

# Neural Network Solvers for Combinatorial Optimization

## Graph 10. Non-autoregressive CO Solvers

1

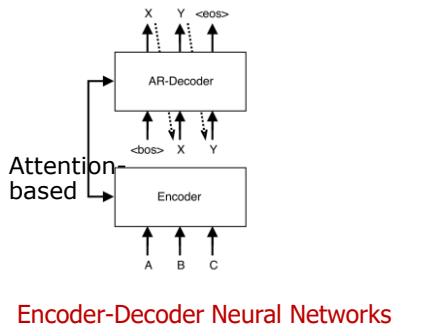
### Outline of 3 Lectures: Graph 9, 10 and 11

- Introduction to ML for Combinatorial Optimization (CO)
- Autoregressive (AR) CO Solvers
- Non-autoregressive (NAR) CO Solvers
  - Reinforcement learning with DIMES [R Qiu\*, Z Sun\* & Y Yang, **NearIPS 2022**]
  - Supervised learning with DIFUSCO [Z Sun & Y Yang, **NearIPS 2023**]
- Pre-trained Large Language Models [C Yang et al., **ICLR 2024**]

2

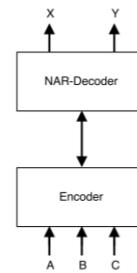
## Decoding Strategies

### Autoregressive (AR) Solvers



Encoder-Decoder Neural Networks

### Non-autoregressive (NAR) Solvers



Encoder + Active Search Algorithms

## DIMES: A differentiable meta solver for CO [Qiu\*, Sun\* & Yang, NearIPS 2022]

### Key Ideas

- 1) A unified framework for CO problems (including TSP and MIS)
- 2) A **NAR** CO solver with an **encoder-only architecture** (removing the expensive decoding parts in the AR solvers), scaled to graphs with  $n = 10,000$  nodes
- 3) Using **meta learning** (in the training-phase) to enhance the accuracy on new graphs

# Unified Framework for CO Problems

- Convert any CO problem into a task of  $\{0,1\}$ -vector generation
  - Let  $\mathcal{F}_s = \{0,1\}^N$  be the space of **feasible solutions** of instance graph  $s$ ;
  - Let  $c_s: \mathcal{F}_s \rightarrow \mathbb{R}$  be the **cost function** for any solution  $f = (f_1, f_2, \dots, f_N) \in \mathcal{F}_s$ .

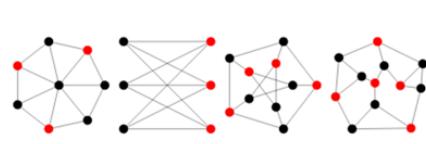
**TSP:** Finding optimal tour  $f \in \{0,1\}^N$  by selecting  $n$  edges from the total of  $N = n^2$  edges

$$c_s^{\text{TSP}}(f) = \sum_{i=1}^N f_i \cdot \text{length}(f_i, s)$$



**MIS:** Finding optimal  $f \in \{0,1\}^N$  by selecting a subset of the nodes from the total of  $N = n$  nodes

$$c_s^{\text{MIS}}(f) = \sum_{i=1}^N (1 - f_i)$$



4/2/2024

@Yiming Yang, 11-741 S24 Graph10 NAR CO Solvers

5

5

## Objective

- We want to minimize the **expected cost of system-generated solutions** for graph  $s$

$$\min_{\theta} \mathcal{L}_{\theta}(s) = \min_{\theta} \sum_{\pi \sim p_{\theta}(\cdot|s)} [c_s(\pi) \ p_{\theta}(\pi|s)] \quad (\text{previous lecture})$$

- Optimizing  $\theta$  via reinforcement learning (RL) with

$$\nabla \mathcal{L}_{\theta}(s) = \sum_{\pi \sim p_{\theta}(\cdot|s)} c_s(\pi) \nabla \log p_{\theta}(\pi|s) \quad (\text{previous lecture})$$

- In AR solvers,  $p_{\theta}(\cdot|s)$  is produced by the decoder. Now, how can we proceed without a decoder?

- Answer:



- Using an **encoder-only** neural network to learn a **heatmap** (of node/edge weights) based on a training set of **<tour, cost> pairs**.
  - Using the heatmap to conduct greedy/active search for plausible solutions during testing.

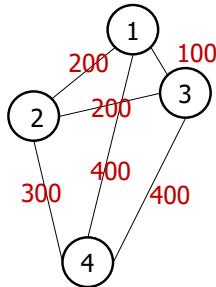
4/2/2024

@Yiming Yang, 11-741 S24 Graph10 NAR CO Solvers

6

6

## A toy example

Graph  $s$ Feasible Solutions & Costs ( $n! = 4 \cdot 3 \cdot 2 \cdot 1 = 24$ )

$$\pi_1 = 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 1, \quad c(\pi_1) = 200 + 200 + 400 + 400 = 1200$$

$$\pi_2 = 1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 1, \quad c(\pi_2) = 200 + 300 + 400 + 100 = 1000$$

$$\pi_3 = 2 \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow 2, \quad c(\pi_3) = 200 + 100 + 400 + 300 = 1000$$

$\{(\pi_i c_i)\} \xrightarrow{\text{intuitively}} \text{Is } e_{31} \text{ better than } e_{41}?$

$\{(\pi_i c_i)\} \xrightarrow{\text{RL training}} \text{Heatmap} = \{\text{edge weights}\} \text{ (larger is better)}$

## Toy Example: NAR Sequence Generation in Testing Phase

Heatmap  $\Theta$ 

		1	2	3	4
1	9	10	1		
2	10	5	6		
3	7	1	8		
4	8	1	1		

$P_\theta(e_{ij} \in \pi^* | s) \propto \Theta_{ij}$

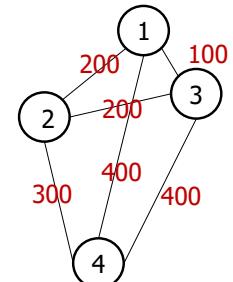
Sampled Tours based on  $\Theta$ 

$$\pi_1: e_{13} \rightarrow e_{34} \rightarrow e_{42} \rightarrow e_{21}$$

$$c(\pi_1) = 100 + 400 + 300 + 200 = 1000$$

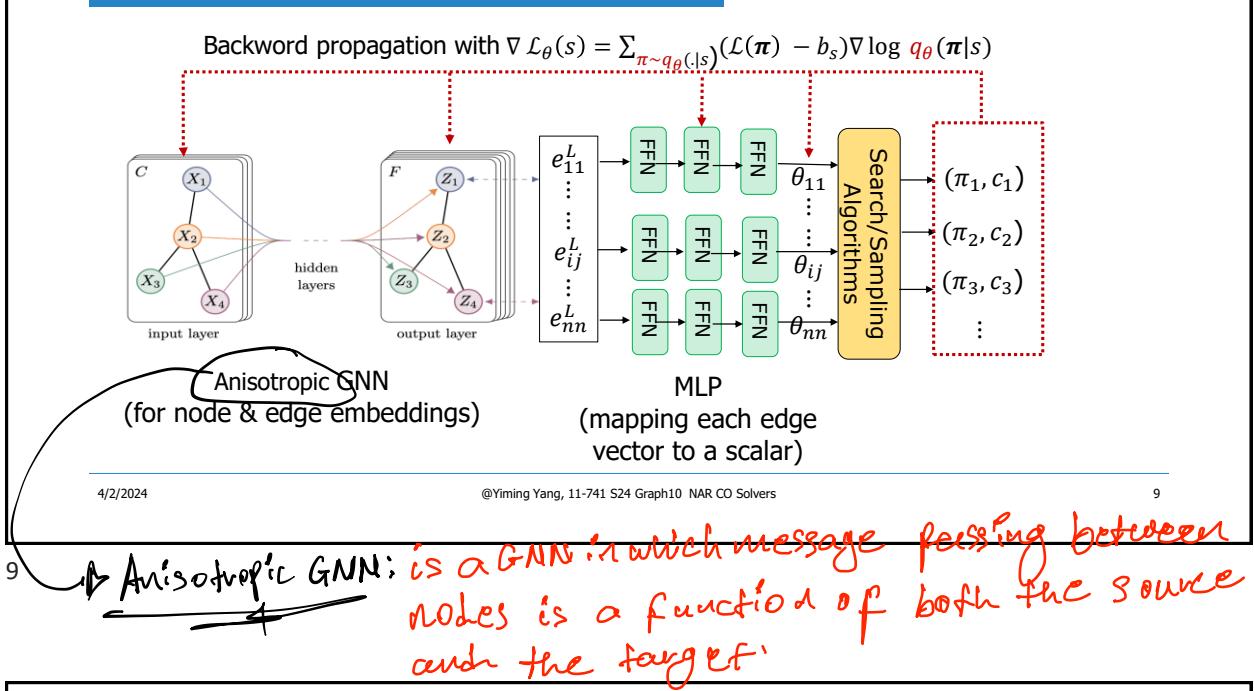
$$\pi_2: e_{21} \rightarrow e_{13} \rightarrow e_{34} \rightarrow e_{42}$$

$$c(\pi_2) = 200 + 100 + 400 + 300 = 1000$$



## How do we obtain the heatmap?

GNN+MLP with RL over a training set of graphs and sampled tours per graph



## Anisotropic Graph Neural Networks

producing edge embedding for solving TSP (Z Sun & Y Yang, NeurIPS 2022)

**Anisotropic Graph Neural Networks** We follow Joshi et al. [33] on the choice of neural architectures. The backbone of the graph neural network is an anisotropic GNN with an edge gating mechanism [9]. Let  $h_i^\ell$  and  $e_{ij}^\ell$  denote the node and edge features at layer  $\ell$  associated with node  $i$  and edge  $ij$ , respectively. The features at the next layer is propagated with an anisotropic message passing scheme:

$$h_i^{\ell+1} = h_i^\ell + \alpha(\text{BN}(\mathbf{U}^\ell h_i^\ell + \mathcal{A}_{j \in \mathcal{N}_i}(\sigma(e_{ij}^\ell) \odot \mathbf{V}^\ell h_j^\ell))), \quad (18)$$

$$e_{ij}^{\ell+1} = e_{ij}^\ell + \alpha(\text{BN}(\mathbf{P}^\ell e_{ij}^\ell + \mathbf{Q}^\ell h_i^\ell + \mathbf{R}^\ell h_j^\ell)). \quad (19)$$

where  $\mathbf{U}^\ell, \mathbf{V}^\ell, \mathbf{P}^\ell, \mathbf{Q}^\ell, \mathbf{R}^\ell \in \mathbb{R}^{d \times d}$  are the learnable parameters of layer  $\ell$ ,  $\alpha$  denotes the activation function (we use SiLU [15] in this paper), BN denotes the Batch Normalization operator [30],  $\mathcal{A}$  denotes the aggregation function (we use mean pooling in this paper),  $\sigma$  is the sigmoid function,  $\odot$  is the Hadamard product, and  $\mathcal{N}_i$  denotes the outlinks (neighborhood) of node  $i$ . We use a 12-layer GNN with width 32.

# Model Training

## ▪ Loss Function

$$\mathcal{L}_\Phi(\mathcal{C}) = \sum_{s \in \mathcal{C}} l_q(\theta_s) = \sum_{s \in \mathcal{C}} l_q[F_\phi(X_s, A_s)]$$

where  $\mathcal{C}$  is the training set of graphs and  $s \in \mathcal{C}$  is an instance graph;

$\Phi$  is the neural network (GNN+MLP) parameter set;

  $\theta_s := F_\phi(X_s, A_s)$  is the heatmap produced by the neural network given input graph  $s$ ;

$X_s$  is the matrix of node features and  $A_s$  is the adjacency matrix of graph  $s$ ;

  $q = q(\theta_s)$  is the auxiliary distribution for efficient sampling of solutions from heatmap  $\theta_s$ .

## ▪ Gradient

$$\nabla_\Phi \mathcal{L}_\Phi(\mathcal{C}) = E_{s \in \mathcal{C}} [\nabla_\Phi \theta_s \cdot \nabla_{\theta_s} l_q(\theta_s)] = E_{s \in \mathcal{C}} [\nabla_\Phi F_\phi(X_s, A_s) \cdot \nabla_{\theta_s} l_q(\theta_s)]$$

## ▪ Forward/backward propagations:

updating  $\Phi$  with the full formula and  $\theta$  with the green part.

# Results

Method	Type	TSP-500			TSP-1000			TSP-10000		
		Length ↓	Drop ↓	Time ↓	Length ↓	Drop ↓	Time ↓	Length ↓	Drop ↓	Time ↓
Concorde	OR (exact)	16.55*	—	37.66m	23.12*	—	6.65h	N/A	N/A	N/A
Gurobi	OR (exact)	16.55	0.00%	45.63h	N/A	N/A	N/A	N/A	N/A	N/A
LKH-3 (default)	OR	16.55	0.00%	46.28m	23.12	0.00%	2.57h	71.77*	—	8.8h
LKH-3 (less trails)	OR	16.55	0.00%	3.03m	23.12	0.00%	7.73m	71.79	—	51.27m
Nearest Insertion	OR	20.62	24.59%	0s	28.96	25.26%	0s	90.51	26.11%	6s
Random Insertion	OR	18.57	12.21%	0s	26.12	12.98%	0s	81.85	14.04%	4s
Farthest Insertion	OR	18.30	10.57%	0s	25.72	11.25%	0s	80.59	12.29%	6s
EAN	RL+S	28.63	73.03%	20.18m	50.30	117.59%	37.07m	N/A	N/A	N/A
EAN	RL+S+2-OPT	23.75	43.57%	57.76m	47.73	106.46%	5.39h	N/A	N/A	N/A
AM	RL+S	22.64	36.84%	15.64m	42.80	85.15%	63.97m	431.58	501.27%	12.63m
AM	RL+G	20.02	20.99%	1.51m	31.15	34.75%	3.18m	141.68	97.39%	5.99m
AM	RL+BS	19.53	18.03%	21.99m	29.90	29.23%	1.64h	129.40	80.28%	1.81h
GCN	SL+G	29.72	79.61%	6.67m	48.62	110.29%	28.52m	N/A	N/A	N/A
GCN	SL+BS	30.37	83.55%	38.02m	51.26	121.73%	51.67m	N/A	N/A	N/A
POMO+EAS-Emb	RL+AS	19.24	16.25%	12.80h	N/A	N/A	N/A	N/A	N/A	N/A
POMO+EAS-Lay	RL+AS	19.35	16.92%	16.19h	N/A	N/A	N/A	N/A	N/A	N/A
POMO+EAS-Tab	RL+AS	24.54	48.22%	11.61h	49.56	114.36%	63.45h	N/A	N/A	N/A
Att-GCN	SL+MCTS	16.97	2.54%	2.20m	23.86	3.22%	4.10m	74.93	4.39%	21.49m
DIMES (ours)	RL+G	18.93	14.38%	0.97m	26.58	14.97%	2.08m	86.44	20.44%	4.65m
DIMES (ours)	RL+AS+G	17.81	7.61%	2.10h	24.91	7.74%	4.49h	80.45	12.09%	3.07h
DIMES (ours)	RL+S	18.84	13.84%	1.06m	26.36	14.01%	2.38m	85.75	19.48%	4.80m
DIMES (ours)	RL+AS+S	17.80	7.55%	2.11h	24.89	7.70%	4.53h	80.42	12.05%	3.12h
DIMES (ours)	RL+MCTS	16.87	1.93%	2.92m	23.73	2.64%	6.87m	74.63	3.98%	29.83m
DIMES (ours)	RL+AS+MCTS	16.84	1.76%	2.15h	23.69	2.46%	4.62h	74.06	3.19%	3.57h

OR: Traditional Solver

RL: Reinforce. Learners

SL: Supervised Learners

[Sampling Strategies:](#)

G: Greedy Readout

S: Sampling

BS: Beam Search

AS: Active Search

MCTS: Monte-Carlo Tree Search

## Outline of 3 Lectures: Graph 9, 10 and 11

- Introduction to ML for Combinatorial Optimization (CO)
- Autoregressive (AR) CO Solvers
- Non-autoregressive (NAR) CO Solvers
  - Reinforcement learning with DIMES [R Qiu\*, Z Sun\* & Y Yang, **NearIPS 2022**]
  - Supervised learning with DIFUSCO [Z Sun & Y Yang, **NearIPS 2023**]
- Pre-trained Large Language Models [C Yang et al., ICLR 2024]

4/2/2024

@Yiming Yang, 11-741 S24 Graph10 NAR CO Solvers

17

17

## DIFUSCO: Graph-based Diffusion Solvers for CO [Z Sun & Y Yang, NeurIPS 2023]

- Borrowing the key idea from recent diffusion models for computer vision
- Developed the 1<sup>st</sup> diffusion-based solver for CO problems
- Beat other SOTA CO solvers (e.g., DIMES) in both accuracy and efficiency



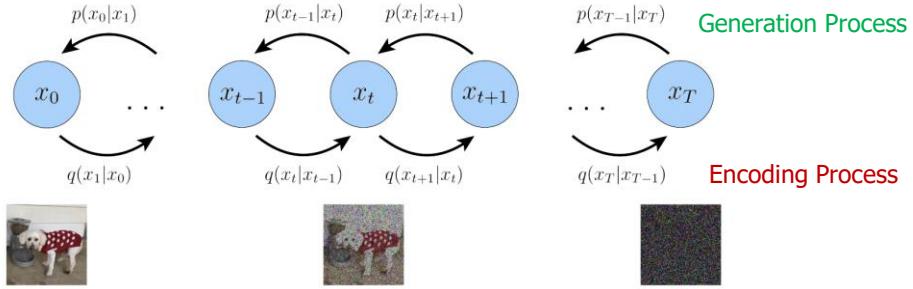
4/2/2024

@Yiming Yang, 11-741 S24 Graph10 NAR CO Solvers

18

18

## Diffusion modeling for generating high-quality images



Rationale: Decomposing a hard problem ( $x_T \rightarrow x_0$ ) into many easier problems ( $x_t \rightarrow x_{t-1}$ )

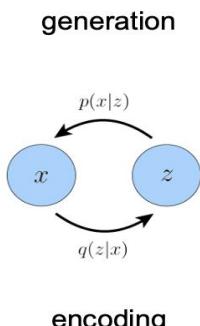
4/2/2024

@Yiming Yang, 11-741 S24 Graph10 NAR CO Solvers

19

19

## Recap: Variational Autoencoder (VAE)



### Evidence Lower Bound (ELBO)

$$\begin{aligned}
 \log p(x) &= \log \int p(x, z) dz && z \sim \mathcal{N}(\cdot) \\
 &= \log \int \frac{p(x, z)q(z|x)}{q(z|x)} dz && (\text{Multiply by } \frac{q(z|x)}{q(z|x)} = 1) \\
 &= \log \mathbb{E}_{q(z|x)} \left[ \frac{p(x, z)}{q(z|x)} \right] && (\text{Definition of Expectation}) \\
 &\geq \mathbb{E}_{q(z|x)} \left[ \log \frac{p(x, z)}{q(z|x)} \right] && (\text{Apply Jensen's Inequality})
 \end{aligned}$$

Maximization of  $\log p(x)$  can be reduced to the problem of maximizing its ELBO.

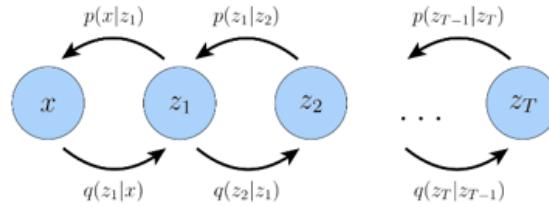
4/2/2024

@Yiming Yang, 11-741 S24 Graph10 NAR CO Solvers

20

20

## Diffusion Model as Markovian Hierarchical VAE (MHVAE)



Generation:  $p(\mathbf{x}, \mathbf{z}_{1:T}) = p(\mathbf{z}_T)p_{\theta}(\mathbf{x}|z_1) \prod_{t=2}^T p_{\theta}(z_{t-1}|z_t)$

Encoding:  $q_{\phi}(\mathbf{z}_{1:T}|\mathbf{x}) = q_{\phi}(\mathbf{z}_1|\mathbf{x}) \prod_{t=2}^T q_{\phi}(\mathbf{z}_t|\mathbf{z}_{t-1})$

4/2/2024

@Yiming Yang, 11-741 S24 Graph10 NAR CO Solvers

21

21

## ELBO of Diffusion Model (replacing $z$ by $z_{1:T}$ )

$$\begin{aligned}
 \log p(\mathbf{x}) &= \log \int p(\mathbf{x}, \mathbf{z}_{1:T}) d\mathbf{z}_{1:T} \\
 &= \log \int \frac{p(\mathbf{x}, \mathbf{z}_{1:T}) q(\mathbf{z}_{1:T}|\mathbf{x})}{q(\mathbf{z}_{1:T}|\mathbf{x})} d\mathbf{z}_{1:T} && \text{(Multiply by } \frac{q(\mathbf{z}|\mathbf{x})}{q(\mathbf{z}|\mathbf{x})} = 1\text{)} \\
 &= \log \mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x})} \left[ \frac{p(\mathbf{x}, \mathbf{z}_{1:T})}{q(\mathbf{z}_{1:T}|\mathbf{x})} \right] && \text{(Definition of Expectation)} \\
 &\geq \mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{x})} \left[ \log \frac{p(\mathbf{x}, \mathbf{z}_{1:T})}{q(\mathbf{z}_{1:T}|\mathbf{x})} \right] && \text{(Apply Jensen's Inequality)}
 \end{aligned}$$

From now on, we change notation as  $x \rightarrow \mathbf{x}_0$ ,  $z_i \rightarrow \mathbf{x}_i$  and  $p(x, z_{1:T}) \rightarrow p_{\theta}(\mathbf{x}_{0:T})$ .

4/2/2024

@Yiming Yang, 11-741 S24 Graph10 NAR CO Solvers

22

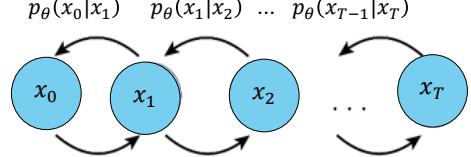
22

## ELBO of Diffusion Model (notation changed)

$$\begin{aligned}
 \log p(\mathbf{x}) &= \log \int p(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T} \\
 &= \log \int \frac{p(\mathbf{x}_{0:T}) q(\mathbf{x}_{1:T} | \mathbf{x}_0)}{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} d\mathbf{x}_{1:T} \\
 &= \log \mathbb{E}_{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \left[ \frac{p(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \right] \\
 &\geq \mathbb{E}_{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \left[ \log \frac{p(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \right]
 \end{aligned}$$

ELBO

## ELBO of Diffusion Model [Calvin Luo, 2022]



$$\begin{aligned}
 \log p(\mathbf{x}) &\geq \mathbb{E}_{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \left[ \log \frac{p(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \right] \\
 &= \underbrace{\mathbb{E}_{q(\mathbf{x}_1 | \mathbf{x}_0)} [\log p_{\theta}(\mathbf{x}_0 | \mathbf{x}_1)]}_{\text{reconstruction term}} - \underbrace{\mathcal{D}_{\text{KL}}(q(\mathbf{x}_T | \mathbf{x}_0) || p(\mathbf{x}_T))}_{\text{prior matching term}} \\
 &\quad - \sum_{t=2}^T \underbrace{\mathbb{E}_{q(\mathbf{x}_t | \mathbf{x}_0)} [\mathcal{D}_{\text{KL}}(q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) || p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t))]}_{\text{denoising matching term}}
 \end{aligned}$$

(Proof skipped: see the Appendix and the tutorial by Clavin Luo for details)

## DIFUSCO as a generic CO Solver

- We define a CO problem (e.g., TSP or MIS) as the task of generating discrete  $\{0,1\}$ -vector  $x \in \{0,1\}^{N(s)}$ , which indicates a sequence (or set) of selected edges or nodes from an input graph, and  $N(s)$  is the number of nodes or edges in the input graph.

- We train a neural network to minimize

$$L(\theta) = \mathbb{E}_{s \in S}[-\log p_\theta(x_s^*)]$$

where  $x_s^*$  is an optimal solution for graph  $s$ ;  $p_\theta(x_s)$  is the system-estimated probability of any  $x \in \{0,1\}^{N(s)}$  to be an optimal solution in this graph.

- We obtain a labeled training set for supervised learning of the model, with an existing heuristic (non-ML) solver that finds an (near-)optimal solution for each instance graph.

25

25

## Forward Encoding (Discrete Model) (a pre-designed procedure, no learning)

- Manually specify hyperparameters  $(\beta_1, \beta_2, \dots, \beta_T)$ , namely the schedule, which contains the step-wise corruption ratios.
- Starting with clean data  $x_0$ , inject noise in step  $t = 1, \dots, T$  with for each element of  $x_t$  as

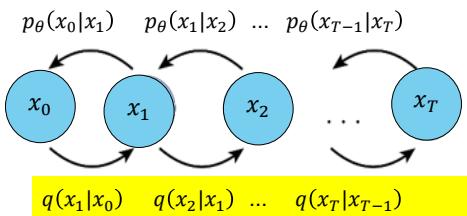
$$q(x_t^{(i)} | x_{t-1}^{(i)}) = \text{Cat}\left(x_t^{(i)} | p_t^{(i)} = \tilde{x}_{t-1}^{(i)} \bar{Q}_t\right)$$

- Multi-step acceleration (e.g., randomly sample 50 time stamps out of  $1, \dots, T=1000$ )

$$q(x_t^{(i)} | x_{t-1}^{(i)}, x_0^{(i)}) = \text{Cat}\left(x_t^{(i)} | p_t^{(i)} = \tilde{x}_0^{(i)} \bar{Q}_t\right)$$

$$\bar{Q}_t \triangleq Q_1 Q_2 \dots Q_t$$

What is this operator?



Efficient computation of  $x_t$  given  $x_0$   
(with closed-form instead of MCMC)

26

## Forward Encoding Details (Discrete Model)

- Each step, the element-wise corruption is defined as

$$q(x_t^{(i)} | x_{t-1}^{(i)}) = \text{Cat}(x_t^{(i)} | p_t^{(i)} = \tilde{x}_{t-1}^{(i)} Q_t)$$

- Applying it recursively yields

$$\begin{aligned} q(x_t^{(i)} | x_{t-1}^{(i)}, x_0^{(i)}) &= \text{Cat}(x_t^{(i)} | p_t^{(i)} = \tilde{x}_{t-1}^{(i)} Q_{t-1}) \\ &= \text{Cat}(x_t^{(i)} | p_t^{(i)} = \tilde{x}_{t-2}^{(i)} Q_{t-2} Q_{t-1}) \\ &= \text{Cat}(x_t^{(i)} | p_t^{(i)} = \tilde{x}_{t-3}^{(i)} Q_{t-3} Q_{t-2} Q_{t-1}) \\ &\vdots \\ &= \text{Cat}(x_t^{(i)} | p_t^{(i)} = \tilde{x}_0^{(i)} Q_1 Q_2 \dots Q_t) \\ &= \text{Cat}(x_t^{(i)} | p_t^{(i)} = \tilde{x}_0^{(i)} \bar{Q}_t) \end{aligned}$$

$$\bar{Q}_t \triangleq Q_1 Q_2 \dots Q_t$$

vector matrix  $x$

$$p_t^{(i)} = \frac{\tilde{x}_{t-1}^{(i)}}{(x_{t-1}^{(i)} - 1 - x_{t-1}^{(i)})} \left( \begin{array}{cc} 1 - \beta_t & \beta_t \\ \beta_t & 1 - \beta_t \end{array} \right)$$

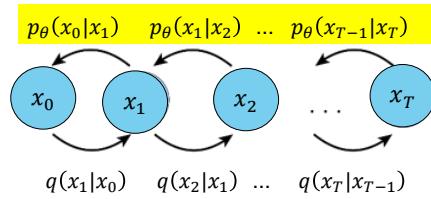
$$= \begin{cases} (1 - \beta_t, \beta_t) & \text{if } x_{t-1}^{(i)} = 1 \\ (\beta_t, 1 - \beta_t) & \text{if } x_{t-1}^{(i)} = 0 \end{cases}$$

## Backward Denoising (Generation) (supervised learning with a graph neural network)

- Start with a fully noisy  $x_T$  whose elements  $x_T^{(i)} \sim \text{Ber}(0.5)$  are sampled independently.
- For  $t = T, \dots, 1$ , calculate

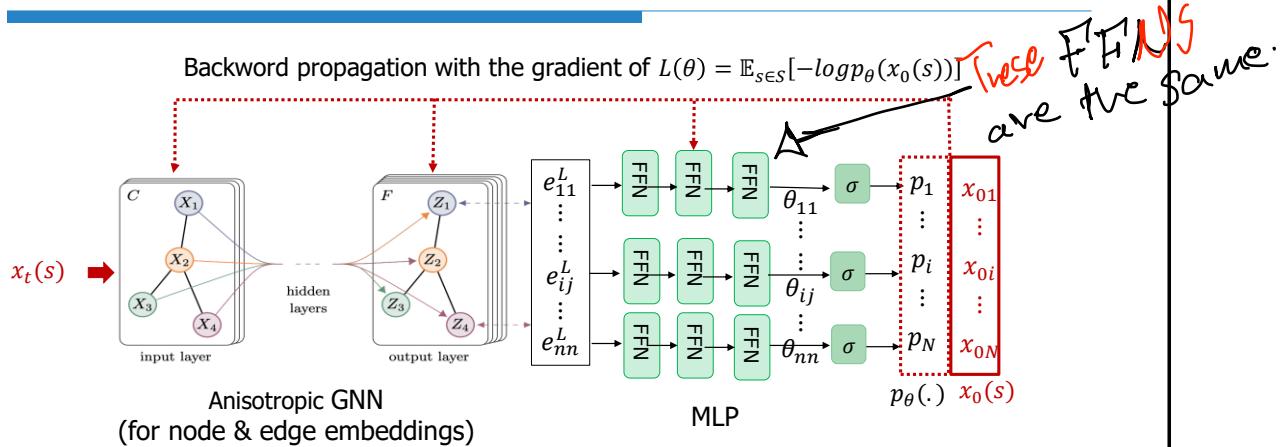
$$p_\theta(x_{t-1} | x_t) = \sum_{\hat{x}_0} q(x_{t-1} | x_t, \hat{x}_0) p_\theta(\hat{x}_0 | x_t)$$

*What is this?*



*(A circle with a cross through it)* where  $p_\theta(\hat{x}_0 | x_t)$  is estimated by a trained neural network for multi-label classification on labeled set  $\mathcal{D} = \{(x_t, x_0)\}_{t=1}^{|\mathcal{D}|}$ , which takes  $x_t$  as the input and predicts  $\hat{x}_0 \in \{0,1\}^N$  with  $p_\theta(\hat{x}_0 | x_t)$ .

## Graph-based Denoising Network for $x_t(s) \xrightarrow{\text{mapping}} x_0(s)$



4/2/2024

@Yiming Yang, 11-741 S24 Graph10 NAR CO Solvers

31

31

## Training the Denoising Network (details)

- Randomly sample K temporal stamps out of the T steps (e.g., K=50, T= 1000);
- Compute  $x_t$  given  $x_0$  for  $t = 1$  to K using a close-form solution (slide #26);
- Train the GNN+MLP model (last slide) on labeled set  $\mathcal{D} = \{(x_t, x_0)\}_{t=1}^K$ ;
- GNN input layers
  - For **TSP**, edge  $e_{ij}^0$  is initialized as the corresponding value in  $x_t(s)$ , and node  $h_i^0$  is initialized as sinusoidal features (e.g., the geographical coordinates) of the nodes in the input graph;
  - For **MIS**, edge  $e_{ij}^0$  is initialized as zeros, and node  $h_i^0$  is initialized as the corresponding values in  $x_t(s)$ ;

4/2/2024

32

32

## Solution generation per graph in the testing phase

- Start with  $x_T \in \{0,1\}^N$  with randomly sampled elements  $x_T^{(i)} \sim Ber(0.5)$ .
- Use the trained denoising network (GNN) to estimate

- $$p_\theta(x_{t-1}^{(i)} | x_t^{(i)}) = \sum_{\hat{x}_0^{(i)} \in \{0,1\}} q(x_{t-1}^{(i)} | x_t^{(i)}, \hat{x}_0^{(i)}) p_\theta(\hat{x}_0^{(i)} | x_t^{(i)})$$
- {
- $q(x_{t-1}^{(i)} | x_t^{(i)}, \hat{x}_0^{(i)})$  is computed efficiently in a closed form (slide #27);
  - $p_\theta(\hat{x}_0^{(i)} | x_t^{(i)})$  is the output at the current step by the GNN.
- Sample  $x_{t-1}^{(i)} \sim p_\theta(x_{t-1}^{(i)} | x_t^{(i)})$  for  $i = 1, \dots, N$ ; set  $t := t - 1$ , and repeat above process until  $t = 0$ .
  - Fast inference with multi-step acceleration can be done (e.g., collapsing every 50 steps into one).

4/2/2024

@Yiming Yang, 11-741 S24 Graph10 NAR CO Solvers

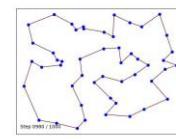
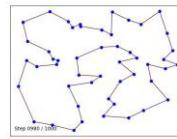
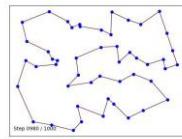
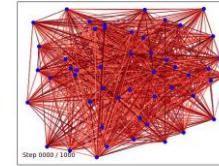
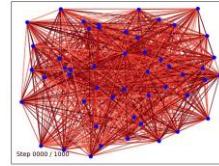
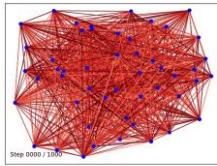
33

33

## Demo of TSP Solution Generation

(Sun &amp; Yang, NeurIPS 2023)

- Diffusion Model can generate diverse solutions



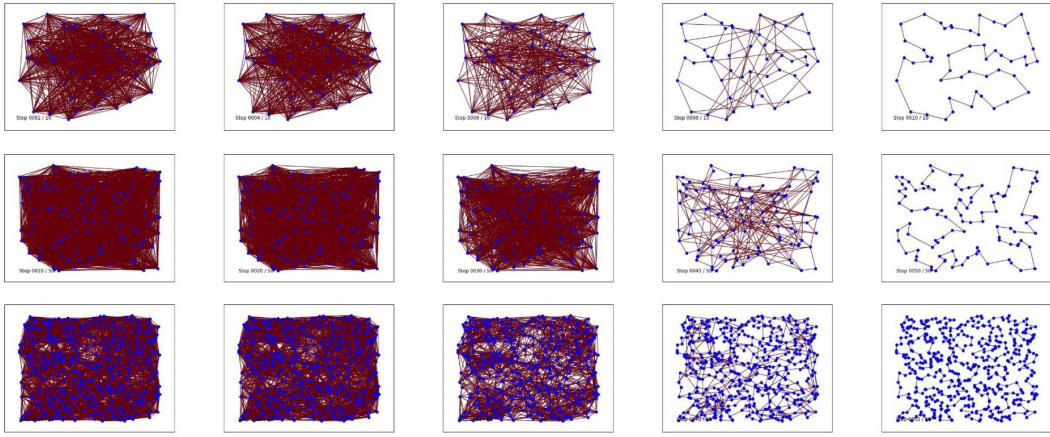
4/2/2024

@Yiming Yang, 11-741 S24 Graph10 NAR CO Solvers

35

35

## DIFUSCO on TSP-50, TSP-100, TSP-500



36

36

### Results on Small Problems [Sun & Yang, NeurIPS 2023]

ALGORITHM	TYPE	TSP-50		TSP-100	
		LENGTH ↓	GAP(%) ↓	LENGTH ↓	GAP(%) ↓
CONCORDE*	EXACT	5.69	0.00	7.76	0.00
2-OPT	HEURISTICS	5.86	2.95	8.03	3.54
AM	GREEDY	5.80	1.76	8.12	4.53
GCN	GREEDY	5.87	3.10	8.41	8.38
TRANSFORMER	GREEDY	5.71	0.31	7.88	1.42
POMO	GREEDY	5.73	0.64	7.84	1.07
SYM-NCO	GREEDY	-	-	7.84	0.94
DPDP	1k-IMPROVEMENTS	5.70	0.14	7.89	1.62
IMAGE DIFFUSION	GREEDY <sup>†</sup>	5.76	1.23	7.92	2.11
<b>OURS</b>	GREEDY <sup>†</sup>	<b>5.70</b>	<b>0.10</b>	<b>7.78</b>	<b>0.24</b>
AM	1k×SAMPLING	5.73	0.52	7.94	2.26
GCN	2k×SAMPLING	5.70	0.01	7.87	1.39
TRANSFORMER	2k×SAMPLING	5.69	0.00	7.76	0.39
POMO	8×AUGMENT	5.69	0.03	7.77	0.14
SYM-NCO	100×SAMPLING	-	-	7.79	0.39
MDAM	50×SAMPLING	5.70	0.03	7.79	0.38
DPDP	100k-IMPROVEMENTS	5.70	0.00	7.77	0.00
<b>OURS</b>	16×SAMPLING	<b>5.69</b>	<b>-0.01</b>	<b>7.76</b>	<b>-0.01</b>

\*Concorde is a TSP solver for integer coordinates.

37

## Results on Large Problems [Sun & Yang, NeurIPS 2023]

ALGORITHM	TYPE	TSP-500			TSP-1000			TSP-10000		
		LENGTH ↓	GAP ↓	TIME ↓	LENGTH ↓	GAP ↓	TIME ↓	LENGTH ↓	GAP ↓	TIME ↓
CONCORDE	EXACT	16.55*	—	37.66m	23.12*	—	6.65h	N/A	N/A	N/A
GUROBI	EXACT	16.55	0.00%	45.63h	N/A	N/A	N/A	N/A	N/A	N/A
LKH-3 (DEFAULT)	HEURISTICS	16.55	0.00%	46.28m	23.12	0.00%	2.57h	71.77*	—	8.8h
LKH-3 (LESS TRAILS)	HEURISTICS	16.55	0.00%	3.03m	23.12	0.00%	7.73m	71.79	—	51.27m
FARDEST INSERTION	HEURISTICS	18.30	10.57%	0s	25.72	11.25%	0s	80.59	12.29%	6s
AM	RL+G	20.02	20.99%	1.51m	31.15	34.75%	3.18m	141.68	97.39%	5.99m
GCN	SL+G	29.72	79.61%	6.67m	48.62	110.29%	28.52m	N/A	N/A	N/A
POMO+EAS-EMB	RL+AS+G	19.24	16.25%	12.80h	N/A	N/A	N/A	N/A	N/A	N/A
POMO+EAS-TAB	RL+AS+G	24.54	48.22%	11.61h	49.56	114.36%	63.45h	N/A	N/A	N/A
DIMES	RL+G	18.93	14.38%	0.97m	26.58	14.97%	2.08m	86.44	20.44%	4.65m
DIMES	RL+AS+G	17.81	7.61%	2.10h	24.91	7.74%	4.49h	80.45	12.09%	3.07h
OURS (DIFUSCO)	SL+G†	18.35	10.85%	3.61m	26.14	13.06%	11.86m	98.15	36.75%	28.51m
OURS (DIFUSCO)	SL+G†+2-OPT	<b>16.80</b>	<b>1.49%</b>	<b>3.65m</b>	<b>23.56</b>	<b>1.90%</b>	<b>12.06m</b>	<b>73.99</b>	<b>3.10%</b>	<b>35.38m</b>
EAN	RL+S+2-OPT	23.75	43.57%	57.76m	47.73	106.46%	5.39h	N/A	N/A	N/A
AM	RL+BS	19.53	18.03%	21.99m	29.90	29.23%	1.64h	129.40	80.28%	1.81h
GCN	SL+BS	30.37	83.55%	38.02m	51.26	121.73%	51.67m	N/A	N/A	N/A
DIMES	RL+S	18.84	13.84%	1.06m	26.36	14.01%	2.38m	85.75	19.48%	4.80m
DIMES	RL+AS+S	17.80	7.55%	2.11h	24.89	7.70%	4.53h	80.42	12.05%	3.12h
OURS (DIFUSCO)	SL+S	17.23	4.08%	11.02m	25.19	8.95%	46.08m	95.52	33.09%	6.59h
OURS (DIFUSCO)	SL+S+2-OPT	<b>16.65</b>	<b>0.57%</b>	<b>11.46m</b>	<b>23.45</b>	<b>1.43%</b>	<b>48.09m</b>	<b>73.89</b>	<b>2.95%</b>	<b>6.72h</b>
ATT-GCN	SL+MCTS	16.97	2.54%	2.20m	23.86	3.22%	4.10m	74.93	4.39%	21.49m
DIMES	RL+MCTS	16.87	1.93%	2.92m	23.73	2.64%	6.87m	74.63	3.98%	29.83m
DIMES	RL+AS+MCTS	16.84	1.76%	2.15h	23.69	2.46%	4.62h	74.06	3.19%	3.57h
OURS (DIFUSCO)	SL+MCTS	<b>16.63</b>	<b>0.46%</b>	<b>10.13m</b>	<b>23.39</b>	<b>1.17%</b>	<b>24.47m</b>	<b>73.62</b>	<b>2.58%</b>	<b>47.36m</b>

4/2/2024

@Yiming Yang, 11-741 S24 Graph10 NAR CO Solvers

38

## Concluding Remarks

- **Bad News:** ML solvers cannot beat exact solvers for small graphs ( $n \leq 100$  nodes) because exact solvers guarantees to find optimal solutions when they can scale.
- **Good News:** ML solvers, especially with the advanced NAR neural solvers (e.g., DIMES and DIFUSCO), can find **near-optimal solutions** for large problems that traditional **exact solvers fail to scale up**.
- **Good News:** The advanced NAR neural solvers have achieved the performance level close to the best **domain-specific heuristic solvers but without hand-craft heuristics**.
- **Follow-up efforts:** training diffusion-based models in unsupervised setting (without knowing the best solution per graph); pretrained LLMs (next)

3/28/2024

@Yiming Yang, 11-741 S24 Graph9 AR CO Solvers

39

## References

- [NeurIPS 2022] R Qiu\*, Z Sun\*, Z. and Y Yang. [DIMES: A differentiable meta solver for combinatorial optimization problems](#)
- [NeurIPS 2023] Sun, Z. and Yang, Y., 2023. [Difusco: Graph-based Diffusion Solvers for Combinatorial Optimization](#)
- [Tutorial 2022] Calvin Luo. Understanding Diffusion Models: [A Unified Perspective: An intuitive, accessible tutorial on diffusion models](#).

## Appendix: ELBO of the Diffusion Model

$$\begin{aligned}
 \log p(\mathbf{x}) &\geq \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[ \log \frac{p(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \right] \\
 &= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[ \log \frac{p(\mathbf{x}_T) \prod_{t=1}^T p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{\prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1})} \right] \\
 &= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[ \log \frac{p(\mathbf{x}_T) p_{\theta}(\mathbf{x}_0|\mathbf{x}_1) \prod_{t=2}^T p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_1|\mathbf{x}_0) \prod_{t=2}^T q(\mathbf{x}_t|\mathbf{x}_{t-1})} \right] \\
 &= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[ \log \frac{p(\mathbf{x}_T) p_{\theta}(\mathbf{x}_0|\mathbf{x}_1) \prod_{t=2}^T p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_1|\mathbf{x}_0) \prod_{t=2}^T q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)} \right] \\
 &= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[ \log \frac{p_{\theta}(\mathbf{x}_T) p_{\theta}(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)} + \log \prod_{t=2}^T \frac{p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)} \right] \\
 &= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[ \log \frac{p(\mathbf{x}_T) p_{\theta}(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)} + \log \prod_{t=2}^T \frac{p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}} \right]
 \end{aligned}$$

Due to Markov property

Due to Bayes rule

## Appendix: ELBO of the Diffusion Model

$$\begin{aligned}
& \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[ \log \frac{p(\mathbf{x}_T)p_{\theta}(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)} + \log \prod_{t=2}^T \frac{p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}} \right] \\
&= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[ \log \frac{p(\mathbf{x}_T)p_{\theta}(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)} + \log \prod_{t=2}^T \frac{p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{\frac{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)q(\mathbf{x}_t|\mathbf{x}_0)}{q(\mathbf{x}_{t-1}|\mathbf{x}_0)}} \right] \\
&= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[ \log \frac{p(\mathbf{x}_T)p_{\theta}(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_1|\mathbf{x}_0)} + \log \frac{q(\mathbf{x}_T|\mathbf{x}_0)}{q(\mathbf{x}_1|\mathbf{x}_0)} + \log \prod_{t=2}^T \frac{p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} \right] \\
&= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[ \log \frac{p(\mathbf{x}_T)p_{\theta}(\mathbf{x}_0|\mathbf{x}_1)}{q(\mathbf{x}_T|\mathbf{x}_0)} + \sum_{t=2}^T \log \frac{p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} \right] \\
&= \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} [\log p_{\theta}(\mathbf{x}_0|\mathbf{x}_1)] + \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[ \log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)} \right] + \sum_{t=2}^T \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[ \log \frac{p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} \right] \\
&= \mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)} [\log p_{\theta}(\mathbf{x}_0|\mathbf{x}_1)] + \mathbb{E}_{q(\mathbf{x}_T|\mathbf{x}_0)} \left[ \log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T|\mathbf{x}_0)} \right] + \sum_{t=2}^T \mathbb{E}_{q(\mathbf{x}_t, \mathbf{x}_{t-1}|\mathbf{x}_0)} \left[ \log \frac{p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t)}{q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)} \right] \\
&= \underbrace{\mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)} [\log p_{\theta}(\mathbf{x}_0|\mathbf{x}_1)]}_{\text{reconstruction term}} - \underbrace{D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{\text{prior matching term}} - \underbrace{\sum_{t=2}^T \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} [D_{\text{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t))]}_{\text{denoising matching term}}
\end{aligned}$$

4/2/2024

@Yiming Yang, 11-741 S24 Graph10 NAR CO Solvers

42