Learning 2-opt Local Search for the Traveling Salesman Problem

Paulo da Costa, Jason Rhuggenaath, Yingqian Zhang and Alp Akcay

Eindhoven University of Technology - School of Industrial Engineering

Traveling Salesman Problem

From a set of n locations:

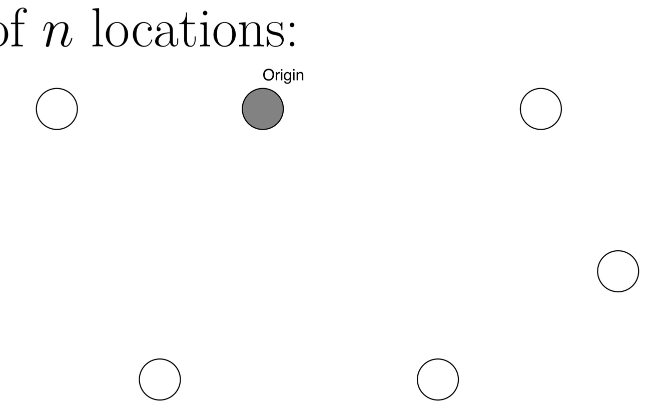

Figure 1:Locations

Find the best tour that visits all locations (and returns to the origin)

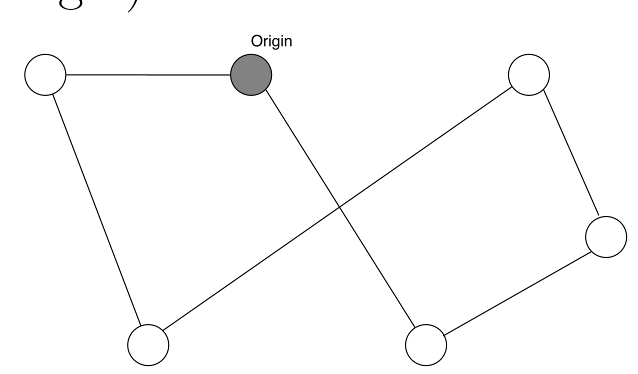

Figure 2:TSP tour

k-opt Heuristics

k edge swaps (k-opt moves) resulting in a shorter tour. Complexity: $O(n^k)$. Simpler: Consider 2-opt and 3-opt moves.

Edge Selection: Usually *Greedy*. e.g. First Improvement (FI), Best Improvement (BI).

Learn better 2-opt strategies?

Learn 2-opt strategies without supervision

Can be expanded to accommodate k-opt.

Trained on smaller instances and transferred to larger instances.

Markov Decision Process

States: A tuple $\bar{S} = (S, S')$, where S and S' are the current and lowest-cost solution.

Actions: (a_1, a_2) composed of two node indices.

Transitions: 2-opt operation to solution S.

Rewards: Improvement upon the current best-found solution, i.e. $R_t = L(S'_t) - L(S'_{t+1})$.

Policy and Value Approximation

Encoder: **Graph topology**, **node ordering** and **symmetry**.

Policy decoder: probability of a k-opt move as

$$\pi_\theta(A|\bar{S}) = \prod_{i=1}^{k} p_\theta\left(a_i | a_{<i}, \bar{S}\right). \tag{1}$$

Policy Gradient updates of the form, b-th instance:

$$\nabla_\theta J(\theta) \approx \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(A_t^b | \bar{S}_t^b)(G_t^b - V_\phi(\bar{S}_t^b)) \tag{2}$$

Value decoder: trained on Monte Carlo returns as

$$\mathcal{L}(\phi) \approx \sum_{t=0}^{T-1} \left\| G_t^b - V_\phi(\bar{S}_t^b) \right\|_2^2 \tag{3}$$

Results

TSP instances of 3 sizes, 20, 50 and 100 nodes. Nodes drawn from a uniform in $[0,1]^2$. Performance: **Optimality Gap (%)**:

	Method (nr. of samples)	Type	TSP20			TSP50			TSP100		
			Cost	Gap	Time	Cost	Gap	Time	Cost	Gap	Time
	Concorde [1]	Solver	3.84	0.00%	(1m)	5.70	0.00%	(2m)	7.76	0.00%	(3m)
H	OR-Tools [2]	S	3.85	0.37%		5.80	1.83%		7.99	2.90%	
C+S	GCN {1280} [3]	SL	3.84	0.01%	(12m)	**5.70**	**0.01**%	(18m)	7.87	1.39%	(40m)
C+S	GAT {1280} [4]	RL	3.84	0.08%	(5m)	5.73	0.52%	(24m)	7.94	2.26%	(1h)
I+S	GAT-T {1000} [5]	RL	3.84	0.03%	(12m)	5.75	0.83%	(16m)	8.01	3.24%	(25m)
I+S	GAT-T {3000} [5]	RL	3.84	0.00%	(39m)	5.72	0.34%	(45m)	7.91	1.85%	(1h)
I+S	GAT-T {5000} [5]	RL	3.84	0.00%	(1h)	5.71	0.20%	(1h)	7.87	1.42%	(2h)
I+S	Ours {500}	RL	3.84	0.01%	(5m)	5.72	0.36%	(7m)	7.91	1.84%	(10m)
I+S	Ours {1000}	RL	**3.84**	**0.00**%	(10m)	5.71	0.21%	(13m)	7.86	1.26%	(21m)
I+S	Ours {2000}	RL	**3.84**	**0.00**%	(15m)	5.70	0.12%	(29m)	**7.83**	**0.87**%	(41m)

Table 1:Performance of TSP methods w.r.t. Concorde (optimal). SL: Supervised Learning, RL: Reinforcement Learning, S: Heuristic Solver

- Learned policies **outperform greedy policies** with and without restarts.
- Our method: **0.87% optimality gap** for TSP100 when rolling out for 2,000 steps.
- It is **more sample efficient** (during training) than GAT-T [5] and achieves lower gaps requiring fewer samples.
- We outperform GAT [4] when sampling only 500 solutions vs 1,280 beam width.
- Our method only falls short of the SL method GCN [3] in the TSP50 case.
- **Policies learned on 50 nodes** result in 1.37% gap on 100 nodes instances **outperforming previous methods**.

Results (cont.)

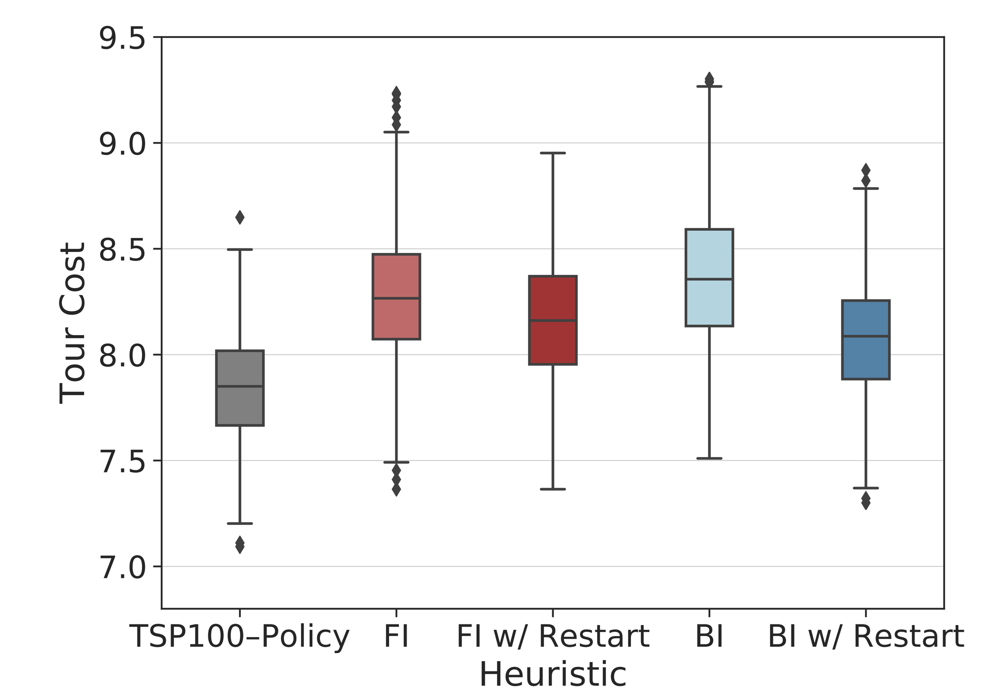

Figure 3:Tour cost comparison of learned and greedy policies.

Conclusion

Learning 2-opt policies can yield better policies than greedy ones.

Adapting such policies to more general TSP may be necessary for instances coming from different distributions.

Learning methods are slowly approaching the quality of handcrafted heuristics, but still cannot compete in scalability and depend on high training times.

References

[1] David Applegate, Ribert Bixby, Vasek Chvatal, and William Cook. Concorde tsp solver. *URL http://www.math.uwaterloo.ca/tsp/concorde*, 2006.

[2] Laurent Perron and Vincent Furnon. Or-tools.

[3] Chaitanya K Joshi, Thomas Laurent, and Xavier Bresson. An efficient graph convolutional network technique for the travelling salesman problem. *arXiv:1906.01227*, 2019.

[4] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

[5] Yaoxin Wu, Wen Song, Zhiguang Cao, Jie Zhang, and Andrew Lim. Learning improvement heuristics for solving the travelling salesman problem. *arXiv:1912.05784*, 2019.

EINDHOVEN UNIVERSITY OF TECHNOLOGY