Graph Convolutional Neural Networks for the Travelling Salesman Problem

(SCSE18-0163)

Chaitanya K. Joshi

School of Computer Science and Engineering
Nanyang Technological University, Singapore

Supervised by Dr. Xavier Bresson

Abstract

1. Novel deep learning approach for approximately solving the **Travelling Salesman Problem**.

2. We train **Graph Convolutional Neural Networks** to predict TSP tours for graphs of up to 100 nodes.

3. Significant gains in **performance** and **speed** compared to all recently proposed deep learning techniques.

Table of Contents

Table of Contents

Motivation

Deep Learning

- Sub-field of Machine Learning.
- Algorithms that learn to perform a task directly from examples and observations.

Combinatorial Optimization

- Sub-field of Operations Research (OR).
- Hand-designed approximation algorithms for practical problems that are intractable to solve at scale.

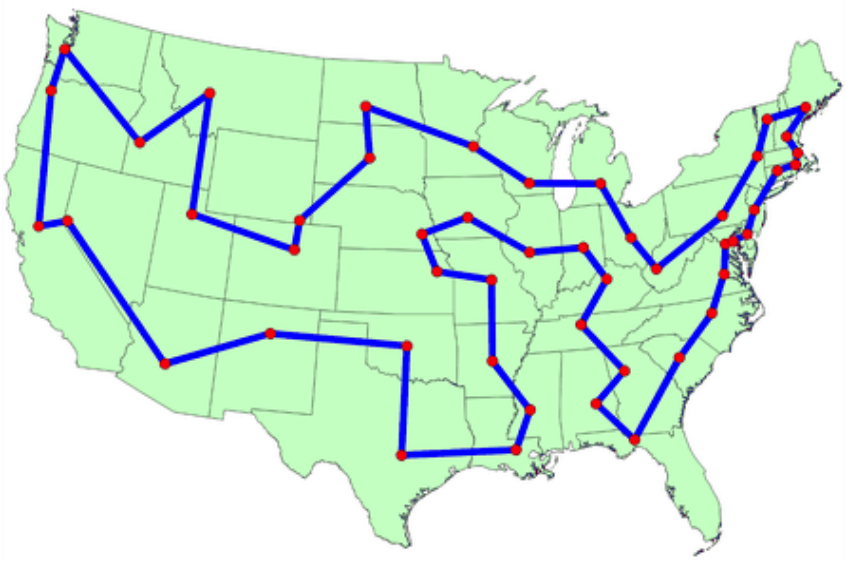
But handcrafting effective algorithms requires:

1. Significant specialized knowledge
2. Years of trial-and-error

Can we use Deep Neural Networks to ***learn*** better combinatorial optimization algorithms instead?
(Bengio et al., 2018)

The Travelling Salesman 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?"



Backbone of **modern industries** such as transportation, supply chain, genetics, and scheduling.

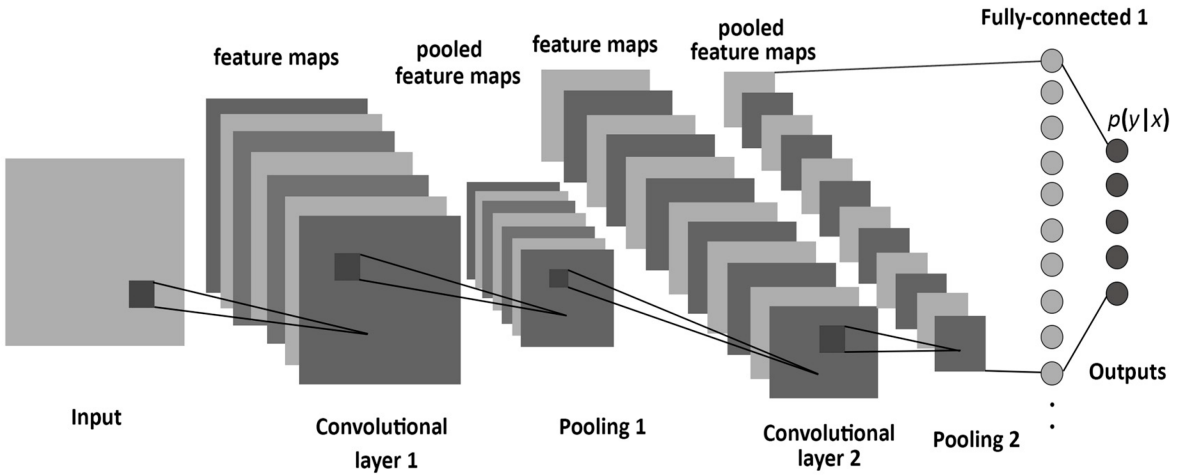
Learning the TSP

- In practice, **Concorde** (Applegate et al., 2006) can solve TSP upto thousands of cities.
- **TSP is highly structured**: Can be formulated as a sequential decision making task on graphs.
  $\Longrightarrow$ **Machine learning methods** have be used to train policies for making these decisions (Vinyals et al., 2015).

Profound Implication

A general learning algorithm for tackling *previously un-encountered* NP-hard problems, especially those that are non-trivial to design heuristics for (Bello et al., 2016).

Artificial Neural Networks

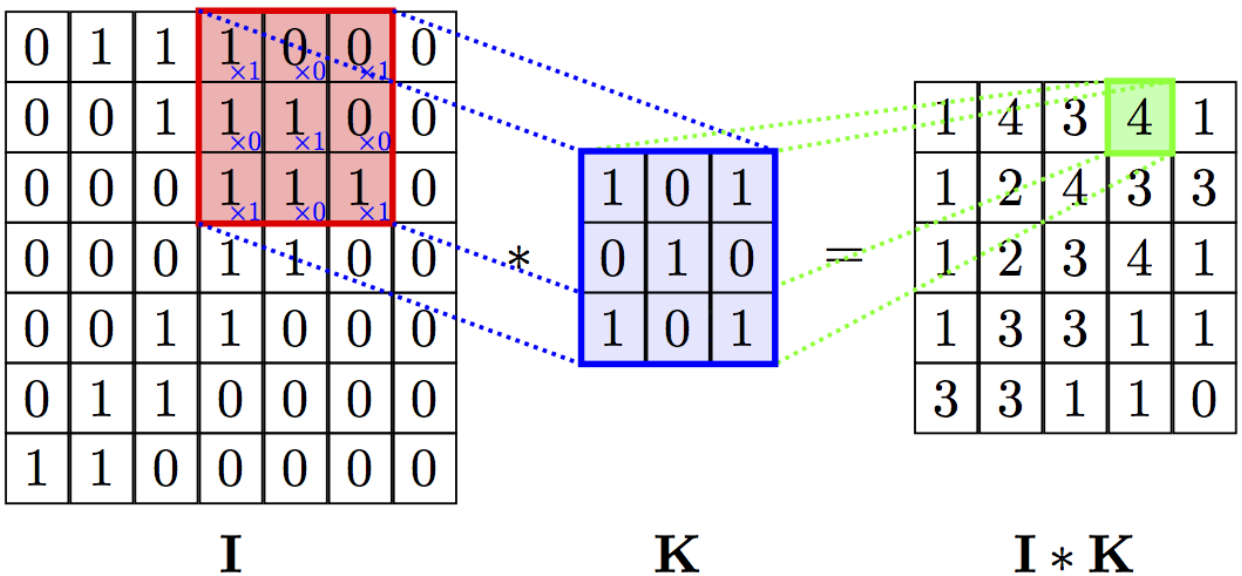


Deep Learning

Learning complicated concepts by building them from simpler ones in a hierarchical or multi-layer manner.

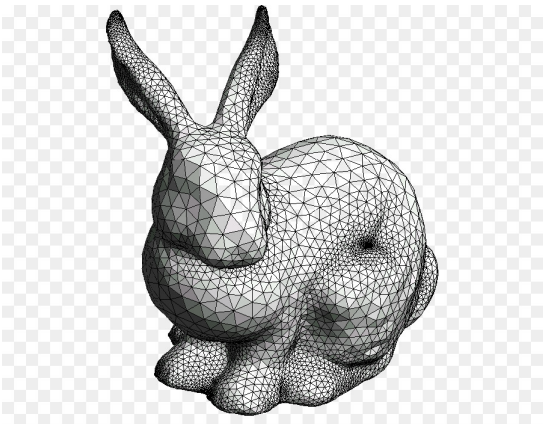
Convolutional Neural Network

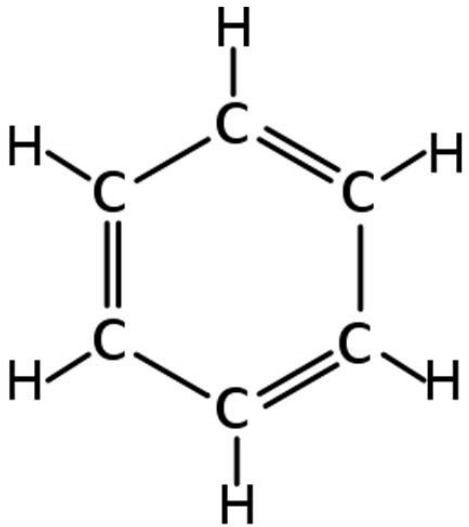
- Architecture that powers modern systems for natural language processing, speech recognition and computer vision.
- Highly parallelizable, scales very well with large **datasets + compute**.



Bottleneck

ConvNets require data domain to be regular: 2D Euclidean grids for images and 1D lines for text/speech.

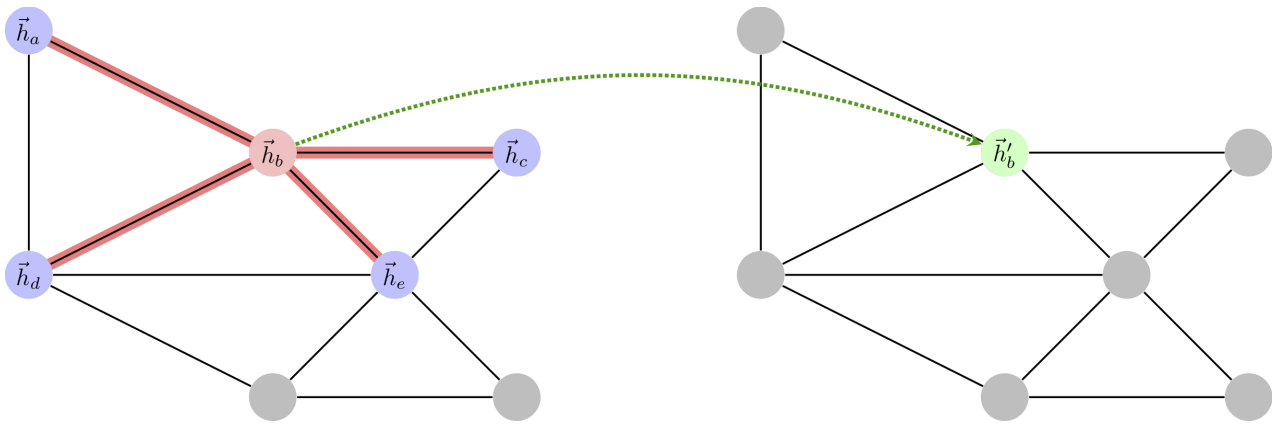




Real-world data in science/engineering

Underlying non-Euclidean structure $\Longrightarrow$ heterogenous graphs

Neural Networks on Graphs

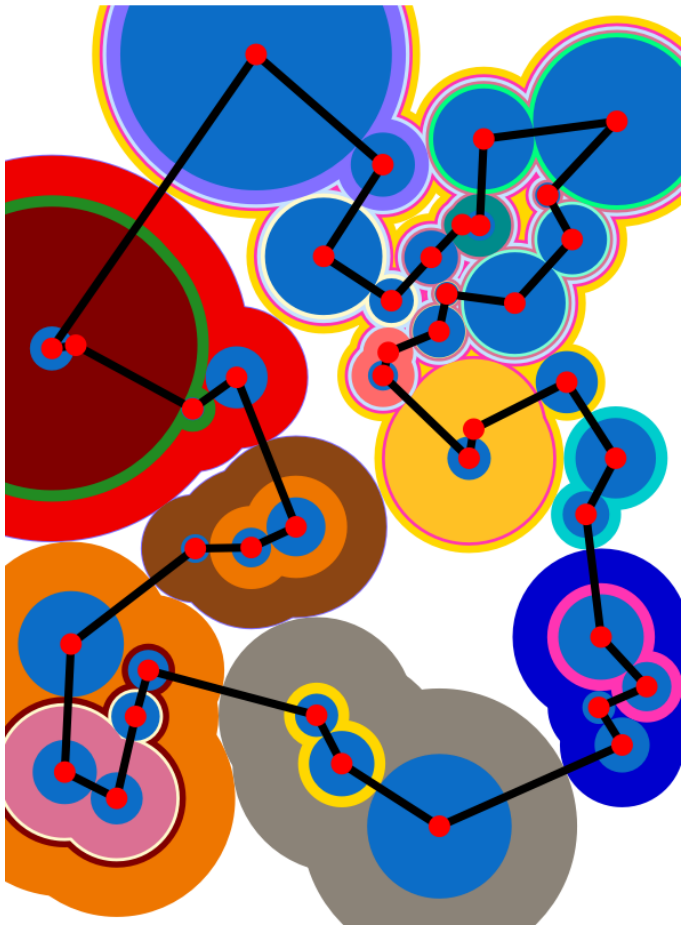


- **Geometric deep learning**: Emerging techniques generalizing neural networks to non-Euclidean domains such as graphs and manifolds (Bronstein et al., 2017).
- **Ideal for TSP**: Operate directly on the graph structure of the combinatorial problem.

Table of Contents

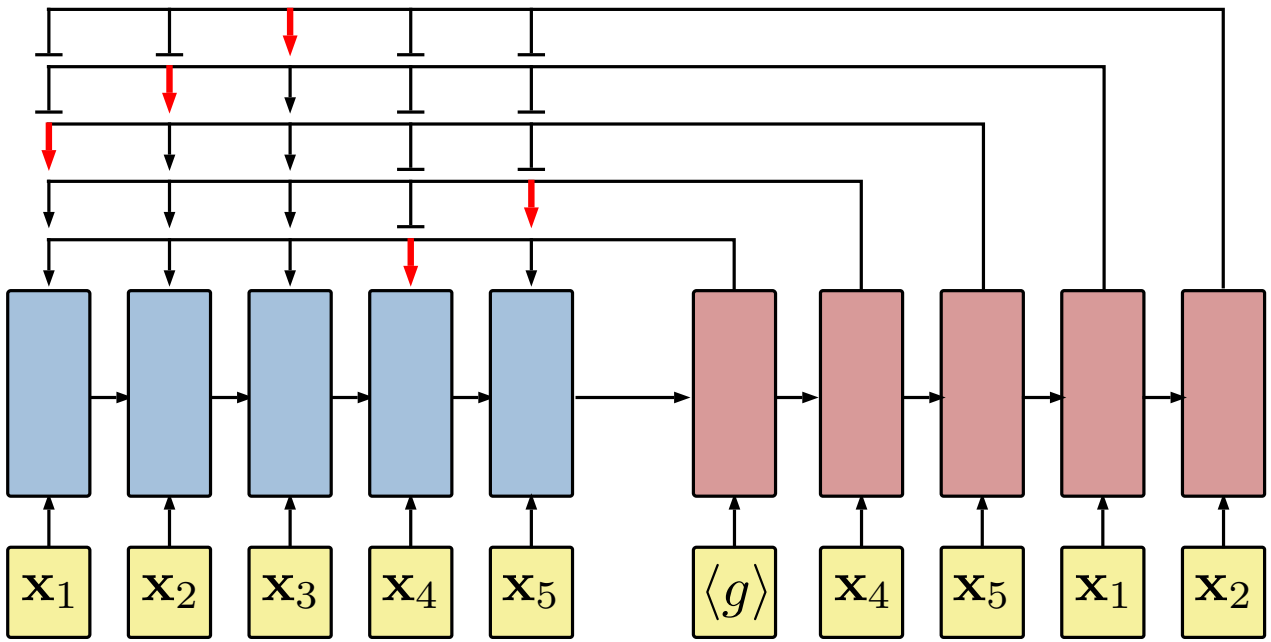
Concorde Solver

- Best known TSP solver today.
- **Cutting plane algorithms** (Dantzig et al., 1954) to iteratively solve linear relaxations of TSP.
- **Branch-and-bound** approach to reduce solution search space.



Sequence-to-Sequence Approach

Treat TSP as SEQ2SEQ task: Read input sequence of graph nodes one-by-one. Use 'Attention' mechanism to output a permutation of the input.



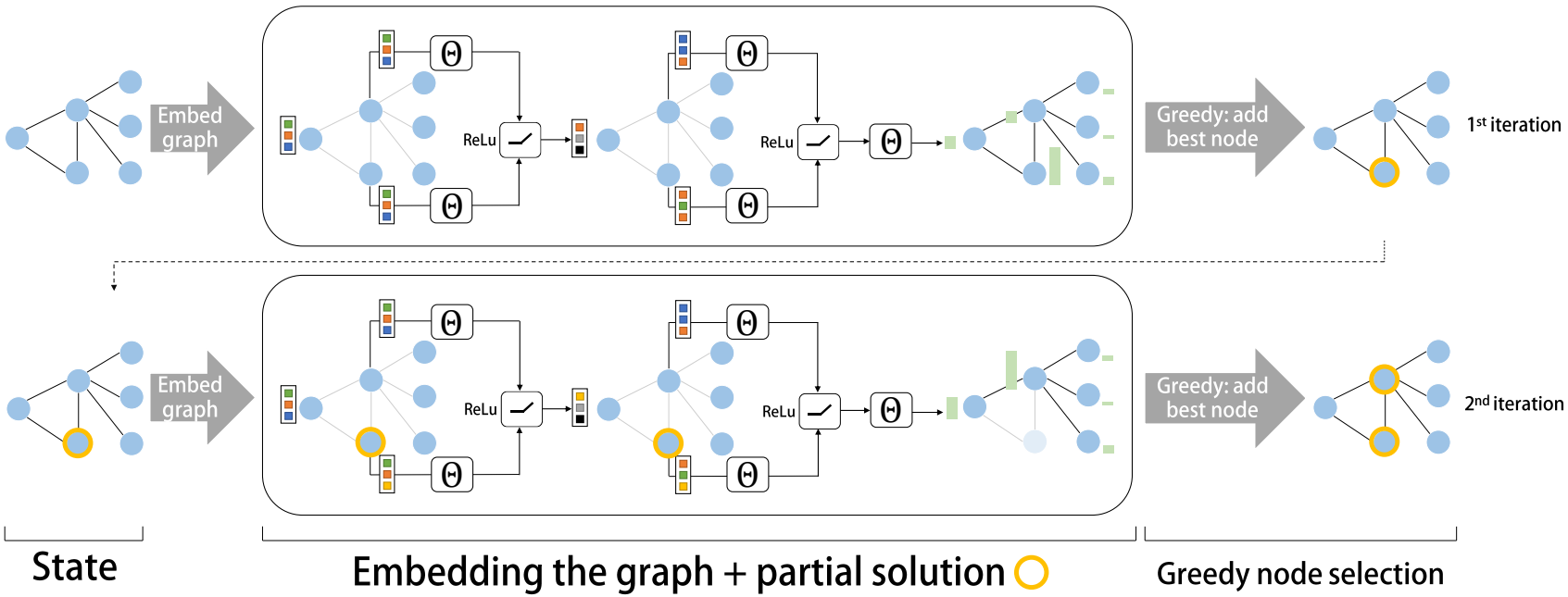
Sequence-to-Sequence Approach

Introduced by Vinyals et al. (2015) and extended by Bello et al. (2016).

- **Neural Network**: Pointer Network (SEQ2SEQ model), no use of graph structure.
- **Training Scheme**: Reinforcement Learning to minimize the length of the output tour (no need to compute optimal TSP solutions).
- **Solution Search**: Probabilistic sampling from learnt policy.
- **Output Type**: Autoregressive, step-by-step generation where the next step is conditioned on the previous step.

Graph Neural Network Approach

Use a graph neural network to build embeddings for each node and decode the TSP tour step-by-step using attention.



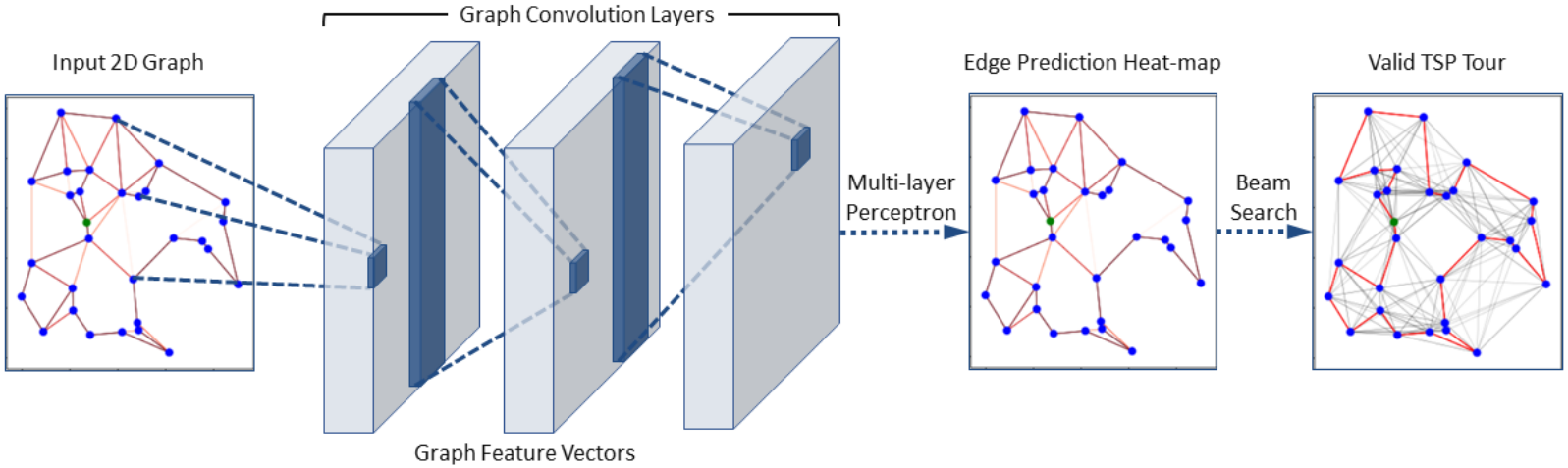
Graph Neural Network Approach

Introduced by Dai et al. (2017) (with greedy decoding), and made more powerful by Kool et al. (2019) using attention-based decoding.

- **Neural Network**: Sequentially applying the Graph Attention Network (GAT) (Veličković et al., 2017), invarient to input node ordering.
- **Training Scheme**: Reinforcement Learning.
- **Solution Search**: Probabilistic sampling from learnt policy.
- **Output Type**: Autoregressive.

Our Approach

Process the input graph in a single pass of a graph neural network to output an edge adjacency matrix. Convert the matrix into a valid TSP tour using post-hoc search.



Our Approach

Introduced by Nowak et al. (2017) and extended by us.

- **Neural Network**: Graph ConvNet (GCN) (Bresson & Laurent, 2017), more powerful than GAT.
- **Training Scheme**: Supervised Learning using pairs of problem instances and optimal solutions generated with Concorde.
- **Solution Search**: Beam search over edge adjacency matrix.
- **Output Type**: Non-autoregressive, edge adjacency matrix is generated in 'one-shot' instead of step-by-step.

Summary

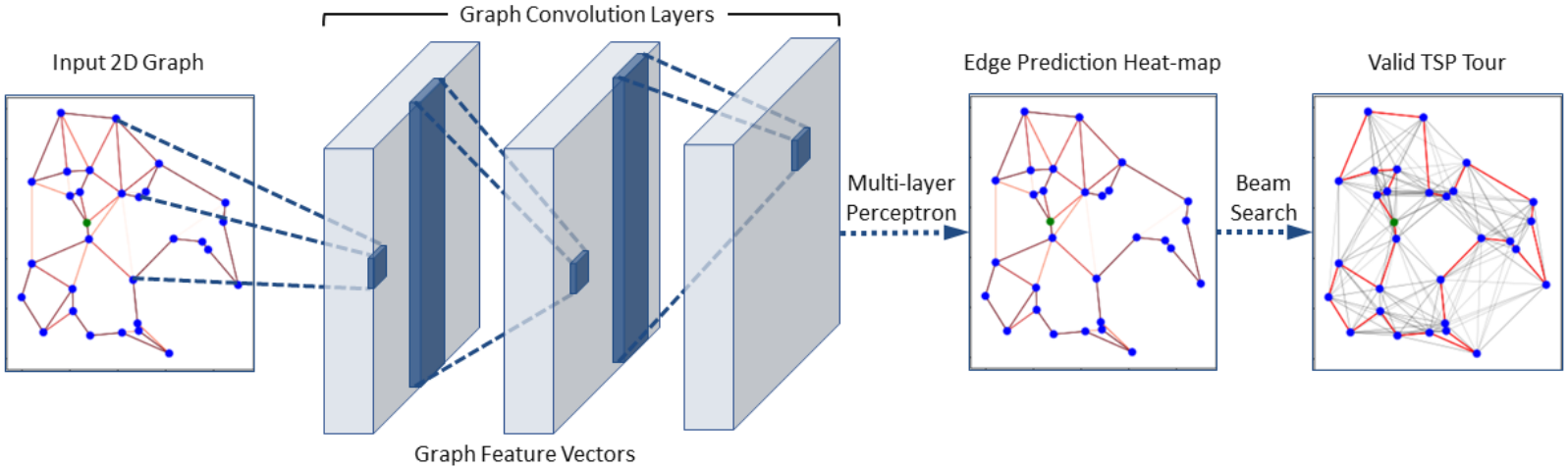
Our experiments will compare among the following approaches:

Method	Neural Network	Model Type	Training Setting	Solution Search Type
Bello et al. (2016)	Pointer Network	Autoregressive	RL	Sample from policy
Kool et al. (2019)	Graph Attention Network	Autoregressive	RL	Sample from policy
Ours	Graph ConvNet	Non-autoregressive	SL	Beam search

We shall also compare to **Concorde** (TSP heuristics), **Guorobi** (an exact LP/QP solver) and **Google OR Tools** (general heuristics).

Table of Contents

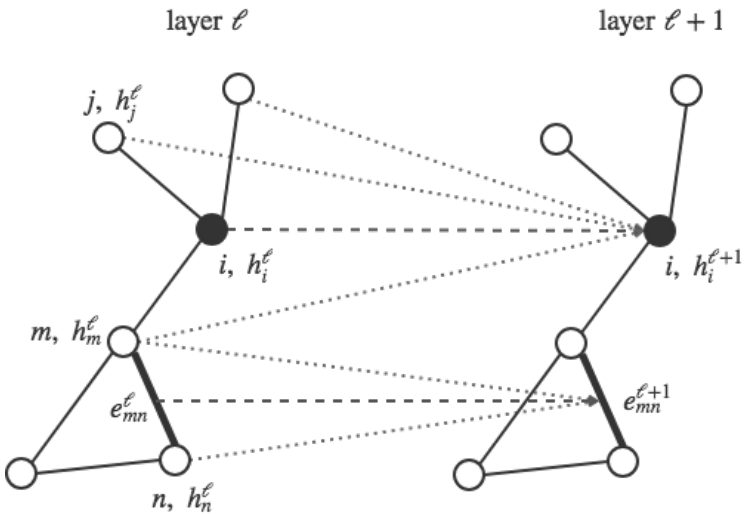
Model Overview



1. Extract compositional feature vectors from input graph by stacking several graph convolution layers.
2. Output is an edge adjacency matrix denoting the probabilities of edges occurring on the TSP tour.
3. Edge prediction *heat-map* is converted to a valid tour using a post-hoc beam search strategy.

Graph Convolution Layer

Graph Convolution layers compute h-dimensional representations h_i for each node i and e_{ij} for the edge between each node i and j in the graph.



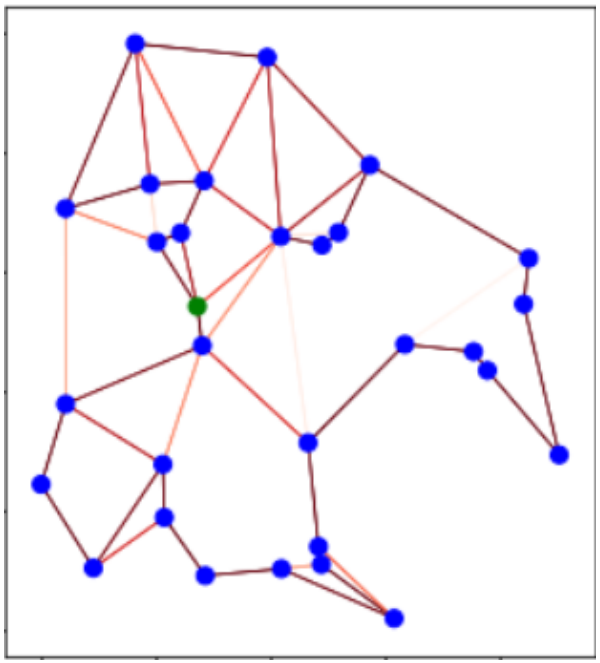
$$h_i^{\ell+1} = \mathrm{ReLU}\left(\theta_4^\ell\ h_i^\ell + \frac{\sum_{j \to i} \eta_{ij}^\ell \odot \theta_5^\ell\ h_j^\ell}{\sum_{j \to i} \eta_{ij}^\ell + \epsilon} \right)$$

$$\eta_{ij}^{\ell+1} = \sigma\left(\theta_6^\ell\ e_{ij}^\ell + \theta_7^\ell\ \left(h_i^\ell + h_j^\ell \right) \right)$$

$$e_{mn}^{\ell+1} = \mathrm{ReLU}\left(\theta_6^\ell\ e_{mn}^\ell + \theta_7^\ell\ \left(h_m^\ell + h_n^\ell \right) \right)$$

Final h-dimensional embedding e'_{ij} for each edge is fed to a Multi-layer Perceptron classifier to compute the probability of that edge being connected in the TSP tour.

Beam Search Decoding



Output of the model: probabilistic heat-map over the adjacency matrix of tour connections.

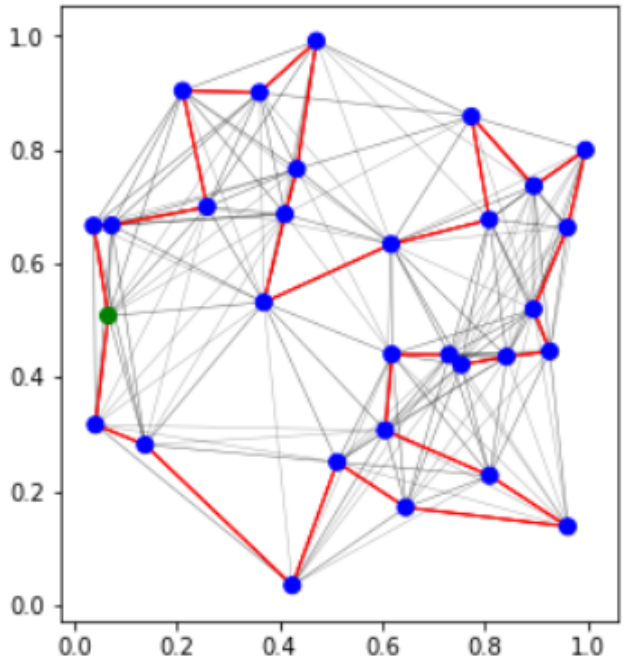
1. Starting from a random node, expand b most probable edge connections among the node's neighbors.
2. Keep expanding the top-b partial tours at each stage till all nodes visited.
3. Final prediction is the tour with the highest probability among the b tours at the end of search.

Table of Contents

Dataset Generation

- **Current paradigm**: Training and evaluating models on TSP instances of fixed sizes. Generate training, validation and test datasets for graphs of sizes 20, 50 and 100 nodes.

- Training sets: **1 Million pairs** of problem instances and solutions, and validation/test sets: **10,000 pairs**.



For each TSP, n node locations sampled randomly in the unit square. Optimal tour found using Concorde.

Measuring Performance

We train separate models for TSP20, TSP50 and TSP100, and compute metrics on held-out test sets of the same problem size:

1. **Predicted tour length**: Average predicted tour length $\hat{c}$ over 10,000 instances: $\frac{1}{m}\sum_{i=1}^{m}\hat{c}_i$.

2. **Optimality gap**: Average percentage ratio of predicted tour length $\hat{c}$ relative to optimal solution c over 10,000 instances: $\frac{1}{m}\sum_{i=1}^{m}\left(\frac{\hat{c}_i}{c_i}-1\right)$.

3. **Evaluation time**: Total wall clock time taken to solve 10,000 instances, either on single GPU (Nvidia 1080Ti) or 32 instances in parallel on a 32 virtual CPU system (2 $\times$ Xeon E5-2630).

Table of Contents

Performance on Fixed-size Instances

Graph ConvNet model outperforms the state-of-the-art deep learning approach (Kool et al., 2019) in terms of both closeness to optimality and evaluation time.

Method	TSP20			TSP50			TSP100		
	Tour Len.	Opt. Gap.	Time	Tour Len.	Opt. Gap.	Time	Tour Len.	Opt. Gap.	Time
Concorde	3.84	0.00%	(1m)	5.70	0.00%	(2m)	7.76	0.00%	(3m)
Gurobi	3.84	0.00%	(7s)	5.70	0.00%	(2m)	7.76	0.00%	(17m)
Google OR Tools	3.85	0.37%	–	5.80	1.83%	–	7.99	2.90%	–
PtrNet (Bello et al., 2016)	3.84	0.10%	–	5.75	0.95%	–	8.00	3.03%	–
GAT (Kool et al., 2019)	3.84	0.08%	(5m)	5.73	0.52%	(24m)	7.94	2.26%	(1h)
GCN (Ours)	**3.84**	**0.20%**	(20s)	**5.72**	**0.42%**	(2m)	**7.92**	**2.06%**	(10m)

Better and Faster Models

Larger Models $\Longrightarrow$ Better Learning

$+$ We attribute our gains in performance to more powerful representation learning using **very deep convolutional architectures** with up to 30 layers.

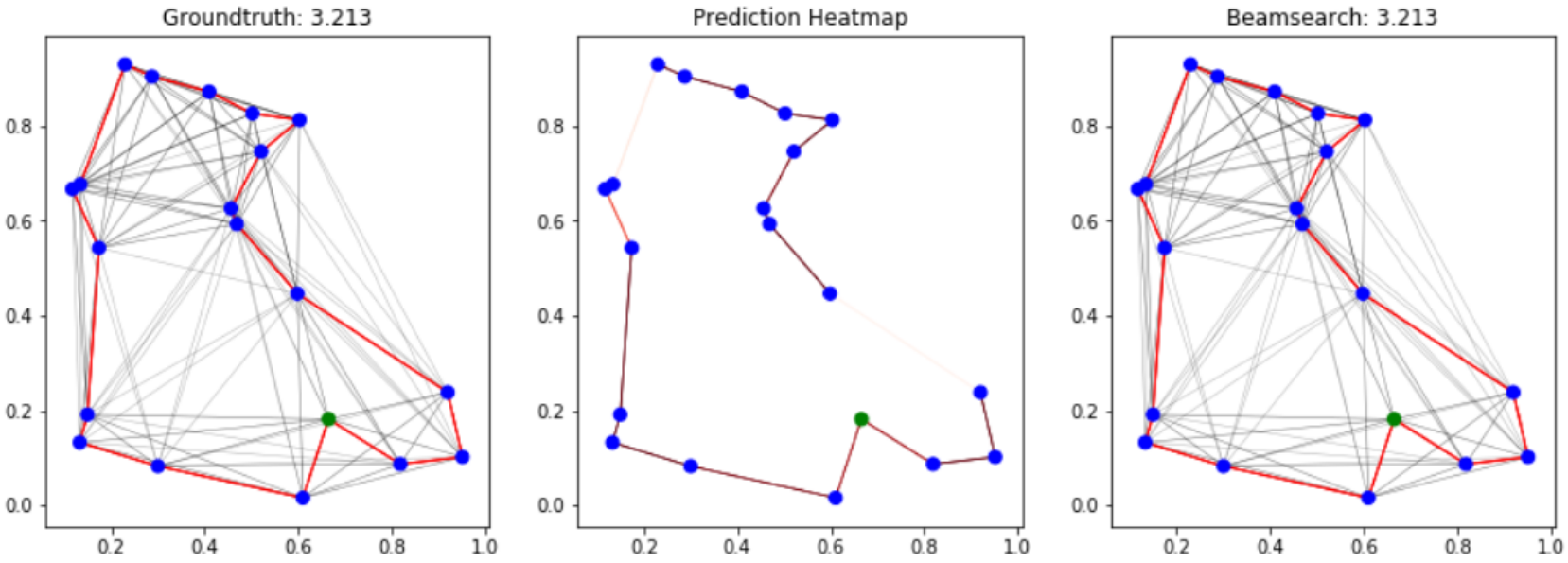
$-$ Kool et al. (2019) use only 3 GAT layers.

Parallelized Search $\Longrightarrow$ Faster Models

$+$ Despite using larger models, our non-autoregressive beam search implementation is **highly parallelized**, leading to fast evaluation.

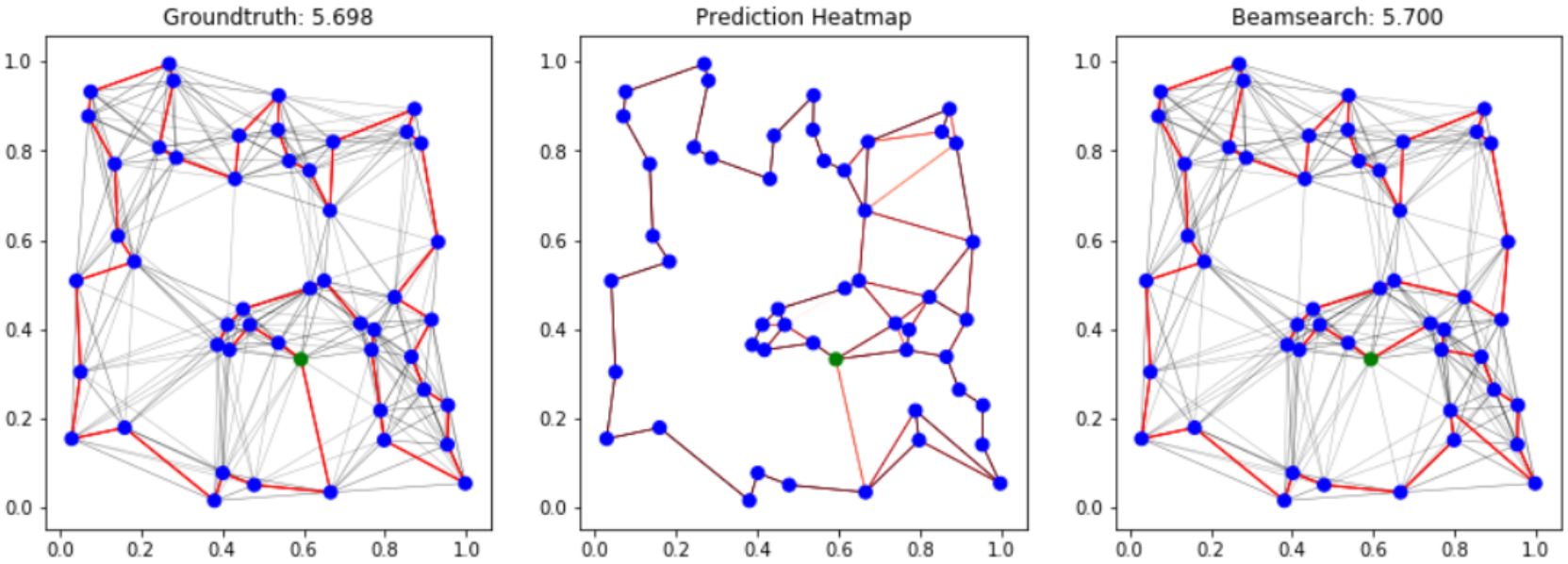
$-$ Autoregressive approaches generate solutions step-by-step by sampling from an RL policy, which cannot be parallelized.

TSP20 Visualization



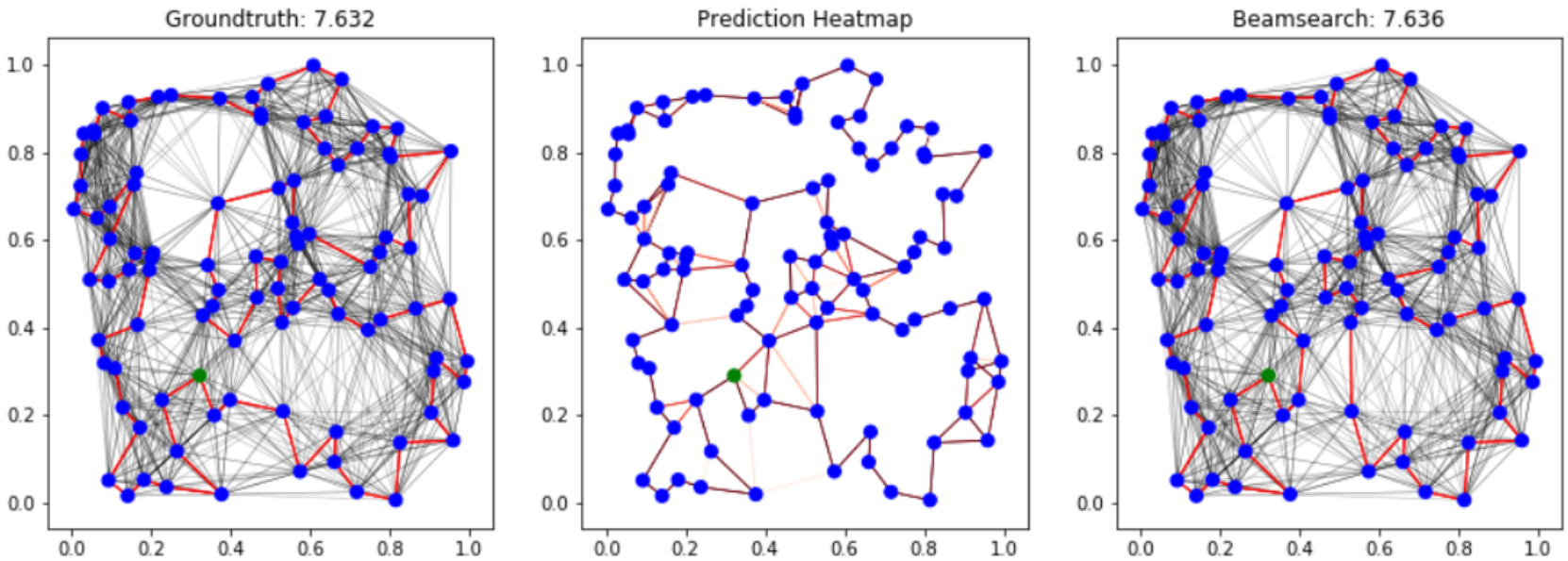
For small instances, the model is able to confidently identify most of the tour edges in the heat-map without beam search.

TSP50 Visualization



As instance size increases, prediction heat-map reflects the *combinatorial explosion* in TSP.

TSP100 Visualization



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

Trade-offs

Supervised Learning

- **Limited data**: Generating labelled datasets for instances beyond hundreds of nodes is costly (computation and time).
- **Better learning**: The model is given more information to learn better solvers.

Reinforcement Learning

- **Infinite data**: Does not require the creation of labelled datasets as long as we can design reward functions.
- **Worse learning**: The model is only given a reward $\Longrightarrow$ less informative for learning.

Table of Contents

Conclusion

We propose a simple framework for learning combinatorial problems that improves over the state-of-the-art:

1. **Powerful learning capacity**: Can efficiently build very deep models.
2. **Faster inference**: All components can be parallelized on GPUs.

Future Work

Our vision for Combinatorial Optimization

- Models are trained on small problem instances (using SL) and effectively generalize to larger instances (using RL).
- Beyond certain graph sizes, Neural Networks will be faster than Concorde/handcrafted solvers due to parallelization.

References I

David L Applegate, Robert E Bixby, Vasek Chvatal, and William J Cook. *The traveling salesman problem: a computational study*. Princeton university press, 2006.

Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *arXiv preprint arXiv:1811.06128*, 2018.

Xavier Bresson and Thomas Laurent. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2017.

References II

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Hanjun Dai, Elias Khalil, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, pp. 6348–6358, 2017.

George Dantzig, Ray Fulkerson, and Selmer Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, 2(4):393–410, 1954.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=ByxBFsRqYm`.

References III

Alex Nowak, Soledad Villar, Afonso S Bandeira, and Joan Bruna. A note on learning algorithms for quadratic assignment with graph neural networks. *arXiv preprint arXiv:1706.07450*, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pp. 2692–2700, 2015.