Scale-conditioned Adaptation for Large Scale Combinatorial Optimization

Minsu Kim* **Jiwoo Son*** **Hyeonah Kim** **Jinkyoo Park**
Korea Advanced Institute of Science and Technology (KAIST)
Dept. Industrial & Systems Engineering
{min-su, sonleave25,hyeonah_kim, jinkyoo.park}@kaist.ac.kr

Abstract

Deep reinforcement learning (DRL) for combinatorial optimization has drawn attention as an alternative for human-designed solvers. However, training DRL solvers for large-scale tasks remains challenging due to combinatorial optimization problems' NP-hardness. This paper proposes a novel *scale-conditioned adaptation* (SCA) scheme that improves the transferability of the pre-trained solvers on larger-scale tasks. The main idea is to design a scale-conditioned policy by plugging a simple deep neural network, denoted as *scale-conditioned network* (SCN), into the existing DRL model. SCN extracts a hidden vector from a scale value, and then we add it to the representation vector of the pre-trained DRL model. The increment of the representation vector captures the context of scale information and helps the pre-trained model effectively adapt the policy to larger-scale tasks. Our method is verified to improve the zero-shot and few-shot performance of DRL-based solvers in various large-scale combinatorial optimization tasks.

1 Introduction

Combinatorial Optimization (CO) is a research field that deals with various important problems. A representative CO problem is the traveling salesman problem (TSP) [1] which aims to find the shortest path of the Hamiltonian cycle: the salesman must visit every city and get back to the initial city. TSP can extend to several practical problems such as capacitated vehicle routing problems (CVRP) [2]. However, TSP is proven to be *NP-hard* [1] so that it is intractable to find an optimal solution in a practical time budget. To this end, several heuristic methods were suggested to find sub-optimal solutions on a reasonable budget [3, 4]. However, these methods are handcrafted by domain experts and are hard to be extended to a similar class of CO problems.

Related Works. Deep reinforcement learning (DRL) methods [5, 6, 7, 8, 9, 10, 11] are drawing considerable attention to replace handcrafted heuristic methods because they can generate design solvers using the high expression power of deep neural network (DNN), which can be trained without a labeled optimal solution. Remarkably, some studies [8, 10, 11, 12] already proposed that a DRL-based solver, a general purpose method, outperforms problem-specialized handcrafted heuristics. However, DRL-based methods suffer from scalability issues; it has only been verified in small-scale CO problems. To tackle this issue, an Effective Active Search (EAS) [12], a transfer learning method for a DRL-based CO solver, was proposed. However, the EAS was verified on insufficiently larger scales $N = 125, 150, 200$ than the previous method $N = 100$, where N is the number of cities to visit in TSP and CVRP.

Contribution. This paper proposes a *scale-conditioned adaptation* (SCA), a fast adaptation scheme combining the EAS with a novel *scale-conditioned network* (SCN). The SCN reduces the number of

*Equal Contribution.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

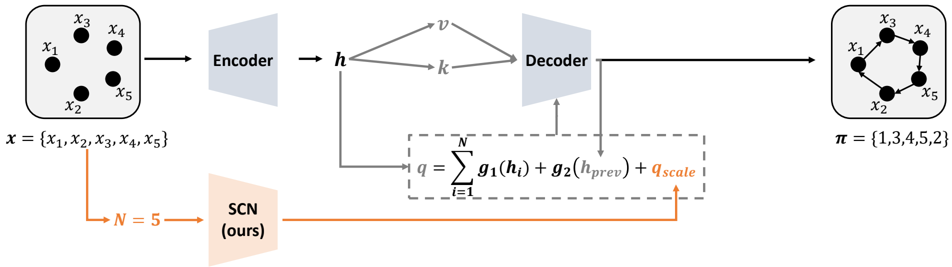


Figure 1: The scale conditioned network combined with DRL model for combinatorial optimization.

transfer iterations (K) of the EAS algorithm and maximize performance with a limited K by tackling *distributional shift* according to scale variations. Specifically, the SCN is a simple DNN model that effectively modifies the hidden representation by adapting *distributional shift* to the original hidden representation for the larger-scale tasks. Technically, the SCN takes the scale value input N and outputs an increment vector for the pre-trained DRL model's hidden representation vector. With the implementation of SCN to a pre-trained DRL model, the overall policy becomes a scale-conditioned policy that can effectively adapt to larger-scale tasks using the EAS. According to the experimental results, our SCA consistently improved the transfer-ability of two representative DRL models (POMO and Sym-NCO) in large-scale ($N = 500, 1000$) CO tasks (TSP and CVRP).

2 Preliminary

2.1 Problem Description with Target DRL models

The Policy Optimization for Multiple Optima (POMO) [9] is a DRL method that trains the attention model (AM) [7], which is a transformer-based encoder-decoder [13] model. The POMO-trained AM using the REINFORCE [14] with their novel shared baseline scheme leverages the TSP's symmetric nature. The Symmetric Neural Combinatorial Optimization (Sym-NCO) [11] is an expansion of the POMO scheme for the general purpose symmetricity learning that achieved higher performance on various TSP variants including CVRP.

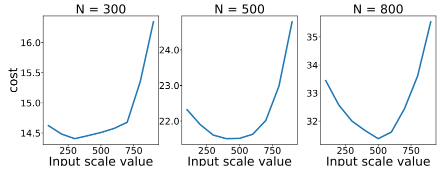
Both POMO and Sym-NCO have a similar structure to generate an instance-conditioned policy $p(\boldsymbol{\pi}|\boldsymbol{x})$ (see Fig. 1 for encoder-decoder processing of policy). They encode a N-scaled instance $\boldsymbol{x} = \{x_i\}_{i=1}^{N}$, which contains 2D euclidean coordinates, into high dimensional hidden vector $\boldsymbol{h} = \{\boldsymbol{h}_i\}_{i=1}^{N}$. Then, the decoder auto-regressively generates permutation index $\boldsymbol{\pi} = \{\pi_i\}_{i=1}^{N}$ of input indices (i.e., $\pi_i \in \{1, ..., N\}$) exploiting $\boldsymbol{h}$. The permutation index $\boldsymbol{\pi}$ becomes the order of visiting the cities in $\boldsymbol{x}$. The decoder architecture has a multi-head attention (MHA) layer, which processes three different vectors from $\boldsymbol{h}$: query $\boldsymbol{q}$, key $\boldsymbol{k}$, value $\boldsymbol{v}$ similar to the transformer. The query $\boldsymbol{q}$ is carefully designed to capture the contextual information of CO instances: $q = g_1(\frac{1}{N}\sum_{i=1}^{N} h_i) + g_2(h_{prev})$ where g_1 and g_2 are linear-projections. The $\frac{1}{N}\sum_{i=1}^{N} h_i$ is designed to capture the global feature of instances. The h_{prev} is a hidden vector of a previously selected city to facilitate the auto-regressive process of the decoder. See [7] for detailed process. Note that both POMO and Sym-NCO are trained on a fixed scale of $N = 100$.

2.2 Effective Active Search: Transfer Learning of Pre-trained DRL model

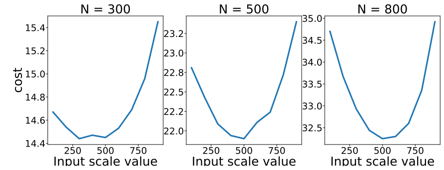
The Effective Active Search (EAS) [12] is a transfer learning method that was validated to improve the POMO's performance on larger scale tasks, $N = 125, 150, 200$. EAS has three different variations; the EAS-lay gives the most powerful performance. EAS-lay adds one multi-layer perceptron (MLP) to process the hidden vector of $\boldsymbol{h}$. The MLP is trained to adapt to larger-scale tasks, whereas other the pre-trained layers are not updated during the adaptation. See [12] for detailed process of EAS.

3 Scale-Conditioned Adaptation

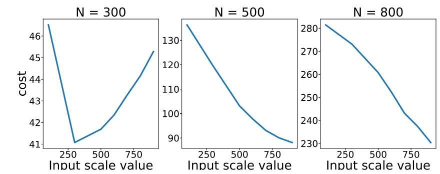
We propose a scale-conditioned adaptation (SCA) method by combining EAS with a novel scale-conditioned network (SCN). The SCN is a simple MLP model $f_\theta(N)$ where the input is N and output



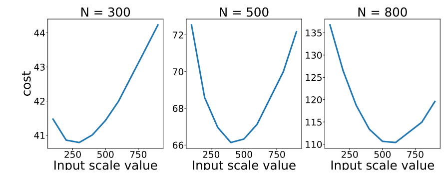
(a) Performance POMO to various scales for TSP



(b) Performance Sym-NCO to various scales for TSP



(c) Performance POMO to various scales for CVRP



(d) Performance Sym-NCO to various scales for CVRP

Figure 2: The zero-shot performance of POMO and Sym-NCO for various scales of TSP and CVRP.

is an increment vector for $\boldsymbol{q}$ (see Fig. 1 for SCA process). The $f_\theta(N)$ can be pre-trained so that can effectively improve the transferability of the DRL model within a limited adaptation time.

The $f_\theta(N)$ outputs q_{scale}, which is an increment of query vector of the pre-trained DRL model as follows:

$$q_{new} = q + q_{scale} = q + f_\theta(N) \tag{1}$$

This simple process can give the scale conditioning to the original policy as: $p(\boldsymbol{\pi}|\boldsymbol{x}, N)$, which can effectively adapt to new N by simply inserting a scale value N during the adaptation phase.

3.1 Pre-training SCN (see Appendix A.2 for detail)

The SCN is trained to capture the contextual features of scale N by training with variations of N, and to infer a proper increment of the query as q_{scale} (which does not need additional training in the adaptation phase). To this end, we train the SCN that is plugged into the pre-trained DRL model, using the EAS algorithm (only with $K = 0$).

3.2 Adaptation Phase (see Appendix A.3 for detail)

During the training phase, we just leverage three task scales $N = 125, 150, 200$. Then, in the adaptation phase, we input much larger scales to the SCA: $N = 500, 1000$. We expect that the scale context can be captured by SCA with variations of small-scaled data. This extrapolation process can support scaling in several practical CO applications because some CO application needs enormous computing cost to evaluate reward in the large-scale task.

Integration of SCN and EAS. We implemented the EAS-lay as the main adaptation scheme for large-scale tasks. In the adaptation phase, our SCA is not updated but only infers q_{scale} from N to support the EAS-lay process. Note that our SCA can improve both the zero-shot transfer-ability of the DRL model (without EAS-lay) and the few-shot transfer-ability of EAS-lay.

4 Experimental Results

4.1 SCA for large scale combinatorial optimization

To verify the effectiveness of SCA, we conducted experiments by adding a neural network that gives the scale information to the pre-trained models, POMO and Sym-NCO, for routing problems (TSP and CVRP). The routing problems aim to minimize the total cost of visiting all customers. Additionally, CVRP [2] considers multi-vehicles with capacity constraints, whereas TSP considers the single-vehicle setting. The size of routing problems is defined by the number of customers N.

Setup. We use the pre-trained POMO and Sym-NCO on $N = 100$ for both TSP and CVRP. We freeze the pre-trained models' parameters and train scale-conditioned network $f_\theta(N)$ to give additional scale information to the original models. We randomly generated $1,000$ TSP and CVRP instances for each size $N = 125, 150, 200$ with the same instance generation rule as Kool et al.[7] to train $f_\theta(N)$. Using the same rule, 100 instances of TSP and CVRP with $N = 500, 1000$ were generated for evaluation.

Table 1: Performance evaluation on K shot adaptation to large-scale CVRP.

	CVRP ($N = 500$)				CVRP ($N = 1,000$)			
	$K = 0$	$K = 1$	$K = 5$	$K = 10$	$K = 0$	$K = 1$	$K = 5$	$K = 10$
POMO	150.93	131.39	81.57	74.83	371.23	315.68	182.70	165.55
POMO + SCA (ours)	**111.17**	**95.52**	**76.33**	**72.33**	**259.08**	**221.25**	**179.99**	**163.39**
Sym-NCO	79.61	72.44	71.54	69.07	202.59	162.51	146.01	137.85
Sym-NCO + SCA (ours)	**68.76**	**68.65**	**68.23**	**67.87**	**154.94**	**144.68**	**140.50**	**135.69**

Table 2: Performance evaluation on K shot adaptation to large scale TSP.

	TSP ($N = 500$)				TSP ($N = 1,000$)			
	$K = 0$	$K = 1$	$K = 5$	$K = 10$	$K = 0$	$K = 1$	$K = 5$	$K = 10$
POMO	22.85	22.10	**21.19**	**21.12**	42.54	39.94	**37.04**	**36.93**
POMO + SCA (ours)	**21.55**	**21.50**	21.34	21.23	**38.22**	**37.96**	37.37	36.96
Sym-NCO	23.33	22.40	**21.19**	**21.17**	43.97	40.38	**37.06**	**37.06**
Sym-NCO + SCA (ours)	**21.80**	**21.75**	21.64	21.76	**38.58**	**38.33**	37.49	37.18

Ablation Study. We experimentally demonstrate the effectiveness of the additional information extracted from scale values via SCN $f_\theta(N)$ by evaluating models with different input values in $[100, 900]$. We evaluate zero-shot performances using 10 instances of TSP and CVRP with various sizes ($N = 300, 500, 800$). As shown in Fig. 2, the average cost tends to increase when the input scale values are not aligned with N. In Fig. 2a, Fig. 2b, and Fig. 2d, the average costs are minimized when the input scale values are approximately at 500 for $N = 800$. We conjecture that SCN suffers from extrapolation since we trained $f_\theta(N)$ with $N \leq 200$ and tested in $N = 800$. This shows that $f_\theta(N)$ captures meaningful contextual features of N, which make the pre-trained models adapt well to different Ns.

Performance Evaluation. We measure the performance of zero-shot and few-shot to demonstrate whether SCA improves the performance of the original model. The performance is calculated as the average cost. We employ EAS-lay described in Section 2.2 to implement the few-shot adaptation. We used $K = 0, 1, 5, 10$, where K refers to the number of transfer iterations (see [12] in details).

As shown in Table 1 and Table 2, SCA successfully improves the performance when $K \in \{0, 1\}$ for TSP task and $K \in \{0, 1, 5, 10\}$ for CVRP task. We observe that SCA is more effective in the early adaptation phase (i.e., K is small) because we conduct EAS with $K = 0$ during the SCA training phase (see Appendix C for additional analysis). Therefore, in the current phase, the SCA can be positioned as an effective adaptation scheme for zero-shot ($K = 0$) and few-shot ($K \leq 10$) adaptation but has limitations on large-shot adaptation ($K > 100$).

5 Future Direction

In this paper, we proposed a new strategy, the *scale-conditioned adaptation* (SCA) for solving large-scale routing problems which are hard to address due to their combinatorial nature. The SCA was effective few shot adaptation (small K) but had limitations on large shot adaptation (large K). To resolve this limitation, we suggest the below strategy as a future direction:

1. Expands the scale-conditioned network (SCN) to have input N and K together: i.e. expands $f_\theta(N)$ as $f_\theta(N, K)$.
2. Train $f_\theta(N, K)$ with variation of N and variation of K (using EAS).
3. Pre-trained $f_\theta(N, K)$ infers q_{scale} adaptively not only with N but also with K.

References

[1] Christos H. Papadimitriou. The euclidean travelling salesman problem is np-complete. *Theoretical Computer Science*, 4(3):237 – 244, 1977.

[2] Stefan Irnich, Paolo Toth, and Daniele Vigo. *Chapter 1: The Family of Vehicle Routing Problems*, pages 1–33. SIAM, 2002.

[3] Vašek Chvátal David Applegate, Robert Bixby and William Cook. Concorde tsp solver.

[4] Keld Helsgaun. An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde University*, 12 2017.

[5] Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning, 2017.

[6] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 6348–6358. Curran Associates, Inc., 2017.

[7] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019.

[8] Sungsoo Ahn, Younggyo Seo, and Jinwoo Shin. Learning what to defer for maximum independent sets. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 134–144. PMLR, 13–18 Jul 2020.

[9] Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. Pomo: Policy optimization with multiple optima for reinforcement learning. *Advances in Neural Information Processing Systems*, 33:21188–21198, 2020.

[10] Yeong-Dae Kwon, Jinho Choo, Iljoo Yoon, Minah Park, Duwon Park, and Youngjune Gwon. Matrix encoding networks for neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 34:5138–5149, 2021.

[11] Minsu Kim, Junyoung Park, and Jinkyoo Park. Sym-nco: Leveraging symmetricity for neural combinatorial optimization. *arXiv preprint arXiv:2205.13209*, 2022.

[12] André Hottung, Yeong-Dae Kwon, and Kevin Tierney. Efficient active search for combinatorial optimization problems. *arXiv preprint arXiv:2106.05126*, 2021.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc., 2017.

[14] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

A Detail of SCA Process

SCA has three phases (1) pre-training the model, which is plugged with scale conditioned network, (2) training the scale conditional network (3) combining with SCA and EAS strategy. We provide the details of this procedure in this section.

A.1 Pre-training the model.

We use the models which are POMO[9] and Sym-NCO[11]. These models are trained on instances with $N = 100$, where N is the number of cities to visit, made available by the POMO authors.

A.2 Training scale conditional network.

The next phase is training the scale conditioned network f_θ. The purpose of this phase is to train scale conditioned network to capture the context information from the scale. Scale conditioned network consists of two layers of MLP with ReLU activation function. This network's input N, the scale of corresponding problem instances, goes into the network; then, the output with the context of scale comes out. As we describe in Section 2.1 POMO and Sym-NCO have encoder-decoder structures. In the model's decoder, there is a query which is one of the components of decoder architecture and contains contextual information of instance. While we train the scale conditioned network, we freeze the model's parameter and only update two layers of MLP, which are composed of scale conditioned network.(see Fig. 3) We use various scale instances for training f_θ and get q_{scale} which adds to model's query q to get new query q_{new} from each scale of instance.

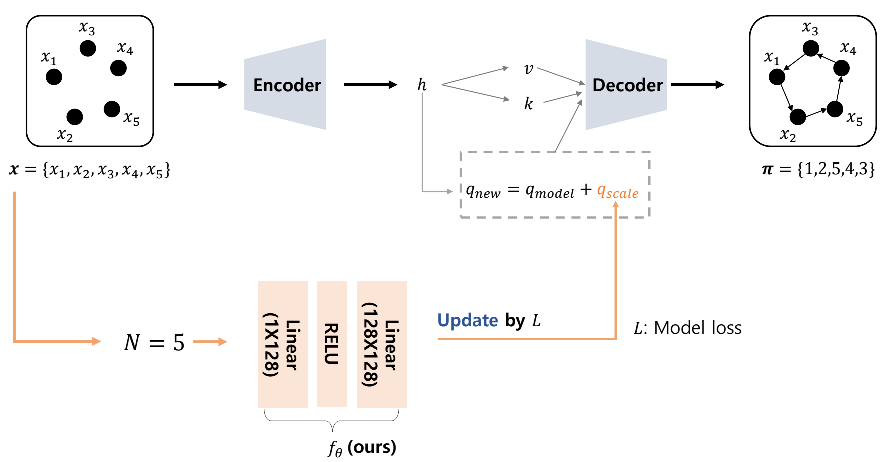


Figure 3: The procedure of training scale conditioned network.

A.3 Combining with SCA and EAS-lay

After training f_θ, we solve the target problem K (number of adaptation) times by employing EAS-lay method. For detail, the trained scale conditioned network is plugged with the model and adjust the model's query $\boldsymbol{q}$ to $\boldsymbol{q}_{new}$ by adding $\boldsymbol{q}_{scale}$, which is extracted from the target instance's scale N. From the $\boldsymbol{q}_{new}$, model infers the solution $\boldsymbol{\pi}$, which is the order of visiting the city and MLP provided by EAS-lay is updated K times by utilizing the EAS-lay method. Note that the model and scale conditioned network are not updated during the EAS-lay. (see Fig. 4)

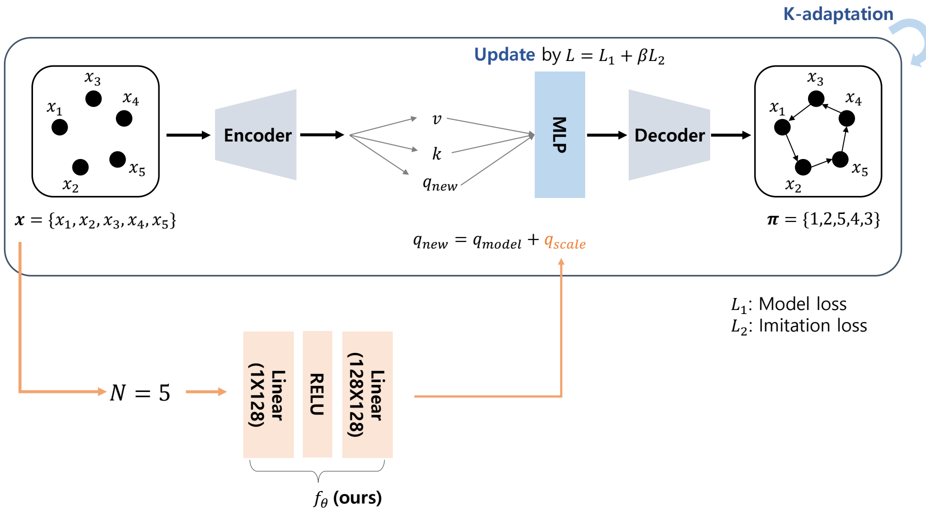


Figure 4: The procedure of adaptation combine with EAS and SCA.

B Implementation of Details of Proposed Method

B.1 Hyperparameter for training scale conditioned network

We set the same hyperparameters to train the scale conditioned model which is plugged to POMO and Sym-NCO

	TSP	CVRP
Batch size	16	16
Learning rate	3.2e-4	4.2e-4
Weight decay	1e-6	1e-6
Epochs	1	1
Epoch size	1000	1000

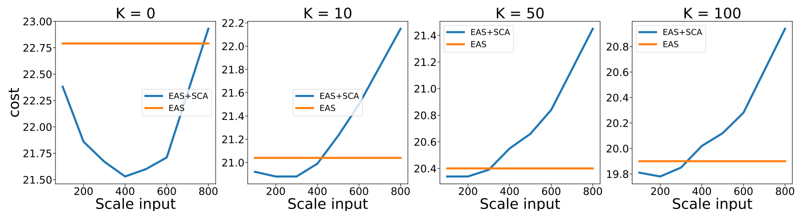
Table 3: Hyperparameter setting of training scale conditioned network.

B.2 Hyperparameter for SCA with EAS-lay

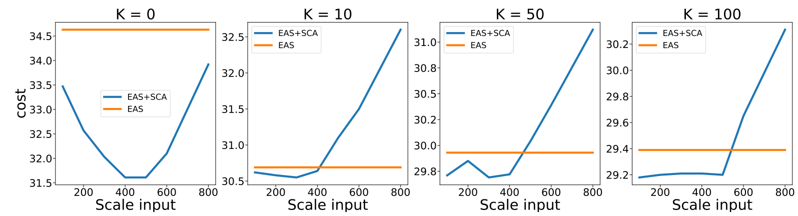
We set the same hyperparameters to employ EAS-lay with SCA

	TSP	CVRP
Learning rate	3.2e-4	4.2e-4
Weight decay	1e-6	1e-6
Imiation rate	1.2e-3	1.3e-3

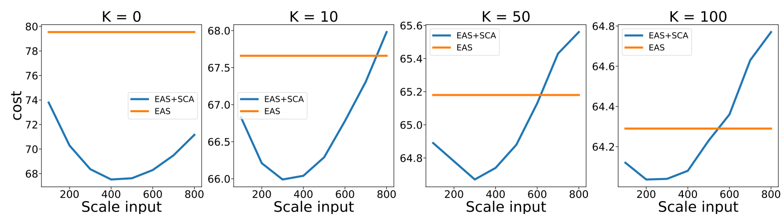
Table 4: Hyperparameter setting of utilizing EAS-lay.



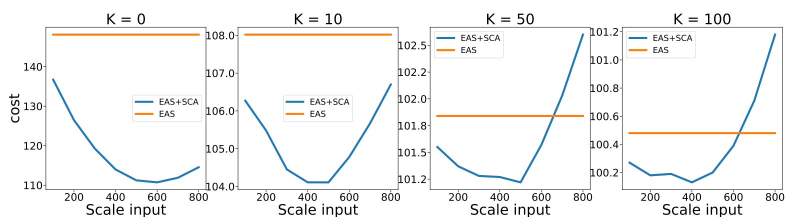
(a) Performance of POMO to various K for TSP 500



(b) Performance POMO to various K for TSP 800



(c) Performance of Sym-NCO to various K for CVRP 500



(d) Performance Sym-NCO to various K for CVRP 800

Figure 5: The few-shot performance for various K of TSP and CVRP.

C Analysis on K-shot Adaptation

We compare K-shot adaptation performances of TSP and CVRP for various K to analyze the effects of SCN in few-shot adaptation. Experiments are conducted on TSP and CVRP with $N = 500, 800$. The parameters of SCN $f_\theta(N)$ trained in Section 4 are used without additional training. Fig. 5 illustrates that SCN with any input scale value outperforms EAS without SCN in zero-shot ($K = 0$). When K increases, SCN gives higher average costs than EAS for large input scale values in both $N = 500$ and 800, but it still gives lower costs with input scale values less than 600. It is noticeable that the cost-minimizing input scale values are shifted to the left as K increases, which means SCN achieves better performances with mismatched input scale values. Thus, we conjecture that SCN is less effective when the number of adaptations increases since SCN is trained for zero-shot adaptation. However, SCN has the potential to adapt K by extending $f_\theta(N)$ as $f_\theta(N, K)$: i.e., conditioning both N and K for shot-adaptive adaptation.