DIMES: A **DI**fferentiable **ME**ta **S**olver for Combinatorial Optimization Problems

Ruizhong Qiu*
Department of Computer Science
University of Illinois Urbana-Champaign
rq5@illinois.edu

Zhiqing Sun*, Yiming Yang
Language Technologies Institute
Carnegie Mellon University
{zhiqings,yiming}@cs.cmu.edu

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Carnegie Mellon University

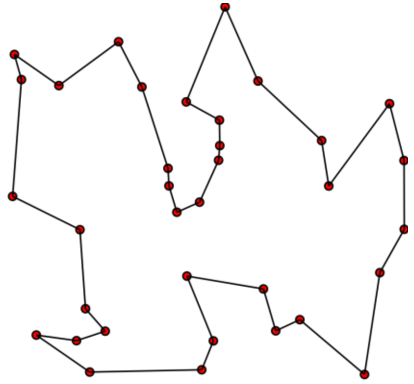
* Equal contribution.

Introduction

- Recent advances of deep reinforcement learning (DRL) has shown promises in solving NP-hard combinatorial optimization (CO) problems without manual injection of domain-specific expert knowledge.

- However, most DRL solvers can only scale to graphs with up to hundreds of nodes.

- We address the scalability challenge by proposing DIMES (**DI**fferentiable **ME**ta **S**olver).
    - We introduce continuous heatmaps to compactly represent feasible solutions.
    - We employ meta-learning over problem instances to capture the common nature across the instances.
    - DIMES can scale to graphs with up to 10,000 nodes.



Formal Definitions

- Given a problem instance s, the goal is finding an optimal solution f_s^* from the feasible solution space $\mathcal{F}_s$ to minimize the cost function $c_s\colon \mathcal{F}_s \to \mathbb{R}$:
$$f_s^* = \operatorname*{argmin}_{f \in \mathcal{F}_s} c_s(f)\,.$$

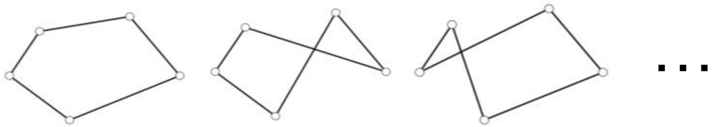
- Solutions are encoded as 0/1 vectors $f \in \{0,1\}^{|\mathcal{V}_s|}$, where $\mathcal{V}_s$ denotes the set of variables for the problem instance s.

- To learn the solution differentiably, we introduce a continuous vector $\theta \in \mathbb{R}^{|\mathcal{V}_s|}$ (called a *heatmap*) to parameterize a probability distribution p_θ over feasible solution space $\mathcal{F}_s$:
$$p_\theta(f \mid s) \propto \exp\left(\textstyle\sum_{i \in \mathcal{V}_s} f_i \cdot \theta_i\right) \qquad \text{subject to} \qquad f \in \mathcal{F}_s.$$
- Optimize θ by minimizing the expected cost $\ell_p(\theta|s) = \mathbb{E}_{f \sim p_\theta}[c_s(f)]$ over p_θ:
$$\theta_s^* = \operatorname*{argmin}_{\theta \in \mathbb{R}^{|\mathcal{V}_s|}} \mathbb{E}_{f \sim p_\theta}[c_s(f)]\,.$$

Problem Definitions

Traveling Salesman Problem (TSP):

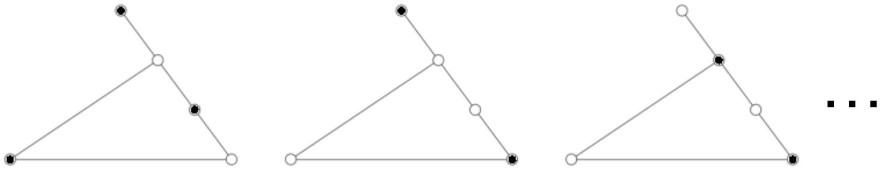
- Feasible solutions $\mathcal{F}_s$ are tours, which visit each node exactly once and return to the start node at the end.
- The cost c_s is the sum of edge lengths in the tour.
- Variables $\mathcal{V}_s$ corresponds to edges, where $f_{i,j} = 1$ means edge (i, j) is in the tour.



$\mathcal{F}_s$ of a 5-node TSP instance

Maximum Independent Set (MIS):

- Feasible solutions $\mathcal{F}_s$ are independent node subsets, in which nodes have no edges to each other.
- The cost c_s is the negation of the size of the independent subset.
- Variables $\mathcal{V}_s$ corresponds to nodes, where $f_i = 1$ means node i is in the independent subset.



$\mathcal{F}_s$ of a 5-node MIS instance

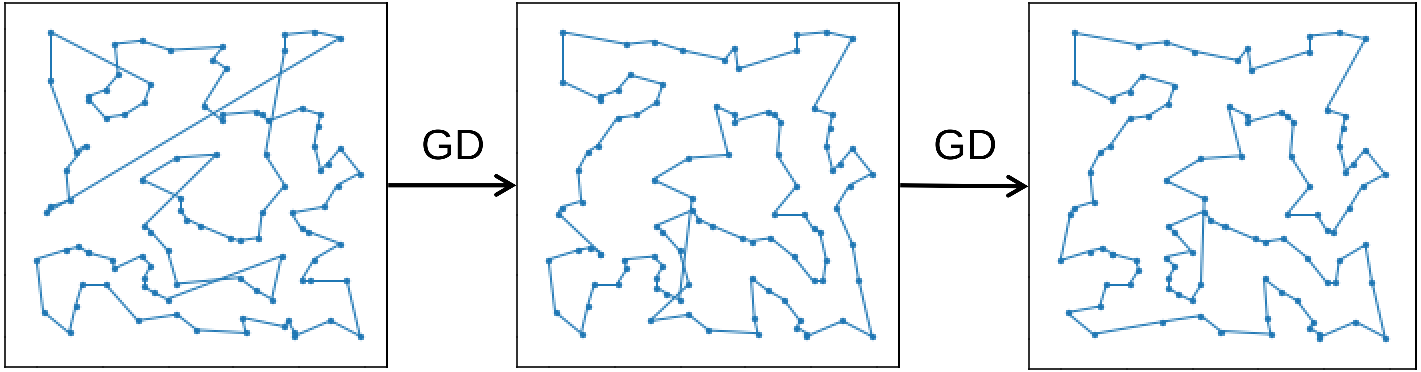
Gradient-based Optimization

- Since sampling from p_θ is inefficient, we propose to design an auxiliary distribution q_θ over $\mathcal{F}_s$, from which sampling is efficient.
- Optimize θ to minimize the expected cost $\ell_q(\theta|s) = \mathbb{E}_{f\sim q_\theta}[c_s(f)]$ over q_θ instead of p_θ.
- Gradient descent (GD) with REINFORCE-based gradient estimator:

$$\nabla_\theta \mathbb{E}_{f\sim q_\theta}[c_s(f)] = \mathbb{E}_{f\sim q_\theta}\left[\left(c_s(f) - b(s)\right)\nabla_\theta \log q_\theta(f)\right].$$

  - $b(s)$: a baseline function to reduce the variance of the gradient estimator.

Illustration for TSP



Auxiliary Distribution Designs

(For brevity, we omit conditional notations on s.)

For TSP on n nodes:

- A feasible solution f as a permutation π_f of n nodes, where $\pi_f(0) = \pi_f(n)$.
- Choose the start node $\pi_f(0)$ randomly:

$$q_\theta^{\mathrm{TSP}}(f) \coloneqq \sum_{j=0}^{n-1} \frac{1}{n} \cdot q_{\mathrm{TSP}}\left(\pi_f \middle| \pi_f(0) = j\right).$$

- Chain rule in the visiting order:

$$q_{\mathrm{TSP}}\left(\pi_f \middle| \pi_f(0)\right) \coloneqq \prod_{i=1}^{n-1} q_{\mathrm{TSP}}\left(\pi_f(i) \middle| \pi_f(< i)\right).$$

- Heatmap: matrix $\theta \in \mathbb{R}^{n\times n}$ for edges.

$$q_{\mathrm{TSP}}\left(\pi_f(i) \middle| \pi_f(< i)\right) \coloneqq \frac{\exp \theta_{\pi_f(i-1),\pi_f(i)}}{\sum_{j=i}^{n} \exp \theta_{\pi_f(i-1),\pi_f(j)}}.$$

For MIS on n nodes:

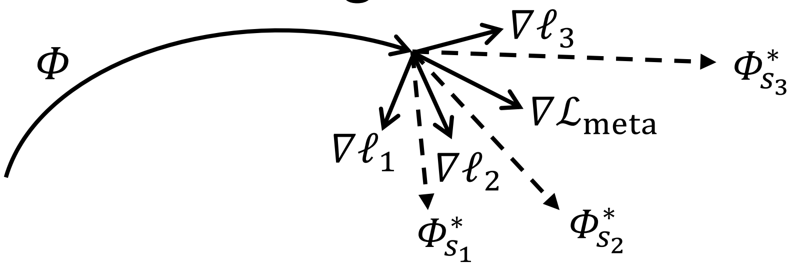
- $\{a\}_f$: the set of all possible orderings a of the nodes in the independent set f.

$$q_\theta^{\mathrm{MIS}}(f) \coloneqq \sum_{a\in\{a\}_f} \prod_{i=1}^{|a|} q_{\mathrm{MIS}}(a_i | a_{<i}).$$

- $\mathcal{G}(a_{<i})$: the set of nodes that have no edge to $\{a_1, \ldots, a_{i-1}\}$.
- Heatmap: vector $\theta \in \mathbb{R}^n$ for nodes.

$$q_{\mathrm{MIS}}(a_i | a_{<i}) \coloneqq \frac{\exp \theta_{a_i}}{\sum_{j\in\mathcal{G}(a_{\{<i\}})} \exp \theta_j}.$$

Meta-Learning Framework



⟶ Meta-learning

- - - → Instance-specific adaptation

- We train a meta-network F_Φ over a collection of problem instances $\mathcal{C} = \{(\kappa_s, A_s)\}$ to predict instance-specific heatmap $\theta_s = F_\Phi(\kappa_s, A_s)$.
- We adapt parameters Φ to each instance s via T gradient steps with learning rate α.

$$\Phi_s^{(0)} = \Phi, \qquad \Phi_s^{(t)} = \Phi_s^{(t-1)} - \alpha \nabla_{\Phi_s^{(t-1)}} \ell_q\left(\theta_s^{(t-1)} \middle| s\right), \qquad t = 1, \dots, T,$$

$$\theta_s^{(t)} = F_{\Phi_s^{(t)}}(\kappa_s, A_s), \qquad t = 0, \dots, T.$$

- Meta-objective:

$$\mathcal{L}_{\text{meta}}(\Phi|\mathcal{C}) = \mathbb{E}_{s\in\mathcal{C}}\left[\ell_q\left(\theta_s^{(T)} \middle| s\right)\right].$$

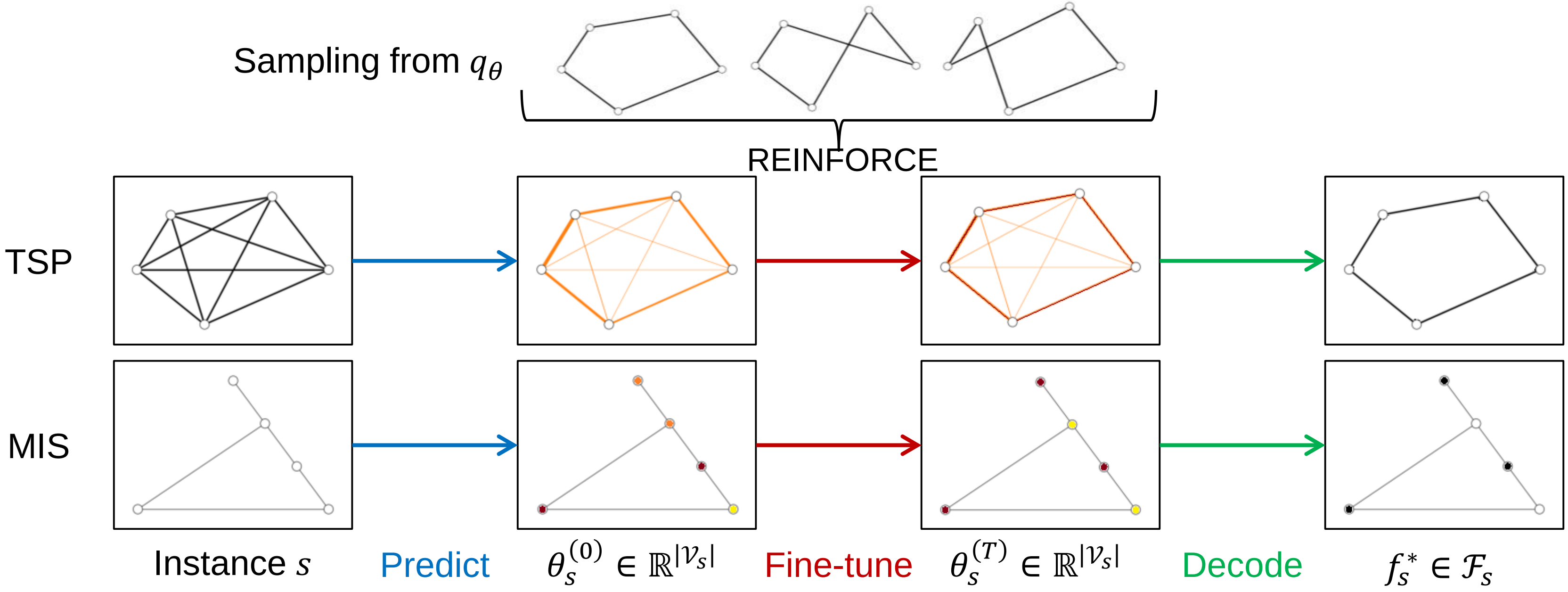
- First-order approximation of meta-gradient:

$$\nabla_\Phi \mathcal{L}_{\text{meta}}(\Phi|\mathcal{C}) \approx \mathbb{E}_{s\in\mathcal{C}}\left[\nabla_{\Phi_s^{(T)}} F_{\Phi_s^{(T)}}(\kappa_s, A_s) \cdot \nabla_{\theta_s^{(T)}} \ell_q\left(\theta_s^{(T)} \middle| s\right)\right].$$

Inference Procedure

Overall inference procedure has three steps:

1. Predict an initial heatmap for the problem instance using the GNN.
2. Fine-tune the heatmap via REINFORCE and sampling from the auxiliary distribution.
3. Decode the heatmap into a feasible solution (Greedy / Sampling / Monte Carlo Tree Search).



Main Results for TSP

- We directly train on large-scale graphs.
- DIMES is able to scale up to graphs with 10,000 nodes.
- DIMES outperforms both DRL and supervised methods.

Method	Type	TSP-500 Length ↓	TSP-500 Drop ↓	TSP-500 Time ↓	TSP-1000 Length ↓	TSP-1000 Drop ↓	TSP-1000 Time ↓	TSP-10000 Length ↓	TSP-10000 Drop ↓	TSP-10000 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
	RL+AS+G	17.81	7.61%	2.10h	24.91	7.74%	4.49h	80.45	12.09%	3.07h
	RL+S	18.84	13.84%	1.06m	26.36	14.01%	2.38m	85.75	19.48%	4.80m
	RL+AS+S	17.80	7.55%	2.11h	24.89	7.70%	4.53h	80.42	12.05%	3.12h
	RL+MCTS	16.87	1.93%	2.92m	23.73	2.64%	6.87m	74.63	3.98%	29.83m
	RL+AS+MCTS	**16.84**	**1.76%**	2.15h	**23.69**	**2.46%**	4.62h	**74.06**	**3.19%**	3.57h

Main Results for MIS

- DIMES significantly outperforms supervised method (Intel) in large-scale settings.
- Despite being a general CO solver, DIMES is competitive with specially designed neural MIS solver (LwD).

Method	Type	SATLIB			ER-[700-800]			ER-[9000-11000]		
		Size ↑	Drop ↓	Time ↓	Size ↑	Drop ↓	Time ↓	Size ↑	Drop ↓	Time ↓
KaMIS	OR	425.96*	—	37.58m	44.87*	—	52.13m	381.31*	—	7.6h
Gurobi	OR	425.95	0.00%	26.00m	41.38	7.78%	50.00m	N/A	N/A	N/A
Intel	SL+TS	N/A	N/A	N/A	38.80	13.43%	20.00m	N/A	N/A	N/A
Intel	SL+G	420.66	1.48%	23.05m	34.86	22.31%	6.06m	284.63	25.35%	5.02m
DGL	SL+TS	N/A	N/A	N/A	37.26	16.96%	22.71m	N/A	N/A	N/A
LwD	RL+S	422.22	0.88%	18.83m	41.17	8.25%	6.33m	**345.88**	**9.29%**	7.56m
DIMES (ours)	RL+G	421.24	1.11%	24.17m	38.24	14.78%	6.12m	320.50	15.95%	5.21m
DIMES (ours)	RL+S	**423.28**	**0.63%**	20.26m	**42.06**	**6.26%**	12.01m	332.80	12.72%	12.51m

Conclusion

- We addresses the scalability challenge of DRL for CO by proposing DIMES, which employs a compact continuous parameterization and a meta-learning strategy.
- For TSP and MIS, DIMES can scale up to graphs with ten thousand nodes. While trained without ground truth solutions, DIMES can outperform supervised methods.
- Future work may extend DIMES to general Mixed Integer Programming (MIP) by reducing each integer value within range $[U]$ to a sequence of $\lceil \log_2 U \rceil$ bits [1].

[1] Nair et al. Solving mixed integer programs using neural networks *arXiv*:2012.13349, 2020.