UNIVERSITY OF AMSTERDAM

ORTEC OPTIMIZE YOUR WORLD

AMLAB Amsterdam Machine Learning Lab

Get the paper!

Attention, Learn to Solve Routing Problems!

Wouter Kool, Herke van Hoof & Max Welling

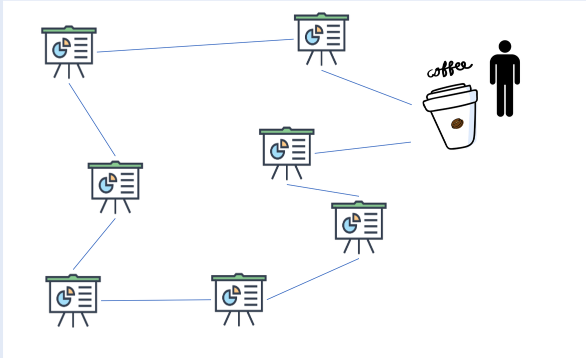
Fork me on GitHub! Get the code!

Travelling S(alesman|cientist) Problem (TSP)

Goal? Learn heuristic algorithms automatically!

Why? Problem is (NP-)hard, development costly!

How? 'Translate' problem into solution…



Travelling Scientist Problem

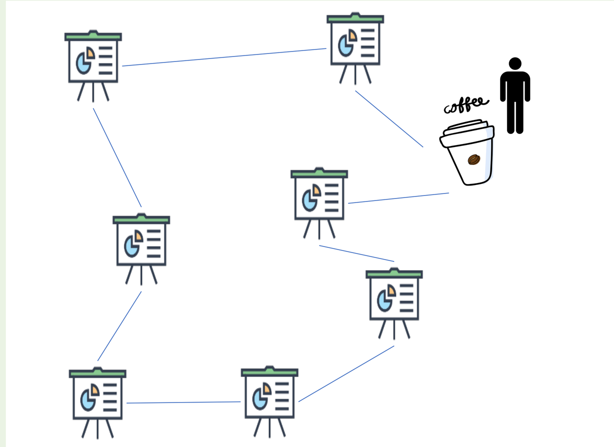
seq2seq

Pointer Networks (PN) (Vinyals et al., 2015)

Math?

- Instance $s = ((x_1, y_1), (x_2, y_2), \dots, (x_n, y_n))$
- Solution $\boldsymbol{\pi} = (\pi_1, \pi_2, \dots \pi_n)$ e.g. (3,1,2,4)
- Model $p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s) = \prod_{t=1}^{n} p_{\boldsymbol{\theta}}(\pi_t | s, \pi_{1:t-1})$

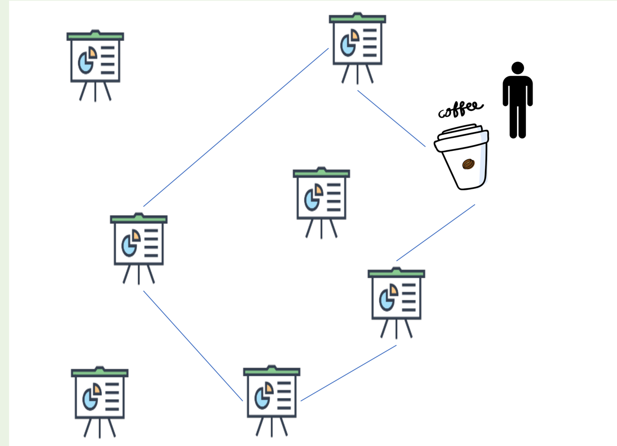
Travelling Salesman Problem (TSP)



Minimize length

Visit all nodes

Orienteering Problem (OP)



Maximize total prize

Max length constraint

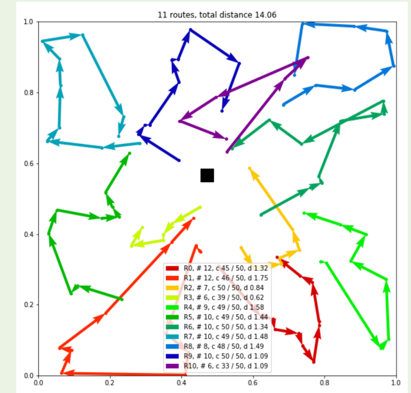
(Stochastic) Prize Collecting TSP ((S)PCTSP)

Minimize length + penalties of unvisited nodes

Collect min. total prize

Vehicle Routing Problem (VRP)

See also Nazari et al. (2018)



Minimize length

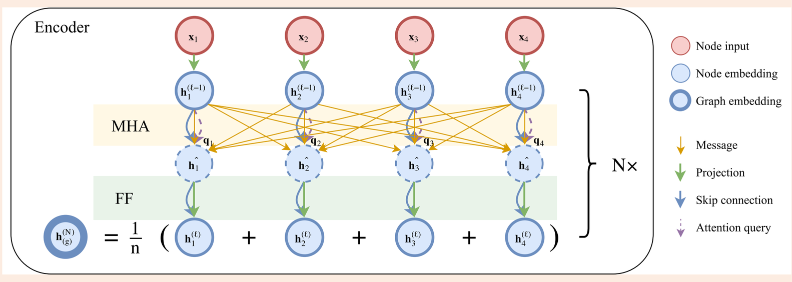
Visit all nodes

Total route demand $\leq$ vehicle capacity

Train for each problem, *same hyperparameters*!

Attention Model (AM)

Encoder



- Compute embeddings of all nodes
- Attention based message passing

Attention Is All You Need (Vaswani et al., 2017)

$$\hat{\boldsymbol{h}}_i = \mathrm{BN}^{\ell}\left(\boldsymbol{h}_i^{(\ell-1)} + \mathrm{MHA}_i^{\ell}\left(\boldsymbol{h}_1^{(\ell-1)}, \dots, \boldsymbol{h}_n^{(\ell-1)}\right)\right)$$

$$\boldsymbol{h}_i^{(\ell)} = \mathrm{BN}^{\ell}\left(\hat{\boldsymbol{h}}_i + \mathrm{FF}_i^{\ell}(\hat{\boldsymbol{h}}_i)\right)$$

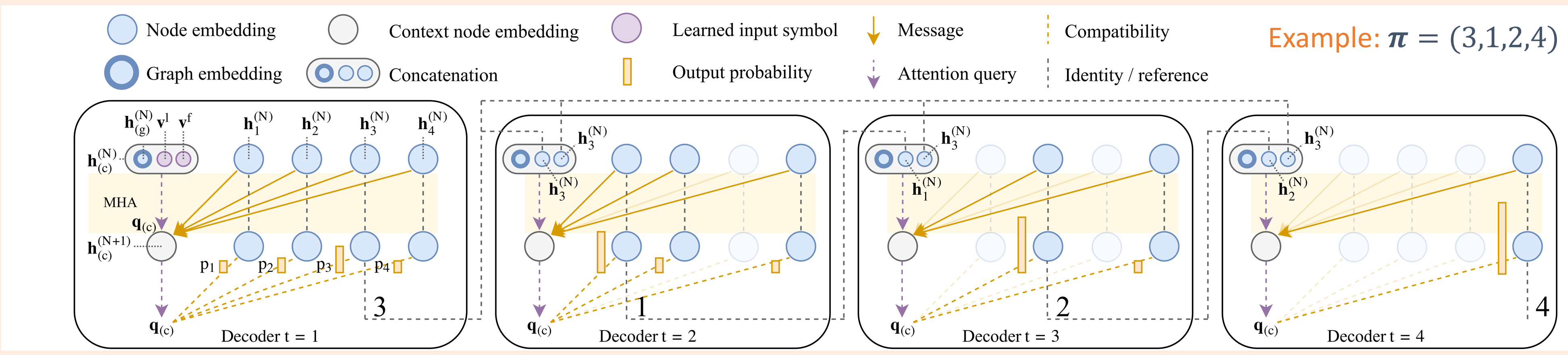
$$\boldsymbol{q}_i = W^Q \boldsymbol{h}_i \qquad \boldsymbol{k}_i = W^K \boldsymbol{h}_i \qquad \boldsymbol{v}_i = W^V \boldsymbol{h}_i$$

$$u_{ij} = \frac{\boldsymbol{q}_i^T \boldsymbol{k}_j}{\sqrt{d_k}} \qquad a_{ij} = \frac{e^{u_{ij}}}{\sum_{j'} e^{u_{ij'}}} \qquad \boldsymbol{h}'_i = \sum_j a_{ij} \boldsymbol{v}_j$$

$$\mathrm{MHA}_i(\boldsymbol{h}_1, \dots, \boldsymbol{h}_n) = \sum_m W_m^O \boldsymbol{h}'_{im}$$

Decoder

Example: $\boldsymbol{\pi} = (3,1,2,4)$



- Output one node at a time (probabilistic, softmax logits = attention)
- Based on context:
  - Graph (what is the problem?)
  - First node (where to go?)
  - Last node (where am I?)
  - Mask (what is already visited?)

$$p_i = p_{\boldsymbol{\theta}}(\pi_t = i | s, \pi_{1:t-1}) = \frac{e^{u_{(c)j}}}{\sum_j e^{u_{(c)j}}}$$

How to train?

Let's REINFORCE… said Bello et al. (2016)

$$\mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s)}\left[\left(L(\boldsymbol{\pi}) - b(s)\right)\nabla \log p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s)\right]$$

tour length baseline

Good baseline should estimate difficulty of s

- ~~Exp. moving avg.~~ (boring!)
- ~~Critic~~ (try it!)
- Rollout (but greedy!)

Similar to Rennie et al. (2017)

Algorithm 1 REINFORCE with Rollout Baseline

Input: number of epochs E, steps per epoch T, batch size B, significance α

Init $\boldsymbol{\theta}$, $\boldsymbol{\theta}^{BL} \leftarrow \boldsymbol{\theta}$

for $epoch = 1, \dots, E$ **do**

  for $step = 1, \dots, T$ **do**

    $s_i \leftarrow$ RandomInstance() $\forall i \in \{1, \dots, B\}$

    $\boldsymbol{\pi}_i \leftarrow$ SampleRollout$(s_i, p_{\boldsymbol{\theta}})$ $\forall i \in \{1, \dots, B\}$

    $\boldsymbol{\pi}_i^{BL} \leftarrow$ GreedyRollout$(s_i, p_{\boldsymbol{\theta}^{BL}})$ $\forall i \in \{1, \dots, B\}$

    $\nabla \mathcal{L} \leftarrow \sum_{i=1}^{B} \left(L(\boldsymbol{\pi}_i) - L(\boldsymbol{\pi}_i^{BL})\right) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\pi}_i)$

    $\boldsymbol{\theta} \leftarrow$ Adam$(\boldsymbol{\theta}, \nabla \mathcal{L})$

  end for

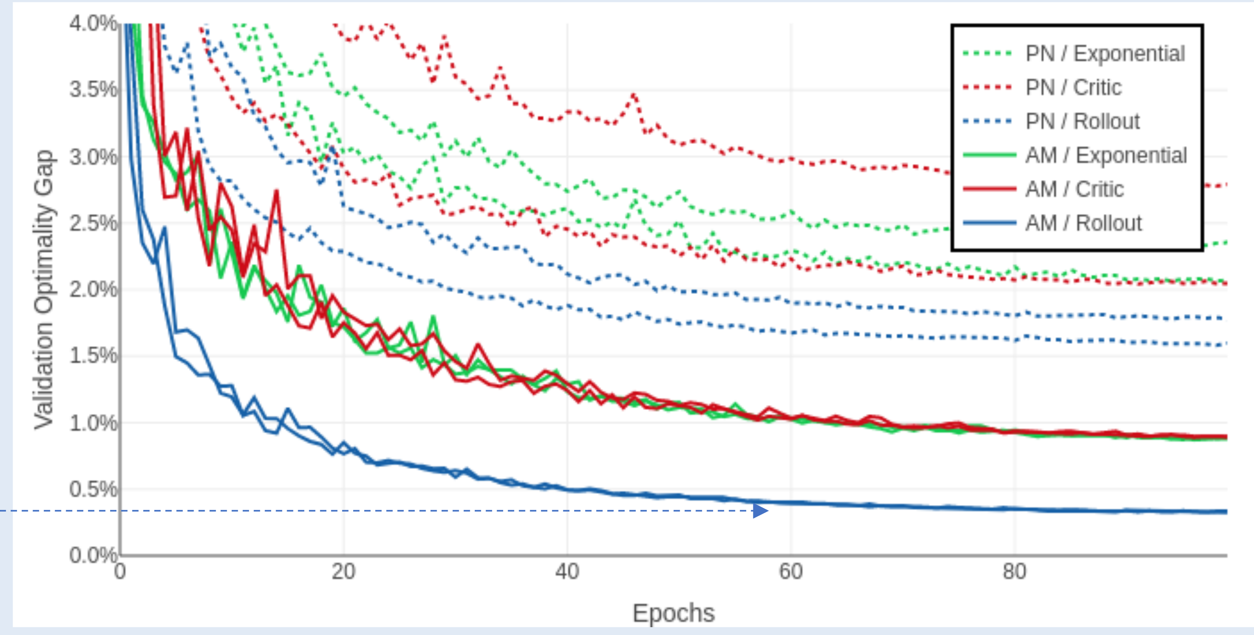
  if OneSidedPairedTTest$(p_{\boldsymbol{\theta}}, p_{\boldsymbol{\theta}^{BL}}) < \alpha$ **then**

    $\boldsymbol{\theta}^{BL} \leftarrow \boldsymbol{\theta}$

  end if

end for

AM vs. PN & baselines (TSP20)



Results

Problem	Method	n = 20 Obj.	Gap	Time	n = 50 Obj.	Gap	Time	n = 100 Obj.	Gap	Time
TSP	Concorde	3.84	0.00%	(1m)	5.70	0.00%	(2m)	7.76	0.00%	(3m)
	LKH3	3.84	0.00%	(18s)	5.70	0.00%	(5m)	7.76	0.00%	(21m)
	Gurobi	3.84	0.00%	(7s)	5.70	0.00%	(2m)	7.76	0.00%	(17m)
	Gurobi (1s)	3.84	0.00%	(8s)	5.70	0.00%	(2m)		-	
	Nearest Insertion	4.33	12.91%	(1s)	6.78	19.03%	(2s)	9.46	21.82%	(6s)
	Random Insertion	4.00	4.36%	(0s)	6.13	7.65%	(1s)	8.52	9.69%	(3s)
	Farthest Insertion	3.93	2.36%	(1s)	6.01	5.53%	(2s)	8.35	7.59%	(7s)
	Nearest Neighbor	4.50	17.23%	(0s)	7.00	22.94%	(0s)	9.68	24.73%	(0s)
	Vinyals et al. (gr.)	3.88	1.15%		7.66	34.48%			-	
	Bello et al. (gr.)	3.89	1.42%		5.95	4.46%		8.30	6.90%	
	Dai et al.	3.89	1.42%		5.99	5.16%		8.31	7.03%	
	Nowak et al.	3.93	2.46%			-			-	
	EAN (greedy)	3.86	0.66%	(2m)	5.92	3.98%	(5m)	8.42	8.41%	(8m)
	AM (greedy)	**3.85**	**0.34%**	(0s)	**5.80**	**1.76%**	(2s)	**8.12**	**4.53%**	(6s)
	OR Tools	3.85	0.37%		5.80	1.83%		7.99	2.90%	
	Chr.f. + 2OPT	3.85	0.37%		5.79	1.65%			-	
	Bello et al. (s.)		-		5.75	0.95%		8.00	3.03%	
	EAN (gr. + 2OPT)	3.85	0.42%	(4m)	5.85	2.77%	(26m)	8.17	5.21%	(3h)
	EAN (sampling)	3.84	0.11%	(5m)	5.77	1.28%	(17m)	8.75	12.70%	(56m)
	EAN (s. + 2OPT)	3.84	0.09%	(6m)	5.75	1.00%	(32m)	8.12	4.64%	(5h)
	AM (sampling)	**3.84**	**0.08%**	(5m)	**5.73**	**0.52%**	(24m)	**7.94**	**2.26%**	(1h)
CVRP	Gurobi	6.10	0.00%			-			-	
	LKH3	6.14	0.58%	(2h)	10.38	0.00%	(7h)	15.65	0.00%	(13h)
	RL (greedy)	6.59	8.03%		11.39	9.78%		17.23	10.12%	
	AM (greedy)	**6.40**	**4.97%**	(1s)	**10.98**	**5.86%**	(3s)	**16.80**	**7.34%**	(8s)
	RL (beam 10)	6.40	4.92%		11.15	7.46%		16.96	8.39%	
	Random CW	6.81	11.64%		12.25	18.07%		18.96	21.18%	
	Random Sweep	7.08	16.07%		12.96	24.91%		20.33	29.93%	
	OR Tools	6.43	5.41%		11.31	9.01%		17.16	9.67%	
	AM (sampling)	**6.25**	**2.49%**	(6m)	**10.62**	**2.40%**	(28m)	**16.23**	**3.72%**	(2h)
SDVRP	RL (greedy)	6.51	4.19%		11.32	6.88%		17.12	5.23%	
	AM (greedy)	**6.39**	**2.34%**	(1s)	**10.92**	**3.08%**	(4s)	**16.83**	**3.42%**	(11s)
	RL (beam 10)	6.34	1.47%		11.08	4.61%		16.86	3.63%	
	AM (sampling)	**6.25**	**0.00%**	(9m)	**10.59**	**0.00%**	(42m)	**16.27**	**0.00%**	(3h)
OP (distance)	Gurobi	5.39	0.00%	(16m)		-			-	
	Gurobi (1s)	4.62	14.22%	(4m)	1.29	92.03%	(6m)	0.58	98.25%	(7m)
	Gurobi (10s)	5.37	0.33%	(12m)	10.96	32.20%	(51m)	1.34	95.97%	(53m)
	Gurobi (30s)	5.38	0.05%	(14m)	13.57	16.09%	(2h)	3.23	90.28%	(3h)
	Compass	5.37	0.36%	(2m)	16.17	0.00%	(5m)	33.19	0.00%	(15m)
	Tsili (greedy)	4.08	24.25%	(4s)	12.46	22.94%	(4s)	25.69	22.59%	(5s)
	AM (greedy)	**5.19**	**3.64%**	(0s)	**15.64**	**3.23%**	(1s)	**31.62**	**4.75%**	(5s)
	GA (Python)	5.12	4.88%	(10m)	10.90	32.59%	(1h)	14.91	55.08%	(5h)
	OR Tools (10s)	4.09	24.05%	(52m)		-			-	
	Tsili (sampling)	5.30	1.62%	(28s)	15.50	4.14%	(2m)	30.52	8.05%	(6m)
	AM (sampling)	**5.30**	**1.56%**	(4m)	**16.07**	**0.60%**	(16m)	**32.68**	**1.55%**	(53m)
PCTSP	Gurobi	3.13	0.00%	(2m)		-			-	
	Gurobi (1s)	3.14	0.07%	(1m)		-			-	
	Gurobi (10s)	3.13	0.00%	(2m)	4.54	1.36%	(32m)		-	
	Gurobi (30s)	3.13	0.00%	(2m)	4.48	0.03%	(54m)		-	
	AM (greedy)	**3.18**	**1.62%**	(0s)	**4.60**	**2.66%**	(2s)	**6.25**	**4.46%**	(5s)
	ILS (C++)	3.16	0.77%	(16m)	4.50	0.36%	(2h)	**5.98**	**0.00%**	(12h)
	OR Tools (10s)	3.14	0.05%	(52m)	4.51	0.70%	(52m)	6.35	6.21%	(52m)
	OR Tools (60s)	**3.13**	**0.01%**	(5h)	**4.48**	**0.00%**	(5h)	6.07	1.56%	(5h)
	ILS (Python 10x)	5.21	66.19%	(4m)	12.51	179.05%	(3m)	23.98	300.95%	(3m)
	AM (sampling)	3.15	0.45%	(5m)	4.52	0.74%	(19m)	6.08	1.67%	(1h)
SPCTSP	REOPT (all)	3.34	2.38%	(17m)	4.68	1.04%	(2h)	6.22	1.10%	(12h)
	REOPT (half)	3.31	1.38%	(25m)	**4.64**	**0.00%**	(3h)	**6.16**	**0.00%**	(16h)
	REOPT (first)	3.31	1.60%	(1h)	4.66	0.44%	(22h)		-	
	AM (greedy)	**3.26**	**0.00%**	(0s)	4.65	0.33%	(2s)	6.32	2.69%	(5s)

References

[1] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio. Neural combinatorial optimization with reinforcement learning. arXiv preprint arXiv:1611.09940, 2016.
[2] M. Nazari, A. Oroojlooy, L. V. Snyder, and M. Takác. Reinforcement learning for solving the vehicle routing problem. arXiv preprint arXiv:1802.04240, 2018.
[3] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel. Self-critical sequence training for image captioning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 7008–7024, 2017.
[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems, pages 5998–6008, 2017.
[5] O. Vinyals, M. Fortunato, and N. Jaitly. Pointer networks. In Advances in Neural Information Processing Systems, pages 2692–2700, 2015.