DIFUSCO: Graph-based Diffusion Solvers for Combinatorial Optimization

2025. 04. 11

AI&OPT
김정현

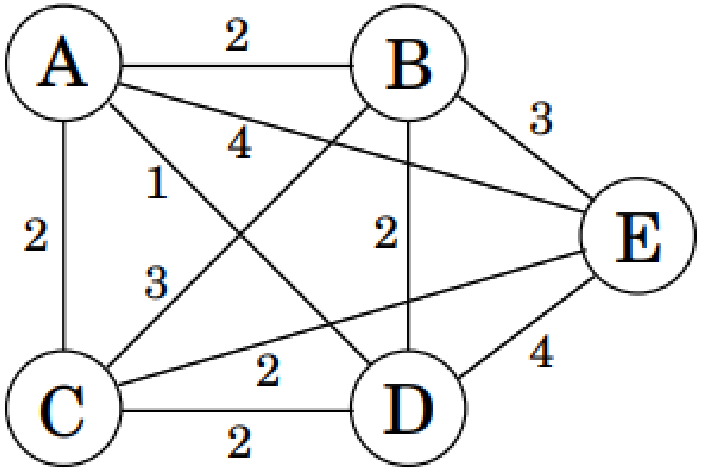
37th Conference on Neural Information Processing Systems (NeurIPS 2023)

Contents

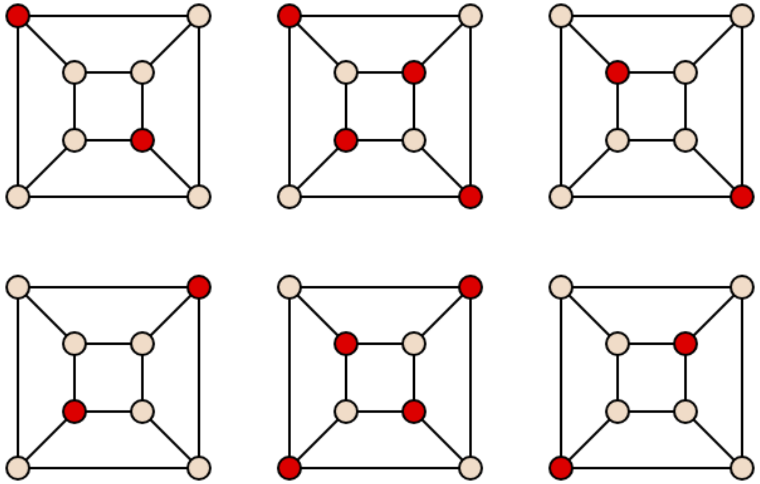
01 Introduction

❖ Combinatorial Optimization (CO) Problem

- ✓ A mathematical problem that aims to find the optimal solution in a discrete solution space.
- ✓ Often belongs to the class of NP-Complete (NPC) problems, making them difficult to solve in polynomial time.
- ✓ Traditionally solved using integer programming and heuristic-based methods.
- ✓ Examples :
  - TSP (Traveling Salesman Problem)
    : Finding the shortest possible route that visits each city exactly once and returns to the starting point.
  - MIS (Maximum Independent Set) :
    : Finding the largest subset of nodes in a graph such that no two nodes are connected.



<TSP>



<MIS>

02 Related Work

- **Autoregressive Constructive Solvers**
  - ✓ Generate solutions sequentially, one element at a time.
  - ✓ High computational complexity
  - ✓ Difficult to scale to large problem sizes

- **Non-Autoregressive Constructive Solvers**
  - ✓ Generate the entire solution in one shot.
  - ✓ Struggles to model multi-modal distributions
  - ✓ Hard to find good solutions when multiple optima exist (e.g., in the same graph)

- **Improvement Heuristic Solvers**
  - ✓ Iteratively refine an initial solution.
  - ✓ Examples: 2-opt, node swap
  - ✓ Sparse rewards and low sampling efficiency → leads to slow training and inference

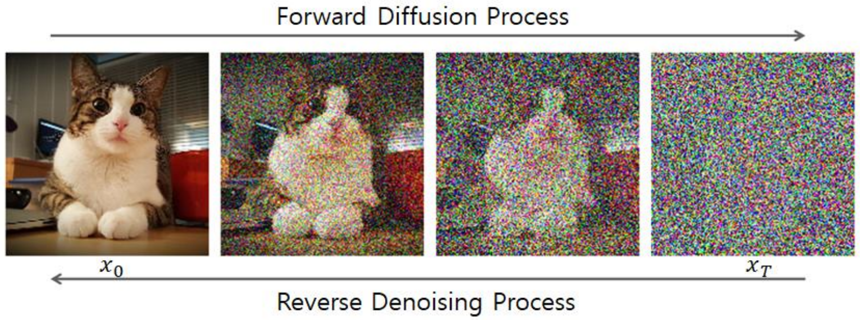
03 Diffusion Model

❖ Problem Definition

- ✓ For a given problem instance s, the set of feasible solutions is represented as a binary vector space $x \in \{0.1\}^n$.
    - TSP: Whether each edge is selected
    - MIS: Whether each node is selected
- ✓ The objective is to minimize the following cost: $c_s(x) = cost(x,s) + valid(x,s)$
    - $cost_{TSP}(x,s) = \sum_i x_i \cdot {d_i}^{(s)}$ (sum of selected edge distances)
    - $cost_{MIS}(x,s) = \sum_i (1 - x_i)$ (maximize number of selected independent nodes)

❖ Diffusion Model

- ✓ Originally used for image and text generation, diffusion models are now being applied to CO problems.
- ✓ Learns to generate optimal solutions via supervised training.
- ✓ Each training sample provides optimal solution ${x_s}^*$ → Train the model to generate ${x_s}^*$ as accurately as possible.
- ✓ Forward process: add noise to clean solution $x_0 \rightarrow x_T$
- ✓ Reverse process: Denoise x_T step-by-step to recover x_0
- ✓ Types of Noise:
    - Continuous (Gaussian) : Used for real-valued data
    - Discrete (Bernoulli) : Used for binary solution spaces



04 DIFUSCO_Dataset preparation

❖ Data Generation

- ✓ Randomly generate *N* cities as 2D coordinates (x, y) in the range [0, 1].
- ✓ Use LKH or Concorde solver to compute the optimal tour.
- ✓ Convert the tour into a binary adjacency matrix.
- ✓ Save each instance to .txt file:
  - [x1 y1 x2 y2 … xN yN output t1 t2 … tN t1]
  - t1 is repeated to form a closed loop.

❖ Dataset Construction

- ✓ Load the .txt file line by line
- ✓ For each line:
  - Extract coordinates → points
  - Extract tour → build adj_matrix

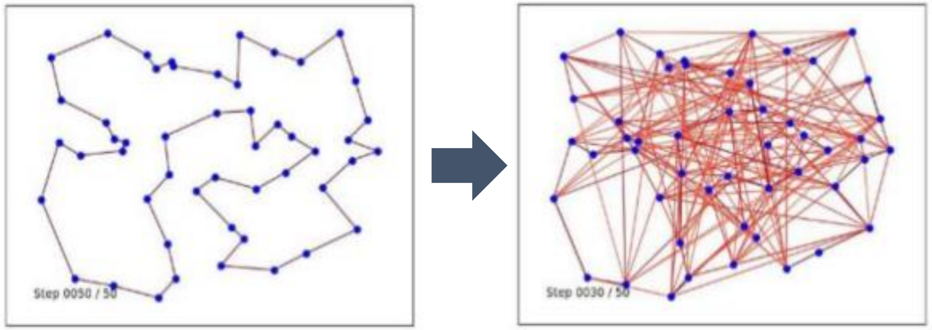
❖ Batching & Model Input

- ✓ DataLoader creates mini-batches (e.g., batch size = 128).
- ✓ Each batch includes: points, adj_matrix, t

04 DIFUSCO_Forward process

❖ **Objective**

- ✓ To create a noisy version xt from the original solution x0.
- ✓ So, the model can learn to denoise it back during training.



❖ **Step-by-step process**

- ✓ From the ground-truth TSP tour, construct the binary adjacency matrix x0.
- ✓ Convert each edge(0 or 1) into a one-hot vector.
- ✓ Randomly sample a timestep t ∈ [1, T].
- ✓ Using the cumulative transition matrix Qt, add noise to x0 to generate a noised sample xt.
- ✓ For training stability:
  - The binary sample xt ∈ {0,1} is scaled to the range [−1,1].
  - Small random noise (5%) is added to further regularize the input.

Tour

0	1	4	3	2

Adj_matrix (x0)

0	1	0	0	0
0	0	0	0	1
1	0	0	0	0
0	0	1	0	0
0	0	0	1	0

xt

0	1	0	0	1
1	0	0	1	1
1	1	0	0	0
1	1	1	0	0
1	0	0	0	1

[−1, 1] scale

-1	1	-1	-1	1
1	-1	-1	1	1
1	1	-1	-1	-1
1	1	1	-1	-1
1	-1	-1	-1	1

+ small random noise → final xt

-1.03	1.01	-1.04	-1.03	1.04
1.03	1.00	-1.01	-1.02	-1.00
1.02	1.01	-1.03	-1.02	-1.01
1.01	1.01	1.03	-1.04	-1.01
1.00	-1.04	-1.03	-1.01	1.02

04 DIFUSCO_Reverse process

❖ **Objective**

✓ To recover the original clean adjacency matrix x0 from a noisy version xt.

❖ **Graph-based Denoising Network (AGNN)**

✓ Input Components:

- Node coordinates(x), Noisy adjacency matrix(xt), current diffusion time step (t)

✓ Embedding Steps::

- Node Embedding: Node, edge, and timestep embeddings.
- Edge Gating:

  → Decide how much info node i receives from node j.
- Message Passing:

  → Aggregate messages and update node features.

✓ Output: x0_pred → edge logits

→ Represents predicted probability of edge presence.

❖ **Loss**

✓ Use cross entropy loss with ground-truth x0.

✓ Predicts whether edge exists (1) or not (0)

x

0.94	0.41
0.89	0.63
0.40	0.79
0.67	0.02
0.95	0.91

xt

-1.03	1.01	-1.04	-1.03	1.04
1.03	1.00	-1.01	-1.02	-1.00
1.02	1.01	-1.03	-1.02	-1.01
1.01	1.01	1.03	-1.04	-1.01
1.00	-1.04	-1.03	-1.01	1.02

x0_pred

0.75	0.75	0.67	0.76	0.74
0.77	0.77	0.67	0.68	0.75
0.67	0.75	0.67	0.66	0.66
0.74	0.75	0.74	0.75	0.66
0.75	0.74	0.76	0.75	0.66

-0.47	-0.47	-0.26	-0.47	-0.47
-0.48	-0.48	-0.26	-0.48	-0.47
-0.25	-0.47	-0.27	-0.26	-0.37
-0.43	-0.44	-0.47	-0.48	-0.26
-0.42	-0.47	-0.46	-0.48	-0.37

x0

0	1	0	0	0
0	0	0	0	1
1	0	0	0	0
0	0	1	0	0
0	0	0	1	0

Cross Entropy Loss
= 0.4961

04 DIFUSCO_Inference

❖ Objective

- ✓ To reconstruct the original clean solution x0 from a fully noised sample xT.

❖ Inference Process

- ✓ Initialization: Start from a fully noised adjacency matrix xT.
- ✓ Iterative Denoising Loop (Reverse Diffusion):
  - Model predicts x0_pred from current xt.
  - Compute the posterior q(xt-1|xt, x0_pred) using diffusion rule.
  - Sample xt-1 from the posterior.
  - Repeat until reaching x0.
- ✓ Output: The final x0 is a denoised adjacency matrix representing edge probabilities.
- ✓ May contain multiple disconnected subtours → requires post-processing

❖ Post-processing

- ✓ Merge disconnected subtours into a valid complete tour
- ✓ Apply 2-opt algorithm to refine the path
- ✓ Compute the final tour cost using Euclidean distance.

04 DIFUSCO_Inference

❖ **Fast Inference Scheduling**

✓ To reduce inference time, skip full diffusion steps:

- Use only a subset of the total steps (e.g., 50 out of 1000)

✓ Step selection strategies:

- Linear: Uniform step intervals
- Cosine: Large intervals early, denser near the end

❖ **Decoding Strategy**

✓ At the final step, the diffusion model outputs a heatmap

✓ Heatmap represents the confidence or probability of each variable (node or edge) being part of the optimal solution

✓ TSP:

- Edge heatmap: Confidence of each edge being part of the tour
- Decoding methods: Greedy + 2-opt, Sampling, MCTS

✓ MIS:

- Node heatmap: Confidence of each node being in the independent set
- Select non-conflicting nodes greedily based on ranking

05 Experiment

❖ **Datasets**

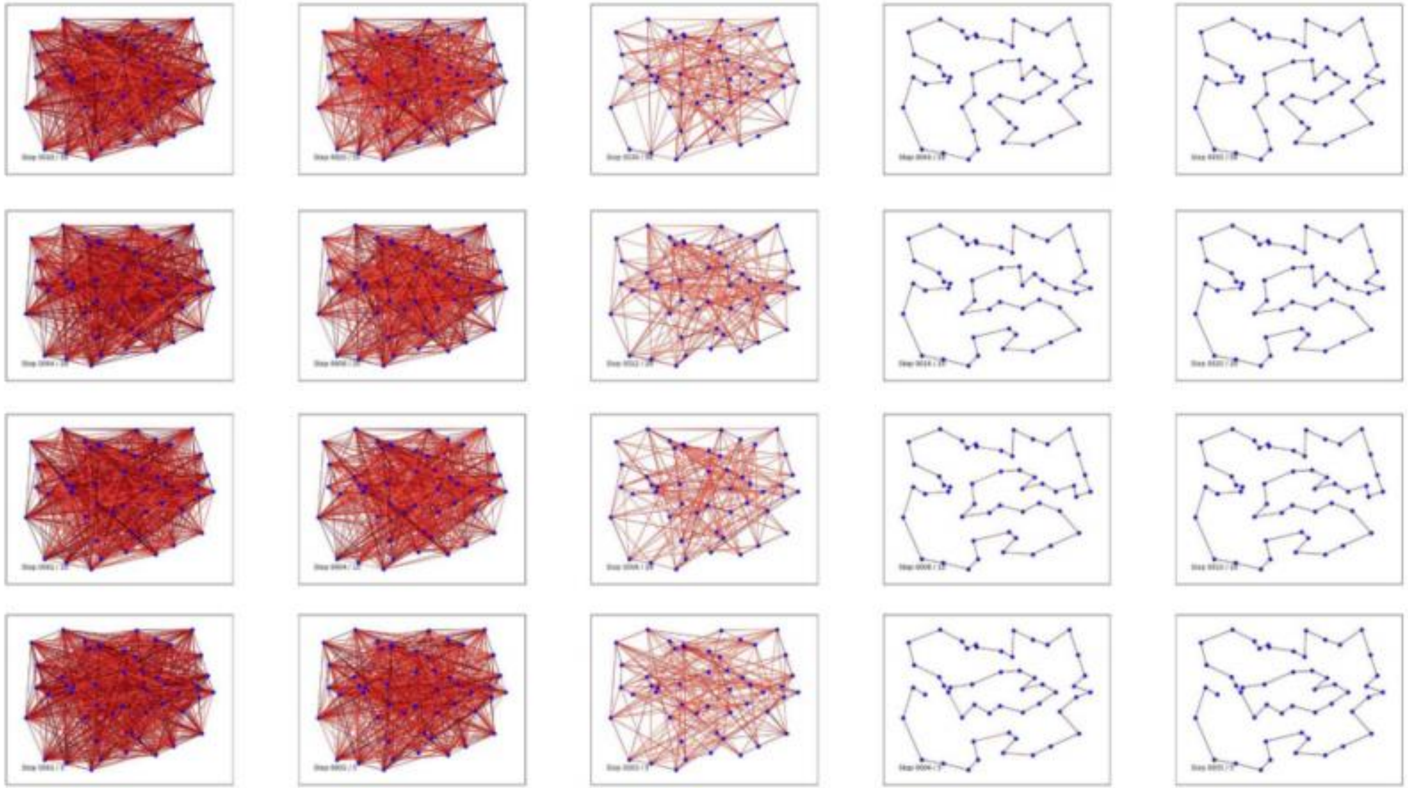
- ✓ TSP50 / TSP100 → Training labels generated using the Concorde exact solver
- ✓ TSP500 / TSP1000 / TSP 10000 → Training labels generated using the LKH-3 heuristic solver
- ✓ Sparse graph: Edge connections are limited to reduce computational complexity

❖ **Model Settings**

- ✓ Denoising step : 1000
- ✓ Linear noise schedule
- ✓ Decoding Strategy : Greedy + 2-opt

❖ **Evaluation Metrics**

- ✓ Tour length
- ✓ Performance gap
- ✓ Run time



❖ DIFUSCO training setting

- Curriculum Learning (TSP-500,1000,10000)

Problem	Dataset	Instances	Epochs	Batch Size
TSP	TSP-50	1,502,000	50	512
	TSP-100	1,502,000	50	256
	TSP-500	128,000	50	64
	TSP-1000	64,000	50	64
	TSP-10000	6,400	50	8
MIS	SATLIB	49,500	50	128
	ER-[700,800]	163,840	50	32

06 Result

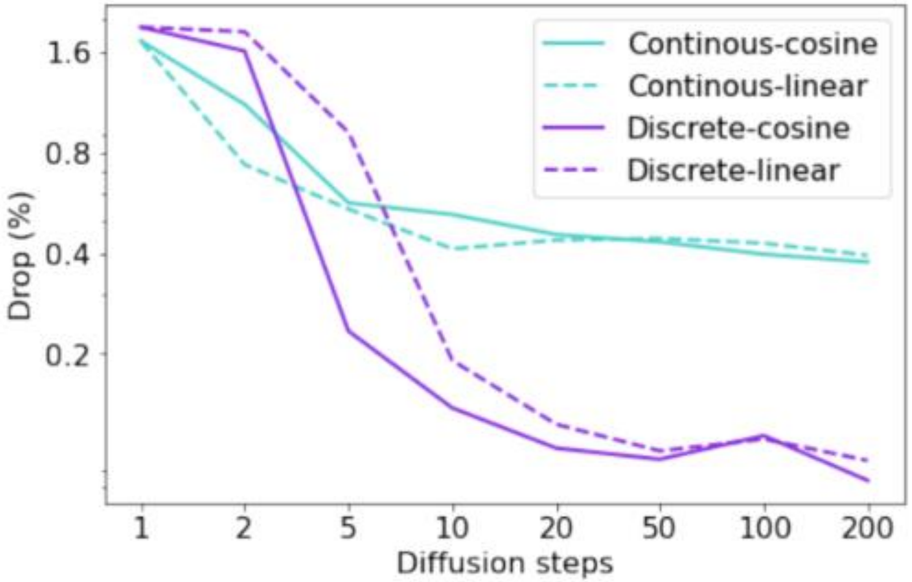
❖ **Design Analysis**

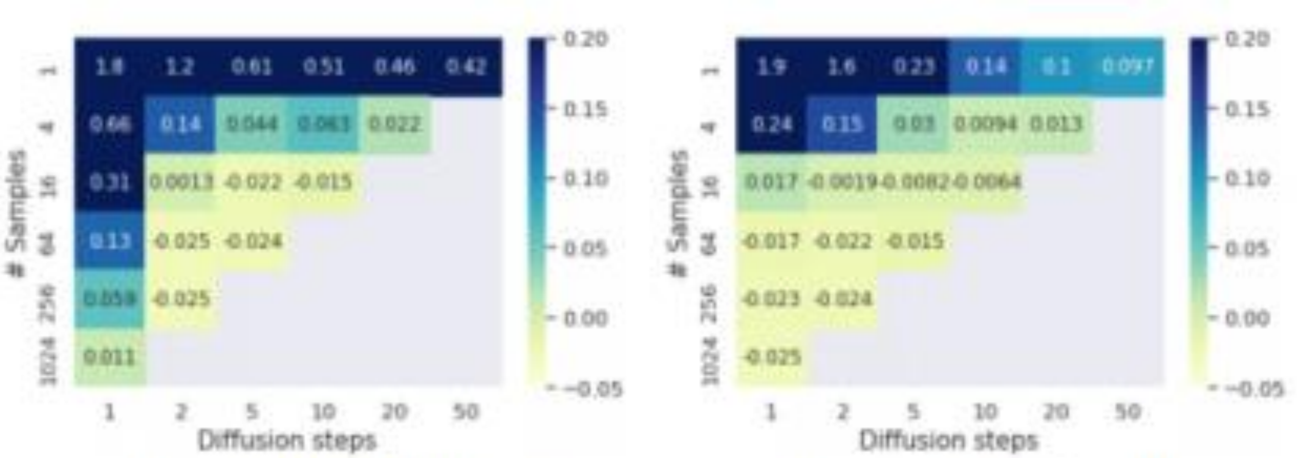
✓ Discrete vs Continuous

- Discrete diffusion with cosine scheduling shows better performance compared to continuous variants

✓ More diffusion iterations vs More sampling

- Results suggest that increased sampling with fewer steps can maintain performance while reducing inference time
- 50 (diffusion steps) X 1 (samples) and 10 (diffusion steps) X 16 (samples)





(a) Continuous diffusion (b) Discrete diffusion

06 Result

❖ Comparison to SOTA methods

ALGORITHM	TYPE	TSP-50 LENGTH↓	TSP-50 GAP(%)↓	TSP-100 LENGTH↓	TSP-100 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†	5.76	1.23	7.92	2.11
OURS	GREEDY†	**5.70**	**0.10**	**7.78**	**0.24**
AM	$1k\times$SAMPLING	5.73	0.52	7.94	2.26
GCN	$2k\times$SAMPLING	5.70	0.01	7.87	1.39
TRANSFORMER	$2k\times$SAMPLING	5.69	0.00	7.76	0.39
POMO	$8\times$AUGMENT	5.69	0.03	7.77	0.14
SYM-NCO	$100\times$SAMPLING	-	-	7.79	0.39
MDAM	$50\times$SAMPLING	5.70	0.03	7.79	0.38
DPDP	$100k$-IMPROVEMENTS	5.70	0.00	7.77	0.00
OURS	$16\times$SAMPLING	**5.69**	**-0.01**	**7.76**	**-0.01**

ALGORITHM	TYPE	TSP-500 LENGTH↓	TSP-500 GAP↓	TSP-500 TIME↓	TSP-1000 LENGTH↓	TSP-1000 GAP↓	TSP-1000 TIME↓	TSP-10000 LENGTH↓	TSP-10000 GAP↓	TSP-10000 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
FARTHEST 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	**16.80**	**1.49%**	**3.65**m	**23.56**	**1.90%**	**12.06**m	**73.99**	**3.10%**	**35.38**m
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	**16.65**	**0.57%**	**11.46**m	**23.45**	**1.43%**	**48.09**m	**73.89**	**2.95%**	**6.72**h
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	**16.63**	**0.46%**	**10.13**m	**23.39**	**1.17%**	**24.47**m	**73.62**	**2.58%**	**47.36**m

06 Result

❖ **Generalization Tests**

✓ Results When Trained on TSP50 and Applied to Larger Instances

METHOD	TYPE	SATLIB			ER-[700-800]		
		SIZE ↑	GAP ↓	TIME ↓	SIZE ↑	GAP ↓	TIME ↓
KAMIS	HEURISTICS	425.96*	—	37.58m	44.87*	—	52.13m
GUROBI	EXACT	425.95	0.00%	26.00m	41.38	7.78%	50.00m
INTEL	SL+G	420.66	1.48%	23.05m	34.86	22.31%	6.06m
INTEL	SL+TS	N/A	N/A	N/A	38.80	13.43%	20.00M
DGL	SL+TS	N/A	N/A	N/A	37.26	16.96%	22.71m
LWD	RL+S	422.22	0.88%	18.83m	41.17	8.25%	6.33m
DIMES	RL+G	421.24	1.11%	24.17m	38.24	14.78%	6.12m
DIMES	RL+S	423.28	0.63%	20.26m	**42.06**	**6.26%**	12.01m
OURS	SL+G	**424.50**	**0.34%**	8.76m	38.83	12.40%	8.80m
OURS	SL+S	**425.13**	**0.21%**	23.74m	41.12	8.36%	26.67m

07 Conclusion

❖ **Contribution**

- ✓ One of the first successful applications of diffusion models to combinatorial optimization (CO) problems
- ✓ Demonstrates stronger scalability, expressiveness, and performance compared to traditional solvers
- ✓ Achieves efficient and generalizable performance on both TSP and MIS tasks, outperforming prior approaches

❖ **Future Work**

- ✓ Extension to broader NP-Complete (NPC) problems
- ✓ Integration of Equivariant Graph Neural Networks (GNNs)
- ✓ Exploration of accelerated inference techniques for faster sampling