Graph Neural Networks for the Travelling Salesman Problem

Chaitanya K. Joshi [1], Thomas Laurent [2], and Xavier Bresson [1]

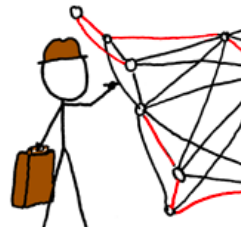
[1] NTU, Singapore, [2] LMU, LA, USA

Boosting Combinatorial Optimization using Machine Learning

(Session at the INFORMS Annual Meeting 2019)

22nd October, 2019

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE



Motivation

- **Operations Research**: solvers for NP-Hard combinatorial problems
  - Backbone of modern industries such as transportation, scheduling, logistics

- **Good OR solvers**
  - expert intuition/domain knowledge
  - years of trial-and-error

Motivation

- **Operations R**: solvers for NP-Hard combinatorial problems
  - Backbone o uling, logistics
- **Good OR solver**
  - expert intuition/domain knowledge
  - years of trial-and-error

*“ We believe that expert intuition can be **automated** and **augmented** through Machine Learning ”*

- Bengio, Lodi, Prouvost, 2018 [1]

[1] Bengio, Lodi, Prouvost, Machine learning for combinatorial optimization: a methodological tour d’horizon, 2018

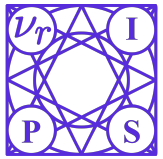
This talk

- Advances in ***end-to-end*** **learning** for OR solvers
  - Results on TSP: intensively studied, practical class of routing problems

- Our focus/specialty: **Graph Neural Networks**
  - New tools for operating directly on the graph structure of problems

Our contributions

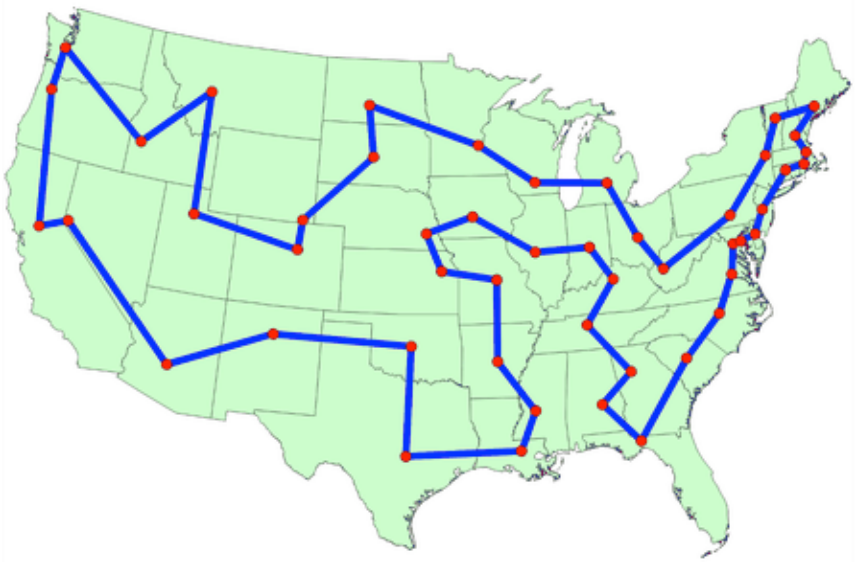
An Efficient Graph ConvNet for TSP: arxiv.org/abs/1906.01227



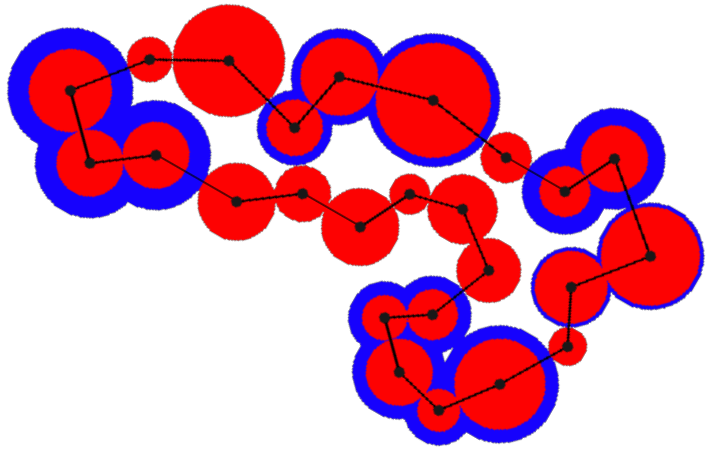
On Learning Paradigms for TSP: arxiv.org/abs/1910.07210

TSP as a graph problem

"Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?"



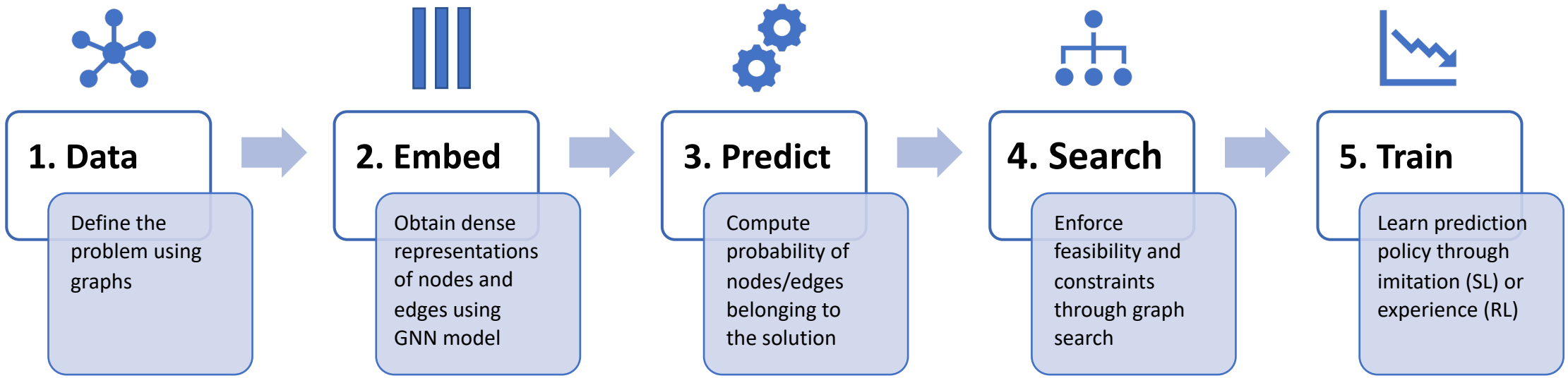
- Concorde Solver [1]: leverages 30 years of research
  - **Cutting plane algorithms** to iteratively solve linear relaxations
  - **Branch-and-bound** to reduce solution search space



- End-to-end learning for TSP [2,3]: Proof-of-concept for learning **previously un-encountered** NP-Hard problems

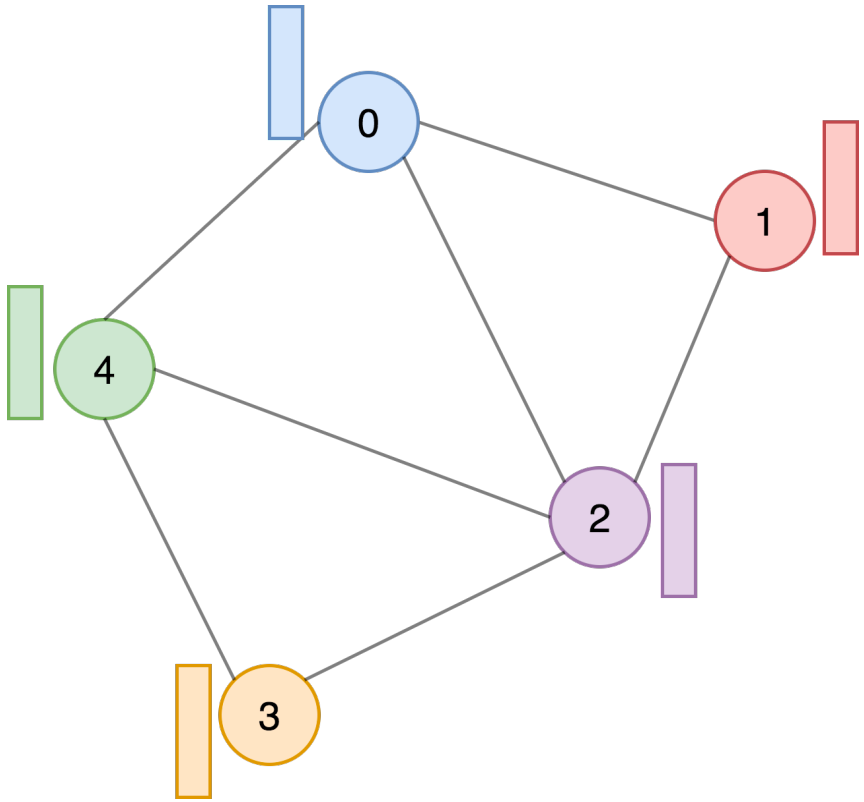
[1] Applegate, Bixby, Chvátal, Cook, The traveling salesman problem: a computational study, 2006
[2] Vinyals, Fortunato, Jaitly, Pointer networks, NeurIPS 2015
[3] Bello, Pham, Le, Norouzi, Bengio, Neural combinatorial optimization with reinforcement learning, ICLR 2017

End-to-end pipeline for OR problems

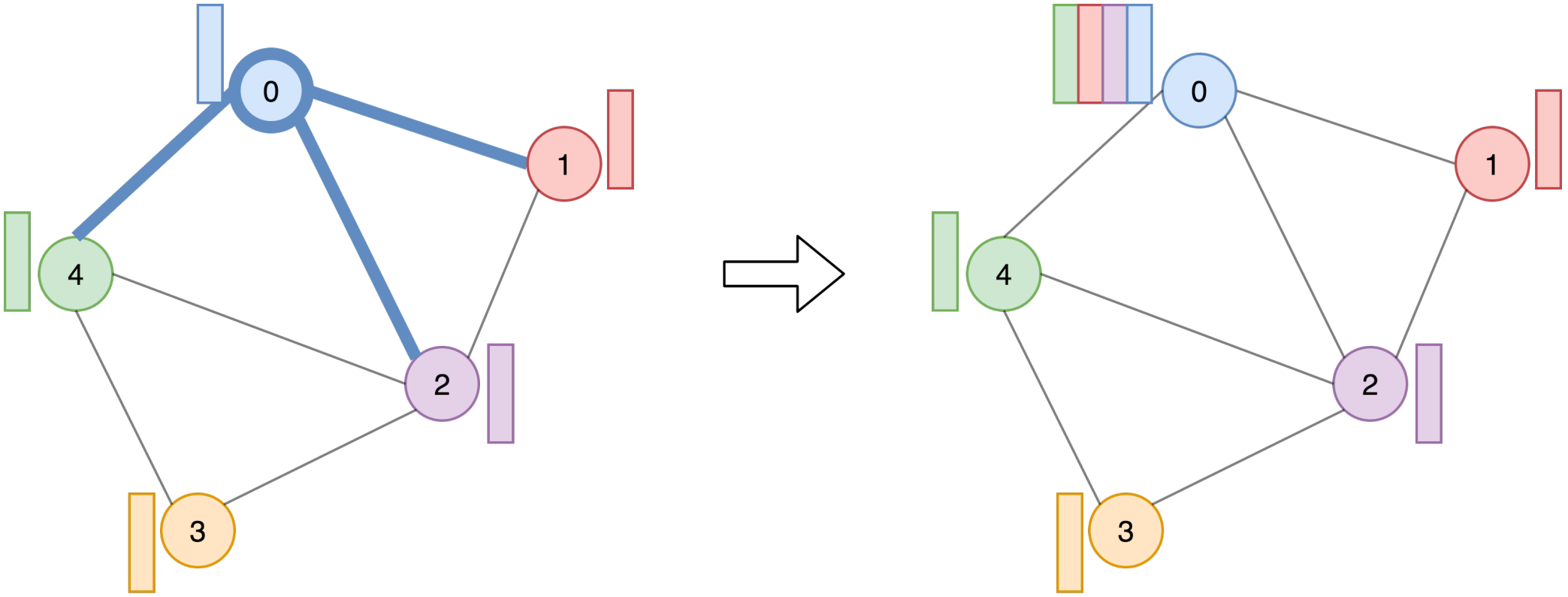


This generic pipeline has been used to tackle TSP, MVC, MaxCut, MIS, VRPs, SAT, etc.

Graph Embedding: features



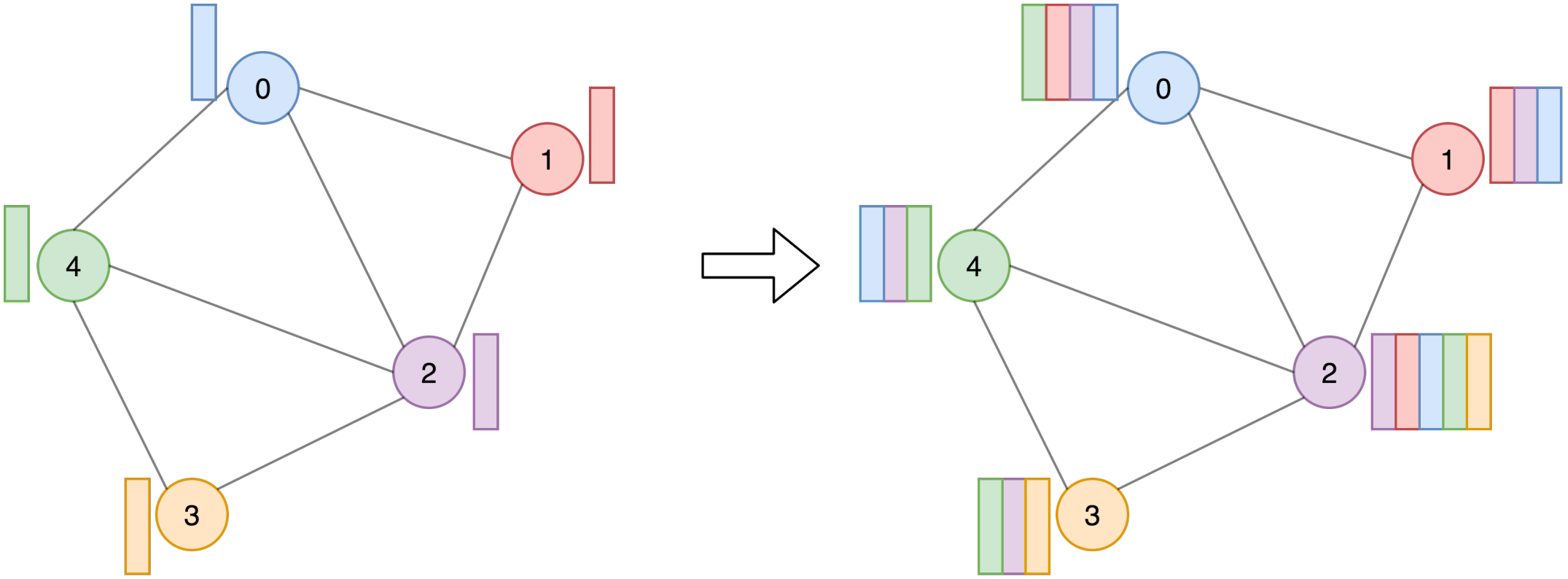
Graph Embedding: message passing



Layer 1

Layer 2

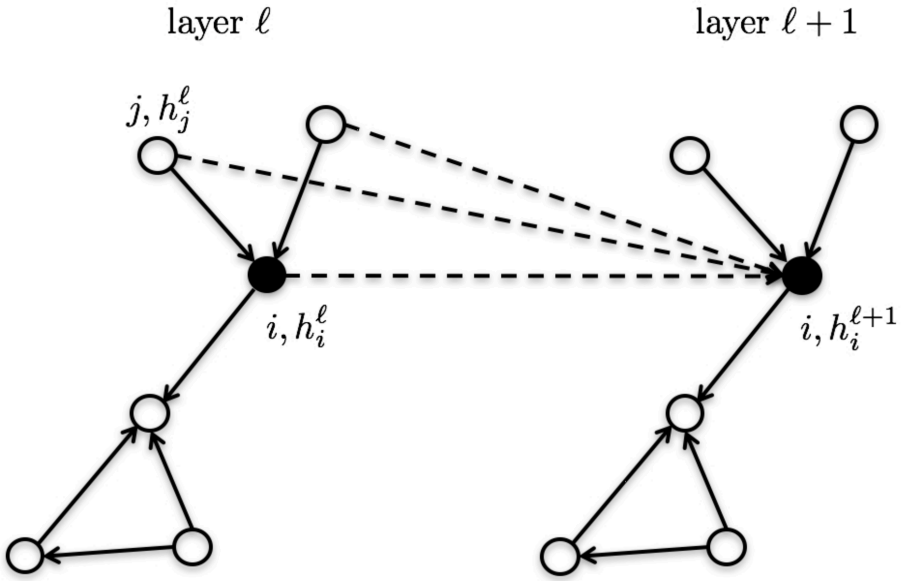
Graph Embedding: aggregation



Layer 1

Layer 2

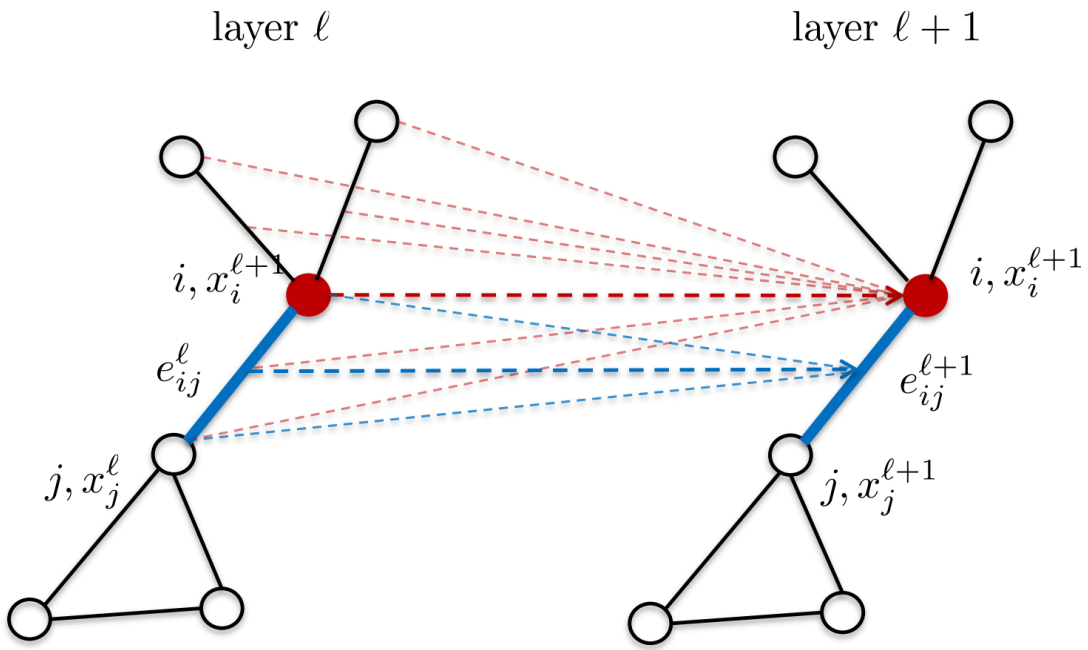
Vanilla Graph ConvNets [1,2]



$$h_i^{\ell+1} = f_{\text{G-VCNN}}^{\ell}\left(\, h_i^{\ell}\,,\, \{h_j^{\ell} : j \to i\}\,\right) = \text{ReLU}\left(U^{\ell} h_i^{\ell} + V^{\ell} \sum_{j \to i} h_j^{\ell}\right)$$

[1] Sukhbaatar, Szlam, Fergus, Learning multiagent communication with backpropagation, NeurIPS 2016
[2] Kipf, Welling, Semi-supervised classification with graph convolutional networks, ICLR 2017

Residual Gated Graph ConvNets [1,2]



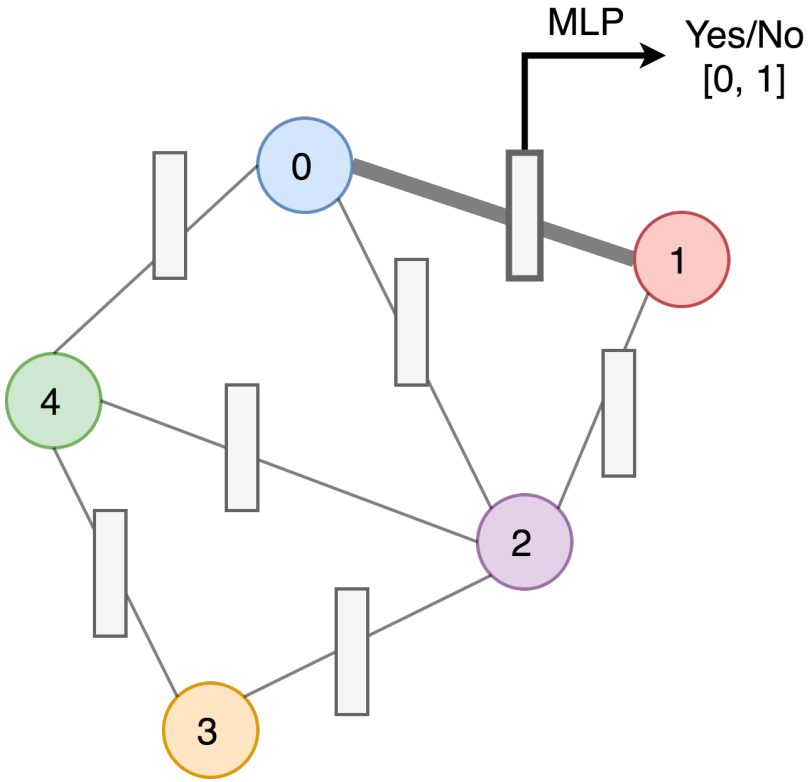
$$x_i^{\ell+1} = x_i^{\ell} + \mathrm{ReLU}\Big(\mathrm{BN}\Big(W_1^{\ell} x_i^{\ell} + \sum_{j\sim i} \eta_{ij}^{\ell} \odot W_2^{\ell} x_j^{\ell}\Big)\Big)$$

$$\eta_{ij}^{\ell} = \frac{\sigma(e_{ij}^{\ell})}{\sum_{j'\sim i} \sigma(e_{ij'}^{\ell}) + \varepsilon}$$

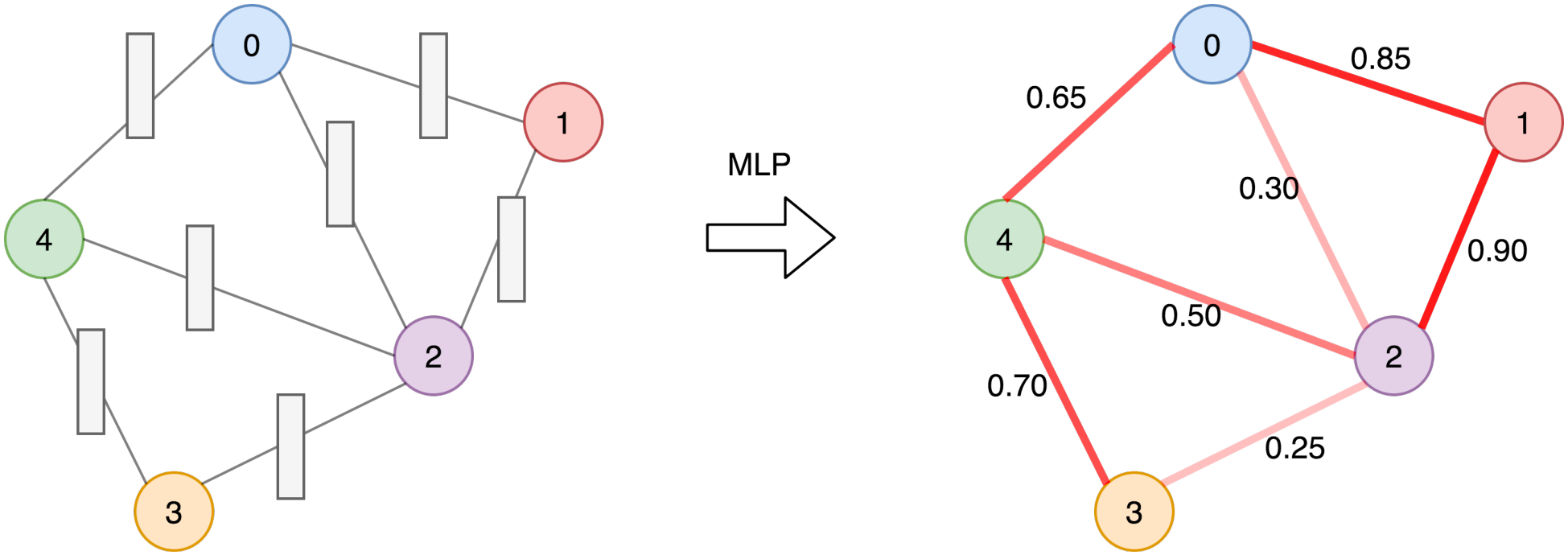
$$e_{ij}^{\ell+1} = e_{ij}^{\ell} + \mathrm{ReLU}\Big(\mathrm{BN}\Big(V_1^{\ell} e_{ij}^{\ell} + V_2^{\ell} x_i^{\ell} + V_3^{\ell} x_j^{\ell}\Big)\Big)$$

[1] Bresson, Laurent, Residual gated graph convnets, ICLR 2018
[2] Joshi, Laurent, Bresson, An efficient graph convolutional network technique for the travelling salesman problem, arXiv 2019

Prediction: does an edge belong to the optimal tour?



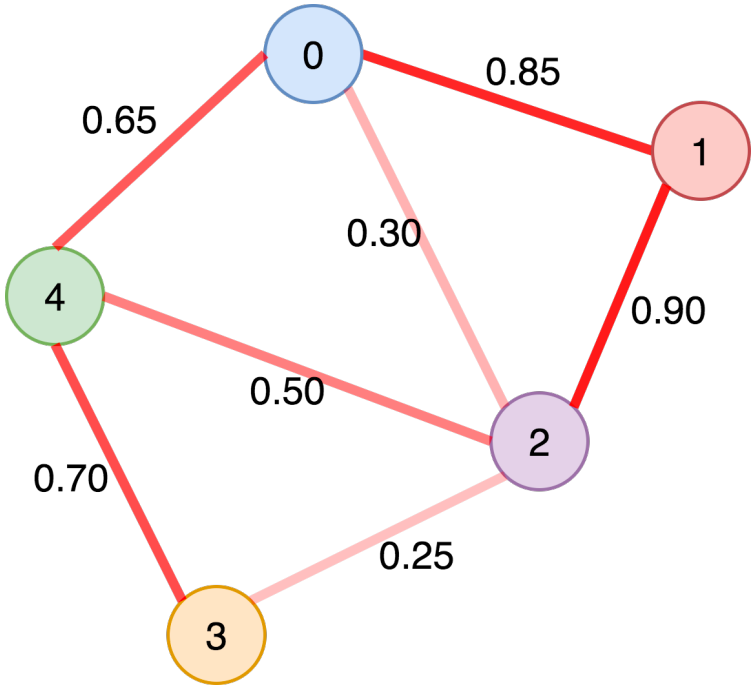
Prediction: probability distribution over edges



Prediction: Non-autoregressive approach [1]

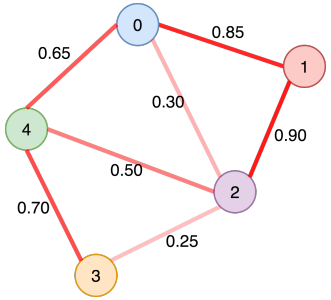
Predictions for all edges are

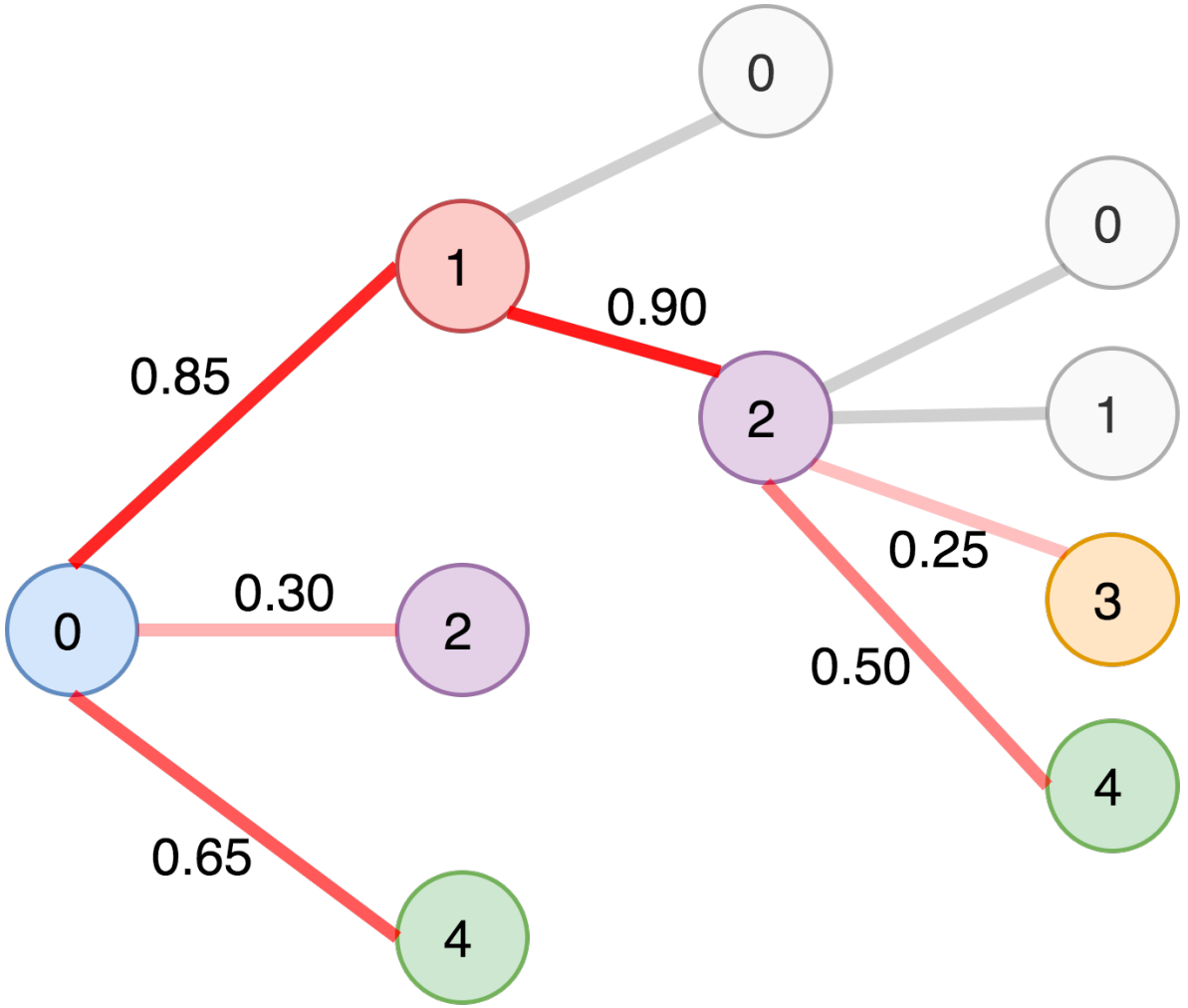
- produced in **one shot**
- **independent** of each other



[1] Nowak, Villar, Bandeira, Bruna, A note on learning algorithms for quadratic assignment with graph neural networks, arXiv 2017

Search for feasible solutions

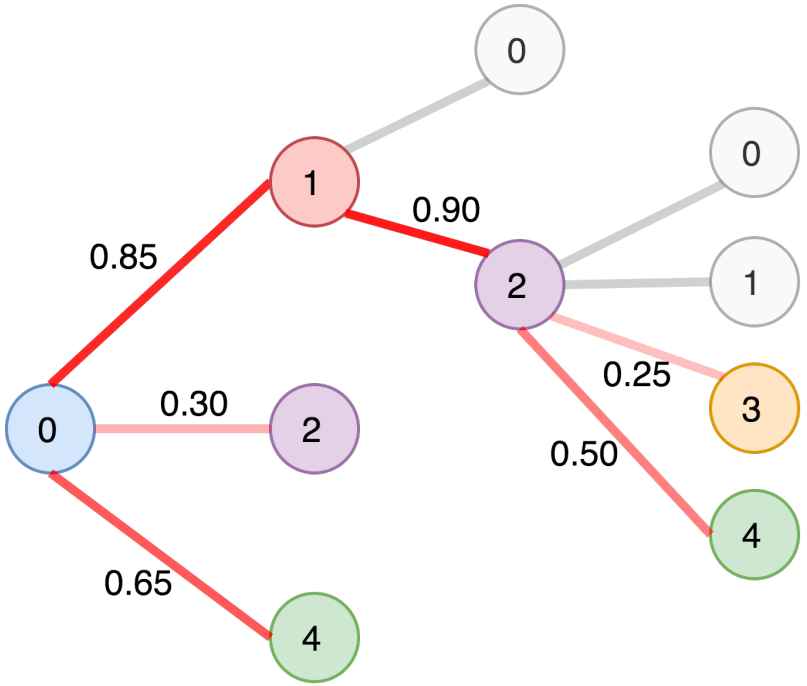




Search for feasible solutions

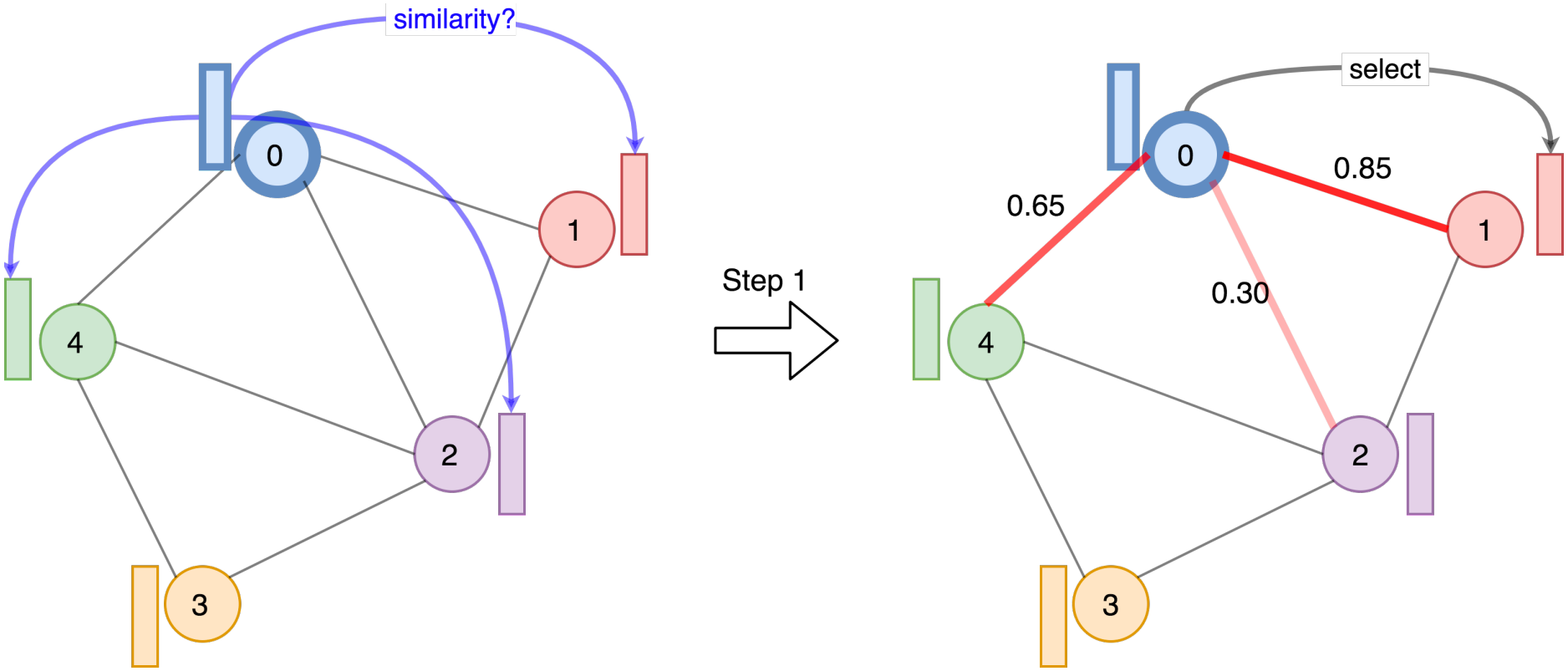
- We can use any search algorithm for graphs + enforce problem constraints:
  - Greedy search
  - Beam search
  - Monte Carlo tree search

- Analogous to search in **machine translation**[1] or **game playing**[2]

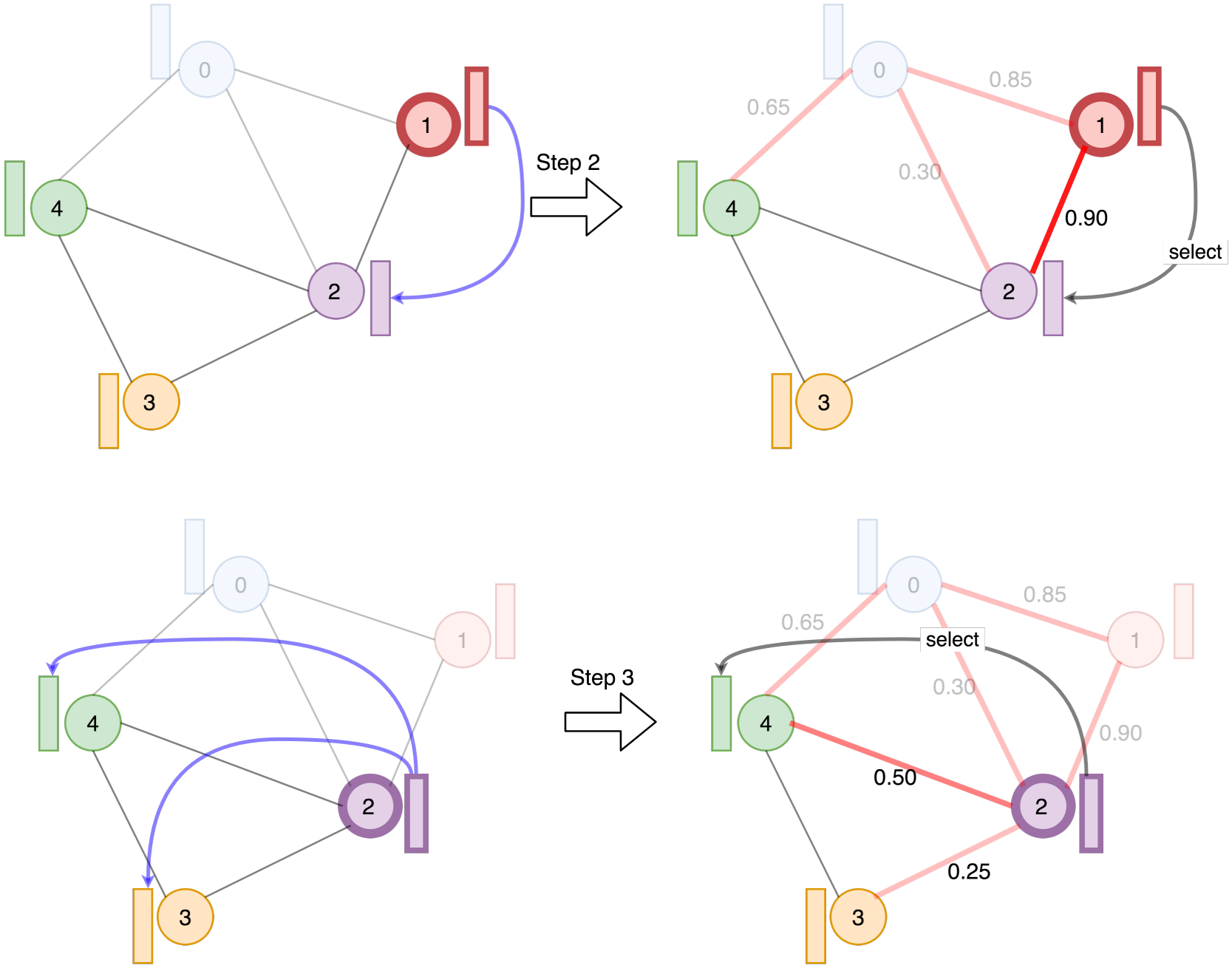


[1] Wu et al., Google's neural machine translation system, arXiv 2016
[2] Silver et al., Mastering the game of Go with deep neural networks and tree search, Nature 2016

Alternate: Autoregressive decoding[1] with Attention[2,3]



[1] Khalil, Dai, Zhang, Dilkina, Song, Learning combinatorial optimization algorithms over graphs, NeurIPS 2017
[2] Deudon, Cournut, Lacoste, Adulyasak, Rousseau, Learning heuristics for the tsp by policy gradient, 2018
[3] Kool, van Hoof, Welling, Attention, learn to solve routing problems!, ICLR 2019


0
1
4
2
3
Step 2
0.65
0.85
0.30
0.90
select
Step 3
0.50
0.25

Training the policy

Learning by **Imitation (SL)**

- Minimize the loss between optimal solutions (Concorde) and model's prediction
- Binary classification problem on edges

Learning by **Exploration (RL)**

- Use REINFORCE (policy gradient) to minimize the length of the tour predicted by the model at the end of decoding

And there are trade-offs for both…

Experiments

Current paradigm: Models are trained and evaluated on TSP instances of fixed sizes: 20, 50 and 100 nodes

Performance on fixed TSP

Method	Type	TSP20			TSP50			TSP100		
		Tour Len.	Opt. Gap.	Time	Tour Len.	Opt. Gap.	Time	Tour Len.	Opt. Gap.	Time
Concorde	Specific	3.84	0.00%	(1m)	5.70	0.00%	(2m)	7.76	0.00%	(3m)
Gurobi	Generic	3.84	0.00%	(7s)	5.70	0.00%	(2m)	7.76	0.00%	(17m)
Bello et al. (2016)	AR, RNN, RL	3.84	0.10%	–	5.75	0.95%	–	8.00	3.03%	–
Deudon et al. (2018)	AR, GAT, RL	3.84	0.11%	(5m)	5.77	1.28%	(17m)	8.75	12.70%	(56m)
Deudon et al. (2018)	(+ 2OPT)	3.84	0.09%	(6m)	5.75	1.00%	(32m)	8.12	4.64%	(5h)
Kool et al. (2019)	AR, GAT, RL	3.84	0.08%	(5m)	5.73	0.52%	(24m)	7.94	2.26%	(1h)
Ours (arXiv '19)	NAR, GCN, SL	**3.84**	**0.10%**	(20s)	**5.71**	**0.26%**	(2m)	**7.92**	**2.06%**	(10m)
Ours (arXiv '19)	(+ shortest h.)	**3.84**	**0.01%**	(12m)	**5.70**	**0.01%**	(18m)	**7.87**	**1.39%**	(40m)

End-to-end solvers can't compete with OR solvers yet, but…

Performance: Supervised learning?

Speed: Non-autoregressive?

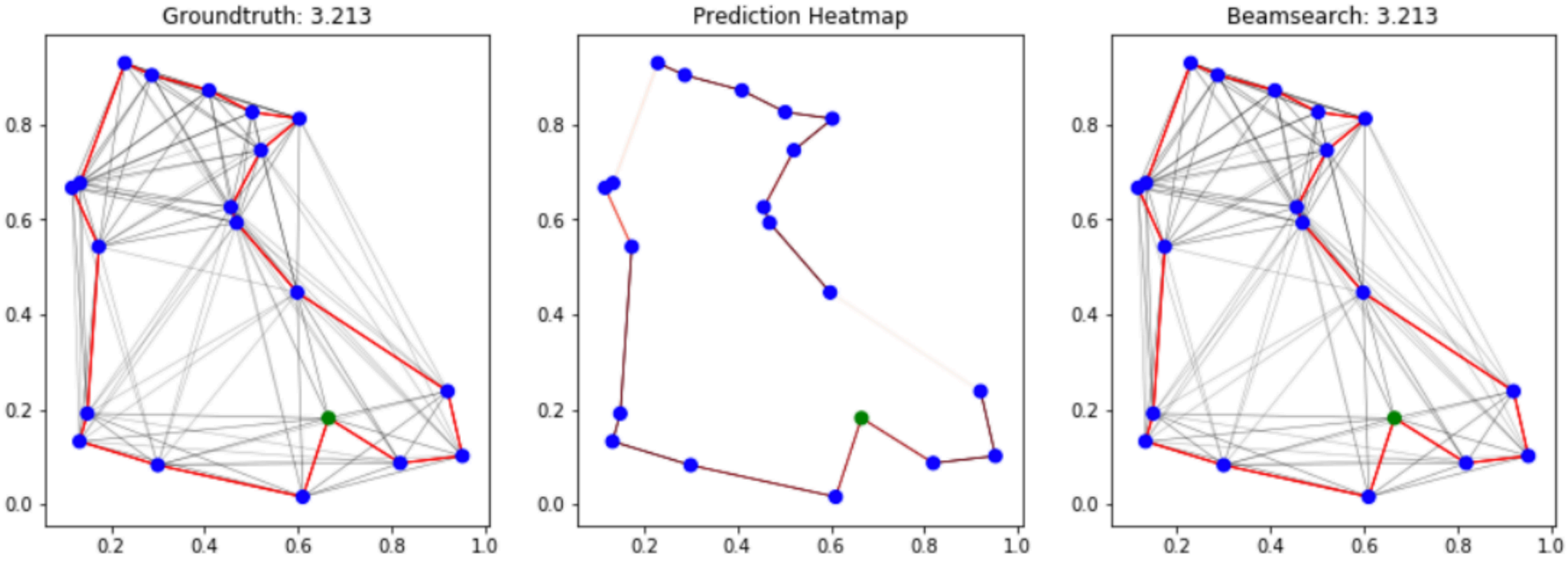
Performance on fixed TSP

Method	Type	TSP20 Tour Len.	TSP20 Opt. Gap.	TSP20 Time	TSP50 Tour Len.	TSP50 Opt. Gap.	TSP50 Time	TSP100 Tour Len.	TSP100 Opt. Gap.	TSP100 Time
Concorde	Specific	3.84	0.00%	(1m)	5.70	0.00%	(2m)	7.76	0.00%	(3m)
Gurobi	Generic	3.84	0.00%	(7s)	5.70	0.00%	(2m)	7.76	0.00%	(17m)
Bello et al. (2016)	AR, RNN, RL	3.84	0.10%	–	5.75	0.95%	–	8.00	3.03%	–
Deudon et al. (2018)	AR, GAT, RL	3.84	0.11%	(5m)	5.77	1.28%	(17m)	8.75	12.70%	(56m)
Deudon et al. (2018)	(+ 2OPT)	3.84	0.09%	(6m)	5.75	1.00%	(32m)	8.12	4.64%	(5h)
Kool et al. (2019)	AR, GAT, RL	3.84	0.08%	(5m)	5.73	0.52%	(24m)	7.94	2.26%	(1h)
Ours (arXiv '19)	NAR, GCN, SL	**3.84**	**0.10%**	(20s)	**5.71**	**0.26%**	(2m)	**7.92**	**2.06%**	(10m)
Ours (arXiv '19)	(+ shortest h.)	**3.84**	**0.01%**	(12m)	**5.70**	**0.01%**	(18m)	**7.87**	**1.39%**	(40m)
Ours (NeurIPS '19)	AR, GAT, SL	3.84	0.01%	(5m)	5.70	0.04%	(24m)	7.86	1.25%	(1h)
Ours (NeurIPS '19)	AR, GAT, RL	3.84	0.09%	(5m)	5.72	0.59%	(24m)	7.97	2.68%	(1h)
Ours (NeurIPS '19)	AR, GCN, SL	3.84	0.01%	(5m)	5.72	0.27%	(24m)	7.96	2.62%	(1h)
Ours (NeurIPS '19)	AR, GCN, RL	3.84	0.10%	(5m)	5.73	0.76%	(24m)	7.97	2.66%	(1h)

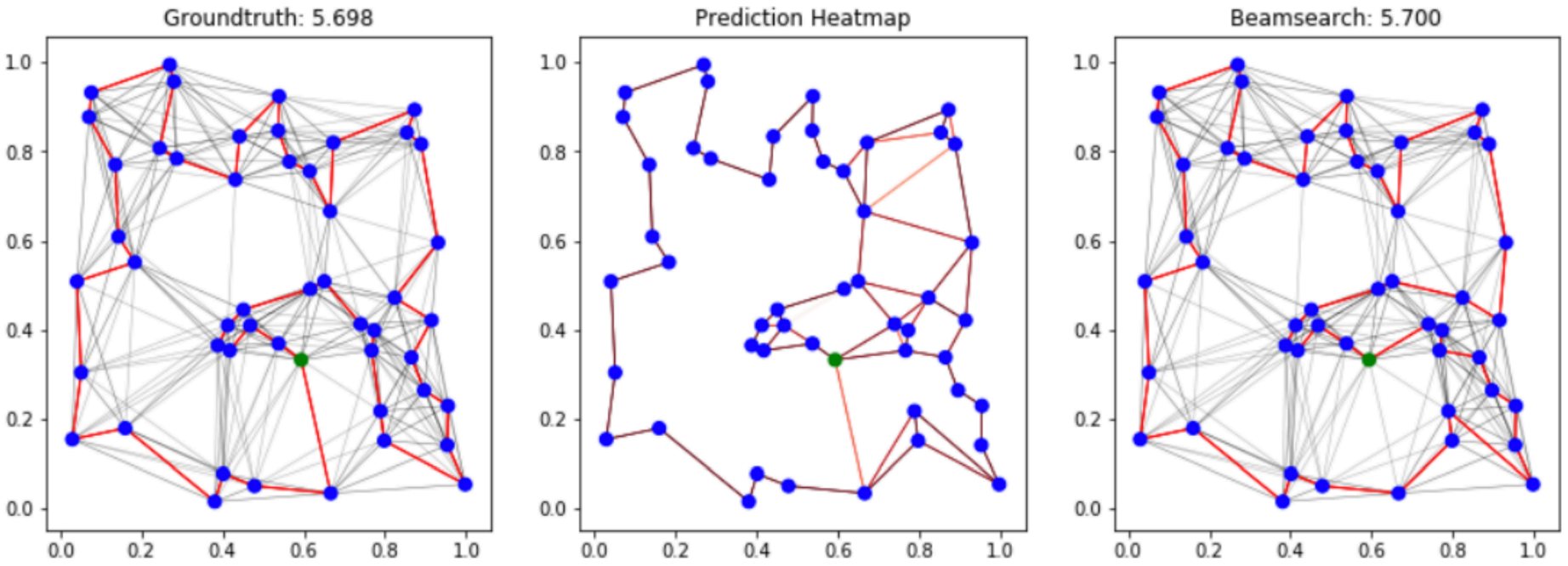
Performance: Supervised learning

Speed: Non-autoregressive

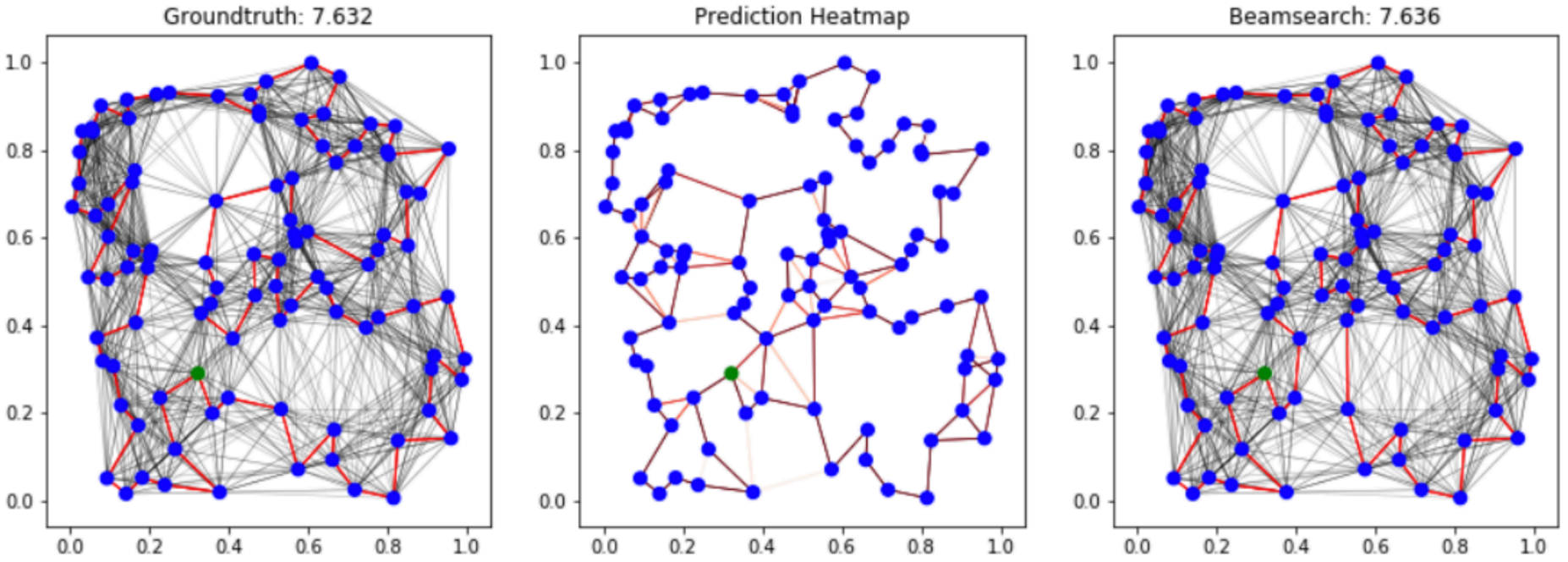
For small instances, the model is able to confidently identify most of the tour edges in the edge probability distribution without beam search



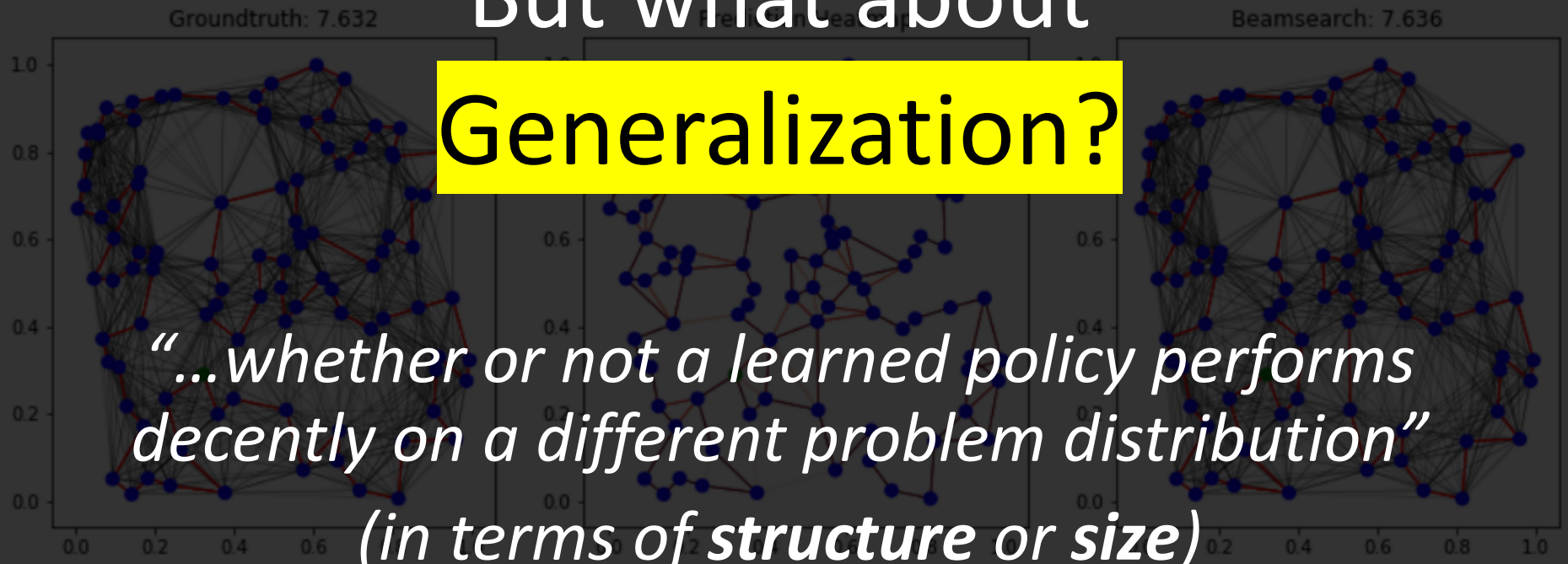
As instance size increases, edge probability distributions reflects the combinatorial explosion in TSP



Beam search is essential for finding the optimal tour for more complex instances



Beam search is essential for finding the optimal tour for more complex instances

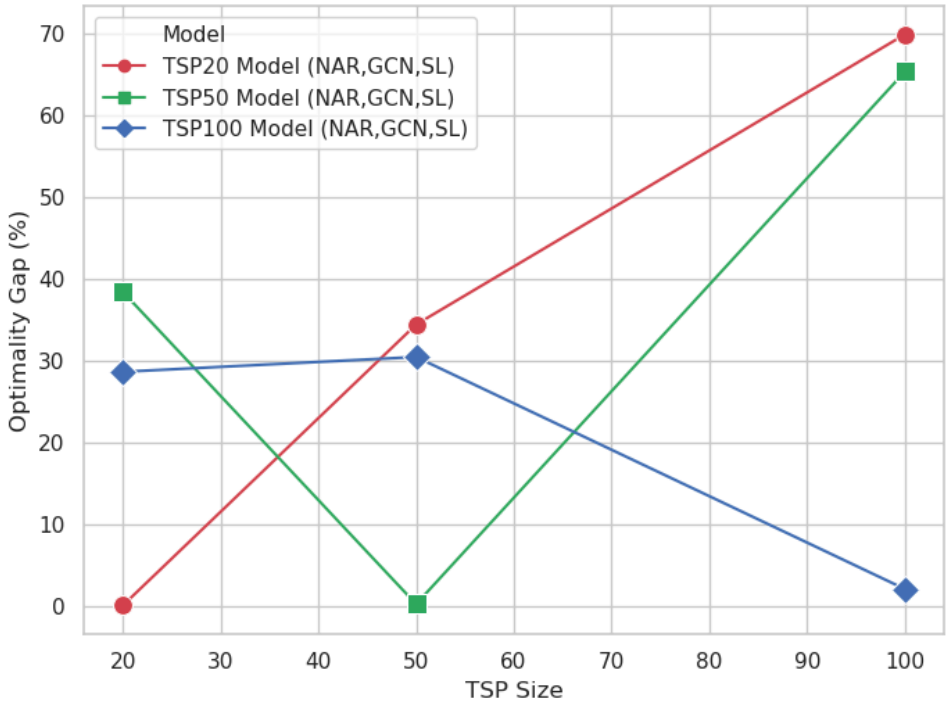


But what about Generalization?

"…whether or not a learned policy performs decently on a different problem distribution"
*(in terms of **structure** or **size**)*

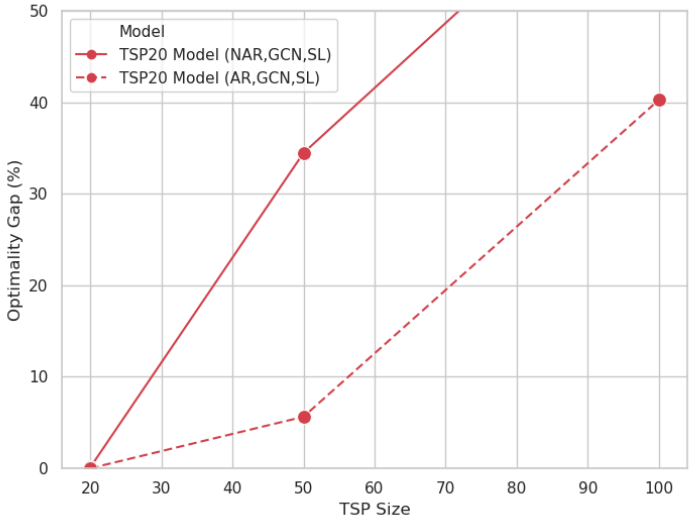
- Bengio, Lodi, Prouvost, 2018

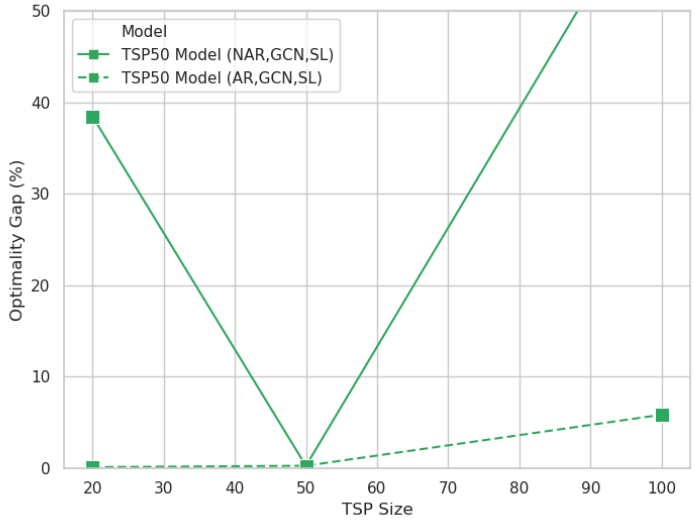
Generalization to variable TSP sizes

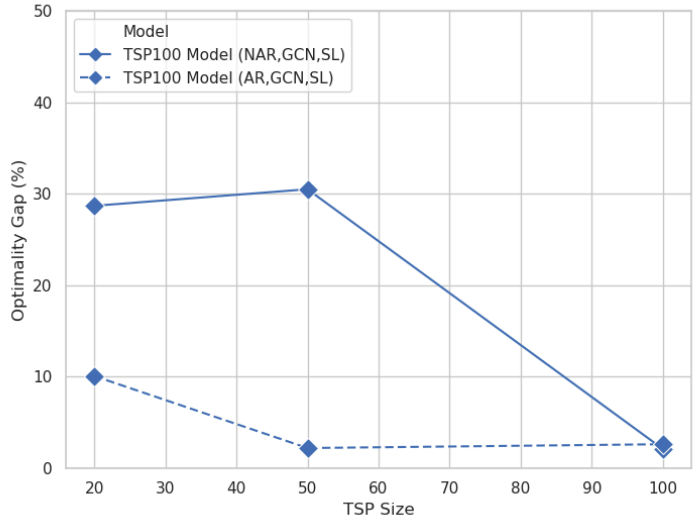


Optimality gap vs. TSP size, for NAR models when using beam search (with width = 1,280)

Generalization: impact of architecture

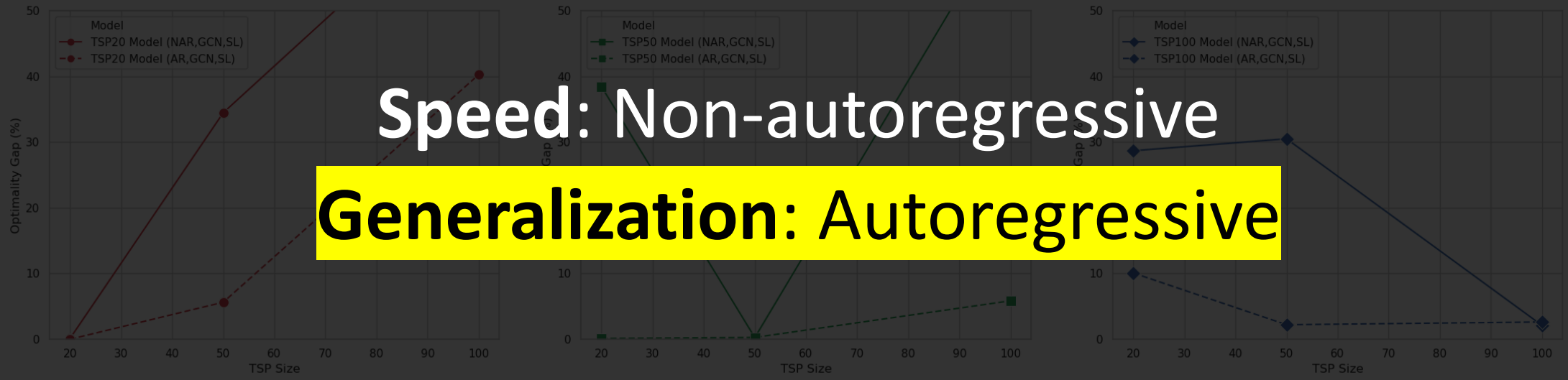






Optimality gap vs. TSP size, for **NAR and AR models** (both trained with SL)

Generalization to variable TSP sizes



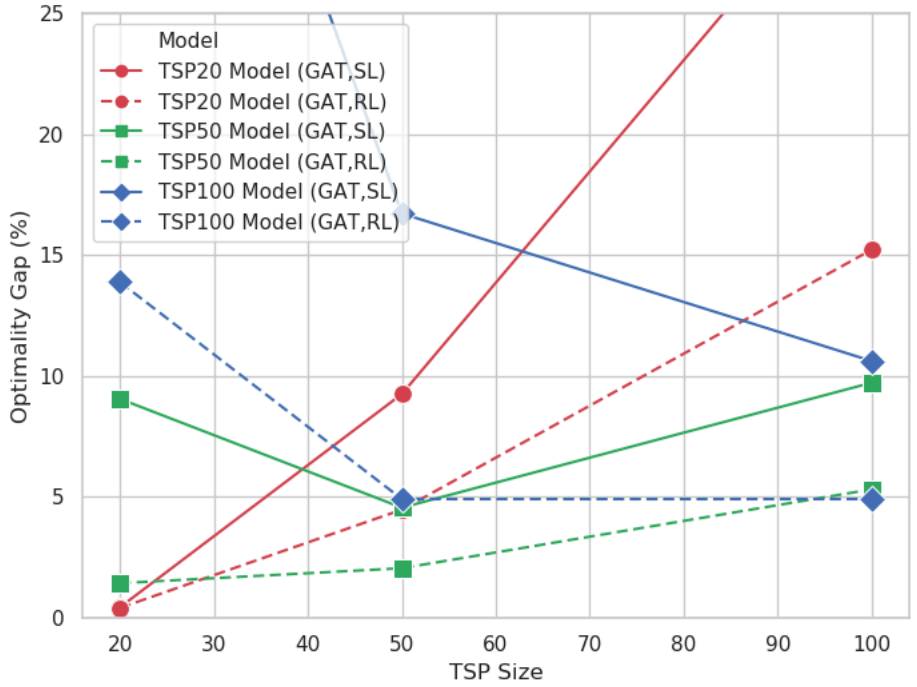
Speed: Non-autoregressive

Generalization: Autoregressive

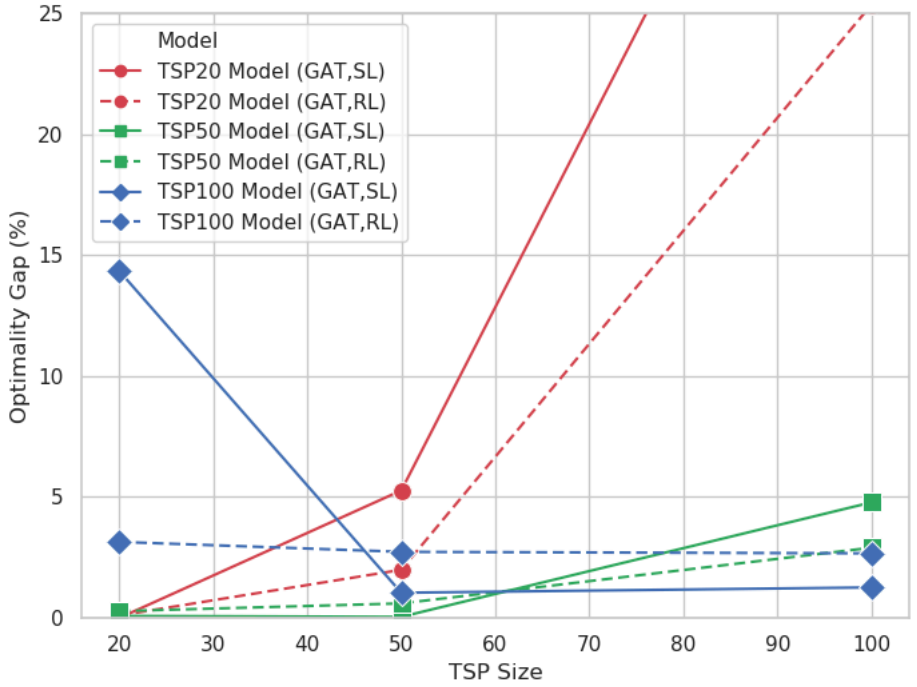
Optimality gap vs. TSP size for **NAR and AR models** (both trained with SL)

Generalization: impact of learning paradigm

Greedy search



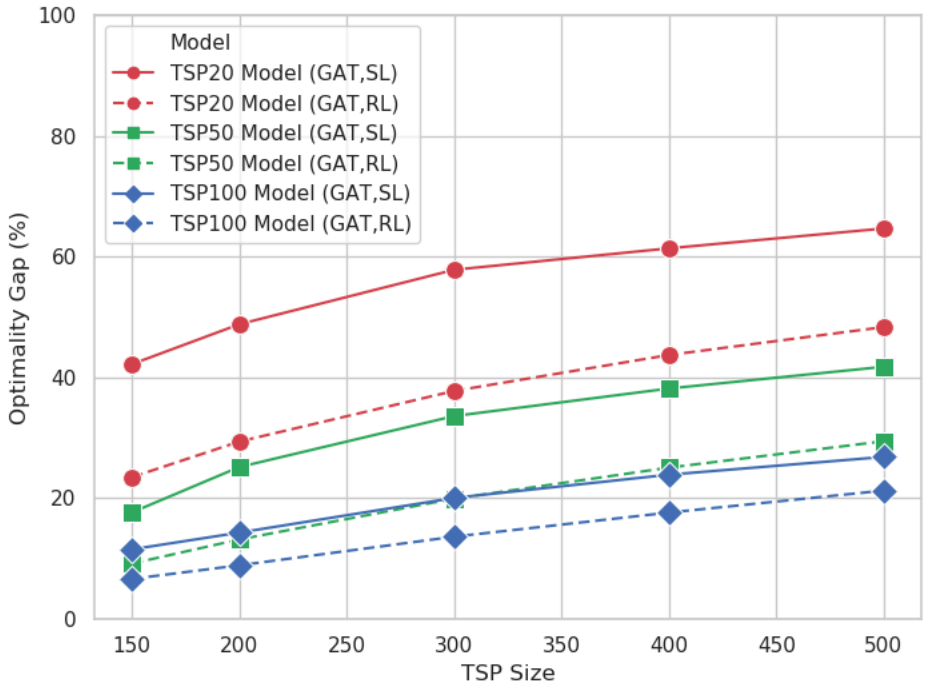
Sampling (1,280 solutions)



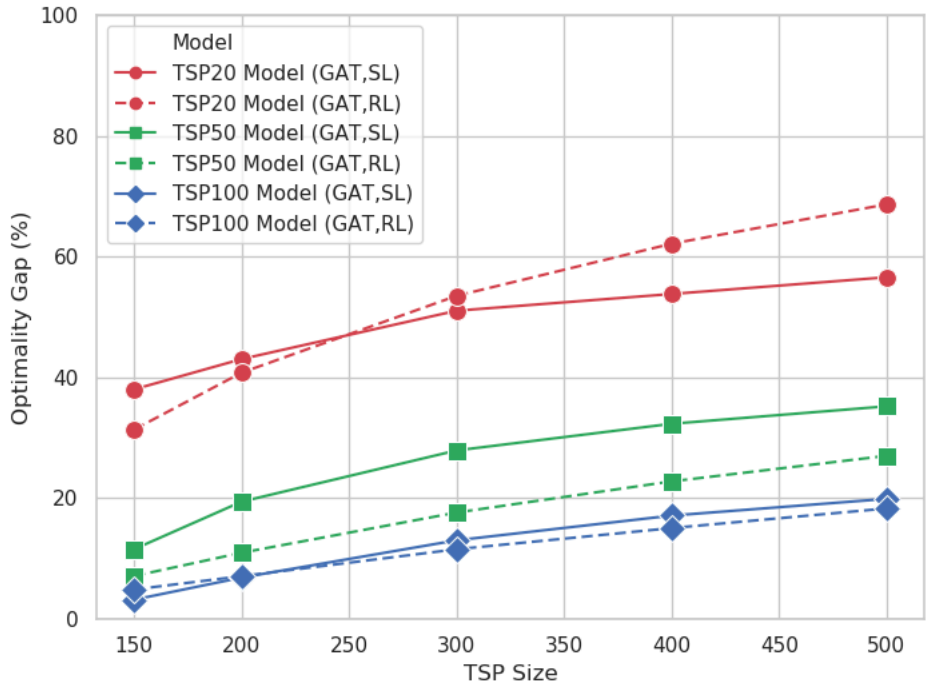
Optimality gap vs. TSP size, for AR models **trained with RL and SL**

Generalization: large-scale instances

Greedy search

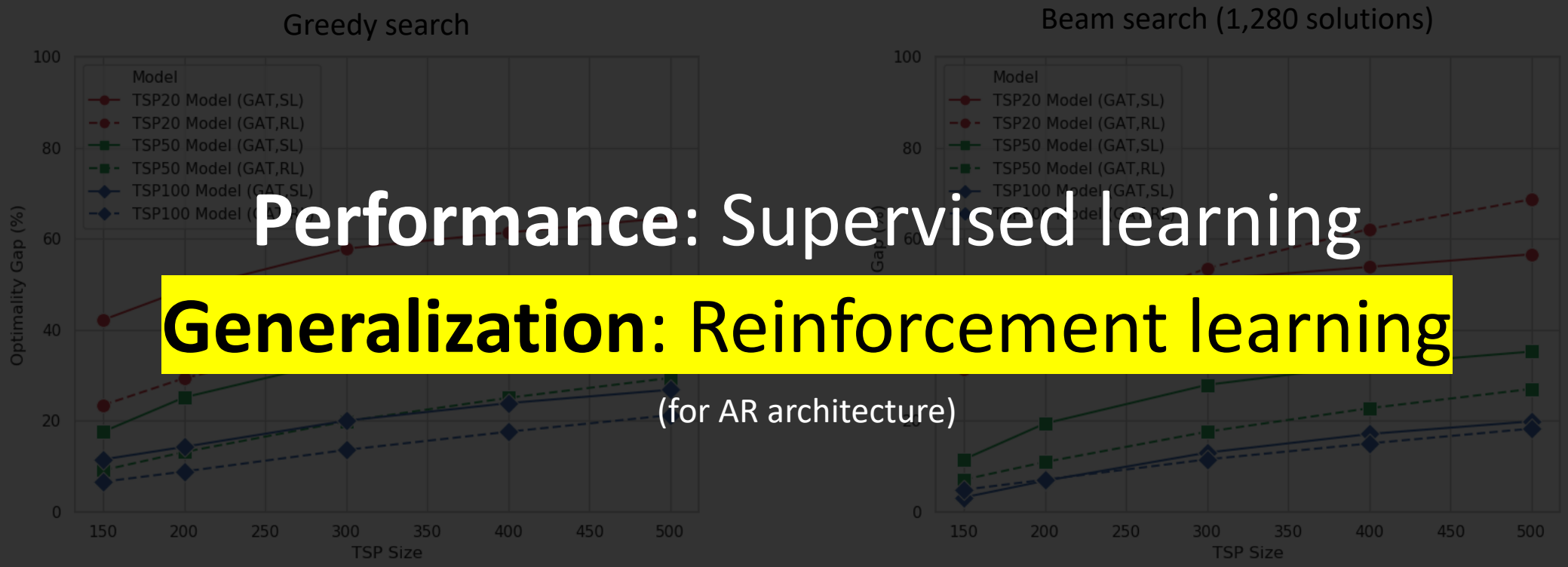


Beam search (1,280 solutions)



Optimality gap vs. TSP size, for AR models **trained with RL and SL**

Generalization: large-scale instances



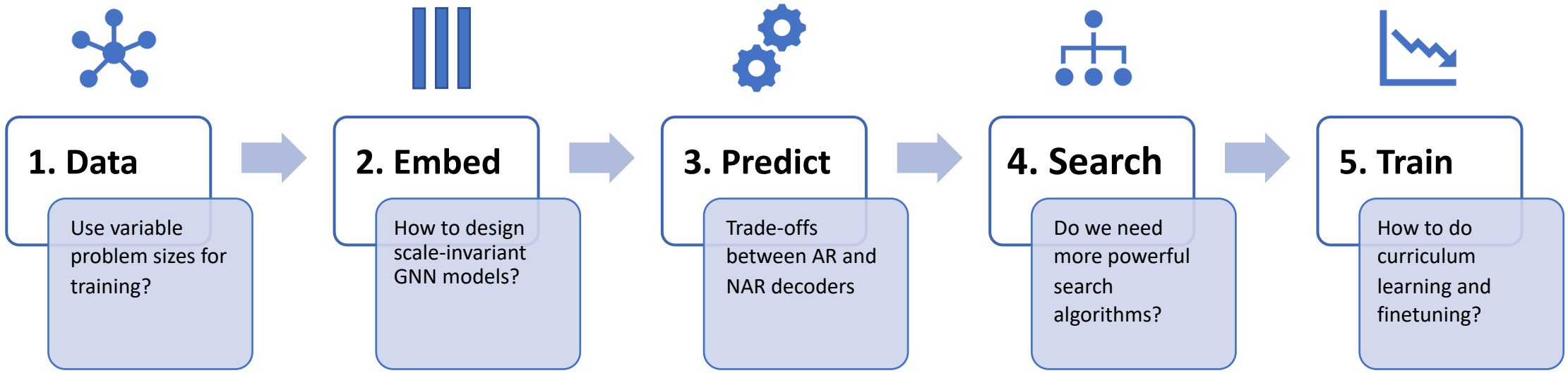
Performance: Supervised learning

Generalization: Reinforcement learning

(for AR architecture)

Optimality gap vs. TSP size, for AR models **trained with RL and SL**

End-to-end pipeline for OR problems



Next steps:

Where can we innovate for better scale-invariant generalization?

Questions?

Get the slides:

Chaitanya Joshi

Research Assistant

Nanyang Technological University

chaitanya.joshi@ntu.edu.sg

chaitjo.github.io

twitter.com/**chaitjo**

github.com/**chaitjo**