Blocked Gibbs meets Diffusion Transformers: Unsupervised Learning for Constraint Optimization

Yudong Will Xu, Wenhao Li, Xiaoyu Wang, Scott Sanner, Elias B. Khalil

UNIVERSITY OF TORONTO

Outline

Background
- Constraint Optimization Problems
- Diffusion
- Diffusion for Constraint Optimization Problems

Our Work

Constraint Optimization

Reasoning under constraints is a crucial challenge in many domains.

Traditional solvers remain complex and often slow.

Machine learning techniques have been explored to accelerate solving.

Figure from Khalil (2024) [1]

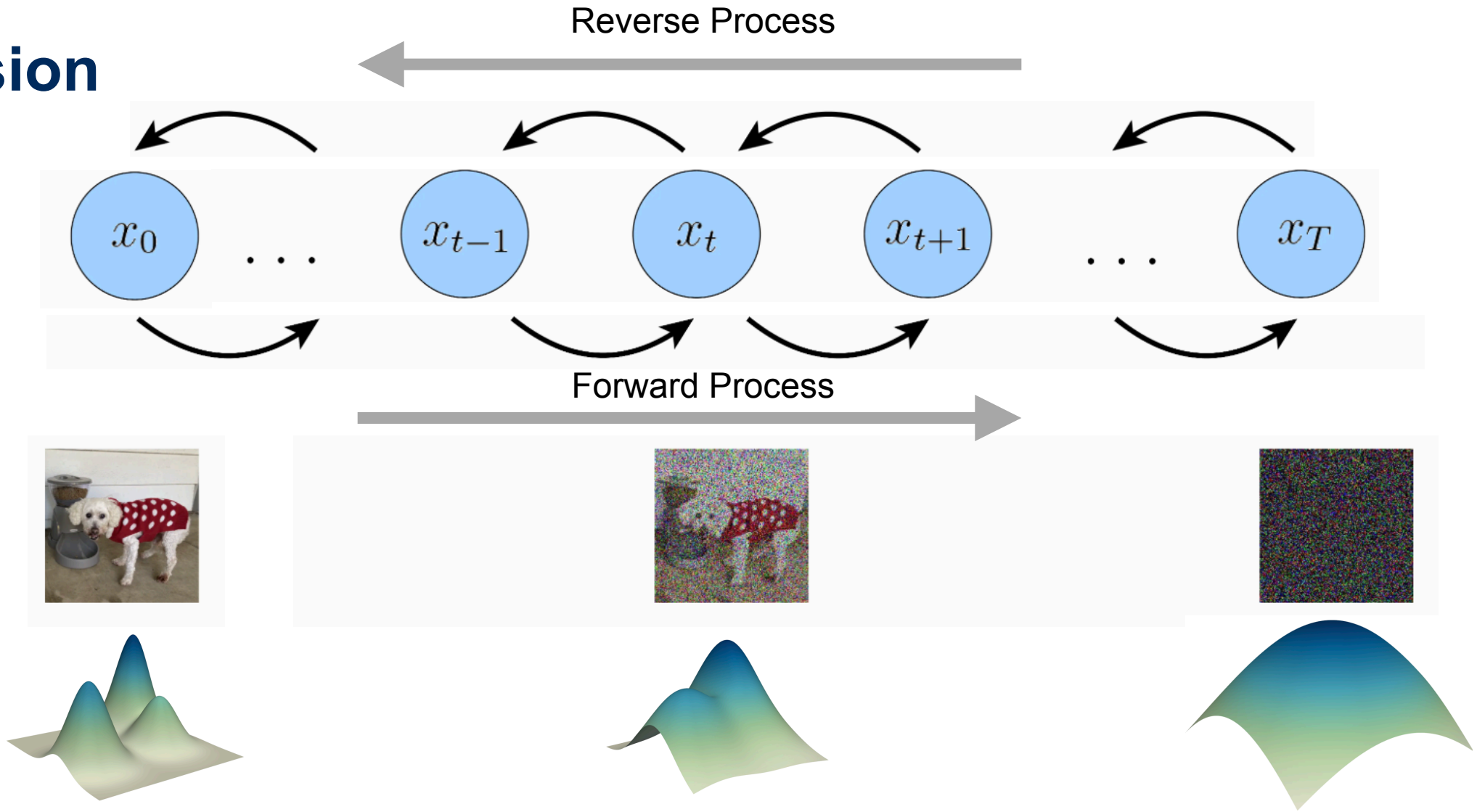
Learning to Solve Constraint Optimization

2		1	9					4
		3				6	9	
			5					3
1		7	2	4		5		9
			3			4	7	8
						3		
9	4	6		5				
			4	6			1	
5		2		3	7			6

2	6	1	9	8	3	7	5	4
8	5	3	7	1	4	6	9	2
7	9	4	5	2	6	1	8	3
1	3	7	2	4	8	5	6	9
6	2	5	3	9	1	4	7	8
4	8	9	6	7	5	3	2	1
9	4	6	1	5	2	8	3	7
3	7	8	4	6	9	2	1	5
5	1	2	8	3	7	9	4	6

Today: Diffusion Models

Diffusion

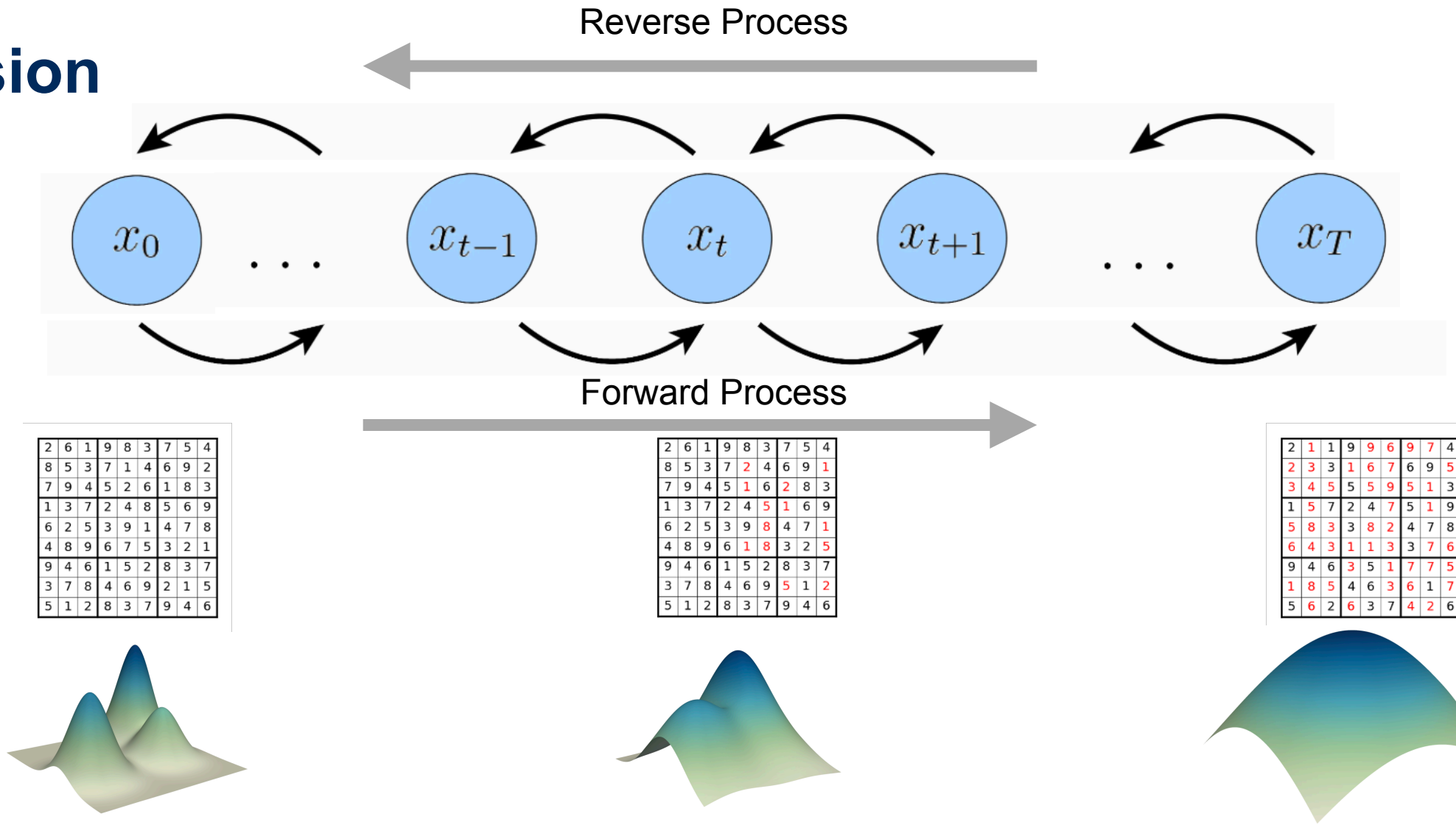


$X_0 \sim p_{data}(X)$

$X_T \sim p_{noise}(X)$

Figure from [6][7]

Diffusion



$X_0 \sim p_{data}(X)$

$X_T \sim p_{noise}(X)$

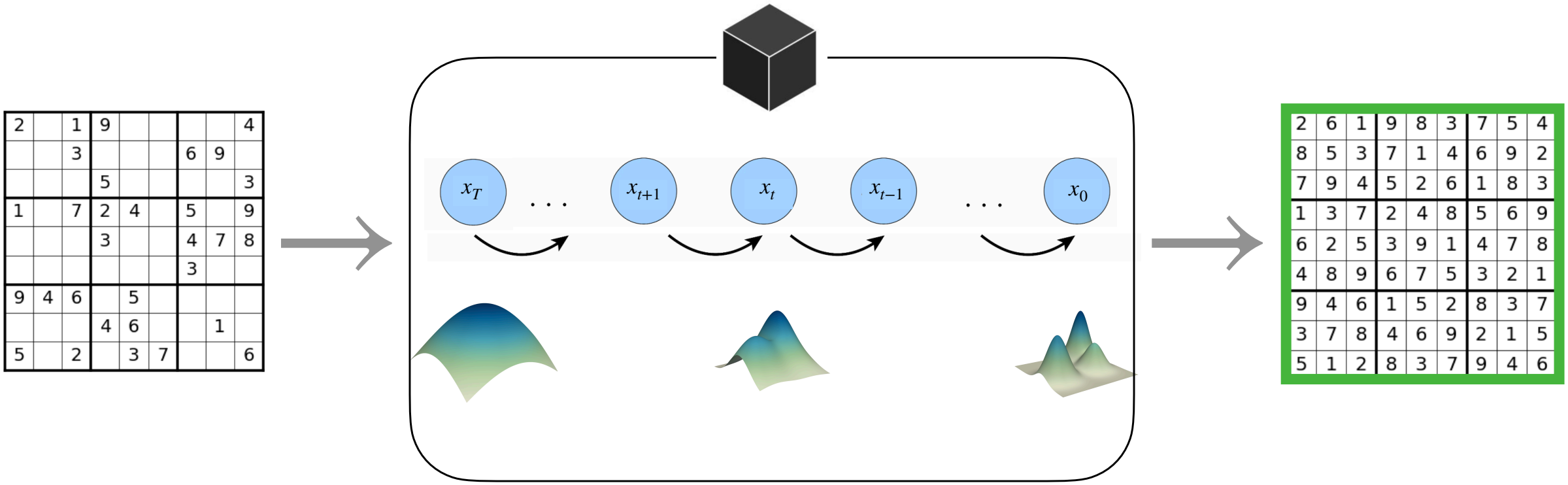
DIFUSCO[8] Figure from [6][7]

Learning to Solve Constraint Optimization

2		1	9					4
		3				6	9	
			5					3
1		7	2	4		5		9
			3			4	7	8
						3		
9	4	6		5				
			4	6			1	
5		2		3	7			6

2	6	1	9	8	3	7	5	4
8	5	3	7	1	4	6	9	2
7	9	4	5	2	6	1	8	3
1	3	7	2	4	8	5	6	9
6	2	5	3	9	1	4	7	8
4	8	9	6	7	5	3	2	1
9	4	6	1	5	2	8	3	7
3	7	8	4	6	9	2	1	5
5	1	2	8	3	7	9	4	6

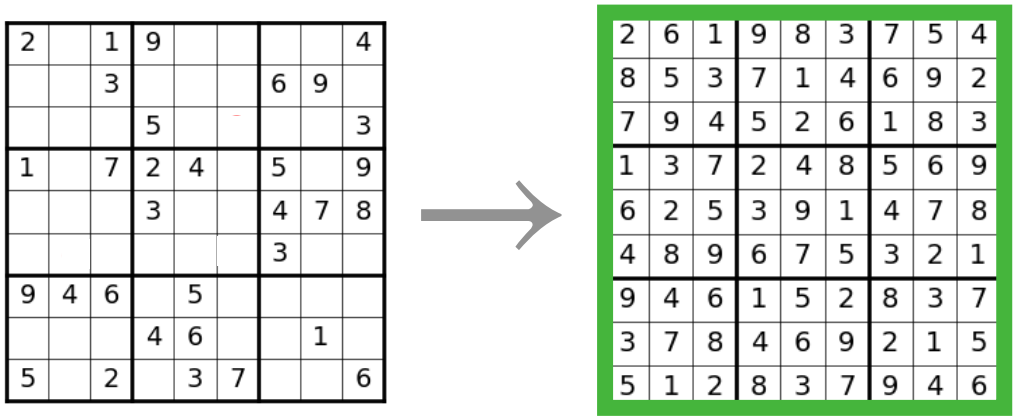
Learning to Solve Constraint Optimization

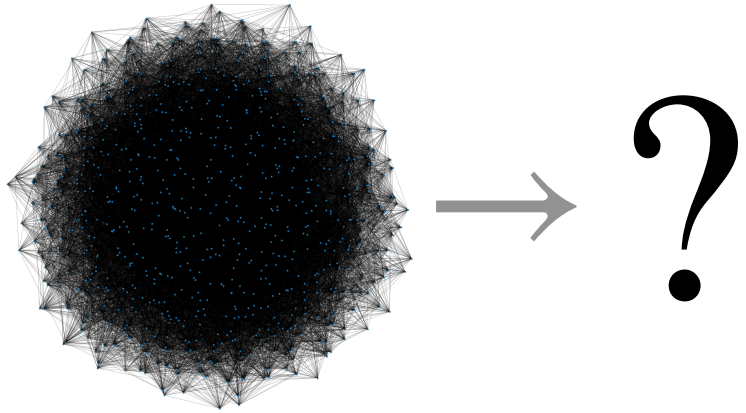


Existing approaches has two limitations

Limitation 1: Learning Approach

Supervised Learning





Solutions are not easy to obtain, especially for harder instances.

What other signal can we use to train the models?

Figure from [2]

Constraint Programming Formulation

Minimize objective

$$\min_{X \in \mathcal{X}} \quad f(X)$$

$$\text{s.t.} \quad c_j(X) = \text{true} \quad \forall j$$

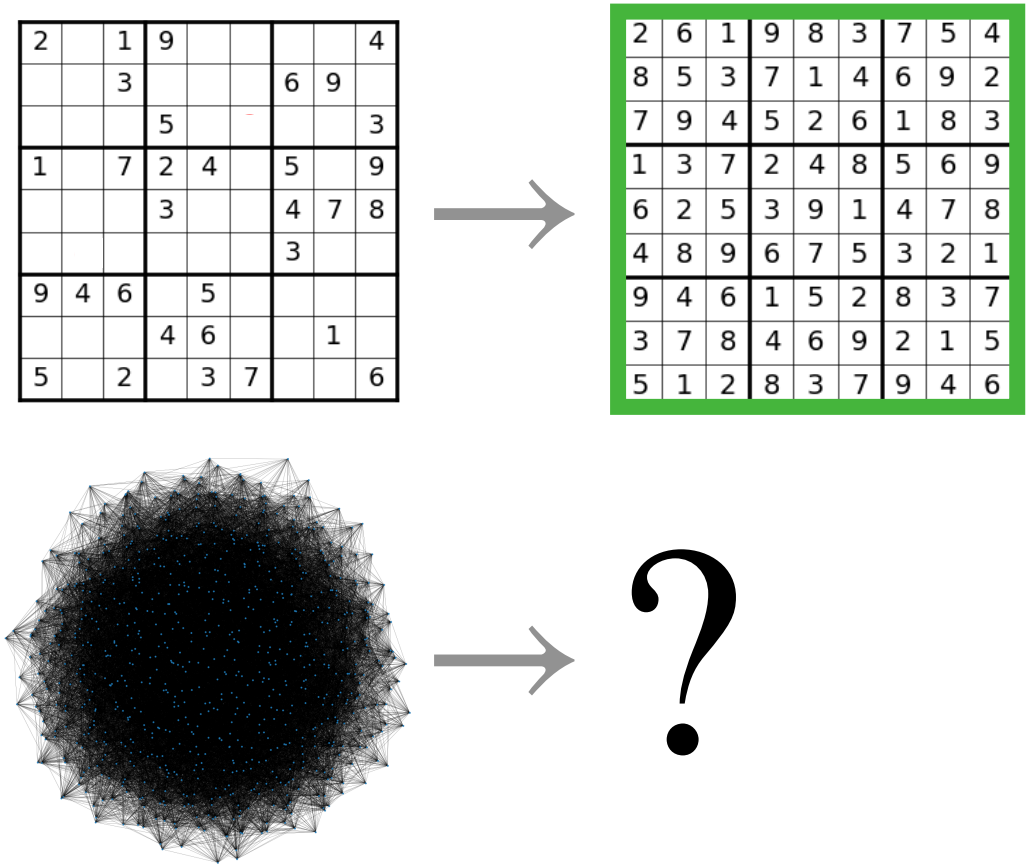
Satisfy constraints

2		1	9					4
		3				6	9	
			5					3
1		7	2	4		5		9
			3			4	7	8
						3		
9	4	6		5				
			4	6			1	
5		2		3	7			6

$$AllDifferent(X_{1,1}, X_{1,2}, \ldots, X_{3,3})$$
$$\ldots$$
$$AllDifferent(X_{1,1}, X_{1,2}, \ldots, X_{3,3})$$
$$\ldots$$
$$AllDifferent(X_{1,1}, X_{1,2}, \ldots, X_{3,3})$$
$$\ldots$$

Learning Approach

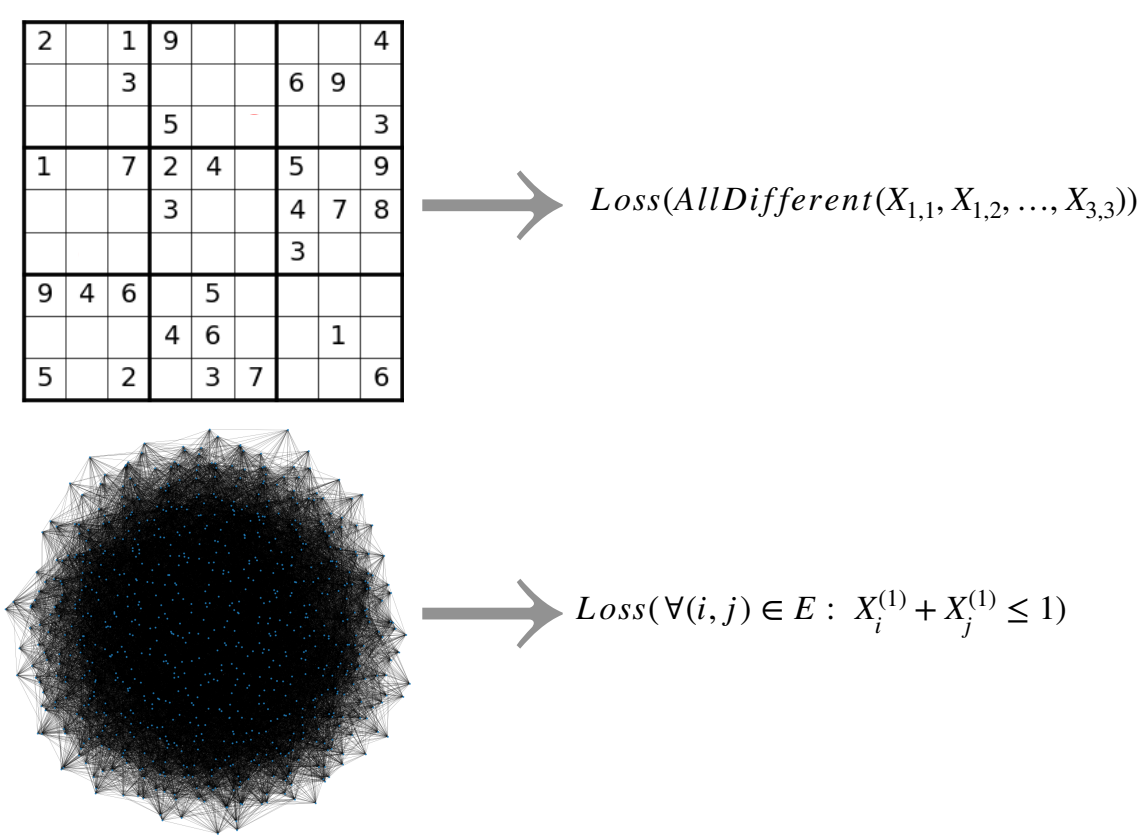
Supervised Learning



Unsupervised Learning

(Distant-supervised)

(Self-supervised)



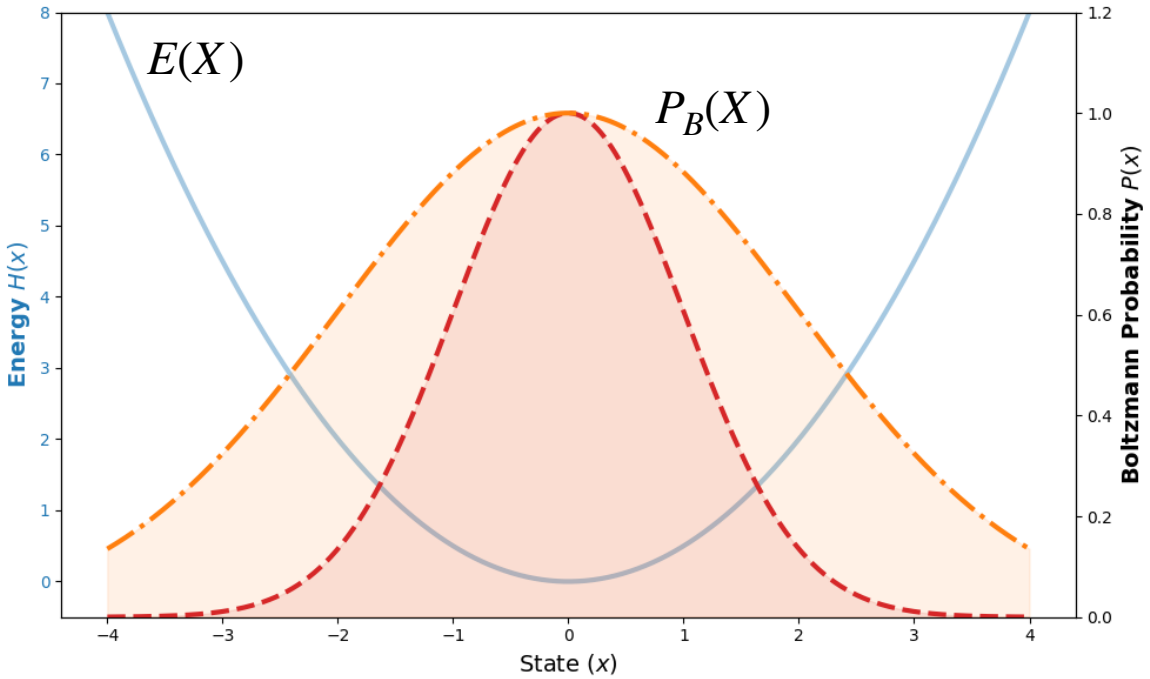
A Different Perspective: CP as Boltzmann Distribution

$$\min_{X \in \mathcal{X}} \; f(X) \quad \text{s.t.} \quad c_j(X) = \text{true} \quad \forall j$$

$$\min_{X \in \mathcal{X}} \; \overbrace{f(X) + \sum_j \lambda_j \phi_j(X)}^{E(X)}, \qquad \phi_j(X) = 0 \iff c_j(X) = \text{True}$$

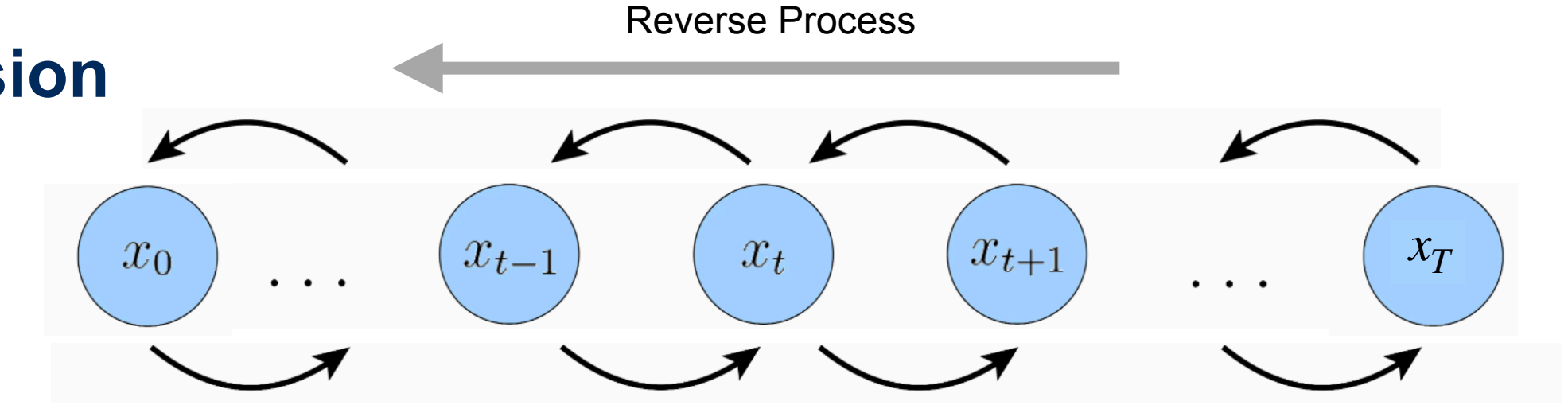
"Energy Function"

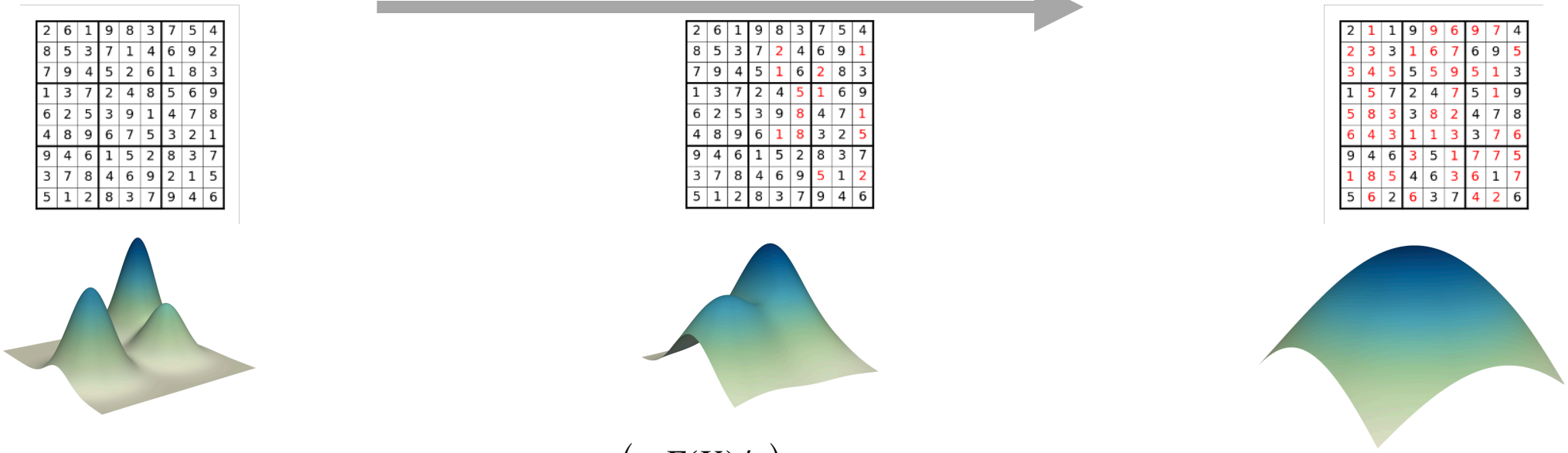
$$p_B(X) = \frac{\exp\left(-E(X)/\tau\right)}{\sum_{X' \in \mathcal{X}} \exp\left(-E(X')/\tau\right)}$$



Sampling from this distribution gives us a solution to the CP problem!

Diffusion





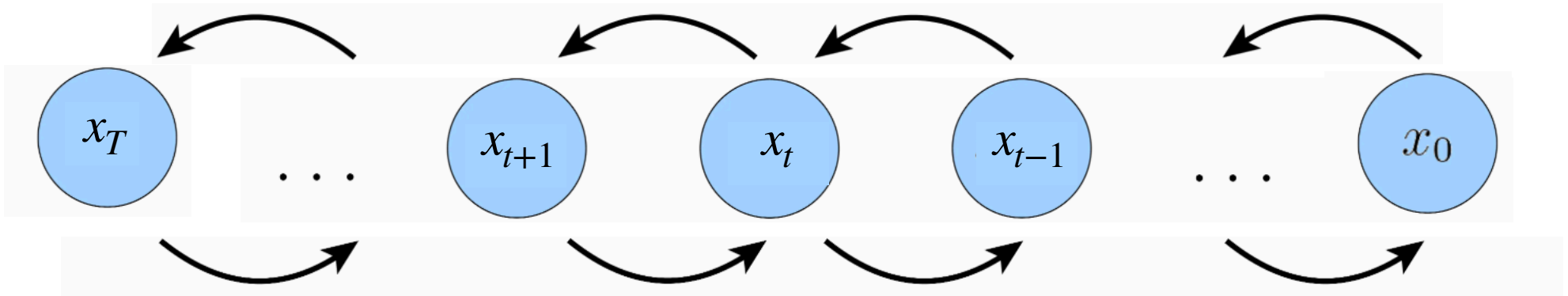
$$X_0 \sim p_{data}(X) = p_B(X) = \frac{\exp\left(-E(X)/\tau\right)}{\sum_{X' \in \mathcal{X}} \exp\left(-E(X')/\tau\right)}$$

$$X_T \sim p_{noise}(X)$$

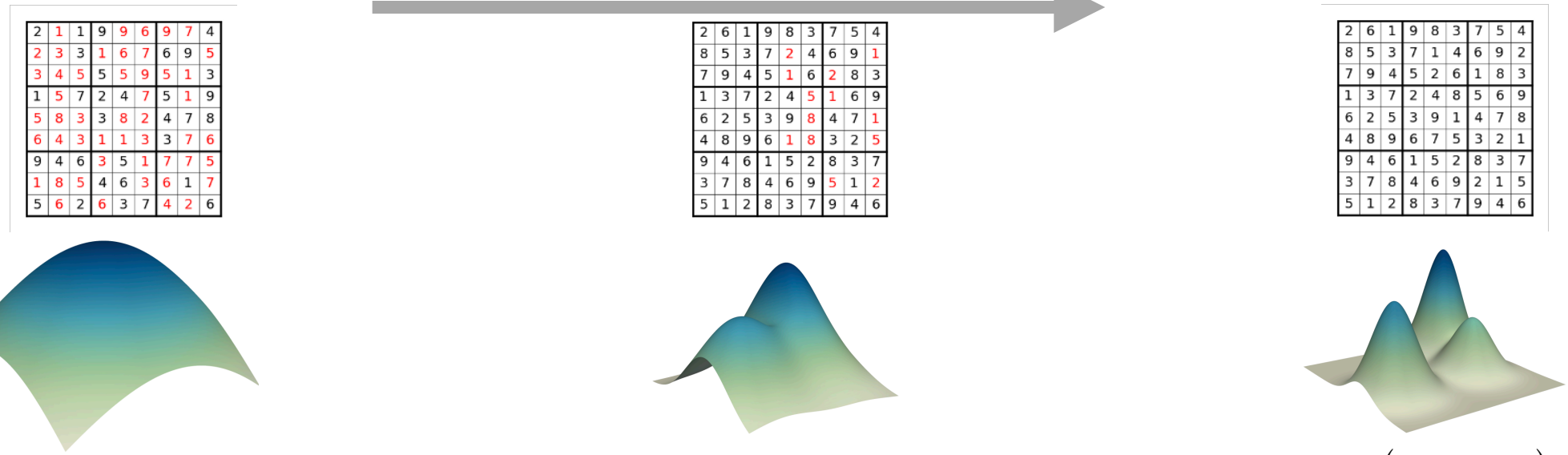
DIFUSCO[8] Figure from [6][7]

Unsupervised Diffusion

Forward Process



Reverse Process



$X_T \sim p_{noise}(X)$

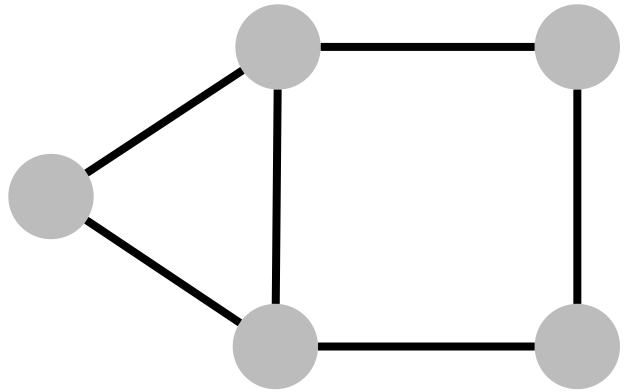
Start with the noise!

$$X_0 \sim p_B(X) = \frac{\exp\left(-E(X)/\tau\right)}{\sum_{X' \in \mathcal{X}} \exp\left(-E(X')/\tau\right)}$$

Limitation 2: Architecture

- Focus on Combinatorial Optimization Problems that are
  - Binary
  - Graph-based

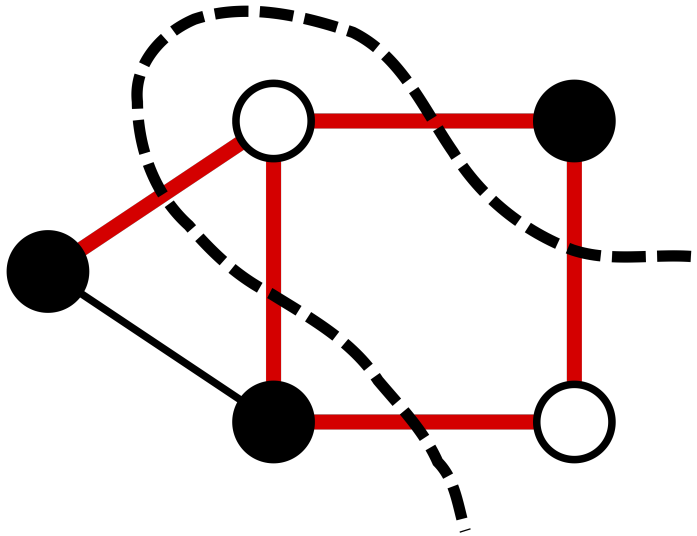
  Implement with Graph Neural Network (GNNs)



Modelling Assumption: GNNs rely on local message passing, which may fail to capture global dependencies effectively.

Training instability: GNNs may experience over-smoothing, leading to nearly indistinguishable node embeddings.

Applicability: Not every constraint optimization problem admits a natural graph formulation.



We look to Transformers to address these limitations.

Limitations of Existing Works

Supervised Learning
requires solved instances.

Unsupervised Learning
trained using the Boltzmann Distribution

Architectural limitations
from Graph Neural Networks

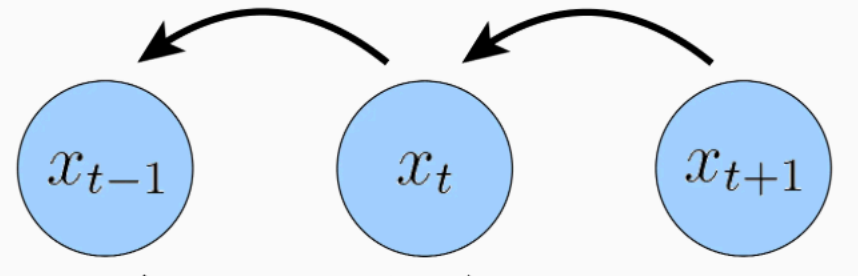
Use Transformers.

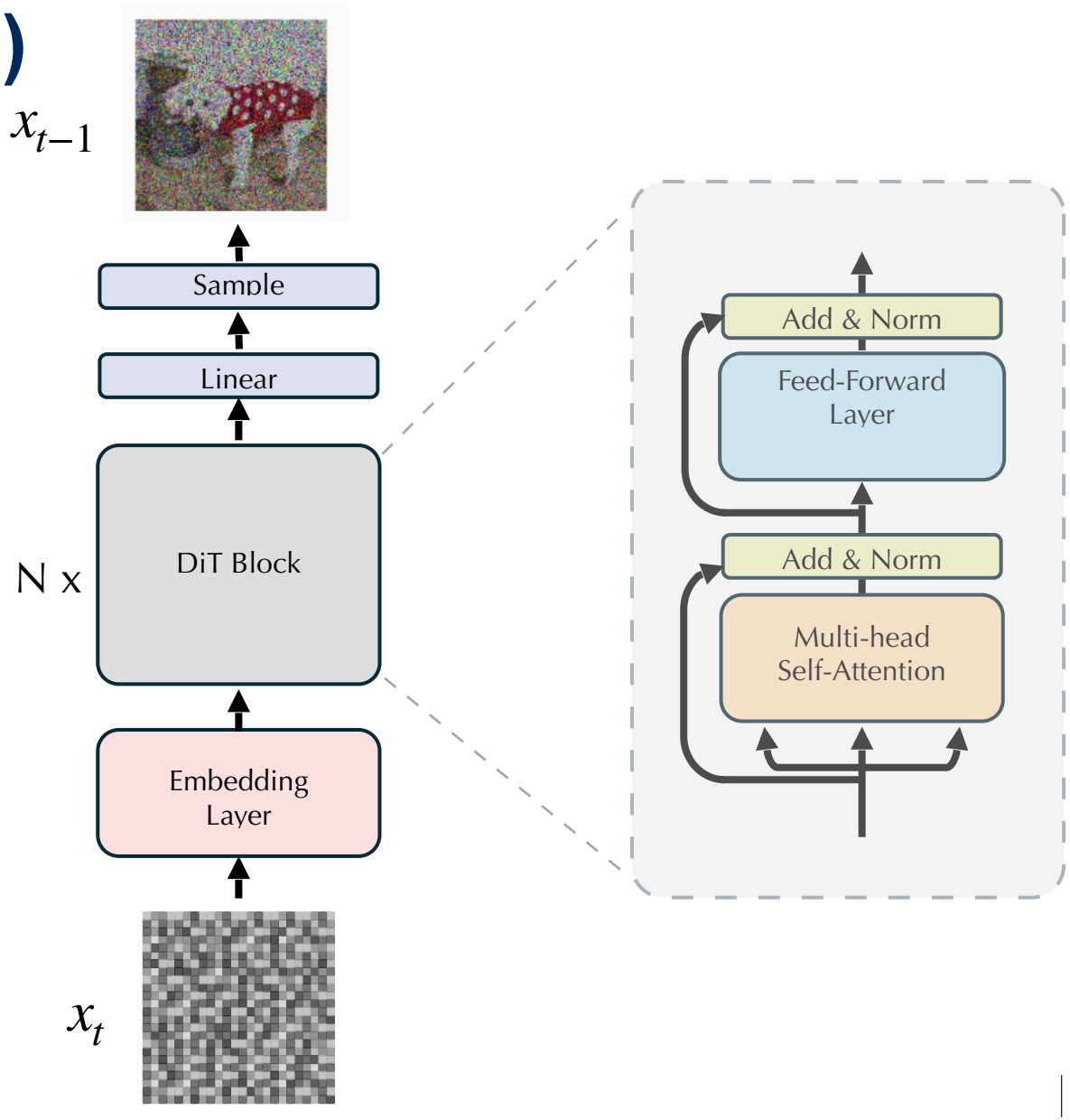
Outline

Background
- Constraint Optimization Problems
- Diffusion
- Diffusion for Constraint Optimization Problems

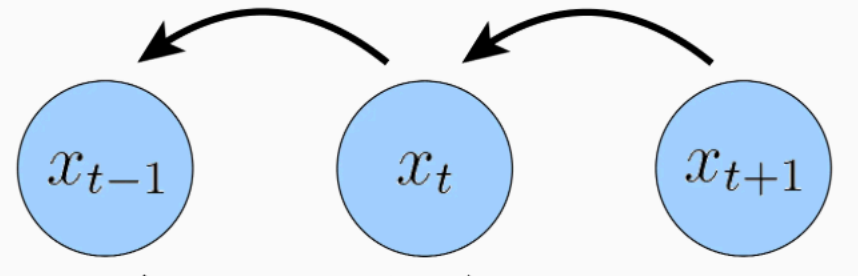
Our Work
- Blocked Gibbs Diffusion Transformers (BloGDiT)

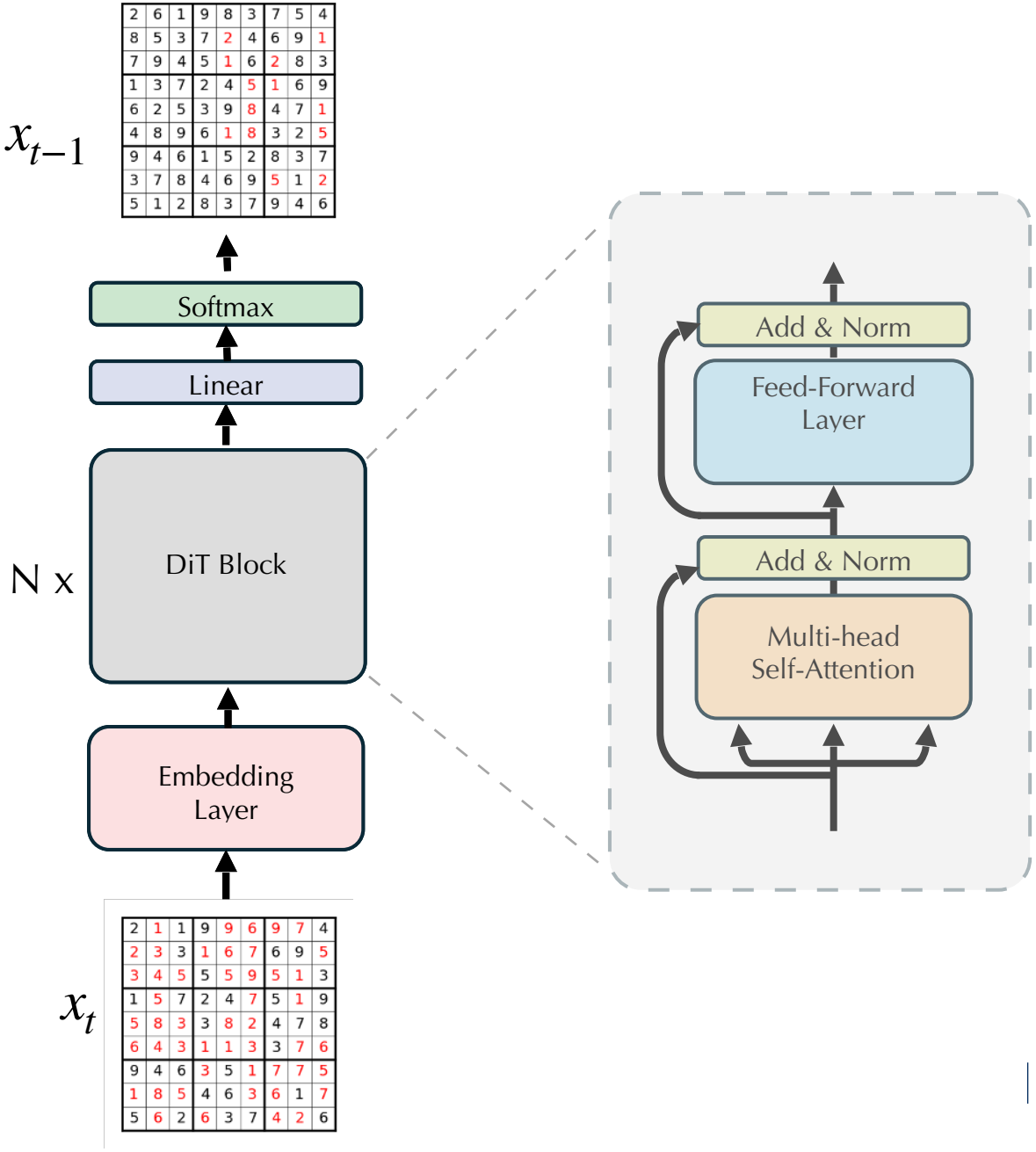
Diffusion Transformers (DiT)



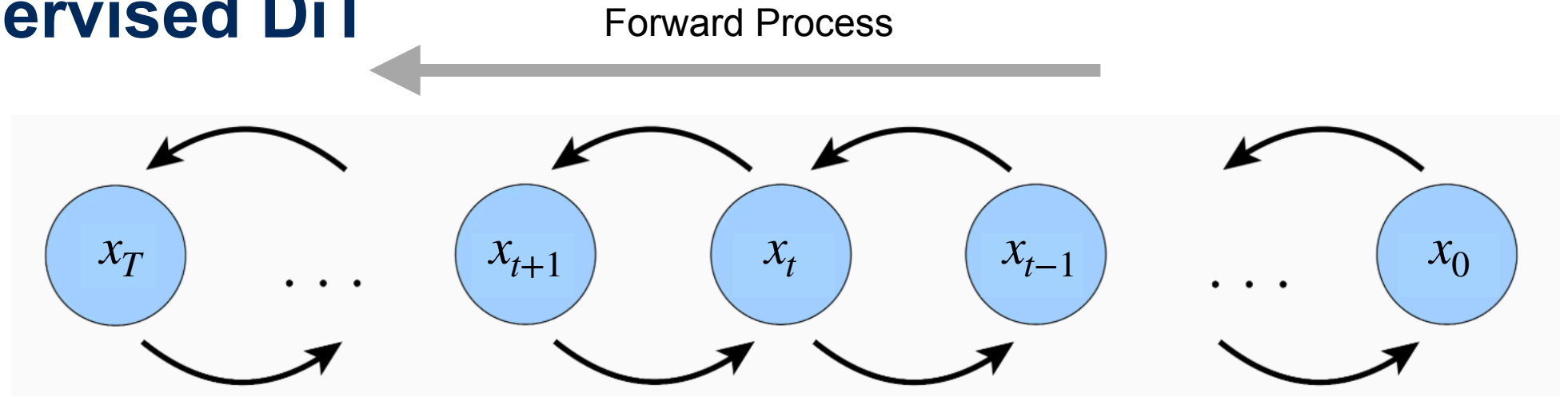


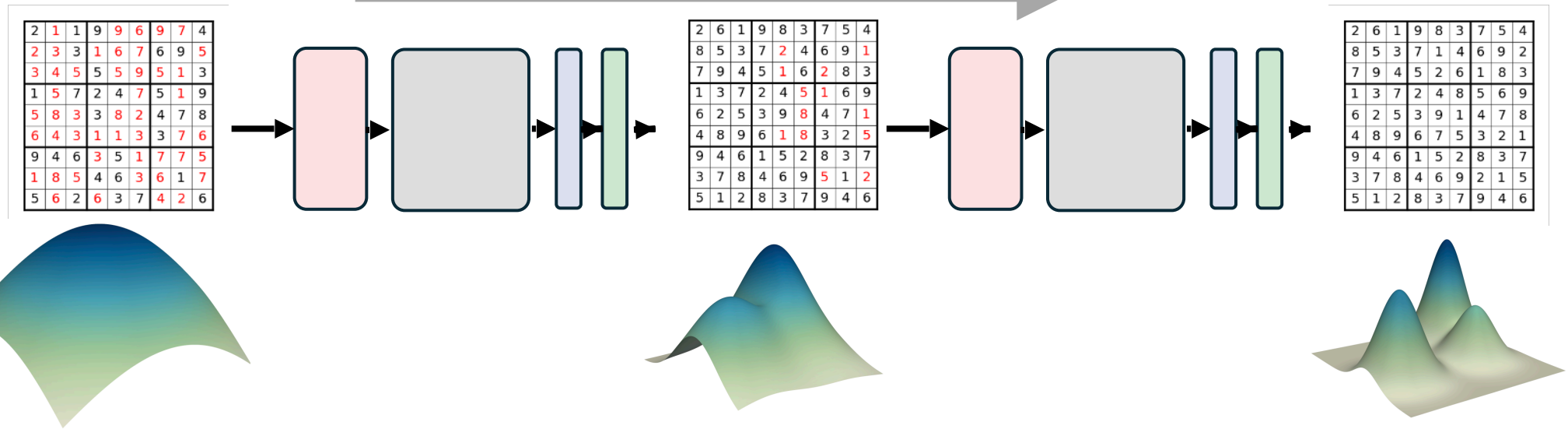
Diffusion Transformers





Unsupervised DiT





$X_T \sim p_{noise}(X)$

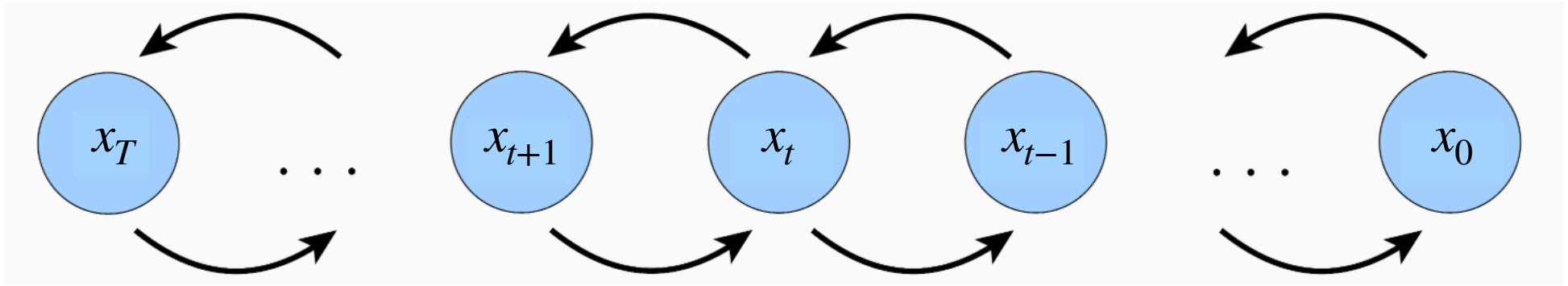
$p_B(X)$

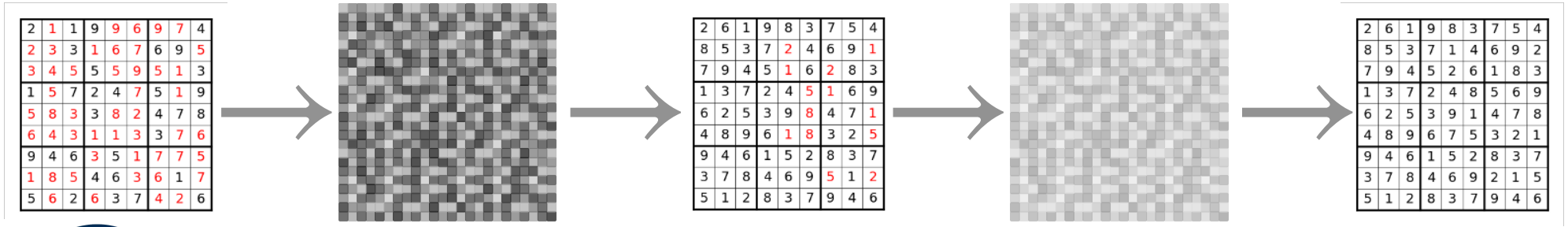
Unsupervised DiT

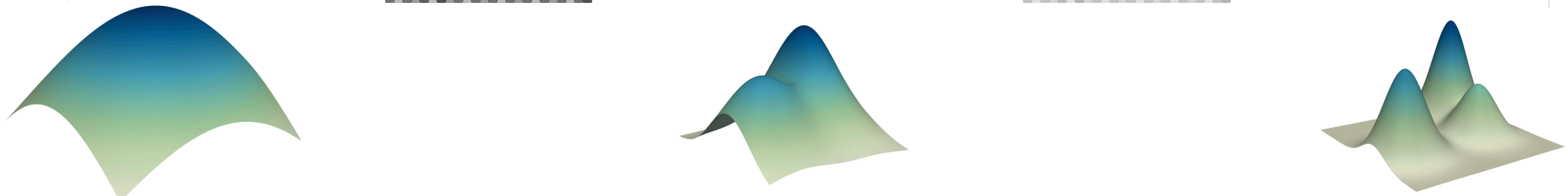
Method	In-distribution Test	OOD Test
OR-Tools	**100**	**100**
SATNet*	98.3	3.2
RRN*	<u>99.8</u>	28.6
Recurrent Transformer*	**100**	32.9
IRED*	99.4	62.1
ConsFormer	**100**	85.8
DiT	**100**	48.6

Does standard Diffusion make sense for CP?

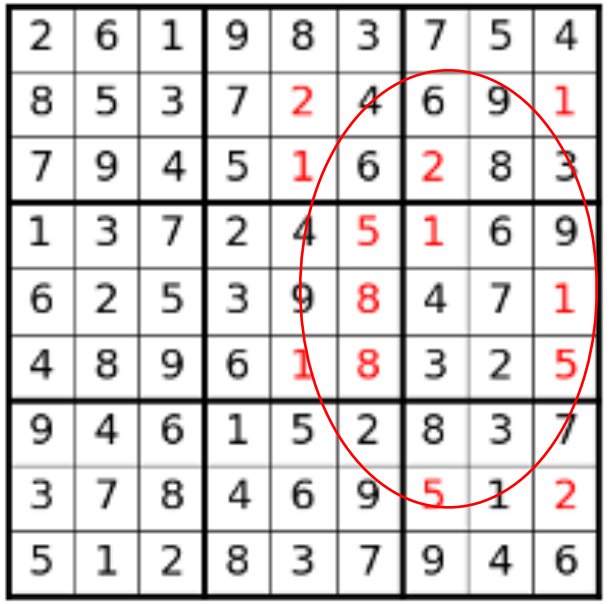
Does standard Diffusion make sense for CP?







Does standard Diffusion make sense for CP?



2	6	1	9	8	3	7	5	4
8	5	3	7	2	4	6	9	1
7	9	4	5	1	6	2	8	3
1	3	7	2	4	5	1	6	9
6	2	5	3	9	8	4	7	1
4	8	9	6	1	8	3	2	5
9	4	6	1	5	2	8	3	7
3	7	8	4	6	9	5	1	2
5	1	2	8	3	7	9	4	6

Should we really make updates to every variable?

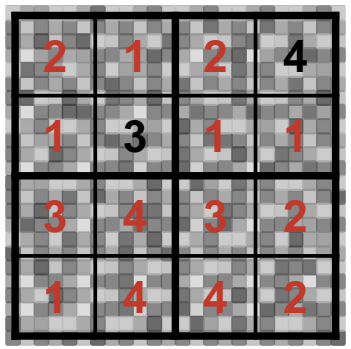
We should make localized, targeted updates instead.

We want our diffusion to do
Local / Large Neighbourhood Search.

Does standard Diffusion make sense for CP?

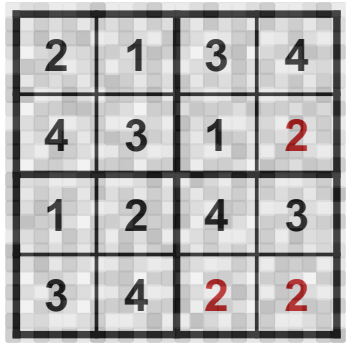
Standard Diffusion

Early Steps



Large Magnitude Noise to ALL Variables

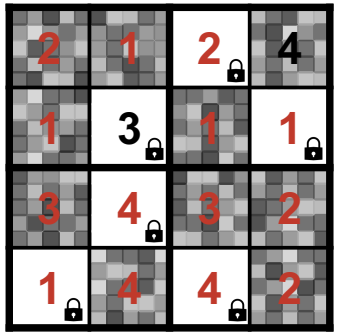
Late Steps



Small Magnitude Noise to ALL Variables

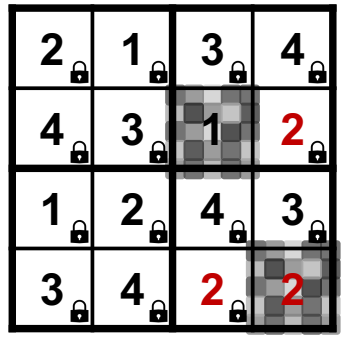
Blocked Gibbs Diffusion

Early Steps



Constant Noise to MANY Variables

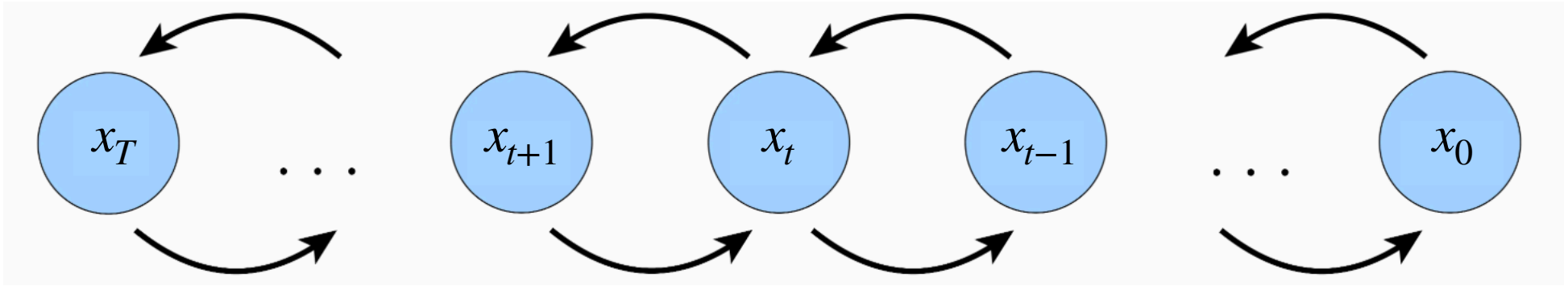
Late Steps

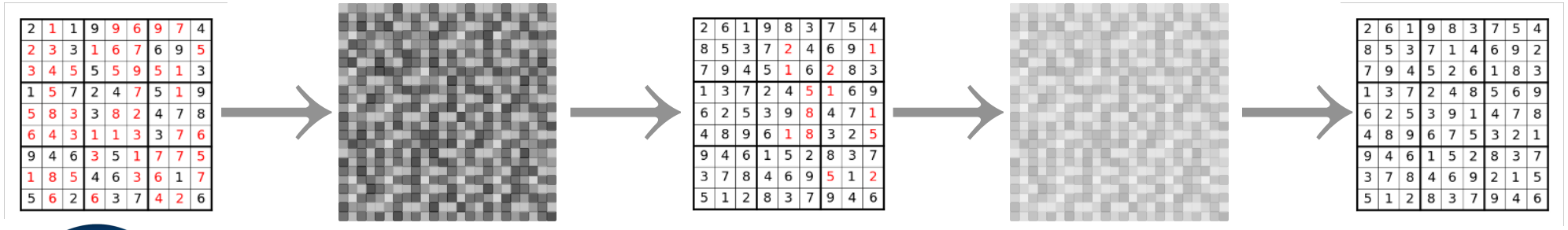


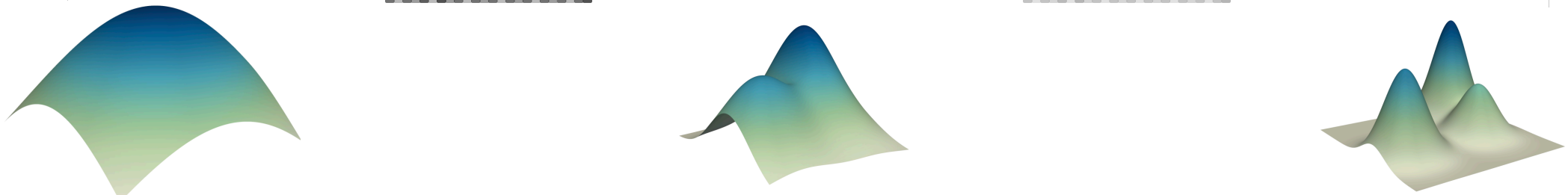
Constant Noise to FEWER Variables

What we want

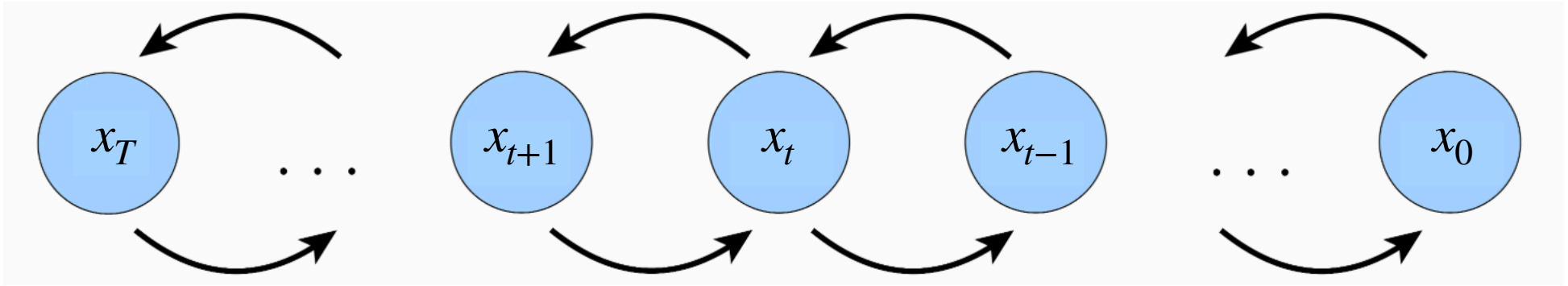
Unsupervised Blocked Gibbs DiT (BlogDiT)



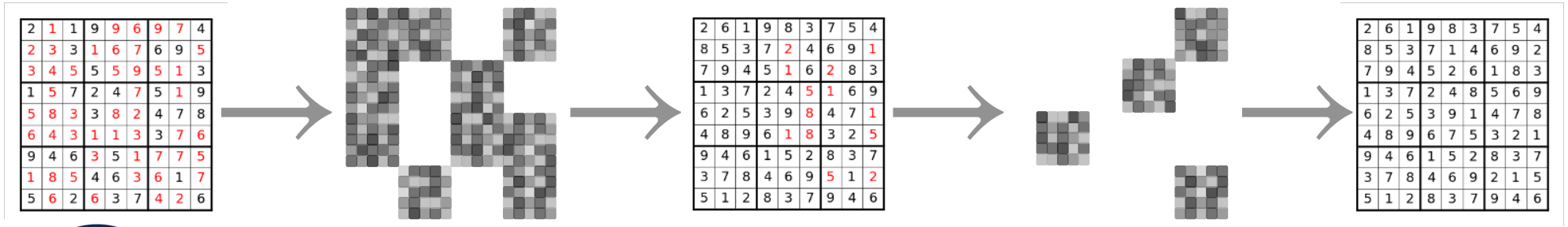


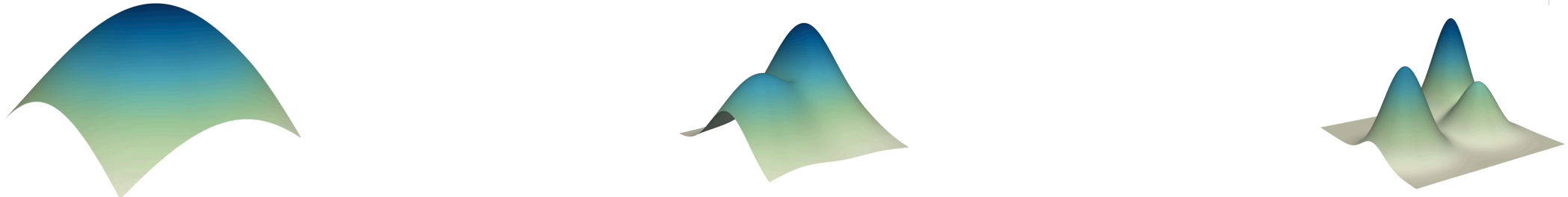


Unsupervised Blocked Gibbs DiT (BlogDiT)



$q_\theta^*(X)$

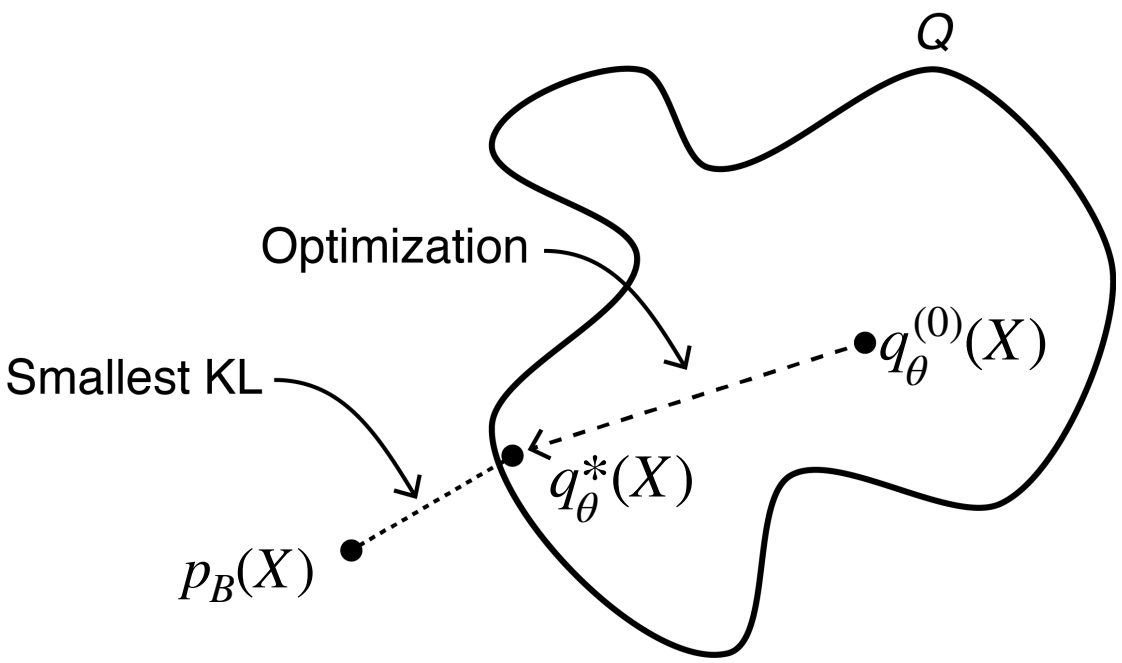




How do we train this?

$p_B(X)$

Training Objective: KL Divergence



$$\mathrm{KL}\left(p_B(X) \,\|\, q_\theta(X)\right) = \mathbb{E}_{X\sim p_B}\left[\log \frac{p_B(X)}{q_\theta(X)}\right]$$

We don't have data!

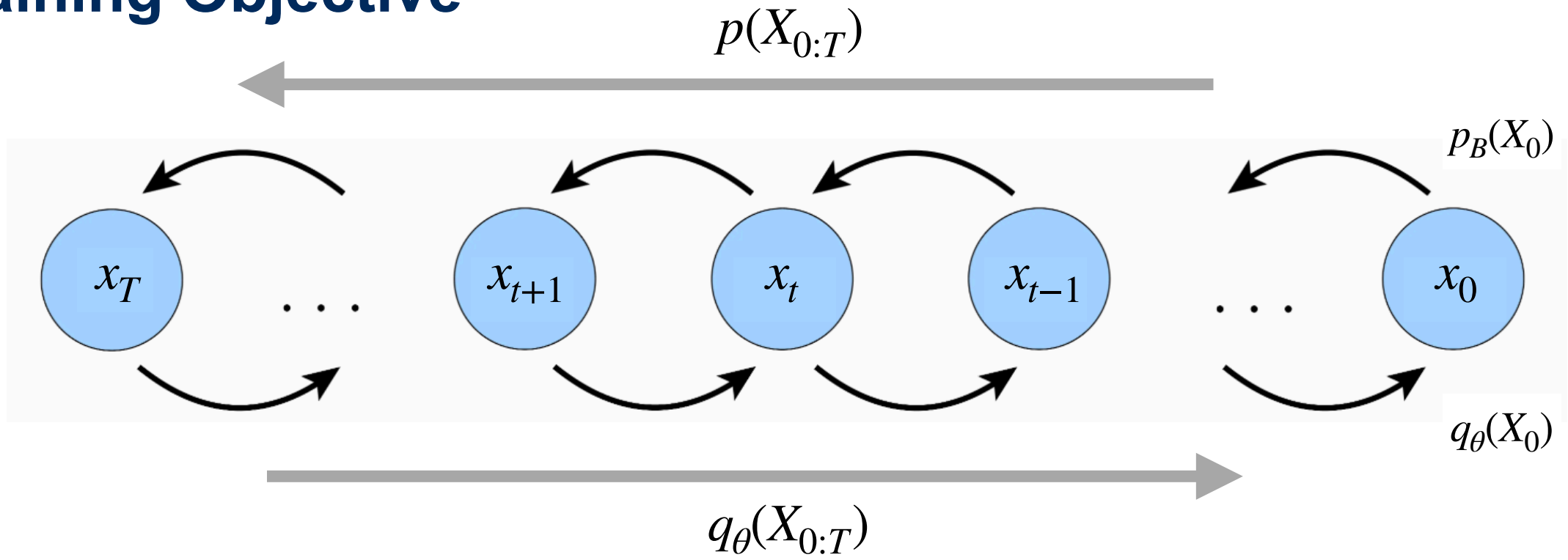
We need the reverse KL

$$\mathrm{KL}\left(q_\theta(X) \,\|\, p_B(X)\right) = \mathbb{E}_{X\sim q_\theta}\left[\log \frac{q_\theta(X)}{p_B(X)}\right]$$

But this is intractable

$$\mathrm{KL}\left(q_\theta(X_0)\|p_B(X_0)\right) \leq \mathrm{KL}\left(q_\theta(X_{0:T})\|p(X_{0:T})\right)$$

Training Objective



$$\mathrm{KL}\left(q_\theta(X_0)\|p_B(X_0)\right) \leq \mathrm{KL}\left(q_\theta(X_{0:T})\|p(X_{0:T})\right) \text{ [9]}$$

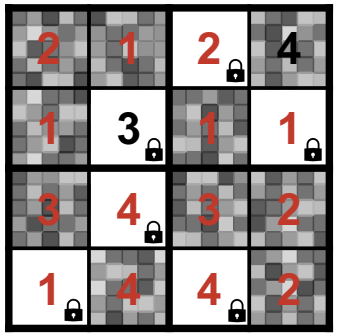
Minimize the KL divergence between the two joints!

$$p(X_{0:T}) = p_B(X_0)\prod_{t=1}^{T} p(X_t \mid X_{t-1}),$$

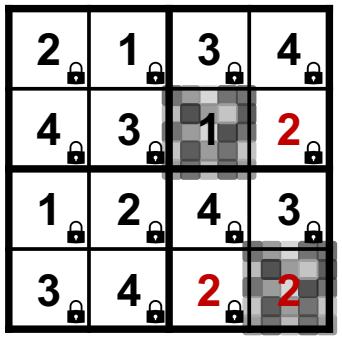
$$q_\theta(X_{0:T}) = q(X_T)\prod_{t=1}^{T} q_\theta(X_{t-1} \mid X_t)$$

Training Objective

Blocked Gibbs Diffusion



Constant Noise to MANY Variables



Constant Noise to FEWER Variables

We perform Blocked Gibbs Diffusion:
Resampling only within the block.

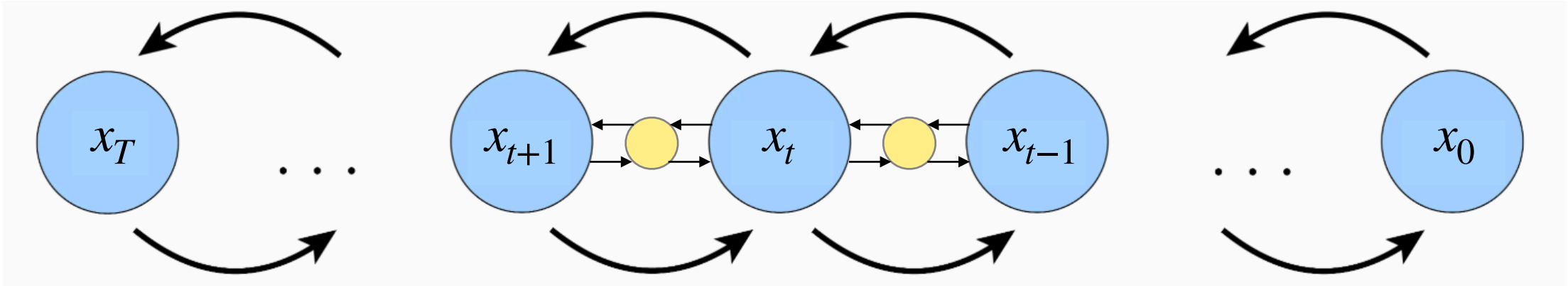
$$q_\theta(X_{t-1} \mid X_t) = \sum_{m_t} q_{\theta,t}(m_t \mid X_t)\, q_\theta(X_{t-1} \mid X_t, m_t)$$

$$p(X_t \mid X_{t-1}) = \sum_{m_t} p_t(m_t \mid X_{t-1})\, p(X_t \mid X_{t-1}, m_t),$$

$$q_\theta(x_{t-1}^{(i)} \mid X_t, m_t) = \begin{cases} \delta(x_{t-1}^{(i)} - x_t^{(i)}) & \text{if } m_t^{(i)} = 0 \\ q_\theta(x_{t-1}^{(i)} \mid X_t) & \text{if } m_t^{(i)} = 1 \end{cases}$$

Training Objective

$p(X_{0:T}, m_{1:T})$



$q_\theta(X_{0:T}, m_{1:T})$

$$\mathrm{KL}\big(q_\theta(X_0)\|p_B(X_0)\big) \le \mathrm{KL}\big(q_\theta(X_{0:T}, m_{1:T})\|p(X_{0:T}, m_{1:T})\big)\,.$$

$$p(X_{0:T}, m_{1:T}) = p_B(X_0)\prod_{t=1}^{T} p_t(m_t \mid X_{t-1})\, p(X_t \mid X_{t-1}, m_t),$$

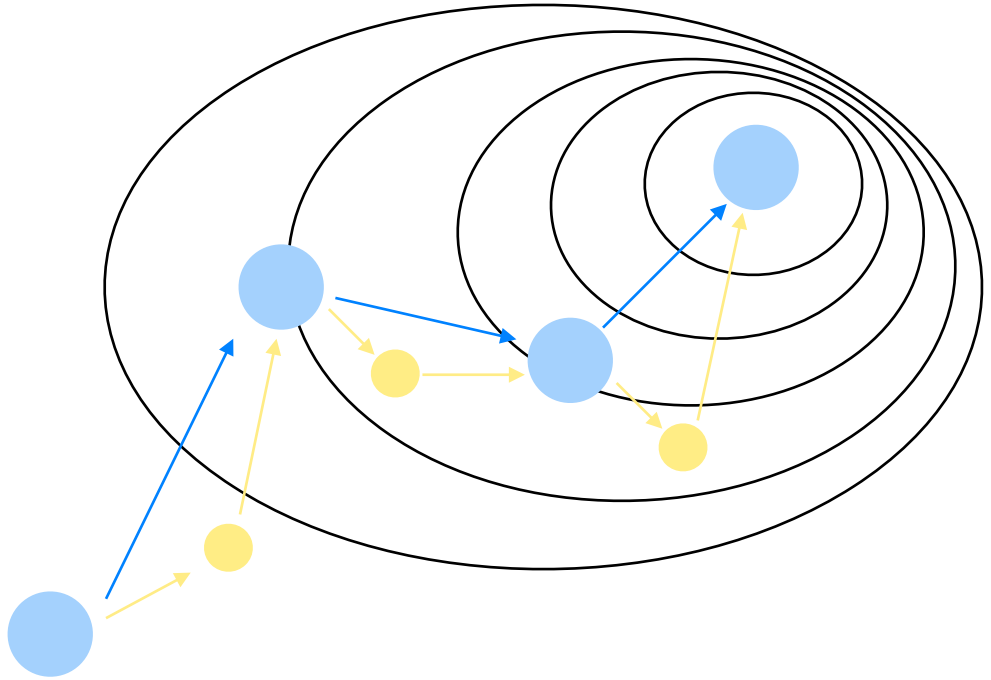
$$q_\theta(X_{0:T}, m_{1:T}) = q(X_T)\prod_{t=1}^{T} q_{\theta,t}(m_t \mid X_t)\, q_\theta(X_{t-1} \mid X_t, m_t)\,.$$

Why is this OK? Augmented MCMC

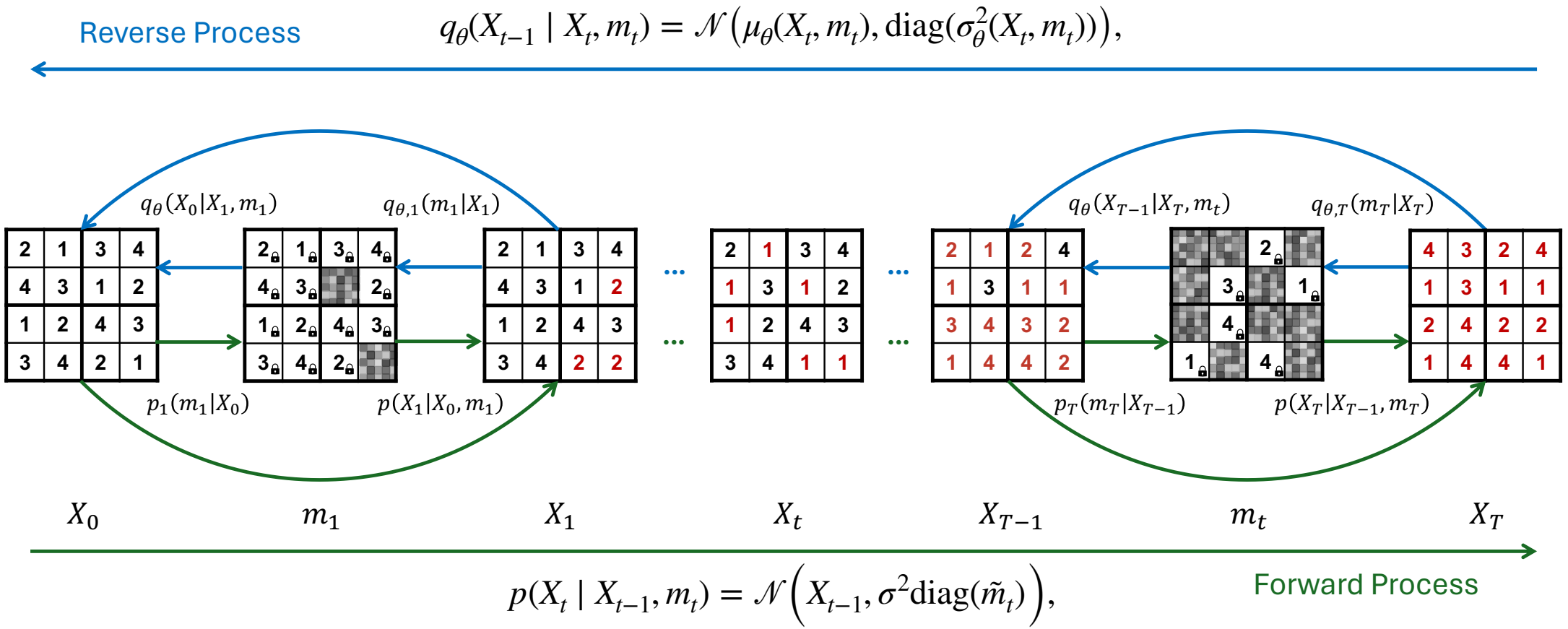
By marginalizing the joint distribution,
we obtain our target data distribution.

We can simply discard the auxiliary variables
from our samples!

$$
\begin{aligned}
&\sum_{\mathbf{m}} q_\theta(X_{0:T}, \mathbf{m}) \\
&= \sum_{m_1,\ldots,m_T} \left[q(X_T) \prod_{t=1}^{T} q_{\theta,t}(m_t \mid X_t) q_\theta(X_{t-1} \mid X_t, m_t) \right] . \\
&= q(X_T) \prod_{t=1}^{T} \left[\sum_{m_t} q_{\theta,t}(m_t \mid X_t) q_\theta(X_{t-1} \mid X_t, m_t) \right] . \\
&= q(X_T) \prod_{t=1}^{T} q_\theta(X_{t-1} \mid X_t) \\
&\equiv q_\theta(X_{0:T}) .
\end{aligned}
$$



Unsupervised Blocked-Gibbs DiT (BlogDiT)



We assume the noise is Gaussian

Simplifying the Objective

$$
\begin{aligned}
\mathrm{KL}\left(q_\theta(X_0)\|p_B(X_0)\right) &\leq \mathrm{KL}\left(q_\theta(X_{0:T}, m_{1:T})\|p(X_{0:T}, m_{1:T})\right) \\
&= \mathbb{E}_{q_\theta}\left[-\log p_B(x_0)\right] \\
&+\mathbb{E}_{q_\theta}\left[\sum_{t=1}^{T} \log q_\theta(X_{t-1} \mid X_t, m_t)\right] \\
&-\mathbb{E}_{q_\theta}\left[\sum_{t=1}^{T} \log p(X_t \mid X_{t-1}, m_t)\right] \\
&+\mathbb{E}_{q_\theta}\left[\sum_{t=1}^{T} \log \frac{q_{\theta,t}(m_t \mid X_t)}{p_t(m_t \mid X_{t-1})}\right] \\
&+C
\end{aligned}
$$

Energy: Ensures Sample Quality

Entropy: Ensures Sample Diversity

Noise Matching: Regulates Step

Mask Matching: Ensures masking consistency

Energy: Ensures Sample Quality

$$E(X) = f(X) + \sum_j \lambda_j \phi_j(X)$$

$$\mathrm{KL}\big(q_\theta(X_0) \| p_B(X_0)\big) \le \mathrm{KL}\big(q_\theta(X_{0:T}, m_{1:T}) \| p(X_{0:T}, m_{1:T})\big)$$

$$= \mathbb{E}_{q_\theta}\left[-\log p_B(x_0)\right]$$

$$+\mathbb{E}_{q_\theta}\left[\sum_{t=1}^{T} \log q_\theta(X_{t-1} \mid X_t, m_t)\right]$$

$$-\mathbb{E}_{q_\theta}\left[\sum_{t=1}^{T} \log p(X_t \mid X_{t-1}, m_t)\right]$$

$$+\mathbb{E}_{q_\theta}\left[\sum_{t=1}^{T} \log \frac{q_{\theta,t}(m_t \mid X_t)}{p_t(m_t \mid X_{t-1})}\right]$$

$$+C$$

$$\longrightarrow = \mathbb{E}_{q_\theta}\left[-\log\left(\frac{\exp\left(-\beta E(X)\right)}{\sum_{X' \in \mathcal{X}} \exp\left(-\beta E(X')\right)}\right)\right]$$

$$\beta \mathbb{E}_{q_\theta}[E(X_0)] + \log Z(\beta)\,.$$

Discrete Constraint $c(x)$	**Continuous Penalty** $\phi_c(x)$
$\textsc{AllDifferent}_{m=n}(x_1, \ldots, x_n)$	$\sum_{j=1}^{m}\left(\left\lvert 1 - \sum_{i=1}^{n} x_i^{(j)}\right\rvert\right)$
$x_i \neq x_k$	$\sum_{j=1}^{m}(x_i^{(j)} \cdot x_k^{(j)})$
$\forall (i,k) \in E:\ x_i^{(1)} + x_k^{(1)} \le 1$	$\sum_{(i,k)\in E}\left(x_i^{(1)} \cdot x_k^{(1)}\right)$

$$\phi_j(X) = 0 \iff c_j(X) = \text{True}$$

Entropy: Ensures Sample Diversity

$$\mathrm{KL}\left(q_\theta(X_0)\|p_B(X_0)\right) \le \mathrm{KL}\left(q_\theta(X_{0:T}, m_{1:T})\|p(X_{0:T}, m_{1:T})\right)$$

$$= \mathbb{E}_{q_\theta}\left[-\log p_B(x_0)\right]$$

$$+\mathbb{E}_{q_\theta}\left[\sum_{t=1}^{T} \log q_\theta(X_{t-1} \mid X_t, m_t)\right] \longrightarrow = \sum_{t=1}^{T} \mathbb{E}_{X_{T:t-1}, m_{T:t} \sim q_\theta}\left[\log q_\theta(X_{t-1} \mid X_t, m_t)\right]$$

$$= \sum_{t=1}^{T} \mathbb{E}_{X_{T:t}, m_{T:t} \sim q_\theta}\left[\mathbb{E}_{X_{t-1}|X_t, m_t} \log q_\theta(X_{t-1} \mid X_t, m_t)\right]$$

$$-\mathbb{E}_{q_\theta}\left[\sum_{t=1}^{T} \log p(X_t \mid X_{t-1}, m_t)\right]$$

$$= -\sum_{t=1}^{T} \mathbb{E}_{X_{T:t}, m_{T:t} \sim q_\theta}\left[H(q_\theta(X_{t-1} \mid X_t, m_t))\right].$$

$$+\mathbb{E}_{q_\theta}\left[\sum_{t=1}^{T} \log \frac{q_{\theta,t}(m_t \mid X_t)}{p_t(m_t \mid X_{t-1})}\right]$$

Gaussian Parametric Noise

$$+C$$

$$\frac{1}{2}\sum_{d=1}^{D} m_{t,d}\left(\log(2\pi e) + \log \sigma^2_{\theta,t,d}\right),$$

Noise Matching: Regulates Step

$$\mathrm{KL}\left(q_\theta(X_0)\|p_B(X_0)\right) \le \mathrm{KL}\left(q_\theta(X_{0:T}, m_{1:T})\|p(X_{0:T}, m_{1:T})\right)$$

$$= \mathbb{E}_{q_\theta}\left[-\log p_B(x_0)\right]$$

$$+\mathbb{E}_{q_\theta}\left[\sum_{t=1}^{T} \log q_\theta(X_{t-1} \mid X_t, m_t)\right]$$

$$-\mathbb{E}_{q_\theta}\left[\sum_{t=1}^{T} \log p(X_t \mid X_{t-1}, m_t)\right]$$

$$+\mathbb{E}_{q_\theta}\left[\sum_{t=1}^{T} \log \frac{q_{\theta,t}(m_t \mid X_t)}{p_t(m_t \mid X_{t-1})}\right]$$

$$+C$$

$$= \sum_{t=1}^{T} \mathbb{E}_{X_{T:t-1}, m_{T:t} \sim q_\theta}\left[\log p(X_t \mid X_{t-1}, m_t)\right].$$

$$= \sum_{t=1}^{T} \mathbb{E}_{X_{T:t}, m_{T:t} \sim q_\theta}\left[\mathbb{E}_{X_{t-1}|X_t, m_t} \log p(X_t \mid X_{t-1}, m_t)\right].$$

Gaussian Parametric Noise

$$\frac{1}{2\sigma^2}\left(\|m_t \odot (X_t - \mu_{\theta,t})\|_2^2 + \mathrm{tr}(\mathrm{diag}(m_t)\Sigma_{\theta,t})\right) + \text{const},$$

Mask Matching: Ensures masking consistency

$$\mathrm{KL}\left(q_\theta(X_0)\|p_B(X_0)\right) \le \mathrm{KL}\left(q_\theta(X_{0:T}, m_{1:T})\|p(X_{0:T}, m_{1:T})\right)$$

$$= \mathbb{E}_{q_\theta}\left[-\log p_B(x_0)\right]$$

$$+\mathbb{E}_{q_\theta}\left[\sum_{t=1}^{T} \log q_\theta(X_{t-1} \mid X_t, m_t)\right]$$

$$-\mathbb{E}_{q_\theta}\left[\sum_{t=1}^{T} \log p(X_t \mid X_{t-1}, m_t)\right]$$

$$+\mathbb{E}_{q_\theta}\left[\sum_{t=1}^{T} \log \frac{q_{\theta,t}(m_t \mid X_t)}{p_t(m_t \mid X_{t-1})}\right] \longrightarrow 0$$

$$q_{\theta,t}(m_t \mid X_t) \equiv p_t(m_t \mid X_{t-1}) \equiv \pi_t(m_t)$$
$$= \prod_{i=1}^{d} \mathrm{Bernoulli}(m_{i,t}; \rho_t),$$

$$+C$$

Adaptive Mask Sampling

1	7	1	5	8	3	3	2	4
6	5	9	2	1	7	3	4	8
3	4	9	4	6	8	2	1	7
7	9	5	1	3	5	6	3	2
1	1	3	6	9	7	1	7	2
6	8	6	3	3	2	7	3	9
9	1	4	8	3	5	1	7	6
9	3	1	7	5	1	4	9	5
5	6	7	4	2	9	7	1	3

Can we do better than randomly sampling the mask?

How do we select neighbourhoods in LNS?

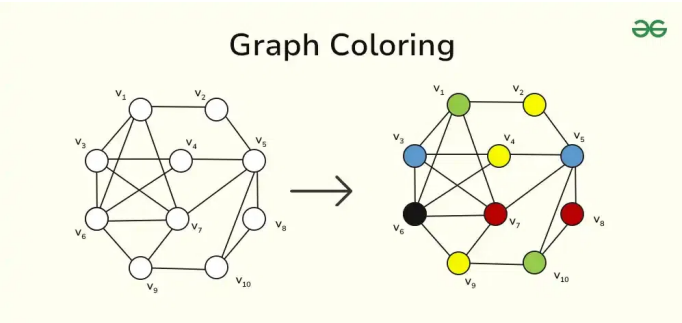
Most Critical → Prefer the variables that violates the most constraints.

Model Confidence → Prefer the variables the model is least confident in.

Experiments

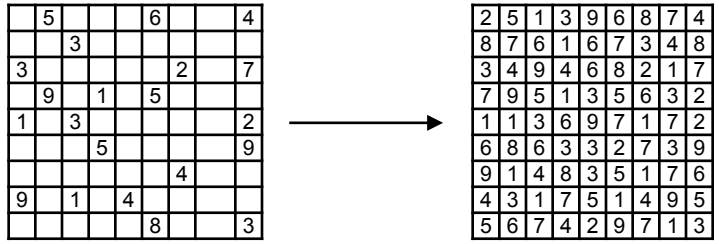
Graph Coloring

Method	Graph-Coloring-5		Graph-Coloring-10	
	$n = 50$	$n = 100$	$n = 100$	$n = 200$
OR-Tools*	**83.08**	**57.16**	52.41	10.25
ANYCSP*	79.17	34.83	0.00	0.00
ConsFormer*	81.00	47.33	<u>52.60</u>	11.92
DiT	79.75	48.91	50.50	14.25
BloGDiT	82.58	52.93	**53.92**	<u>17.83</u>
BloGDiT+AS	<u>82.93</u>	<u>54.91</u>	**53.92**	**18.67**



Sudoku

Method	In-distribution Test	OOD Test
OR-Tools	**100**	**100**
SATNet*	98.3	3.2
RRN*	<u>99.8</u>	28.6
Recurrent Transformer*	**100**	32.9
IRED*	99.4	62.1
ConsFormer	**100**	85.8
DiT	**100**	48.6
BloGDiT	**100**	87.5
BloGDiT+AS	**100**	<u>93.0</u>



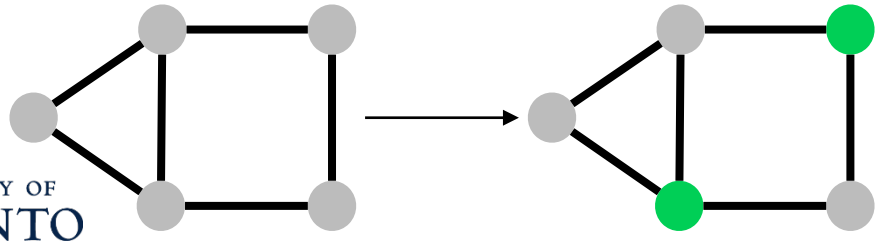
Experiments

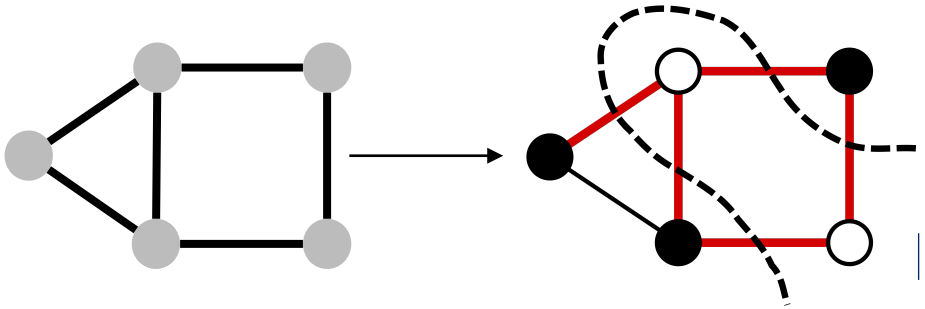
Maximum Independent Set

Method	RB-small	RB-large
OR-Tools	**20.10**	**42.66**
LwtD*	19.01	32.32
INTEL*	18.47	34.47
DGL*	17.36	34.50
LTFT*	19.18	37.48
DiffUCO*	18.88	38.10
SDDS*	<u>19.62</u>	<u>39.99</u>
DiT	8.05	0.29
BloGDiT	8.68	0.04
BloGDiT+AS	18.87	35.13

Max Cut

Method	$\|V\|$=800	$\|V\|$=1K	$\|V\|$=2K	$|V|\geq$3K
OR-Tools*	143.89	112.78	365.89	378.62
Greedy*	411.44	359.11	737.00	774.25
SDP*	245.44	229.22	-	-
RUNCSP*	185.89	156.56	357.33	401.00
ECO-DQN*	65.11	54.67	157.00	428.25
ECORD*	8.67	8.78	39.22	187.75
ANYCSP*	<u>1.22</u>	<u>2.44</u>	<u>13.11</u>	<u>51.63</u>
DiffUCO*	4.11	6.33	31.67	116.75
ConsFormer*	16.33	12.44	52.11	115.25
DiT	56.11	50.0	101.33	162.0
BloGDiT	2.11	3.22	18.89	88.63
BloGDiT+AS	**0.11**	**1.33**	**7.88**	**32.75**





Conclusion

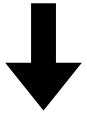
We utilize **Diffusion models** as neural heuristic solvers for **Constraint Optimization Problems.**

Existing methods are limited by the **GNN architecture** and the **supervised learning approach**.

We introduced a **Blocked Gibbs Diffusion Transformer** trained by **Unsupervised Learning.**

Limitations

We are currently doing “variational” Gibbs, since we are learning to do Gibbs proposals.



Future work:
Add Metropolis-Hastings correctors to ensure sample quality.

References

[1] Khalil, Elias., “Course Overview MIE1666: Machine Learning for Mathematical Optimization” [Lecture Slides], University of Toronto (2024)
[2] https://developers.google.com/optimization/routing/tsp
[3] Feng, Shengyu, and Yiming Yang. "Regularized Langevin Dynamics for Combinatorial Optimization." *Forty-second International Conference on Machine Learning*. (2025)
[4] Sun, Haoran, Katayoon Goshvadi, Azade Nova, Dale Schuurmans, and Hanjun Dai. "Revisiting sampling for combinatorial optimization." In *International Conference on Machine Learning*, pp. 32859-32874. PMLR, 2023.
[5] Gundersen, G. (2021, April 16). *The ELBO in variational inference*. Gregory Gundersen. https://gregorygundersen.com/blog/2021/04/16/variational-inference/
[6] He, Kaiming., “Diffusion Models” [Lecture Slides], MIT (2024). https://mit-6s978.github.io/assets/pdfs/lec5_diffusion.pdf
[7] Luo, Calvin. "Understanding diffusion models: A unified perspective." *arXiv preprint arXiv:2208.11970* (2022).
[8] Sun, Zhiqing, and Yiming Yang. "Difusco: Graph-based diffusion solvers for combinatorial optimization." *Advances in neural information processing systems* 36 (2023): 3706-3731.
[9] Sanokowski, Sebastian, Sepp Hochreiter, and Sebastian Lehner. "A Diffusion Model Framework for Unsupervised Neural Combinatorial Optimization." *International Conference on Machine Learning*. PMLR, 2024.