Learning Generalizable Models for Vehicle Routing Problems via Knowledge Distillation

Jieyi Bi[1,†], Yining Ma[2,†], Jiahai Wang[1,*] , Zhiguang Cao[3,*] , Jinbiao Chen[1],Yuan Sun[4], and Yeow Meng Chee[2]

[1]School of Computer Science and Engineering, Sun Yat-sen University
[2]National University of Singapore
[3]Singapore Institute of Manufacturing Technology, A*STAR
[4]University of Melbourne

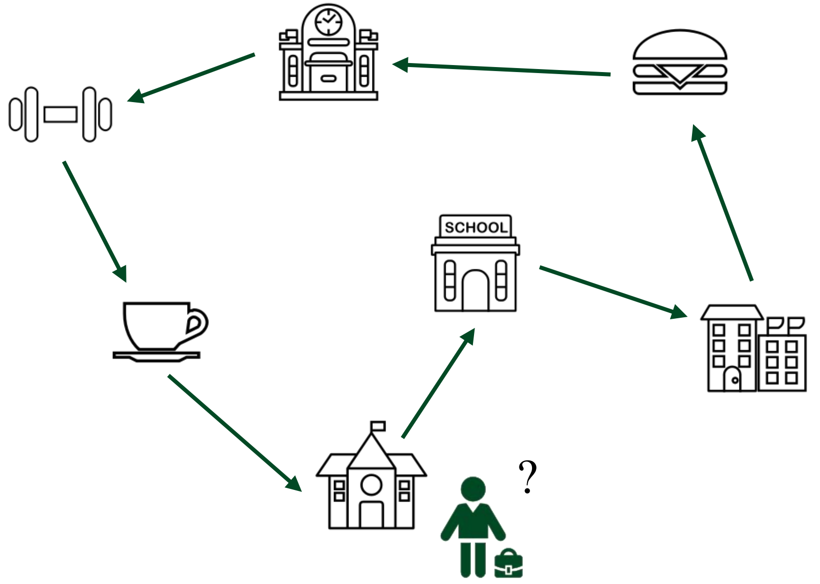
[†] Equally contributed; * Jiahai Wang and Zhiguang Cao are the corresponding authors.

Outlines

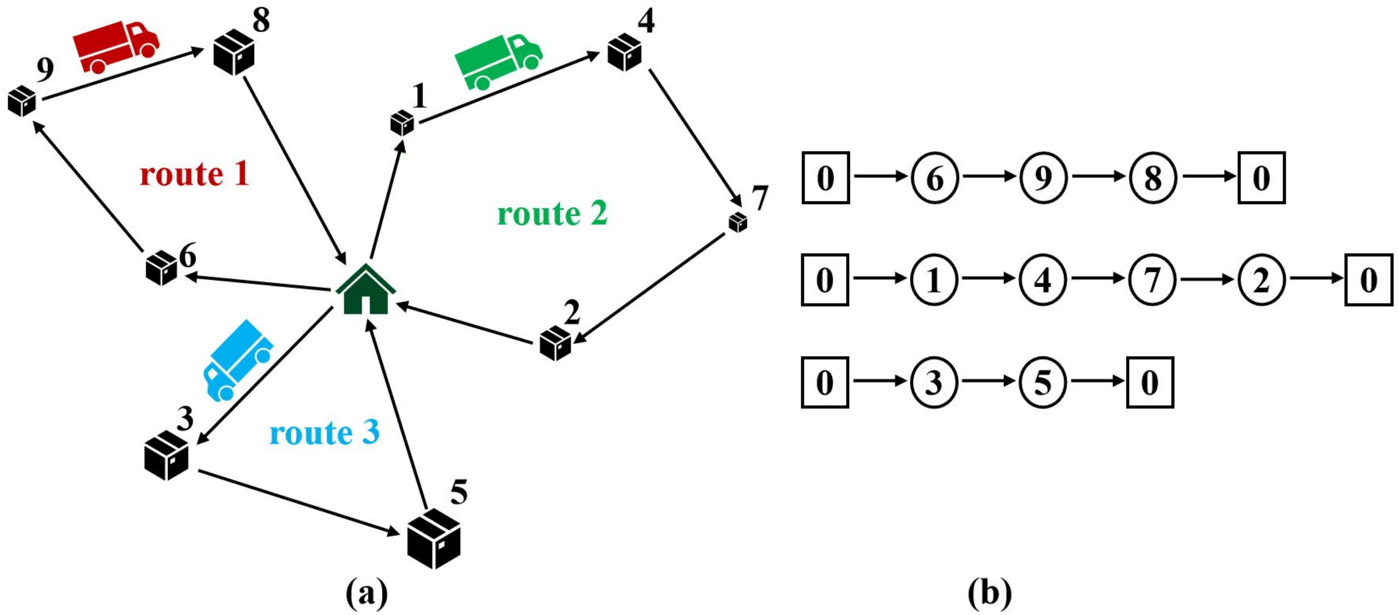
- Introduction & Motivation
- Methodology
- Experimental results
- Conclusion & Future work

1. Introduction

- **Vehicle Routing Problem (VRP)** is a class of NP-hard **combinatorial optimization problems**.
- Two representative VRPs: TSP and CVRP



An example for Traveling Salesman Problem (TSP)



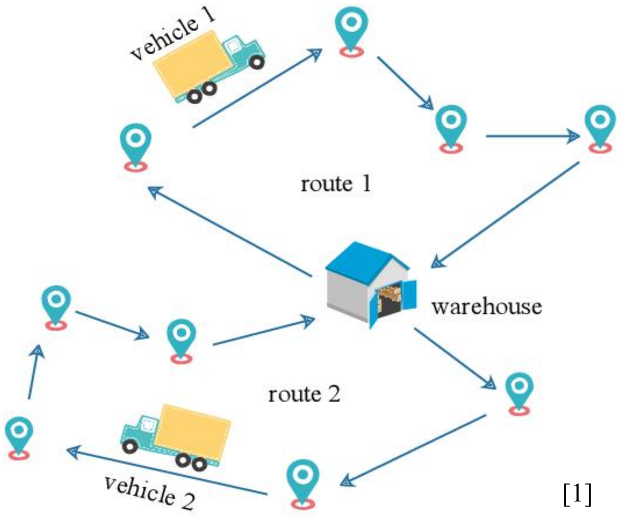
An example for Capacitated Vehicle Routing Problem (CVRP)

PROBLEM DEFINITION: We define VRPs over a complete graph $G = \{V, E\}$, where $v_i \in V$ represents the (customer) node, $e(v_i, v_j) \in E$ represents the edge between two nodes. $C[e(v_i, v_j)]$ represents the cost (we use length in this paper) of the edge. By referring tour τ (a.k.a. solution) to a permutation of nodes in V, **the objective is usually to find the optimal tour $\tau*$ with the least total cost (length) over a finite searchspace S containing all possible tours.**

$$\tau^* = \underset{\tau' \in \mathcal{S}}{\arg\min}\, L(\tau'|\mathcal{G}) = \underset{\tau' \in \mathcal{S}}{\arg\min} \sum_{e(v_i, v_j) \in \tau'} C\left[e(v_i, v_j)\right].$$

1. Introduction

- Various **practical applications**: freight delivery, last-mile logistics, ride-hailing and etc.



[1]

photo credit: urbantz.com

photo credit: dreamstime.com

[1] Lu Duan, Yang Zhan, Haoyuan Hu, Yu Gong, et. al.. Efficiently solving the practical vehicle routing problem: A novel joint learning approach. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pages 3054–3063, 2020.

1. Motivation

- Recent neural methods for vehicle routing problems always train and test the deep models on the same instance distribution (i.e., uniform).
- **Cross-distribution generalization issue:** when the learned policy (trained on uniform distribution) is applied to infer the out-of-distribution (OoD) instances, the solution quality is usually low.

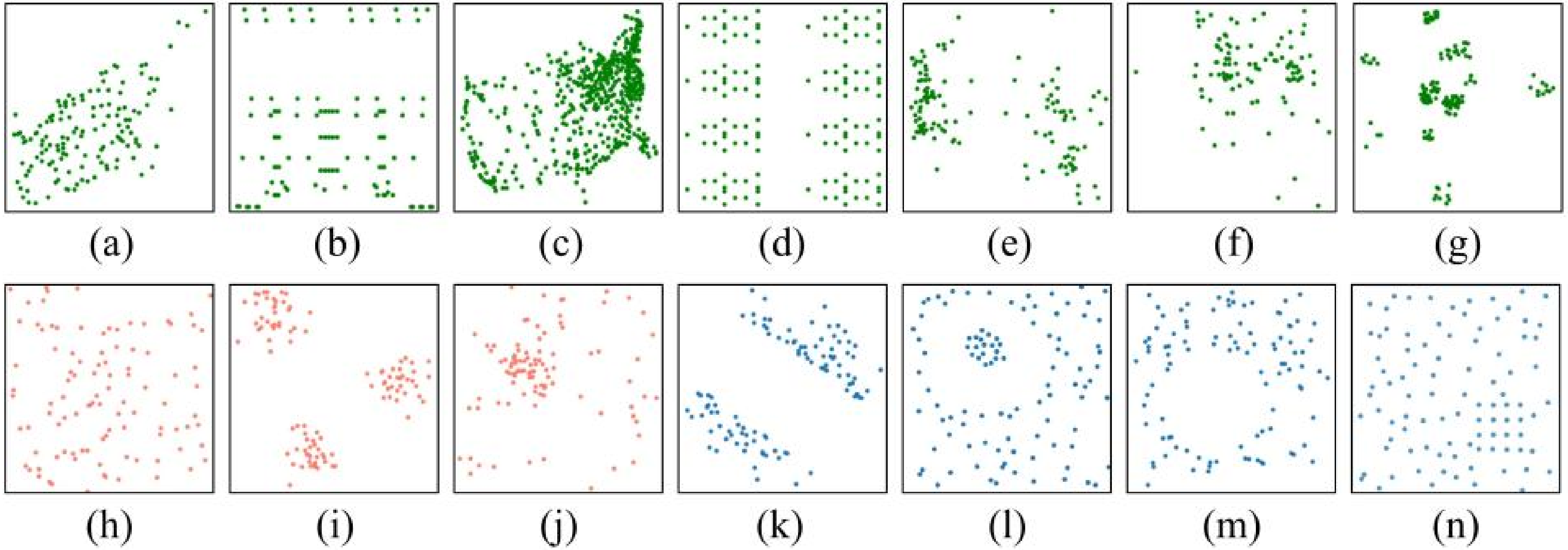
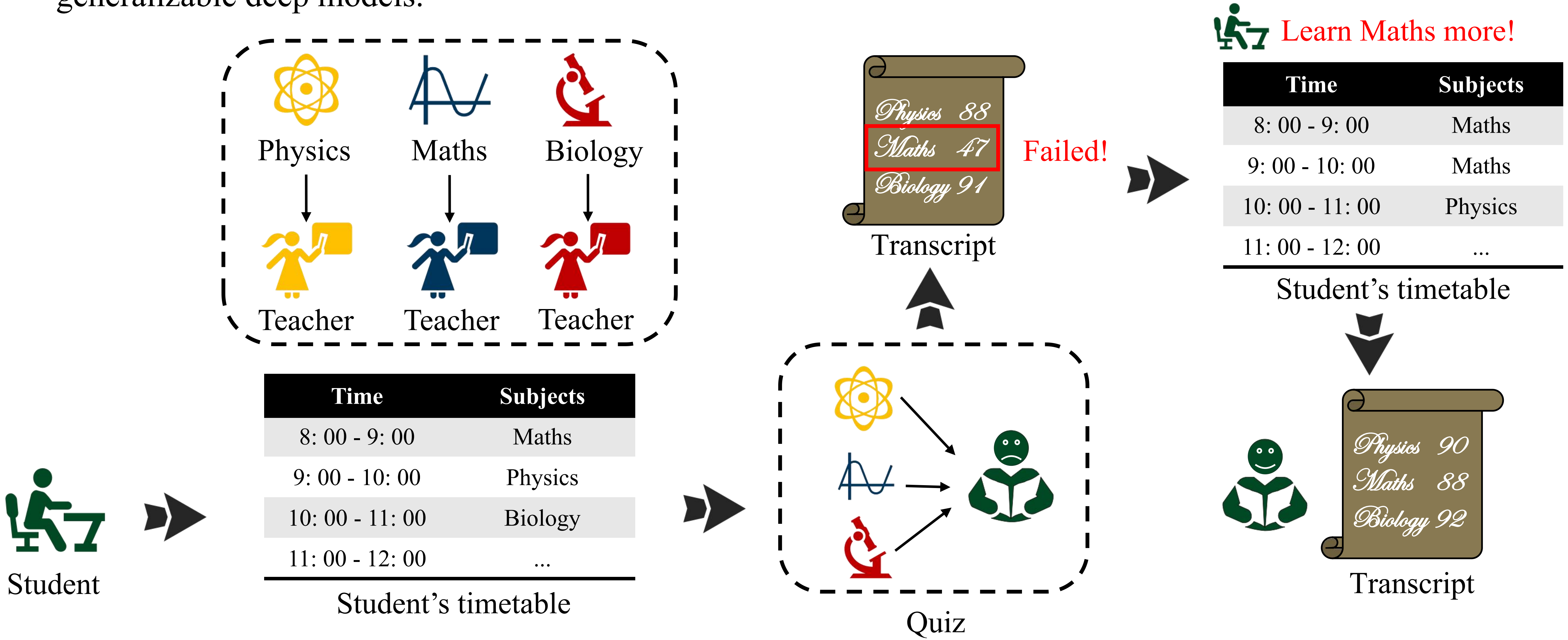


Figure 1: VRP instances following various distributions from the literature: (a) gr137, (b) lin105, (c) att532, (d) pr136, (e) X-n125-k30, (f) bier127, (g) Tai150d, (h) Uniform, (i) Cluster, (j) Mixed, (k) Expansion, (l) Implosion, (m) Explosion, (n) Grid, where instances (a)-(g) are from TSPLIB and CVRPLIB. In this paper, we consider instances following distributions (h)-(j) for training and other unseen distributions (k)-(n), as well as unseen benchmark datasets for testing.

1. Motivation

- To tackle the cross-distribution generalization concerns, we bring the knowledge distillation to this field and propose an **A**daptive **M**ulti-**D**istribution **K**nowledge **D**istillation scheme for learning more generalizable deep models.



Time	Subjects
8: 00 - 9: 00	Maths
9: 00 - 10: 00	Physics
10: 00 - 11: 00	Biology
11: 00 - 12: 00	...

Student's timetable

Time	Subjects
8: 00 - 9: 00	Maths
9: 00 - 10: 00	Maths
10: 00 - 11: 00	Physics
11: 00 - 12: 00	...

Student's timetable

2. Methodology

- **Three stages: teacher pre-training, student training and inference.**

Algorithm 1 Adaptive Multi-Distribution Knowledge Distillation (AMDKD)

Input: A backbone model $\mathcal{M}$ (e.g., AM), exemplar distributions $\mathcal{D}$ (e.g., $\mathcal{D} = \{U, C, M\}$).
Randomly initialize teacher networks θ_d^{T} ($\forall d \in \mathcal{D}$) and student network θ^{S} according to $\mathcal{M}$;
Perform teacher training or leverage pre-train ones (if any) to attain well-learned θ_d^{T} ($\forall d \in \mathcal{D}$);
for epoch $= 1, 2, ..., E$ **do**
  Pick distribution d and its teacher θ_d^{T} with an adaptive probability according to Eq. (5);
  for step $= 1, 2, ..., T$ **do**
    Let student θ^{S} sample tours $\tau_{\theta^{\mathrm{S}}}^{i}$ for each $\{\mathcal{G}_i\}_{i=1}^{B}$ according to its own policy $p_{\theta^{\mathrm{S}}}$;
    Get $\nabla\mathcal{L}_{\mathrm{Task}}$ by estimating Eq. (6) as per the original design of $\mathcal{M}$;
    Get $\nabla\mathcal{L}_{\mathrm{KD}}$ by computing the gradients of Eq. (7);
    $\theta^{\mathrm{S}} \leftarrow \theta^{\mathrm{S}} + \eta\nabla\mathcal{L}$ where $\nabla\mathcal{L} \leftarrow \alpha\nabla\mathcal{L}_{\mathrm{Task}} + (1-\alpha)\nabla\mathcal{L}_{\mathrm{KD}}$.
  end for
end for

$$p^{\mathrm{adaptive}}(d) = \begin{cases} \mathrm{Softmax}\left(\mathrm{AvgGap}\left[\{\tau_{\theta^{\mathrm{S}}}\}|\mathcal{Z}_d, \{\tau_{\mathrm{solver}}\}|\mathcal{Z}_d\right]\right), & \text{if } E \geq E' \\ \frac{1}{|\mathcal{D}|}, & \text{otherwise} \end{cases}$$

$$\mathcal{L}_{\mathrm{Task}} = -J(\theta^{\mathrm{S}}|d) = -\mathbb{E}_{\mathcal{G}\sim d, \tau\sim p_{\theta^{\mathrm{S}}}(\tau|\mathcal{G})}\left[L(\tau|\mathcal{G})\right]$$

$$\mathcal{L}_{\mathrm{KD}} = \frac{1}{B}\sum_{i=1}^{B}\sum_{a_j\in\tau_{\theta^{\mathrm{S}}}} p_{\theta_d^{\mathrm{T}}}(a_j|\mathcal{G}_i)\left(\log p_{\theta_d^{\mathrm{T}}}(a_j|\mathcal{G}_i) - \log p_{\theta^{\mathrm{S}}}(a_j|\mathcal{G}_i)\right)$$

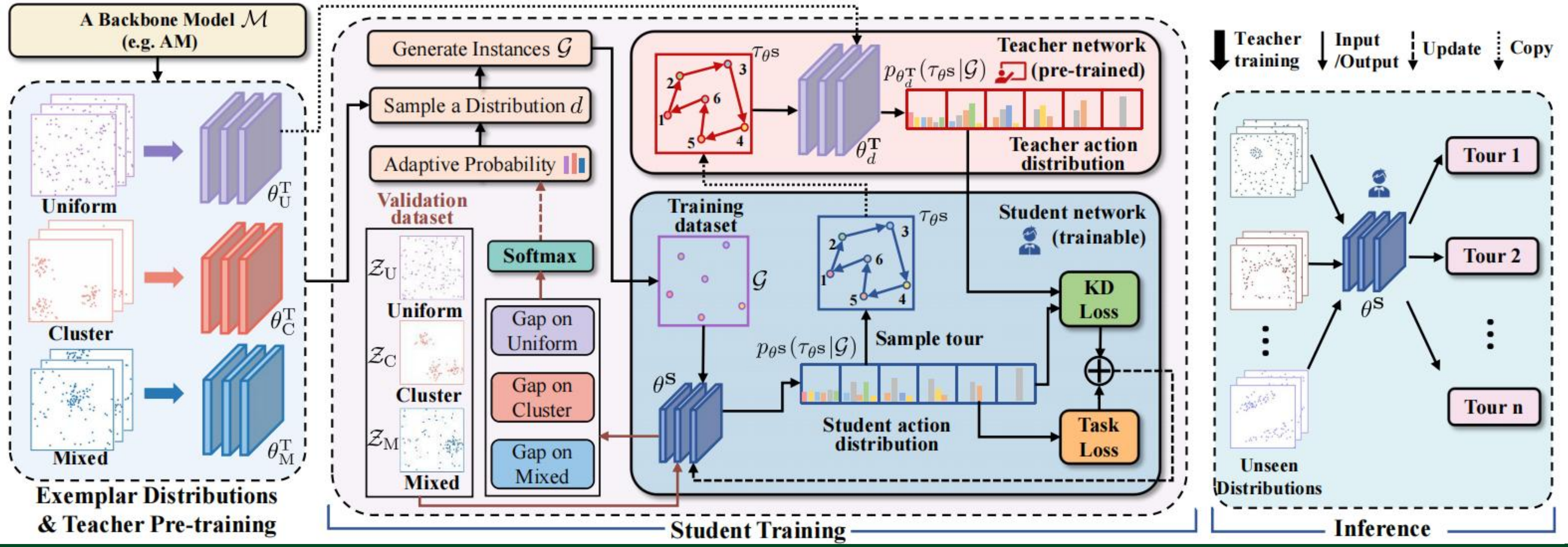


Figure 2: Framework of our AMDKD scheme.

From left to right: teacher pre-training, student training and inference.

The Uniform is selected as the current exemplar distribution for an example.

3. Experimental results

- **Effectiveness analysis of AMDKD**

Table 1: Distillation effectiveness of AMDKD on three exemplar distributions.

	Model	Size (M)	$n=20$				$n=50$				$n=100$			
			G_U	G_C	G_M	Avg.	G_U	G_C	G_M	Avg.	G_U	G_C	G_M	Avg.
TSP	AM(U)	0.68	0.09%	0.26%	0.19%	0.18%	0.59%	2.24%	1.36%	1.39%	2.10%	7.49%	4.06%	4.55%
	AM(C)	0.68	0.17%	0.10%	0.27%	0.18%	1.41%	0.80%	2.14%	1.45%	3.76%	6.97%	4.39%	5.04%
	AM(M)	0.68	0.15%	0.16%	0.13%	0.15%	1.19%	1.71%	0.87%	1.26%	3.08%	5.65%	2.55%	3.76%
	AMDKD-AM	**0.26**	0.02%	0.06%	0.05%	**0.04%**	0.25%	1.64%	0.86%	**0.91%**	1.21%	5.63%	3.55%	**3.46%**
	POMO(U)	1.20	0.00%	0.01%	0.01%	0.01%	0.04%	0.42%	0.21%	0.22%	0.17%	1.97%	0.92%	1.02%
	POMO(C)	1.20	0.00%	0.00%	0.01%	0.00%	0.09%	0.07%	0.21%	0.12%	0.41%	0.29%	0.83%	0.51%
	POMO(M)	1.20	0.00%	0.01%	0.00%	0.00%	0.08%	0.17%	0.08%	0.11%	0.77%	1.17%	0.34%	0.76%
	AMDKD-POMO	**0.49**	0.00%	0.00%	0.00%	**0.00%**	0.05%	0.05%	0.09%	**0.06%**	0.34%	0.35%	0.41%	**0.37%**
CVRP	AM(U)	0.68	1.98%	1.99%	1.98%	1.98%	2.53%	4.33%	2.99%	3.28%	3.10%	9.87%	4.57%	5.85%
	AM(C)	0.68	1.62%	1.43%	1.74%	1.60%	3.08%	2.75%	3.35%	3.06%	4.27%	3.89%	4.93%	4.36%
	AM(M)	0.68	2.09%	2.19%	2.05%	2.11%	2.74%	3.17%	2.31%	2.74%	3.95%	6.26%	3.41%	4.54%
	AMDKD-AM	**0.26**	0.53%	0.59%	0.64%	**0.59%**	1.61%	2.66%	1.92%	**2.07%**	2.08%	5.06%	3.01%	**3.38%**
	POMO(U)	1.20	0.36%	0.49%	0.51%	0.45%	0.80%	1.53%	1.07%	1.13%	0.95%	2.34%	1.31%	1.53%
	POMO(C)	1.20	0.41%	0.40%	0.54%	0.45%	1.16%	0.93%	1.07%	1.05%	0.93%	1.28%	1.21%	1.14%
	POMO(M)	1.20	0.36%	0.51%	0.40%	0.42%	1.22%	1.34%	0.85%	1.14%	1.89%	2.07%	0.96%	1.64%
	AMDKD-POMO	**0.49**	0.35%	0.40%	0.41%	**0.39%**	0.81%	0.97%	0.89%	**0.89%**	1.06%	1.36%	0.99%	**1.13%**

Note: Unless otherwise stated, the gaps are computed w.r.t. the strong traditional solvers Gurobi [5] (for TSP) and LKH [4] (for CVRP).

1. reduce the size of the teacher model from 0.68 to 0.26 M (a 61.8% reduction) for AM and from 1.20 to 0.49 M (a 59.2% reduction) for POMO;
2. improve overall performance for both TSP and CVRP on all the three sizes.

3. Experimental results

- **Generalization analysis of AMDKD**

baselines
- deep models
  - AM
  - POMO
  - DACT
  - LCP
- other methods specialized for generalization
  - HAC
  - DROP
  - GANCO
  - PSRO (a.k.a LIH)

Table 2: Generalization on unseen in-distribution (ID) and out-of-distribution (OoD) instances.

	Model	Size (M)	$n=20$ G_{ID}	$n=20$ G_{OoD}	$n=20$ Time*	$n=50$ G_{ID}	$n=50$ G_{OoD}	$n=50$ Time*	$n=100$ G_{ID}	$n=100$ G_{OoD}	$n=100$ Time*
TSP	Gurobi	-	-	-	0.01s (7s)	-	-	0.08s (51s)	-	-	0.7s (7.6m)
	HAC#	0.68	0.11%	0.07%	0.03s (48s)	1.05%	0.57%	0.09s (4m)	4.68%	2.97%	0.19s (16m)
	LCP#	2.03	0.11%	0.03%	1.2s (43m)	6.70%	0.99%	1.5s (2.8h)	32.70%	8.24%	2.4s (6.4h)
	DACT(T=1,280)#	0.27	0.11%	0.08%	12s (2.1m)	0.31%	0.34%	19s (7.2m)	2.37%	3.13%	28s (23m)
	AM# (128)	0.68	0.18%	0.10%	0.03s (48s)	1.48%	0.74%	0.09s (4m)	4.15%	2.84%	0.19s (16m)
	AM# (64)	0.26	0.21%	0.11%	0.03s (35s)	1.74%	0.83%	0.08s (2.8m)	5.93%	3.85%	0.17s (11m)
	AMDKD-AM (64)	0.26	**0.04%**	**0.02%**	0.03s (35s)	**0.91%**	**0.37%**	0.08s (2.8m)	**3.46%**	**1.87%**	0.17s (11m)
	POMO# (128)	1.20	0.00%	0.00%	0.03s (5s)	0.07%	0.05%	0.09s (16s)	**0.30%**	**0.28%**	0.13s (1.1m)
	POMO# (64)	0.49	0.02%	0.01%	0.02s (4s)	0.18%	0.16%	0.04s (11s)	0.69%	0.59%	0.12s (50s)
	AMDKD-POMO (64)	0.49	**0.00%**	**0.00%**	0.02s (4s)	**0.06%**	**0.05%**	0.04s (11s)	0.37%	0.41%	0.12s (50s)
	AMDKD+EAS†	0.49	**0.00%**	**0.00%**	5s (4.5m)	**0.01%**	**0.01%**	12s (28m)	**0.11%**	**0.10%**	28s (2.3h)
CVRP	LKH3	-	-	-	7.7s (1.3h)	-	-	31s (5.3h)	-	-	56s (9.6h)
	DACT(T=1,280)#	0.27	0.47%	0.42%	28s (4.3m)	3.28%	3.16%	55s (14m)	9.11%	9.38%	1.5m (34m)
	AM# (128)	0.68	2.00%	1.98%	0.05s (1.2m)	3.39%	2.66%	0.12s (5m)	5.42%	4.24%	0.26s (18m)
	AM# (64)	0.26	2.02%	2.02%	0.05s (49s)	3.62%	2.65%	0.11s (3.6m)	6.83%	4.56%	0.23s (13m)
	AMDKD-AM (64)	0.26	**0.59%**	**0.55%**	0.05s (49s)	**2.07%**	**1.69%**	0.11s (3.6m)	**3.38%**	**2.79%**	0.23s (13m)
	POMO# (128)	1.20	0.42%	0.39%	0.05s (7.8s)	0.92%	0.94%	0.10s (18s)	1.14%	1.21%	0.19s (1.3m)
	POMO# (64)	0.49	0.55%	0.51%	0.05s (6.1s)	1.19%	1.21%	0.08s (15s)	1.43%	1.50%	0.18s (1.1m)
	AMDKD-POMO (64)	0.49	**0.39%**	**0.36%**	0.05s (6.1s)	**0.89%**	**0.90%**	0.08s (15s)	**1.13%**	**1.21%**	0.18s (1.1m)
	AMDKD+EAS†	0.49	**-0.04%**	**-0.06%**	9s (7.8m)	**0.07%**	**0.04%**	20s (37m)	**-0.03%**	**-0.04%**	40s (3.3h)

* We report the average time to solve one instance, and the total time to solve 10,000 instances in (·) with batch parallelism allowed (one GPU).
# The corresponding model is trained on a mixed training dataset that contains instances from all the three exemplar distributions.
† For EAS, we adopt its EAS-lay version (T=100) for demonstration purpose.

Table 3: Generalization performance on selected instances ($100 \leq n \leq 200$) from benchmark datasets.

	PSRO	AM (128)	GANCO	HAC	AM#(128)	**AMDKD-AM** (64)	POMO (128)	DROP	POMO#(128)	**AMDKD-POMO** (64)	**AMDKD+EAS**
TSPLIB	4.47%	42.63%	4.87%	6.06%	17.60%	**3.53%**	29.73%	10.79%	**0.87%**	1.08%	**0.74%**
CVRPLIB	-	29.36%	-	-	13.88%	**7.43%**	14.19%	8.67%	6.80%	**4.38%**	**1.26%**

4. Conclusion & Future work

cross-distribution generalization issue → **AMDKD**

- **exhibit competitive performance in generalizing to other unseen out-of-distribution instances**
- **consume less computational resources**
- **model-agnostic; generic for all deep models**

Future work

1) generalizing AMDKD for different/larger problem sizes;

2) considering the improvement models like DACT as the backbone;

3) performing online distillation to jointly and efficiently train the teachers and the student models;

4) assessing the impact of the quality of the validation dataset on the distillation;

5) enhancing the interpretability of AMDKD .

THANKS

Learning Generalizable Models for Vehicle Routing Problems via Knowledge Distillation

Jieyi Bi, Yining Ma, Jiahai Wang, Zhiguang Cao, Jinbiao Chen,Yuan Sun, and Yeow Meng Chee

bijy6@mail2.sysu.edu.cn, yiningma@u.nus.edu
wangjiah@mail.sysu.edu.cn, zhiguangcao@outlook.com
chenjb69@mail2.sysu.edu.cn, yuan.sun@unimelb.edu.au,
ymchee@nus.edu.sg

This work is supported in part by the National Key R&D Program of China (2018AAA0101203), the National Natural Science Foundation of China (62072483), and the Guangdong Basic and Applied Basic Research Foundation (2022A1515011690, 2021A1515012298); in part by the Agency for Science Technology and Research Career Development Fund (C222812027).