AI for Traveling Salesman Problem

Academy of Mathematics and Systems Sciences, CAS

May 9, 2023

Outline

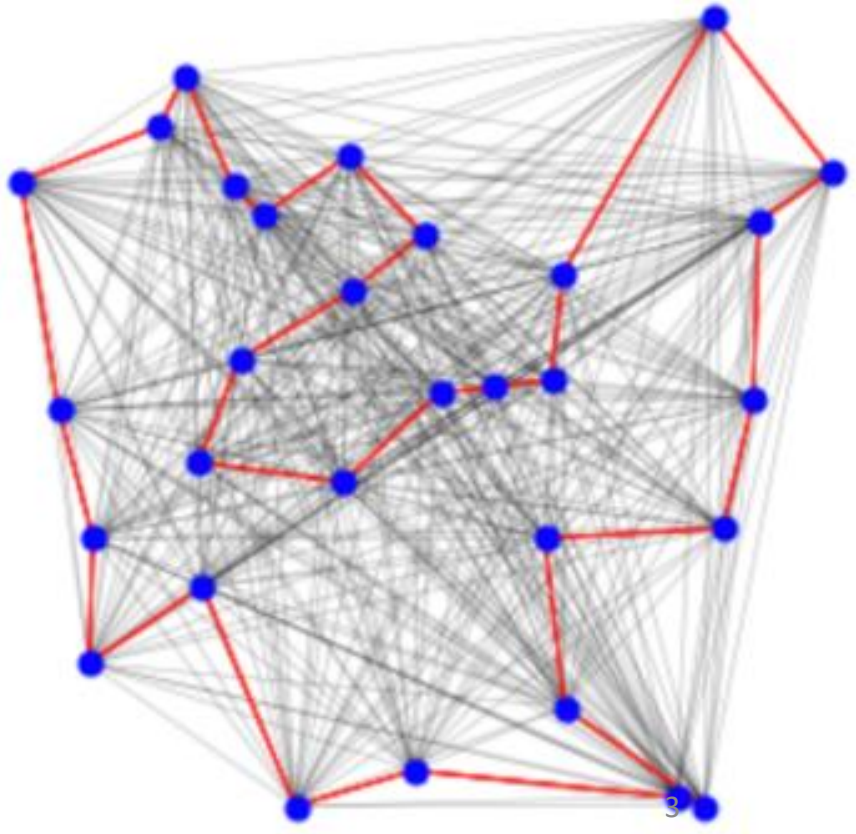
- History of TSP
- Traditional Solvers
  - Exact algorithms
  - Heuristic solvers
- Neural Network Solvers
  - Transformer based methods
  - GCN based methods
- Numerical Experiments

History of Traveling Salesman Problem (TSP)

- Definition: Given a list of cities and the distances between each pair of cities, what is the shortest possible path that visits each city exactly once and returns to the origin city?
- First mathematical formulation by W. Hamilton 1805-1865.

- TSP belongs to the class of routing problems: warehouse, transportation, supply chain, hardware design, manufacturing, etc.).

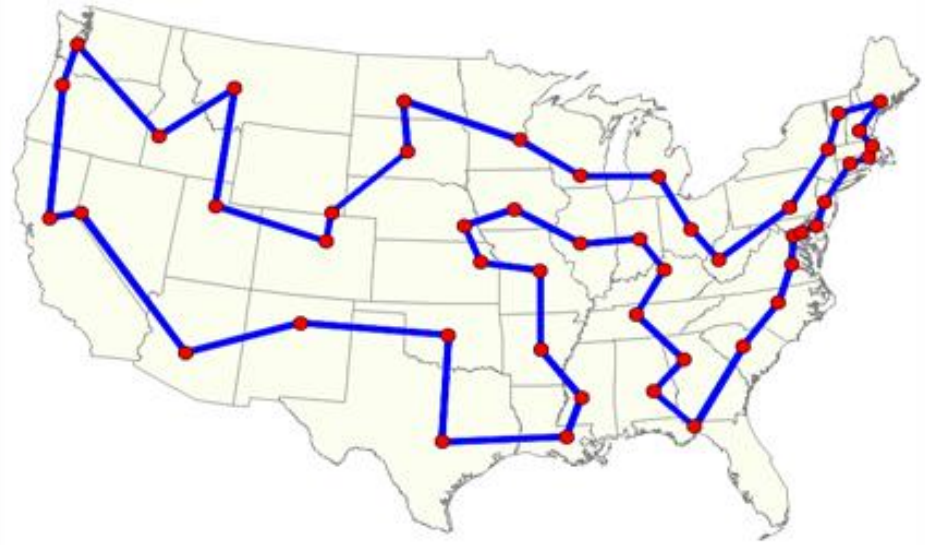
- TSP is NP-hard, exhaustive search has a complexity in O(n!).



History of TSP

- TSP is the most popular and most studied combinatorial problem, starting with von Neumann (1951).

- TSP has driven the discovery of several important optimization techniques: Cutting Planes[1], Branch-and-Bound[2], Local Search[3], Lagrangian Relaxation[4], and Simulated Annealing[5].



[1] Dantzig, Fulkerson, Johnson, 1954

[2] Bellman, Held, Karp, 1962

[3] Croes, A method for solving traveling-salesman problems, 1958

[4] Fisher, The Lagrangian Relaxation Method for Solving Integer Programming Problems, 1981

[5] Kirkpatrick, Gelatt, Vecchi, Optimization by Simulated Annealing, 1983

Traditional Solvers

- Two traditional approaches to tackle combinatorial problems:
  - Exact algorithms
    - Exhaustive search, Dynamic or Integer Programming
    - These algorithms are guaranteed to find optimal solutions, but they become intractable when n grows.
  - Approximate/heuristic algorithms
    - These algorithms trade optimality for computational efficiency.
    - They are problem-specific, often designed by iteratively applying a simple mancrafted rule, known as heuristic.
    - Their complexity is polynomial and their quality depends on an approximate ratio that characterizes the (worst/average-case) error w.r.t the optimal solution.

Traditional Solvers: Exact algorithms

- Exact algorithms:
  - Dynamic programming[1]: $O(n^2 2^n)$, intractable for $n > 40$.
  - Gurobi[2]: General purpose Integer Programming (IP) solver with Cutting Planes (CP) and Branch-and-Bound (BB).
  - Concorde[3]: Highly specialized linear IP+CP+BB for TSP. Concorde is widely regarded as the fastest exact TSP solver, for large instances, currently in existence.

[1] Held, Richard, A dynamic programming approach to sequencing problems, 1962
[2] Gu, Rothberg, Bixby, Gurobi, 2008
[3] Applegate, Bixby, Chvatal, Cook, The Traveling Salesman Problem: A Computational Study, 2006

Traditional Solvers

- TSP can be formulated as Integer Linear Programming (ILP) problem:

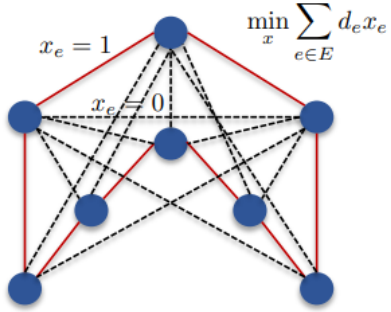
$$\min_x \sum_{e \in E} d_e x_e \quad \text{s.t.}$$

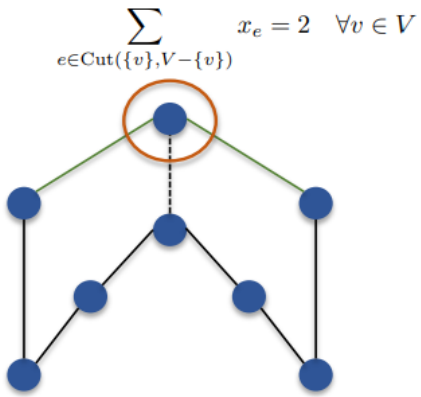
$$\sum_{e \in \mathrm{Cut}(\{v\}, V-\{v\})} x_e = 2 \quad \forall v \in V$$

$$\sum_{e \in \mathrm{Cut}(S, S^c)} x_e \geq 2 \quad \forall S \subset V, S \neq \emptyset, S \neq V$$

$$\text{where } \mathrm{Cut}(A, A^c) = \{e_{vv'} \text{ s.t. } v \in A, v' \in A^c\}$$

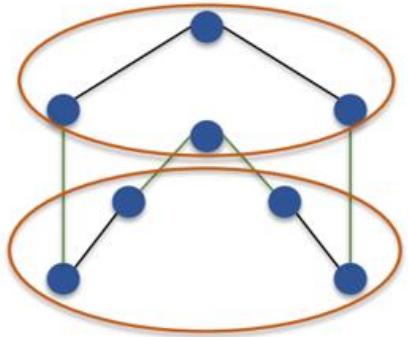
$$x_e \in \{0, 1\} \quad \forall e \in E$$



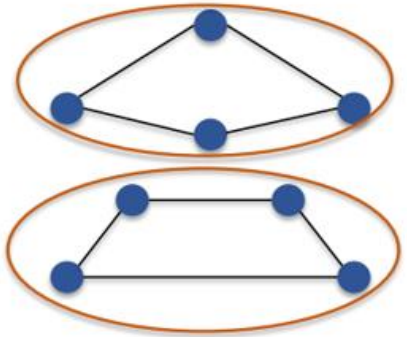


Sub-contour constraints that guarantee a feasible tour.

$$\sum_{e \in \mathrm{Cut}(S, S^c)} x_e \geq 2 \quad \forall S \subset V$$



Satisfied

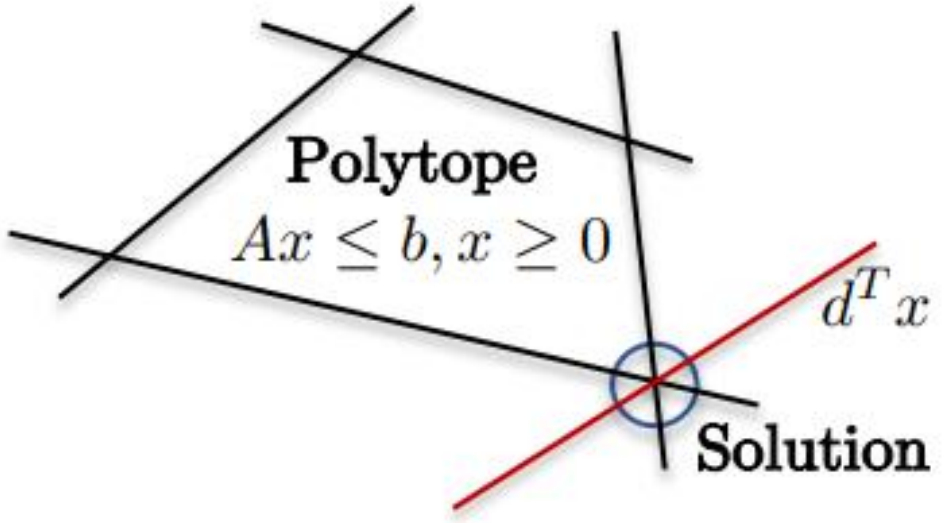


Not satisfied

Traditional Solvers

- IP Solvers
  - ILP problems are NP-hard: the space of optimization is binary $x_e \in \{0,1\}$.
  - ILP can be relaxed as a Linear Programming (LP) problem[1] with $x_e \in [0,1]$ of the standard form:

$$\min_x d^T x \qquad \text{s.t. } Ax \leq b, x \geq 0$$

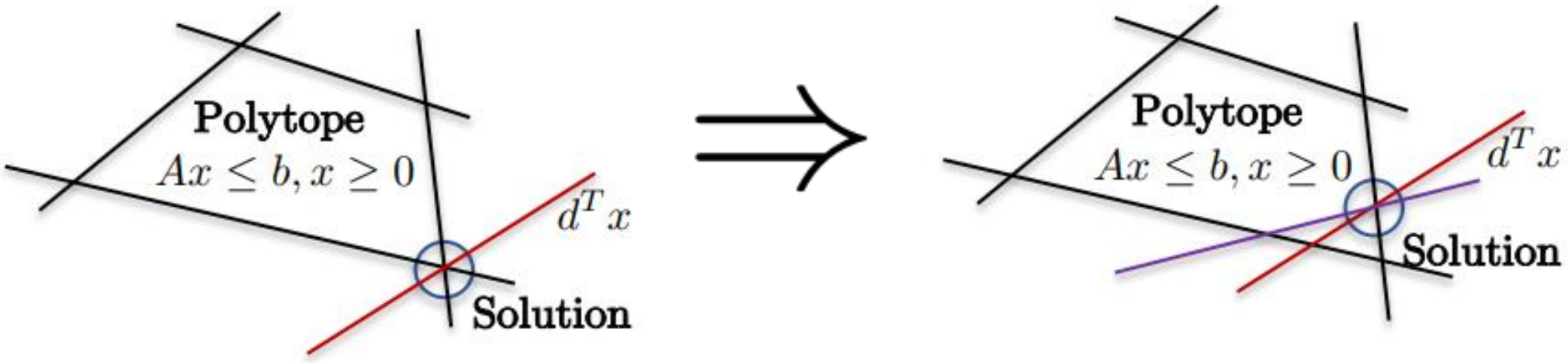


[1] Dantzig, Fulkerson, Johnson, 1954

Traditional Solvers

- LP problems can be solved in $O(n^{2.5})$ but all possible sets S makes the problem intractable.
- Candidates S that violate the sub-group constraint must be identified and added to the LP problem to get a valid tour.

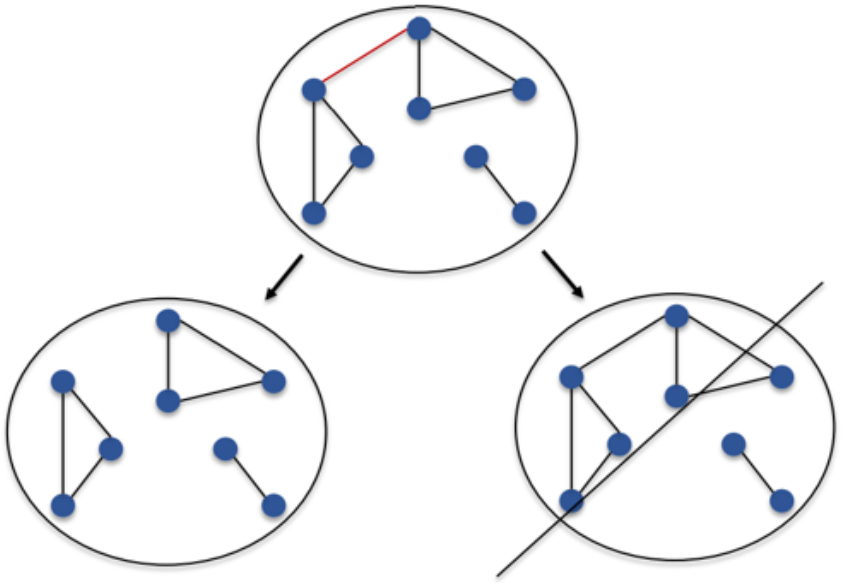
- This leads to the Cutting Planes technique, which iteratively solves LP problems by adding linear inequality constraints.



Traditional Solvers

- Solving a LP problem does not guarantee a discrete solution $x_e \in \{0,1\}$ and continuous values lead to a choice to select or not the edge in the tour (hence changing the candidates S).
- This leads to a tree of possible solutions, and the branch-and-bound technique[1] discards branches of the search space that provably do not contain an optimal solution.

- Overall complexity for Concorde/Gurobi is $O(n^{2.5}b(n))$, with $O(n^{2.5})$ for LP and $O(b(n))$ for the number of branches to explore.

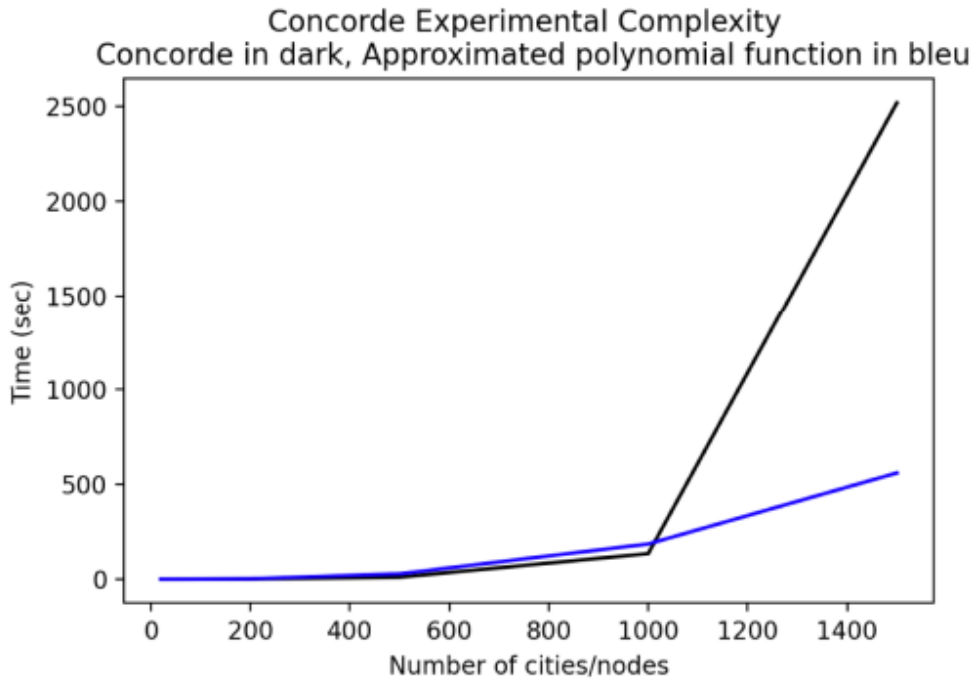


Discarded solution with a upper bound solution (for discrete x) and lower bound (for continuous x).

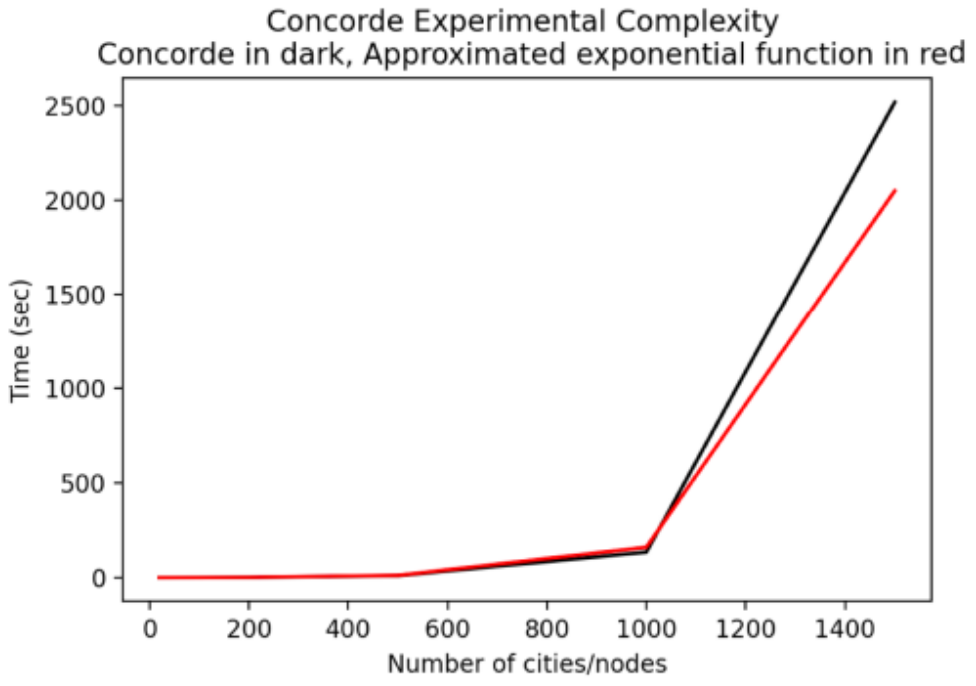
[1] Bellman, Held, Karp, 1962

Traditional Solvers

- Concorde[1] Complexity
  - Experimental complexity :



Polynomial complexity
$O(n^{2.72})$

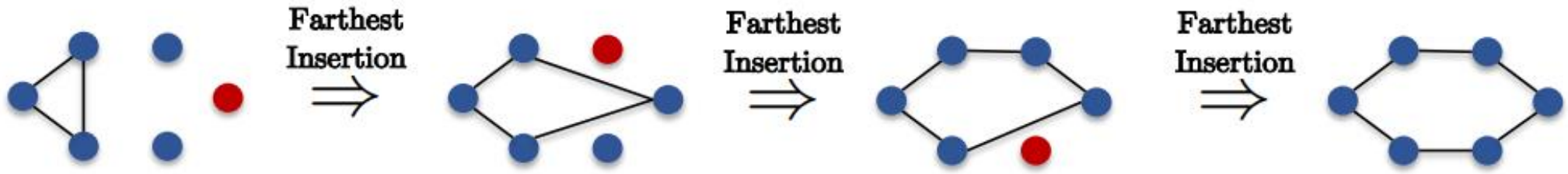


Exponential complexity
$O(e^{0.005n})$

[1] Applegate, Bixby, Chvatal, Cook, The Traveling Salesman Problem, 2006

Traditional Solvers: Heuristic Solvers

- Heuristic Solvers
  - Approximate/heuristic algorithms :
    - Christofides algorithm[1]: Approximation based on minimum spanning tree. Polynomial time algorithm with $\boldsymbol{O(n^2\ logn)}$ that is guaranteed to find a solution within a factor $\mathbf{1.5}$ of the optimal solution.
    - Farthest/nearest/greedy insertion algorithm[2]: Complexity is $\boldsymbol{O(n^2)}$ and the approximation ratio is $\mathbf{2.43}$ for farthest insertion (best insertion in practice).



[1] Christofides, Worst-case analysis of a new heuristic for the travelling salesman problem, 1976

[2] Johnson, Local Optimization and the Traveling Salesman Problem, 1990

Traditional Solvers

- Approximate/heuristic algorithms :
  - 2-Opt algorithm[1,2]: Heuristic is based on a move that replaces two edges to reduce the tour length. Complexity is $\boldsymbol{O(n^2 m(n))}$, where n^2 is the number of node pairs and $m(n)$ is the number of times all pairs must be tested to reach a local minimum with worst-case being $O(2^{n/2})$.
  - The approximation ratio is then $\boldsymbol{4/\sqrt{n}}$.
  - Extension to 3-Opt move (replacing 3 edges) and more[3].

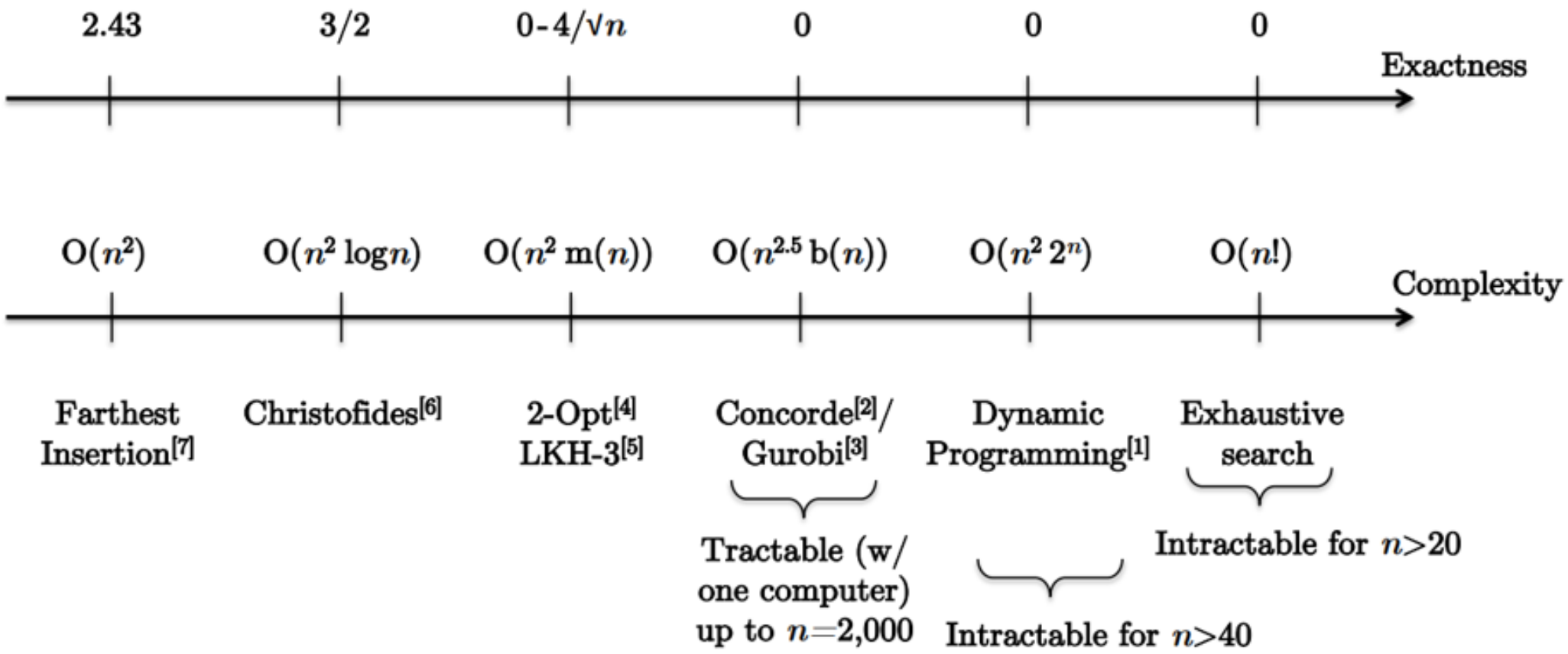
[1] Lin, Computer solutions of the traveling salesman problem, 1965

[2] Johnson, McGeoch, The traveling salesman problem: A case study in local optimization, 1995

[3] Blazinskas, Misevicius, Combining 2-Opt, 3-Opt and 4-Opt with K-SWAP-KICK perturbations for the traveling salesman problem, 2011

Traditional Solvers

- Hierarchy of traditional TSP algorithms:



Outline

- History of TSP
- Traditional Solvers
  - Exact algorithms
  - Heuristic solvers
- Neural Network Solvers
  - Transformer based methods
  - GCN based methods
- Numerical Experiments

Neural Network Solvers

- Deep learning (DL) has significantly improved Computer Vision, Natural Language Processing and Speech Recognition by replacing hand-crafted visual/text/speech features with features learned from data.

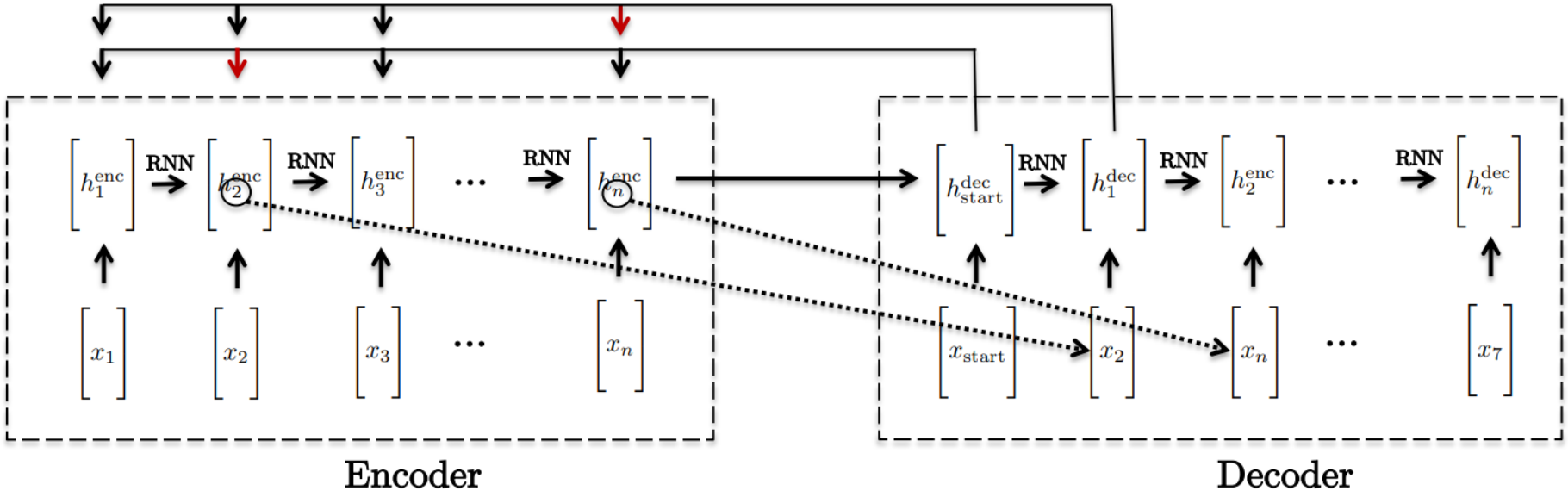
- For combinatorial problems, the main question is whether DL can learn better heuristics from data, i.e. replacing human-engineered heuristics.
  - This is appealing because developing algorithms to tackle efficiently NP-hard problems may require years of research.
  - TSP has been actively studied for 70 years.
  - Besides, many industry problems are combinatorial by nature.

- Hopfield Nets[1]: First NN to solve (small) TSPs.

[1] Hopfield, Tank, Neural computation of decisions in optimization problems, 1985

Neural Network Solvers

- (Graph) neural networks have been capable to learn new combinatorial algorithms with supervised or reinforcement learning.
- **PointerNets**[1]: use DL to tackle TSP and CO problems.
- Combine recurrent networks to encode the cities and decode the node sequence of the tour (auto-regressive decoding), and attention mechanism.
- Supervised learning with approximate TSP solutions.



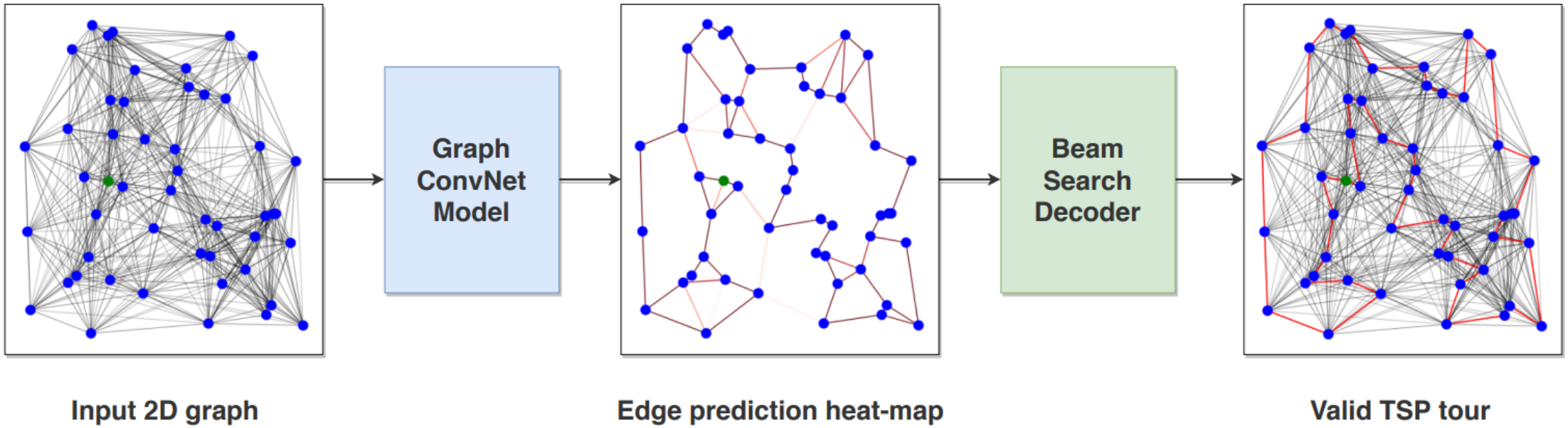
[1] Vinyals, Fortunato, Jaitly, Pointer networks, 2015

Neural Network Solvers

- The DL solvers are all approximate/heuristic methods.
  - They are either supervised learning with approximate TSP solutions (ex. TSP50, TSP100) or improved with Reinforcement Learning (no TSP solutions required).

- The recent DL solvers can be mainly divided into the following categories:
  - The GNN-based methods
    - Graph network is suitable to handle the information of city location
  - The transformer-based methods
    - TSP can be regarded as a “translation” problem
    - The translation problem has seen significant progress with the Transformers

GNN based Methods



- Train a graph ConvNet model to output an adjacency matrix corresponding to a TSP tour[1].

[1] Joshi, Laurent, Bresson, An Efficient Graph Convolutional Network Technique for the Travelling Salesman Problem, 2019

GNN based Methods

- Input layer
  - $x_i \in [0,1]^2$, embedded features $\alpha_i = A_1 x_i + b_1$
  - Euclidean distance d_{ij} and k-nearest indicator function δ_{ij}^{k-NN}, embedded features $\beta_{ij} = A_2 d_{ij} + b_2 \,||\, A_3 \delta_{ij}^{k-NN}$
- Graph Convolution layer

$$x_i^{\ell+1} = x_i^{\ell} + \mathrm{ReLU}\left(\mathrm{BN}\left(W_1^{\ell} x_i^{\ell} + \sum_{j\sim i} \eta_{ij}^{\ell} \odot W_2^{\ell} x_j^{\ell}\right)\right) \text{ with } \eta_{ij}^{\ell} = \frac{\sigma(e_{ij}^{\ell})}{\sum_{j'\sim i} \sigma(e_{ij'}^{\ell}) + \varepsilon},$$

$$e_{ij}^{\ell+1} = e_{ij}^{\ell} + \mathrm{ReLU}\left(\mathrm{BN}\left(W_3^{\ell} e_{ij}^{\ell} + W_4^{\ell} x_i^{\ell} + W_5^{\ell} x_j^{\ell}\right)\right),$$

- MLP classifier

$$p_{ij}^{\mathrm{TSP}} = \mathrm{MLP}(e_{ij}^{L})$$

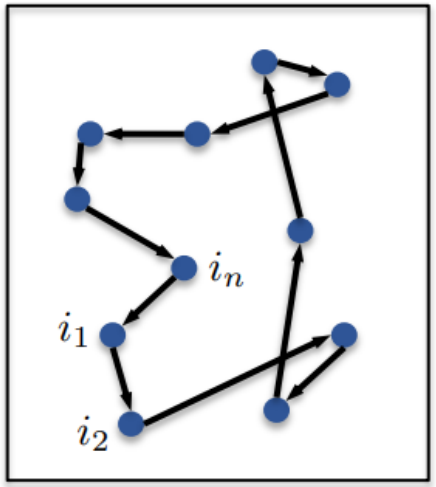
- Loss function
  - Minimize the cross-entropy loss between the ground-truth TSP tour permutation π and output probabilistic heatmap P.
- Apply beam search decoding to yield the TSP tour.

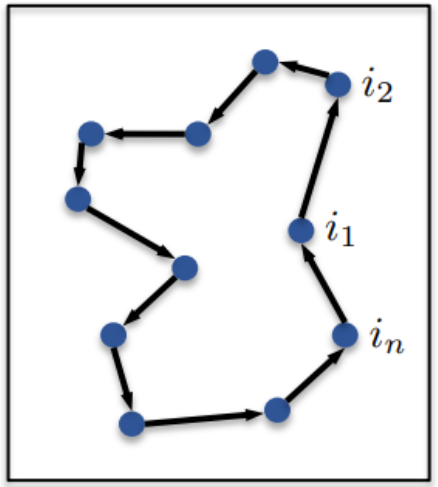
Tour Decoding

- A tour is represented as an ordered sequence of city indices: $seq_n = \{i_1, i_2, \dots, i_n\}$
- TSP can be cast as a sequence optimization problem:

$$\max_{seq_n=\{i_1,\dots,i_n\}} P^{TSP}(seq_n|x) = P^{TSP}(i_1, \dots, i_n|x)$$

$P^{TSP}(\cdot)$ is the probability of the sequence to be a solution of the TSP given a set x of n 2-D points.





$P^{\text{TSP}}(\text{seq}_n^1|x)$ < $P^{\text{TSP}}(\text{seq}_n^2|x)$

Tour Decoding

- For auto-regressive decoding, i.e. selecting a city one at a time, P^{TSP} can be factorized with the chain rule :

$$P^{\text{TSP}}(i_1, ..., i_n|x) = \underbrace{P(i_1|x)}_{\substack{\text{Probability of selecting city } i_1 \\ \text{given } x}} . \underbrace{P(i_2|i_1, x)}_{\substack{\text{Probability of selecting city } i_2 \\ \text{given } i_1, x}} . \underbrace{P(i_3|i_2, i_1, x)}_{\substack{\text{Probability of selecting city } i_3 \\ \text{given } i_2, i_1, x}} ... \underbrace{P(i_n|i_{n-1}, i_{n-2}, ..., x)}_{\substack{\text{Probability of selecting city } i_n \\ \text{given } i_{n-1}, i_{n-2}, ..., x}}$$

- The decoding problem aims at finding the sequence $i_1, i_2, ..., i_n$ that maximizes the objective:

$$\max_{i_1,...,i_n} \ \Pi_{t=1}^{n} \ P(i_t|i_{t-1}, i_{t-2}, ...i_1, x)$$

Conditional probability approximated by a neural network

Tour Decoding: Decoding Techniques

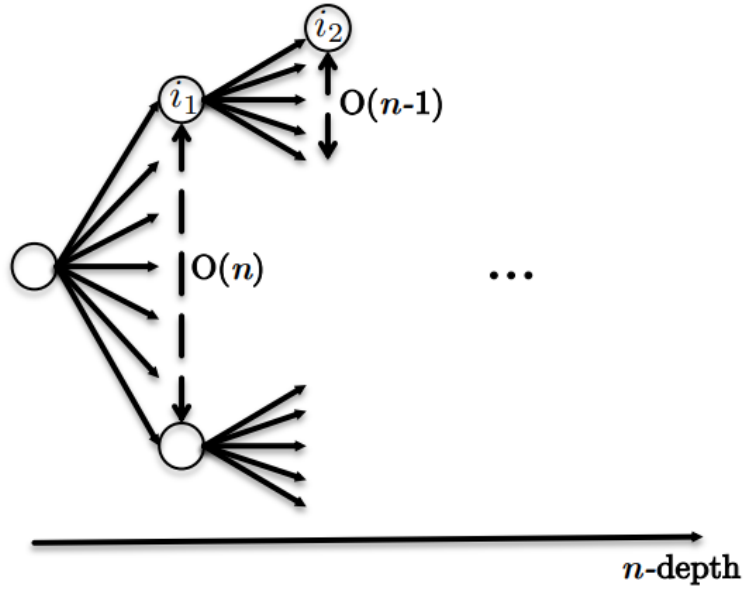
- Exact search: Exhaustive search is intractable with $O(n!)$
- Greedy search: At each time step, select the next city with the highest probability:

$$i_t = \arg\max_i \; P(i|i_{t-1}, i_{t-2}, \ldots i_1, x)$$

  Complexity is $O(n)$

- Sampling techniques such as beam search or Monte Carlo Tree Search (MTCS) are known to improve results in NLP and TSP.

  Complexity is $O(Bn)$.



Tour Decoding

- Beam Search[1]
  - Beam search is a breadth-first search (BFS) technique where the breath has a limited size B.
  - Beam search decoding problem :

$$\max_{\{i_1^b,\ldots,i_n^b\}_{b=1}^{B}} \Pi_{b=1}^{B} P(i_1^b,\ldots,i_n^b|x) \quad \text{s.t.} \quad \{i_1^b,\ldots,i_n^b\} \neq \{i_1^{b'},\ldots,i_n^{b'}\},\ \forall b \neq b'$$

B sequences/ beams

Two beams are different.

  - For B=1, the solution is given by greedy decoding

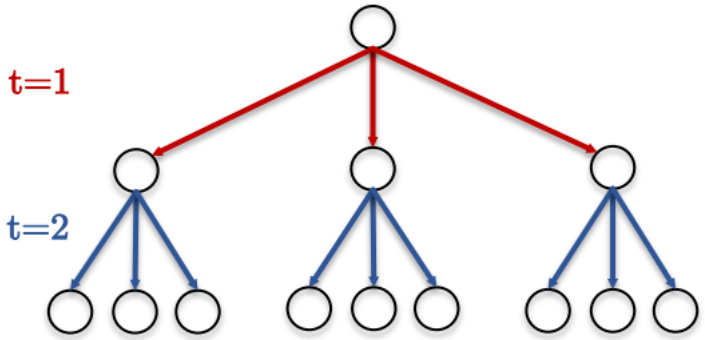
Tour Decoding

- For $B > 1$, the solution at t is determined by considering all possible extensions of B beams, and only keeping the Top-B probabilities:

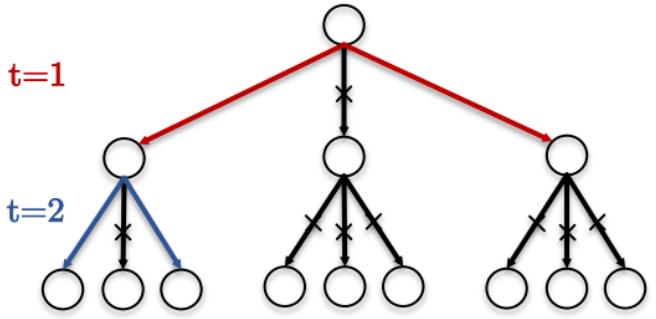
$$\{i_1^b, ..., i_t^b\}_{b=1}^{B} = \text{Top-B}\left\{\Pi_{k=1}^{t} P(i_k^b | i_{k-1}^b, i_{k-2}^b, ..., i_1^b, x)\right\}_{b=1}^{B.(n-t)}$$

Or equivalently (for better numerical stabilities) :

$$\{i_1^b, ..., i_t^b\}_{b=1}^{B} = \text{Top-B}\left\{\sum_{k=1}^{t} \underbrace{\log P(i_k^b | i_{k-1}^b, i_{k-2}^b, ..., i_1^b, x)}_{\text{Score } s(i_k | i_{k-1}, i_{k-2}, ..., i_1, x)}\right\}_{b=1}^{B.(n-t)}$$



BFS explores the search space by expanding to all children nodes first.



Beam Search explores the search space by expanding to a limited set of children nodes selected by a criteria. Beam size is B=2.

Other GNN based Methods

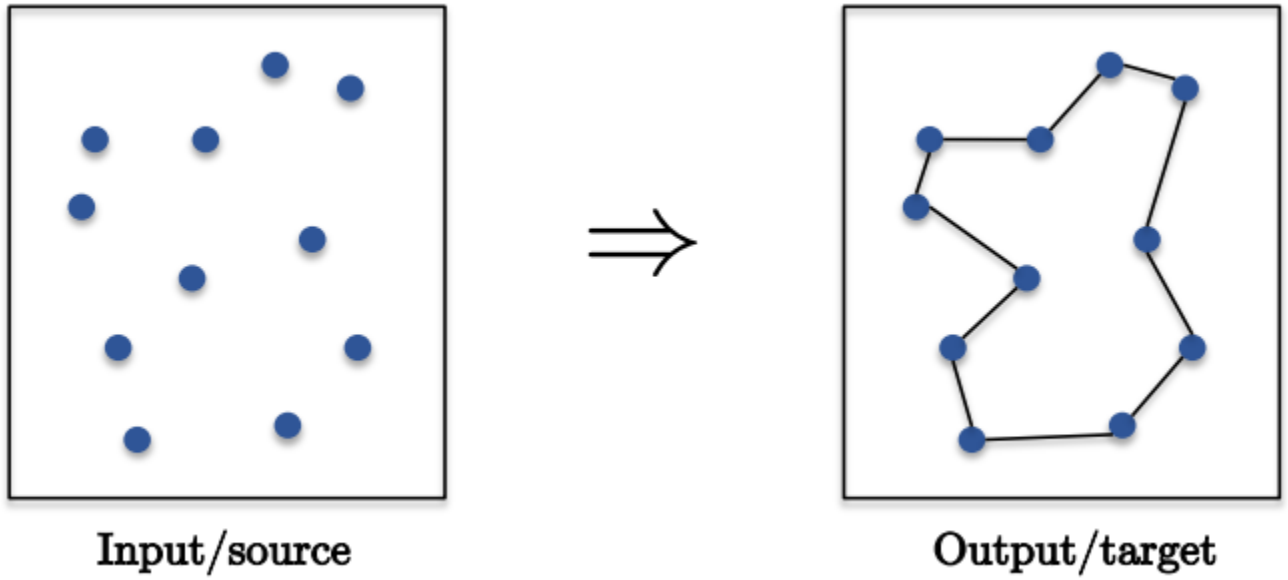
- Quadratic Assignment Problem[1]: TSP can be formulated as a QAP, which is NP-hard and hard to approximate.
  - A graph network based on the powers of the adjacency matrix of node distances is trained in a supervised manner.
  - The loss is the KL distance between the adjacency matrix of the ground truth cycle and its network prediction.
  - A feasible tour is computed with beam search.
- GNNs with Monte Carlo Tree Search[2]: Inspired by AlphaGo, augment graph network with MCTS to improve the search exploration of tours by evaluating multiple next node candidates in the tour (auto-regressive methods cannot go back to the selection of the nodes).

[1] Nowak, Villar, Bandeira, Bruna, A Note on Learning Algorithms for Quadratic Assignment with Graph Neural Networks, 2017

[2] Xing, Tu, A Graph Neural Network Assisted Monte Carlo Tree Search Approach to Traveling Salesman Problem, 2020

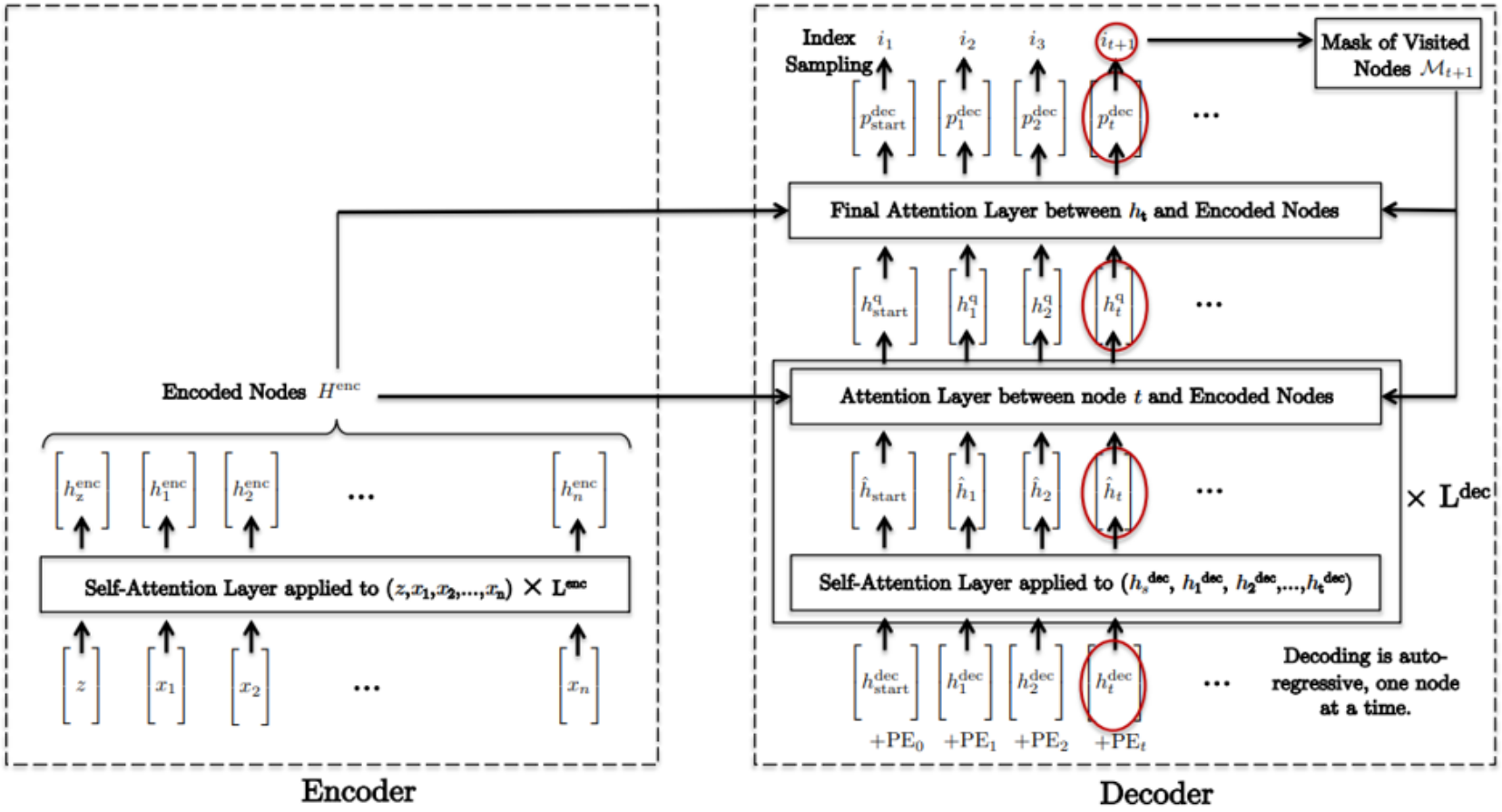
Transformer based Methods

- TSP can be regarded as a “translation” problem:
  - Source is a set of 2D points
  - Target is a tour (sequence of indices) with minimal length
- The translation problem has seen significant progress with Transformers[1].



Transformer based Methods

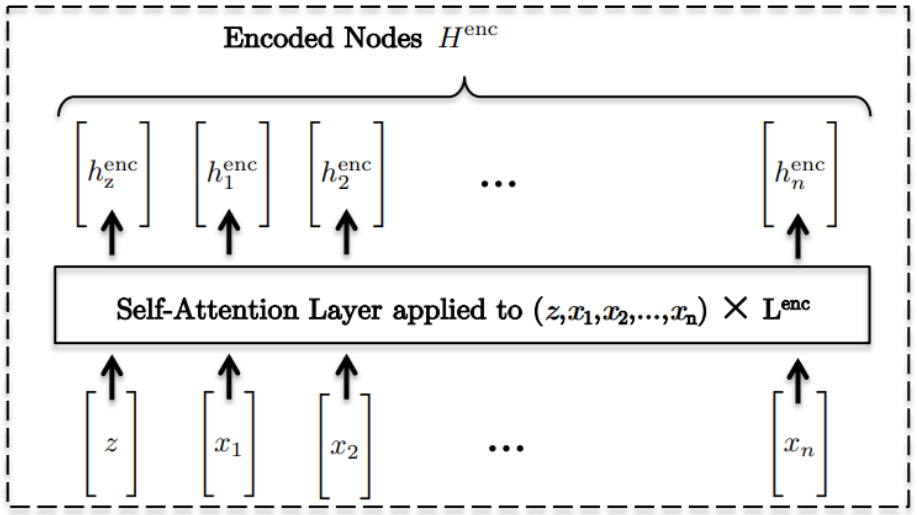
- Bresson et al.'s[1] Architecture



[1] Bresson X, Laurent T. The transformer network for the traveling salesman problem[J]. arXiv preprint arXiv:2103.03012, 2021.

Transformer based Methods

- Encoder:
  - Standard Transformer encoder:
    - Multi-head attention, residual connection, batch normalization are used
    - Memory/speed complexity is $\boldsymbol{O(n^2)}$



$$H^{\text{enc}} = H^{\ell=L^{\text{enc}}} \in \mathbb{R}^{(n+1)\times d}$$

where

Start token, initialized at random.

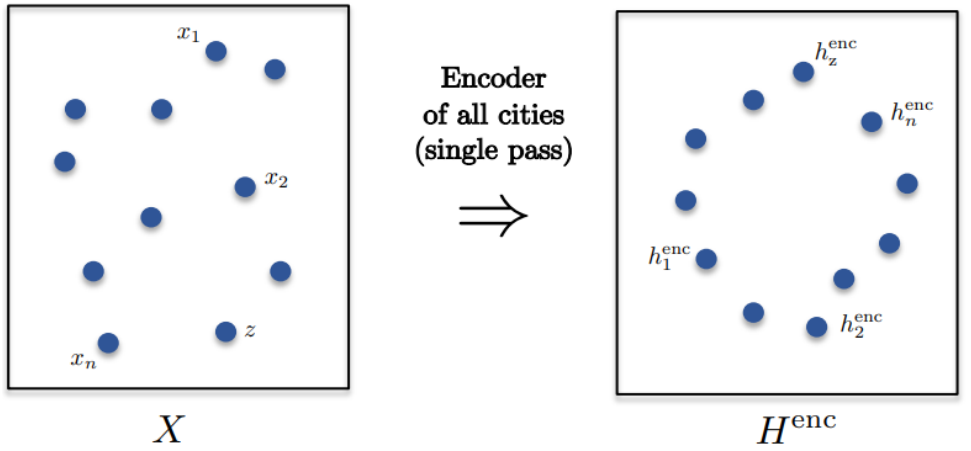
$$H^{\ell=0} = \text{Concat}(z, X) \in \mathbb{R}^{(n+1)\times 2}, z \in \mathbb{R}^2, X \in \mathbb{R}^{n\times 2}$$

$$H^{\ell+1} = \text{softmax}(\frac{Q^\ell {K^\ell}^T}{\sqrt{d}})V^\ell \in \mathbb{R}^{(n+1)\times d},$$

$$Q^\ell = H^\ell W_Q^\ell \in \mathbb{R}^{(n+1)\times d}$$

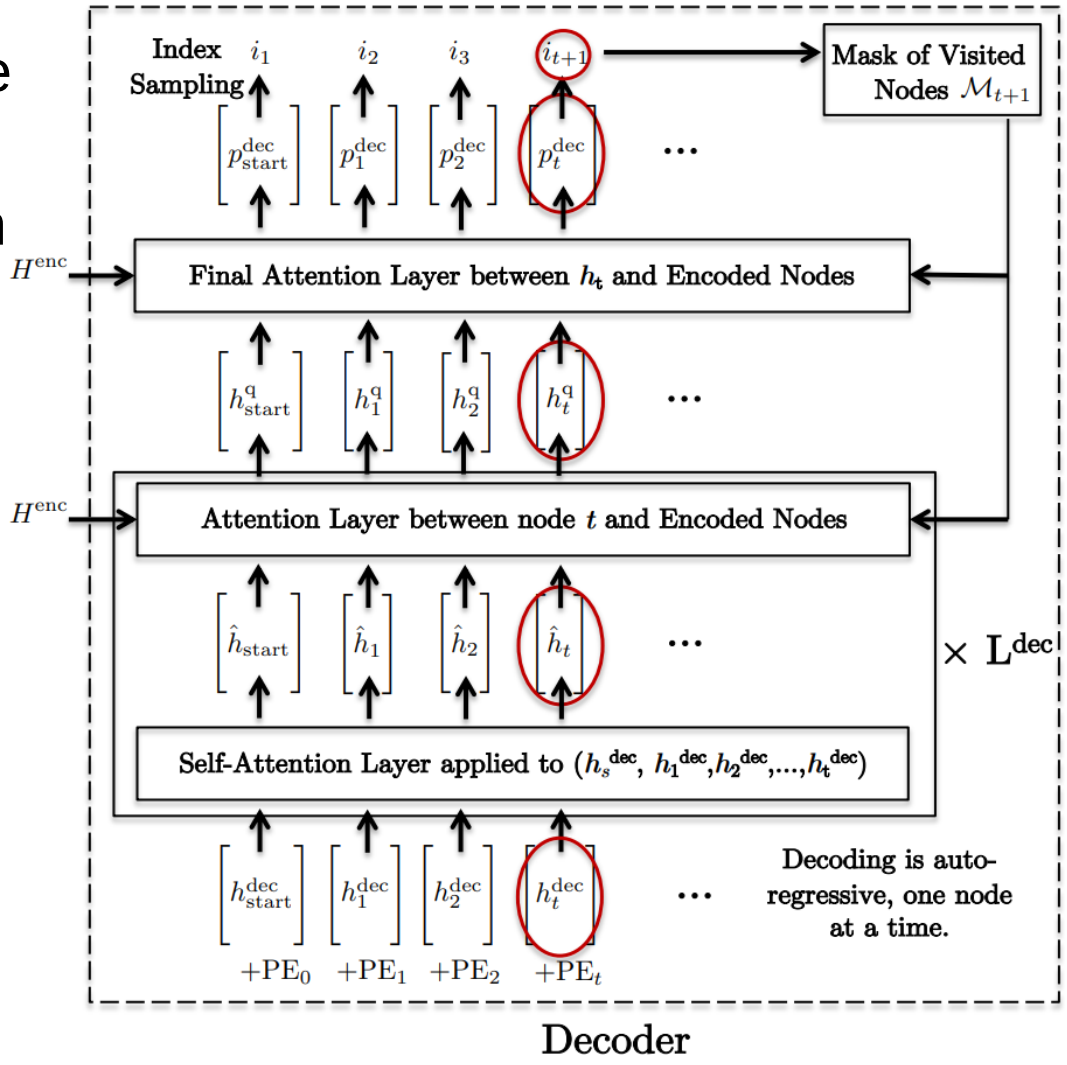
$$K^\ell = H^\ell W_K^\ell \in \mathbb{R}^{(n+1)\times d}$$

$$V^\ell = H^\ell W_V^\ell \in \mathbb{R}^{(n+1)\times d}$$



Transformer based Methods

- Decoder
  - Decoding is auto-regressive, one city at a time
  - Suppose the first t cities in the tour have been decoded, then to predict the next city
  - The decoding process has four steps:
    - Part 1: Start with encoding of the i_t city + positional encoding
    - Part 2: Encode t^{th} city with the partial tour using self-attention
    - Part 3: Query the next city with the non-visited cities using multi-head attention.
    - Part 4: Final query using a single-head attention + index sampling.



Transformer based Methods

- Part 1: Start decoding with the encoding of the previously selected i_t city:

$$h_t^{\text{dec}} = h_{i_t}^{\text{enc}} + \text{PE}_t \in \mathbb{R}^d$$

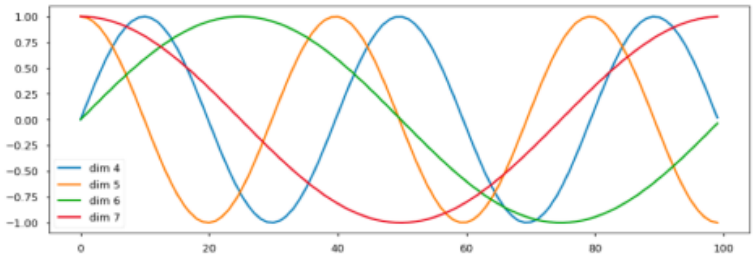
$$h_{t=0}^{\text{dec}} = h_{\text{start}}^{\text{dec}} + \text{PE}_{t=0} \in \mathbb{R}^d, \text{ with } h_{\text{start}}^{\text{dec}} = h_{\text{z}}^{\text{enc}}$$

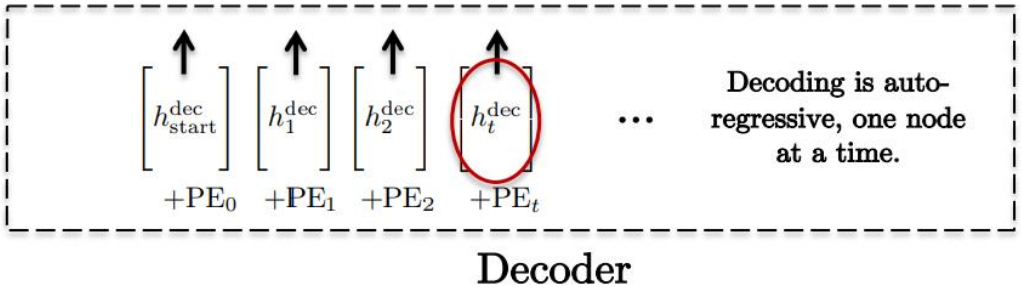
  - Add positional encoding (PE) to order the nodes in the tour :

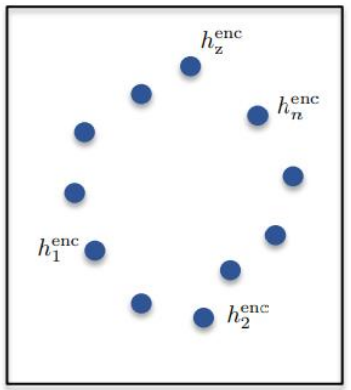
$$\text{PE}_t \in \mathbb{R}^d$$

$$\text{PE}_{t,i} = \begin{cases} \sin(2\pi f_i t) \text{ if } i \text{ is even,} \\ \cos(2\pi f_i t) \text{ if } i \text{ is odd,} \end{cases}$$

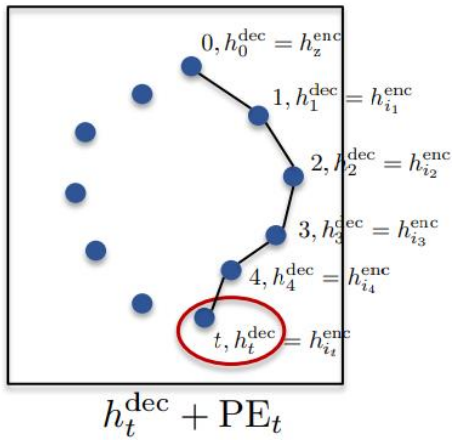
$$\text{with } f_i = \frac{10,000^{\frac{d}{\lfloor 2i \rfloor}}}{2\pi}.$$







Start with previously selected city + add PE



Transformer based Methods

- Part 2: Encode t^{th} city with the partial tour using self-attention.
  - Multi-head attention, residual connection, layer normalization are used.
  - Memory/speed complexity is $O(t)$ at each recursive step.

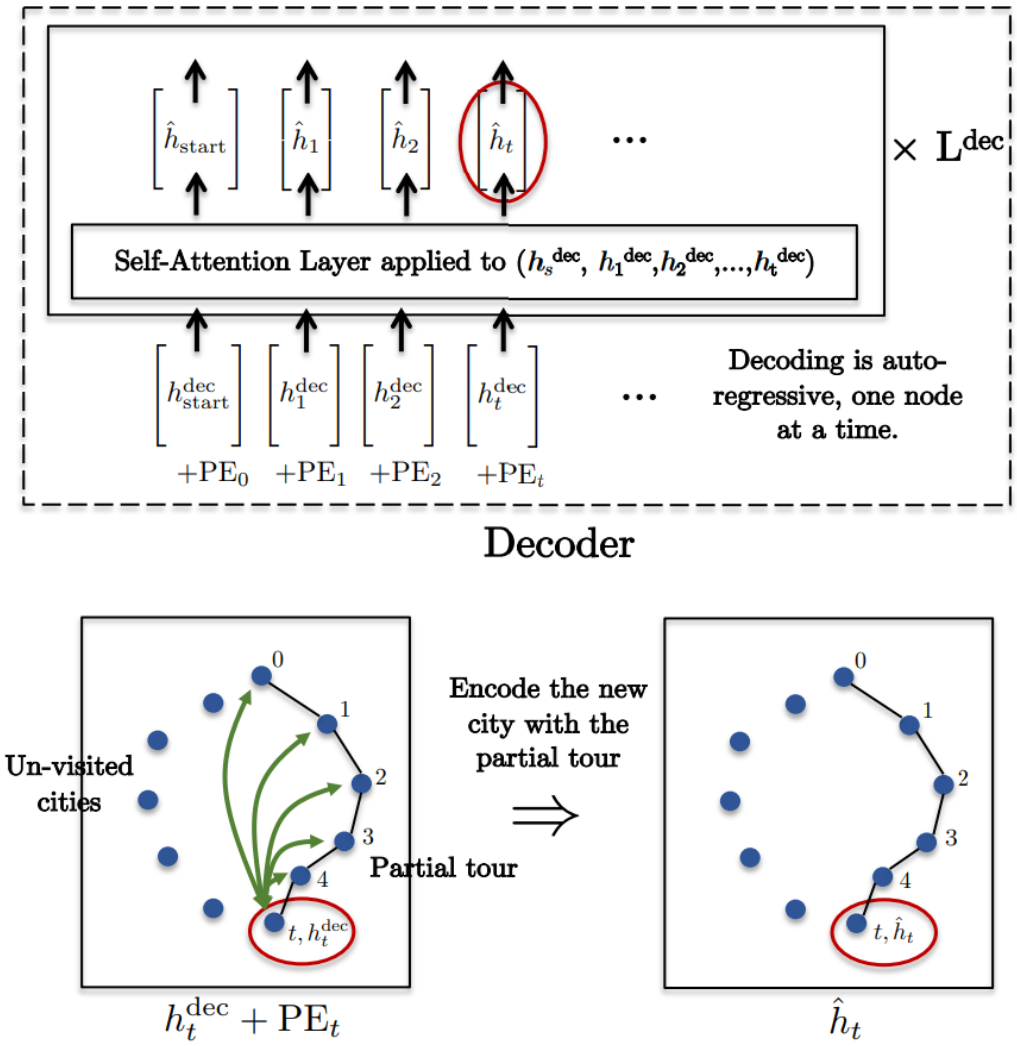
$$\hat{h}_t^{\ell+1} = \text{softmax}(\frac{q^\ell {K^\ell}^T}{\sqrt{d}})V^\ell \in \mathbb{R}^d,\ \ell = 0, ..., L^{\text{dec}} - 1$$

$$q^\ell = \hat{h}_t^\ell \hat{W}_q^\ell \in \mathbb{R}^d$$

$$K^\ell = \hat{H}_{1,t}^\ell \hat{W}_K^\ell \in \mathbb{R}^{t \times d}$$

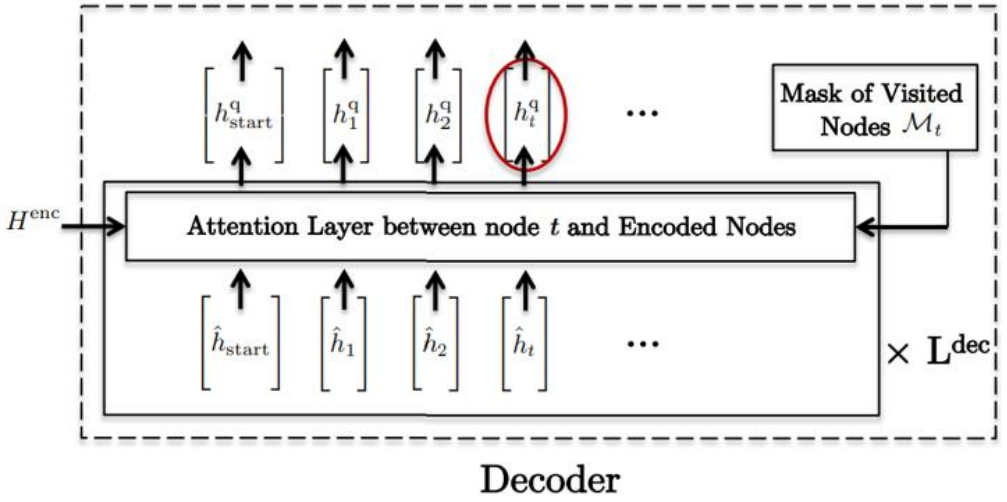
$$V^\ell = \hat{H}_{1,t}^\ell \hat{W}_V^\ell \in \mathbb{R}^{t \times d}$$

$$\hat{H}_{1,t}^\ell = [\hat{h}_1^\ell, .., \hat{h}_t^\ell],\ \hat{h}_t^\ell = \begin{cases} h_t^{\text{dec}} \text{ if } \ell = 0 \\ h_t^{\text{q},\ell} \text{ if } \ell > 0 \end{cases}$$



Transformer based Methods

- Part 3: Query the next city with the nonvisited cities using attention.
  - Multi-head attention, residual connection, layer normalization are used.
  - Memory/speed complexity is $O(n)$ at each recursive step.

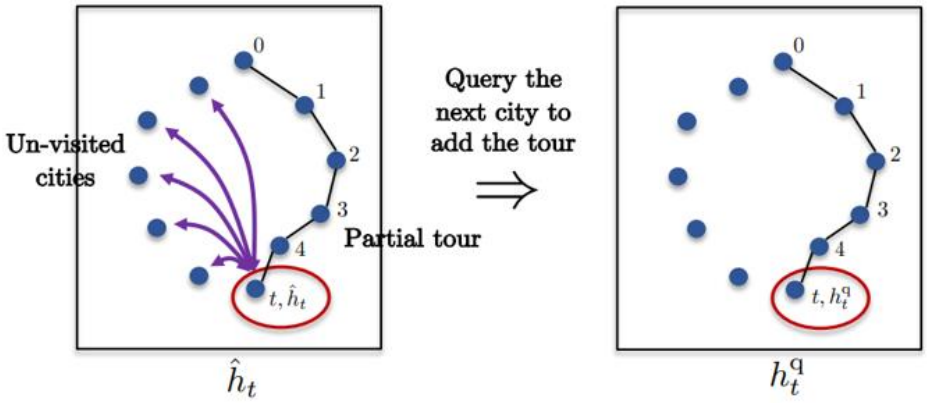


$$h_t^{q,\ell+1} = \text{softmax}\left(\frac{q^\ell {K^\ell}^T}{\sqrt{d}} \odot \mathcal{M}_t\right) V^\ell \in \mathbb{R}^d,\ \ell = 0, ..., L^{\text{dec}} - 1$$

$$q^\ell = \hat{h}_t^{\ell+1} \tilde{W}_q^\ell \in \mathbb{R}^d$$

$$K^\ell = H^{\text{enc}} \tilde{W}_K^\ell \in \mathbb{R}^{t \times d}$$

$$V^\ell = H^{\text{enc}} \tilde{W}_V^\ell \in \mathbb{R}^{t \times d}$$



Transformer based Methods

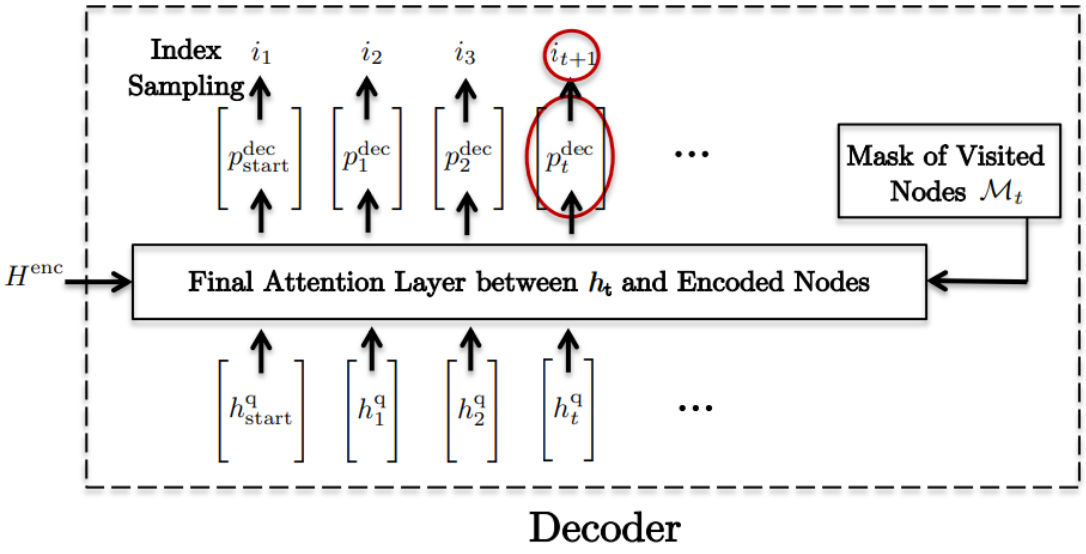
- Part 4: Final query using attention + index sampling
  - Single-head attention is used to get a distribution over the non-visited cities.
  - Finally, the next node i_{t+1} is sampled from the distribution using Bernoulli during training and greedy (index with max probability) at inference time.
  - Memory/speed complexity is $O(n)$.

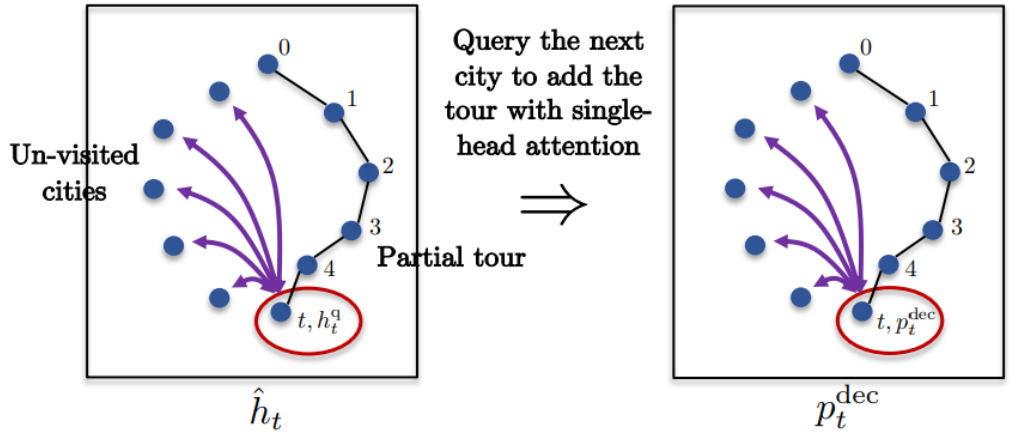
$$p_t^{\text{dec}} = \text{softmax}(C \tanh(\frac{qK^T}{\sqrt{d}} \odot \mathcal{M}_t)) \in \mathbb{R}^n,$$

with

$$q = h_t^{\text{q}} \bar{W}_q \in \mathbb{R}^d$$

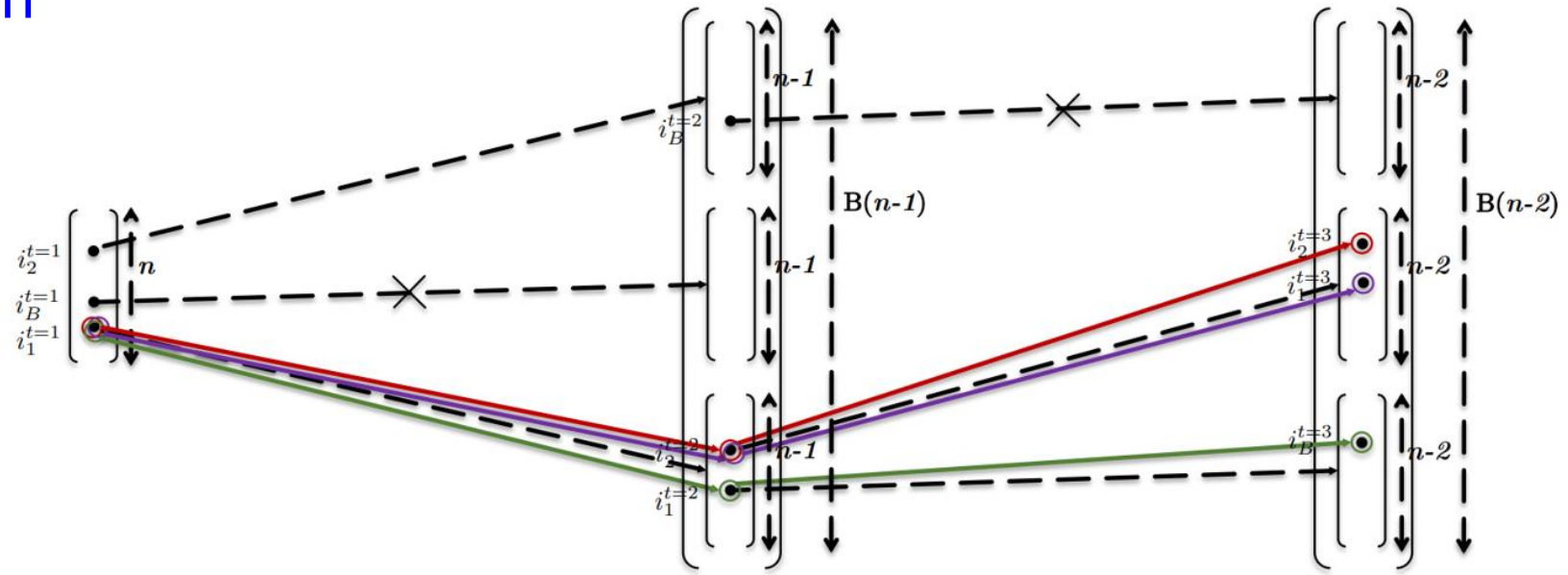
$$K = H^{\text{enc}} \bar{W}_K \in \mathbb{R}^{n \times d}$$



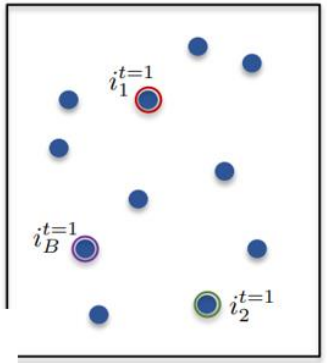


Transformer based Methods: Beam Search

- Beam Search

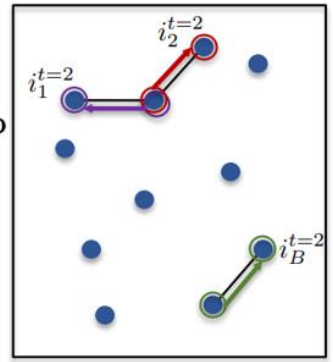


At t=1, run the net and get the starting cities with the top-B scores.



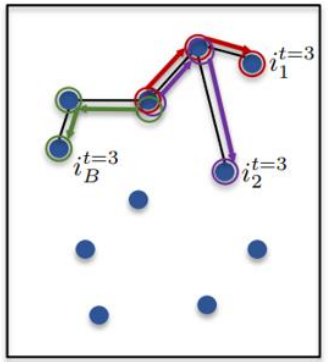
At t=2, run the net for each beam that can expand to B(n-1) possible cities. Keep the beams with top-B scores:

$s(i_2|i_1, x) + s(i_1|x)$



At t=3, run the net for each beam that can expand to B(n-2) possible cities. Keep the beams with top-B scores:

$s(i_3|i_2, i_1, x) +$
$s(i_2|i_1, x) + s(i_1|x)$



Transformers for NLP (translation) vs TSP (combinatorial optimization)

- Order of input sequence is not relevant for TSP
  - Order of output sequence is coded with PEs for both TSP and NLP

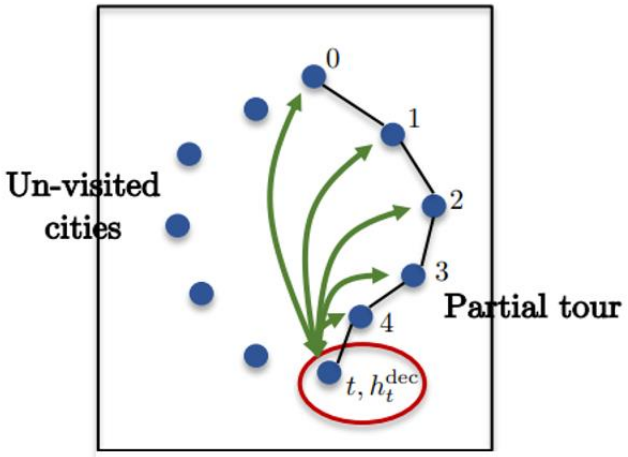
- TSP-Encoder benefits from Batch Normalization
  - TSP-Decoder uses Layer Normalization as with NLP (auto-regressive decoding)

- TSP transformer is learned by Reinforcement Learning (no TSP solutions/approximations required)

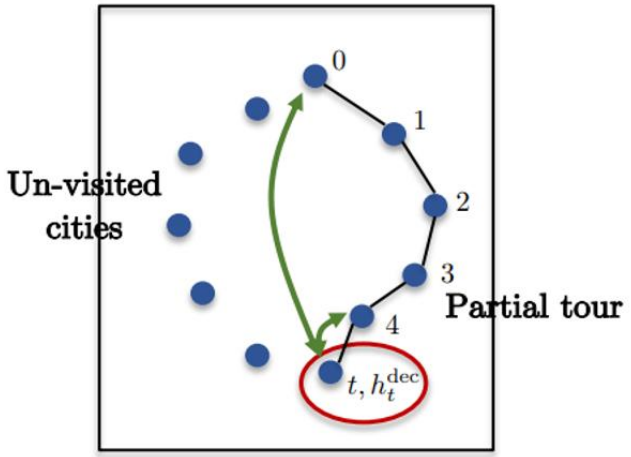
- Both transformers for NLP and TSP have quadratic complexity $O(n^2 L)$

Other Transformer based Methods

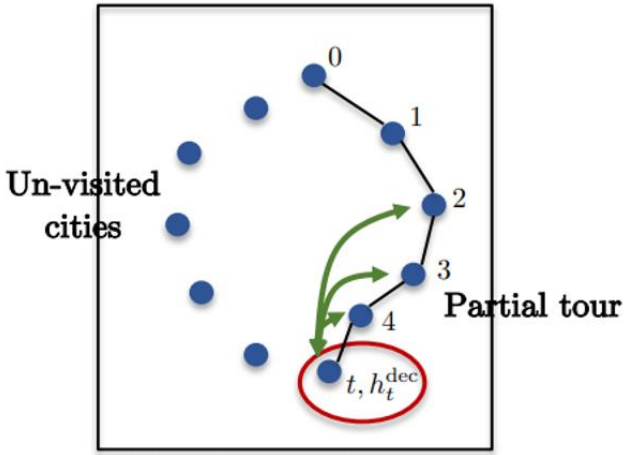
- Kool et al.[1] and Deudon et al.[2]
  - They use the same transformer-encoder (with BN).
  - The decoding architecture is different :
    - Bresson et al.'s query uses all cities in the partial tour with a self-attention module
    - Kool et al. use the first and last cities with a global representation of all cities as the query for the next city.
    - Deudon et al. define the query with the last three cities in the partial tour.



Bresson-etal's query



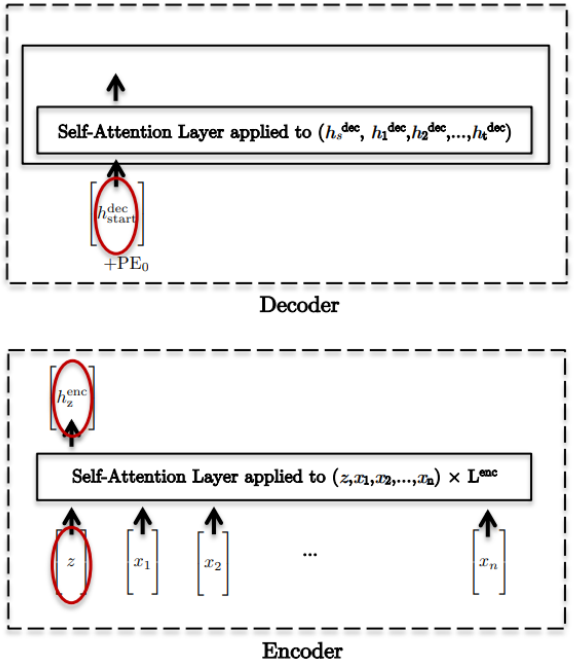
Kool-etal's query

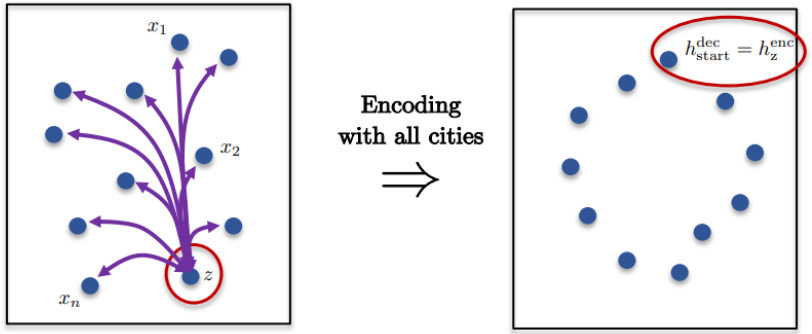


Deudon-etal's query

Other Transformer based Methods

- The decoding process starts differently:
  - Bresson et al. use a token city z.
  - This city does not exist and aims at starting the decoding at the best possible location by querying all cities with a self-attention module.



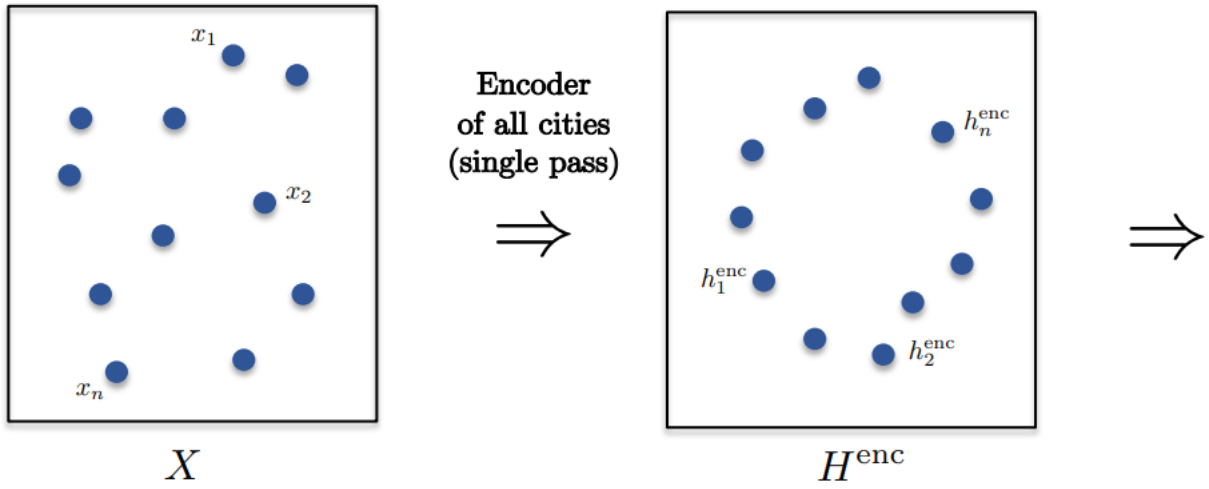


[1] Kool, Van Hoof, Welling, Attention, learn to solve routing problems!, 2018

[2] Deudon, Cournut, Lacoste, Adulyasak, Rousseau, Learning Heuristics for the TSP by Policy Gradient, 2018

Transformer based Methods

- Kool et al. start the decoding with the mean representation of the encoding cities and a random token of the first and current cities.
- Deudon et al. start the decoding with the mean representation of the encoding cities and a random token of the last three cities



Kool-etal's starting query :

$$h_{\text{start}}^{\text{dec}} = \text{Concat}(z_0, z_{t-1}, g),$$

$$\text{with } z_0, z_{t-1} \in \mathcal{N}_{0,1}^d, \; g = \frac{1}{n}\sum_{i=1}^{n} h_i^{\text{enc}} \in \mathbb{R}^d$$

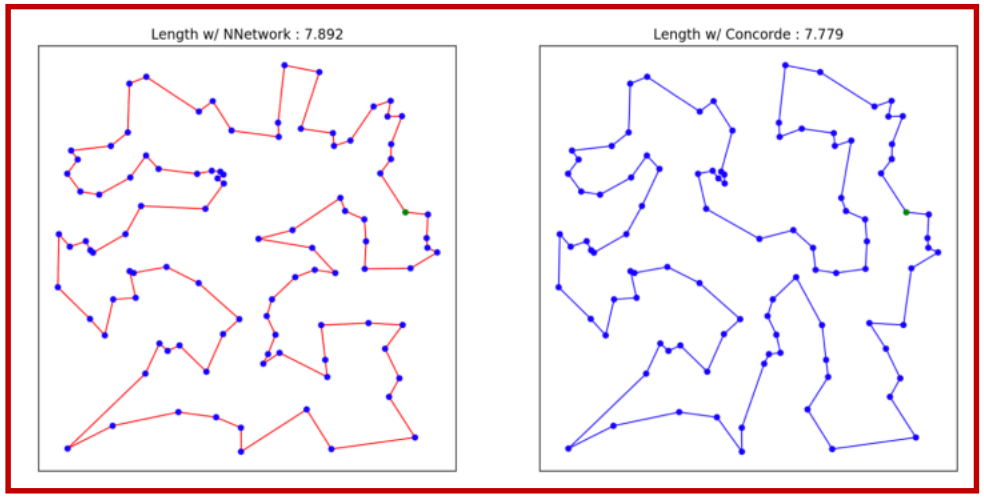
Deudon-etal's starting query :

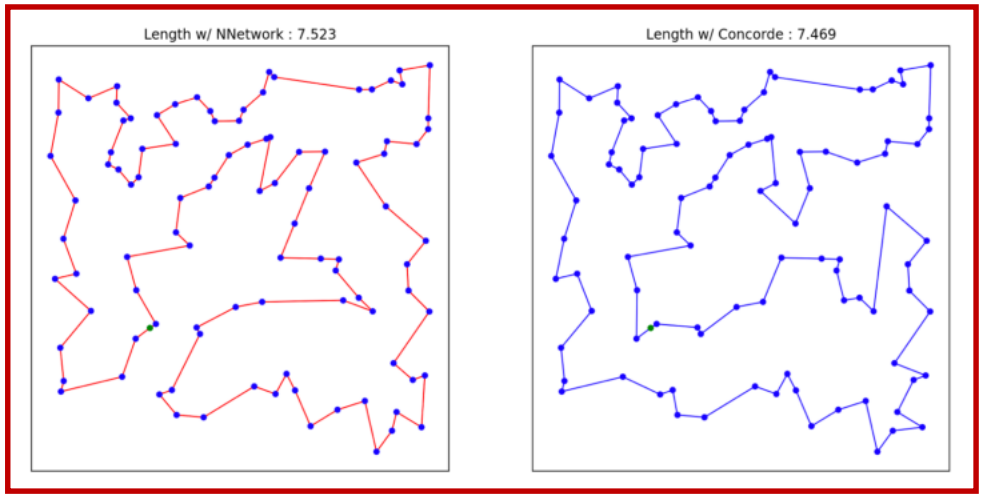
$$h_{\text{start}}^{\text{dec}} = \text{Concat}(z_{t-1}, z_{t-2}, z_{t-3}),$$

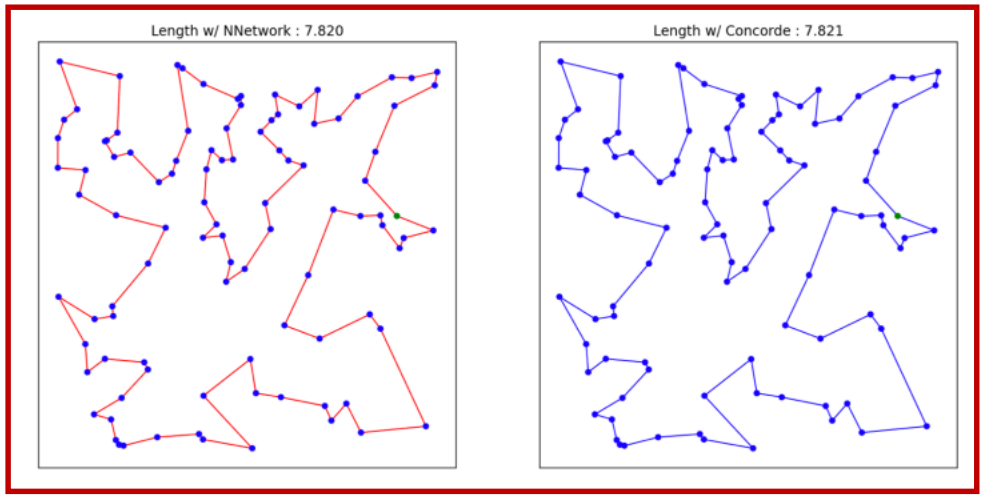
$$\text{with } z_{t-1}, z_{t-2}, z_{t-3} \in \mathcal{N}_{0,1}^d$$

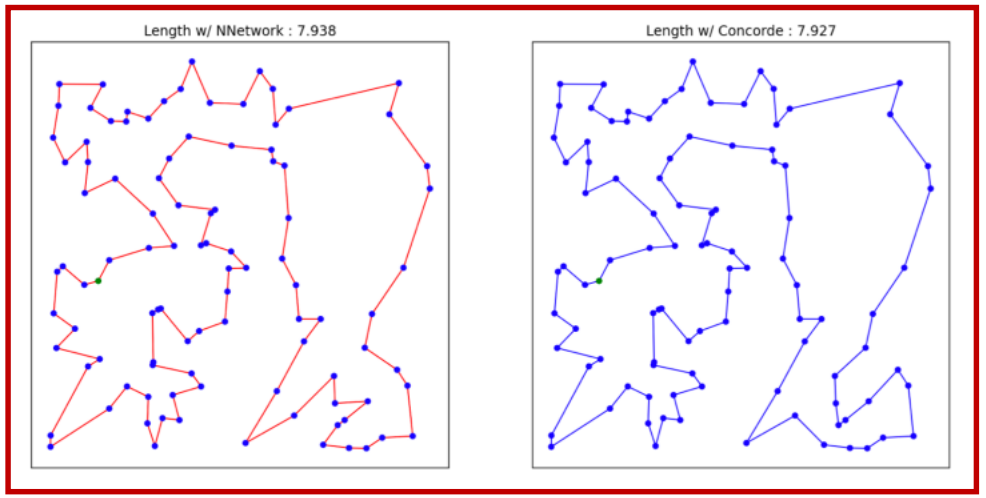
Numerical Experiments

- Bresson et al. technique vs. Concorde for TSP100





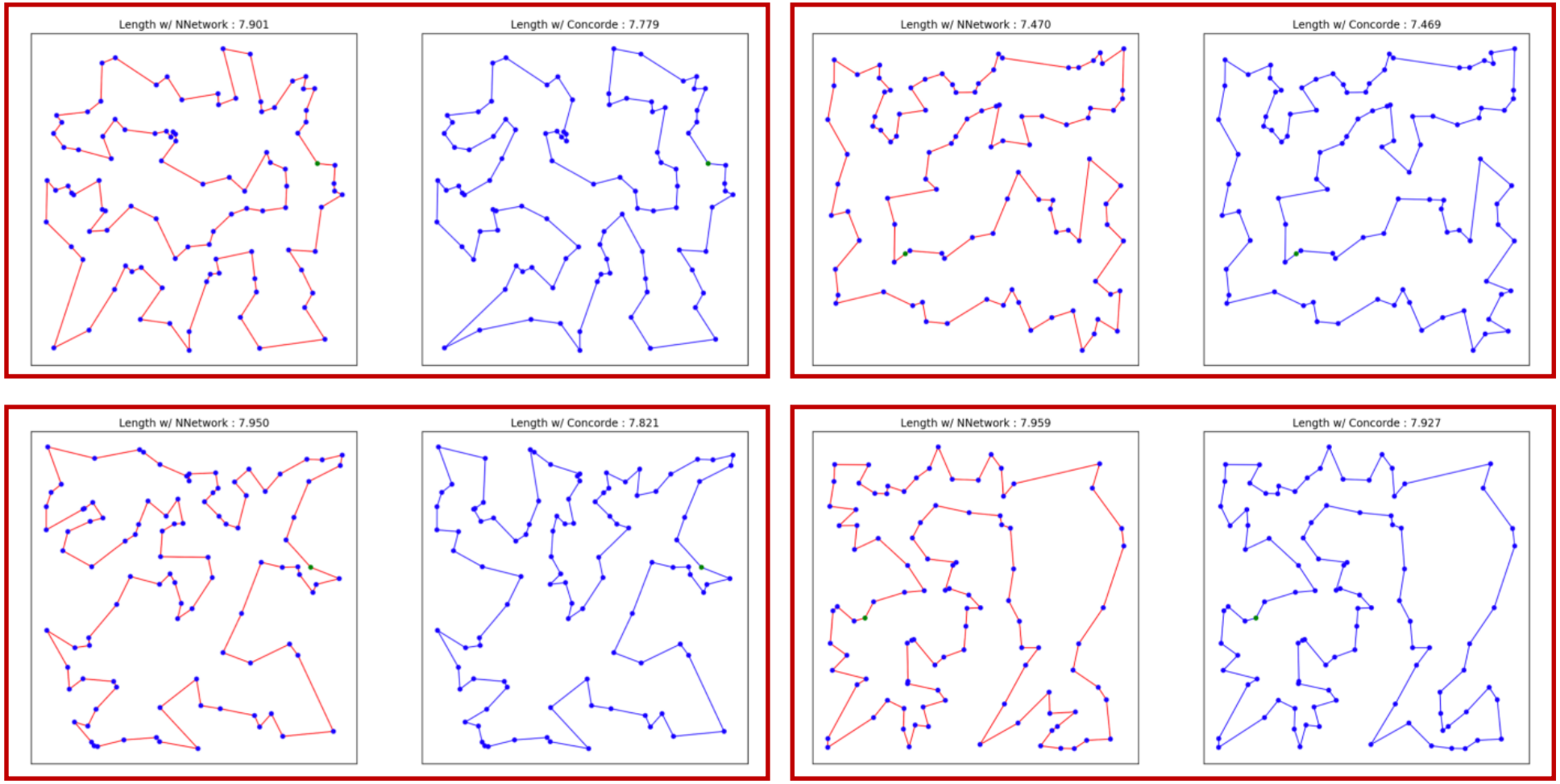




Numerical Experiments

- Bresson et al. technique vs. Concorde for TSP50⇒TSP100

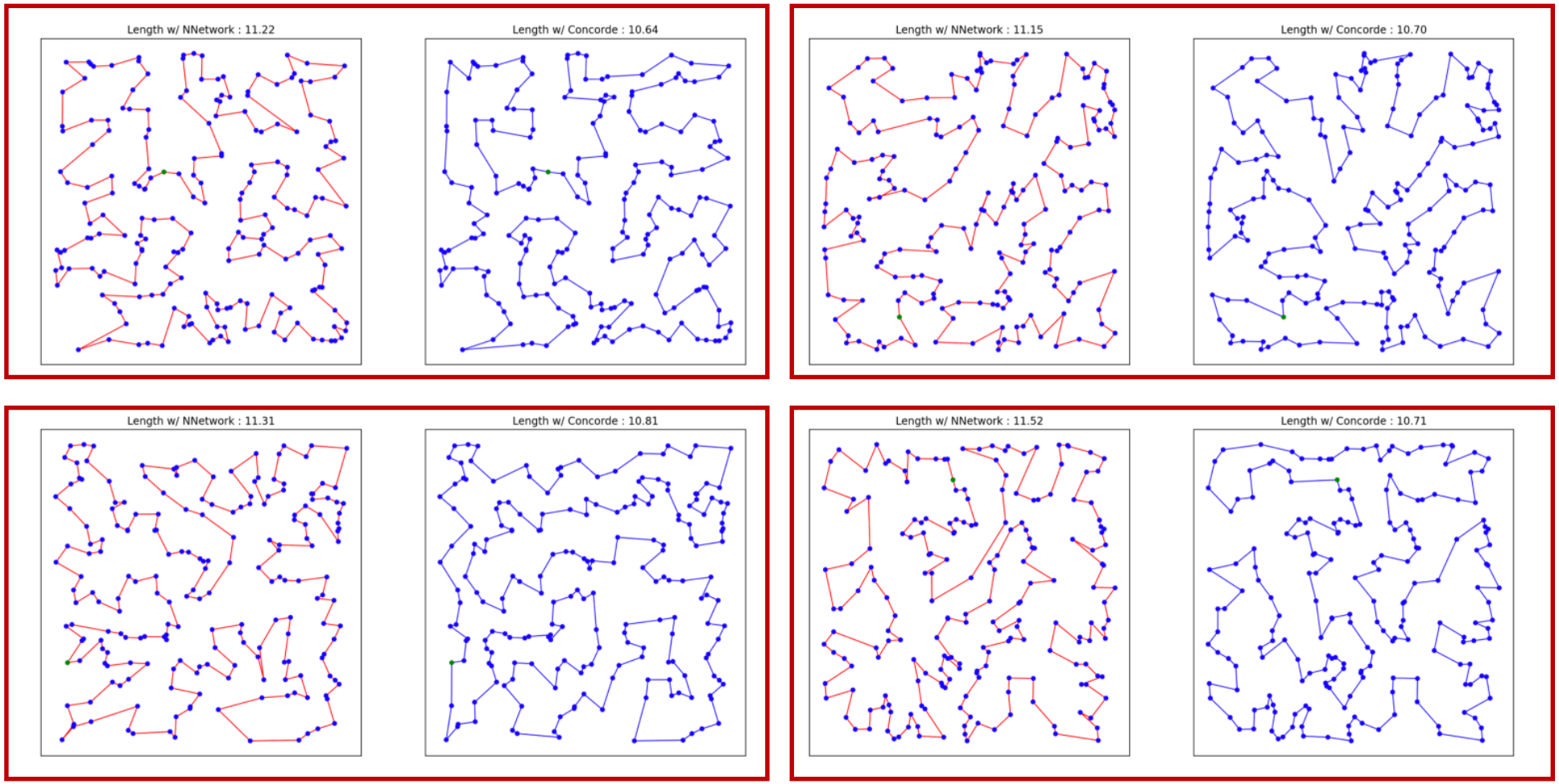
Method	TSP50 → TSP100 Obj	Gap	T Time	I Time
Concorde	7.765	0.00%	3m*	0.22s
Our model (greedy)	8.008	3.12%	4.4sec	0.13s
Our model (B=1000)	7.872	1.37%	1h	0.50s



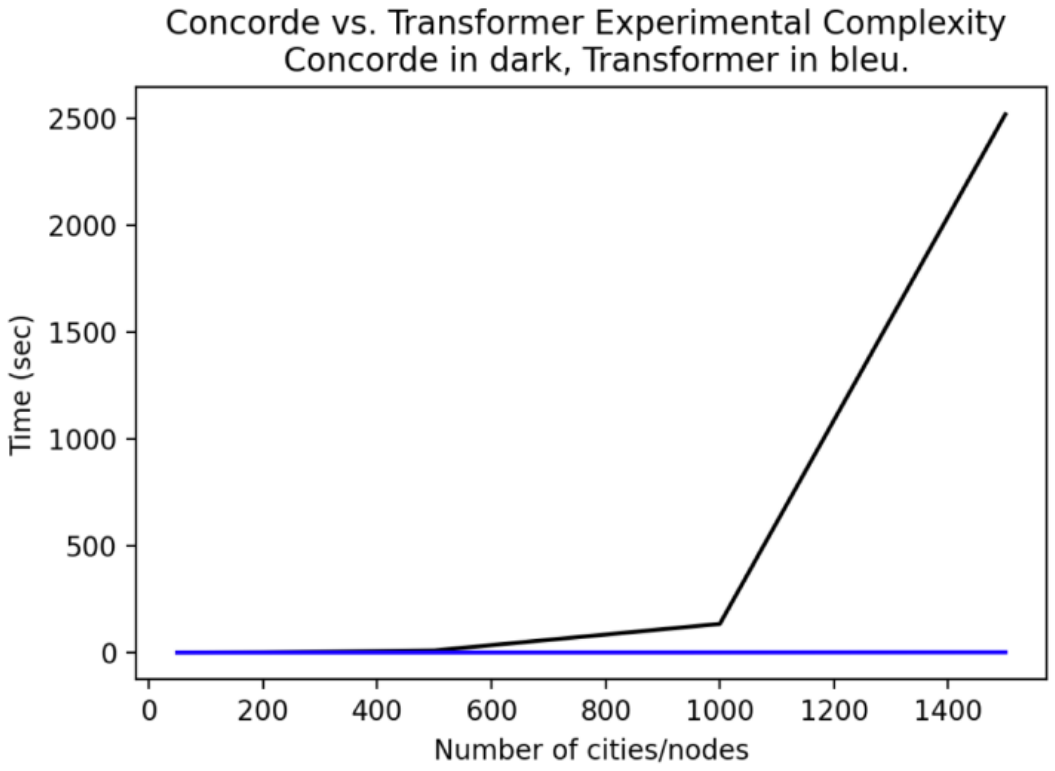
Numerical Experiments

- Bresson et al. technique vs. Concorde for TSP100⇒TSP200

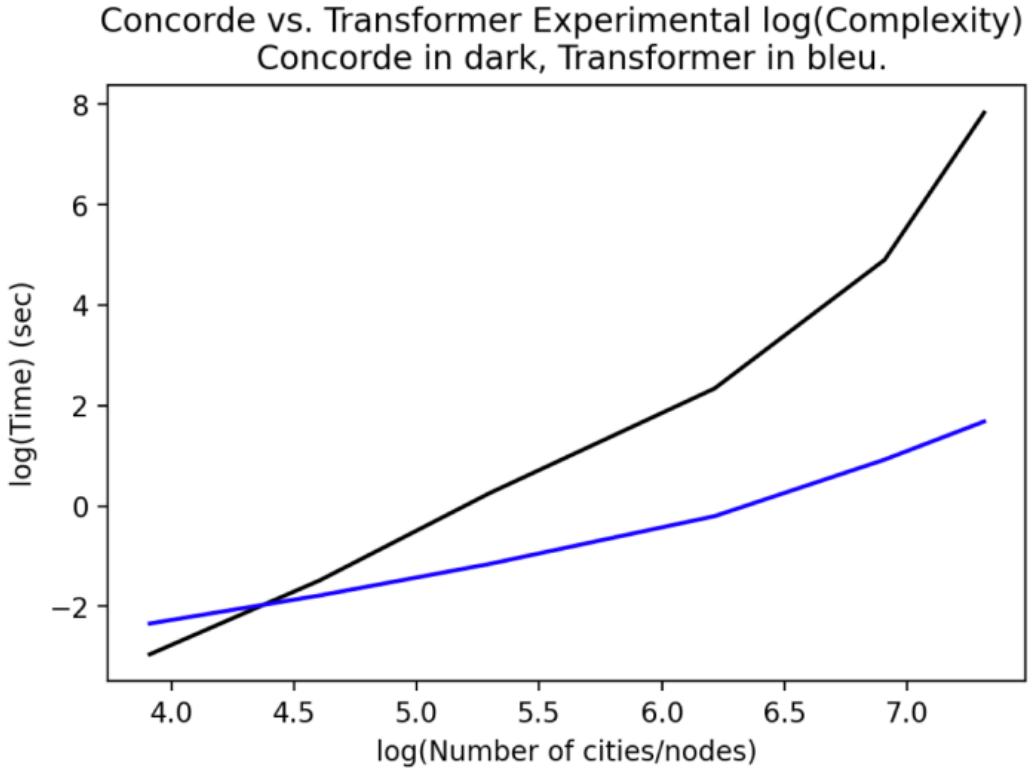
Method	TSP100 → TSP200 Obj	Gap	T Time	I Time
Concorde	10.708	0.00%	-	1.29s
Our model (greedy)	11.353	6.02%	10.7s	0.24s
Our model (B=1000)	11.181	4.41%	2.2h	0.96s



Experimental Complexity (inference time for a single TSP)



Complexity



Logarithmic complexity

Comparison of Deep Learning Solvers

	Method	TSP50 Obj	TSP50 Gap	TSP50 T Time	TSP50 I Time	TSP100 Obj	TSP100 Gap	TSP100 T Time	TSP100 I Time
MIP	Concorde'06	**5.689**	0.00%	2m*	0.05s	**7.765**	0.00%	3m*	0.22s
	Gurobi'08	-	0.00%*	2m*	-	7.765*	0.00%*	17m*	-
Heuristic	Nearest insertion	7.00*	22.94%*	0s*	-	9.68*	24.73%*	0s*	-
	Farthest insertion	6.01*	5.53%*	2s*	-	8.35*	7.59%*	7s*	-
	OR tools'15	5.80*	1.83%*	-	-	7.99*	2.90%*	-	-
	LKH-3'17	-	0.00%*	5m*	-	**7.765***	0.00%*	21m*	-
Neural Network Greedy Sampling	Vinyals et-al'15	7.66*	34.48%*	-	-	-	-	-	-
	Bello et-al'16	5.95*	4.46%*	-	-	8.30*	6.90%*	-	-
	Dai et-al'17	5.99*	5.16%*	-	-	8.31*	7.03%*	-	-
	Deudon et-al'18	5.81*	2.07%*	-	-	8.85*	13.97%*	-	-
	Kool et-al'18	5.80*	1.76%*	2s*	-	8.12*	4.53%*	6s*	-
	Kool et-al'18 (our version)	-	-	-	-	8.092	4.21%	-	-
	Joshi et-al'19	5.87	3.10%	55s	-	8.41	8.38%	6m	-
	Bresson et-al	**5.707**	0.31%	13.7s	0.07s	**7.875**	1.42%	4.6s	0.12s
Neural Network Advanced Sampling	Kool et-al'18 (B=1280)	5.73*	0.52%*	24m*	-	7.94*	2.26%*	1h*	-
	Kool et-al'18 (B=5000)	5.72*	0.47%*	2h*	-	7.93*	2.18%*	5.5h*	-
	Joshi et-al'19 (B=1280)	5.70	0.01%	18m	-	7.87	1.39%	40m	-
	Xing et-al'20 (B=1200)	-	0.20%*	-	3.5s*	-	1.04%*	-	27.6s*
	Wu et-al'20 (B=1000)	5.74*	0.83%*	16m*	-	8.01*	3.24%*	25m*	-
	Wu et-al'20 (B=3000)	5.71*	0.34%*	45m*	-	7.91*	1.85%*	1.5h*	-
	Wu et-al'20 (B=5000)	5.70*	0.20%*	1.5h*	-	7.87*	1.42%*	2h*	-
	Bresson et-al (B=100)	5.692	0.04%	2.3m	**0.09s**	7.818	0.68%	4m	**0.16s**
	Bresson et-al (B=1000)	5.690	0.01%	17.8m	0.15s	7.800	0.46%	35m	0.27s
	Bresson et-al (B=2500)	**5.689**	4e-3%	44.8m	0.33s	**7.795**	0.39%	1.5h	0.62s

Take Home Message

- The DL solvers are all approximate/heuristic methods.
  - They are either supervised learning with approximate TSP solutions (ex. TSP50, TSP100) or improved with Reinforcement Learning (no TSP solutions required).

- The recent DL solvers can be mainly divided into the following categories:
  - The GNN-based methods
    - Graph network is suitable to handle the information of city location
  - The transformer-based methods
    - TSP can be regarded as a “translation” problem
    - The translation problem has seen significant progress with the Transformers

Reference

- Bresson, X., & Laurent, T. (2021). The transformer network for the traveling salesman problem. arXiv preprint arXiv:2103.03012.
- Kool, W., Van Hoof, H., & Welling, M. (2018). Attention, learn to solve routing problems!. arXiv preprint arXiv:1803.08475.
- Deudon, M., Cournut, P., Lacoste, A., Adulyasak, Y., & Rousseau, L. M. (2018). Learning heuristics for the tsp by policy gradient. In Integration of Constraint Programming, Artificial Intelligence, and Operations Research: 15th International Conference, CPAIOR 2018, Delft, The Netherlands, June 26–29, 2018, Proceedings 15 (pp. 170-181). Springer International Publishing.
- Vinyals, O., Fortunato, M., & Jaitly, N. (2015). Pointer networks. Advances in neural information processing systems, 28.
- Croes, G. A. (1958). A method for solving traveling-salesman problems. Operations research, 6(6), 791-812.
- Fisher, M. L. (1981). The Lagrangian relaxation method for solving integer programming problems. Management science, 27(1), 1-18.
- Kirkpatrick, S., Gelatt Jr, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. science, 220(4598), 671-680.
- Held, M., & Karp, R. M. (1962). A dynamic programming approach to sequencing problems. Journal of the Society for Industrial and Applied mathematics, 10(1), 196-210.

Reference

- Jünger, M., Reinelt, G., & Rinaldi, G. (1995). The traveling salesman problem. Handbooks in operations research and management science, 7, 225-330.
- Christofides, N. (1976). Worst-case analysis of a new heuristic for the travelling salesman problem. Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group.
- Johnson, D. S. (2005, June). Local optimization and the traveling salesman problem. In Automata, Languages and Programming: 17th International Colloquium Warwick University, England, July 16–20, 1990 Proceedings (pp. 446-461). Berlin, Heidelberg: Springer Berlin Heidelberg.
- Lin, S. (1965). Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, *44*(10), 2245-2269.
- Johnson, D. S., & McGeoch, L. A. (1997). The traveling salesman problem: A case study in local optimization. Local search in combinatorial optimization, 1(1), 215-310.
- Blazinskas, A., & Misevicius, A. (2011). Combining 2-opt, 3-opt and 4-opt with k-swap-kick perturbations for the traveling salesman problem. Kaunas University of Technology, Department of Multimedia Engineering, Studentu St, 50-401.
- Lin, S., & Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. Operations research, 21(2), 498-516.
- Helsgaun, K. (2000). An effective implementation of the Lin–Kernighan traveling salesman heuristic. European journal of operational research, 126(1), 106-130.

Reference

- Hopfield, J. J., & Tank, D. W. (1985). “Neural” computation of decisions in optimization problems. Biological cybernetics, 52(3), 141-152.
- Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.
- Bello, I., Pham, H., Le, Q. V., Norouzi, M., & Bengio, S. (2016). Neural combinatorial optimization with reinforcement learning. arXiv preprint arXiv:1611.09940.
- Nazari, M., Oroojlooy, A., Snyder, L., & Takác, M. (2018). Reinforcement learning for solving the vehicle routing problem. Advances in neural information processing systems, 31.
- Khalil, E., Dai, H., Zhang, Y., Dilkina, B., & Song, L. (2017). Learning combinatorial optimization algorithms over graphs. Advances in neural information processing systems, 30.
- Joshi, C. K., Laurent, T., & Bresson, X. (2019). An efficient graph convolutional network technique for the travelling salesman problem. arXiv preprint arXiv:1906.01227.