Learning Improvement Heuristics for Solving Routing Problems

Yaoxin Wu, Wen Song, Zhiguang Cao, Jie Zhang, Andrew Lim

2022.11.28
발표자 : 이성진

Contents

Introduction
Routing Problems
Routing Problem Solvers

The Method
RL Formulation
Policy Network
Training Algorithm
Deployment

Related Work

Experimental Results
Comparison with State-of-the-art Methods
Comparison with Conventional Policies
Enhancement by Diversifying
Generalization Analysis
Visualization
Test on Real-World Dataset

Preliminaries
TSP and CVRP
Improvement Heuristics

Conclusions and Future Work

Introduction

- ✓ Routing Problems
- ✓ Routing Problem Solvers

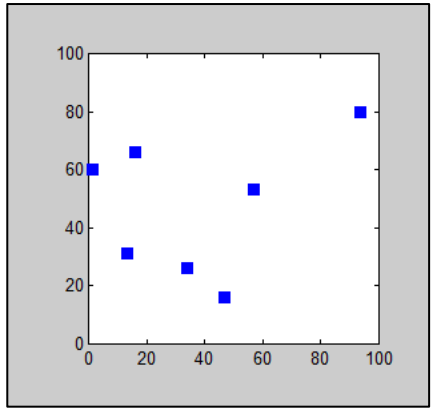
Routing Problems

Combinatorial Optimization Problems

- Subfield of mathematical optimization which consists of finding an optimal object from a finite set of objects.
- Achieving satisfactory results is still challenging, due to their **NP-hardness**.
  - ✓ There are no polynomial-time algorithms for NP-hard problems.

Travelling Salesman Problem

- Finding the shortest possible route that visits each city exactly once and returns to the origin city.
- Applications in planning, logistics, and the manufacture of microchips.

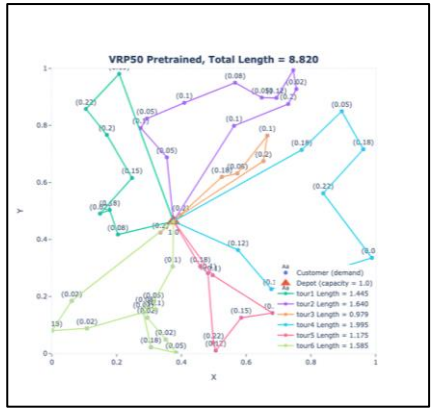


[TSP Evaluation]

[SK T-map TMS API]

Capacitated Vehicle Routing Problem

- Capacitated vehicle starts from depot.
- Finding the shortest possible route that visits each city which has customer demand.
- Vehicle can visit depot multiple times.



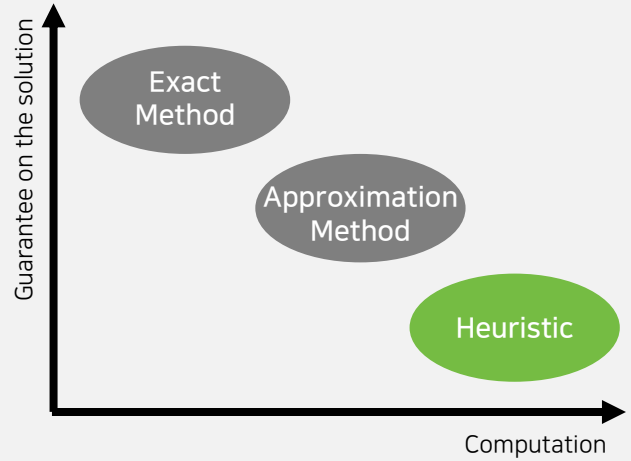
[CVRP Solution]

[Coupang Water Deliver]

Routing Problem Solvers

Classical Method to Solve Routing Problems

Exact Method	✓ Theoretical guarantee of finding the optimal solution ✓ Exponential complexity in the worst case
Approximation Method	✓ Find suboptimal solutions with probable worst-case guarantees in polynomial time ✓ Only exist for specific problems with poor approximation ratios
Heuristic	✓ Most commonly applied ✓ No theoretical guarantee on the solution quality ✓ Find desirable solutions within reasonable computational time



Deep Learning Approaches for Heuristics

Construction Heuristics

- Consider the routing problems as a sequence generation task
  - ✓ Seq2Seq, RNN, Attention
- Try additional procedure to overcome the limitation
  - ✓ Adding a node to a partial solution at each step
  - ✓ Sampling or beam search
- Limited capabilities since they rely on the same trained construction policy.

Improvement Heuristics

- Traditional improvement heuristics
  - ✓ Guided by hand-crafted search policies
  - ✓ Require substantial domain knowledge to design
- **RL formulation for the improvement heuristics**
  - ✓ **Novel architecture based on self-attention to parameterize the policy.**
  - ✓ **Use commonly used pairwise local operators such as 2-opt.**

Related Work

Related Work

Construction Heuristics

Pointer Network

- Vinyals et al. (2015)
  - ✓ RNN based Seq2Seq model
  - ✓ Supervised way to solve TSP
- Bello et al. (2017)
  - ✓ Use RL to train pointer network
  - ✓ No need to label optimal samples
- Nazari et al. (2018)
  - ✓ Linear map each node to higher dimensional space

Attention

- Kool et al. (2019)
  - ✓ Replace RNN with attention module
- Kaempfer et al. (2018)
  - ✓ Permutation invariant pooling
- Deudon et al. (2018)
  - ✓ Attention based embedding
  - ✓ Rely on additional 2-opt based local search

Others

- Khalil et al. (2017)
  - ✓ Adoped deep Q-network
  - ✓ GNN represents the internal states
- Nowak et al. (2017)
  - ✓ GNN learns normalized embeddings
  - ✓ Reconstruct adjacent matrix of TSP graph in supervised way

Improvement Heuristics

NewRewriter (SOTA)

- Chen et al. (2019)
  - ✓ On CVRP, outperform Kool et al. (2019) which use attention
  - ✓ Needs to train two policies to separately decide the rewritten region and solution selection
  - ✓ Relies on complex node features and customized local operations

- Change into only one policy network
- Use only raw features
- Typical local operators that are commonly applied to routing problems

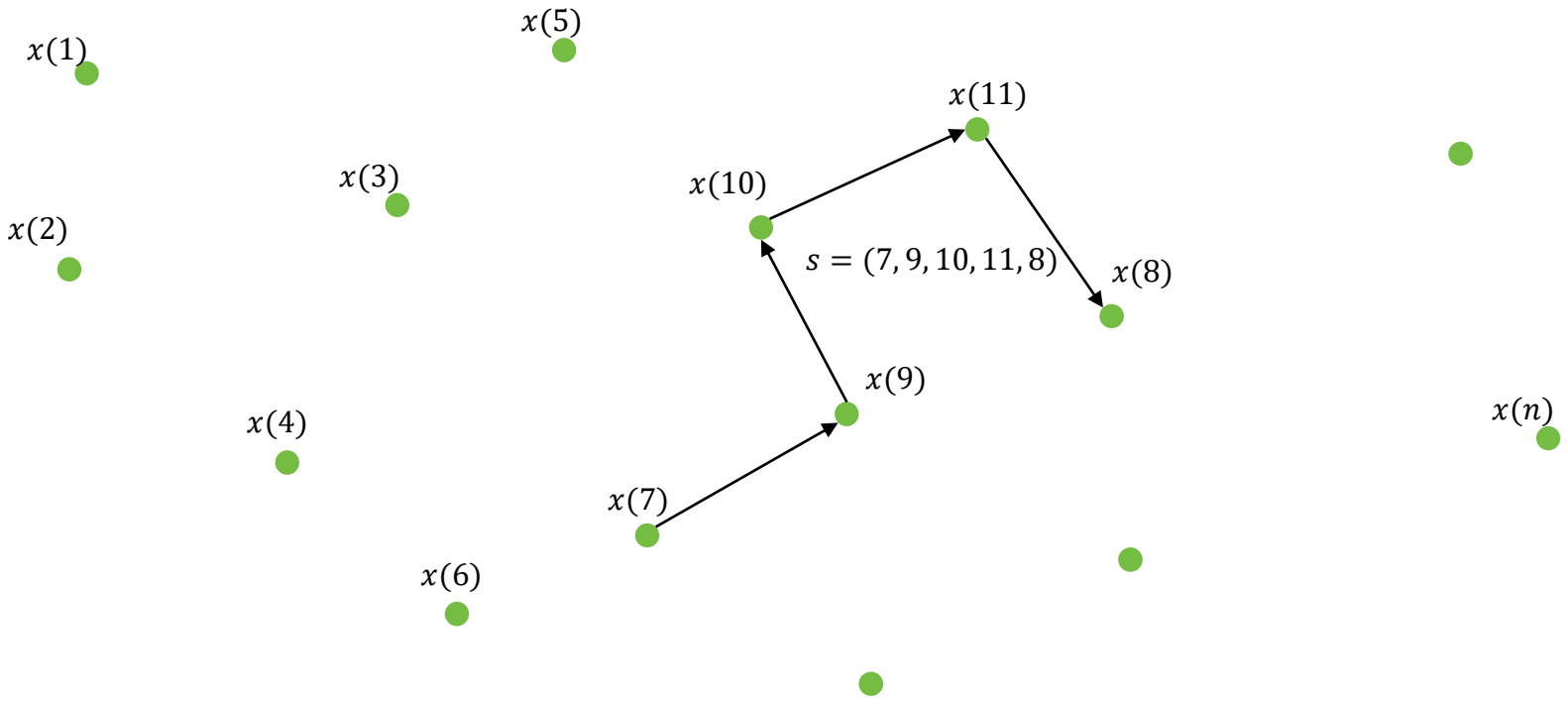
Preliminaries

- ✓ TSP and CVRP
- ✓ Improvement Heuristics

TSP and CVRP

General Setting of Routing Problem

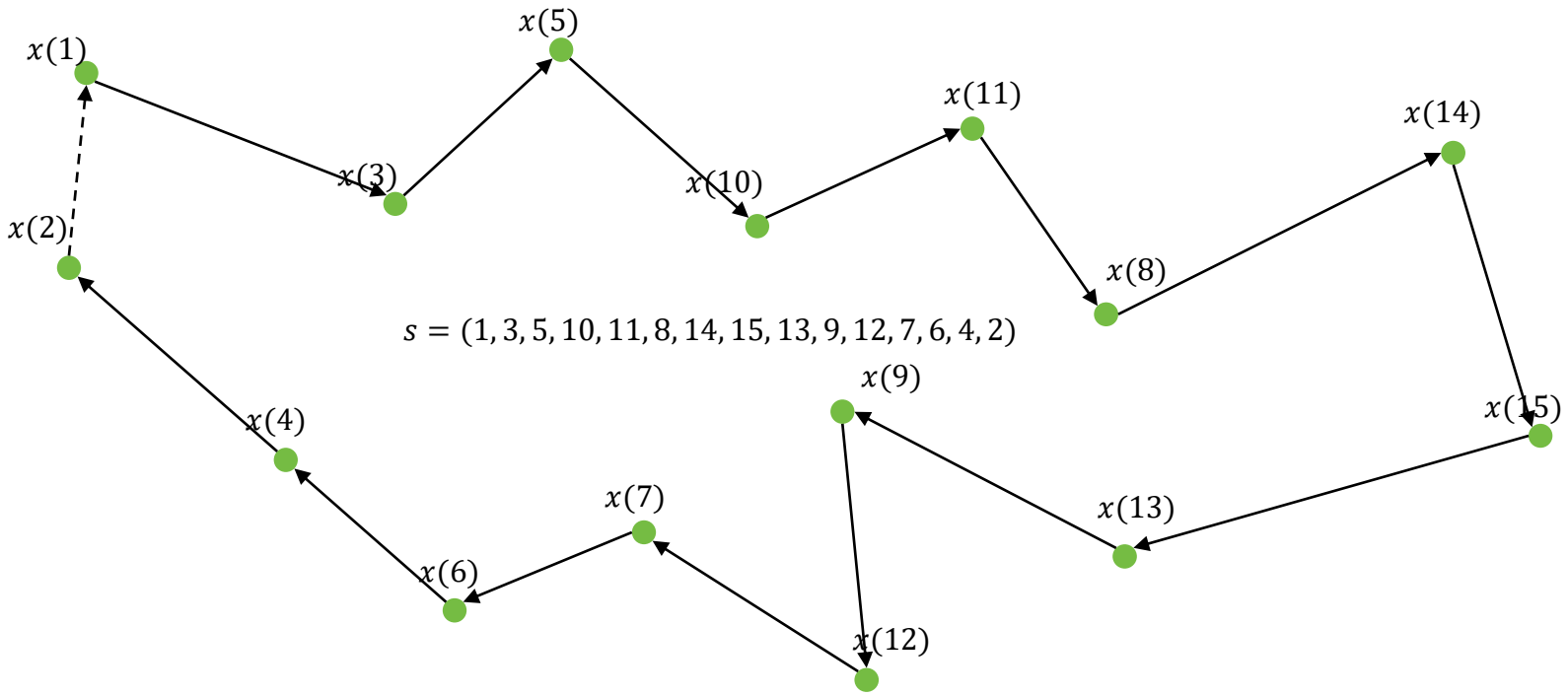
- Instance of routing problems can be defined on a graph with a set of n nodes $V = \{1, \dots, n\}$.
- Each $v \in V$ has features $x(v)$
- Solution $s = (s^1, \dots, s^I)$ is a tour, i.e., a sequence of nodes with length I, with each element $s^i (i \in \{1, \dots, I\})$ being a node in V.
- A feasible tour should satisfy problem-specific constraints



TSP and CVRP

TSP (Travelling Salesman Problem)

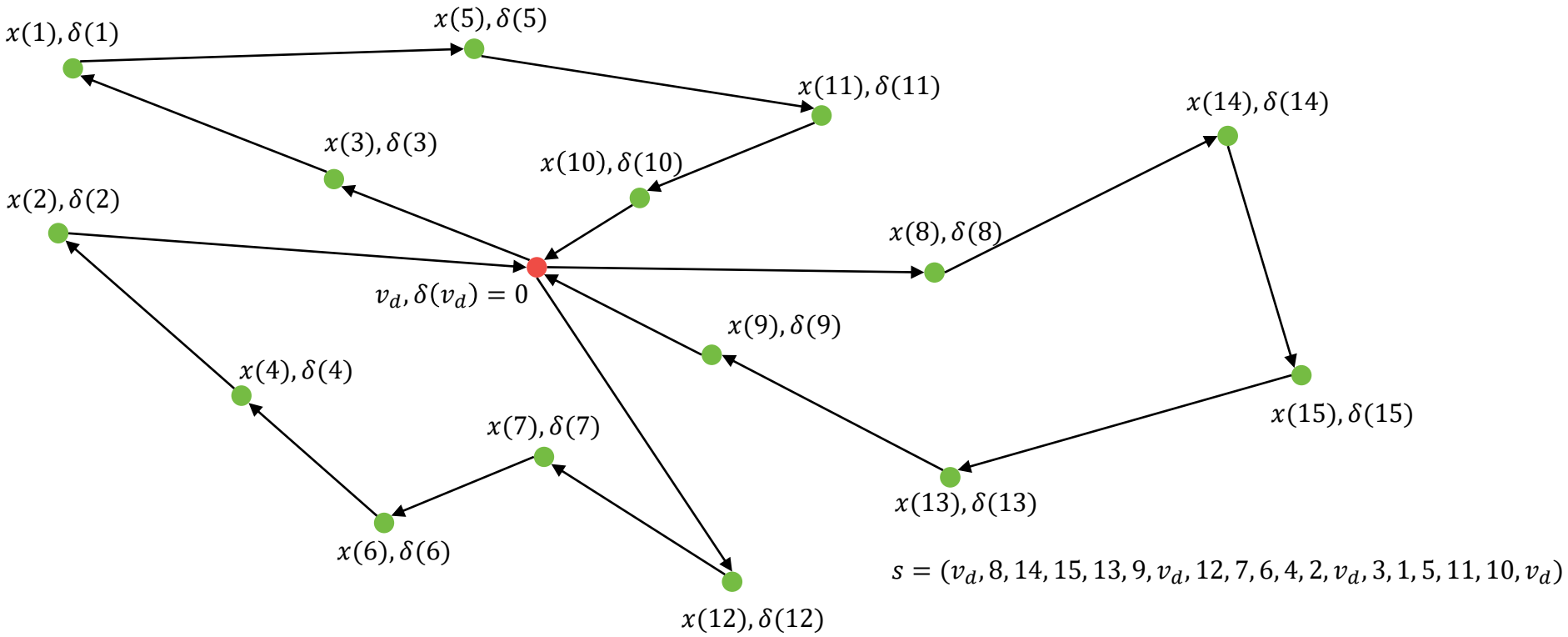
- The tour visits each node exactly once, hence I = n.



TSP and CVRP

CVRP (Capacitated Vehicle Routing Problem)

- The original n nodes represent customers, each with demand $\delta(v)$.
- Another node v_d called depot is added to V and $\delta(v_d) = 0$.
- The tour consists of multiple routes $(r_1, \dots, r_M), M > 1$.
- Each customer must be visited exactly once but v_d could be visited multiple times, hence $I > n + 1$.
- Additionally, the total customer demand on each route can not surpass the given capacity D.

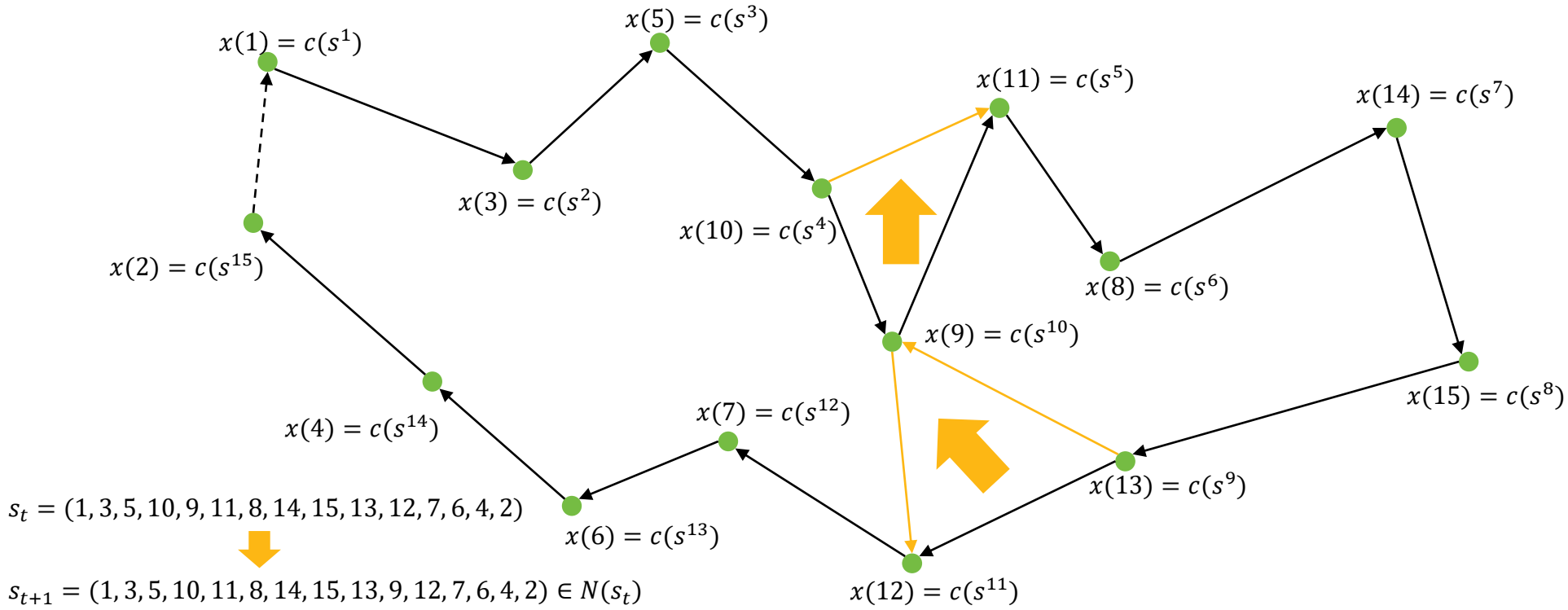


$s = (v_d, 8, 14, 15, 13, 9, v_d, 12, 7, 6, 4, 2, v_d, 3, 1, 5, 11, 10, v_d)$

Improvement Heuristics

Improvement Heuristics

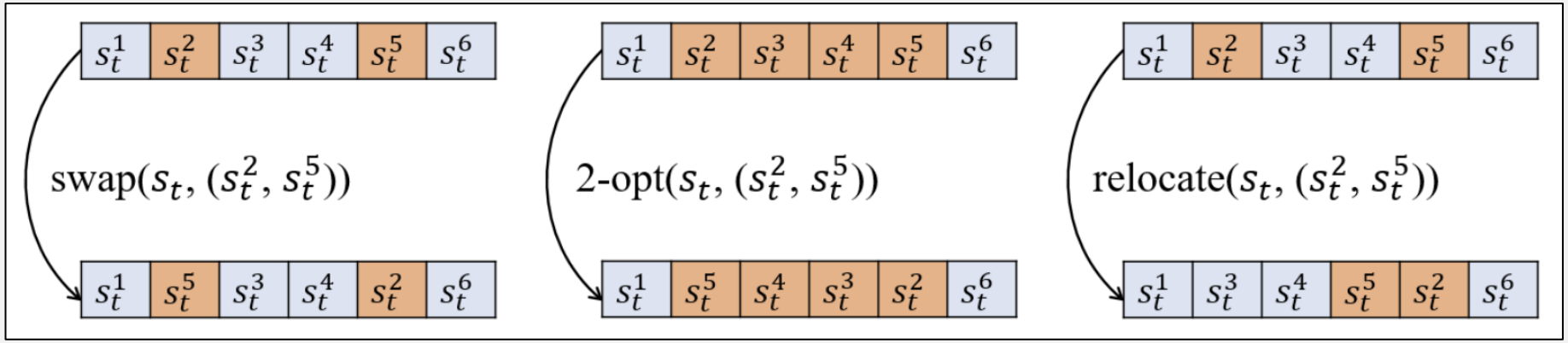
- Starting from an initial solution s_0, improvement heuristics iteratively replace the solution.
- Replace the current solution s_t at step t with a new solution s_{t+1} picked from neighborhood $N(s_t)$, towards the direction of minimizing f.
- Picks from $N(s_t)$ the solution $\bar{s}_{t+1}$ with the smallest f, replaces s_t with $\bar{s}_{t+1}$ only if $f(\bar{s}_{t+1}) < f(s_t)$
- Terminates when no such solution exists



Improvement Heuristics

Pairwise Operators

- Transform a solution s_t to s_{t+1} by performing operation l on a pair of nodes $\left(s_t^i, s_t^j\right), i.e., s_{t+1} = l(s_t, (s_t^i, s_t^j))$



[Three Typical Pairwise Operators]

RL Framework to get solution

- Use pairwise operators as an action of RL agent
- Follow the “always accept” rule, i.e., the solution picked by them policy will always be accepted, to avoid being stuck in local minimum
- User-specified maximum step T to stop the rum

The Method

- ✓ RL Formulation
- ✓ Policy Network
- ✓ Training Algorithm
- ✓ Deployment

RL Formulation

Markov Decision Process (MDP)

- Transform a solution s_t to s_{t+1} by performing operation l on a pair of nodes $\left(s_t^i, s_t^j\right), i.e., s_{t+1} = l(s_t, (s_t^i, s_t^j))$

State

- s_t, a sequence of nodes.

Action

- $a_t = (s_t^i, s_t^j)$ selected from the s_t

Transition

- $s_t = \left(\dots, s_t^i, s_t^{i+1}, \dots, s_t^{j-1}, s_t^j, \dots\right)$
- $s_{t+1} = (\dots, s_t^j, s_t^{j-1}, \dots, s_t^{i+1}, s_t^i, \dots)$

Reward

- Mission : Improve the initial solution as much as possible within the step limit T
- $r_t = r(s_t, a_t, s_{t+1}) = f(s_t^*) - min\{f(s_t^*), f(s_{t+1})\}$

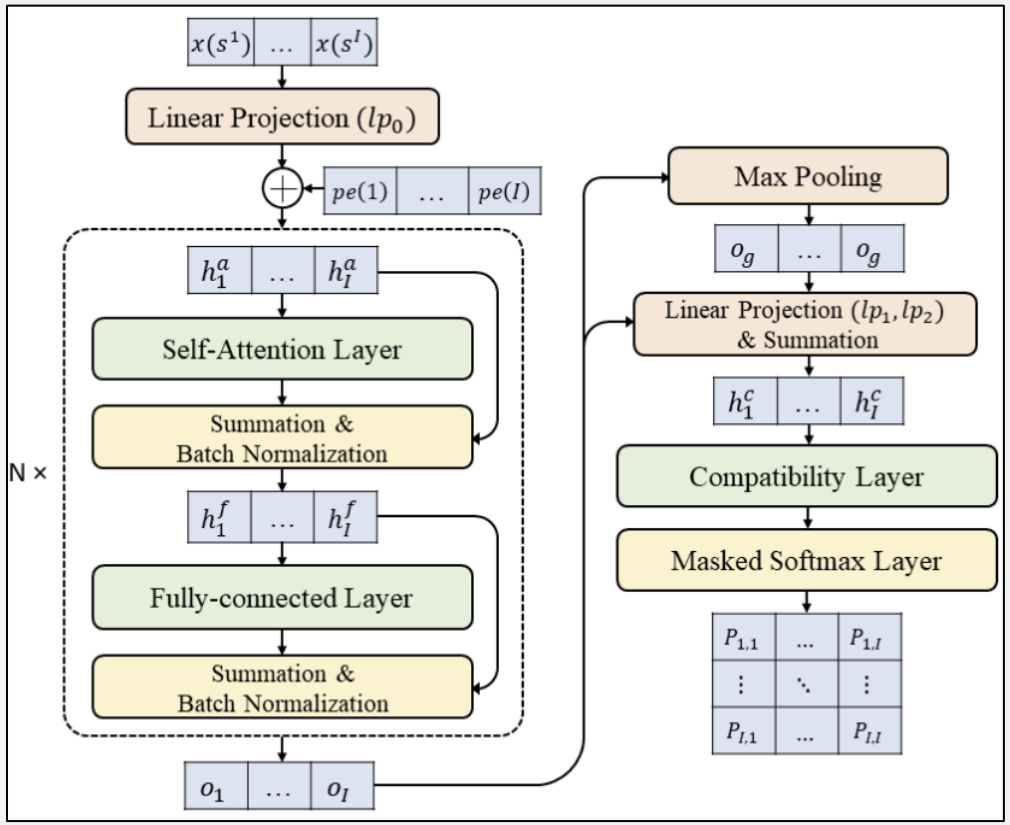
Policy

- Stochastic policy π picks an action at at each step t, which will lead to s_{t+1}, until reaching the step limit T
- $P(s_T|s_0) = \prod_{t=0}^{T-1} \pi(a_t|s_t)$

Policy Network

Policy Network

- The former part elegantly embeds the nodes in sequence.
- The latter part produces a probability matrix of selecting each node pair.
- Each element in the matrix refers to the probability of selecting the corresponding node pair for local operation



[The architecture of policy network]

Node Embedding

- Positional Encoding
  - ✓ $h_i = h_i + pe(i, \cdot)$
  - ✓ $pe(i,d)\begin{cases}\sin(i/10000^{\frac{\lfloor d/2 \rfloor}{d_h}}), if\ d\ mod\ 2 = 0\\ \cos(i/10000^{\frac{\lfloor d/2 \rfloor}{d_h}}), if\ d\ mod\ 2 = 1\end{cases}$
- Self-attention Layer

Node Pair Selection

- Compatibility Layer
  - ✓ Vaswani et al. (2017) use compatibility layer to represent the relation between words in sentence
  - ✓ Predict the node pair selection
- Masked softmax layer
  - ✓ $\tilde{Y}_{ij} = \begin{cases} C \cdot \tanh(Y_{ij}), & if\ i \neq j \\ -\infty, & if\ i = j \end{cases}$

Training Algorithm

Training Algorithm

ALGORITHM 1: n-step actor-critic (continuing task)

Input: actor network π_θ with trainable parameters θ; critic network v_ϕ with trainable parameters ϕ; number of epochs E, batches B; step limit T.

for $e = 1, 2, \cdots, E$ **do**
  generate M problem instances randomly;
  for $b = 1, 2, \cdots, B$ **do**
    retrieve batch M_b; $t \leftarrow 0$;
    while $t < T$ **do**
      reset gradients: $d\theta \leftarrow 0$; $d\phi \leftarrow 0$;
      $t_s = t$; get state s_t;
      while $t - t_s <$ **n** *and* $t \neq T$ **do**
        sample a_t based on $\pi_\theta(a_t|s_t)$;
        receive reward r_t and next state s_{t+1};
        $t \leftarrow t + 1$;
      end
      $R = v_\phi(s_t)$;
      for $i \in \{t-1, \ldots, t_s\}$ **do**
        $R \leftarrow r_i + \gamma R$; $\delta \leftarrow R - v_\phi(s_i)$;
        $d\theta \leftarrow d\theta + \sum_{M_b} \delta \nabla log \pi_\theta(a_i|s_i)$;
        $d\phi \leftarrow d\phi + \sum_{M_b} \delta \nabla v_\phi(s_i)$;
      end
      update θ and ϕ by $\frac{d\theta}{|M_b|(t-t_s)}$ and $\frac{d\phi}{|M_b|(t-t_s)}$
    end
  end
end

[The architecture of policy network]

- Adopt the actor-critic algorithm with Adam optimizer to train π_θ based on the REINFORCE
- Extra trainable critic network updated by bootstrapping

Actor Network

- Same policy network designed before

Critic Network

- Critic $v_\emptyset$ estimate the cumulative reward at each state.`
- Our design of $v_\emptyset$ is similar to that of actor, except that:
  1) mean-pooling is used to obtain graph embedding;
  2) the fused node embeddings are processed by a FC layer similar to the one used in policy network, but with single-value output.
- Use n-step return for efficient reward propagation and bias-variance trade-off

Deployment

Deployment

TSP

- Given a solution, the node feature is $x(s_i) = c(s_i)$.
- To avoid solution cycling, mask the node pair selected at the previous step to forbid the local operation to be reversed

CVRP

- Add multiple dummy depots to the end of initial solutions
  - ✓ Process a batch of instances using solutions with the same length
  - ✓ Number and positions of depot in a solution (sequence) can be learned automatically
  - ✓ $s_t = (v_d, s_1, s_2, v_d, \dots)$ can be turned into $s_{t+1} = (v_d, v_d, s_2, s_1, \dots)$ by 2-opt, with s_{t+1} being equivalent to $= (v_d, v_d, s_2, \dots)$
- Node feature as a 7-dimensional vector $x(s_i) = (c(s_{i-1}), c(s_i), c(s_{i+1}), \delta(s_i))$.
- Mask the node pairs in the matrix P that result in infeasible solutions, as well as the one selected at the previous step.

Experimental Results

- ✓ Comparison with State-of-the-art Methods
- ✓ Comparison with Conventional Policies
- ✓ Enhancement by Diversifying
- ✓ Generalization Analysis
- ✓ Visualization
- ✓ Test on Real-World Dataset

Experimental Setting

Experiment Instance

- Euclidean TSP and CVRP with 20, 50 and 100 nodes.
  - ✓ TSP20, TSP50, TSP100, CVRP20, CVRP50 and CVRP100
- Coordinates of each node are randomly sampled in the unit square $[0,1] \times [0,1]$ with uniform distribution.

TSP

- 10,240 random instances are generated on the fly and split into 10 batches
- Random initial solution
- Set small step limit $T = 200$
- $\gamma = 0.99, n = 4\ for\ n - step$

CVRP

- 3,840 instances in each epoch and split into 10 batches
- Nearest insertion heuristic create initial solution
- Add dummy depot $I^* = 40, 100\ and\ 125\ for\ CVRP20, 50\ and\ 100$
- $T = 360,\ n = 10\ and\ \gamma = 0.996\ for\ CVRP20$
- $T = 480,\ n = 12\ and\ \gamma = 0.996\ for\ CVRP50\ and\ CVRP100$

- Train 200 epochs with initial learning rate 10^{-4} and decaying 0.99 per epoch for convergence

Experiment Specifications

- Single Tesla V100 GPU
  - ✓ Each epoch takes on average 8m 20s, 16m 30s and 31m 00s for TSP20, TSP50 and TSP100
  - ✓ Each epoch takes on average 20m 17s, 56m 25s and 58m 53s for CVRP20, CVRP50 and CVRP100
- 2-opt operator as pairwise operator
  - ✓ Best result among 2-opt, node swap and relocation

Comparison with State-of-the-art Methods

Baselines

Concorde	✓ Efficient exact solver specialized for TSP
LKH3	✓ Heuristic solver that achieves state-of-the-art performance on various routing problems
OR-Tools	✓ A mature and widely used solver for routing problems based on metaheuristics
Attention Model	✓ State-of-the-art DL based methods on TSP and CVRP which learn construction heuristics
NewRewriter	✓ State-of-the-art DL based methods on CVRP which learn improvement heuristics

Performance

Methods	TSP20			TSP50			TSP100			CVRP20			CVRP50			CVRP100		
	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time
Concorde	3.83	0.00%	5m	5.69	0.00%	13m	7.76	0.00%	1h	-	-	-	-	-	-	-	-	-
LKH3	3.83	0.00%	42s	5.69	0.00%	6m	7.76	0.00%	25m	6.11	0.00%	1h	10.38	0.00%	5h	15.64	0.00%	9h
OR-Tools	3.86	0.94%	1m	5.85	2.87%	5m	8.06	3.86%	23m	6.46	5.68%	2m	11.27	8.61%	13m	17.12	9.54%	46m
AM ($\mathbb{N}$=1,280)	3.83	0.06%	14m	5.72	0.48%	47m	7.94	2.32%	1.5h	6.26	2.56%	22m	10.61	2.20%	53m	16.17	3.34%	2h
AM ($\mathbb{N}$=5,000)	3.83	0.04%	47m	5.72	0.47%	2h	7.93	2.18%	5.5h	6.25	2.31%	1.5h	10.59	2.01%	3.5h	16.12	3.03%	8h
NeuRewriter	-	-	-	-	-	-	-	-	-	6.15	-	-	10.51	-	-	16.10	-	-
Ours (T=1,000)	**3.83**	**0.03%**	12m	5.74	0.83%	16m	8.01	3.24%	25m	6.16	0.90%	23m	10.71	3.16%	48m	16.30	4.16%	1h
Ours (T=3,000)	**3.83**	**0.00%**	39m	**5.71**	**0.34%**	45m	**7.91**	**1.85%**	1.5h	**6.14**	**0.61%**	1h	10.55	1.65%	2h	16.11	2.99%	3h
Ours (T=5,000)	**3.83**	**0.00%**	1h	**5.70**	**0.20%**	1.5h	**7.87**	**1.42%**	2h	**6.12**	**0.39%**	2h	**10.45**	**0.70%**	4h	**16.03**	**2.47%**	5h

[Comparison with state-of-the-art methods]

Comparison with Conventional Policies

Baselines

- First-improvement select the first cost-reducing solution in the neighborhood.
- Best-improvement select the best cost-reducing solution in the neighborhood.

Performance

	Methods	TSP 20	TSP 50	TSP 100	CVRP 20	CVRP 50	CVRP 100
First	T=1,000	3.84	5.81	8.17	6.18	11.08	17.14
	T=3,000	3.84	5.75	8.04	6.16	10.93	16.93
	T=5,000	3.84	5.73	8.00	6.15	10.87	16.85
Best	T=1,000	3.84	5.75	8.05	6.15	10.79	16.72
	T=3,000	3.84	5.71	7.99	6.14	10.70	16.61
	T=5,000	3.84	5.70	7.94	6.13	10.67	16.55
Ours	T=1,000	**3.83**	5.74	8.01	6.16	10.71	**16.30**
	T=3,000	**3.83**	5.71	**7.91**	6.14	**10.55**	**16.11**
	T=5,000	**3.83**	**5.70**	**7.87**	**6.12**	**10.45**	**16.03**

[Comparison with Conventional Policies]

Enhancement by Diversifying

Baselines

- Multi-Run (MR) directly run the final policy multiple times.
- Multi-Policy (MP) run polices of the last several training epochs instead of only the final one.

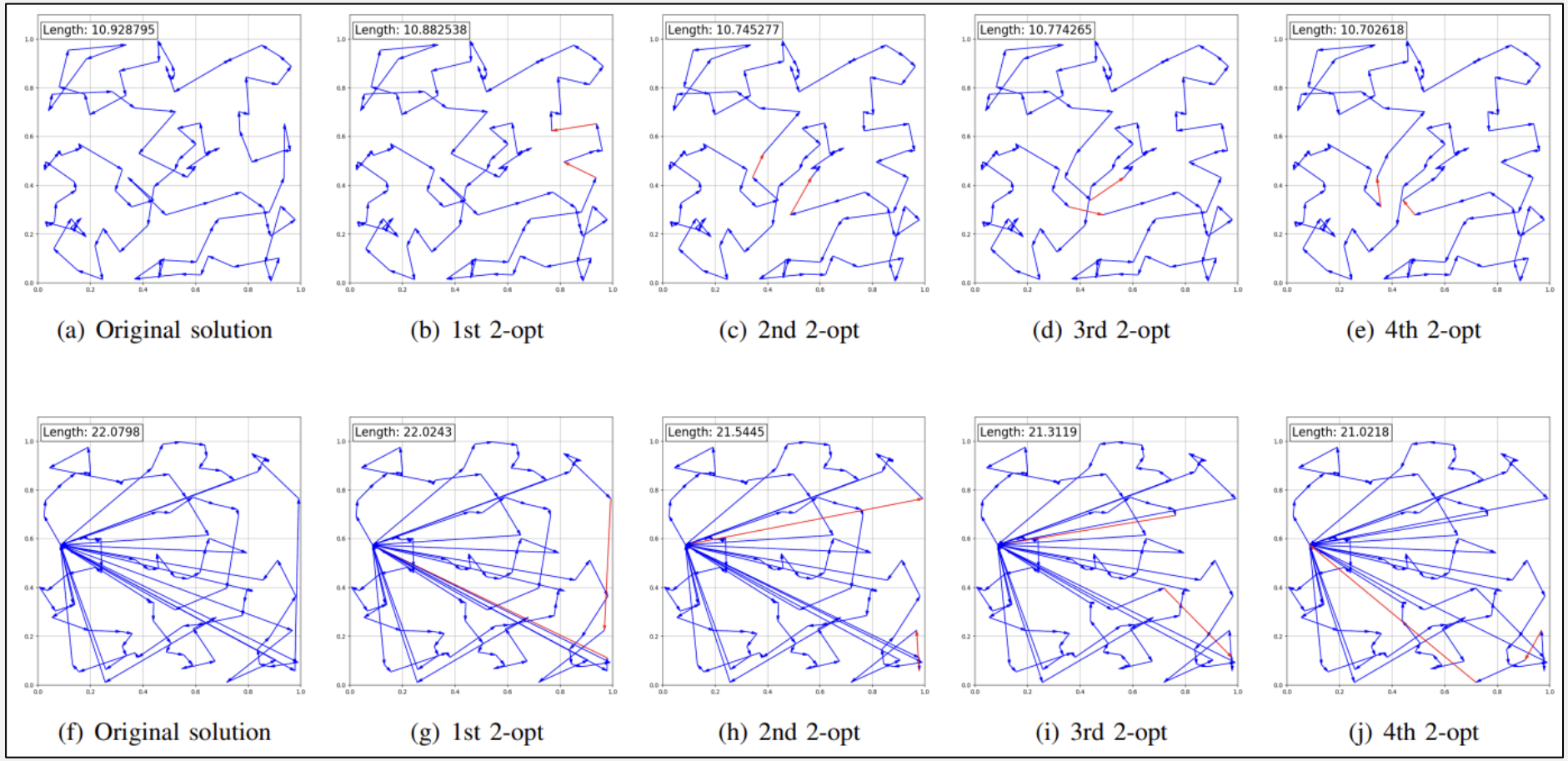
Performance

Methods		TSP			CVRP		
		20	50	100	20	50	100
MP(4)[1]	T=1,000	3.831	5.707	7.897	6.144	10.452	16.110
	T=3,000	3.831	5.701	7.839	6.134	10.426	16.055
	T=5,000	3.831	5.700	7.822	6.123	10.416	16.018
MP(8)	T=1,000	3.831	5.703	7.875	6.134	10.436	16.036
	T=3,000	3.831	5.700	7.824	6.127	10.404	16.000
	T=5,000	3.831	5.699	7.811	6.121	10.395	15.967
MR(4)[2]	T=1,000	3.832	5.707	7.895	6.144	10.482	16.110
	T=3,000	3.831	5.702	7.836	6.132	10.417	15.979
	T=5,000	3.831	5.700	7.821	6.122	10.399	15.923
MR(8)	T=1,000	3.831	5.703	7.866	6.135	10.445	16.057
	T=3,000	3.831	5.700	7.820	6.125	10.393	15.922
	T=5,000	3.831	**5.699**	**7.807**	**6.117**	**10.384**	**15.880**

[Comparison with Conventional Policies]

Visualization

Visualization of TSP and CVRP



[Visualization of learned policies on TSP and CVRP]

Test on Real-World Dataset

Real-World Dataset

TSPlib	✓ http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.htm
CVRPlib	✓ http://vrp.galgos.inf.puc-rio.br/index.php/en/

Test on CVRP

Instance	Opt.	OR-Tools	AM (N=1280)	AM (N=5000)	Ours (T=5,000)
X-n101-k25	27,591	29,405	39,437	37,702	**29,716**
X-n106-k14	26,362	27,343	28,320	28,473	**27,642**
X-n110-k13	14,971	16,149	15,627	**15,443**	15,927
X-n115-k10	12,747	13,320	13,917	**13,745**	14,445
X-n120-k6	13,332	14,242	14,056	**13,937**	15,486
X-n125-k30	55,539	58,665	75,681	75,067	**60,423**
X-n129-k18	28,940	31,361	30,399	**30,176**	32,126
X-n134-k13	10,916	13,275	13,795	13,619	**12,669**
X-n139-k10	13,590	15,223	14,293	**14,215**	15,627
X-n143-k7	15,700	17,470	17,414	**17,397**	18,872
X-n148-k46	43,448	46,836	79,611	79,514	**50,563**
X-n153-k22	21,220	22,919	38,423	37,938	**26,088**
X-n157-k13	16,876	17,309	21,702	21,330	**19,771**
X-n162-k11	14,138	15,030	15,108	**15,085**	16,847
X-n167-k10	20,557	22,477	22,365	**22,285**	24,365
X-n172-k51	45,607	50,505	86,186	87,809	**51,108**
X-n176-k26	47,812	52,111	58,107	58,178	**57,131**
X-n181-k23	25,569	26,321	27,828	27,520	**27,173**
X-n186-k15	24,145	26,017	25,917	**25,757**	28,422
X-n190-k8	16,980	18,088	37,820	36,383	**20,145**
X-n195-k51	44,225	50,311	79,594	79,276	**51,763**
X-n200-k36	58,578	61,009	78,679	76,477	**64,200**
Avg. Gap	0	8.06%	32.97%	31.62%	14.27%

[Generalization on CVRPlib]

Test on TSP

Instance	Opt.	OR-Tools	AM (N=1,280)	AM (N=5,000)	Ours (T=3,000)
eil51	426	436	436	**435**	438
berlin52	7,542	7,945	7,717	**7,668**	8,020
st70	675	683	691	**690**	706
eil76	538	561	564	**563**	575
pr76	108,159	111,104	111,605	111,250	**109,668**
rat99	1,211	1,232	1,483	**1,394**	1,419
KroA100	21,282	21,448	44,385	38,200	**25,196**
KroB100	22,141	23,006	35,921	35,511	**26,563**
KroC100	20,749	21,583	31,290	30,642	**25,343**
KroD100	21,294	21,636	34,775	32,211	**24,771**
KroE100	22,068	22,598	28,596	27,164	**26,903**
rd100	7,910	8,189	8,169	8,152	**7,915**
eil101	629	664	668	667	**658**
lin105	14,379	14,824	53,308	51,325	**18,194**
pr107	44,303	45,072	208,531	205,519	**53,056**
pr124	59,030	62,519	183,858	167,494	**66,010**
bier127	118,282	122,733	210,394	207,600	**142,707**
ch130	6,110	6,284	6,329	**6,316**	7,120
pr136	96,772	102,213	103,470	**102,877**	105,618
pr144	58,537	59,286	225,172	183,583	**71,006**
ch150	6,528	6,729	6,902	6,877	7,916
KroA150	26,524	27,592	44,854	42,335	**31,244**
KroB150	26,130	27,572	45,374	43,114	**31,407**
pr152	73,682	75,834	106,180	103,110	**85,616**
u159	42,080	45,778	124,951	115,372	**51,327**
rat195	2,323	2,389	3,798	3,661	**2,913**
d198	15,780	15,963	78,217	68,104	**17,962**
KroA200	29,368	29,741	62,013	58,643	**35,958**
KroB200	29,437	30,516	54,570	50,867	**36,412**
ts225	126,643	128,564	141,951	**141,628**	158,748
tsp225	3,916	4,046	25,887	24,816	**4,701**
pr226	80,369	82,968	105,724	101,992	**97,348**
gil262	2,378	2,519	2,695	**2,693**	2,963
pr264	49,135	51,954	361,160	338,506	**65,946**
a280	2,579	2,713	13,087	11,810	**2,989**
pr299	48,191	49,447	513,809	513,673	**59,786**
Avg. Gap	0	3.46%	146.12%	133.54%	17.12%

[Generalization on TSPlib]

Conclusions and Future Work

Conclusion and Future Work

Contribution

- Propose a deep reinforcement learning framework to **automatically learn improvement heuristics** for routing problems.
- Design a novel neural architecture based on self-attention to enable learning with pairwise local operators

Results

- **Outperforms state-of-the-art deep models on both TSP and CVRP**, and further narrows the gap to highly optimized solvers.
- Learned policies generalize well to different initial solutions and problem sizes.
- Give reasonably good solutions on real-world datasets.

Future Work

- Extend the current framework to support multiple operators
- Apply to learn better solution picking policies for more advanced search schemes such as simulated annealing, tabu search, large neighborhood search, and LKH
- Apply to other important types of combinatorial optimization problems, e.g., scheduling

감사합니다