r denoiser makes a prediction of the true value of the i-th variable in the solution x_i by taking into account the noisy values of the r-hop neighbours of variable i in G.

Since our architecture is recursive, we will first describe the architecture for $r = 1$ and subsequently describe the construction of the radius-r network from a radius-$r-1$ network. For simplicity, we outline the architecture for k-SAT and describe the (minor) differences in the case of k-XORSAT in Appendix A.3.

A.1 Architecture for $r = 1$

For a target variable i, let $\partial i \subseteq [M]$ denote the clauses in its $1-$neighbourhood in the factor graph, i.e. $\partial i := \{a \in F, (i, a) \in E\}$. Similarly, $\partial a := \{i \in V, (i, a) \in E\} \subseteq [N]$. At a high level, the architecture comprises of a multi-layer perceptron (MLP) that serves as the denoiser, taking in a representation of the $1-$neighbourhood of the target variable i (and possibly corruption level $(L-\ell)/L$ in continuous diffusion), returning a prediction for the $i-$th variable $\in \mathbb{R}$ (continuous diffusion) or the distribution of the i-th variable (discrete diffusion). This 1-neighbourhood representation for variable i is constructed by a taking a convex combination of a learned representation of each clause $a \in \partial i$, with the weights assigned to each clause in the convex combination also learnt via a MLP.

More precisely, our $r = 1$ architecture consists of the following components:

- A ClauseMLP : $\mathbb{R} \times \{0, 1\}^2 \to \mathbb{R}^d$ (continuous diffusion) or ClauseMLP : $\{0, 1\}^5 \to \mathbb{R}^d$ (discrete diffusion); here d is the embedding dimension. The ClauseMLP constructs a representation $u_{a,j}$ for literal $j \in [k]$ in clause a. In each clause $a \in \partial i$ this MLP concatenates the noisy value of the literal $Y_{i_a(j)} \in \mathbb{R}$ (continuous) or a one-hot representation of $Y_{i_a(j)} \in \{-1, +1, *\}$ ($Y_{i_a(j)} \mapsto \texttt{OneHot}(Y_{i_a(j)}) \in \{0, 1\}^3$, discrete—recall that $*$ represents masked variable) with a one-hot representation of the sign $s_{a,j}$.
- For each clause a, the representation for the clause to the target variable i $\mathrm{Emb}_i(a) \in \mathbb{R}^{kd}$ is then constructed by concatenating the representation of all k literals $u_{a,j}, j \in [k]$. Importantly, in the concatenation the target literal j^* corresponding to the i-th variable must go first—this ensures that the representation of the clause depends on the target variable i. ClauseMLP comprises of a single linear layer followed by a `ReLU` activation.
- A WeightMLP : $\mathbb{R}^{kd} \to \mathbb{R}$ assigns a $s_i(a)$ to each $a \in \partial i$, so that the "weight" of clause $a, w(a \mid i) \propto \exp(s_i(a))$ with $\sum_{a \in \partial i} w(a \mid i) = 1$. An (intermediate) representation for the $1-$ neighbourhood of variable i is then given by the convex combination $\widetilde{h}_i^{(1)} = \sum_{a \in \partial i} w(a \mid i)\mathrm{Emb}_i(a)$. WeightMLP is a one hidden layer MLP (dimension $= d$) with `Tanh` activation.
- A ProjMLP : $\mathbb{R}^{kd} \to \mathbb{R}^d$ that maps the intermediate representation $\widetilde{h}_i^{(1)} \in \mathbb{R}^{kd}$ to the final representation $h_i^{(1)}$. This is a single linear layer followed by a `ReLU` activation.
- A DenoiserMLP : $\mathbb{R}^d \times [0, 1] \to \mathbb{R}$ (continuous) or DenoiserMLP : $\mathbb{R}^d \to \mathbb{R}^2$, which takes in $h_i^{(1)}$ and the normalized noise level $(L - \ell)/L$ (continuous) or just $h_i^{(1)}$ (discrete) to return an estimate for the $i-$th variable in the solution $\widehat{x}_{\mathrm{sol},i}$ (continuous) or the logits $(\sigma_{+1,i}, \sigma_{-1,i})$ for the $i-$th variable (discrete). We do not encode the noise level $(L - \ell)/L$ and provide it as a one-dimensional input to the network. DenoiserMLP is a one hidden layer MLP (dimension $= d$) with `ReLU` activation.

Throughout our experiments, we set $d = 128$. With this choice, the number of parameters in the radius-1 denoiser network are $\approx$ 130K. A pseudocode for the forward pass in this network is provided in Algorithm 1.

A.2 Architecture for $r \geq 2$

As mentioned previously, the radius r embedding of variable i, $h_i^{(r)}$, is built from $h_1^{(r-1)}, \ldots, h_N^{(r-1)}$ recursively; this embedding is then provided as input to the DenoiserMLP. The architecture is similar to the $r = 1$ case, with the primary difference being at the clause level where in addition to the signs/noisy value of each literal the $r - 1$ embedding of each variable is also provided as an input to the network; thereby allowing it to incorporate a message-passing like update for radius r. Two additional MLPs are included: (1) ClauseFF1 : $\mathbb{R}^{2d} \to \mathbb{R}^d$ which takes in $u_{a,j}$ (the representation of literal j in clause a) and injects the (precomputed) radius $(r - 1)$ embedding $h_{i_a(j)}^{(r-1)}$ of the variable corresponding to that literal $i_a(j)$ so as to obtain a joint representation of these in $\widetilde{u}_{a,j}$; and (2) ClauseFF2 : $\mathbb{R}^d \to \mathbb{R}^d$ which further refines this embedding. Both ClauseFF1 and ClauseFF2 comprise of one linear layer followed by a `ReLU` activation.

Note that while the number of parameters of the network for $r \geq 2$ is greater than that for $r = 1$ (due to the inclusion of the ClauseFF1 and ClauseFF2 MLPs), the number of parameters stays

Algorithm 1: Radius-1 Learnt Denoiser (Forward Pass)

Input: Factor graph $G = (V, F, E)$, noisy assignment $\boldsymbol{Y} \in \mathbb{R}^N$ (continuous) or $\boldsymbol{Y} \in \{0, 1, *\}^N$ (discrete); normalized time $(L - \ell)/L$ (for continuous)
Output: Prediction for solution $\widehat{\mathbf{x}}_{\mathrm{sol},i}$ (continuous) or logits $(\sigma_{-1,i}, \sigma_{+1,i})$ for target variable (discrete)

`Compute clause embeddings`
for each clause $a \in \partial i$ **do**
  $j^* \leftarrow \arg\min_{j \in [k]} \mathbf{1}\{i_a(j) = i\}$ `// Index of literal corresponding to variable` i
  for each literal $j \in [k]$ **do**
    $v_{a,j} \leftarrow y_{i_a(j)} \in \mathbb{R}$ `// Continuous`
    $v_{a,j} \leftarrow \texttt{OneHot}(y_{i_a(j)}) \in \mathbb{R}^3$ `// Discrete`
    $u_{a,j} \leftarrow \textsc{ClauseMLP}([\, v_{a,j}, \texttt{OneHot}(s_{a,j})\,]) \in \mathbb{R}^d$
  $\mathrm{Emb}_i(a) \leftarrow \mathrm{concat}\big(u_{a,j^*}, \{u_{a,j}\}_{j \in [k] \setminus \{j^*\}}\big) \in \mathbb{R}^{kd}$ `// Target literal goes first`
  $s_i(a) \leftarrow \textsc{WeightMLP}(\mathrm{Emb}_i(a))$
`Aggregate clause embeddings, project and denoise`
$w(a \mid i) \leftarrow \mathrm{softmax}_{a \in \partial i}\big(s_i(a)\big)$
$\widetilde{h}_i^{(1)} \leftarrow \sum_{a \in \partial i} w(a \mid i)\, \mathrm{Emb}_i(a) \in \mathbb{R}^{kd}$
$h_i^{(1)} = \textsc{ProjMLP}(\widetilde{h}_i^{(1)}) \in \mathbb{R}^d$
$\widehat{x}_{\mathrm{sol},i} \leftarrow \textsc{DenoiserMLP}([h_i^{(1)}, \tau]) \in \mathbb{R}$ `// Continuous`
return $\widehat{x}_i$
$(\sigma_{-1,i}, \sigma_{+1,i}) \leftarrow \textsc{DenoiserMLP}(h_i^{(1)}) \in \mathbb{R}^2$ `// Discrete`
return $(\sigma_{-1,i}, \sigma_{+1,i})$

constant for all $r \geq 2$. In fact, the number of parameters depends only on k and d—with $d = 128$ and $k = 4$ (i.e. our experiment setting) there are ≈ 329K parameters. The forward-pass pseudocode for this radius-r network is provided in Algorithm 2; a visual depiction of the architecture is provided in Figure 7.

A.3 Architecture for $k-$XORSAT

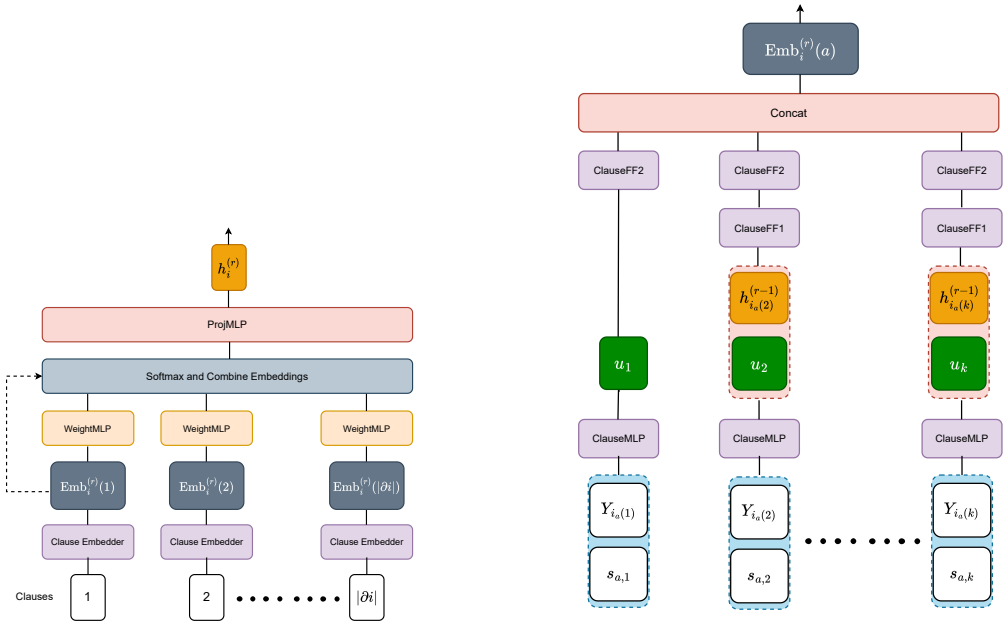
The architecture in the $k-$XORSAT setting is almost identical to the $k-$SAT architecture described in Appendices A.1 and A.2. Recall from Section 2 that a constraint in the $k-$XORSAT problem corresponds to $\sum_{j=1}^{k} z_{i_a(j)} = b_a \bmod 2$ where $x_{i_a(j)} = (-1)^{z_{i_a(j)}}$; thus, there are no signs corresponding to each literal as in the $k-$SAT setting. Correspondingly, the ClauseMLP is modified to take as input the bit $b_a \in \{0, 1\}$ instead of $\texttt{OneHot}(s_{a,j})$ as in the $k-$SAT case. Consequently, in the continuous case $\textsc{ClauseMLP} : \mathbb{R} \times \{0, 1\} \to \mathbb{R}^d$, and in the discrete case $\textsc{ClauseMLP} : \{0, 1\}^4 \to \mathbb{R}^d$. All the other components including WeightMLP, ProjMLP, DenoiserMLP and in the $r \geq 2$ case ClauseFF1 and ClauseFF2 remain the same.

B Experimental setup

In this section, we specify details pertaining to the training and evaluation of the learnt denoiser (both continuous and discrete) that were used in numerical experiments.

B.1 Denoiser training

Dataset generation. For k-SAT, we train the denoiser on 10,000 instance–solution pairs $(G, \boldsymbol{x})$, and for k-XORSAT we use 100,000 such pairs, in both cases with $N = 50$ variables and $k = 4$. For 4-SAT, the training data are uniformly distributed across 18 constraint densities $\alpha := M/N$ spanning $[0.5, 9]$. For 4-XORSAT, the training data are uniformly distributed across 16 values of α spanning $[0.05, 0.8]$. In the case of $k-$SAT, we experimented with two methods of generating the training data:



(a) (b)

Figure 7: Neural network denoiser architecture for $r \geq 2$, with target variable i. (a) illustrates the overall architecture, assuming for simplicity that $\partial i = \{1, 2, \ldots, |\partial i|\}$. (b) illustrates the detailed architecture of the ClauseEmbedder component (assuming again for simplicity that $i_a(1) = i$) which creates a radius r representation for the clause a w.r.t. variable i, by performing a message passing-like update that utilizes the radius $r-1$ representation of each variable present in the clause.

- **Randomly generated formula with solver generated solution.** In this method, a formula/factor graph G was generated uniformly at random, and then a solver is used to generate the corresponding solution $\boldsymbol{x}$. In order to form G, clauses were constructed by sampling the variables in each clause $S_a \sim \mathsf{Unif}\binom{[N]}{k}$, $S_a = \{i_a(1), \ldots, i_a(k)\}$, i.i.d. for $a \in [M]$ (i.e. sampling a $k-$sized subset of $[N]$ uniformly at random) and then the signs of each literal in the clause $s_{a,j} \in \mathsf{Unif}\{+1, -1\}$ were assigned uniformly at random for $a \in [M], j \in [k]$. Upon generating G, the solver UniGen [39, 40, 41] was used to generate a solution to G. While UniGen does not exactly sample from the uniform measure over solutions μ_G, it enjoys guarantees of approximate uniformity as shown in the cited literature. Any randomly generated formula for which a solution was not found was discarded from the training data set.
- **Random assignment and planted formula.** In this case, first a solution is sampled $\boldsymbol{x} \sim \mathsf{Unif}(\{-1, +1\}^N)$, and then a random CSP is sampled as $G \sim \mathsf{Unif}\{\widetilde{G} : \boldsymbol{x} \text{ is a solution to } \widetilde{G}\}$. For the latter, variables in each clause are sampled $S_a \sim \mathsf{Unif}\binom{[N]}{k}$ i.i.d. for $a \in [M]$ and then the signs of each literal in the clause $(s_{a,1}, \ldots, s_{a,k}) \in \mathsf{Unif}(\mathscr{S}_a)$ where $\mathscr{S}_a \subset \{+1, -1\}^k, |\mathscr{S}_a| = 2^k - 1$ consists of all possible sign assignments except the one where $s_{a,j} = -x_{i_a(j)}$ for all $j \in [k]$ (i.e., the sign assignment for which none of the k literals would evaluate to True). We note that this method of generating training data induces a distribution mismatch since at test-time we always evaluate our solver on a randomly generated formula (see Appendix B.2 for details). We observe (see Figure 8) that results from models trained with planted training data are very similar to results from models trained with UniGen generated data. Unless mentioned otherwise we train models on data generated with planted solutions.

For k-XORSAT, we construct the dataset by sampling a random instance ($S_a = \{i_a(1), \ldots, i_a(k)\} \sim \mathsf{Unif}\binom{[N]}{k}$ and $b_a \sim \mathsf{Unif}(\{-1, +1\})$ for $a \in [M]$ i.i.d.) and solving the resulting system of linear

Algorithm 2: Radius-r Learnt Denoiser (Forward Pass), $r \geq 2$

Input: Factor graph $G = (V, F, E)$; noisy assignment $\boldsymbol{Y} \in \mathbb{R}^N$ (continuous) or $\boldsymbol{Y} \in \{0, 1, *\}^N$ (discrete); normalized time $\tau = (L - \ell)/L$ (for continuous); radius $r \geq 2$.
Output: Prediction for solution $\widehat{x}_{\mathrm{sol},i} \in \mathbb{R}$ (continuous) or logits $(\sigma_{-1,i}, \sigma_{+1,i})$ for target variable (discrete).

Initialization: Construct radius-$(r-1)$ embeddings $h_l^{(r-1)} \in \mathbb{R}^d$ for $l \in [N]$
`Compute radius-`R` clause embeddings in the 1-neighbourhood of` i
for each clause $a \in \partial i$ **do**
  $j^\star \leftarrow \arg\min_{j \in [k]} \mathbf{1}\{i_a(j) = i\}$ `// index of the literal corresponding to` i
  for each literal $j \in [k]$ **do**
    $v_{a,j} \leftarrow y_{i_a(j)} \in \mathbb{R}$ `// Continuous`
    $v_{a,j} \leftarrow \mathtt{OneHot}(y_{i_a(j)}) \in \mathbb{R}^3$ `// Discrete`
    $u_{a,j} \leftarrow \textsc{ClauseMLP}\big([\, v_{a,j},\, \mathtt{OneHot}(s_{a,j})\,]\big) \in \mathbb{R}^d$
  `Inject radius-`$(r-1)$` embeddings for non-target literals`
  for each literal $j \in [k] \setminus j^\star$ **do**
    $\tilde{u}_{a,j} \leftarrow \textsc{ClauseFF1}\big([\, u_{a,j},\, h_{i_a(j)}^{(r-1)}\,]\big) \in \mathbb{R}^d$
  $\tilde{u}_{a,j^\star} \leftarrow u_{a,j^\star}$
  `Refine all literals`
  for each literal $j \in [k]$ **do**
    $u_{a,j}^{(r)} \leftarrow \textsc{ClauseFF2}\big(\tilde{u}_{a,j}\big) \in \mathbb{R}^d$
  $\mathrm{Emb}_i^{(r)}(a) \leftarrow \mathrm{concat}\Big(u_{a,j^\star}^{(r)},\, \{u_{a,j}^{(r)}\}_{j \in [k] \setminus \{j^\star\}}\Big) \in \mathbb{R}^{kd}$ `// Target literal goes first in concatenation`
`Aggregate clause embeddings, project, and denoise`
$s_i^{(r)}(a) \leftarrow \textsc{WeightMLP}(\mathrm{Emb}_i^{(r)}(a)) \in \mathbb{R}$
$w^{(r)}(a \mid i) \leftarrow \mathrm{softmax}_{a \in \partial i}\big(s_i^{(r)}(a)\big)$
$\widetilde{h}_i^{(r)} \leftarrow \sum_{a \in \partial i} w^{(r)}(a \mid i)\, \mathrm{Emb}_i^{(r)}(a) \in \mathbb{R}^{kd}$
$h_i^{(r)} \leftarrow \textsc{ProjMLP}(\widetilde{h}_i^{(r)}) \in \mathbb{R}^d$
$\widehat{x}_{\mathrm{sol},i} \leftarrow \textsc{DenoiserMLP}([\, h_i^{(r)},\, \tau\,]) \in \mathbb{R}$ `// Continuous`
return $\widehat{x}_{\mathrm{sol},i}$
$(\sigma_{-1,i}, \sigma_{+1,i}) \leftarrow \textsc{DenoiserMLP}(h_i^{(r)}) \in \mathbb{R}^2$ `// Discrete`
return $(\sigma_{-1,i}, \sigma_{+1,i})$

equations via Gaussian elimination. For an underspecified system, Gaussian elimination will yield several "free" variables that can be set to either of $+1$ or -1 to provide a solution to the system. Depending on how we set these free variables we obtain two methods to construct the dataset—one with 'biased solutions,' where the free variables are set to $+1$ and one with 'unbiased solutions,' where the free variables are sampled $\sim \mathsf{Unif}\{-1, +1\}$.

It is easy to see that unbiased solutions are uniformly random solutions by construction. However, in the discrete masked diffusions, we observe that denoisers trained from scratch on unbiased solutions perform poorly. We hypothesize that this is due to vanishing (or near-zero) gradients early in training. In order to overcome this difficulty, we adopt a two-stage procedure: we first train on biased solutions for the initial 50% of epochs and then fine-tune on unbiased solutions (Figure 9). We do not observe this degradation for continuous diffusions and for larger r in the discrete diffusions, and in those cases we train directly on unbiased solutions.

Training details. For k-SAT, we train continuous diffusion denoisers for 100 epochs and discrete-diffusion denoisers for 200 epochs. For k-XORSAT, we train both continuous and discrete denoisers for 200 epochs, except that continuous-diffusion denoisers at $r \in 6, 9$ are trained for 300 epochs. For the continuous NN denoiser, L was chosen to be 500 and a cosine schedule was used [42] to generate diffusion parameters γ_ℓ, t_ℓ. A simplified pseudocode (with batch size 1) for the training of both the continuous learnt diffusion denoiser (DDPM) and discrete learnt diffusion denoiser (MDM) is provided in Algorithm 3. In the implementation, all k-SAT models were trained with

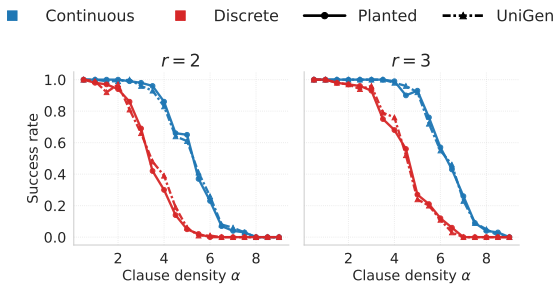


Figure 8: **4-SAT** ($N = 100$): Comparison of training SAT learned NN denoiser with planted formulas and solutions generated with UniGen. Success rate as a function of clause density α on random 4-SAT instances with $r = 2$ (*left*) and $r = 3$ (*right*). Success rates are computed over 100 random formulas.

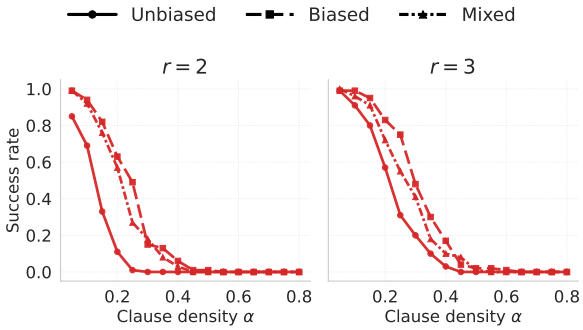


Figure 9: **4-XORSAT** ($N = 100$)**:** Comparison of learned NN denoisers trained on biased solutions, unbiased solutions, and a mixed curriculum that begins with biased solutions and then transitions to unbiased solutions. Success rate as a function of clause density α on random 4-XORSAT instances with $r = 2$ (*left*) and $r = 3$ (*right*). Success rates are computed over 100 random formulas.

batch size 32 using AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$). Because k-XORSAT is evaluated at lower clause densities, which yields fewer effective message-passing interactions per instance, we use a larger training set (100,000 CSP instance–solution pairs) and a correspondingly larger batch size of 256. For both, we use a learning rate of 3×10^{-4} with linear warm-up followed by a cosine decay schedule. Following earlier works on diffusion models [14, 4], we use exponential moving average (EMA) of parameters with decay 0.99 for evaluation. Unless stated otherwise, we do not perform hyperparameter tuning. During model training, we periodically log the performance (i.e. success rate) on a held-out validation set at the median α (0.45 for k-XORSAT and 5 for k-SAT). We save a new model checkpoint everytime either (i) the validation performance exceeds the best previously obtained validation performance, or (ii) the training loss drops more than 2% from the last epoch. We report our test performance (across all α values) with the latest saved model.

Diffusion timesteps. To select an appropriate number of diffusion timesteps (L), Figure 10 compares the performance of continuous diffusion with a BP denoiser on 4-SAT across different values of L. We observe that performance saturates by $L \approx 500$, with no additional gains for larger L. Accordingly, we use $L = 500$ timesteps for continuous diffusion in all experiments, including those with learned NN denoisers.

B.2 Diffusion and inference

Diffusion. Upon obtaining trained models $\text{ContRadiusRNet}_\theta$ and $\text{DiscRadiusRNet}_\phi$ we utilize them to generate a solution to an arbitrary instance G. The pseudocode for this diffusion process is provided in Algorithm 4 (continuous diffusion) and Algorithm 5 (discrete diffusion). Recalling our time parametrization in continuous diffusion from (2), we note that a smaller ℓ corresponds to higher noise.

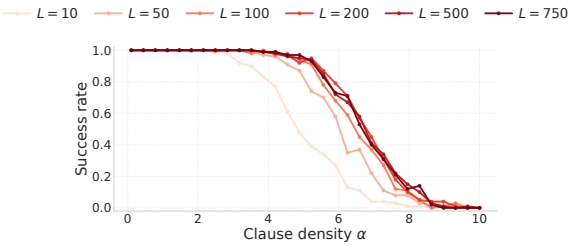


Figure 10: **4-SAT** ($N = 100$): Comparison of diffusion timesteps (L) for continuous diffusion with a local BP denoiser. Success rate as a function of clause density α on 100 random instances, evaluated with $r = 3$.

Algorithm 3: Training for Learnt Denoisers (Continuous DDPM and Discrete MDM)

Input: Training set $\mathcal{D} = \{(G_j, \boldsymbol{x}_{\mathrm{sol},j})\}_{j=1}^{n_{\mathrm{train}}}$; epochs E; radius r; L; Noising schedule $t_\ell, \ell \in [L]$ (for continuous, recall (2)).
Models: ContRadiusRNet$_\theta$ (continuous), DiscRadiusRNet$_\phi$ (discrete).
Output: Trained parameters θ and ϕ.
for epoch $= 1, \ldots, E$ **do**
  for each training sample $(G, \boldsymbol{x}_{\mathrm{sol}}) \in \mathcal{D}$ **do**
    `Continuous corruption`
    Sample $\ell \sim \mathsf{Unif}([L])$
    Sample $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I}_N)$ and form noisy solution $\boldsymbol{Y}_\ell = \sqrt{t_\ell}\, \boldsymbol{x}_{\mathrm{sol}} + \sqrt{1 - t_\ell}\, \epsilon \in \mathbb{R}^N$
    `Discrete corruption`
    Sample corruption level $\ell \sim \mathsf{Unif}([N])$
    Sample a subset $S \subseteq [N]$ with $|S| = N - \ell$ uniformly at random
    $\boldsymbol{Y}_{t,S} \leftarrow *$ for $k \in S$ (masking), and $\boldsymbol{Y}_{t,[N]\setminus S} \leftarrow \boldsymbol{x}_{\mathrm{sol},[N]\setminus S}$
    `Forward pass`
    $\widehat{\boldsymbol{x}}_{\mathrm{sol}} \leftarrow$ ContRadiusRNet$_\theta$.fwd$(G, \boldsymbol{Y}_\ell, r, (L - \ell)/L) \in \mathbb{R}^N$ `// Continuous, recover entire vector`
    Sample target variable $i \sim \mathrm{Unif}([N])$
    $(\sigma_{-1,i}, \sigma_{+1,i}) \leftarrow$ DiscRadiusRNet$_\phi$.fwd$(G, \boldsymbol{Y}_\ell, r, i) \in \mathbb{R}^2$. `// Discrete, recover logits for target variable`
    `Loss computation`
    $\mathcal{L}_{\mathrm{cont}} \leftarrow \|\widehat{\boldsymbol{x}}_{\mathrm{sol}} - \boldsymbol{x}_{\mathrm{sol}}\|_2^2 = \mathrm{MSE}(\widehat{\boldsymbol{x}}_{\mathrm{sol}}, \boldsymbol{x}_{\mathrm{sol}})$. `// Continuous`
    $\mathcal{L}_{\mathrm{disc}} \leftarrow -\log\left(\frac{\exp(\sigma_{x_{\mathrm{sol},i},i})}{\exp(\sigma_{-1,i}) + \exp(\sigma_{1,i})}\right) =$ CrossEntropy$\big((\sigma_{-1,i}, \sigma_{1,i}), x_{\mathrm{sol},i}\big)$. `// Discrete`
    `Backward pass`
    $\theta \leftarrow \theta - \eta\, \nabla_\theta \mathcal{L}_{\mathrm{cont}}$. `// Continuous`
    $\phi \leftarrow \phi - \eta\, \nabla_\phi \mathcal{L}_{\mathrm{disc}}$. `// Discrete`

Evaluation. For each experimental setting and each value of α, we evaluate on 500 randomly generated test formulae and report the fraction for which the trained learnt solver returns an accurate solution. We emphasize that regardless of how the training data is generated (random formula or planted solution), at test-time the *formula* is generated uniformly at random. Unlike during training, we do not discard unsatisfiable CSP instances, and count an unsatisfiable instance as a failure for the solver. Because the model architecture does not depend on N, we use the same model checkpoints trained at $N = 50$ when evaluating larger values of N. Similarly, we use a single checkpoint trained on the mixture over α and report performance separately on test sets at each fixed α. We trained models for $r = 2, 3, 6$ and 9.

B.3 Compute requirements

All experiments were run on a single node with 128 CPU cores and NVIDIA A100 80 GB GPUs, using Ubuntu 22.04 and CUDA 12.1. Evaluations of the BP denoiser require no training and were performed entirely on CPU. For learned denoiser evaluations, we trained eight models each for

Algorithm 4: Continuous Diffusion Sampling of CSP Solutions with NN Denoiser

Input: CSP G, Diffusion steps L; noise schedule $\{\beta_\ell, t_\ell\}_{\ell=1}^{L}$ with $t_\ell = \prod_{i=\ell}^{L-1}(1-\beta_i)$; trained denoiser $\textsc{ContRadiusRNet}_\theta$
Output: Candidate solution $\widehat{\boldsymbol{x}} \in \{-1,+1\}^N$.
Sample $\boldsymbol{Y}_0 \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_N)$.
for $\ell = 0, 1, \ldots, L-1$ **do**
  $\widehat{\boldsymbol{x}}_{\text{sol}} \leftarrow \textsc{ContRadiusRNet}_\theta(G, \boldsymbol{Y}_\ell, (L-\ell)/L) \in \mathbb{R}^N$. `// Denoise`
  $\widehat{\boldsymbol{\epsilon}} \leftarrow \dfrac{\boldsymbol{Y}_\ell - \sqrt{t_\ell}\,\widehat{\boldsymbol{x}}_{\text{sol}}}{\sqrt{1-t_\ell}} \in \mathbb{R}^N$. `// Implied noise`
  $\boldsymbol{\mu}_\ell \leftarrow \dfrac{1}{\sqrt{1-\beta_\ell}}\left(\boldsymbol{Y}_\ell - \dfrac{\beta_\ell}{\sqrt{1-t_\ell}}\,\widehat{\boldsymbol{\epsilon}}\right) \in \mathbb{R}^N$. `// DDPM posterior mean`
  $\sigma_\ell^2 \leftarrow \beta_\ell$
  Sample $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_n)$
  $\boldsymbol{Y}_{\ell+1} \leftarrow \boldsymbol{\mu}_\ell + \sigma_\ell \boldsymbol{z}$.
$\widehat{\boldsymbol{x}} \leftarrow \text{sign}(\boldsymbol{Y}_L) \in \{-1,+1\}^N$
return $\widehat{\boldsymbol{x}}$

Algorithm 5: Discrete Diffusion Sampling of CSP Solutions with NN Denoiser

Input: CSP G; Trained denoiser $\textsc{DiscRadiusRNet}_\phi$.
Output: Candidate solution $\widehat{\boldsymbol{x}} \in \{-1,+1\}^n$.
$\boldsymbol{Y} \leftarrow \{*\}^n$, $S_{\text{masked}} = [n]$
while $|S_{\text{masked}}| > 0$ **do**
  Sample $i \sim \text{Unif}(S_{\text{masked}})$
  $(\sigma_{-1,i}, \sigma_{+1,i}) = \textsc{DiscRadiusRNet}_\phi(G, \boldsymbol{Y}, i)$ `// Probability assignment for target`
  Sample $b \sim \text{Bern}\left(\frac{\exp(\sigma_{+1,i})}{\exp(\sigma_{+1,i})+\exp(\sigma_{-1,i})}\right)$
  $Y_i \leftarrow 2b-1$ `// Update` $\boldsymbol{Y}$ `to reveal one more variable`
  $S_{\text{masked}} \leftarrow S_{\text{masked}} \setminus \{i\}$
$\widehat{\boldsymbol{x}} \leftarrow \boldsymbol{Y}$
return $\widehat{\mathbf{x}}$

continuous and discrete diffusions: k-SAT with $r \in \{2, 3, 6, 9\}$ and k-XORSAT with $r \in \{2, 3, 6, 9\}$ ($N = 50$ was fixed throughout). At inference time, we reuse the corresponding checkpoint while varying α, N, and (for k-XORSAT) the variable ordering. Each of the sixteen training runs took less than two GPU-hours.

B.4 Inference budget

The inference budget is essentially proportional to the number of forward passes.

- In Fig. 15 (4-SAT) we do $N = 900$ forward passes for discrete diffusions and 500 forward passes for continuous diffusions (equal to number of diffusion steps). The computational budget for discrete diffusions is $1.8\times$ larger than the one for continuous diffusions.
- In Fig. 16 (4-SAT) we do $N = 5000$ forward passes for discrete diffusions and 500 forward passes for continuous diffusions. The computational budget for discrete diffusions is $10\times$ larger than the one for continuous diffusions.

In principle with discrete diffusions we could save some computation performing inference only over the depth-r neighborhood of node i of interest, but for the values of r and α we use, this would save us at most a small factor. (At $\alpha = 2$, $k = 4$, the average $r = 2$ neighborhood is already already of order 576, so the maximum saving would be $N/576$, i.e. $1.6\times$ in the first example above and $8.6\times$ in the second example.)

More generally, we used throughout the paper a variety of computational budgets at inference time. The comparison between discrete and continuous diffusions is extremely stable (as long as computation is above a minimum threshold).

C Details on belief propagation

In this section we describe the belief propagation (BP) message passing algorithm that we use as a *benchmark denoiser* both for *discrete* and *continuous* diffusion samplers. We refer to [19] for a general introduction to belief propagation for constraint satisfaction problems, and to [25, 23] for earlier applications to diffusions sampling.

C.1 BP denoiser for k-SAT discrete diffusion

Random k-SAT instances As described in the main text, a k-SAT clause $a \in [M]$ takes the form

$$\big(z_{i_a(1)}\big) \;\vee\; \big(z_{i_a(2)}\big) \;\vee\; \cdots \;\vee\; \big(z_{i_a(k)}\big) = +1, \qquad z_{i_a(j)} \;=\; s_{a,j}\, x_{i_a(j)}, \tag{7}$$

where $i_a(1), \dots, i_a(k) \subseteq [N]$ are distinct indices, $s_{a,j} \in \{+1, -1\}$ are clause signs, and $x \vee x' = \max(x, x')$ corresponds to logical OR under the $\{+1, -1\}$ encoding. For $i \in \partial a$, we write $s_{a,i} := s_{a,j}$ whenever $i = i_a(j)$, and $\bar{s}_{a,i} := -\,s_{a,i}$. Thus $x_i = s_{a,i}$ satisfies clause a through the literal attached to edge (a, i), while $x_i = \bar{s}_{a,i}$ does not.

Factor graph representation. We represent an instance by a bipartite factor graph with variable nodes $V = \{1, \dots, N\}$ and clause (factor) nodes $F = \{1, \dots, M\}$. We write $\partial a \subseteq V$ for the set of variables appearing in clause a, and $\partial i \subseteq F$ for the set of clauses incident to variable i. Whenever $i \in \partial a$ we denote by $\partial a \setminus i$ and $\partial i \setminus a$ the corresponding neighborhoods with the edge (a, i) removed.

For an adjacent variable node i and clause node a (i.e. $i \in \partial a$), we define the two subsets of neighboring clauses of i (excluding a):

$$\partial_+ i(a) \;\equiv\; \big\{\, b \in \partial i \setminus \{a\} \;:\; \bar{s}_{b,i} = \bar{s}_{a,i} \,\big\}, \qquad \partial_- i(a) \;\equiv\; \big\{\, b \in \partial i \setminus \{a\} \;:\; \bar{s}_{b,i} = -\bar{s}_{a,i} \,\big\}. \tag{8}$$

Intuitively, $\partial_+ i(a)$ are clauses whose preferred value for x_i agrees with clause a, while $\partial_- i(a)$ are those that disagree.

We next define the BP messages on the directed edges $i \to a$ and $a \to i$ and give their update equations; we then explain how these same equations are used as a denoising step in our discrete and continuous diffusion samplers.

Variable-to-clause update (iterative form). Let $r \in \{0, 1, 2, \dots\}$ index BP iterations, and denote by $u^{(r)}_{b \to i}$ and $h^{(r)}_{i \to a}$ the clause-to-variable and variable-to-clause messages at round r. Given the incoming clause-to-variable messages $\{u^{(r)}_{b \to i}\}_{b \in \partial i \setminus a}$, the cavity field $h^{(r+1)}_{i \to a}$ is updated by splitting neighbors according to (8):

$$h^{(r+1)}_{i \to a} \;=\; \sum_{b \in \partial_+ i(a)} u^{(r)}_{b \to i} \;-\; \sum_{b \in \partial_- i(a)} u^{(r)}_{b \to i}. \tag{9}$$

This is the "sum of incoming influences with a sign" rule: clauses in $\partial_+ i(a)$ contribute positively, while clauses in $\partial_- i(a)$ contribute negatively.

Clause-to-variable update (iterative form). For a clause a and a variable $i \in \partial a$, the message $u^{(r+1)}_{a \to i}$ is updated as a function of the incoming cavity fields $\{h^{(r+1)}_{j \to a}\}_{j \in \partial a \setminus i}$ from the other variables in the clause:

$$u^{(r+1)}_{a \to i} \;=\; -\frac{1}{2} \log \left(1 - (1 - \epsilon)\, 2^{-(k-1)} \prod_{j \in \partial a \setminus i} \big(1 - \tanh\big(\bar{s}_{a,j}\, h^{(r+1)}_{j \to a}\big)\big) \right), \tag{10}$$

where $\bar{s}_{a,j} \in \{\pm 1\}$ is the unsatisfying value of x_j for clause a, and $\epsilon \in [0, 1)$ is an optional noise/relaxation parameter used to avoid hard warnings.[3]

[3] Setting $\epsilon = 0$ gives the standard BP update; $\epsilon > 0$ weakens clause influences.

Equations (9)–(10) define one synchronous BP iteration: first compute all $\{h_{i\to a}^{(r+1)}\}$ from $\{u_{b\to i}^{(r)}\}$, then compute all $\{u_{a\to i}^{(r+1)}\}$ from $\{h_{j\to a}^{(r+1)}\}$, and repeat for r rounds.

From messages to marginals. At any BP round r, once the clause-to-variable messages $\{u_{a\to i}^{(r)}\}$ have been updated, we form an (approximate) variable "field" by summing incoming influences with their signs:

$$H_i^{(r)} \;=\; \sum_{a\in\partial i} s_{a,i}\, u_{a\to i}^{(r)}, \tag{11}$$

and the corresponding magnetization and Bernoulli marginal are

$$m_i^{(r)} = \tanh\!\big(H_i^{(r)}\big), \qquad \widehat{\mathbb{P}}^{(r)}(x_i=+1) = \frac{1+m_i^{(r)}}{2}. \tag{12}$$

Initialization and stopping. We initialize the clause-to-variable messages $\{u_{a\to i}^{(0)}\}$ according to the selected policy: (i) *zero init*, $u_{a\to i}^{(0)}=0$ for all directed edges; (ii) *warm init*, where we reuse the previous-step messages on the surviving edges from the previous diffusion step t; or (iii) *cavity init*, where $u_{a\to i}^{(0)}$ are sampled i.i.d. from the population-dynamics approximation of the fixed-point distribution $\widehat{Q}(u)$ (see Appendix E). Starting from $\{u_{a\to i}^{(0)}\}$, we run r synchronous BP rounds using (9)–(10). Optionally, we terminate early if the maximum change in messages across one round falls below a tolerance,

$$\max_{(a,i)\in E}\left|u_{a\to i}^{(r+1)} - u_{a\to i}^{(r)}\right| \;<\; \tau, \tag{13}$$

but in our sweeps we primarily compare methods at a fixed computational budget by fixing r and setting $\tau=0$.

Implementation details. Our implementation is mathematically equivalent to Eqs. (9), (10), but is organized to avoid storing both message families explicitly. We maintain only the clause-to-variable messages $\{u_{a\to i}\}$ and reconstruct the variable-to-clause cavity fields $\{h_{i\to a}\}$ on the fly via the cavity identity. Specifically, defining the full field at variable i as

$$H_i \;=\; \sum_{b\in\partial i} s_{b,i}\, u_{b\to i}, \tag{14}$$

the cavity field for the directed edge $(i\to a)$ is obtained exactly by removing the contribution of the destination clause,

$$h_{i\to a} \;=\; \sum_{b\in\partial i\setminus\{a\}} s_{b,i}u_{b\to i} \;=\; H_i - s_{a,i}u_{a\to i}. \tag{15}$$

This allows us to evaluate the clause update (10) without a separate state for h. For each clause a and neighbor $i\in\partial a$, the update depends on a leave-one-out product over the remaining variables $j\in\partial a\setminus\{i\}$; in code we compute the oriented terms $t_{j\to a}\equiv 1-\tanh\!\big(s_{a,j}\,h_{j\to a}\big)$ for all $j\in\partial a$ and form all leave-one-out products $\prod_{j\in\partial a\setminus\{i\}} t_{j\to a}$ efficiently using prefix/suffix products in $\mathcal{O}(|\partial a|)$ time per clause (rather than $\mathcal{O}(|\partial a|^2)$ by recomputing a product for each i). Finally, for numerical stability we clip all LLR-like quantities (fields and messages) to a fixed range and lower-bound the argument of the logarithm in (10) away from zero; these safeguards do not alter the intended fixed-point equations but prevent overflow/underflow in finite precision.

Matching asymptotic predictions. It has been repeatedly observed in the literature that BP nearly matches asymptotic predictions already at moderate N ($N\gtrsim 100$). Partial justifications for this fact exist in the literature, which predicts the finite-N gap to be of order $1/N$.

To confirm this general observation, we carried numerical experiments with 4-SAT comparing BP computations of the marginals (both at finite r and $r=\infty$) with a 'gold standard' given by rejection sampling (for $N\le 150$) or MCMC (which we tested for mixing).

In these simulations, we measured the mean absolute error $\mathrm{MAE} := N^{-1}\sum_{i=1}^{N}|\widehat{\mu}_G(x_i=+1) - \mu_G(x_i=+1)|$. We measured the following errors in marginals computation:

Table 1: MAE in marginals computation.

N	$r=1$	$r=2$	$r=\infty$
30	1.01×10^{-3}	4.82×10^{-3}	5.16×10^{-3}
50	6.61×10^{-4}	3.85×10^{-3}	4.00×10^{-3}
75	5.47×10^{-4}	3.15×10^{-3}	3.17×10^{-3}
100	4.11×10^{-4}	2.51×10^{-3}	2.79×10^{-3}
150	3.36×10^{-4}	2.28×10^{-3}	2.23×10^{-3}
200	3.46×10^{-4}	1.55×10^{-3}	1.89×10^{-3}

A least square fit to the $r = 1$ data show that they are consistent with

$$\mathrm{MAE}(N; r = 1) \approx \frac{0.03}{N}$$

for $r = 1$, hence we conclude that the gap with respect to large N asymptotics is negligible with respect to the effects we see in our experiments.

C.2 Continuous variant (used inside continuous diffusion)

The continuous BP denoiser acts on the same k-SAT factor graph and uses the same clause-to-variable BP updates as in the discrete case. At diffusion step ℓ, the current observation is $\boldsymbol{Y}_\ell \in \mathbb{R}^N$, and the goal is to approximate the denoiser $\boldsymbol{m}(\boldsymbol{Y}_\ell; t_\ell)$ associated with the tilted measure $\mu_G^{\boldsymbol{Y}_\ell, t_\ell}$ from Eq. (3).

For diffusion level $t_\ell \in (0, 1)$, we form the variable field

$$H_i(\ell) \;=\; \frac{\sqrt{t_\ell}}{1 - t_\ell}\, Y_{\ell,i} \;+\; \sum_{b \in \partial i} s_{b,i}\, u_{b \to i}(\ell). \tag{16}$$

and compute cavity fields on edges via the same identity as in the discrete case,

$$h_{i \to a}(\ell) \;=\; H_i(\ell) - s_{a,i}\, u_{a \to i}(\ell). \tag{17}$$

Given $\{h_{j \to a}(\ell)\}_{j \in \partial a \setminus i}$, clause-to-variable messages are then updated using Eq. (10) (with the same leave-one-out products, clipping, and other numerical safeguards as described in Section C.1). Iterating this update for r rounds yields final fields $H_i(\ell)$ and magnetizations $m_i(\ell) = \tanh(H_i(\ell))$, which serve as the BP-based continuous denoiser output at step ℓ.

C.3 Belief propagation in k-XORSAT

Similarly to the previous section, in this section we describe the belief propagation (BP) / message passing algorithm that we use as a *denoiser* inside both our *discrete* and *continuous* diffusion samplers to generate assignments satisfying random k-XORSAT formulas.

Problem definition and notation. A k-XORSAT instance consists of M parity constraints over N binary variables. We use the spin encoding $\boldsymbol{x} = (x_1, \ldots, x_N) \in \{+1, -1\}^N$, and write the instance as a factor graph with variable nodes $i \in [N]$ and clause (factor) nodes $a \in [M]$. Each clause a involves exactly k variables with indices $\partial a = \{i_a(1), \ldots, i_a(k)\} \subseteq [N]$, sampled uniformly at random without replacement, and has a random sign (parity) $s_a \in \{+1, -1\}$. The constraint enforced by clause a is

$$\prod_{i \in \partial a} x_i \;=\; s_a\,. \tag{18}$$

Equivalently, in the Boolean encoding $\{0, 1\}$ with $x_i = (-1)^{\tilde{x}_i}$, each constraint is a linear equation modulo 2, $\tilde{x}_{i_a(1)} \oplus \cdots \oplus \tilde{x}_{i_a(k)} = b_a$, where $s_a = +1$ if $b_a = 0$ and $s_a = -1$ if $b_a = 1$.

For a variable i, we denote by $\partial i = \{a \in [M] : i \in \partial a\}$ the set of clauses incident to i. Throughout, messages are passed along directed edges of the factor graph: we write $u_{a \to i}$ for clause-to-variable messages and $h_{i \to a}$ for variable-to-clause messages. The corresponding *local field* (or belief field) at variable i is denoted by H_i, and its associated magnetization by $m_i = \tanh(H_i)$ (in the continuous formulation).

Discrete BP message passing. In the discrete (hard) variant, messages take values in the three-point set $\{+\infty, 0, -\infty\}$. We denote by $h_{i\to a}^{(r)}$ the variable-to-clause message and by $u_{a\to i}^{(r)}$ the clause-to-variable message at BP iteration r. Given a partial reveal vector $\boldsymbol{y} = (y_1, \dots, y_N)$ with $y_i \in \{+\infty, 0, -\infty\}$ (where $y_i = \pm\infty$ if variable i is revealed and $y_i = 0$ otherwise), the BP updates are

$$u_{a\to i}^{(r+1)} = s_a \prod_{j\in\partial i\setminus a} h_{j\to a}^{(r)}\,,$$
$$h_{i\to a}^{(r)} = \operatorname{sign}\Big(y_i + \sum_{b\in\partial i\setminus a} u_{b\to i}^{(r)}\Big)\,.$$

The "sign of an ∞-valued sum" is understood with the convention that the sum never contains both $+\infty$ and $-\infty$ simultaneously (hence it is well-defined), and returns 0 when all contributing terms are 0. After r_{BP} iterations, the final local field is computed as

$$H_i \;=\; \operatorname{sign}\Big(y_i + \sum_{b\in\partial i} u_{b\to i}^{(r)_{\mathrm{BP}}}\Big)\,, \tag{19}$$

Continuous BP message passing. In the continuous (soft) variant, messages are real-valued and we again denote by $h_{i\to a}^{(r)} \in \mathbb{R}$ the variable-to-clause messages and by $u_{a\to i}^{(r)} \in \mathbb{R}$ the clause-to-variable messages at BP iteration r. Given the current diffusion state $\boldsymbol{y} \in \mathbb{R}^N$ at time ℓ (the "noisy observation"), the BP updates are

$$u_{a\to i}^{(r+1)} = \operatorname{atanh}\Big(s_a \prod_{j\in\partial i\setminus a} \tanh(h_{j\to a}^{(r)})\Big) \tag{20}$$
$$h_{i\to a}^{(r)} = \lambda_\ell\, y_i + \sum_{b\in\partial i\setminus a} u_{b\to i}^{(r)} \tag{21}$$

After r iterations, we form the (approximate) local field

$$H_i \;=\; \lambda_\ell\, y_i + \sum_{b\in\partial i} u_{b\to i}^{r_{\mathrm{BP}}}\,, \tag{22}$$

and the corresponding magnetization estimate

$$m_i \;=\; \tanh(H_i) \in [-1, 1]\,. \tag{23}$$

Denoiser in the continuous diffusion. In our continuous diffusion sampler, for each reverse step we run the above BP recursion on the factor graph with input $\boldsymbol{y}$ and time-dependent scale λ_ℓ (determined by the diffusion schedule), and we use the resulting magnetizations $\boldsymbol{m} = (m_i)_{i\le N}$ as a denoising estimate of the underlying spins (i.e., as the BP-based "denoiser output" plugged into the reverse update). Equivalently, H_i plays the role of a BP-informed score/field from which the denoiser $\tanh(H_i)$ is obtained.

Implementation details. We implement both denoisers as BP on the k-XORSAT factor graph.

Discrete BP (messages in $\{+\infty, 0, -\infty\}$). We store directed messages in Python dicts keyed by edges: `h[(i,a)]` for $h_{i\to a}$ and `u[(a,i)]` for $u_{a\to i}$, using adjacency lists `var_to_clauses` and `clause_to_vars`. Clause-to-variable updates compute $u_{a\to i}$ as the signed product of incoming $\operatorname{sign}(h_{j\to a})$ over $j \in \partial a \setminus i$, outputting 0 if any incoming message is 0. Variable-to-clause updates use an "∞-safe" $\operatorname{sign}(\sum)$ routine: it returns $+\infty$ (resp. $-\infty$) if only $+\infty$ (resp. only $-\infty$) appears among incoming messages, and 0 otherwise.

Continuous BP (real-valued messages). For speed we flatten edges into arrays and store the factor graph in CSR-like form. Clause updates compute $u_{a\to i} = \operatorname{atanh}\big(s_a \prod_{j\in\partial a\setminus i} \tanh(h_{j\to a})\big)$ using prefix/suffix products to avoid division-by-zero, clipping both fields and the atanh argument for numerical stability; optional early-stop tolerance control convergence.

Use in diffusion. In both diffusion samplers, BP provides the denoiser at each reverse step: discrete BP is run with partially revealed variables ($y_i \in \{\pm\infty, 0\}$), while continuous BP takes the current noisy $\mathbf{y} \in \mathbb{R}^N$ and uses $\lambda_\ell = \sqrt{t_\ell}/(1 - t_\ell)$ to produce magnetizations $m_i = \tanh(H_i)$ that enter the reverse diffusion update.

C.4 Belief propagation in NAE-SAT

A NAE-k-SAT clause uses the same literal notation as in Eq. (7), namely $z_{i_a(j)} = s_{a,j} x_{i_a(j)}$. The difference from k-SAT is only in the clause constraint: clause a is satisfied when its literals are not all equal. Equivalently, the two unsatisfying assignments are the all $+1$ and all -1 literal configurations.

The factor graph, variable-to-clause messages, initialization, marginal computation, and other implementation details are the same as in Appendix C.1. The only change is the clause-to-variable update. For a clause a and a variable $i \in \partial a$, the message $u_{a\to i}^{(r+1)}$ is updated as:

$$u_{a\to i}^{(r+1)} = \frac{1}{2}\log\left(1 - \frac{(1-\epsilon)}{2^{k-1}} \prod_{j\in\partial a\setminus i} \left(1 + \tanh\left(s_{a,j} h_{j\to a}^{(r+1)}\right)\right)\right) - \frac{1}{2}\log\left(1 - \frac{(1-\epsilon)}{2^{k-1}} \prod_{j\in\partial a\setminus i} \left(1 - \tanh\left(s_{a,j} h_{j\to a}^{(r+1)}\right)\right)\right). \tag{24}$$

C.5 Belief propagation in hypergraph bicoloring

A k-hypergraph bicoloring instance is the unsigned special case of NAE-k-SAT. Each variable $x_i \in \{\pm 1\}$ denotes the color of vertex i, and each hyperedge a is satisfied when the variables in ∂a are not all equal. Equivalently, the two unsatisfying assignments are the all $+1$ and all -1 color configurations on the hyperedge. This is exactly NAE-k-SAT with $s_{a,i} = +1$ for every edge (a, i). The factor graph, variable-to-clause messages, initialization, marginal computation, and implementation details are the same as in Appendix C.4.

D Computing phase transition thresholds α_{diff} and α_{mask}

In this section we describe the theoretical predictions for α_{diff} and α_{mask}, following [25]. In short, these phase transitions are identified with a 'reconstruction' (or 'extremality') phase transition for a Gibbs measure on an infinite tree that is identified as the local weak limit of the uniform measure μ_G. We emphasize that the definitions given here are rigorous, but establishing that they actually correspond to a phase transition in the generative algorithm remain an open problem. Rigorous results can be established for XORSAT under discrete diffusions (the result in that case is an easy modification of standard results [37, 28]), and the spin glass models of [22].

D.1 Tree models and local weak convergence

Under any of the random CSPs ensembles studied in this paper (k-SAT, k-XORSAT, k-NAESAT, hypergraph bicoloring) the random factor graph $G = (V, F, E)$ converges (in the sense of local weak convergence, as $N, M \to \infty$ with $N/M \to \alpha$) to a rooted infinite tree formula $T = (V_T, F_T, E_T)$ with root o, which is described next. T is a bipartite infinite tree with two type of vertices 'variable nodes' V_T and 'factor nodes' F_T, connected by edges E_T. The tree is rooted at a variable node o. Each variable node is connected at the next level of the tree to Poisson$(k\alpha)$ factor nodes (hence has total degree Poisson$(k\alpha) + 1$, with the exception of the root, which has degree Poisson$(k\alpha)$). Each factor node is connected to $k-1$ variable nodes at the next level (hence has total degree k) We denote by $T(L)$ the finite subtree obtained by truncating the above after L generations of variable nodes (in particular, $T(0)$ is the trivial tree that only contains the root).

To each variable node $i \in V_T$, we associate a variable $x_i \in \{+1, -1\}$, and write $\boldsymbol{x} = (x_i : i \in V_T)$. To each factor node $a \in F_T$ we associate a constraint $\psi_a(\boldsymbol{x}_{\partial a})$ which we draw uniformly at random among all those of the CSP under study. With a slight abuse of notation, we denote by G the factor graph, together with the constraints $(\psi_a : a \in F_T)$. We can therefore speak of a configuration $\boldsymbol{x}$ satisfying (or not) the (infinite) formula G.

The free boundary Gibbs measure μ_T^{free} is a way of defining a probability measure over solutions of the infinite formula G, which plays an important role for us. Letting $\mu_{T(L)}$ denote the uniform measure over solutions of the sub-formula $T(L)$ (extended arbitrarily beyond level L), we obtain

μ_T^{free} by taking the weak limit of $\mu_{T(L)}$ as $L \to \infty$. Existence of the limit (along a subsequence) is guaranteed by standard compactness arguments.

We then imagine sampling from $\boldsymbol{x}_* \sim \mu_T^{\text{free}}$, and revealing information $\boldsymbol{y} = (y_i : i \in V_T)$ where in continuous diffusions we have $\boldsymbol{y} = \sqrt{t}\boldsymbol{x}_* + \sqrt{1-t}\,\boldsymbol{g}$, for $\boldsymbol{g} = (g_i)_{i \in V_T} \sim \mathsf{N}(0, \boldsymbol{I}_{V_T})$ and in discrete masked diffusions we have $y_i = x_{*,i}$ independently with probability t for each $i \in V_T$, and $y_i = *$ otherwise.

We then define $\mu_T^{\boldsymbol{y},t}(\boldsymbol{x})$ to be the conditional distribution of $\boldsymbol{x}$ given $\boldsymbol{y}$ under the above joint distribution. Equivalently, this can be defined by by taking the limit $L \to \infty$ of the measures

$$\mu_{T(L)}^{\boldsymbol{y},t}(\boldsymbol{x}) := \frac{1}{Z_{T(L)}} \prod_{i \in V_{T(L)}} e^{\sqrt{t} y_i x_i/(1-t)} \prod_{a \in F_{T(L)}} \psi_a(\boldsymbol{x}_{\partial a})\,, \tag{25}$$

$$\mu_{T(L)}^{\boldsymbol{y},t}(\boldsymbol{x}) := \frac{1}{Z_{T(L)}} \prod_{i \in V_{T(L)}} \mathbf{1}_{y_i = x_i} \prod_{a \in F_{T(L)}} \psi_a(\boldsymbol{x}_{\partial a})\,, \tag{26}$$

respectively for continuous and discrete diffusions.

D.2 Reconstruction/extremality phase transition and α_{diff}, α_{mask}

Let $\boldsymbol{x}_{\ge \ell} := (x_i : d_T(o,i) \ge \ell)$ denote the variables at level ℓ or below in T. In words, the reconstruction/extremality question for the Gibbs measure μ_T^{free} asks whether observing variables $\boldsymbol{x}_{\ge \ell}$ increases the chance of guessing correctly x_o by an amount that is bounded away from zero as $\ell \to \infty$. If this happens, we say that reconstruction is possible (and the measure is non-extremal): this is a measure of how correlated is the tree Gibbs measure. Since [43, 16], it was realized that this phase transition coincides with the statistical physics prediction for the dynamical phase transition $\alpha_{\text{d}}(k)$. Again, while this identification is fully rigorous, the fact that the physics prediction for $\alpha_{\text{d}}(k)$ actually coincided with shattering in the space of solutions and hardness of sampling has only been shown in certain cases, see e.g. [28, 44, 45, 46].

The reconstruction phase transition can be analogously defined for the conditional measure $\mu_T^{\boldsymbol{y},t}$. Formally, we define the reconstruction parameter by

$$R(t,\alpha) := \lim_{\ell \to \infty} \mathbb{E} \Big\| \mu_T^{\boldsymbol{y},t}\big(x_o \in \cdot \,\big|\boldsymbol{x}_{\ge \ell}\big) - \mu_T^{\boldsymbol{y},t}\big(x_o \in \cdot\,\big) \Big\|_{\text{TV}}\,. \tag{27}$$

As mentioned above, $R(t,\alpha)$ measure how much better we can reconstruct the root value x_o using the information $\boldsymbol{x}_{\ge \ell}, \boldsymbol{y}$ than just using $\boldsymbol{y}$.

We then define (this definition applies both to discrete and continuous diffusions, depending on which of the two measures $\mu_T^{\boldsymbol{y},t}$ is used):

$$\alpha_{\text{mask/diff}}(k) := \sup\big\{\alpha :\; R(t,\alpha) = 0 \forall t \in [0,1]\big\}\,. \tag{28}$$

The limit quantity $R(t,\alpha)$ (and hence the thresholds) can be characterized by a distributional recursion which we can either study analytically or approximate numerically.

In the next sections we provide further details about these computations.

D.3 Computation of the threshold α_{mask} for XORSAT

For XORSAT and discrete diffusion, the calculation of $R(t,\alpha)$ reduces to studying a simple one dimensional recursion for the quantity q_ℓ, which is the probability that the root value x_o is completely determined given the information $\boldsymbol{x}_{\ge \ell}, \boldsymbol{y}$. The recursion for q_ℓ is given by (see e.g. [19])

$$q_\ell \;=\; t + (1-t)\Big(1 - e^{-k\alpha q_\ell^{k-1}}\Big) \;=\; 1 - (1-t)e^{-k\alpha q_\ell^{k-1}}\,. \tag{29}$$

Denote by q_ℓ^z the sequence thus defined with initialization $q_0^z = z$. The resulting threshold is

$$\alpha_{\text{mask}} \;=\; \sup\big\{\alpha \in \mathbb{R}_{\ge 0} :\; \lim_{\ell \to \infty}(q_\ell^1 - q_\ell^0) = 0\,\forall t \in [0,1]\big\}\,. \tag{30}$$

Equivalently, it is the largest α such that the iteration has a unique fixed point $q \in [0,1]$:

$$q \;=\; 1 - (1-t)e^{-k\alpha q^{k-1}}\,. \tag{31}$$

At the critical point, the fixed point becomes *degenerate*, i.e. $f(x) = x$ has a triple contact: $f(x) = x$, $f'(x) = 1$, and $f''(x) = 0$.

Differentiating (31) with respect to q once and twice yields the two additional conditions

$$1 \;=\; (1-t)\, k\alpha\, (k-1)\, q^{k-2}\, e^{-k\alpha x^{k-1}}\,, \tag{32}$$

and

$$0 \;=\; (1-t)\, k\alpha\, e^{-k\alpha q^{k-1}} \Big[(k-1)(k-2)\, q^{k-3} \;-\; (k-1)^2\, k\alpha\, q^{2k-4} \Big]\,. \tag{33}$$

Solving the system (31), (32), (33) we get

$$x_* \;=\; \frac{k-2}{k-1}, \qquad 1 - t_* \;=\; \frac{e^{\frac{k-2}{k-1}}}{k-1}\,, \tag{34}$$

$$\alpha_{\rm mask}(k) \;=\; \frac{1}{k\, x_*^{k-1}} \cdot \frac{k-2}{k-1} \;=\; \frac{1}{k} \left(\frac{k-1}{k-2} \right)^{k-2}, \qquad (k \geq 3)\,. \tag{35}$$

D.4 Distributional recursion

We next describe the distributional recursion characterization of the reconstruction parameter $R(t, \alpha)$ (and consequently $\alpha_{\rm diff}/\alpha_{\rm mask}$, per definition (28)). For the sake of concreteness, we will focus on the case of k-XORSAT and continuous diffusions, but the suitable generalizations can be provided.

In general, the approach is to write a distributional recursion for the log likelihood appearing (implicitly) in Eq. (27), namely

$$h_\ell^+ := \frac{1}{2} \log \frac{\mu_T^{\boldsymbol{y},t}\big(x_o = +1 \big| \boldsymbol{x}_{\geq \ell}\big)}{\mu_T^{\boldsymbol{y},t}\big(x_o = -1 \big| \boldsymbol{x}_{\geq \ell}\big)}\,, \qquad h_\ell^0 := \frac{1}{2} \log \frac{\mu_T^{t,\boldsymbol{y}_{\leq \ell}}\big(x_o = +1\big)}{\mu_T^{t,\boldsymbol{y}_{\leq \ell}}\big(x_o = -1\big)}\,. \tag{36}$$

An important technical simplification for XORSAT is that —by a symmetry argument— we can assume that $s_a = +1$ (recall that s_1 is the sign defining the a-th constraints) and that $\boldsymbol{x}_{\geq \ell} = +\mathbf{1}$.

We denote by Q_ℓ^+, Q_ℓ^0 the laws of h_ℓ^+, h_ℓ^0. These satisfy the following distributional recursion (here we slightly abuse notation because –a priori– Q_ℓ, $\widehat{Q}_\ell$ are only probability measures, not functions)

$$\widehat{Q}_{\ell+1}(u) = \int \delta_{u - f(h_1^{k-1})} \prod_{i=1}^{k-1} {\rm d}Q_\ell(h_i)\,, \tag{37}$$

$$Q_\ell(h) = \sum_{m \geq 0} \mathsf{P}_m \int \delta_{h - h_0 - \sum_{i=1}^m u_i} \prod_{i=1}^m {\rm d}\widehat{Q}_\ell(u_i)\, {\rm d}G_t(h_0)\,, \tag{38}$$

where $\mathsf{P}_m = \mathsf{Poisson}(k\alpha)$ and $G_t = \mathsf{N}(t, t)$. Further $f(h_1^{k-1}) = \operatorname{atanh}(\tanh(x_1) \cdots \tanh(x_{k-1}))$.

The distributions Q_ℓ^+, Q_ℓ^0 are defined by the recursion (37), (38), with two different initializations: $Q_0^0 = \delta_0$, and $Q_0^+ = \delta_{+\infty}$ (note that this makes sense because $h \mapsto \tanh(h)$ can be extended by continuity to $[-\infty, +\infty]$).

Given the sequence of distributions Q_ℓ^+ and Q_ℓ^0, the threshold $\alpha_{\rm diff}$ is characterized by

$$\alpha_{\rm diff} \;=\; \sup \Big\{ \alpha \in \mathbb{R}_{\geq 0} :\; \lim_{\ell \to \infty} \Big[\int \tanh(h) {\rm d}Q_\ell^+(h) - \int \tanh(h) {\rm d}Q_\ell^0(h) \Big] = 0 \;\; \forall t \in [0,1] \Big\}\,. \tag{39}$$

D.5 Numerical computation of the thresholds $\alpha_{\rm diff}/\alpha_{\rm mask}$

We next describe a numerical algorithm that subsamples the about iteration to estimate $\alpha_{\rm diff}$ [32, 43].

Algorithm We keep two samples of size N (say $N = 10^4$), representing respectively $\widehat{Q}$ and Q:

$$\widehat{\mathcal{Q}}_N^0 := \{u_1, \dots, u_N\}, \qquad \mathcal{Q}_N^0 := \{h_1, \dots, h_N\}\,, \tag{40}$$

$$\widehat{\mathcal{Q}}_N^+ := \{u_1, \dots, u_N\}, \qquad \mathcal{Q}_N^+ := \{h_1, \dots, h_N\}\,. \tag{41}$$

Table 2: Threshold values for k-XORSAT.

k	α_{mask}	α_{diff}	α_{d}
3	0.666667	0.736	0.818469
4	0.562500	0.632	0.772280
5	0.474074	0.531	0.701780
6	0.406901	0.511	0.637081
7	0.355474	0.396	0.581775
8	0.315203	0.384	0.534997
9	0.282944	0.366	0.495255
10	0.256578	0.350	0.461197

The only difference is that $\widehat{\mathcal{Q}}_N^0, \mathcal{Q}_N^0$ are initialized to be all 0, while $\widehat{\mathcal{Q}}_N^+, \mathcal{Q}_N^+$ are initialized to be all $+A$, with A a large positive number (e.g. 10). We will use $\widehat{\mathcal{Q}}_N, \mathcal{Q}_N$ to refer to either of the two initialization. The 0 and $+$ samples are never mixed.

Then the following cycle is repeated for T iterations:

1. Update $\widehat{\mathcal{Q}}_N$ as follows.
   For each $i \in \{1, \dots, N\}$:
   (a) Sample indices $j(1), \dots, j(k-1) \sim_{iid} \mathsf{Unif}([N])$ (with replacement)
   (b) For each $a \in \{1, \dots k-1\}$, set
   - $\hat{h}_a = h_{j(a)}$ (from the sample $\mathcal{Q}_N$).

   (c) Replace $u_i \leftarrow f(\hat{h}_1^{k-1})$
2. Update $\mathcal{Q}_N$ as follows.
   For each $i \in \{1, \dots, N\}$:
   (a) Sample $m \sim_{iid} \mathsf{Poisson}(k\alpha)$
   (b) Sample indices $j(1), \dots, j(m) \sim_{iid} \mathsf{Unif}([N])$ (with replacement)
   (c) Sample $h_{0,i} \sim \mathsf{N}(t, t)$ (mean t, variance t).
   (d) Replace

$$h_i \leftarrow h_{0,i} + \sum_{a=1}^{m} u_{j(a)}$$

After these T iterations, we are left with populations $\mathcal{Q}_N^+, \mathcal{Q}_N^0$. The size can be increased by taking the union over iterations $T, T+1, \dots T+\Delta-1$ (thus obtaining populations of size $N \cdot \Delta$). We compute the mean difference

$$\Delta(t, \alpha) := \frac{1}{N} \sum_{i=1}^{N} \tanh h_i(\mathcal{Q}^+) - \frac{1}{N} \sum_{i=1}^{N} \tanh h_i(\mathcal{Q}^0) \tag{42}$$

For a given value of α, evaluate $\Delta(t, \alpha)$ for $t \in [0, 1]$ on a grid $S \subseteq [0, 1]$. Fixing a small tolerance Δ_0, we estimate:

$$\hat{\alpha}_{\mathrm{diff}} := \sup \left\{ \alpha \in \mathbb{R}_{\geq 0} : \ \Delta(t, \alpha) \leq \Delta_0 \ \ \forall t \in S \right\}. \tag{43}$$

For k-XORSAT, the values of α_{mask}, α_{diff}, and α_{d} are reported in Table 2.

E Cavity initialization

We next describe the BP_{c} initialization. To be definite, we focus on k-SAT but the definition is more general. Recall from Appendix C that BP operates on messages that are associated to directed edges in G, each message being a probability distribution over $\{+1, -1\}$ or —equivalently— log-likelihood ratios. In $\mathrm{BP}_{\mathsf{c}}(r)$ we initialize messages from variable notes

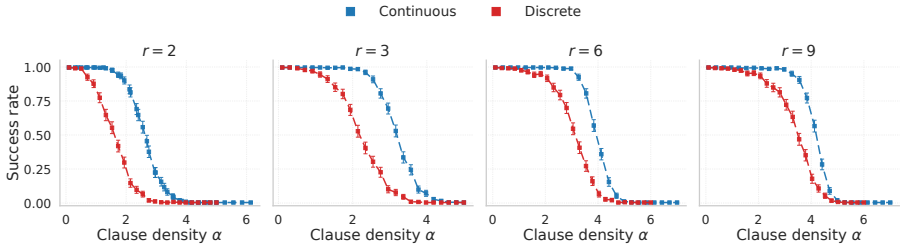


Figure 11: **4-NAE-SAT** ($N = 100$): Success rate as a function of clause density α for continuous (blue) and discrete (red) diffusions using BP denoiser. Each panel corresponds to a different locality radius for the denoisers $r \in \{2, 3, 6, 9\}$, success rates are computed over 500 random formulas. The error bars are 95% Wilson confidence intervals.

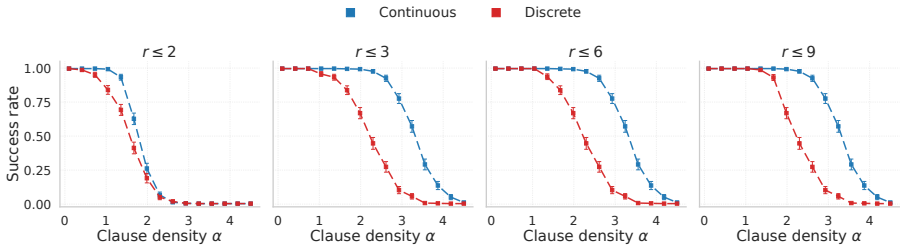


Figure 12: **4-Hypergraph-Bicoloring** ($N = 100$): Success rate as a function of clause density α for continuous (blue) and discrete (red) diffusions using BP denoiser. Each panel corresponds to the optimal performance obtained with a locality radius less or equal than $r \in \{2, 3, 6, 9\}$ (in this case we observed strong non-monotone behavior in r). Success rates are computed over 500 random formulas. The error bars are 95% Wilson confidence intervals.

$i \in V$ to factor nodes $a \in F$ $(\nu_{i\to a}(+1), \nu_{i\to a}(-1))$ (or equivalently, the log-likelihoods $h_{i\to a} = (1/2)log\{\nu_{i\to a}(+1)/\nu_{i\to a}(-1)\}$. to be i.i.d. with common distribution Q.

To define Q, recall the definition of free-boundary Gibbs measure on the infinite tree T, μ_T^{free}, defined in Section D.1. Then Q is the distribution of the log-likelihood at the root under μ_T^{free}, i.e. the distribution of

$$h_o := \frac{1}{2} \log \frac{\mu_T^{t,\boldsymbol{y}}(x_o = +1)}{\mu_T^{t,\boldsymbol{y}}(x_o = -1)} \,. \tag{44}$$

It is standard to derive the following distributional equation for the Q (involving the auxiliary distribution $\hat{Q}$)

$$\widehat{Q}(u) = \int \delta_{u - f(h_1^{k-1})} \prod_{i=1}^{k-1} \mathrm{d}Q(h_i) \,, \tag{45}$$

$$Q(h) = \frac{t}{2}\delta_{+\infty}(h) + \frac{t}{2}\delta_{-\infty}(h) + (1-t)Q^0(h) \,, \tag{46}$$

$$Q^0(h) = \sum_{m_+, m_- \geq 0} \mathsf{P}_{m_+}\mathsf{P}_{m_-} \int \delta_{h - \sum_{i=1}^{m_+} u_i^+ + \sum_{i=1}^{m_-} u_i^-} \prod_{i=1}^{m_+} \mathrm{d}\widehat{Q}(u)(u_i^+) \prod_{i=1}^{m_-} \mathrm{d}\widehat{Q}(u)(u_i^-) \,, \tag{47}$$

In order to approximate the distribution Q, we use again a finite sample approximation described in Algorithm 6. This yields two samples of size N, representing respectively $\widehat{Q}(u)$ and Q^0 :

$$\widehat{\mathcal{Q}}_N := \{u_1, \dots, u_N\}, \qquad \mathcal{Q}_N := \{h_1, \dots, h_N\} \,. \tag{48}$$

These empirical populations are used to initialize the $r = 1$ BP denoiser (for discrete diffusions) as follows. To estimate the marginal of a variable x_i, we run BP on a local neighborhood of x_i for a single iteration. For each directed variable-to-clause message $h_{j\to a}$, we set the initial condition according to whether x_j has already been decoded:

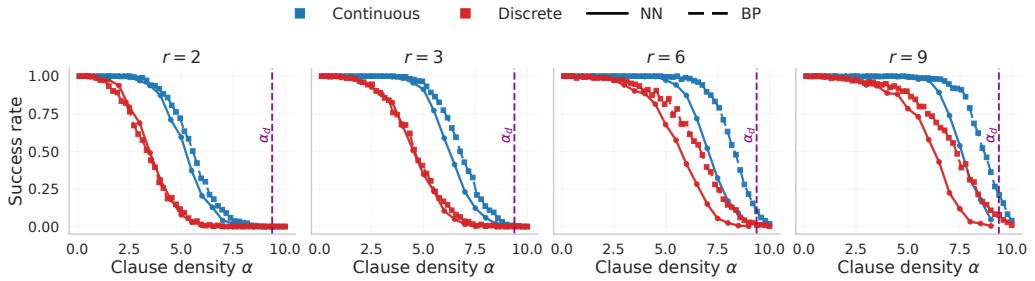


Figure 13: **4-SAT ($N = 100$): learned NN denoiser (solid lines) vs. BP denoiser (dashed lines).** Success rate (probability of generating actual solutions) as a function of clause density α for continuous (blue) and discrete (red) diffusions. Each panel corresponds to a different locality radius for the denoisers $r \in \{2, 3, 6, 9\}$. Success rates are computed over 500 random formulas. The vertical line marks the dynamical phase transition $\alpha_{\rm d}(k = 4) \approx 9.38$.

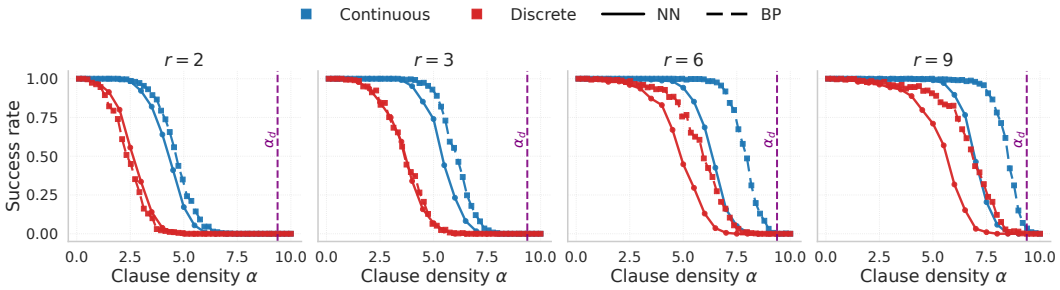


Figure 14: **4-SAT ($N = 300$)**: learned NN denoiser (solid lines) vs. BP denoiser (dashed lines). Success rate as a function of clause density α for continuous (blue) and discrete (red) diffusions. Each panel corresponds to a denoiser $r \in \{2, 3, 6, 9\}$, success rates are computed over 500 random formulas. The vertical line marks the dynamic phase transition threshold $\alpha_{\rm d} \approx 9.38$ for $k = 4$.

- If x_j is fixed, we initialize $h_{j\to a} = +\infty$ when $x_j = +1$ and $h_{j\to a} = -\infty$ when $x_j = -1$.
- Otherwise, we initialize $h_{j\to a}$ by drawing an independent sample from the empirical population $\mathcal{Q}_N$.

We then compute the resulting belief at i and sample x_i from this marginal.

F Other CSPs

Beyond k-SAT and k-XORSAT, we also evaluate on two additional classes of CSPs using the BP denoiser.

NAE-SAT. Figure 11 reports success rate results for k-Not-All-Equal SAT (NAE-SAT), with $k = 4$ (see Section C.4 for definitions). using both continuous and discrete diffusions and BP denoiser, at several values of the locality radius r. As for SAT and XORSAT, we observe a significant advantage of continuous over discrete diffusiions.

Hypergraph bicoloring. Figure 12 reports results for hypergraph bicoloring, with $k = 4$ (i.e., every 'hyperdge' comprises 4 vertices), see Section C.5 for definitions. Again, we use continuous and discrete diffusions and BP denoiser, at several values of the locality radius. One important observation is that, when using uniform initialization $\mathrm{BP}_{\rm u}(r)$ (which is what we do here) we observe a non-monotonicity of accuracy in the number of BP iterations. We thus thake for each r the optimal accuracy with number of iterations at most r. Our main conclusion is confirmed here: continuous diffusions outperforms discrete diffusions.

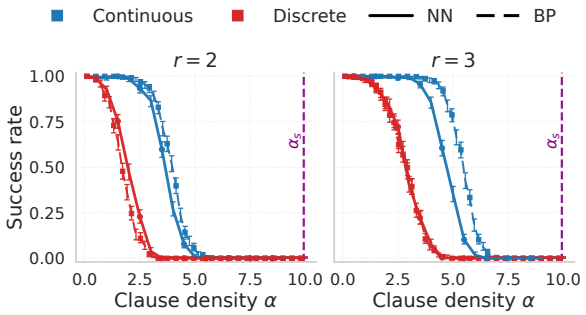


Figure 15: **4-SAT** ($N = 900$): learned NN denoiser (solid lines) vs. BP denoiser (dashed lines). Success rate as a function of clause density α for continuous (blue) and discrete (red) diffusions. Each panel corresponds to a different locality radius for the denoisers $r \in \{2, 3\}$, success rates are computed over 500 random formulas. The vertical line marks the dynamic phase transition threshold $\alpha_s \approx 9.38$ for $k = 4$. The error bars are 95% Wilson confidence intervals.

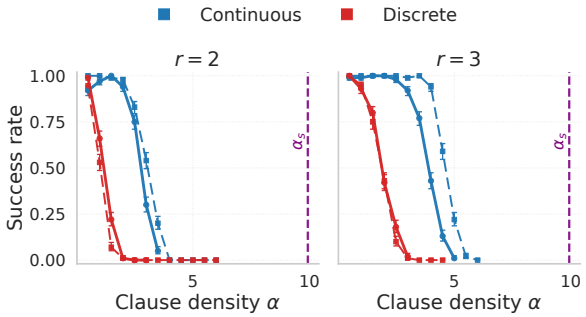


Figure 16: **4-SAT** ($N = 5000$): learned NN denoiser (solid lines) vs. BP denoiser (dashed lines). Success rate as a function of clause density α for continuous (blue) and discrete (red) diffusions. Each panel corresponds to a different locality radius for the denoisers $r \in \{2, 3\}$, success rates are computed over 500 random formulas. The vertical line marks the dynamic phase transition threshold $\alpha_s \approx 9.38$ for $k = 4$. The error bars are 95% Wilson confidence intervals.

Algorithm 6: Population Dynamics for Cavity Initialization (Discrete Diffusion)

Input: clause size k; clause density α; diffusion reveal rate $t \in [0, 1]$; population size N; iterations T; (optional) pooling window $\Delta \geq 1$; update map $f(\cdot)$.

Output: populations $\widehat{\mathcal{Q}}_N = \{u_1, \ldots, u_N\}$ (approximating $\widehat{Q}(u)$) and $\mathcal{Q}_N = \{h_1, \ldots, h_N\}$ (approximating $Q^0(h)$).

Initialize $h_i \leftarrow 0$ and $u_i \leftarrow 0$ for all $i \in \{1, \ldots, N\}$.

for $\tau = 1, 2, \ldots, T$ **do**

  Update $\widehat{\mathcal{Q}}_N$**:**

  for $i = 1, 2, \ldots, N$ **do**

    Sample indices $j(1), \ldots, j(k-1) \sim_{\text{i.i.d.}} \mathsf{Unif}([N])$ (with replacement).

    for $a = 1, 2, \ldots, k-1$ **do**

      Set $\hat{h}_a \leftarrow +\infty$ w.p. $t/2$, $\hat{h}_a \leftarrow -\infty$ w.p. $t/2$, and $\hat{h}_a \leftarrow h_{j(a)}$ w.p. $1 - t$.

    $u_i \leftarrow f(\hat{h}_{k-1})$.

  Update $\mathcal{Q}_N$**:**

  for $i = 1, 2, \ldots, N$ **do**

    Sample $m_+, m_- \sim_{\text{i.i.d.}} \mathsf{Poisson}(k\alpha/2)$.

    Sample indices $j_+(1), \ldots, j_+(m_+) \sim_{\text{i.i.d.}} \mathsf{Unif}([N])$ (with replacement).

    Sample indices $j_-(1), \ldots, j_-(m_-) \sim_{\text{i.i.d.}} \mathsf{Unif}([N])$ (with replacement).

    $h_i \leftarrow \sum_{a=1}^{m_+} u_{j_+(a)} - \sum_{b=1}^{m_-} u_{j_-(b)}$.

if $\Delta > 1$ **then**

  (Optional) Pool the last Δ iterations to obtain $N\Delta$ samples for each population.

return $(\widehat{\mathcal{Q}}_N, \mathcal{Q}_N)$

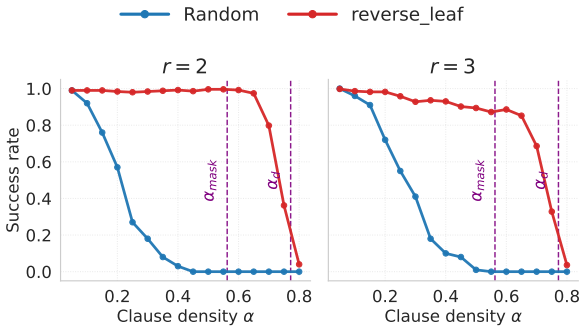


Figure 17: **4-XORSAT** ($N = 100$): reversed-leaf (red) vs random (blue) decoding ordering. Success rate as a function of clause density α obtained using **discrete** diffusion with learned NN denoiser. Each panel corresponds to a denoiser $r \in \{2, 3\}$, success rates are computed over 500 random formulas. The two vertical lines are the theoretical thresholds for random (α_{mask}) and optimal (α_{d}) decoding ordering.

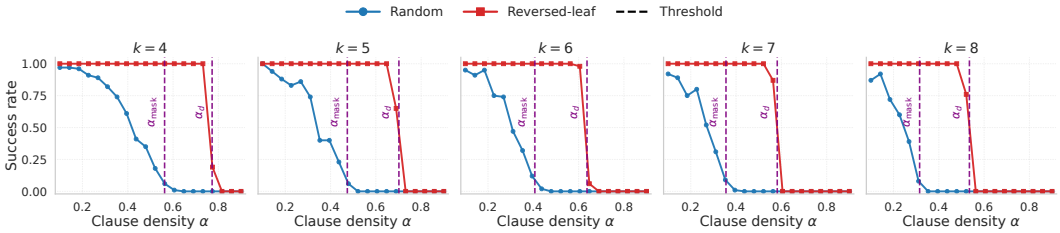


Figure 18: k**-XORSAT** ($n = 1000$) : reverse-leaf (red) vs random (blue) decoding ordering. Success rate as a function of clause density α for $k \in \{4, \dots, 8\}$ obtained using discrete diffusion with BP denoiser. The two vertical lines are the theoretical thresholds for random (α_{mask}) and optimal (α_{d}) decoding ordering at every value of k.

G Further numerical results

In this section, we report certain experimental results not presented in the main paper due to space limitations.

k-SAT for larger N. In addition to the k-SAT results for $N = 300$ with an $r = 3$ local denoiser shown in Figure 1, we report performance for several additional settings (:

- Figure 14: $N = 300$ with $r \in \{2, 3, 6, 9\}$;
- Figure 13: $N = 100$ with $r \in \{2, 3, 6, 9\}$;
- Figure 15: $N = 900$ with $r \in \{2, 3\}$;
- Figure 16: $N = 5000$ with $r \in \{2, 3\}$.

In each of these cases we report the success rate with continuous and discrete diffusions, both using BP and learned denoisers.

The results are remarkably consistent and confirm the conclusions of Section 3.1. Continuous diffusion consistently outperforms discrete diffusion, and the learned local NN denoiser performs comparably to the BP denoiser. For $N = 900$ (Figure 15) and $N = 5000$ (Figure 16), we additionally include 95% Wilson confidence intervals, indicating that statistical uncertainty is negligible relative to the observed effect.

Reverse leaf ordering for k-XORSAT. We carry out additional experiments with XORSAT, comparing reversed leaf ordering and random ordering:

- Figure 18: impact of ordering on discrete diffusion at $N = 1000$ for $k \in \{4, 5, 6, 7, 8\}$. (Figure 4 in the main text focused on $N = 300$, $k = 4$.)

- Figure 17: impact of ordering on discrete diffusion with an NN denoiser. (Figure 4 in the main text used BP denoiser.)

Again, we observe that the conclusions in the main text are confirmed by these additional experiments.

H Proofs

H.1 Proof of Theorem 3.1

We first recall the definition of local (weak) convergence for Gibbs measures (see e.g. [47]), adapting it to the present setting. For concreteness, we consider k-SAT, but the case of general CSP can be treated analogously.

Recall the definition of 'free boundary Gibbs measure' from Section D.2. In particular, let $T = (V_T, F_T, E_T)$ be the infinite tree SAT formula rooted at variable node $o \in V_T$, defined by the fact that the L-generation tree $T(L)$ of Section 3.2 is obtained by taking the first L generations of T. HEre we regard the free boundary Gibbs measure as a joint law over $(T, \boldsymbol{x}, \boldsymbol{y})$ with $\boldsymbol{x}$ a satisfying assignment of T. The assumption of the theorem implicitly requires that the the subsequential limit used to define this measure is unique. We lift this to a joint law of $(T, \boldsymbol{x}, \boldsymbol{y})$, by letting $\boldsymbol{y} = \sqrt{t}\boldsymbol{x} + \sqrt{1-t}\boldsymbol{g}$, for $\boldsymbol{g} = (g_i)_{i\in V_T} \sim_{iid} \mathsf{N}(0,1)$. We denote this joint law by $\mathbb{P}$ and its marginal on $T(r)$ by $\mathbb{P}_{T(r)}$

Let $I \sim \mathsf{Unif}([N])$, and let $\mathbb{P}_{N,r}$ be the joint law of $(\mathsf{B}_G(I,r), \boldsymbol{x}_{\mathsf{B}_G(I,r)}, \boldsymbol{y}_{\mathsf{B}_G(I,r)})$ (were we view $\mathsf{B}_G(I,r)$ as rooted at I. We say that the measure $\mu_G^{\boldsymbol{y},t}$ converges locally to the free boundary Gibbs measure if, for every $r > 0$, we have $\mathbb{P}_{N,r} \Rightarrow \mathbb{P}_{T(r)}$, where $\Rightarrow$ denotes weak convergence. Since $\hat{m}_{\mathrm{BP}_{\mathsf{c}}(r)}(\,\cdot\,)$ is a bounded continuous function of $\boldsymbol{Y}_{\mathsf{B}_G(I,r)} = (\mathsf{B}_G(I,r), \boldsymbol{y}_{\mathsf{B}_G(I,r)})$ (plus additional randomness for the initialization), we have

$$\lim_{N\to\infty} \mathbb{E}_N\big\{\big(x_I - \hat{m}_{\mathrm{BP}_{\mathsf{c}}(r)}(\boldsymbol{Y}_{\mathsf{B}_G(I,r)})\big)^2\big\} = \mathbb{E}\big\{\big(x_o - \hat{m}_{\mathrm{BP}_{\mathsf{c}}(r)}(\boldsymbol{Y}_{T(r)})\big)^2\big\}\,. \tag{49}$$

Further , the infimum on the right-hand side of Eq. (6) can be restricted to estimators $\hat{m}$ such that $\sup_{\boldsymbol{Y}} |\hat{m}(\boldsymbol{Y})| \le 1$ (because $|x_I| \le 1$) Define

$$\mathcal{M}_{L,C} := \Big\{\hat{m} : \|\hat{m}\|_\infty \le 1,\ |\hat{m}(\mathsf{B},\boldsymbol{y}) - \hat{m}(\mathsf{B},\boldsymbol{y}')| \le L\|\boldsymbol{y}-\boldsymbol{y}'\|\ \forall |\mathsf{B}| \le C;\ \forall \boldsymbol{y},\boldsymbol{y}'\Big\}\,. \tag{50}$$

Using the fact that $\lim_{C\to\infty} \sup_N \mathbb{P}(|\mathsf{B}_G(i,r)| \ge C) = 0$, and the fact that Lipschitz functions are dense in L^2, we conclude that for every ε there exists L, C independent of N such that

$$\inf_{\hat{m}\in\mathcal{M}_{L,C}} \mathbb{E}_N\big\{\big(x_i - \hat{m}(\boldsymbol{Y}_{\mathsf{B}_G(I,r)})\big)^2\big\} \le \inf_{\hat{m}} \mathbb{E}_N\big\{\big(x_i - \hat{m}(\boldsymbol{Y}_{\mathsf{B}_G(I,r)})\big)^2\big\} + \frac{\varepsilon}{4}\,. \tag{51}$$

Let $\hat{m}_N \in \mathcal{M}_{L,C}$ an estimator that nearly achieves the infimum, i.e. such that

$$\mathbb{E}_N\big\{\big(x_i - \hat{m}_N(\boldsymbol{Y}_{\mathsf{B}_G(I,r)})\big)^2\big\} \le \inf_{\hat{m}\in\mathcal{M}_{L,C}} \mathbb{E}_N\big\{\big(x_i - \hat{m}(\boldsymbol{Y}_{\mathsf{B}_G(I,r)})\big)^2\big\} + \frac{\varepsilon}{4}\,. \tag{52}$$

Therefore using the last two displays we have:

$$\mathbb{E}_N\big\{\big(x_i - \hat{m}_N(\boldsymbol{Y}_{\mathsf{B}_G(I,r)})\big)^2\big\} \le \inf_{\hat{m}} \mathbb{E}_N\big\{\big(x_i - \hat{m}(\boldsymbol{Y}_{\mathsf{B}_G(I,r)})\big)^2\big\} + \frac{\varepsilon}{2}\,. \tag{53}$$

By Ascoli-Arzelá and a diagonal argument, for any sequence (N_j^0) there exists a further subsequence $(N_j) \subseteq (N_j^0)$ and $\hat{m}_\infty \in \mathcal{M}_{L,C}$ such that, for all $j \ge 1$ $\sup_{|\mathsf{B}|\le C, \|\boldsymbol{y}\|^2 \le Cj} |\hat{m}_{N_j}(\mathsf{B},\boldsymbol{y}) - \hat{m}_\infty(\mathsf{B},\boldsymbol{y})| \le \varepsilon \cdot 2^{-j}$. Hence, using the previous equation, for all j

$$\begin{aligned}\mathbb{E}_{N_j}\big\{\big(x_I - \hat{m}_\infty(\boldsymbol{Y}_{\mathsf{B}_G(I,r)})\big)^2\big\} \le \inf_{\hat{m}} \mathbb{E}_{N_j}\big\{\big(x_I - \hat{m}_\infty(\boldsymbol{Y}_{\mathsf{B}_G(I,r)})\big)^2\big\} + \frac{\varepsilon}{2^j} \\ + \frac{\varepsilon}{2} + 4\cdot\mathbb{P}\Big(\|\boldsymbol{Y}_{\mathsf{B}_G(I,r)}\|^2 \ge Cj; |\mathsf{B}_G(I,r)| \le C\Big)\,.\end{aligned}$$

Hence, using standard tail bounds on chi-squared random variables, and eventually taking a further subsequence, we conclude that for any sequence $(N_j^0 : j \ge 1)$, there exists a further subsequence $(N_j : j \ge 1) \subseteq (N_j^0 : j \ge 1)$, such that

$$\mathbb{E}_{N_j}\big\{\big(x_I - \hat{m}_\infty(\boldsymbol{Y}_{\mathsf{B}_G(I,r)})\big)^2\big\} \le \inf_{\hat{m}} \mathbb{E}_{N_j}\big\{\big(x_I - \hat{m}(\boldsymbol{Y}_{\mathsf{B}_G(I,r)})\big)^2\big\} + \frac{3}{4}\varepsilon\,. \tag{54}$$

Taking the limit $j \to \infty$, and using the fact that the above holds for every initial subsequence (N_j^0), we get (notice that the limit of the left-hand side depends on the subsequence (N_j) only through $\hat{m}_\infty$):

$$\mathbb{E}\big\{\big(x_o - \hat{m}_\infty(\boldsymbol{Y}_{T(r)})\big)^2\big\} \le \liminf_{N\to\infty} \inf_{\hat{m}\in\mathcal{M}_{L,C}} \mathbb{E}_N\big\{\big(x_I - \hat{m}_\infty(\boldsymbol{Y}_{\mathsf{B}_G(I,r)})\big)^2\big\} + \frac{3}{4}\varepsilon\,. \tag{55}$$

Therefore

$$\inf_{\hat{m}} \mathbb{E}\big\{\big(x_o - \hat{m}(\boldsymbol{Y}_{T(r)})\big)^2\big\} \le \liminf_{N\to\infty} \inf_{\hat{m}\in\mathcal{M}_{L,C}} \mathbb{E}_N\big\{\big(x_I - \hat{m}_\infty(\boldsymbol{Y}_{\mathsf{B}_G(I,r)})\big)^2\big\} + \frac{3}{4}\varepsilon\,. \tag{56}$$

The infimum on the left hand side is achieved by $\hat{m}_{\mathrm{BP}_{\mathsf{C}}(r)}$ by construction and therefore by Eq. (49)

$$\lim_{N\to\infty} \mathbb{E}_N\big\{\big(x_I - \hat{m}_{\mathrm{BP}_{\mathsf{C}}(r)}(\boldsymbol{Y}_{\mathsf{B}_G(I,r)})\big)^2\big\} \le \liminf_{N\to\infty} \inf_{\hat{m}} \mathbb{E}_N\big\{\big(x_I - \hat{m}_\infty(\boldsymbol{Y}_{\mathsf{B}_G(I,r)})\big)^2\big\} + \frac{3}{4}\varepsilon\,. \tag{57}$$

This proves our claim.

H.2 Proof of Theorem 3.3

It is convenient to replace the $\{+1,-1\}$ representation of variables by the $\{0,1\}$ representation, with multiplications replaced by sum modulo two (we still use $\boldsymbol{x}$ to denote the variables in this domain). A formula can therefore be represented by a matrix $\boldsymbol{A} \in \{0,1\}^{M\times N}$, where row $a \in \{1,\dots M\}$ has non-zero entries in correspondence to the indices $i_1(a),\dots,i_k(a)$ of variable participating constraint a, and a vector $\boldsymbol{b} \in \{0,1\}^M$ which represents the signs s_a (with $b_a = 0$ if and only if $s_a = +1$). In this representation, $\boldsymbol{x} \in \{0,1\}^N$ is a solution if and only if $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \mod 2$.

We will assume that the leaf removal succeeds. Under this ordering the matrix $\boldsymbol{A}$ is upper triangular, with ones on the main diagonal:

$$\boldsymbol{A} = \Big[\boldsymbol{T}\Big|\boldsymbol{B}\Big]\,, \tag{58}$$

where $\boldsymbol{T} \in \{0,1\}^{M\times M}$ is square upper triangular, with ones on the diagonal, and $\boldsymbol{B} \in \{0,1\}^{M\times(N-M)}$, If we correspondingly decompose $\boldsymbol{x} = [\mathbf{u}|\mathbf{v}]$ with $\mathbf{v} \in \{0,1\}^{N-M}$, we have that each $\mathbf{v}$ extends uniquely to a solution of the overall linear system.

Hence a uniformly random solution can be generated by letting $\mathbf{u} \sim \mathsf{Unif}(\{0,1\}^{N-M})$, and extending it uniquely. For $i \le M$, we let $x_i^*(\mathbf{u})$ be such unique extension. We next will describe this procedure in a different but obviously equivalent fashion.

We note that, in this notation, the reversed leaf ordering is (without loss of generality): $\sigma(1) = N$, $\sigma(2) = N-1$,$\dots\sigma(N) = 1$. Hence a uniformly random solution can be generated by the following procedure:

- (I) For $\ell \in \{0,1,\dots,N-M-1\}$, sample $\overline{x}_{\sigma(\ell+1)} \sim \widehat{\mu}_G^{\sigma,\ell}(\,\cdot\,) = \mathsf{Unif}(\{0,1\})$.
- (II) For $\ell \in \{N-M,1,\dots,N-1\}$, sample $\overline{x}_{\sigma(\ell+1)} \sim \widehat{\mu}_F^{\sigma,\ell}(\,\cdot\,)$, where $\widehat{\mu}_F^{\sigma,\ell}(x_{\sigma(\ell+1)} = x^*_{\sigma(\ell+1)}(\overline{\boldsymbol{x}}_{M+1:N})) = 1$.

Note that the last point is just an indirect way of saying that we set $\overline{x}_{\sigma(\ell+1)} = x^*_{\sigma(\ell+1)}(\overline{\boldsymbol{x}}_{M+1:N})$.

Comparing with the definition of masked discrete diffusions, we are only left with the task of showing that the $\mathrm{BP}_{\mathsf{u}}(\infty)$ marginals coincide with the marginals $\widehat{\mu}$ described above. Fix any $\ell \in \{0,\dots,N-1\}$. By the above construction, the system is always solvable (under the assumption that leaf removal is successful), and therefore (by recasting $\boldsymbol{x}$ as $\boldsymbol{x} - \boldsymbol{x}_0$ for $\boldsymbol{x}_0$ a specific solution), we can and will assume $\boldsymbol{b} = 0$. By the same argument, we can assume $\overline{\boldsymbol{x}}^{\sigma}_{1:\ell} = \mathbf{0}_\ell$.

Under these reductions the the analysis of BP_{u} reduces to the analysis of BP decoding for a low-density parity check code under the binary erasure channel. We recall three elementary facts about this theory (see, e.g. [48, Chapter 3] or [19, Section 15.3]): (i) The BP estimate of the marginal of x_i after any number r of iterations is either $\nu_i^{(r)} = (1,0)$ (where $\nu_i^{(r)} = (\nu_i^{(r)}(0),\nu_i^{(r)}(1))$) or

$\nu_i^{(r)} = (1/2, 1/2)$; (ii) Estimates are 'monotone' in the iteration number, namely if $\nu_i^{(r)} = (1, 0)$, then $\nu_i^{(r')} = (1, 0)$ for all $r' \geq r$; (iii) If $\nu_i^{(r)} = (1, 0)$ at some r, then the actual conditional marginal of variable x_i is $\mu_G^{\sigma,\ell}(x_i = 0 | \overline{\boldsymbol{x}}_{1:\ell}^{\sigma} = \mathbf{0}_\ell) = 1$.

We are now in position to finish the argument:

- (I) For $\ell \in \{-1, 0, \dots, N - M - 2\}$ (hence $\sigma(\ell + 1) \in \{N, \dots, M + 1\}$) we know that $\mu^{\sigma,\ell}(x_{\sigma(\ell+1)} = 0 | \overline{\boldsymbol{x}}_{1:\ell}^{\sigma} = \mathbf{0}_\ell) = 1/2$ (because as explained above the marginal distribution on $\boldsymbol{x}_{M+1,N}$ is uniform on $\{0, 1\}^{N-M}$) and therefore by property (iii) above the BP marginal is also $(1/2, 1/2)$ for all iterations.
- (II) For $\ell \in \{N - M - 1, \dots, N\}$ (hence $\sigma(\ell + 1) \in \{M, \dots, 1\}$) we know that (by the upper triangular structure of $\boldsymbol{A}$, variable $x_{\sigma(\ell+1)}$ appears in an equation of the form $x_{\sigma(\ell+1)} + x_{j_2} + \cdots + x_{j_k} = 0, \mod 2$, where $j_2, \dots, j_k > \sigma(\ell+1)$, and hence we are conditioning to $x_{j_2} = 0, \dots x_{j_k} = 0$. Hence after one iteration of BP (i.e. for $r \geq 1$), $\nu_{\sigma(\ell+1)}^{(r)} = (1, 0)$.

This proves that the $\mathrm{BP}_{\mathrm{u}}(\infty)$ estimate coincide with $\widehat{\mu}$ given above, hence completing the proof of the first part of the statement. Finally the 'in particular' part of the statement follows immediately from the fact that the leaf removal procedure succeeds with high probability for $\alpha < \alpha_{\mathrm{d}}(k)$.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The claims made match the obtained theoretical and experimental results.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: Limitations are adequately discussed.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes].

Justification: Full assumptions and proofs are provided.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes].

   Justification: All information required to reproduce experimental results is provided.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: While code is not provided, the paper provides sufficient detail on the experiments to reproduce all results.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: All experimental details are provided.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes].

   Justification: Error bars are reported.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes].

   Justification: All information regarding computational resources used is provided.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducteded in the paper conforms in every respect with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: There is no societal impact of the work performed.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [N/A]

    Justification: The paper does not use existing assets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A].

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.