Learning Heuristics for Minimum Latency Problem with RL & GNN

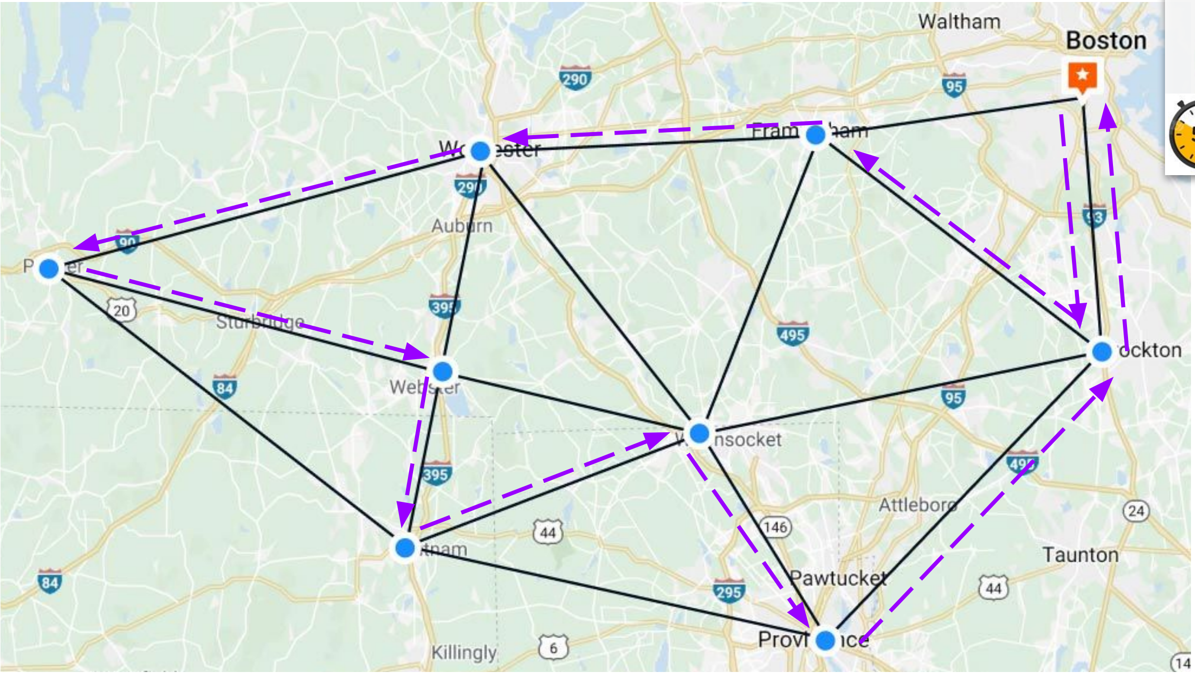
Siqi Hao, Salar Hosseini, Philip Huang, Mohamed Khodeir

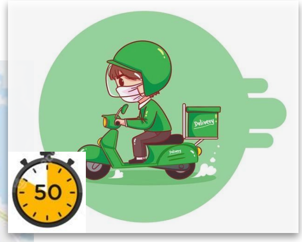
December 6, 2021

Problem

- Minimum Latency Problem
- Why is it important?

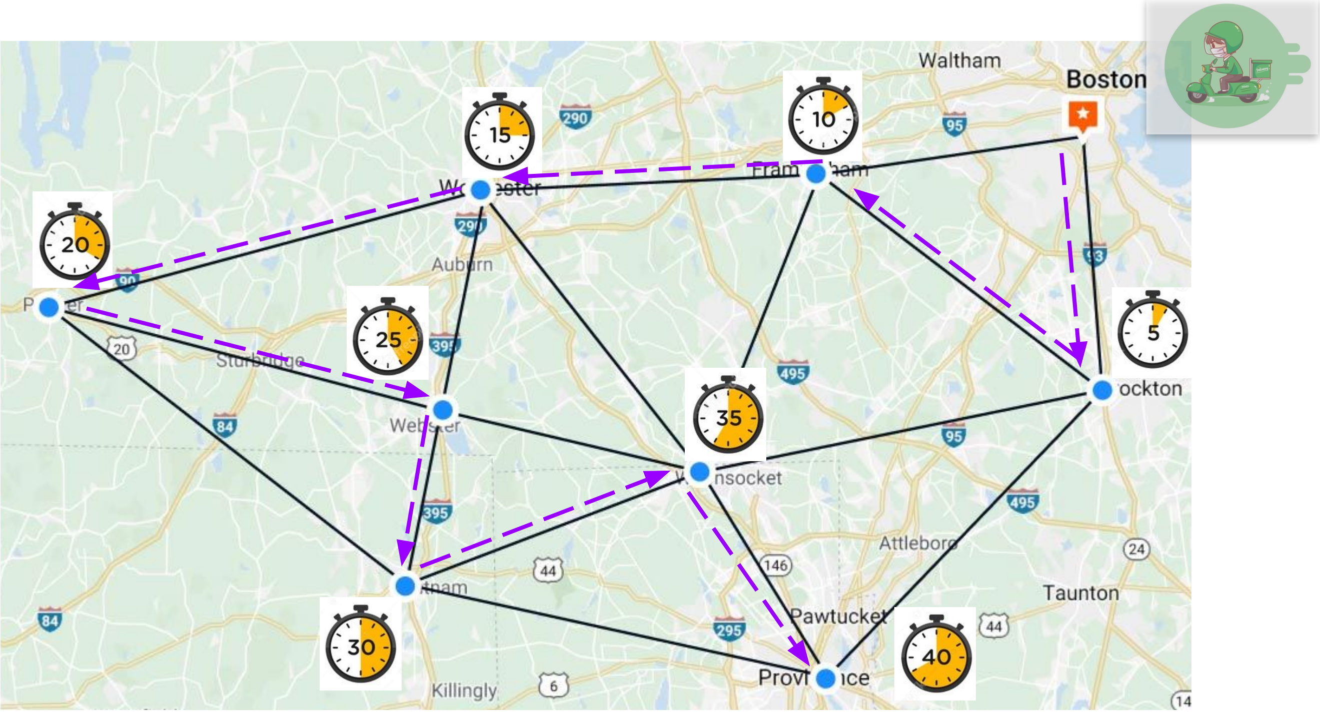
Typically in Traveling Salesman Problem ...





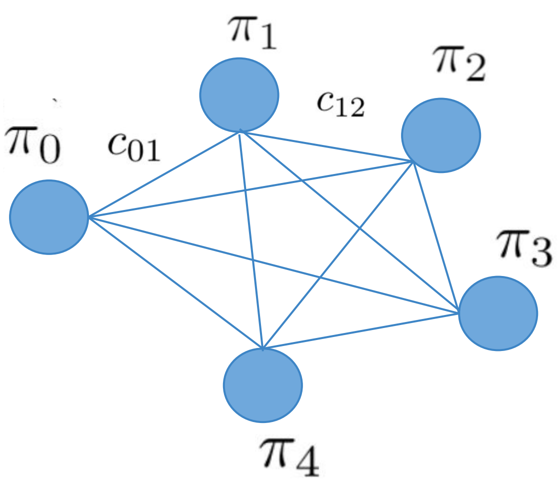
Minimize the total travel time of the delivery person

What if we want to be more customer-oriented ?



Minimize the total latency experienced at every node

Minimum Latency Problem (MLP)



Graph: $G = (V, E)$

Cost: $c(\pi_i, \pi_j)$ between every pair of nodes

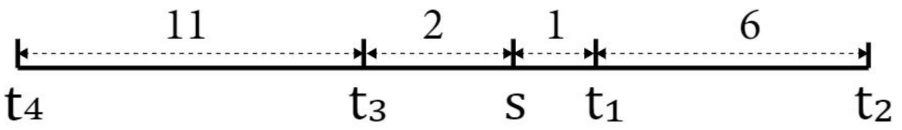
Path: $\pi = \{\pi_0, \pi_1, \pi_2, ...\pi_n\}$

MLP Objective

$$\min_{\pi} \underbrace{\sum_{i=1}^{n} \overbrace{\sum_{j=0}^{i-1} c(\pi_j, \pi_{j+1})}^{\text{Latency at node i}}}_{\text{Total latency except the starting node}}$$

Can a TSP solution solve MLP too?

Example in 1-d



An optimal TSP route

$s \rightarrow t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow t_4$

has a total latency of

$1 + 7 + 16 + 27 = 51$

The best MLP route

$s \rightarrow t_3 \rightarrow t_1 \rightarrow t_2 \rightarrow t_4$

has a total latency of

$2 + 5 + 11 + 31 = 49$

Early decisions can have a significant impact on overall cost (because the latency adds up)

No way to decompose the problem into smaller subproblems easily

Main Contributions

- We apply reinforcement learning and attention-based graph neural networks to solve the NP-hard minimum latency problem.

- Our solution are on par with domain-specific, hand-engineered solutions from literature.

Related Works

- GILS-RVND
- RL for CO
- GNN

History of Problem Formulations

Delivery man problem [1]

- Symmetric Graph

Traveling repairman problem [2]

- Each node also takes some time to repair
- Asymmetric Graph (by adding repair time of the node to the travel time of all outgoing edges)

Exact / Approximate / Heuristic Solutions

Exact Solutions

- **Integer Programming Formulations** [1, 2, 3] and solve with CPLEX
- Branch-cut-price [4] can solve 106 nodes (largest graph with optimal solution)

Approximate Solutions

- Blum et.al [5] gives a polynomial time, 72-approximation ratio algorithm
- Chaudhri et.al [6] gives a 3.59-approximation ratio algorithm

Heuristics

- **GILS-RVND** [7] can give high-quality solutions for up-to 1000 nodes

GILS-RVND

For i = 1 … M:

Construct an initial solution with **GRASP** (Greedy Randomized Adaptive Search Procedures)

do

Improve the solution with **RVND** (Randomized-order Variable Neighborhood Descent)

Update the best seen solution if possible

Perturb the current solution locally

until **ILS** (iterative local search) has not improved the best seen solution for N steps

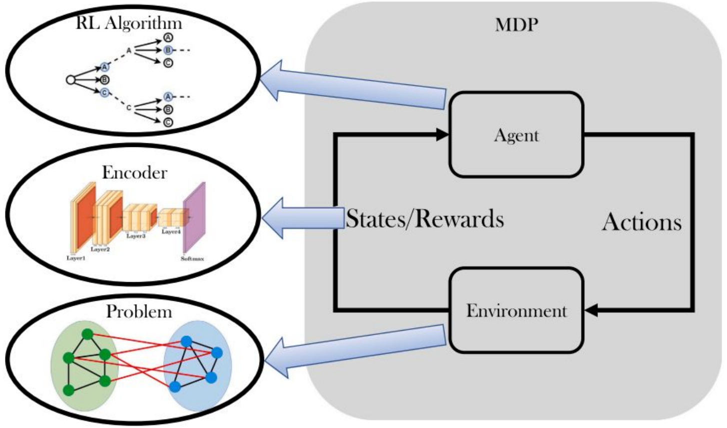
RL for combinatorial optimization

Formulate CO as an MDP and

- Learn a construction heuristic
- Learn an improvement heuristic
- Learn a branching policy in branch-and-bound
- ...

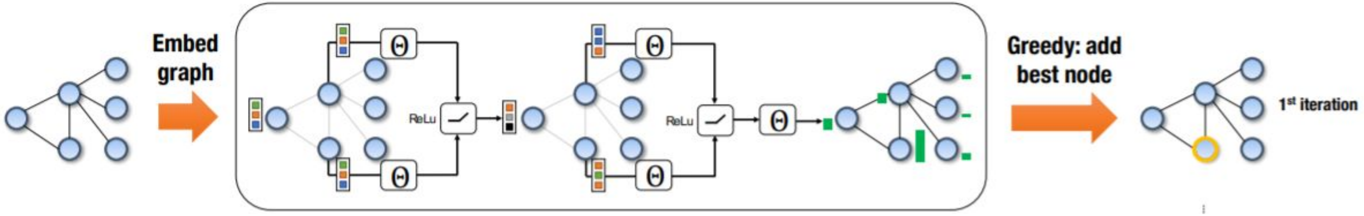
RL Training (see survey [8])

- Value-based (i.e. Q-learning, DQN)
- Policy-based (i.e. REINFORCE, PPO, A3C,)
- MCTS



How to encode graph problem structure?

- Key idea:
  - **Learn a representation at each node that encodes crucial graph structure for the CO problem**
  - Scale linearly with # nodes and # edges
- Structure2Vec (S2V) in Khalil et al. [9]



- Survey Paper from Cappart et.al. [10]
- Attention-based encoder-decoder in Kool et al. [11]

Methodology

- Graph Attention Network
- Optimization
- Implementation Details

Problem Formulation

Given:

- Problem instance *s* with a fixed starting node and *n* nodes to visit (each specified by 2d-coordinates)

Goal:

- Construct a tour through all graph nodes $\pi = \{\pi_0, \pi_1, \pi_2, \ldots , \pi_n\}$ (π_0 fixed) using a stochastic policy $p_\theta(\pi|s) = \prod_{t=1..n} p_\theta(\pi_t|s,\pi_{0:t-1})$ with parameters θ

Problem Formulation

Goal:

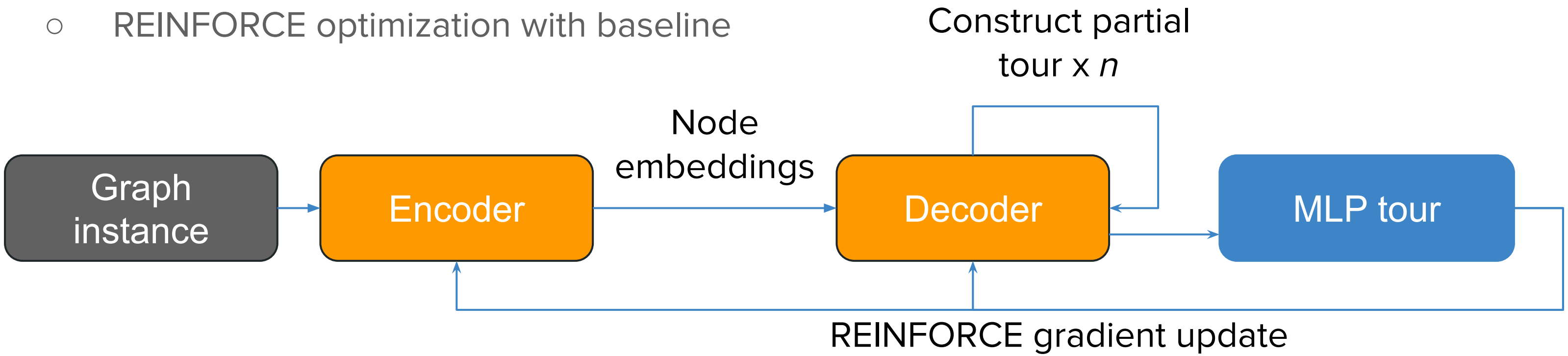
- Construct a tour through all graph nodes $\pi = \{\pi_0, \pi_1, \pi_2, \dots, \pi_n\}$ (π_0 fixed) using a stochastic policy $p_\theta(\pi|s) = \prod_{t=1..n} p_\theta(\pi_t|s, \pi_{0:t-1})$ with parameters θ

MDP formulation (for each of the *n* timesteps *t*):

- State: the partial tour constructed $\pi_{0:t-1}$
- Actions: select an unvisited node π_t
- Reward: negative cost of partial tour $\pi_{0:t}$
- Discount factor: 1

Method Overview

- To encode the node selection policy $p_\theta(\pi|s)$, adapt the encoder-decoder based graph attention network implemented by Kool et al. [11]
  - Effectiveness already demonstrated on multiple routing problems such as TSP
  - Previously tested on graphs with up to 100 nodes
- This model is autoregressive, so outputs can be conditioned on partial tours
- Main components:
  - Attention-based Encoder & Decoder
  - REINFORCE optimization with baseline



Attention-based Encoder

- All node coordinates $\mathbf{x}_i$ are embedded to 128-d $\mathbf{h}_i^{(N)}$ using $N=3$ attention layers
  - Each layer consists of a multi-headed attention (MHA) and feed-forward (FF) sublayer
  - Each MHA has $M=8$ attention heads, and FF layer has hidden dimension 512 & RELU activation
- Graph embedding $\mathbf{h}_{(g)}^{(N)}$ is computed as mean of all $\mathbf{h}_i^{(N)}$

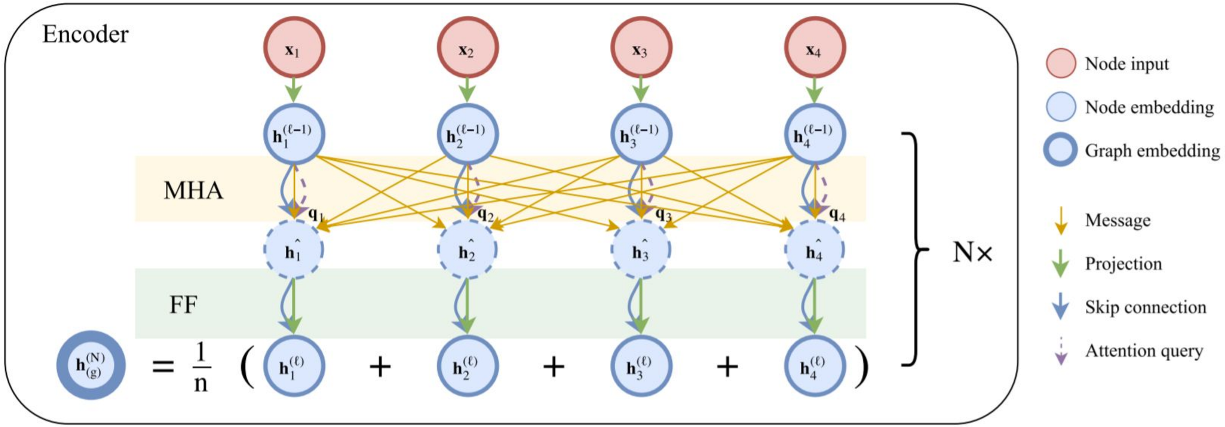


Image source: [11]

Attention-based Decoder

- At each time step $t = 1..n$, the decoder computes $p_\theta(\pi_t = i \mid s, \pi_{0:t-1})$ $\forall\ i \in \{1, \ldots, n\}$
  - A context embedding $\mathbf{h}_{(c)}^{(N)} = [\mathbf{h}_{(g)}^{(N)}, \mathbf{h}_{t-1}^{(N)}]$ and all node embeddings $\mathbf{h}_i^{(N)}$ are inputted to a MHA layer which computes a new context node embedding $\mathbf{h}_{(c)}^{(N+1)}$
  - A single attention head + softmax layer computes compatibility with all unvisited nodes
- Visited nodes are masked out

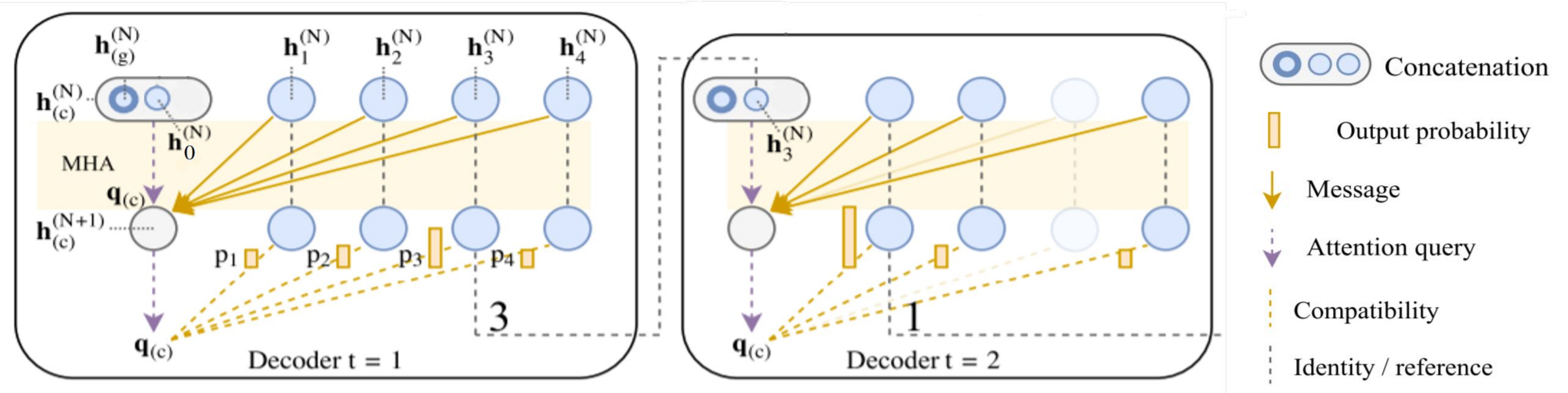


Image source: [11]

Decoding Methods

- Given $p_\theta(\pi_t = i \mid s, \pi_{0:t-1})$ at each time step t, either greedy or sampling-based decoding can be used:
  - **Greedy decoding**: select the node i with the highest probability
  - **Sampling-based decoding**: randomly select a node using the given probability distribution
- Trade-off between runtime and solution quality:
  - Greedy is faster as it only produces 1 solution of reasonable quality
  - Sampling can be used to sample W solutions and select the best (slower but higher quality)
    - Following [11], we use $W = 1280$

Policy Optimization

- Given the policy $p(\pi|s) = p_\Theta(\pi|s)$ and the cost of the sampled MLP tour $L(\pi)$, the loss function is

$$\mathcal{L}(\Theta|s) = \mathbb{E}_{p(\pi|s)}[L(\pi)]$$

- To optimize $\mathcal{L}(\Theta|s)$, use grad. descent on the REINFORCE grad. estimate [12]

$$\nabla\mathcal{L}(\Theta|s) = \mathbb{E}_{p(\pi|s)}[(L(\pi) - b(s))\nabla log(p(\pi|s))]$$

- *b(s)* is a greedy baseline (tour cost from best greedy policy so far)
- After each epoch, the baseline is replaced by the training policy if there is significant improvement (according to a t-test over 10k instances)

Implementation Details

- Used the ADAM optimizer with a constant learning rate of 10^{-4}
  - Trained the policy using 1 GPU with a batch size of 1024
  - Trained for 140 epochs (except for 100-node graphs, trained for 200 epochs)
- Used CPU during test time for fair comparison to other methods
- The code is largely based on that of Kool et al. [M1] with modifications to problem environment, loss function, decoder context, and data generation.

Implementation Details

- Used 1 GPU for training and multiple CPUs during test time for fair comparison to other methods
- The code is largely based on that of Kool et al. [11] with modifications to the loss function, decoder context, and data generation

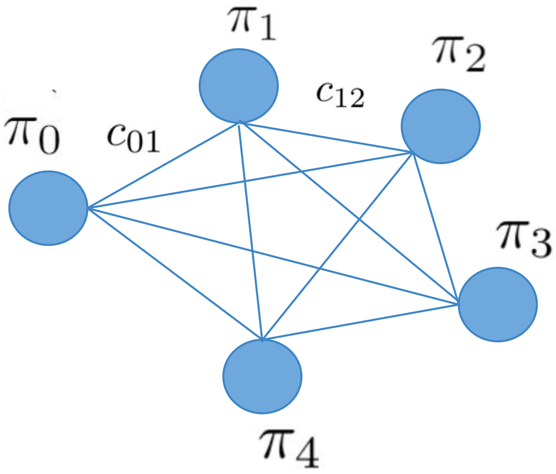
Experimental Setup

- Dataset
- Baseline Methods
- Evaluation Metrics

Dataset

2-D synthetic datasets

- Locations randomly sampled from $[0, 1]^2$
- Cost matrix $C = (c_{ij})$ defines the edge costs



Service time for node i

$$c_{ij} = t_{ij} + s_i$$

Euclidean distance between node i and node j

$$s_i = 0$$ symmetric

Dataset

Three classes [3]:

S0: $s_i = 0$ symmetric

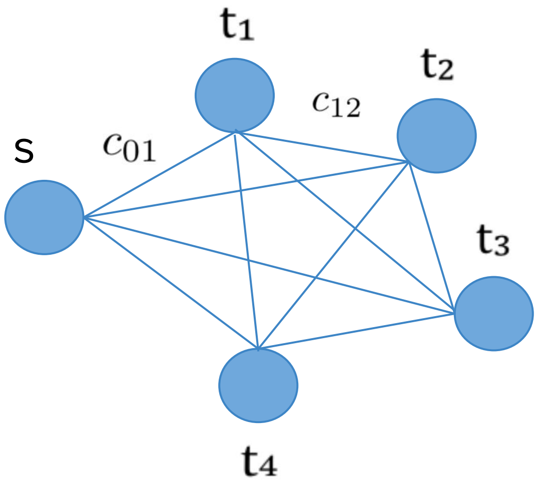
S1: $s_i \sim [0, \frac{t_{max} - t_{min}}{2}]$

S2: $s_i \sim [\frac{t_{max} + t_{min}}{2}, \frac{3t_{max} - t_{min}}{2}]$

asymmetric

2-D synthetic datasets

- Locations randomly sampled from $[0, 1]^2$
- Cost matrix $C = (c_{ij})$ defines the edge costs



Service time for node i

$$c_{ij} = t_{ij} + s_i$$

$s_i = 0$ for starting node

Euclidean distance between node i and node j

Baseline Methods

- **Exact**: CPLEX MIP [3]
    - Solve for small graphs with up to 30 nodes
    - Default CPLEX setting
    - 2 hour limit

Baseline Methods

- **Exact**: CPLEX MIP [3]
  - Solve for small graphs with up to 30 nodes
  - Default CPLEX setting
  - 2 hour limit
- **Heuristics**:
  - Nearest Neighbor (NN) -- greedy
  - Nearest Neighbor-softmax (NN-softmax) -- sampling-based
  - GILS-RVND [9] -- **state-of-the-art** MLP heuristic

Evaluation Metrics

Symmetric graphs with N = 10, 20, 30, 50, 100
25 test instances per graph size

Greedy construction methods: RL + greedy decoding, Nearest Neighbor (NN)

Sampling-based methods: RL + sampling-based decoding, NN-softmax, GILS-RVND

Evaluation Metrics

Symmetric graphs with N = 10, 20, 30, 50, 100
25 test instances per graph size

Greedy construction methods: RL + greedy decoding, Nearest Neighbor (NN)

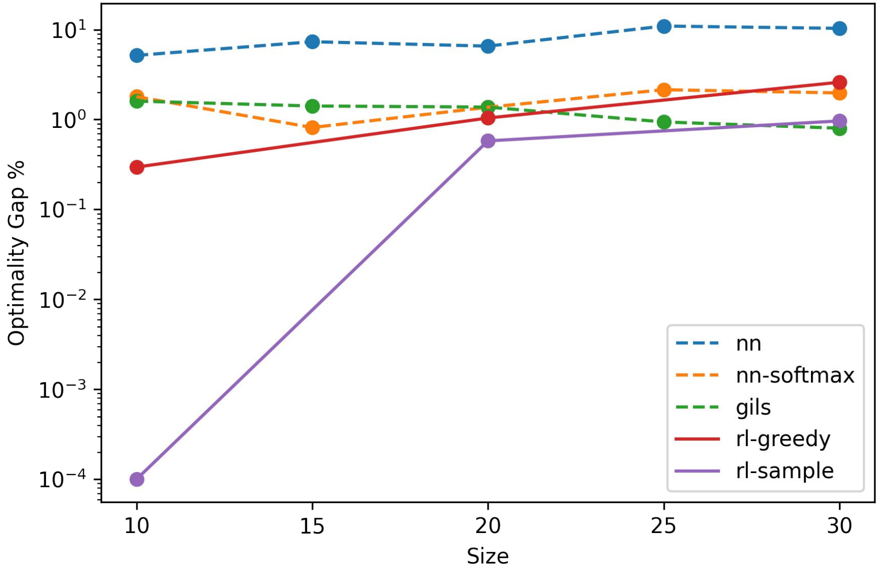
Sampling-based methods: RL + sampling-based decoding, NN-softmax, GILS-RVND

- Quality of solutions
  - Optimality gap for small graphs (up to 30 nodes)
  - Objective values of different heuristics for all graph sizes
- Runtime
- Generalization to different sizes

Experimental Results

- Optimality on Small Graphs
- Scaling to Large Graphs
- Generalization Over Graph Sizes

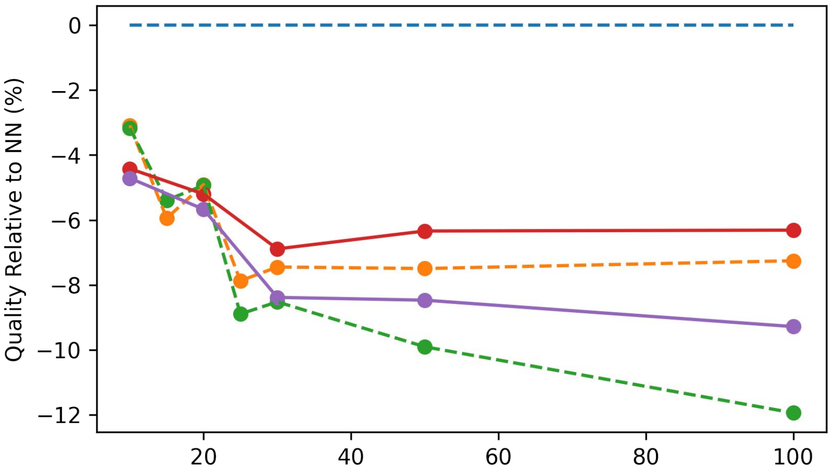
Optimality on Small Graphs

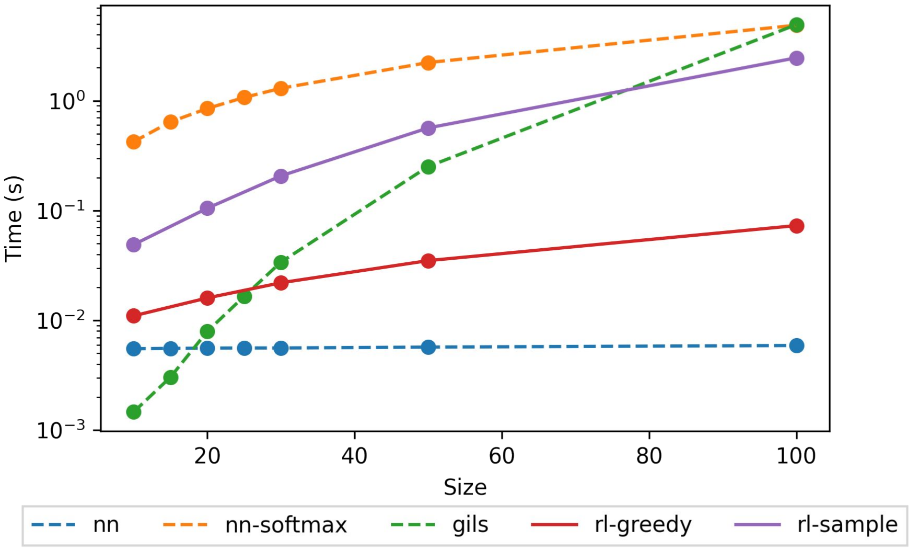


size	nn	rl-greedy	nn-softmax	gils	rl-sample
10	5.1664	**0.2960**	1.7978	1.6027	**0.0001**
15	7.3307	-	0.8155	1.4142	-
20	6.5527	**1.0409**	1.3706	1.3758	**0.5790**
25	10.9506	-	2.1460	0.9404	-
30	10.3219	**2.5990**	1.9768	**0.8012**	0.9650

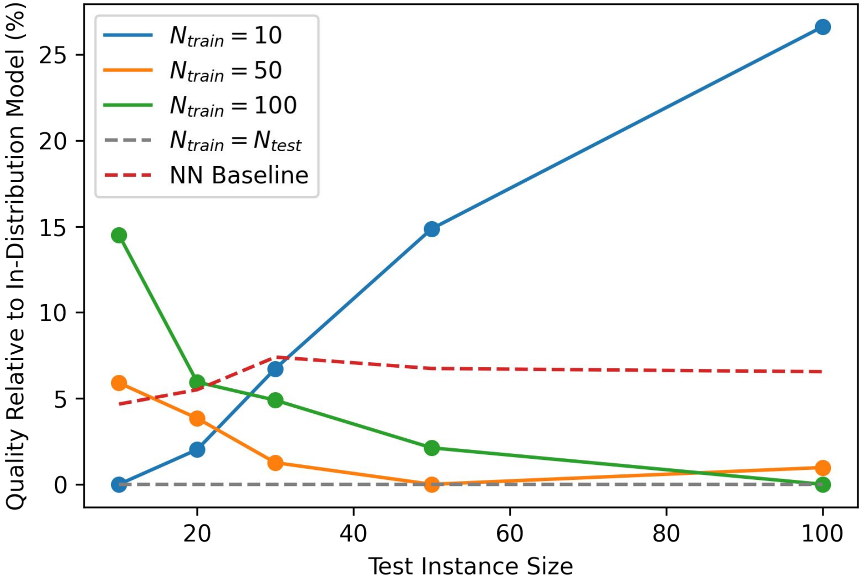
Scaling to Larger Graphs

size	nn cost	nn runtime	rl-greedy cost	rl-greedy runtime	nn-softmax cost	nn-softmax runtime	gils cost	gils runtime	rl-sample cost	rl-sample runtime
10	13.70	0.01	**13.09**	0.01	13.27	0.42	13.26	0.00	**13.05**	0.05
15	22.87	0.01	-	-	21.51	0.64	21.64	0.00	-	-
20	36.25	0.01	**34.36**	0.02	34.47	0.85	34.46	0.01	**34.19**	0.10
25	51.46	0.01	-	-	47.41	1.07	46.88	0.02	-	-
30	65.61	0.01	61.09	0.02	60.72	1.30	**60.02**	0.03	60.11	0.21
50	140.83	0.01	**131.90**	0.04	130.27	2.23	**126.88**	0.25	128.90	0.57
100	390.56	0.01	**365.90**	0.07	362.22	4.88	**343.92**	4.93	354.30	2.46



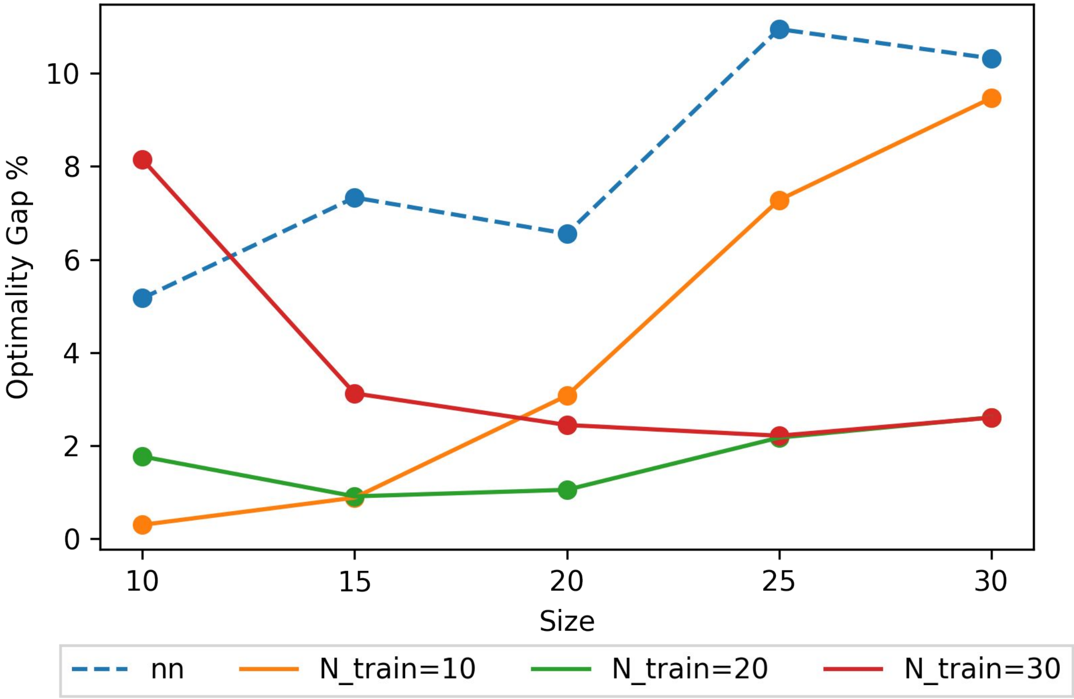


Generalization to Different Sizes



N_{test}	$N_{train} = 10$	$N_{train} = 50$	$N_{train} = 100$
10	**0.00**	5.90	14.50
20	**2.02**	3.84	5.95
30	6.73	**1.26**	4.89
50	14.86	**0.00**	2.12
100	26.61	0.98	**0.00**

Generalization to Different Sizes (Optimality)



Conclusion

- RL is a compelling approach for deriving construction heuristics for the Minimum Latency Problem
    - Competitive with hand-engineered approaches at low run-times

Next Steps

- Can the solutions constructed by RL be synergistically combined with local search methods (i.e. GILS) to produce even higher quality results?
- Evaluate on asymmetric graphs where service times are non-zero

Thank you for listening!

Questions?

References

[1] F. Angel-Bello, A. Alvarez, and I. García, "Two improved formulations for the minimum latency problem,"Applied Mathematical Modelling, vol. 37, no. 4, pp. 2257–2266, 2013.

[2] van Ca Cleola Eijl, "A polyhedral approach to the delivery man problem," 1995.

[3] I. Méndez-Díaz, P. Zabala, and A. Lucena, "A new formulation for the traveling deliveryman problem,"Discrete applied mathematics, vol. 156, no. 17, pp. 3223–3237, 2008.

[4] H. Abeledo, R. Fukasawa, A. Pessoa, and E. Uchoa, "The time dependent traveling salesman problem: polyhedra and algorithm,"Mathematical Programming Computation, vol. 5, no. 1, pp.27–55, 2013.

[5] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, and M. Sudan, "The minimum latency problem," inProceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, ser. STOC '94.New York, NY, USA: Association for Computing Machinery, 1994, p. 163–171.

[6] K. Chaudhuri, B. Godfrey, S. Rao, and K. Talwar, "Paths, trees, and minimum latency tours," in FOCS '03: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science.IEEE, January 2003, p. 36.

[7] M. M. Silva, A. Subramanian, T. Vidal, and L. S. Ochi, "A simple and effective metaheuristic for the minimum latency problem", European Journal of Operational Research, vol. 221, no. 3, pp. 513–520, 2012.

[8] N. Mazyavkina, S. Sviridov, S. Ivanov, and E. Burnaev, "Reinforcement learning for combinatorial optimization: A survey,"Computers Operations Research, vol. 134,p. 105400, 2021

[9] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs,"

[10] Q. Cappart, D. Chételat, E. B. Khalil, A. Lodi, C. Morris, and P. Veliˇckovi´c, "Combinatorial optimization and reasoning with graph neural networks," 2021.

[11] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!" 2019.

[12] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," Mach. Learn., vol. 8, no. 3–4, p. 229–256, May 1992. [Online]. Available: https://doi.org/10.1007/BF00992696