End-to-end Reinforcement Learning for the Large-scale Traveling Salesman Problem

Yan Jin

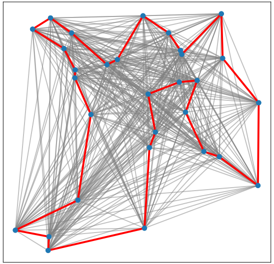
jinyan@hust.edu.cn
Huazhong University of Science and Technology

Table of Contents

Traveling Salesman Problem (TSP)

TSP

- Given a set of cities and the distances between each pair of cities
- The objective is to find a shortest path that starts from a certain city, visits each city exactly once and returns to the start city



- NP-hard: O(n!)
- Popular: One of the most studied routing problems
- Challenging: Solve the instances with tens of thousands of cities in time sensitive scenarios, e.g. on-call routing, ride hailing service

Related Work - Traditional Solvers

- **Exact solver: Concorde**
  - Integer Programming solver with cutting planes and branch-and-bound
  - Widely regard as the fastest exact TSP solver
  - Cannot handle large-scale TSP due to the memory and time limits

- **Heuristic solver: LKH3**
  - Iterative search with 2-Opt/3-Opt operators, apply a minimum spanning tree to estimate edge candidates
  - Widely regarded as the best heuristic TSP solver
  - Can handle large-scale TSP but time-consuming

Related Work - Neural Network Solvers

Search based solvers:

- L2I
  - Select an improvement operator by a reinforcement learning based controller
  - Select a perturbation operator by a rule-based controller
- Att-GCN+MCTS
  - Best neural network solver for large-scale TSP, but time-consuming (up to 10,000 cities)
  - Train a small-scale model by supervised learning
  - Merge sub-heat maps to a complete heat map
  - Monte carlo tree search with the guidance of heat map

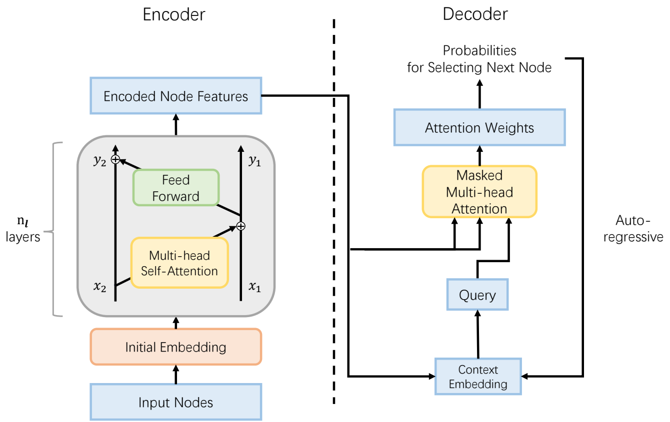
Related Work - Neural Network Solvers

End-to-end solvers:

- Point network and two variants
  - First Neural network solver, encoder with RNN, auto-regressive decoder, supervised learning
  - Use reinforcement learning approaches, reward: tour length
- Attention models
  - AM model
    - A transformer encoder without positional encoding
    - Auto-regressive decoder (graph embedding and the embeddings with first and last nodes)
    - Reinforce algorithm with a rollout baseline
  - POMO model
    - Best constructive solver for small-scale TSP ($\leq$ 100 cities)
    - Start from each node of one instance for decoder
    - A shared baseline for policy gradients
    - Multiple greedy trajectories for inference

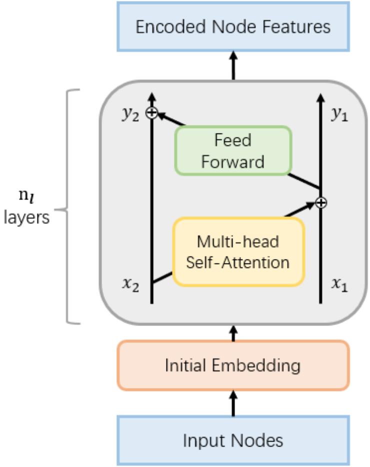
An End-to-end Model - Pointerformer

- **Motivation**: Design a deep reinforcement learning model to attain high quality solutions of TSP (more than 100 cities) in seconds
- **Architecture**: Attention model consists of encoder and decoder



Encoder

- Input data: $[x, y, \theta]$
- Feature augmentation => get 24 features for each node
- Multi-Head Attention (MHA), Feed Forward (FF), residual connection, batch normalization
- Use reversible residual network, reduce memory complexity from $\max\left(bld_{ff}, bn_h l^2\right) n_l$ to $\max\left(bld_{ff}, bn_h l^2\right)$, n_l layers, n_h-head attention, d_{ff}-dimension feed forward, b: batch size, l: number of cities
- Maintain a pair of input and output embedding features (X_1, X_2) and (Y_1, Y_2) => calculate derivations directly $X_2 = Y_2 - FF(Y_1), X_1 = Y_1 - MHA(X_2)$



Pointerformer - Decoder

- Auto-regressive decoder, one city at a time
- Enhanced context embedding as query: graph embedding ($h_g = \sum_{i=1}^{N} h_i^{enc}$), partial routing embedding ($h_\tau = \sum_{i=1}^{t-1} h_{\tau_i}^{enc}$), the last node embedding $h_{\pi_{t-1}}$ and the first node embedding h_{π_1}

$$q_t = \frac{1}{N}(h_g + h_\tau) + h_{\pi_{t-1}} + h_{\pi_1}$$

- A multi-pointer network: extend the single-pointer network to the multi-pointer network, but different from the existing multi-pointer network in the literature

$$PN = \frac{1}{H}\sum_{h=0}^{H} \frac{(\mathbf{q}_t W_h^q)^T (\mathbf{k}_j W_h^k)}{\sqrt{d_\mathrm{k}}}, score_{ij} = PN - cost(i,j)$$

- Query interacts with all unvisited nodes, visited nodes are masked

$$u_{ij} = \begin{cases} C \cdot \tanh(score_{ij}) & \text{node } j \text{ is to be visited} \\ -\infty & \text{Otherwise} \end{cases}$$

- Compute output probability vector p with a softmax

Pointerformer - Improvement on REINFORCE

- Apply and improve the REINFORCE algorithm for training
- Reduce the variance by subtracting the mean
- Divide by the variance so that each sample has the same variance to improve training speed

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} \left(\frac{R\left(\tau^i\right) - \mu(s)}{\sigma(s)} \right) \nabla_\theta \log p_\theta \left(\tau^i \mid s\right)$$

where $\mu(s) = \frac{1}{N} \sum_{j=1}^{N} R\left(\boldsymbol{\tau}^j\right)$; $\sigma(s) = \frac{1}{N} \sum_{j=1}^{N} \left(R\left(\boldsymbol{\tau}^j\right) - \mu(s)\right)^2$

Pointerformer - Experiments

Method	TSP_random20			TSP_random50			TSP_random100			TSP_random200			TSP_random500		
	Len	Gap(%)	Time	Len	Gap(%)	Time	Len	Gap(%)	Time	Len	Gap(%)	Time	Len	Gap(%)	Time
OPT	3.83			5.69			7.76			10.72			16.55		
AM	3.83	0.06	5.22s	5.72	0.49	12.76m	7.94	23.20	32.72m	-	-	-	-	-	-
POMO	3.83	0.00	36.86s	5.69	0.02	1.15m	7.77	0.16	2.17m	-	-	-	-	-	-
AM+LCP	3.84	0.00	30.00m	5.70	0.02	6.89h	7.81	0.54	11.94h	-	-	-	-	-	-
DRL+2opt	3.83	0.00	3.33h	5.70	0.12	4.62m	7.82	0.78	6.57h	-	-	-	-	-	-
Att-GCN+MCTS	3.83	0.00	1.6m	5.69	0.01	7.90m	7.76	0.04	15m	10.81	0.88	2.5m	16.97	2.54	5.9m
Pointerformer	**3.83**	**0.00**	5.82s	**5.69**	0.02	11.63s	7.77	0.16	52.34s	**10.79**	**0.68**	**5.54s**	17.14	3.56	59.35s

E: End-to-end DRL; S: Search-based DRL.

Method	TSPLIB1∼100			TSPLIB101∼500			TSP501∼1002		
	Len	Gap(%)	Time	Len	Gap(%)	Time	Len	Gap(%)	Time
OPT	19454.17			40842.43			62427.71		
AM	22283.67	15.36	0.23s	72137.93	78.18	0.86s	140664.29	139.02	5.79s
POMO	**19628.67**	1.20	1.41s	43652.77	6.99	1.55s	82162.29	26.93	3.49s
DRL+2opt	19916.50	2.43	15.20m	46651.40	13.85	27.92m	82797.71	42.57	1.24h
Pointerformer(Model100)	19728.50	1.33	0.20s	**42963.20**	5.43	0.46s	75081.43	18.65	5.14s
Pointerformer(Model200)	20135.00	2.91	0.20s	43810.67	8.37	0.46s	**73915.57**	18.20	5.14s

- Can scale to TSP instances with up to 500 nodes
- Comparable results as search-based DRL, but in shorter time
- Well generalize to practical instances with varied distributions without re-training

A Hierarchical Reinforcement Learning Model (H-TSP)

- **Motivation**: Design a deep reinforcement learning model to solve larger TSP (up to tens of thousands of cities) in several minutes
- **Architecture**: Following the divide-and-conquer approach, upper-level and lower-level models are responsible for generating sub-problems and solving sub-problems

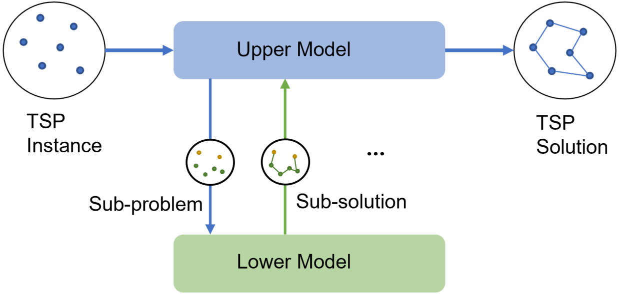
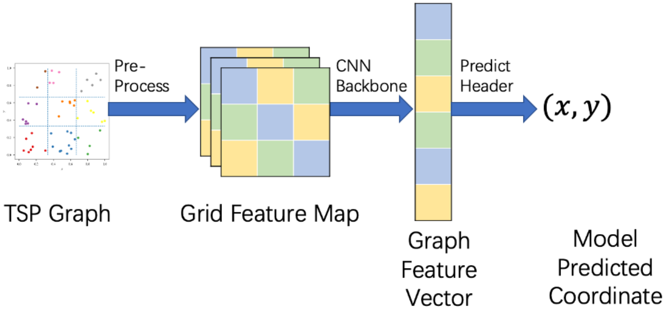


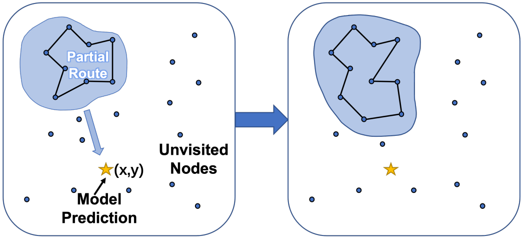
Figure: The hierarchical architecture

Upper-level Model: A Gird-based Encoder



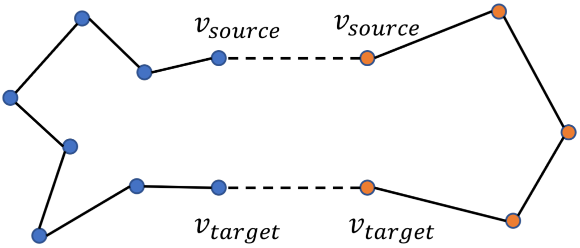
- Input data (B, N, D) : B instances with N nodes and D features
- Discretize evenly the 2D space of each instance into H × W grids
- Form a pseudo-image by high-dimension project, max pooling, zero padding
- Apply a convolutional neural network on the pseudo-image to generate embeddings
- Use an actor-critic architecture to predict a coordinate

Upper-level Model - Sub-problem Generation



- Select unvisited nodes in a local neighborhood:
  1. Set start point as the node closest to predicted coordinate
  2. Breadth-first search on the simplified *k*-NN graph
  3. Select a set of unvisited nodes according to BFS
- Select visited nodes to refine current partial route:
  1. Set start point as the node closest to predicted coordinate
  2. Expand to both directions on the route with equal nodes
  3. Generate a set of visited nodes, and set the two endpoints

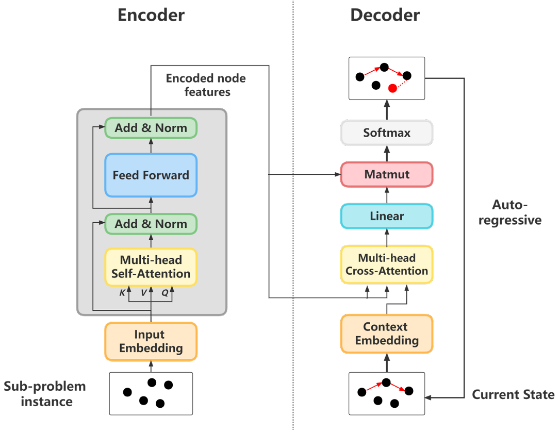
H-TSP - Sub-problem Generation and Merging



- **Sub-problem generation (open-loop TSP with fixed endpoints):** Start from source node, visit all other nodes exactly once and end in target node => *solved by lower-level model*

- **Sub-solution merging:** Two open-loop TSPs can be easily merged to a close-loop path

H-TSP: Lower-level Model



- Solve small-scale open-loop TSPs with prescribed starting and ending cities
- Transformer based Encoder, auto-regressive Decoder
- Construct the symmetry property of open-loop TSPs and take the advantage of average tour length of each graph as baseline

H-TSP - Experiments

Algorithm	Random1000			Random2000		
	Length	Gap (%)	Time (s)	Length	Gap (%)	Time (s)
Concorde	23.12	0.00	487.89	32.48	0.00	7949.97
LKH-3	23.16	0.17	22.01	32.64	0.49	79.75
OR-Tools	24.23	4.82	104.34	34.04	4.82	532.14
POMO	30.52	32.01	4.28	46.49	43.15	35.89
DRL-2opt	37.90	63.93	55.56	115.59	255.92	827.43
Att-GCN +MCTS	23.86	3.22	5.85	33.42	2.91	200.28
H-TSP	24.65	6.62	**0.33**	34.88	7.39	**0.72**

Algorithm	Random5000			Random10000		
	Length	Gap (%)	Time (s)	Length	Gap (%)	Time (s)
LKH-3	51.36	0.00	561.74	72.45	0.00	4746.59
OR-Tools	53.35	3.86	5368.24	74.95	3.44	21358.66
POMO	80.79	57.29	575.63	OOM	OOM	OOM
DRL-2opt	754.91	1369.76	2308.48	2860.86	3848.66	6073.43
Att-GCN +MCTS	52.83	2.86	377.47	74.93	3.42	395.85
H-TSP	55.01	7.10	**1.66**	77.75	7.32	**3.32**

- Solution quality: achieve comparable results to the SOTA methods
- Efficiency: outperform all baselines and reduce the time consumption up to two orders of magnitude
- Have potential in real- world scenarios that require solving large-scale TSP in a short time even real-time

Conclusion and Future work

Conclusion

- Propose effective models based on deep reinforcement learning
- Take advantage of inference efficiency of end-to-end models
- Will be useful for time-sensitive practical applications
- Have potentials to be extended to other large-scale optimization problems

Future directions

- Find an effective mechanism to replace self-attention
- Handle the large-scale TSP challenges such as the World TSP with 1,904,711-cities
- Can tackle various TSP-type problems, VRP-type problems and other optimization problems

Thank you !
Q & A

References:

- Yan Jin, Yuandong Ding, Xuanhao Pan, Kun He, Li Zhao, Tao Qin, Lei Song, Jiang Bian. Pointerformer: Deep Reinforced Multi-Pointer Transformer for the Traveling Salesman Problem. Accepted by AAAI 2023.
- Xuanhao Pan, Yan Jin*, Yuandong Ding, Mingxiao Feng, Li Zhao, Lei Song, Jiang Bian. H-TSP: Hierarchically Solving the Large-Scale Travelling Salesman Problem. Accepted by AAAI 2023.